IBM i

バージョン 7.4

データベース
Db2 for i SQL 解説書

IBM i

バージョン **7.4**

データベース
Db2 for i SQL 解説書

注記

本書および本書で紹介する製品をご使用になる前に、 2209 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM i 7.4 (製品番号 5770-SS1) に適用されます。また、改訂版で断りがない限り、それ以降のすべてのリリースおよびモディフィケーションに適用されます。このバージョンは、すべての RISC モデルで稼働するとは限りません。また CISC モデルでは稼働しません。

本書にはライセンス内部コードについての参照が含まれている場合があります。ライセンス内部コードは機械コードであり、IBM 機械コードのご使用条件の条項に基づいて使用権を許諾するものです。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： IBM i
Version 7.4
Database
DB2 for i SQL Reference

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

目次

SQL 解説書について

本書では、Db2® for IBM® i でサポートされている構造化照会言語 (SQL) の内容を明示しています。本書には、システムの管理、データベースの管理、アプリケーション・プログラミング、および操作のタスクに関する参照情報が記載されています。また、システムで使用する SQL ステートメントそれぞれについて、その構文、使用上の注意、キーワード、および例を示しています。

標準への準拠

Db2 for i 7.3 は、以下の IBM および業界の SQL 標準に準拠しています。

分散リレーショナル・データベース体系

- Open Group Publications: DRDA V5 Vol. 1: Distributed Relational Database Architecture(DRDA) (https://www2.opengroup.org/ogsys/catalog/C112)
- Open Group Publications: DRDA V5 Vol. 2: Formatted Data Object Content Architecture(FD:OCA) (https://www2.opengroup.org/ogsys/catalog/C113)
- Open Group Publications: DRDA V5 Vol. 3: Distributed Data Management Architecture (DDM) (https://www2.opengroup.org/ogsys/catalog/C114)

Character Data Representation Architecture

- Character Data Representation Architecture Reference and Registry (http://www-01.ibm.com/software/globalization/cdra/)

Unicode 標準

- Unicode 標準 (http://www.unicode.org)

SQL 標準

Db2 for i 7.3 は、 SQL に関する以下の業界標準に準拠しています。

- *ISO/IEC 9075-1:2016, Information technology - Database languages - SQL - Part 1: Framework (SQL/Framework)*
- *ISO/IEC 9075-2:2016, Information technology - Database languages - SQL - Part 2: Foundation (SQL/Foundation)*
- *ISO/IEC 9075-3:2016, Information technology - Database languages - SQL - Part 3: Call-Level Interface (SQL/CLI)*
- *ISO/IEC 9075-4:2016, Information technology - Database languages - SQL - Part 4: Persistent Stored Modules (SQL/PSM)*
- *ISO/IEC 9075-10:2016, Information technology - Database languages - SQL - Part 10: Object Language Bindings (SQL/OLB)*
- *ISO/IEC 9075-11:2016, Information technology - Database languages - SQL - Part 11: Information and Definition Schemas (SQL/Schemata)*
- *ISO/IEC 9075-14:2016, Information technology - Database languages - SQL - Part 14: XML-Related Specifications (SQL/XML)*

Db2 for i 7.3 は、 SQL に関する以下の業界テクニカル・レポートに準拠しています。

- *ISO/IEC TR 19075-6:2016, Information technology - Database languages - SQL Technical Reports - Part 6: SQL support for JavaScript Object Notation (JSON)*

標準を厳守するために、標準オプションを使用するようにしてください。標準オプションは、以下のインターフェースを使用して指定できます。

表 *1.* 標準オプション・インターフェース

SQL インターフェース	指定
組み込み SQL	SQL プログラム作成 (CRTSQLxxx) コマンドの SQLCURRULE(*STD) パラメーター。 SET OPTION ステートメントを使用して、SQLCURRULE 値を設定することもできます。 (CRTSQLxxx コマンドについて詳しくは、「組み込み SQL プログラミング」を参照してください。)
SQL ステートメント実行	SQL ステートメント実行 (RUNSQLSTM) コマンドの SQLCURRULE(*STD) パラメーター。 (RUNSQLSTM コマンドの詳細については、「SQL プログラミング」を参照。)
サーバー上の呼び出しレベル・インターフェース (CLI)	SQL_ATTR_HEX_LITERALS 接続属性 (CLI について詳しくは、「SQL 呼び出しレベル・インターフェース (ODBC)」を参照してください。)
IBM IBM Developer Kit for Java™ を使用したサーバーの JDBC または SQLJ	16 進数変換接続プロパティー・オブジェクト (JDBC および SQLJ について詳しくは、「IBM Developer Kit for Java」を参照してください。)
IBM i Access Family ODBC ドライバーを使用したクライアントの ODBC	ODBC セットアップにおける 16 進パーサー・オプション (ODBC の詳細については、「IBM i Access」を参照。)
IBM i Access Family OLE DB Provider を使用したクライアントの OLE DB	16 進パーサー・オプション接続オブジェクト・プロパティー (OLE DB の詳細については、「IBM i Access」を参照。)
IBM i Access FamilyADO .NET プロバイダーを使用しているクライアントの ADO .NET	接続オブジェクト・プロパティーの HexParserOption。 (ADO .NET の詳細については、「IBM i Access」を参照。)
IBM Toolbox for Java を使用したクライアントの JDBC	JDBC セットアップ内で SQL 16 進定数をバイナリー・データとして解釈 (JDBC の詳細については、「IBM i Access」を参照。) (IBM Toolbox for Java について詳しくは、「IBM Toolbox for Java」を参照してください。)

SQL ステートメントの例に関する前提事項

本書に示す SQL ステートメントの例では、以下の事項を前提としています。

- SQL のキーワードは、太字で示されています。
- 例で使用されている表名は、「SQL プログラミング」トピック集に示されているサンプル表です。この付録に示されていない表名は、ユーザーが作成するスキーマを作成する必要があります。ユーザー独自のスキーマで次の SQL ステートメントを発行することにより、1 組のサンプル表を作成できます。

  ```
  CALL QSYS.CREATE_SQL_SAMPLE ('ユーザー作成のスキーマ名')
  ```

- SQL の命名規則を使用しています。
- COBOL の例では、(COBOL ではデフォルトではありませんが)、プリコンパイラー・オプションの APOST と APOSTSQL を前提としています。 SQL およびホスト言語のステートメント内の文字ストリング定数は、アポストロフィ (') で区切られています。
- 照合順序 *HEX を使用しています。

これらの前提事項と異なる例では、必ずその旨を明記しています。

構文図の見方

本書で使用される構文図には、以下の規則が適用されます。

- 構文図は、直線で示される経路にしたがって、左から右、上から下の方向に読んでください。

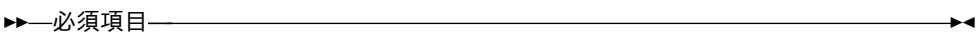
  記号 ▸▸── は、構文図の始まりを示します。

  記号 ──▸ は、構文が次の行に続くことを示します。

  記号 ▸── は、構文が前の行から続いていることを示します。

  記号 ──▸◂ は、構文図の終わりを示します。

  構文単位を示す図は、記号 ├── で始まり、記号 ──┤ で終わります。
- 必須項目は、次のように水平方向の線 (メインパス) 上に示します。



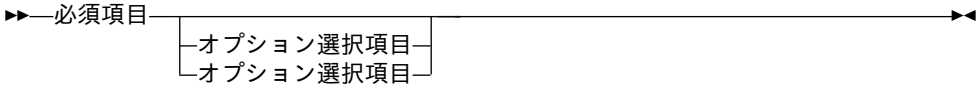
- 任意指定項目は、次のようにメインパスの下に示します。

  メインパスより上に示される項目は、単に読みやすさのために使用されるオプション項目であり、ステートメントの実行には影響を与えません。

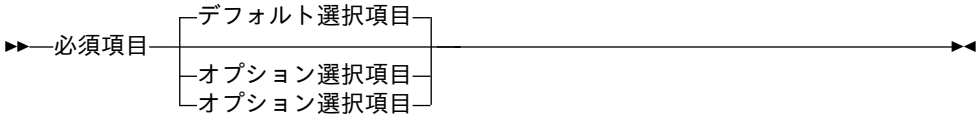
- 複数の項目からユーザーが選択できる場合は、それらの項目を縦方向に並べて示します。

  項目の中から必ずどれか 1 つを選択しなければならない場合は、縦方向に並んでいる選択項目のうちの 1 つをメインパス上に示します。

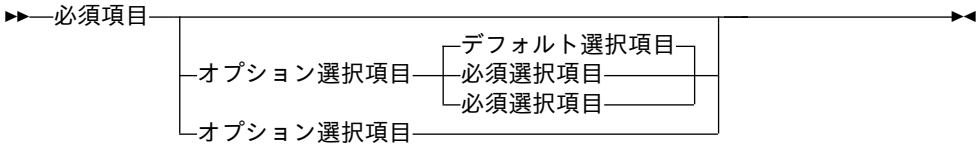
  項目を選択しても選択しなくてもよい場合は、縦方向に並んでいる選択項目をすべてメインパスの下に示します。



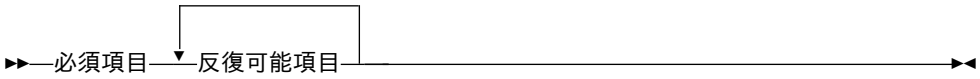
  項目の中にデフォルトの選択項目がある場合は、その選択項目をメインパスの上に示し、残りの選択項目をメインパスの下に示します。



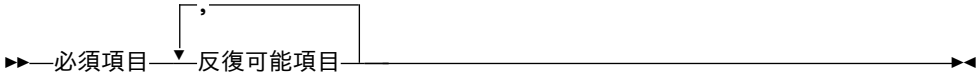
  指定されていないオプション項目にデフォルトがある場合、デフォルトはメインパスの上に示します。



- メインパスの上を通って左に戻る矢印は、反復可能な項目を示します。

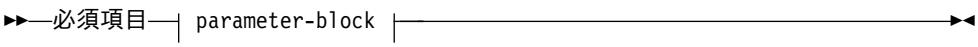


  反復可能を示す矢印にコンマが入っている場合は、反復可能な項目を指定するときに、項目相互間をコンマで区切る必要があります。



  項目群の上にある反復可能を示す矢印は、その項目群にある項目を反復して指定できることを示します。

- キーワードは大文字で示されます (例: FROM)。キーワードは、構文図に示されているとおりのつづりで正確に入力する必要があります。変数は小文字で現れます (例えば、列名)。これらはユーザーが指定する名前または値を表しています。
- 構文図に句読記号、括弧、算術演算子、またはその他の記号が示されている場合には、それらを構文の一部として入力しなければなりません。
- 構文図には、優先キーワードまたは標準キーワードのみが含まれています。標準キーワードに加えて、標準外の同義語もサポートされている場合、そのような標準外同義語の説明は、構文図の中でなく、注のセクションで扱っています。最大の移植性を確保するためには、優先キーワードまたは標準キーワードを使用してください。
- 1 つの変数が構文のより大きなフラグメントを表すこともあります。例えば以下の図で、変数 parameter-block は **parameter-block** とラベル付けされた構文フラグメントを表しています。



parameter-block:

本書の規則

このセクションでは、本書全体で使用されるいくつかの規則を示します。

強調表示の規則

太字	例の中で使用される SQL キーワードを、そのキーワードに関連した説明を導入するときに示します。
イタリック	以下のいずれかを示します。 • 構文図の項目を表す変数。 • 新しい用語の紹介。 • 別の情報源の参照。

混合データの値の記述規則

混合データの値を例の中で示す場合は、以下のような規則が適用されています。

規則	意味
$^{S}_{O}$	EBCDIC シフトアウト制御文字 (X'0E') を表す
$^{S}_{I}$	EBCDIC シフトイン制御文字 (X'0E') を表す
SBCS ストリング	ゼロ以上の 1 バイト文字のストリングを表す
DBCS ストリング	ゼロ以上の 2 バイト文字のストリングを表す
'	DBCS アポストロフィ (EBCDIC X'427D') を表す
G	DBCS G (EBCDIC X'42C7') を表す

混合データの値の記述規則

混合データの値の記述規則は以下のとおりです。X' 0E' によって表されるシフトアウト文字、 X' 0F' によって表されるシフトイン文字、 sbcs-string によって表される 1 バイト文字、 dbcs-string によって表される 2 バイト文字、 EBCDIC X' 427D によって表される DBCS のアポストロフィ、および EBCDIC X' 42C7 によって表される DBCS の G。

Unicode データを記述するための規則

特定の Unicode UTF-16 コード・ポイントを参照する場合は、U+*n* という形式でコード・ポイントを記述できます (*n* は、4 桁から 6 桁の 16 進数字です)。コード・ポイントの 16 進数字の桁数が 4 桁より少ない場合を除き、先行ゼロは省略されます。例えば、以下の値は、UTF-16 コード・ポイントの有効な表記です。

```
U+00001  U+0012  U+0123  U+1234  U+12345  U+123456
```

SQL アクセシビリティー

IBM では、身体に障害のある方々にとってアクセスしやすいインターフェースおよびドキュメンテーションを提供することをコミットしています。

IBM のアクセシビリティー・サポートに関する一般情報については、

http://www.ibm.com/able  の Accessibility Center を参照してください。

SQL アクセシビリティー・サポートは、次の 2 つの主要なカテゴリーに分かれています。

- System i® ナビゲーターは、IBM i オペレーティング・システムと Db2 for i のグラフィカル・ユーザー・インターフェースです。 Windows のグラフィカル・ユーザー・インターフェースでサポートされるアクセシビリティー機能については、Windows のヘルプの索引から「アクセシビリティー」を参照してください。
- オンライン・ドキュメンテーション、オンライン・ヘルプ、およびプロンプト形式の SQL インターフェースには、Windows リーダー・プログラム (IBM ホー

ムページ・リーダーなど) を使用してアクセスすることができます。 IBM ホームページ・リーダーおよびその他のツールについて詳しくは、Accessibility Center を参照してください。

IBM ホームページ・リーダーを使用すると、本書に収めてあるすべての本文、SQL Information Center に収めたすべての記事、およびすべての SQL メッセージにアクセスすることができます。ただし、SQL の構文図は、性質が複雑なため、リーダーではスキップされます。使い勝手に応じて選択できる方法が、このほかにも 2 つあります。

- 対話式 SQL および Query Manager

  対話式 SQL および Query Manager は、SQL ステートメントに関するプロンプトを提供する従来型のファイル・インターフェースです。これらの機能は、IBM Db2 Query Manager and SQL Development Kit for i に組み込まれているものです。対話式 SQL と Query Manager の詳細については、『SQLプログラミング』と「Query マネージャーの使用」 を参照してください。
- SQL 支援

  SQL 支援は、SQL ステートメントへの指示インターフェースを提供するグラフィカル・ユーザー・インターフェースです。これは System i ナビゲーターの中の一機能です。 詳しくは、System i ナビゲーターのオンライン・ヘルプおよび Information Center を参照してください。

「SQL 解説書」の PDF ファイル

これを使用して、この情報の PDF を表示および印刷します。

本書の PDF 版を表示またはダウンロードするには、「SQL 解説書」を選択します。

PDF ファイルの保存

表示または印刷のために PDF をワークステーションに保存するには、以下のようにします。

1. ブラウザーで PDF リンクを右クリックする。
2. PDF をローカルに保存するオプションをクリックする。
3. PDF を保存したいディレクトリーに進む。
4. 「保存」をクリックする。

Adobe Reader のダウンロード

これらの PDF を表示または印刷するには、Adobe Reader がご使用のシステムにインストールされている必要があります。このアプリケーションは、 Adobe Web サイト (http://get.adobe.com/reader/) から無償でダウンロードできます。

IBM i 7.4 の新機能

このトピックでは、IBM i 7.4 のトピック集に加えられた変更点を中心に説明します。

本書で説明している主要な新機能には、以下のものが含まれます。

- MD5、SHA1、SHA-256、および SHA-512 アルゴリズムを使用した新しい HASH 関数: 480 ページの『HASH_MD5、HASH_SHA1、 HASH_SHA256、および HASH_SHA512』。以前 HASH という名前であった関数が、HASH_VALUES に名前変更されました: 482 ページの『HASH_VALUES』
- Db2 ミラーの新しいトリガー属性: 1302 ページの『CREATE TRIGGER』, 1004 ページの『ALTER TRIGGER』
- デフォルト値のみを含む行の INSERT: 1556 ページの『INSERT』
- SET *transition-variable* ステートメントが *assignment-statement* に併合されました: 1782 ページの『割り当てステートメント』

さらに、以下の機能拡張は、初期リリース 7.3 以降の新機能です。

- 新しい OLAP 関数: PERCENT_RANK: 224 ページの『OLAP 指定』
- IS JSON および JSON_EXISTS 述部: 256 ページの『IS JSON 述部』 および 258 ページの『JSON_EXISTS 述部』
- LISTAGG 集約関数: 327 ページの『LISTAGG』
- DECFLOAT_FORMAT スカラー関数: 425 ページの『DECFLOAT_FORMAT』
- JSON パブリッシングおよびスカラー関数: 318 ページの『JSON_ARRAYAGG』、 322 ページの『JSON_OBJECTAGG』、 498 ページの『JSON_ARRAY』、 502 ページの『JSON_OBJECT』、 506 ページの『JSON_QUERY』、および 512 ページの『JSON_VALUE』
- EXTRACT 関数のオプションが追加されました: 460 ページの『EXTRACT』
- LTRIM および RTRIM 関数が 2 番目の引数をサポートします: 538 ページの『LTRIM』 および 618 ページの『RTRIM』
- NOW 関数でオプションの精度を使用できます: 568 ページの『NOW』
- REPLACE 関数の 3 番目の引数はオプションです: 602 ページの『REPLACE』
- VARCHAR_FORMAT が、マイクロ秒とミリ秒の 2 つの新しいタイム・スタンプ形式ストリング・エレメントをサポートします: 678 ページの『VARCHAR_FORMAT』
- JSON_TABLE 表関数: 744 ページの『JSON_TABLE』
- LIMIT と OFFSET の制限が除去されました: 837 ページの『fetch-clause』 および 836 ページの『offset-clause』
- ネストされた表の式、表関数、JSON_TABLE、XMLTABLE、および UNNEST の表相関名はオプションです: 794 ページの『table-reference』
- 関数を STATEMENT DETERMINISTIC として定義できます: 1076 ページの『CREATE FUNCTION (外部スカラー)』、 1100 ページの『CREATE

FUNCTION (外部表)』、 1135 ページの『CREATE FUNCTION (SQL スカラー)』、 1151 ページの『CREATE FUNCTION (SQL 表)』

- SQL プロシージャーまたは関数内の非修飾の関数、変数、およびタイプは、SYSROUTINEDEP に CURRENT PATH の値を記録します: 1208 ページの『CREATE PROCEDURE (SQL)』、 1135 ページの『CREATE FUNCTION (SQL スカラー)』、および 1151 ページの『CREATE FUNCTION (SQL 表)』
- CREATE TRIGGER で動的ステートメントが 3 パートの名前を使用できます: 1302 ページの『CREATE TRIGGER』
- DELETE および UPDATE で ORDER BY、FETCH、および OFFSET を使用できます: 1404 ページの『DELETE』 および 1750 ページの『UPDATE』
- スキーマでの GRANT および REVOKE: 1529 ページの『GRANT (スキーマ特権)』 および 1644 ページの『REVOKE (スキーマ特権)』
- INCLUDE ステートメントが SQL プロシージャー、関数、およびトリガーにサポートされました: 1816 ページの『INCLUDE ステートメント』
- SYSPARTITIONSTAT が部分トランザクションに関する TEXT 値と情報を戻します: 2017 ページの『SYSPARTITIONSTAT』
- SYSPROGRAMSTAT がモジュール名を戻します: 2027 ページの『SYSPROGRAMSTAT』

新規情報または変更情報の見分け方

技術上の変更が加えられた場所を見分けるのに役立つように、Information Center では以下のイメージを使用しています。

-  イメージにより、新規または変更された情報の開始点を示します。
-  イメージにより、新規または変更された情報の終了点を示します。

今回のリリースの新規情報または変更情報に関するその他の情報は、プログラム資料説明書を参照してください。

第 1 章 概念

この章では、構造化照会言語 (SQL) を使用する際に理解しておくべき概念について概観します。本書の残りの部分に含まれる参照資料では、より詳細な観点から説明します。

リレーショナル・データベース

リレーショナル・データベース は、一組の表と見なすことができ、データの関係モデルにしたがって扱うことができるデータベースです。リレーショナル・データベースには、データの保管、アクセス、および管理に使用される一組のオブジェクトが含まれます。このようなオブジェクトには、表、ビュー、索引、別名、ユーザー定義タイプ、関数、プロシージャー、シーケンス、変数、およびパッケージが含まれます。

ユーザーが IBM i からアクセスできるリレーショナル・データベースには、以下の 3 つのタイプがあります。

システム・リレーショナル・データベース
: IBM i にはデフォルトのリレーショナル・データベースが 1 つあります。システム・リレーショナル・データベースは、常に IBM i にとってローカルのデータベースです。このデータベースは、IBM i に接続しているディスクに存在しているデータベース・オブジェクトのうち、独立補助記憶域プールに保管されているものを除くすべてのオブジェクトから成っています。独立補助記憶域プールについての詳細は、IBM i Information Center のシステム管理カテゴリーを参照してください。

  デフォルトでは、システム・リレーショナル・データベースの名前は IBM i システム名と同じです。ただし、ADDRDBDIRE (RDB ディレクトリー項目追加) コマンドまたは System i ナビゲーター を使用して、別の名前を割り当てることもできます。

ユーザー・リレーショナル・データベース
: ユーザーは、システム上に独立補助記憶域プールを構成することにより、IBM i 上に追加のリレーショナル・データベースを作成することができます。個々の 1 次独立補助記憶域プールが、それぞれ 1 つのリレーショナル・データベースとなります。このデータベースには、独立補助記憶域プール・ディスク上にあるすべてのデータベース・オブジェクトが含まれます。さらに、独立補助記憶域プールが接続している IBM i 製品のシステム・リレーショナル・データベース内のすべてのデータベース・オブジェクトも、論理的にユーザー・リレーショナル・データベースに含まれます。したがって、1 つのユーザー・リレーショナル・データベース内に作成するスキーマの名前は、そのユーザー・リレーショナル・データベースまたはそれに関連したシステム・リレーショナル・データベースの中に既に存在する名前であってはなりません。

  システム・リレーショナル・データベース内のオブジェクトは、論理的にはユーザー・リレーショナル・データベースに含まれていますが、以下に示す

ように、システム・リレーショナル・データベースとユーザー・リレーショナル・データベースの間で、オブジェクト相互間に依存関係を持たせることができない場合が幾つかあります。

- ビューは、そのビューが参照する表およびビューと同じリレーショナル・データベースに存在するスキーマの中に作成する必要があります。ただし、QTEMP 内に作成するビューは、ユーザー・リレーショナル・データベース内の表およびビューを参照することができます。
- 索引を作成するときは、その索引が参照する表と同じリレーショナル・データベース内にあるスキーマの中に作成する必要があります。
- トリガーまたは制約を作成するときは、その基本表と同じリレーショナル・データベース内にあるスキーマの中に作成する必要があります。
- 1 つの参照制約の中の親表および従属表は、両方が同じリレーショナル・データベースの中にあることが必要です。
- システム・リレーショナル・データベース内のオブジェクトが参照できるのは、同じシステム・リレーショナル・データベース内の関数、プロシージャー、およびタイプのみです。ただし、ユーザー・リレーショナル・データベース内のオブジェクトは、システム・リレーショナル・データベースまたは同じユーザー・リレーショナル・データベース内の関数、プロシージャー、およびタイプを参照することができます。しかし、相手方のリレーショナル・データベースが使用可能になっていないと、このようなオブジェクトに対する操作は失敗することがあります。例えば、ユーザー・リレーショナル・データベースをオフに変更し、さらに別のシステムに対してオンに変更した場合などが、これに相当します。

ユーザー・リレーショナル・データベースは、独立補助記憶域プールがオンにされている間は、IBM i にとってローカルの関係にあります。独立補助記憶域プールは、1 つの IBM i ではオフに変更し、別の IBM i ではオンに変更することができます。したがって、同じユーザー・リレーショナル・データベースが、特定の IBM i にとってある時点ではローカルになり、別の時点ではリモートになることがあります。独立補助記憶域プールについての詳細は、IBM i Information Center のシステム管理カテゴリーを参照してください。

デフォルトでは、ユーザー・リレーショナル・データベースの名前は独立補助記憶域プール名と同じです。ただし、ADDRDBDIRE (RDB ディレクトリー項目追加) コマンドまたは System i ナビゲーター を使用して、別の名前を割り当てることもできます。

リモート・リレーショナル・データベース

他の IBM i 製品および IBM i 以外の製品上にあるリレーショナル・データベースには、リモート・アクセスすることができます。この種のリレーショナル・データベースは、ADDRDBDIRE (RDB ディレクトリー項目追加) コマンドまたは System i ナビゲーター を使用して登録する必要があります。

データベース・マネージャーとは、IBM iライセンス内部コード と、リレーショナル・データベースを管理するコードの Db2 for i 部分を総称的に指す名前です。

構造化照会言語 (SQL)

構造化照会言語 (SQL) は、リレーショナル・データベース内のデータを定義および操作するための標準化言語です。データの関係モデルの概念にしたがうと、データベースは表の集合であると考えられます。また、表内の値によって関連が表現されるとともに、1 つ以上の基本表から得られる結果表を指定することによってデータが検索されます。

SQL ステートメントは、データベース・マネージャーによって実行されます。データベース・マネージャーには、結果表の指定を、データ検索を最適化する一連の内部命令に変換する機能があります。SQL ステートメントを準備 するときに、この変換が行われます。この変換を行うことを、バインドする とも言います。

SQL ステートメントを実行するには、あらかじめ実行可能 SQL ステートメントがすべて準備されている必要があります。準備の結果は、ステートメントの実行可能形式、つまり実行形式 です。 SQL は、SQL ステートメントを準備する方式とステートメントの実行形式の持続期間によって、静的 SQL と動的 SQL に区別されます。

静的 SQL

静的 SQL ステートメントのソース (原始) 形式は、ホスト言語 (COBOL、C、または Java など) で書かれたアプリケーション・プログラム内部に組み込まれます。このステートメントは、アプリケーション・プログラムの実行前に準備されます。また、このステートメントの実行形式は、アプリケーション・プログラムが実行された後も残ります。

静的 SQL ステートメントが入っているソース (原始) プログラムは、コンパイルする前に、SQL プリコンパイラーで処理する必要があります。プリコンパイラーは、SQL ステートメントの構文をチェックしてホスト言語のコメントに変換し、さらにデータベース・マネージャーを呼び出すホスト言語のステートメントを生成します。

SQL アプリケーション・プログラムの準備には、プリコンパイル、その静的 SQL ステートメントの準備、および変更されたソース・プログラムのコンパイルが含まれます。

動的 SQL

組み込み動的 SQL ステートメントを含むプログラムは、静的 SQL を含むプログラムの場合と同じように、しかし静的 SQL とは異なり、プリコンパイルを行う必要があり、動的 SQL ステートメントは実行時に構成および準備されます。動的 SQL ステートメントのソース形式は、静的 SQL ステートメントの PREPARE または EXECUTE IMMEDIATE を使用して、プログラムからデータベース・マネージャーに文字ストリングまたはグラフィック・ストリングとして渡されます。PREPARE ステートメントを使用して準備されたステートメントは、DECLARE CURSOR、DESCRIBE、または EXECUTE のいずれかのステートメント内で参照できます。このステートメントの実行形式は、接続が継続している間、または最後の SQL プログラムが呼び出しスタックから出るまで存続します。

REXX アプリケーションに組み込まれた SQL ステートメントは動的 SQL です。対話式 SQL 機能または呼び出しレベル・インターフェース (CLI) にサブミットされる SQL ステートメントも、動的 SQL ステートメントであるといえます。

拡張動的 SQL

拡張動的 SQL ステートメントは、完全に静的でもなく、完全に動的でもありません。 QSQPRCED API は、拡張動的 SQL 機能を提供します。動的 SQL 機能と同様に、この API を使用して、ステートメントを準備し、記述し、実行することができます。動的 SQL とは異なり、この API によってパッケージ中に準備された SQL ステートメントは、そのパッケージまたはステートメントが明示的に削除されるまで、存続します。詳しくは、IBM i Information Center のプログラミング・カテゴリーの中で、データベースおよびファイル API に関する情報を参照してください。

対話式 SQL

対話式 SQL 機能は、すべてのデータベース・マネージャーに関連付けられています。あらゆる対話式 SQL 機能は、本質的には、ワークステーションからステートメントを読み取り、ステートメントを動的に準備して実行し、その結果をユーザーに表示する SQL アプリケーション・プログラムです。このような SQL ステートメントは、対話式に 出されるステートメントと呼ばれます。

Db2 for i の対話式機能は、STRSQL コマンド、STRQM コマンド、または System i ナビゲーターの SQL スクリプト実行サポートによって呼び出されます。SQL の対話式機能の詳細については、「SQL プログラミング」および「Query Manager ご使用の手引き  を参照してください。

SQL コール・レベル・インターフェースおよび Open Database Connectivity

Db2 コール・レベル・インターフェース (CLI) は、動的 SQL ステートメントを処理するための関数をアプリケーション・プログラムに提供するアプリケーション・プログラミング・インターフェースです。Db2 CLI により、ユーザーはどの ILE 言語を使用している場合でも、Db2 for i が提供するサービス・プログラムへのプロシージャー呼び出しを使用して、SQL 機能に直接アクセスできます。CLI プログラムは、Microsoft や他のベンダーから入手できる、ODBC データ・ソースへのアクセスを可能にする Open Database Connectivity (ODBC) ソフトウェア開発キットを使用してコンパイルすることもできます。組み込み SQL とは異なり、プリコンパイルの必要はありません。このインターフェースを使用して開発されたアプリケーションは、個々のデータベースごとにコンパイルせずに、さまざまなデータベースに対して実行できます。インターフェースを通して、アプリケーションは実行時にプロシージャーを呼び出し、データベースに接続し、SQL ステートメントを実行し、戻されたデータや状況に関する情報を入手します。

Db2 CLI インターフェースは、組み込み SQL では利用不能な多くの機能を備えています。例えば次のようなものがあります。

- CLI が提供する関数呼び出しは、Db2 データベース管理システム・ファミリーに共通の、一貫した方法でのデータベース・システム・カタログ情報の照会、取

り出しをサポートしています。これにより、アプリケーション・サーバー特定のカタログ照会を作成する必要性が減ります。

- CLI を使用して作成されたアプリケーション・プログラムから呼び出されたストアード・プロシージャーは、これらのプログラムに対して結果セットを戻すことができます。

使用できる機能とその構文の詳細については、「Db2 UDB for iSeries SQL 呼び出しレベル・インターフェース (ODBC)」を参照してください。

Java DataBase Connectivity (JDBC) および組み込み SQL for Java (SQLJ) プログラム

Db2 for i は、標準に準拠した 2 つの Java プログラミング API、つまり、Java Database Connectivity (JDBC) と embedded SQL for Java (SQLJ) をインプリメントしています。どちらも、Db2 にアクセスする Java アプリケーションやアプレットを作成できます。

JDBC 呼び出しは、Java 固有の方式によって、Db2 CLI への呼び出しに変換されます。Db2 for i データベースには、2 つの JDBC ドライバーを介してアクセスできます。IBM Developer Kit for Java ドライバーまたは IBM Toolbox for Java JDBC ドライバーです。IBM Toolbox for Java JDBC ドライバーについての詳しい情報は、「IBM Toolbox for Java」を参照してください。

JDBC では、静的 SQL は使用できません。SQLJ アプリケーションは、データベースへの接続や SQL エラーの処理などの作業の基礎として JDBC を使用しますが、SQLJ ソース・ファイルに 静的 SQL ステートメントを含めることもできます。SQLJ ソース・ファイルは、まず SQLJ 変換プログラムを使って変換しないと、得られた Java ソース・コードをコンパイルできません。

JDBC および SQLJ アプリケーションの詳細については、「Developer Kit for Java」を参照してください。

OLE DB and ADO (ActiveX Data Object)

IBM i Access for Windows には、Windows クライアント PC からの Db2 クライアント/サーバー・アプリケーション開発を迅速かつ容易にするものとして、Programmer's Toolkit とともに OLE DB Provider が組み込まれています。詳しくは、IBM i Information Center の中のIBM i Access for Windows OLE DB Provider の説明を参照してください。

.NET

IBM i Access for Windows には、Windows クライアント PC からの Windows クライアント/サーバー・アプリケーション開発を迅速かつ容易にするものとして、.NET Provider が組み込まれています。詳しくは、IBM i Information Center の中の IBM i Access for Windows .NET Provider の説明を参照してください。

スキーマ

リレーショナル・データベース内のオブジェクトはいくつかに分けられ、スキーマというまとまりに編成されます。スキーマは、リレーショナル・データベース内のオブジェクトを論理的に分類したものです。

スキーマ名 は、SQL オブジェクト名 (表、ビュー、索引、トリガーなど) の修飾子として使用します。スキーマは、コレクションまたはライブラリーとも呼ばれます。

スキーマには名前があります。別のシステム名を付けることもできます。システム名は、IBM i オペレーティング・システムが使用する名前です。SQL ステートメントでスキーマ名を指定する場所では、どちらの名前でも使用できます。

スキーマと XML スキーマは別々のものであり、混同すべきではありません。XML スキーマは、XML 文書の構造を記述し、内容を検証するための標準です。

各データベース・マネージャーは、そのデータベース・マネージャーが使用するために予約されているスキーマのセットをサポートします。このようなスキーマのことを、システム・スキーマといいます。スキーマ SESSION、および 'SYS' と 'Q' で始まるスキーマはシステム・スキーマです。

ユーザー・オブジェクトを SESSION 以外のシステム・スキーマの中に作成してはなりません。 SESSION は常に、宣言済みの一時テーブルのスキーマ名として使用されます。ユーザーが 'SYS' や 'Q' で始まるスキーマを作成することはできません。

スキーマは、リレーショナル・データベース内のオブジェクトでもあります。CREATE SCHEMA ステートメントを使用すれば、スキーマを明示的に作成できます。[1]詳しくは、 1224 ページの『CREATE SCHEMA』を参照してください。

スキーマに含まれるオブジェクトは、オブジェクトが作成されるときに、スキーマに割り当てられます。割り当てられるスキーマは、オブジェクトの名前 (スキーマ名で特別に修飾されている場合) またはデフォルトのスキーマ名 (修飾されていない場合) によって決まります。

例えば、ユーザーが C という名前のスキーマを作成するとします。

```
CREATE SCHEMA C
```

その後、次のステートメントを実行すると、スキーマ C の中に X と呼ばれる表を作成できます。

```
CREATE TABLE C.X (COL1 INT)
```

1. CRTLIB CL コマンドを使用してスキーマを作成することもできますが、CREATE SCHEMA ステートメントを使用した場合に作成されるカタログ・ビュー、ジャーナル、ジャーナル・レシーバーは、CRTLIB では作成されません。

表

表 は、データベース・マネージャーが管理する論理構造です。表は、列と行から形成されます。表の各行には、固有の順序はありません。各列と行が交差する個所には、値 と呼ばれるデータ項目があります。列 とは、同一のタイプに属する値の集まりを指します。行 とは、先頭から *n* 番目の値が、表の *n* 番目の列の値になるように並んでいる一連の値を指します。

表には、以下の 3 つのタイプがあります。

- 基本表 は CREATE TABLE ステートメントにより作成され、持続的なユーザー・データを保持するために使用されます。詳しくは、 1238 ページの『CREATE TABLE』を参照してください。

  基本表は、名前を持ち、異なるシステム名を持つ場合があります。システム名は、IBM i オペレーティング・システムによって使用される名前です。SQL ステートメントで表名 を指定する場所には、どちらの名前でも使用できます。

  基本表の列は、名前を持ち、異なるシステム列名を持つ場合があります。システム列名は、IBM i オペレーティング・システムによって使用される名前です。SQL ステートメントで列名 を指定する場所には、どちらの名前でも使用できます。詳しくは、 1238 ページの『CREATE TABLE』を参照してください。

  マテリアライズ照会表 は、CREATE TABLE ステートメントで作成する基本表であり、選択ステートメント から派生する (マテリアライズする) データを格納するための表です。ソース表には、基本表、ビュー、表式、またはユーザー定義表関数のいずれかを指定できます。選択ステートメント は、マテリアライズ照会表にあるデータを最新表示するために使用する照会を指定します。

  マテリアライズ照会表を使用すると、SQL 照会のパフォーマンスを高めることができます。データベース・マネージャーは、マテリアライズ照会表のデータを使用して照会の一部を解決できると判断した場合に、マテリアライズ照会表を使用するように照会を書き直す可能性があります。マテリアライズ照会表の作成についての詳細は、 1238 ページの『CREATE TABLE』を参照してください。

  テンポラル表 は、表内のデータに対するすべての変更に時間ベースの状態情報を関連付けます。データの過去の状態を照会できるように、データベースは履歴表 に履歴行 (削除された行、または更新された行の元の行) を保管します。テンポラル表の作成についての詳細は、 982 ページの『ADD VERSIONING USE HISTORY TABLE history-table-name』を参照してください。

  パーティション表 は、1 つ以上のローカル・パーティション (メンバー) に含まれているデータから成る表です。どの行をどのパーティションに挿入するかは、2 つのメカニズムで指定できるようになっています。範囲区分化というメカニズムを使用すると、パーティションごとに異なる範囲の値を指定することができます。行が挿入されると、行に指定された値と各パーティションに指定された範囲が比較され、どのパーティションが適切かが判別されます。ハッシュ・パーティションというメカニズムを使用すると、パーティション・キーを指定することができ、そのキーを元にハッシュ・アルゴリズムを実行して適切なパーティション

を判別することができます。パーティション・キー とは、パーティション表内の 1 つ以上の列の集まりであり、このキーを使用して行がどのシステムに属するべきかを判別します。

分散表 とは、そのデータがノード・グループに分配されている表を指します。ノード・グループ とは、複数のシステムが集まった論理的なグループを提供するオブジェクトです。パーティション・キー とは、分散表内の 1 つ以上の列の集まりであり、このキーを使用して行がどのシステムに属するべきかを判別します。分散表の詳細については、「Db2 UDB for iSeries マルチ・システム」を参照してください。

- 結果表 とは、データベース・マネージャーが照会から、選択または生成を行った列の集まりです。照会の詳細については、 787 ページの『第 6 章 照会』を参照してください。
- 宣言済み一時表 は、DECLARE GLOBAL TEMPORARY TABLE ステートメントによって作成され、単一のアプリケーションに代わって一時データを保持するために使用されます。この表は、そのアプリケーションがデータベースから切断されたときに、暗黙的に除去されます。

キー

キー とは、索引、ユニーク制約、または参照制約の記述の中で識別されている 1 つ以上の式です。同一の式が複数のキーに含まれることもあります。

複合キー は、同じ基本表のいくつかの式を順序付けしたものです。これらの式の順序付けは、基本表内で順序による制約は受けません。値 という用語は、複合キーに関して使用した場合には、複合値のことを指します。したがって、「外部キーの値は、基本キーの値に等しくなければならない」などの規則は、外部キーの値の各コンポーネントが、基本キーの値の対応するコンポーネントに等しくなければならないことを意味しています。

制約

制約 は、データベース・マネージャーが実施する規則です。

制約には、以下の 3 つのタイプがあります。

- ユニーク制約 は、1 つの表の中の 1 つ以上の列に重複値があってはならないという規則です。固有キーと基本キーが、ユニーク制約としてサポートされています。例えば、製造業者テーブルの製造業者識別子にユニーク制約を定義すると、1 つの製造業者識別子に 2 つの製造業者を指定できなくなります。
- 参照制約 は、1 つ以上の表にある 1 つ以上の列の値に関する論理規則です。例えば、ある企業の製造業者に関する情報がいくつかの表で共有されているとします。製造業者の ID はときどき変更されます。そこで、参照制約を定義して、表の製造業者の ID は製造業者情報の製造業者 ID と一致していなければならないこととします。この制約によって、ともすると製造業者情報が欠落してしまいかねない挿入、更新、削除の操作を防ぐことができます。
- チェック制約。これは、特定の表に追加されるデータに制約を課します。例えば、人事情報が含まれている表で給与データを追加/更新するときに、従業員の給与レベルが最低でも $20 000 以上になるように、チェック制約を設定できます。

固有制約

ユニーク制約 は、キーの値が固有な場合にのみ有効であることを示す規則です。固有の値を持つように制約されているキーは、固有キー と呼ばれます。固有制約は、固有索引の使用によって強制実施されます。固有索引は、INSERT や UPDATE ステートメントの実行の過程でデータベース・マネージャーによって使用され、キーの値の固有性が確保されます。

固有制約には、次の 2 つのタイプがあります。

- CREATE TABLE または ALTER TABLE ステートメントを使用して、固有キーを基本キーとして定義する。 1 つの基本表で複数の基本キーを持つことはできません。基本キーを構成する桁には NULL 値が許されないという規則を強制する、表検査制約が暗黙的に追加されます。基本キーに基づく固有索引は、基本索引 と呼ばれます。
- CREATE TABLE または ALTER TABLE ステートメントの UNIQUE 文節を使用して、固有キーを定義する。 1 つの基本表で、「固有」キーのセットを複数持つことができます。複数の NULL 値が許されます。

参照制約の外部キーによって参照される固有キーは、親キー と呼ばれます。親キーは、基本キー、または UNIQUE キーのいずれかです。基本表が、参照制約において親として定義される場合、その基本キーが、デフォルトの親キーになります。

固有制約を強制実施するのに使用する固有索引は、固有制約の定義時に暗黙的に作成されます。または、CREATE UNIQUE INDEX ステートメントを使用して定義することもできます。

固有制約の定義についての詳細は、 942 ページの『ALTER TABLE』または 1238 ページの『CREATE TABLE』を参照してください。

参照制約

参照保全 とは、すべての外部キーのすべての値が有効であるデータベースの状態です。外部キー とは、参照制約の定義の一部であるキーです。

参照制約 は、外部キーの値が以下の場合にのみ有効であることを示す規則です。

- それらが、親キーの値として現れている。または、
- 外部キーのコンポーネントに NULL のコンポーネントがある。

親キーを含む基本表は、参照制約の親表 と呼ばれ、その外部キーを含む基本表は、その表に従属 していると呼ばれます。

参照制約は、任意指定であり、CREATE TABLE ステートメントや ALTER TABLE ステートメントで定義することができます。参照制約は、INSERT、UPDATE、および DELETE ステートメントの実行の過程で、データベース・マネージャーによって課せられます。参照制約は、行が処理されるときに課せられる RESTRICT の削除や更新の規則の場合を除き、ステートメントの完了時に効果的に課せられます。

RESTRICT の削除や更新の規則を伴う参照制約は、常に、他の参照制約よりも前に課せられます。他の参照制約は、順序に関係のない形で課せられます。すなわち、その順序が操作の結果に影響することはありません。1 つの SQL ステートメント内で、

- 行に、CASCADE の削除規則を伴う任意の数の参照制約によって削除のマークを付けることができます。
- 行は、SET NULL または SET DEFAULT の削除規則を伴う 1 つの参照制約によってのみ更新することができます。
- ある参照制約によって更新された行には、CASCADE の削除規則を伴う他の参照制約によって削除のマークを付けることはできません。

参照保全の規則には、以下の概念や用語が関連します。

親キー
: 参照制約の基本キーまたは固有キー。

親行
: 少なくとも 1 つの従属行を持つ行。

親表
: 少なくとも 1 つの参照制約における親である基本表。基本表は、任意の数の参照制約で親として定義することができます。

従属表
: 少なくとも 1 つの参照制約において従属である基本表。基本表は、任意の数の参照制約で従属として定義することができます。従属表は、親表になることもできます。

下層表
: 基本表が、基本表 T の従属であるか、または表 T の従属の下層である場合、その表は表 T の下層です。

従属行
: 少なくとも 1 つの親行を持つ行。

下層行
: 行が、行 p の従属であるか、または行 p の従属の下層である場合、その行は、行 p の下層です。

参照サイクル
: その集合の各表がそれ自身の下層である参照制約の集合。

自己参照行
: それ自身の親である行。

自己参照表
: 同一の参照制約で親であると同時に従属である基本表。この制約を、自己参照制約 といいます。

参照制約の挿入規則では、外部キーの非 NULL の挿入値は、親表の親キーのいずれかの値に一致しなければなりません。複合外部キーのコンポーネントのいずれかの値が NULL である場合、その複合外部キーの値は NULL になります。

参照制約の更新規則は、その参照制約を定義する時点で指定します。選択できる項目は、NO ACTION と RESTRICT です。更新規則は、親または従属表の行が更新される時点で適用されます。参照制約の更新規則では、外部キーの非 NULL の更新値は、親表の親キーのいずれかの値に一致しなければなりません。複合外部キーの値は、そのいずれかのコンポーネントが NULL の場合には NULL として扱われます。

参照制約の削除規則は、その参照制約を定義する時点で指定します。選択できる項目は、RESTRICT、NO ACTION、CASCADE、 SET NULL または SET DEFAULT です。 SET NULL は、外部キーの列に NULL 値が許される列がある場合にのみ指定することができます。

参照制約の削除規則は、親表の行が削除される時点で適用されます。厳密には、この規則は、親表の行が、削除または波及削除操作の対象で (以下で説明)、しかもその行が参照制約の従属表に従属している場合に適用されます。P は親表を、D は従属表を、また p は削除あるいは波及削除操作の対象である親行を表すものとします。削除規則が、

- RESTRICT または NO ACTION の場合、エラーが戻され、行の削除は行われません。
- CASCADE の場合、削除操作は、D の p の従属行に波及します。
- SET NULL の場合、D の p の各従属行の外部キーのNULL 可能な各列は NULL に設定されます。
- SET DEFAULT の場合、D の p の各従属行の外部キーの各列はそのデフォルト値に設定されます。

表が親である各参照制約は、それ自体の削除規則を持ち、適用可能なすべての削除規則が、削除操作の結果の判別に使用されます。したがって、行が RESTRICT または NO ACTION の削除規則を伴う参照制約に従属する場合、または削除が RESTRICT または NO ACTION の削除規則を伴う参照制約に従属する下層のいずれかにカスケードする場合は、その行は削除できません。

親表 P からの行の削除は、他の表を巻き込み、それらの表の行に影響を与えることがあります。

- 表 D が P に従属し、削除規則が RESTRICT または NO ACTION の場合、D はその操作に関与しますが、その操作による影響を受けません。
- D が P に従属し、削除規則が SET NULL の場合、D はその操作に関与し、D の行はその操作の過程で更新されることがあります。
- D が P に従属し、削除規則が SET DEFAULT の場合、D はその操作に関与し、D の行はその操作の過程で更新されることがあります。
- D が P に従属し、削除規則が CASCADE の場合、D はその操作に関与し、D の行はその操作の過程で削除されることがあります。

  D の行が削除される場合、 P での削除操作を D への伝搬といいます。 D が親表でもある場合、今度は、ここリストされているアクションが D の従属にも適用されます。

基本表が P に対する削除操作に関与することがある場合は、その表は P に対して連結削除にある と言います。したがって、ある基本表が表 P に従属しているか、または P からの削除操作のカスケード先である基本表に従属している場合は、その基本表は表 P に対して連結削除にあることになります。

参照制約の定義方法について詳しくは、 942 ページの『ALTER TABLE』または 1238 ページの『CREATE TABLE』を参照してください。

表検査制約

表検査制約 は、基本表のすべての行で許される値を指定する規則です。表検査制約の定義には、基本表のどの行についても FALSE であってはならないという検索条件が含まれます。

表 T の表検査制約の検索条件で参照される各列は、T の列を識別する必要があります。検索条件の詳細については、 275 ページの『検索条件』を参照してください。

1 つの基本表が複数の表検査制約を持つことができます。基本表で定義されている各表検査制約は、以下のいずれかの場合にデータベース・マネージャーにより強制実施されます。

- 基本表に行が挿入される場合
- 基本表の行が更新される場合

その基本表で挿入または更新されるそれぞれの行に対して、表検査制約の検索条件を適用することによって、表検査制約が強制されます。いずれかの行に対する検索条件の結果が FALSE であれば、エラーが戻されます。

検査制約の定義方法について詳しくは、 942 ページの『ALTER TABLE』または 1238 ページの『CREATE TABLE』を参照してください。

索引

索引 とは、基本表の各行に対するポインターの集まりです。それぞれの索引は、1 つ以上の表の列内のデータ値をもとにしています。索引は、表内のデータとは独立のオブジェクトです。索引が作成されると、データベース・マネージャーは索引の構造を構築し、それを自動的に維持します。

索引は、名前を持ち、さらに別のシステム名を持つことができます。システム名は、IBM i オペレーティング・システムによって使用される名前です。SQL ステートメントで索引名 を指定する場所には、どちらの名前でも使用できます。詳しくは、 1166 ページの『CREATE INDEX』を参照してください。

データベース・マネージャーは、次の 2 つのタイプの索引を使用します。

- 2 進基数ツリー索引

  2 進基数ツリー索引は、表の行に対して特定の順序を与えます。データベース・マネージャーは、次のことを目的としてこれを使用します。

  – パフォーマンスを向上させる。ほとんどの場合、索引を使用した方がデータへのアクセスは高速になります。
  – 固有性を確実にする。固有索引がある表には、同一のキーを持つ行を入れることはできません。
- コード化ベクトル索引

  コード化ベクトル索引は、表の行に対して特定の順序を与えることはありません。データベース・マネージャーは、パフォーマンスを向上させるためにのみこれらの索引を使用します。

  コード化ベクトルのアクセス・パスはコード化ベクトル索引の助けにより機能し、コードを異なるキー値に割り当ててからこれらの値を配列で表すことによっ

て、データベース・ファイルへのアクセスを提供します。配列のエレメントは、表すべき異なる値の数によって、長さは 1、2、または 4 バイトになります。このサイズが小さく、比較的単純であるために、コード化ベクトルのアクセス・パスによってスキャンが高速化され、並列処理をより容易に行うことができるようになります。

索引 は CREATE INDEX ステートメントで作成されます。索引の作成についての詳細は、 1166 ページの『CREATE INDEX』を参照してください。

索引について詳しくは、IBM Db2 for i 索引付けの方法および戦略 (英語) を参照してください。

トリガー

トリガー は、指定の表またはビューに対する削除、挿入、または更新操作が行われるたびに自動的に実行される一組のアクションを定義します。この種の SQL 操作が実行されることを、トリガーがアクティブ化されたと言います。

この一組のアクションには、システム上で可能なほとんどすべての操作を含めることができます。許されない操作は、以下のような数少ない操作です。

- コミットまたはロールバック (同一のコミットメント定義がトリガーのアクション、およびトリガー対象のイベントで使用されている場合)
- CONNECT、SET CONNECTION、DISCONNECT、および RELEASE ステートメント
- SET SESSION AUTHORIZATION

制約の詳細なリストについては、 1302 ページの『CREATE TRIGGER』 および「データベース・プログラミング」を参照してください。

トリガーは、データ保全性の規則を適用するために、参照制約や検査制約と合わせて使用できます。トリガーを使用すると、他の表を更新する、挿入または更新される行の値を自動的に生成または変換する、あるいは データベース・マネージャーの内部と外部の両方の操作を実行する関数を呼び出す、といったこともできるので、制約より強力です。例えば、新規の値が所定の数量を超えている場合、トリガーでは、列の更新を防ぐ代わりに、有効な値で置き換え、管理者に無効な更新について通知することができます。

トリガーは、異なる状態のデータが含まれる過渡的ビジネス規則 (例えば、給与は 10 % までしか増やせない) を定義し、適用するのに便利なメカニズムです。このような制限は、増加前と増加後の給与の値を比較することが必要です。1 つの状態のデータしか含まれていない規則の場合は、参照制約や検査制約の使用を考えてください。

トリガーを使用すると、ビジネス規則を適用するのに必要なアプリケーション論理をデータベース内に移動することができるので、アプリケーションの開発が迅速になり、保守も容易になりますが、それはビジネス規則が複数のアプリケーションで繰り返されることがなくなり、特定の規則がトリガーに集約されたためです。データベース内の論理 (例えば、前述のような、表の給与列の増加に関する制限) を使用

して、データベース・マネージャーはアプリケーションが給与列に加える変更の妥当性を検査します。また、論理が変更されても、アプリケーション・プログラムを変更する必要がありません。

トリガーの作成についての詳細は、 1302 ページの『CREATE TRIGGER』を参照してください。[2]

トリガーはオプションであり、CREATE TRIGGER ステートメントまたは**物理ファイル・トリガーの追加 (ADDPFTRG)** CL コマンドを使用して定義されます。トリガーを除去するには、DROP TRIGGER ステートメントまたは**物理ファイル・トリガーの除去 (RMVPFTRG)** CL コマンドを使用します。トリガーの作成について詳しくは、CREATE TRIGGER ステートメントの項を参照してください。トリガー全般についての詳しい情報は、 1302 ページの『CREATE TRIGGER』 ステートメントの項の説明、または SQL プログラミングおよび データベース・プログラミングを参照してください。

トリガーの作成時に定義される、トリガーを起動する時期を決めるのに使われる基準が、いくつかあります。

- サブジェクト表 (トリガー表とも呼ばれます) は、トリガーが定義される表またはビューを定義します。
- トリガー・イベント は、対象表を変更する特定の SQL 操作を定義します。この操作としては、削除、挿入、更新が可能です。
- トリガー起動時 は、トリガーを起動するのは、対象表に対するトリガー・イベントの実行前であるか、実行後であるかを定義します。

トリガーを起動するステートメントには、影響を受ける行の集合 が含まれています。これらは、対象表の中の削除、挿入、または更新される行を表します。トリガー細分性 は、トリガー・アクションを実行するのは、そのステートメントに対して一度であるのか、影響を受ける行集合の各行ごとに一度であるのかを定義します。

トリガー・アクション は、オプションの検索条件と、トリガーが起動されるたびに実行される 1 組みの SQL ステートメントから構成されます。SQL ステートメントは、検索条件が指定されない場合、または指定の検索条件が真と評価されるときにだけ実行されます。

トリガー・アクションは、影響を受ける行集合の値を参照することもできます。これは、遷移変数 の使用を通してサポートされます。遷移変数は、対象表の中の列名を使用し、その参照が古い値 (更新前) に対するものか、新しい値 (更新後) に対するものかを識別する指定名によって修飾します。新しい値は、更新または挿入トリガーの前に、SET 遷移変数ステートメントを使用して変更することもできます。影響を受ける行集合の値を参照するもう 1 つの方法として、遷移表 の使用があります。遷移表も対象表の列名を使用しますが、影響を受ける行集合全体を 1 つの表として扱うことができる指定名を持っています。遷移表はトリガーの後でしか使用できません。古い値と新しい値に対して別々の遷移表を定義することも可能です。

2. ADDPFTRG CL コマンドもトリガーを定義します。これには、読み取り操作で活動化されるトリガーも含まれます。

表、イベント、起動時の 1 つの組み合わせに対して、複数のトリガーを指定できます。トリガーが起動される順序は、トリガーが作成された順序と同じです。つまり、最新に作成されたトリガーが、最後に起動されるトリガーになります。

トリガーの起動によって、トリガー・カスケード が生じることがあります。これは、あるトリガーの起動によって、SQL ステートメントが実行され、その実行によって別のトリガーが起動されたり、同じトリガーが再度起動されたりする結果起きるものです。トリガー・アクションでは、最初の変更の結果として更新が行われ、その結果としてさらにトリガーが起動されるといったことも起こります。トリガー・カスケードを使用すると、有効なトリガー・チェーンを起動することが可能で、単一の削除、挿入、または更新ステートメントによって、データベースに対する多数の変更を行うことができます。

トリガーとして実行されるアクションは、トリガーの実行を引き起こす操作の一環であると見なされます。したがって、分離レベルが NC (コミット不可) 以外で、トリガーのアクションがトリガー・イベントと同一のコミットメントを使用して行われる場合には、

- データベース・マネージャーは、操作とその操作の結果として実行されるトリガーがいずれも、すべて完了であるか、またはすべてバックアウトであることを確認します。トリガーの実行を引き起こす操作に先立って行われた操作は、影響を受けません。
- データベース・マネージャーは、該当の操作および関連するトリガーが実行された後で、すべての制約 (RESTRICT 削除規則を伴う制約を除く) を効果的にチェックします。

トリガーの実行を引き起こす SQL ステートメントに既に挿入あるいは更新された行について、削除または更新を許可するかどうかを指定する属性がトリガーにはあります。

- トリガーを定義した際に ALWREPCHG(*YES) が指定されている場合、1 つの SQL ステートメント内で、
  - 同じ SQL ステートメントで挿入または更新した行がある場合、トリガーはその行を更新あるいは削除することができます。これには、トリガーが挿入または更新した行や同じ SQL ステートメントで生じた参照制約も含まれます。
- トリガーを定義した際に ALWREPCHG(*NO) が指定されている場合、1 つの SQL ステートメントで、
  - 行は、その行が、同一のその SQL ステートメントによって挿入、または更新されていない場合にのみ、トリガーによって削除することができます。分離レベルが NC (コミット不可) 以外で、トリガーのアクションがトリガー・イベントと同一のコミットメント定義を使用して行われる場合には、トリガー、または同一の SQL ステートメントにより生じた参照制約による挿入や更新が含まれます。
  - 行は、その行が、同一の SQL ステートメントによって挿入、または更新されていない場合にのみ、トリガーによって更新することができます。分離レベルが NC (コミット不可) 以外で、トリガーのアクションがトリガー・イベントと同一のコミットメント定義を使用して行われる場合には、トリガー、または同一の SQL ステートメントにより生じた参照制約による挿入や更新が含まれます。

CREATE TRIGGER ステートメントを使用して作成されたトリガーは、すべて暗黙的に ALWREPCHG(*YES) 属性を持っています。

ビュー

ビュー は、1 つ以上の表のデータを見るための代替方法を提供します。

ビューは、結果表の名前の付いた指定です。その指定は、ビューが SQL ステートメントで参照される時点で実際に実行される SELECT ステートメントです。したがって、ビューは、基本表と同様に、列や行を持つものとして考えることができます。検索の場合、すべてのビューを基本表と同様に使用することができます。挿入、更新、または削除の操作で、ビューを使用できるか否かは、その定義に依存します。

ビューに対して索引を作成することはできません。ただし、ビューの基礎となる表に対して作成された索引は、そのビューの操作のパフォーマンスを向上させることがあります。

ビューの列が、基本表の列から直接派生する場合、その列は、基本表の列に適用される制約をいずれも継承します。例えば、ビューがその基本表の外部キーを含む場合、そのビューを使用する INSERT および UPDATE 操作は、基本表と同じ参照制約が課せられます。同様に、ビューの基本表が親表である場合、そのビューを使用する DELETE 操作は、基本表に関する DELETE 操作と同一の規則に従います。また、ビューは、その基本表に適用されるトリガーをいずれも継承します。例えば、ビューの基本表が更新トリガーを持つ場合、そのトリガーは、そのビューに更新が行われる時点で実行されます。

ビューは名前を持ち、また異なるシステム名を持つこともできます。システム名は、IBM i オペレーティング・システムによって使用される名前です。 SQL ステートメントでビュー名 を指定する場所には、どちらの名前でも使用できます。

ビューの列は名前を持ち、また異なるシステム列名を持つことができます。システム列名は、IBM i オペレーティング・システムによって使用される名前です。SQL ステートメントで列名 を指定する場所には、どちらの名前でも使用できます。

ビュー は CREATE VIEW ステートメントで作成されます。ビューの作成についての詳細は、 1342 ページの『CREATE VIEW』を参照してください。

ユーザー定義タイプ

ユーザー定義タイプ は、CREATE ステートメントを使用してデータベースに定義されるデータ・タイプです。

ユーザー定義のデータ・タイプには、以下の 2 つのタイプがあります。

- 特殊タイプ
- 配列タイプ

特殊タイプ は、その内部表現を組み込みのデータ・タイプ (そのソース・タイプ) と共用するユーザー定義のタイプですが、大部分の演算では別のデータ・タイプ、および、互換性のないデータ・タイプと考えられます。特殊タイプは、SQL

CREATE TYPE (特殊) ステートメントを使用して作成します。特殊タイプを使用して、表の列またはルーチンのパラメーターを定義できます。詳しくは、 1328 ページの『CREATE TYPE (特殊)』、および 104 ページの『ユーザー定義タイプ』を参照してください。

配列タイプ は、組み込みデータ・タイプの 1 つの列配列を定義するユーザー定義タイプです。配列タイプは、SQL CREATE TYPE (配列) ステートメントを使用して作成します。配列タイプは、プロシージャーまたはスカラー関数のパラメーターとして、および SQL プロシージャーまたは SQL スカラー関数の変数として使用できます。詳しくは、 1323 ページの『CREATE TYPE (配列)』、および 104 ページの『ユーザー定義タイプ』を参照してください。

別名

別名 は、表またはビューの代替名です。

既存の表またはビューを参照できる場合は、別名を使用して表またはビューを参照できます。[3] 別名によって表またはビューを参照するというオプションは、明示的に構文図に示されることはありません。また、SQL ステートメントの説明で記述されることもありません。表およびビューと同様に、別名は作成し、除去し、それに関係した注記あるいはラベルを付けることができます。別名を使用する際に、権限は不要です。ただし、別名で参照する表とビューにアクセスするには、現在のステートメントに関する適切な権限がやはり必要になります。

別名は、名前を持ち、さらに別のシステム名を持つことができます。システム名は、IBM i オペレーティング・システムによって使用される名前です。SQL ステートメントで別名 を指定する場所には、どちらの名前でも使用できます。

別名 は CREATE ALIAS ステートメントで作成されます。別名の作成についての詳細は、 1065 ページの『CREATE ALIAS』を参照してください。

パッケージとアクセス・プラン

パッケージ は、SQL ステートメントを実行するために使用する制御構造を含むオブジェクトです。

パッケージは、分散プログラムを準備するときに作成されます。制御構造は、バインドされた形式または操作可能形式の SQL ステートメントと考えることができます。[4]パッケージ内のすべての制御構造は、単一のソース・プログラム内に組み込まれている SQL ステートメントをもとにして作成されます。

本書では、アクセス・プラン という用語を、 SQL ステートメントを実行するために使用する制御構造を含むパッケージ、プロシージャー、関数、トリガー、およびプログラムやサービス・プログラムなど全般を表すものとして使用しています。例

3. すべてのコンテキストで別名を使用できるわけではありません。例えば、ほとんどの SQL スキーマ・ステートメントでは、データベース・ファイル内の個々のメンバーを指す別名は使用できません。詳しくは、 1065 ページの『CREATE ALIAS』を参照してください。

4. 非分散 SQL プログラム、非分散サービス・プログラム、SQL 関数、および SQL プロシージャーの場合は、SQL ステートメントを実行するために使用される制御構造は、オブジェクトに関連したスペースに保管されます。

えば、DROP ステートメントについて、あるオブジェクトをドロップすると、そのオブジェクトを参照するアクセス・プランすべてが無効になると説明されています (1441 ページの『DROP』を参照)。この説明の意味は、ドロップされたオブジェクトを参照している制御構造を含むパッケージ、プロシージャー、関数、トリガー、およびプログラムやサービス・プログラムはすべて無効になるということです。

無効になったアクセス・プラン は、それに関連した SQL ステートメントが次に実行されたときに、暗黙的にビルドし直されます。例えば、SELECT INTO ステートメントのアクセス・プラン で使用されている索引がドロップされた場合、次にその SELECT INTO ステートメントが実行されるときに、アクセス・プランはビルドし直されます。

パッケージは、Process Extended Dynamic SQL (QSQPRCED) API を使用して作成することもできます。 Process Extended Dynamic SQL (QSQPRCED) API を使用して作成したパッケージは、Process Extended Dynamic SQL (QSQPRCED) API によってのみ使用することができます。このようなパッケージは、DRDA プロトコルを使用してアプリケーション・サーバーで使用することはできません。詳しくは、IBM i Information Center のプログラミング・カテゴリーの中で、データベースおよびファイル API に関する情報を参照してください。

QSQPRCED API は IBM i Access for Windowsで使用され、SQL ODBC、JDBC、SQLJ、OLD DB、および .NET インターフェースを介して実行される SQL ステートメントをキャッシュに入れるためのパッケージを作成します。

ルーチン

ルーチン とは、実行可能な SQL オブジェクトのことです。

ルーチンには 2 つのタイプがあります。

関数

関数 とは、他の SQL ステートメント内から起動されて、値または表を戻すルーチンのことです。詳しくは、 182 ページの『関数』を参照してください。

関数は、SQL 関数か外部関数のいずれかに分類されます。 SQL 関数は SQL ステートメントを使用して作成され、総称して SQL プロシージャー型言語とも呼ばれます。外部関数は、SQL ステートメントを含む場合も、含まない場合もあるホスト言語プログラムを参照します。

関数 は CREATE FUNCTION ステートメントを使用して作成されます。関数の作成についての詳細は、 1070 ページの『CREATE FUNCTION』を参照してください。

プロシージャー

プロシージャー (ストアード・プロシージャー とも呼ばれます) は、ホスト言語ステートメントと SQL ステートメントの両方を組み込んだ操作を実行するために呼び出すことができるルーチンです。

プロシージャーは、SQL プロシージャーと外部プロシージャーに類別されます。SQL プロシージャーは SQL ステートメントを使用して作成され、総称して SQL プロシージャー型言語とも呼ばれます。外部プロシージャーは、SQL ステートメントを含む場合も、含まない場合もあるホスト言語プログラムを参照します。

プロシージャー は CREATE PROCEDURE ステートメントを使用して作成されます。プロシージャーの作成についての詳細は、 1187 ページの『CREATE PROCEDURE』を参照してください。

SQL のプロシージャーは、ホスト言語のプロシージャーと同様の利点をもたらします。すなわち、共通のコード部分を一度だけ作成し、メンテナンスすることによって、複数のプログラムから呼び出すことができます。ホスト言語、および SQL の両者は、そのローカル・システムに存在するプロシージャーを呼び出すことができます。ただし、SQL は、リモート・システムに存在するプロシージャーも呼び出すことができます。事実、SQL のプロシージャーの主要な利点は、プロシージャーを分散アプリケーションのパフォーマンス特性の向上に使用することができる点にあります。

リモート・システムで、複数の SQL ステートメントの実行が必要であると想定します。そのための方法は、2 とおりあります。プロシージャーがない場合、最初の SQL ステートメントが実行されると、アプリケーション・リクエスターは、該当の操作を行うよう求める要求をアプリケーション・サーバーに送ります。その上で、アプリケーション・リクエスターは、そのステートメントが正常に実行されたか否かを示す応答を待ち、必要に応じて結果を戻します。2 番目およびそれ以降の SQL ステートメントが実行される時点で、アプリケーション・リクエスターは、別の要求を送り、その応答を待ちます。

アプリケーション・サーバーのプロシージャー内に同じ SQL ステートメントが保管されていれば、そのリモート・プロシージャーを参照する CALL ステートメントを実行することができます。その CALL ステートメントが実行されると、アプリケーション・リクエスターは、そのプロシージャーを呼び出す単一の要求を現行サーバーに送ります。その上で、アプリケーション・リクエスターは、そのプロシージャーが正常に実行されたか否かを示す単一の応答を待ち、必要に応じて結果を戻します。

次の 2 つの図は、分散アプリケーションでストアード・プロシージャーを使用することによって、リモート要求の数をどのように減らすことができるかを示しています。 20 ページの図 1 には、多くのリモート要求を作成するプログラムが示されています。

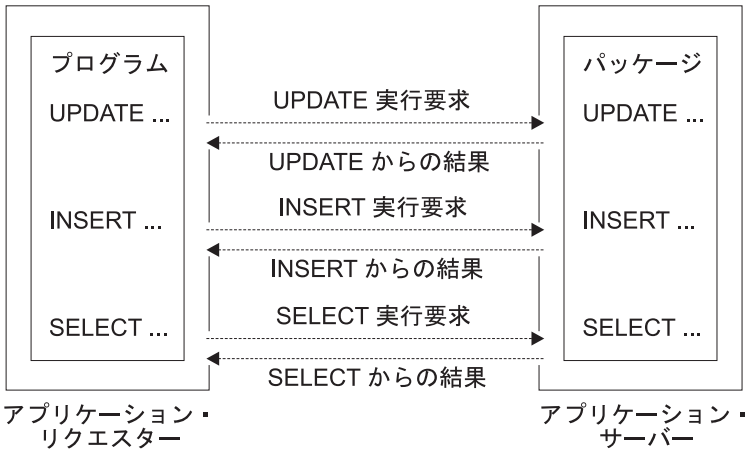


図 *1.* リモート・プロシージャーを持たないアプリケーション

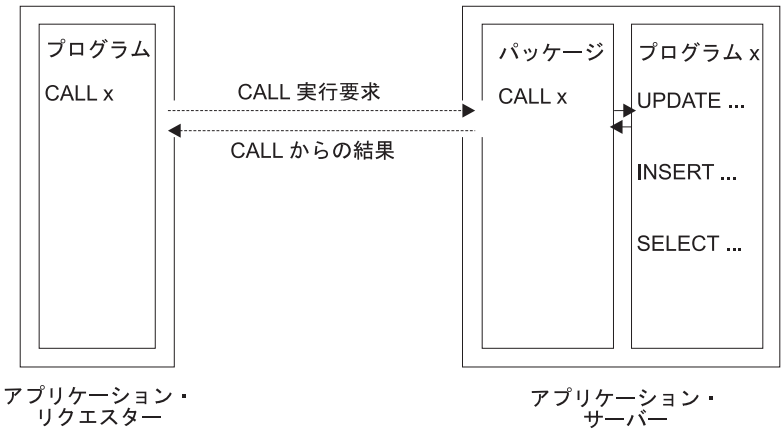


図 *2.* リモート・プロシージャーを持つアプリケーション

シーケンス

シーケンスは、数列を単純な昇順 (または降順) で生成するだけの保管オブジェクトです。シーケンスは、データベース・マネージャーに固有の整数および 10 進数主キーを自動的に生成させて、複数の行および表にわたってキーを調整する手段として使用できます。

シーケンスを使用することにより、プログラマチックに最後に使用した値をロックして増分し、固有の値を生成する代わりに、並列化を活用できます。

シーケンスは、固有キー値を生成するタスクに適しています。多数の表に 1 つのシーケンスを使用したり、生成されるキーを必要とする表ごとに別個のシーケンスを作成したりすることができます。シーケンスは、以下の特性を持ちます。

- シーケンスがリセットされず、かつ値の循環使用が許可されていなければ、固有値が保証されます。
- 定義された範囲内で、値を増加または減少させることができます。
- 連続する値から次の値への増分値として 1 以外の値を持つことができます (デフォルトは 1)。

- リカバリー可能です。

どのシーケンスでも、値はデータベース・マネージャーによって自動的に生成されます。データベースでシーケンスを使用することにより、アプリケーションがデータベース外にシーケンスをインプリメントする場合に生じるパフォーマンス・ボトルネックを回避することができます。シーケンスのカウンターは、トランザクションから独立して増分 (または減分) されます。

シーケンス内にギャップが生じることがあります。ギャップは、トランザクションがシーケンスを 2 回増分すると発生します。トランザクションは、他のトランザクションが同じシーケンスを並行して増分しているために生成された 2 つの数値のギャップを認識する場合があります。ユーザーは、他のユーザーが同じシーケンスから取り出していることに気が付かない場合があります。また、シーケンス数値を生成したトランザクションがロールバックしたために、シーケンスが数値のギャップを生成したように見えることがあります。シーケンスの更新は、トランザクションのリカバリー単位には含まれません。

シーケンスは、CREATE SEQUENCE ステートメントを使用して作成されます。シーケンスは、シーケンス参照 を使用して参照できます。シーケンス参照は、式を使用できる場所であればほとんどの場合使用できます。新しく生成された値を値として戻すか、あるいはその前に生成されていた値を値として戻すかを、シーケンス参照で指定することができます。詳しくは、 237 ページの『シーケンス参照』および 1230 ページの『CREATE SEQUENCE』を参照してください。

シーケンスと ID 列は、類似点はありますが異なります。 ID 列が表の一部であるのに対して、シーケンスはオブジェクトです。複数の表で 1 つのシーケンスを使用できますが、ID 列は単一の表の一部です。

権限、特権、およびオブジェクト所有権

ユーザーは (権限 ID によって識別され) 指定した機能を行う権限がある場合にのみ、 SQL ステートメントを正しく実行することができます。表を作成するには、表を作成する権限が必要であり、表を除去するには、表を除去する権限が必要であるという具合です。

許可には、管理権限と特権の 2 つの形式があります。

管理権限

管理権限の所有者は、リレーショナル・データベースを管理する作業を担当し、データの安全と保全性について責任を負います。

データベース管理者権限
: データベース管理者権限は、リレーショナル・データベース内のすべてのオブジェクトの作成と管理を行う能力をユーザーに付与します。データベース管理者権限の保有者は、リレーショナル・データベース内のすべてのオブジェクトに対して暗黙的にすべての特権を持ちます。

  セキュリティー担当者、および *ALLOBJ 権限を持つすべてのユーザーは、データベース管理者権限を持っています。

セキュリティー管理者権限

セキュリティー管理者権限は、リレーショナル・データベースのセキュリティーを管理する能力をユーザーに付与します。セキュリティー管理者権限の保有者は、リレーショナル・データベースのすべての特権および権限の認可と取り消し、およびオブジェクトの所有権移動を行う能力を持ちます。セキュリティー管理者権限の保有者は、機密データを含む表の行および列のアクセス制御を実施することにより、セキュリティー・ポリシーも管理します。

セキュリティー管理者権限には、表に保管されたデータにアクセスするための特権は包含されません。

QIBM_DB_SECADM 機能使用を持つユーザーはセキュリティー管理者権限を保有します。

特権

特権 とは、ユーザーが実行することを許可されているアクティビティーのことです。権限を持つユーザーは、任意のオブジェクトを作成し、ユーザー自身が所有するオブジェクトにアクセスし、また GRANT ステートメントを使用して、そのユーザー自身が所有するオブジェクトに関する特権 を他のユーザーに与えることができます。

特権は、特定のユーザーまたは PUBLIC に対して認可することができます。PUBLIC を指定すると、特権はユーザー (権限 ID) の集合に対して認可されます。この集合は、該当の表またはビューに対して私的に認可された特権を持っていないユーザー (後で追加されるユーザーも含む) で構成されます。このことは、私的な認可に影響します。例えば、SELECT が PUBLIC に認可されている場合に、UPDATE が HERNANDZ に認可されると、この私的な認可により、HERNANDZ は SELECT 特権を持つのを妨げられます。

REVOKE ステートメントを使用して、以前に与えた特権 を取り消すことができます。 1 つの権限 ID からある特権を取り消すと、すべての権限 ID によって認可されているその特権は取り消されます。ある権限 ID からある特権を取り消したとき、その権限 ID によって同じ特権が他の権限 ID に対して認可されているとしても、その特権がそれらの他の権限 ID から取り消されることはありません。

行の許可および列マスク

行の許可 は、特定の表の行のアクセス制御ルールを表します。行の許可は、ユーザーがいずれの行に対してアクセス権限を持つのかを記述する検索条件の形式をしています。行の許可は、表特権 (SELECT や INSERT など) を検査した後で適用されます。

列マスク は、表の特定の列のアクセス制御ルールを表します。列マスクは、ユーザーがいずれの列値に対してアクセス権限を持つのかを記述する CASE 式の形式をしています。列マスクは、表特権 (SELECT や INSERT など) を検査した後で適用されます。

データベース・セキュリティー管理者権限の保有者のみが、 CREATE MASK、CREATE PERMISSION、ALTER MASK、ALTER PERMISSION、および DROP ステートメントを使用して、行の許可および列マスクを、作成、変更、および除去することができます。アクセス権またはマスクの定義では、他のオブジェク

トを参照できます。データベース・セキュリティー管理者権限の保有者は、それらのオブジェクトを参照するために、行の許可または列マスクの定義内に追加の権限 (表からの取り出しのための SELECT 権限、ユーザー定義関数を呼び出すための EXECUTE 権限など) を必要としません。単一の表に対して、複数の行の許可および列マスクを作成できます。表内の各列に対して作成できる列マスクは 1 つのみです。行の許可や列マスクを作成できるのは、行または列のアクセス制御が表に実施されるまでです。行の許可および列マスクの定義は、Db2 カタログに保管されます。ただし、このアクセス権およびマスクは、表に対して ACTIVATE ROW ACCESS CONTROL 文節を持つ ALTER TABLE ステートメントによって行のアクセス制御が実施されるか、ACTIVATE COLUMN ACCESS CONTROL 文節を持つ ALTER TABLE ステートメントによって列のアクセス制御が実施されて初めて有効になります。

ALTER TABLE ステートメントを使用して、表の行アクセス制御を明示的に活動化すると、表に対するアクセスをすべて防止する、表のデフォルトの行の許可が暗黙で作成されます。表に対する行アクセス制御が活動化された後で、この表がデータ変更ステートメントで明示的に参照されており、この表に対して複数の行の許可が定義されている場合は、定義されている各行の許可の検索条件で論理 OR 演算子を使用して、行アクセス制御検索条件が派生されます。

ALTER TABLE ステートメントを使用して、表の列アクセス制御を明示的に活動化した場合に、表に対するアクセスは制限されません。ただし、SQL ステートメントでこの表が参照されている場合は、この表に対して作成されているすべての列マスクを適用して、照会の出力で参照されている列値をマスクしたり、データ変更ステートメントで使用される列値が決定されたりします。

行の許可または列マスクが適用されている表を参照する SQL ステートメントの権限 ID は、それらの行の許可または列マスクの定義内で指定されているオブジェクトを参照する権限を必要としません。

所有権

オブジェクトを作成するときには、ステートメントの権限 ID に、指定したスキーマ内で暗黙的にまたは明示的にオブジェクトを作成する特権が必要です。以下のいずれかの場合に、ステートメントの権限 ID はスキーマ内でオブジェクトを作成する特権を持ちます。

- スキーマの所有者である。
- 所有者に、スキーマに対する CREATEIN 特権がある。

オブジェクトを作成すると、1 つの権限 ID にそのオブジェクトの所有権 が割り当てられます。所有権を持つユーザーは、オブジェクトを完全に管理することができます (オブジェクトを除去する特権も持ちます)。オブジェクトに対する特権は、オブジェクトの所有者が付与することも、取り消すこともできます。この場合、その所有者は、その特権を必要とする操作を一時的に行うことができなくなります。ただし、所有者なので、常に、その特権を自分自身に復権することができます。

オブジェクトを作成したときに、所有者は次のようになります。

- SQL 名を指定した場合は、オブジェクトの所有者 は、作成したオブジェクトが入れられるスキーマと同じ名前のユーザー・プロファイルが存在していれば、そ

のユーザー・プロファイルです。その他の場合は、オブジェクトの所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

- システム名を指定した場合は、オブジェクトの所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

SQL オブジェクトの *PUBLIC に認可される権限は、オブジェクトを作成した時に使用した命名規則に依存します。 *SYS 命名規則を使用している場合には、*PUBLIC はそのオブジェクトが作成されたライブラリーの作成権限 ((CRTAUT)) を獲得します。 *SQL 命名規則を使用した場合は、*PUBLIC は *EXCLUDE 権限を獲得します。

本書の権限の項では、オブジェクトの所有者が、その作成以降にオブジェクトからどのような特権も取り消していないことを前提にしています。オブジェクトがビューの場合には、そのビューの所有者が、直接、または間接に従属する表やビューからシステム権限の *READ を取り消していないことを前提にしています。所有者は、そのビューの定義で参照されている表やビューのすべてに対してシステム権限の *READ を持ち、またあるビューが参照されている場合には、そのビューの定義で参照されている表やビューのすべてに対してシステム権限の *READ を持ちます。以下同様です。権限および特権について詳しくは、「機密保護解説書」を参照してください。

カタログ

データベース・マネージャーは、データベース内のオブジェクトに関する情報が入っている一組の表を維持管理しています。これらの表とビューをまとめてカタログと呼びます。カタログ表 には、システムに存在する表、ビュー、索引、パッケージ、および制約などのオブジェクトについての情報が入っています。

カタログ内の表およびビューは、データベースの他の表およびビューに似ています。カタログ表またはビューに対する SELECT 特権を持つユーザーであれば、カタログ表またはビュー内のデータを読み取ることができます。しかし、ユーザーはカタログ表またはビューを直接変更することはできません。データベース・マネージャーは、常に、データベース内のオブジェクトの正確な記述がカタログに入っているように管理します。

データベース・マネージャーは、他の IBM SQL プロダクトのカタログ・ビューとの高い整合性を備えた一組のビューと、ANSI および ISO 規格のカタログ・ビュー (規格では情報スキーマ と呼ばれる) との互換性を備えた一組のカタログ・ビューを提供します。

スキーマを CREATE SCHEMA ステートメントを使用して作成した場合、スキーマ内のオブジェクトに関する情報だけを含むビューもスキーマに含まれます。

カタログの表およびビューの詳細については、 1913 ページの『付録 F. Db2 for i のカタログ・ビュー』を参照してください。

アプリケーション・プロセス、並行性、およびリカバリー

SQL プログラムはすべてがアプリケーション・プロセス の一環として実行されます。 IBM i オペレーティング・システムでは、アプリケーション・プロセスのことをジョブといいます。 ODBC、JDBC、OLE DB、.NET、および DRDA の場合は、使用しているジョブが終了していなくて再使用可能であっても、接続が終了した時点でアプリケーション・プロセスは終了します。

アプリケーション・プロセスは、1 つ以上の活動化グループから成り立っています。活動化グループには、それぞれに 1 つ以上のプログラムの実行が含まれます。プログラムの実行は、非デフォルトの活動化グループ、またはデフォルトの活動化グループのもとで行われます。ILE コンパイラーにより作成されたプログラムを除き、すべてのプログラムはデフォルトの活動化グループのもとで実行されます。例えば、LANGUAGE JAVA 外部関数は、デフォルトの活動化グループのもとで実行されます。

活動化グループの詳細については、「ILE 概念」  を参照してください。

コミットメント制御を使用するアプリケーション・プロセスは、その実行に 1 つ以上のコミットメント定義を使用することができます。コミットメント定義を使用すると、コミットメント制御を活動化グループ・レベルまたはジョブ・レベルの範囲で行うことができます。コミットメント制御を使用する活動化グループは、一時点で、ただ 1 つのコミットメント定義に関連付けられます。

コミットメント定義を明示的に開始するには、コミットメント制御開始 (STRCMTCTL) コマンドを使用します。まだ開始されていないコミットメント定義の場合は、COMMIT(*NONE) 以外の分離レベルのもとで最初に SQL ステートメントが実行される時点で、暗黙に開始されます。1 つのジョブ・コミットメント定義を複数の活動化グループで共用することができます。

26 ページの図 3 は、アプリケーション・プロセス、そのアプリケーション・プロセス内の活動化グループ、およびコミットメント定義の関係を示しています。活動化グループの A と B は、その活動化グループを有効範囲とするコミットメント制御を伴って実行されます。これらの活動化グループは、それぞれ独自のコミットメント定義を持っています。活動化グループ C の実行はどのようなコミットメント制御も伴いません。この活動化グループには、コミットメント定義がありません。

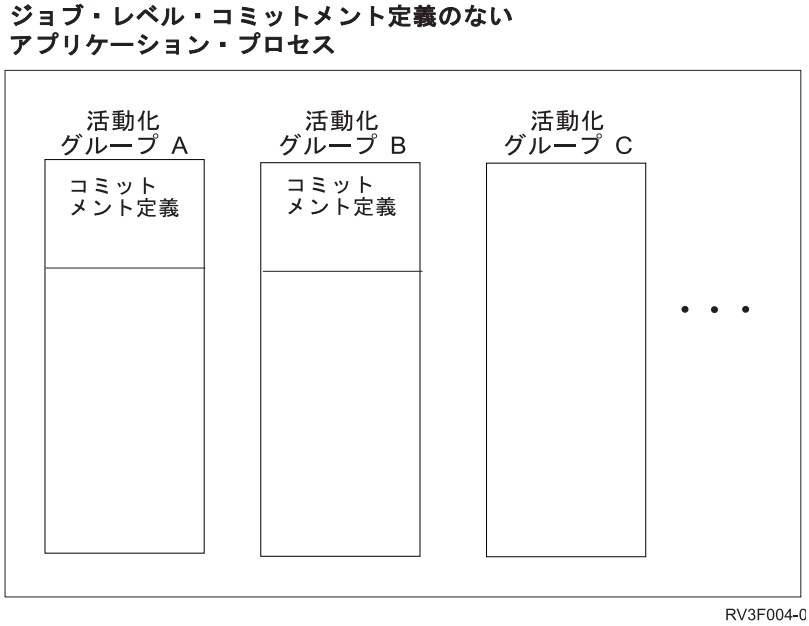


図 3. ジョブのコミットメント定義のない活動化グループ

27 ページの図 4 は、アプリケーション・プロセス、そのアプリケーション・プロセス内の活動化グループ、およびコミットメント定義を示しています。活動化グループの中には、ジョブのコミットメント定義によって実行されているものがあります。活動化グループの A と B は、ジョブのコミットメント定義のもとで実行されています。コミットメント制御は同じコミットメント定義によって行われるので、活動化グループの A または B におけるコミットまたはロールバック操作は、両方の活動化グループが影響を与えます。この例の活動化グループ C は、別のコミットメント定義を持っています。この活動化グループで行われるコミットおよびロールバック操作は、C における操作にのみ影響します。

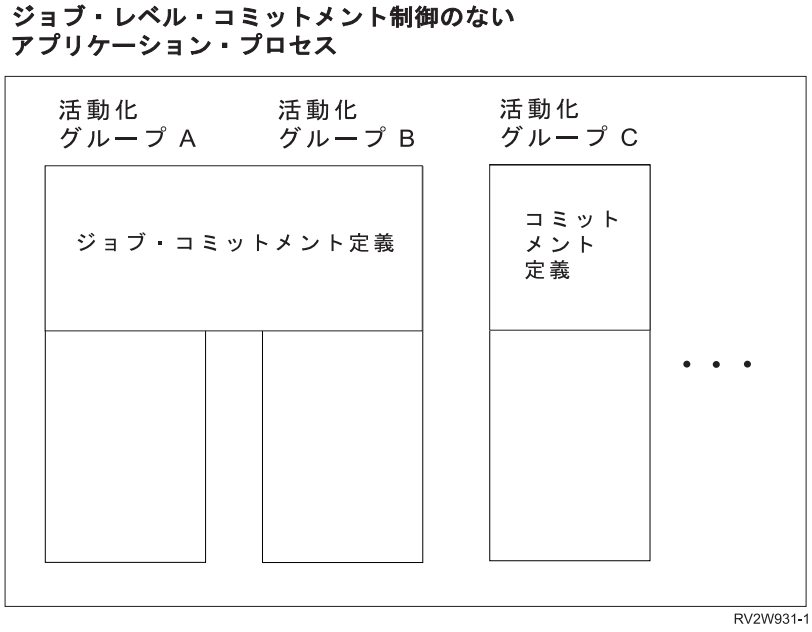


図 4. ジョブのコミットメント定義のある活動化グループ

コミットメント定義についての詳細は、コミットメント制御のトピックを参照してください。

ロック、コミット、およびロールバック

別のコミットメント定義を使用するアプリケーション・プロセスおよび活動化グループは、同時に同じデータに対するアクセスを要求することができます。このような状況でデータの保全性を維持するために、ロック が使用されます。ロックによって、2 つのアプリケーション・プロセスが同じデータの行を同時に更新するような事態を防止できます。

データベース・マネージャーは、異なるコミットメント定義を使用する活動化グループからは検出されないある活動化グループによるコミットされていない変更を保持するために、ロックを獲得します。オブジェクトのロックおよびその他のリソースは、活動化グループに割り振られます。行のロックは、コミットメント定義に割り振られます。

デフォルトの活動化グループ以外の活動化グループが正常に 終了すると、データベース・マネージャーは、その活動化グループによるロックをすべて解除します。ユーザーは、ロックをより迅速に解除することを明示的に要求することもできます。この操作をコミット と呼びます。コミット後もオープンのままのカーソルと関連するオブジェクトのロックは、解除されません。

データベース・マネージャーのリカバリー機能には、コミットメント定義で行われた変更がまだコミットされていない場合に、その変更を取り消す方法が用意されています。データベース・マネージャーは以下のような場合に、コミットされていない変更を暗黙にバックアウトすることがあります。

- アプリケーション・プロセスが終了すると、デフォルトの活動化グループに関連するコミットメント定義のもとで行われたすべての変更はバックアウトされま

す。デフォルトの活動化グループ以外の活動化グループが、異常 終了すると、その活動化グループに関連するコミットメント定義のもとで行われた変更はすべてバックアウトされます。

- 分散作業単位を使用し、リモート・システムで変更をコミットしようとした時点で障害が起こると、リモート接続に関連するコミットメント定義のもとで行われた変更はすべてバックアウトされます。
- 分散作業単位を使用し、リモート・システムでの障害によりリモート・システムからバックアウトの要求を受け取った場合には、リモート接続に関連するコミットメント定義のもとで行われた変更はすべてバックアウトされます。

ユーザーは、データベースの変更のバックアウトを明示的に要求することができます。この操作をロールバック と呼びます。

活動化グループに代わってデータベース・マネージャーが獲得したロックは、その作業単位が終了するまで保持されます。LOCK TABLE ステートメントによって明示的に獲得されたロックは、COMMIT HOLD または ROLLBACK HOLD を使用して作業単位を終了させると、作業単位の終了後も保持することができます。

カーソルによって、カーソルの置かれている行が暗黙のうちにロックされる場合があります。このロックによって、次のような事態が防止されます。

- 別のコミットメント定義に関連した、他のカーソルによる同一行のロック。
- 別のコミットメント定義に関連した、DELETE または UPDATE ステートメントによる同一行のロック。

作業単位

作業単位 (トランザクション、論理作業単位、またはリカバリー単位 とも呼ばれる) は、リカバリー可能な一連の操作を指します。それぞれのコミットメント定義には、1 つ以上の作業単位の実行が含まれます。どのような時点をとっても、1 つのコミットメント定義には 1 つの作業単位があります。

作業単位は、コミットメント定義が開始された時点、または前の作業単位がコミットやロールバック操作によって終了した時点で開始されます。作業単位は、コミット操作、ロールバック操作、または活動化グループ終了のいずれかによって終了します。コミットまたはロールバック操作は、そのコミットまたはロールバックによって終了する作業単位内で行われたデータベースの変更にのみ影響します。変更のコミットが済まない間は、分離レベル COMMIT(*CS)、COMMIT(*RS)、および COMMIT(*RR) のもとで実行されている異なるコミットメント定義を使用する他の活動化グループは、変更を認知することができません。コミットが行われるまでは、変更を取り消すことができます。変更のコミットが済むと、異なるコミットメント定義で実行されている活動化グループからその変更にアクセスできるようになり、取り消しは不能になります。

1 つの作業単位の開始と終了によって、活動化グループ内の整合点が定義されます。例えば、銀行業務のトランザクションで、ある口座から別の口座に送金を行う場合があります。このようなトランザクションでは、最初の口座から差し引いた金額を、2 番目の口座に加算する必要があります。最初の口座から金額を差し引いたステップの後では、データに整合性はありません。2 番目の口座に金額を加算して初めて、再度整合性が確立されます。この両方のステップが完了した時点で、コミ

ット操作を使用して作業単位を終了させることができます。コミット操作が終わると、異なるコミットメント定義を使用する活動化グループが変更を使用できるようになります。

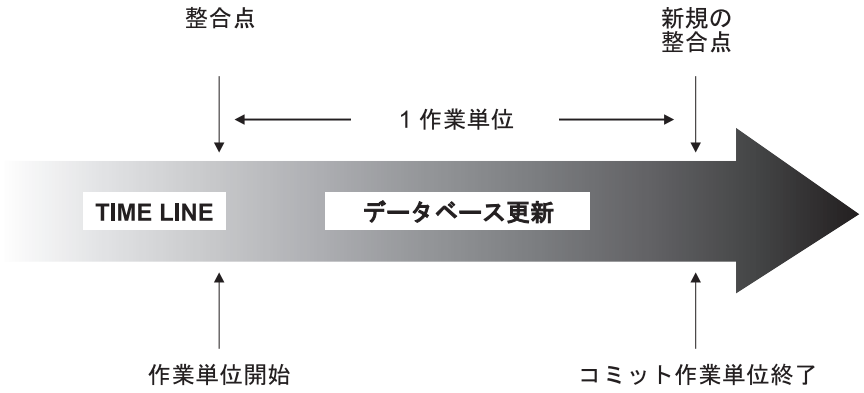


図 *5. COMMIT (*コミット*)* ステートメントと作業単位

作業のロールバック

データベース・マネージャーは、1 つの作業単位内で行われたすべての変更をバックアウトすることも、選択された変更のみをバックアップすることもできます。ただし、整合点が達成されるのはすべての変更をバックアウトした場合だけです。

すべての変更のロールバック

TO SAVEPOINT 文節を伴わない SQL ROLLBACK ステートメントを使用すると、フル・ロールバック操作が行われます。このようなロールバック操作が正常に実行されると、データベース・マネージャーは、コミットされていない変更をバックアウトして、作業単位の開始時点で存在していたものと見なされるデータの整合性を復元します。つまり、データベース・マネージャーは、以下の図に示すように、作業を取り消します。

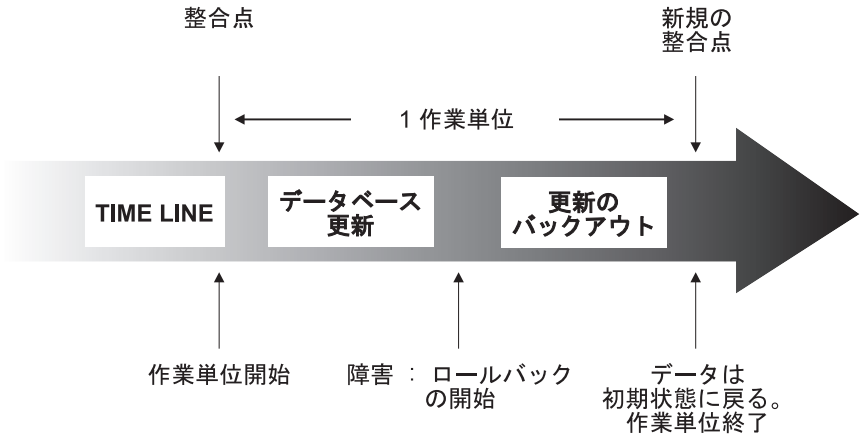


図 *6. ROLLBACK (*ロールバック*)* ステートメントと作業単位

セーブポイントを使用して選択した変更のロールバック

セーブポイント *(savepoint)* は、1 つの作業単位の中の特定時点におけるデータの状態を表します。アプリケーション・プロセスは、作業単位内にセーブポイントを設定しておき、ロジックの指示に従って、特定のセーブポイントの設定後に行われた変更のみをロールバックすることができます。例えば、旅行予約トランザクションには、航空券予約とホテルの予約が含まれることがあります。ここで、航空券は予約できたが、ホテルが予約できなかったという場合、アプリケーション・プロセスで航空券予約だけを取り消して、航空券予約より前にトランザクション内で行われたデータベース変更は取り消さないようにしたい場合があります。このような場合に、SQL プログラムは、SQL SAVEPOINT ステートメントを使用してセーブポイントを設定し、TO SAVEPOINT 文節を伴う SQL ROLLBACK ステートメントを使用して特定のセーブポイントまたは最後に設定されたセーブポイントまで変更を取り消し、そして、RELEASE SAVEPOINT ステートメントを使用してセーブポイントを削除することができます。

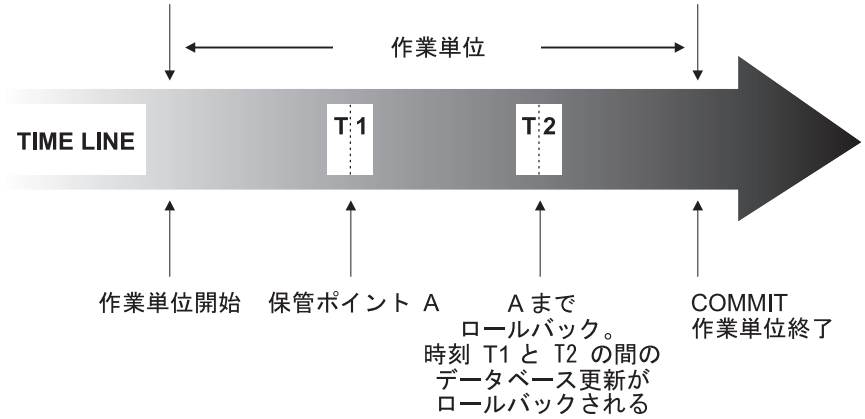


図 *7. ROLLBACK* (ロールバック) ステートメントおよび *SAVEPOINT* ステートメントと作業単位

スレッド

IBM i オペレーティング・システムでは、アプリケーション・プロセスも 1 つ以上のスレッドから構成することができます。デフォルトでは、スレッドは同じコミットメント定義を共用し、そのジョブにおける他のスレッドのようにロックします。このため、1 つのスレッドがコミットあるいはロールバックする場合、そのスレッドはすべてのスレッドが行ったすべての変更をコミットあるいはロールバックすることができるように、それぞれのスレッドは同じ作業単位で機能することができます。このタイプの処理は、複数のスレッドで協調して単一のタスクを並列に実行する場合に便利です。

その他のケースとしては、あるスレッドがジョブ内の他のスレッドとは独立して、変更を行うような場合に便利です。この場合、そのスレッドではコミットメント定義を共用したり、他のスレッドとともにロックする必要はありません。さらに、複数データベースの接続とトランザクション情報について、よりすぐれた、きめ細かい制御を行うために、ジョブは SQL サーバー・モードを使用することができま

す。典型的なマルチスレッドのジョブでは、このような制御が必要になる場合があります。このタイプの処理方法は、いくつかあります。

- スレッドで実行するプログラムは、必ず、別の活動化グループを使用するようにします (ACTGRP(*NEW) を使用しないように注意します)。
- 最初の SQL ステートメントを出す前から、ジョブは、必ず、SQL サーバー・モードで実行しているようにします。SQL サーバー・モードは、アプリケーションでデータ・アクセスが生じる前に以下のいずれかのメカニズムを使用することによって、ジョブに対して活動化することができます。
  - データ・アクセスが行われる前に、ODBC API、SQLSetEnvAttr() を使用して、SQL_ATTR_SERVER_MODE 属性を SQL_TRUE に設定する。
  - データ・アクセスが行われる前に、Change Job API、QWTCHGJB() を使用して、'Server mode for Structured Query Language (SQL のサーバー・モード)' キーを設定する。
  - JAVA を使用し、JDBC を介してデータベースをアクセスする。JDBC は、自動的にサーバー・モードを使用し、必要な JDBC の意味体系を維持する。

SQL サーバー・モードが確立されると、すべての SQL ステートメントは、要求を取り扱う独立したサーバー・ジョブに渡されます。SQL 動作に関するサーバー・モード動作には、以下のものが含まれます。

- 組み込み SQL の場合、ジョブの各スレッドはデータベースに対する唯一の接続を (したがって、それ自体のコミット可能なトランザクションも) 暗黙的に取得します。
- ODBC/CLI、JDBC、OLE DB、および .NET の場合、それぞれの接続はデータベースに対する独立型の接続を表しており、別々のエンティティーとしてコミット可能であり、使用することができます。

詳しくは、SQL 呼び出しレベル・インターフェース (ODBC) を参照してください。

以下の SQL サポートは、スレッド・セーフではありません。

- DRDA 経由のリモート・アクセス
- ALTER FUNCTION
- ALTER PROCEDURE
- ALTER SEQUENCE
- ALTER TABLE
- COMMENT
- CREATE ALIAS
- CREATE FUNCTION
- CREATE INDEX
- CREATE PROCEDURE
- CREATE SCHEMA
- CREATE SEQUENCE
- CREATE TABLE
- CREATE TRIGGER

- CREATE TYPE
- CREATE VARIABLE
- CREATE VIEW
- DECLARE GLOBAL TEMPORARY TABLE
- DROP
- GRANT
- LABEL
- REFRESH TABLE
- RENAME
- REVOKE

詳細については、IBM i Information Center のプログラミング・トピックのマルチスレッド・アプリケーションを参照してください。

分離レベル

SQL ステートメントの実行中に使用する分離レベル によって、活動化グループが並行して実行される他の活動化グループから分離される度合いが決定されます。

したがって、活動化グループ P が SQL ステートメントを実行すると、分離レベルによって次のことが決定されます。

- P によって検索される行、および P によって行われるデータベースの変更が、並行して実行される他の活動化グループで使用できる度合い。
- 並行して実行される活動化グループによって行われるデータベースの変更が、P に影響を及ぼす度合い。

分離レベルは、DELETE、INSERT、SELECT INTO、UPDATE、または選択ステートメントに対して明示的に指定できます。分離レベルを明示的に指定しない場合、SQL ステートメントを実行したときには デフォルト分離レベル という分離レベルが使用されます。

Db2 for i では、デフォルト分離レベル を指定するために、いくつかの方法を提供しています。

表 2. デフォルトの分離レベル・インターフェース

SQL インターフェース	指定
組み込み SQL	「SQL プログラム作成」(CRTSQLxxx) コマンドの COMMIT パラメーター。 SET OPTION ステートメントを使用して、COMMIT 値を設定することもできます。 (CRTSQLxxx コマンドについて詳しくは、「組み込み SQL プログラミング」を参照してください。)
SQL 関数と SQL プロシージャー	SQL 関数と SQL プロシージャーに組み込んだ静的 SQL ステートメントは、その SQL 関数または SQL プロシージャーの作成時に有効だった分離レベルを使用します。SET OPTION ステートメント (COMMIT) を使用して分離レベルを設定することもできます。

表 2. デフォルトの分離レベル・インターフェース (続き)

SQL インターフェース	指定
SQL ステートメント実行	「SQL ステートメントの実行」(RUNSQLSTM) コマンドの COMMIT パラメーター。 (RUNSQLSTM コマンドの詳細については、「SQL プログラミング」を参照。)
SET TRANSACTION SQL ステートメント	作業単位内でデフォルトの分離レベルをオーバーライドします。その作業単位が終了すると、分離レベルはその作業単位の開始時点での値に戻ります。このステートメントは、その作業単位内の静的および動的 SQL ステートメントに関する分離レベルの他の指定もすべてオーバーライドします。 (SET TRANSACTION ステートメントについて詳しくは、 1733 ページの『SET TRANSACTION』を参照してください。)
isolation-clause	SELECT、SELECT INTO、INSERT、UPDATE、DELETE、および DECLARE CURSOR ステートメントの *isolation-clause* は、特定のステートメントまたはカーソルについてデフォルトの分離レベルをオーバーライドします。分離レベルが有効なのは、*isolation-clause* を含むステートメントを実行する場合のみであり、現行の作業単位内で保留中の変更に対しては無効です。 (*isolation-clause* について詳しくは、 859 ページの『ISOLATION 文節』を参照してください。)
サーバー上の呼び出しレベル・インターフェース (CLI)	SQL_ATTR_COMMIT または SQL_TXN_ISOLATION 環境変数または接続オプション (CLI について詳しくは、「SQL 呼び出しレベル・インターフェース (ODBC)」を参照してください。)
IBM IBM Developer Kit for Java を使用したサーバーの JDBC または SQLJ	トランザクション分離プロパティー・オブジェクト (JDBC および SQLJ について詳しくは、「IBM Developer Kit for Java」を参照してください。)
IBM i Access Family ODBC ドライバーを使用したクライアントの ODBC	ODBC セットアップでのコミット・モード (ODBC の詳細については、「IBM i Access」を参照。)
IBM Toolbox for Java を使用したクライアントの JDBC	JDBC セットアップでの分離レベル (JDBC の詳細については、「IBM i Access」を参照。) (IBM Toolbox for Java について詳しくは、「IBM Toolbox for Java」を参照してください。)
IBM i Access Family OLE DB Provider を使用したクライアントの OLE DB	IsolationLevel 接続オブジェクト・プロパティー (OLE DB の詳細については、「IBM i Access」を参照。)
IBM i Access FamilyADO .NET プロバイダーを使用しているクライアントの ADO .NET	接続オブジェクト・プロパティー内の IsolationLevel (ADO .NET の詳細については、「IBM i Access」を参照。)

これらの分離レベルは、該当するデータを自動的にロックすることによってサポートされます。ロックのタイプに応じて、異なるコミットメント定義を使用して、並行して実行される活動化グループによるデータのアクセスが制約、または禁止されます。それぞれのデータベース・マネージャーでは、少なくとも次に示す 2 つのロックのタイプをサポートしています。

共用　異なるコミットメント定義を使用する、並行して実行される活動化グループを、データに対する読み取り専用操作に限定します。

排他　異なるコミットメント定義を使用する、並行して実行される活動化グループによるデータの更新および削除を防止します。並行して実行される活動化グループが、COMMIT(*RS)、COMMIT(*CS)、または COMMIT(*RR) を実行している異なるコミットメント定義を使用する場合は、それによるデータの読み取りを防止します。並行して実行される活動化グループが、COMMIT(*UR) または COMMIT(*NC) を実行している異なるコミットメント定義を使用する場合は、それによるデータの読み取りを許します。

分離レベルに関する以下の説明は、行単位で行われるデータのロックについて述べています。個々のインプリメンテーションでは、基本表の行よりも大きな物理単位でデータをロックすることができる場合があります。ただし、論理的には、ロックはすべての製品において基本表の行レベルで行われます。同様に、データベース・マネージャーでは、ロックをより上位のレベルにまで拡大することができます。活動化グループには、少なくとも要求される最低限のロック・レベルが保証されます。

レコード・ロック期間について詳しくは、「SQL プログラミング」トピック集にあるトピックコミットメント制御の説明および表を参照してください。

Db2 for i では、5 つの分離レベルをサポートします。コミット不可以外のすべての分離レベルで、データベース・マネージャーは、挿入、更新、または削除されるすべての行に排他ロックします。これにより、ある作業単位の過程で変更された行は、その作業単位が完了するまで、異なるコミットメント定義を使用する他の活動化グループにより変更されることはありません。

反復可能読み取り

反復可能読み取り (RR) 分離レベルを使用すると、次のようになります。

- ある作業単位の過程で読み取られた行は、その作業単位が完了するまでは、別のコミットメント定義を使用する他の活動化グループによって変更されることはない。[5]
- 別のコミットメント定義を使用する他の活動化グループによって変更された行 (あるいは現在 UPDATE のための行ロックでロックされている行) は、その行がコミットされるまで読み取ることはできない。

分離レベル RR で実行されている活動化グループは、任意の排他ロックに加えて、少なくともその活動化グループが読み取ったすべての行に共用ロックします。さらに、その活動化グループが、異なるコミットメント定義を使用する、並行して実行される活動化グループの影響から完全に分離されるようにロックされます。

5. **WITH HOLD** カーソルの場合は、この規則は行が実際に読み取られたときに適用されます。読み取り専用 **WITH HOLD** カーソルの場合は、事前の作業単位で行が既に実際に読み取られている場合があります。

SQL 2003 コア標準では、反復可能読み取りは逐次化可能と呼ばれています。

Db2 for i では、COMMIT(*RR) によって反復可能読み取りをサポートします。反復可能読み取り分離レベルは、読み取りまたは更新の対象となる行が含まれている表をロックすることによってサポートされます。

読み取り固定

分離レベル RR と同様に、分離レベル読み取り固定 (RS) を使用すると、次のようになります。

- ある作業単位の過程で読み取られた行は、その作業単位が完了するまでは、別のコミットメント定義を使用する他の活動化グループにより変更されることはない。[6]
- 別のコミットメント定義を使用する他の活動化グループによって変更された行 (あるいは現在 UPDATE のための行ロックでロックされている行) は、その行がコミットされるまで読み取ることはできない。

RR とは異なり、分離レベル RS では、同時に実行されている別のコミットメント定義を使用する活動化グループの影響から、完全には分離されません。分離レベル RS では、活動化グループから同じ照会を複数回出すと追加の行を見ることがあります。このような付加行は幻像読み取り行 と呼ばれます。

例えば、単独読み取り行が発生するのは次のような場合です。

1. 活動化グループ P1 で、何らかの検索条件を満たす行 *n* の集合を読み取る。
2. 次に、活動化グループ P2 で上記の検索条件を満たす 1 つ以上の行を挿入し、その挿入をコミットする。
3. ここで、P1 から前回と同じ検索条件で行の集合を読み取ると、当初の行と P2 によって挿入された行の両方が入手される。

分離レベル RS で実行されている活動化グループは、それ自体が読み取るすべての行に、たとえ排他ロックされている場合でも、それに加えて少なくとも共用ロックします。

SQL 2003 コア標準では、読み取り固定は反復可能読み取りと呼ばれています。

Db2 for i では、COMMIT(*ALL) または COMMIT(*RS) によって読み取り固定をサポートします。

カーソル固定

分離レベル RR および RS と同様に、分離レベルカーソル固定 (CS) では、別のコミットメント定義を使用して他の活動化グループが変更した行 (あるいは現在 UPDATE 行ロックでロックされている行) は、コミットされるまで読み取ることはできません。ただし、RR および RS の場合とは異なり、分離レベル CS で保証されるのは、すべての更新可能なカーソルの現在行が、異なるコミットメント定義を使用する他の活動化グループによって変更されることはないということだけです。したがって、ある作業単位の過程で読み取られた行を、別のコミットメント定義を

6. **WITH HOLD** カーソルの場合は、この規則は行が実際に読み取られたときに適用されます。読み取り専用 **WITH HOLD** カーソルの場合は、事前の作業単位で行が既に実際に読み取られている場合があります。

使用する別の活動化グループにより変更することができます。分離レベル CS で実行されている活動化グループには、任意の排他ロックに加えて、すべてのカーソルの現行行に対する共用ロックを獲得することもできます。

SQL 2003 コア標準では、カーソル固定はコミット読み取りと呼ばれています。

Db2 for i では、COMMIT(*CS) によってカーソル固定をサポートします。

非コミット読み取り

SELECT INTO、読み取り専用カーソル付きの FETCH、副照会、または INSERT ステートメントで使用される全選択の場合は、分離レベル非コミット読み取り (UR) で以下のことが可能になります。

- ある作業単位の過程で読み取られた行は、別のコミットメント定義の下で実行されている他の活動化グループにより変更できる。
- 別のコミットメント定義の下で実行されている他の活動化グループによって変更された行 (あるいは現在 UPDATE 行ロックでロックされている行) は、いずれも、変更のコミットメントが行われていない場合でも読み取ることができる。

それ以外の操作の場合は、分離レベル CS の規則が適用されます。

SQL 2003 コア標準では、非コミット読み取りは読み取り非コミットと呼ばれています。

Db2 for i では、COMMIT(*CHG) または COMMIT(*UR) によって非コミット読み取りをサポートします。

コミット不可

すべての操作に関して、以下を除いて、分離レベル UR の規則がコミット不可 (NC) に適用されます。

- SQL ステートメントで、コミットおよびロールバックの操作は無効です。カーソルはクローズされず、また LOCK TABLE のロックは解除されません。ただし、解除保留状態の接続は終了します。
- 変更はいずれも、正常に行われた各変更操作の終了時に効果的にコミットされ、異なるコミットメント定義を使用する他のアプリケーション・グループによるアクセスまたは変更をただちに行うことができます。

Db2 for i では、COMMIT(*NONE) または COMMIT(*NC) によってコミット不可をサポートします。

注: (分散アプリケーションに関する注意) 要求した分離レベルをアプリケーション・サーバーがサポートしていない場合、分離レベルは、サポートされている次に高い分離レベルまで拡大されます。例えば、アプリケーション・サーバーが RS をサポートしていない場合は、RR 分離レベルが使用されます。

分離レベルの比較

次の表は、分離レベルに関する情報を要約したものです。

	NC	UR	CS	RS	RR
アプリケーションが、他のアプリケーション・プロセスによって実行されたコミットされていない変更を表示できるか。	可	可	不可 [2]	不可 [2]	不可
アプリケーションが、他のアプリケーション・プロセスによって実行されたコミットされていない変更を更新できるか。	不可	不可	不可	不可	不可
ステートメントの再実行は、他のアプリケーション・プロセスによる影響を受けるか。下記の現象 *P3 (幻像)* を参照してください。	可	可	可	可	不可
「更新された」行は、他のアプリケーション・プロセスで更新可能か。	可	不可	不可	不可	不可
「更新された」行は、 UR と NC 以外の分離レベルで実行している他のアプリケーション・プロセスで読み取り可能か。	可	不可	不可	不可	不可
「更新された」行は、 UR と NC の分離レベルで実行している他のアプリケーション・プロセスで読み取り可能か。	可	可	可	可	可
「アクセスされた」行は、他のアプリケーション・プロセスで更新可能か。 RS の場合、「アクセスされた行」とは一般に選択された行のことを指します。 RR の場合は、製品固有の資料を参照してください。下記の現象 *P2 (反復不能読み取り)* を参照してください。	可	可	可	不可	不可
「アクセスされた」行は、他のアプリケーション・プロセスで読み取り可能か。	可	可	可	可	可
「現在」行は他のアプリケーション・プロセスで更新または削除可能か。下記の現象 *P1 (ダーティー読み取り)* を参照してください。	注 1 を参照	注 1 を参照	注 1 を参照	不可	不可

注 **1:** 可能かどうかは、「現在」行に置かれているカーソルが更新可能かどうかによって決まります。

- カーソルが更新可能の場合は、他のアプリケーション・プロセスで現在行を更新または削除することはできません。
- カーソルが更新不能の場合は、次のようになります。
  - 分離レベル UR または NC では、他のアプリケーション・プロセスで現在行を更新または削除することができます。
  - 分離レベル CS では、特定の環境で現在行を更新することができます。

注 **2:** USE CURRENTLY COMMITTED 節を使用してください。さらに、表スキャンを使用して照会が実装される場合は、QAQQINI オプション CONCURRENT_ACCESS_BEHAVIOR を値 *STRICTSCAN で使用してください。

	NC	UR	CS	RS	RR

現象の例:

P1 ダーティー読み取り。作業単位 UW1 が行を変更します。作業単位 UW2 は、UW1 が COMMIT を実行する前にその行を読み取ります。次に、UW1 は ROLLBACK を実行します。こうして、UW2 は存在しない行を読み取ることになります。

P2 反復不能読み取り。作業単位 UW1 が行を読み取ります。作業単位 UW2 はその行を変更して、COMMIT を実行します。次に、UW1 はその行をもう一度読み取って、変更されたデータ値を取得します。

P3 幻像 。作業単位 UW1 が、ある検索条件を満たしている *n* 個の行を読み取るとします。次に、作業単位 UW2 が検索条件を満たす 1 つ以上の行を挿入します。すると、作業単位 UW1 は、同一の検索条件を使って最初に行った読み取りを繰り返したのに、元の行に挿入された行が加えられて結果を取得します。

記憶構造

IBM i プロダクトは、オブジェクト・ベースのシステムです。Db2 for i のすべてのデータベース・オブジェクト (表および索引など) は、IBM i オペレーティング・システムのオブジェクトです。単一レベルの記憶管理機能がデータベースのすべての記憶を管理しているため、データベース特有の記憶構造 (例えば、表スペース) は不要です。

分散表により、異なるデータベース区画にまたがって、データを置くことができます。含まれる区画は、表の作成または変更時に指定されるノード・グループによって決まります。ノード・グループは、1 つ以上の IBM i プロダクトのグループです。区分化されたマップは、それぞれのノード・グループに関連しています。区分化されたマップは、データベース・マネージャーが使用し、ノード・グループのどのシステムが所定のデータ行を格納するかを決めます。ノード・グループおよびデータ区分化についての詳細は、Db2 UDB for iSeries マルチ・システムを参照してください。

また、表には、外部ファイルに保管されたデータへのリンクを登録する列も含めることができます。これについてのメカニズムは、DataLink データ・タイプです。通常の表に記録された DataLink 値が、外部ファイル・サーバーに保管されたファイルを指しています。

ファイル・サーバー上の Db2 ファイル・マネージャーが、Db2 と共同で以下の任意選択の機能を提供します。

- 現在、Db2 にリンクしているファイルが削除または名前変更されないようにするための参照保全
- DataLink 列で適切な SQL 特権を持ったものだけが、その列にリンクしたファイルを読み取ることができるようにするためのセキュリティー

DataLinker は、次の 2 つの機能から成っています。

DataLink ファイル・マネージャー
Db2 にリンクした特定のファイル・サーバーのすべてのファイルを登録する。

DataLink フィルター
: ファイル・システム・コマンドをフィルターに掛け、登録されたファイルが削除または名前変更されないように確認する。任意選択として、コマンドをフィルターに掛け、適切なアクセス権限のあることを確認する。

文字変換

ストリング は、文字を表す一連のバイトです。 1 つのストリングの中では、すべての文字が共通のコード表示で表されます。これらの文字を別のコード表示に変換しなければならない場合があります。この変換プロセスは、文字変換 と呼ばれます。

SQL ステートメントをリモート側で実行するときに、文字変換が行われることがあります。[7]例えば、次の 2 つの場合を考えてみます。

- 変数の値が、アプリケーション・リクエスターから現行サーバーに送信される。
- 結果の列の値が、現行サーバーからアプリケーション・リクエスターに送信される。

上記のどちらの場合も、送信側と受信側のシステムでストリングの表現が異なる可能性があります。同一のシステムにおけるストリング操作でも、変換が行われる場合があります。

SQL ステートメントはストリングであるため、ステートメントが文字変換の影響を受けることに注意してください。

以下のリストは、文字変換の説明で使用される用語のいくつかを定義しています。

文字セット
: 定義された文字の集合。例えば、次のような文字セットを持つコード・ページがあります。
  - A から Z までのアクセントなしの文字 (26 文字)
  - a から z までのアクセントなしの文字 (26 文字)
  - 0 から 9 までの数字
  - . , : ; ? ( ) ' " / - _ & + = < >

コード・ページ
: コード・ポイントに対して文字を割り当てた集合。例えば、EBCDIC では、「A」にはコード・ポイント X'C1' が割り当てられ、「B」にはコード・ポイント X'C2' が割り当てられます。 1 つのコード・ページ内では、それぞれのコード・ポイントが特定の意味を 1 つだけ持ちます。

コード・ポイント
: コード・ページ内の文字を表す固有のビット・パターン。

コード化文字セット
: 文字セットを確立するとともに、セット内の文字とそのコード表示との間に 1 対 1 の関係を確立する明確な規則の集合。

7. 文字変換が必要な場合、変換は自動的に行われ、正常に変換された場合は、アプリケーションは変換が行われたことを認識しません。したがって、ステートメントの実行に関連するすべてのストリングが同一の方法で表現されている場合には、変換の知識は必要ありません。したがって、多くの読者の場合、文字変換の知識は必要ないはずです。

エンコード・スキーム
: 文字データを表現するために使用する規則の集合。例えば次のようなものがあります。

  - 1 バイト EBCDIC
  - 1 バイト ASCII
  - 2 バイト EBCDIC
  - 1 バイト ASCII および 2 バイト ASCII 混合の [8]
  - Unicode (UTF-8、UCS-2、および UTF-16 汎用コード化文字セット)。

置換文字
: 文字変換で、ソースのコード表示の文字に対応する文字が、ターゲットのコード表示に存在しない場合に、その文字に置き換わる固有の文字。

Unicode
: 書き記された文字やテキストのデータを国際的に交換できるように定められた、汎用コード化体系。 Unicode には、全世界で使用可能な文字セットの標準が規定されています。 16 ビット・エンコード方式が採用されており、65,000 を超える文字のコード・ポイント、およびさらに 100 万文字もエンコードを可能にする UTF-16 と呼ばれる拡張セットが提供されています。また Unicode は文字ごとに数値と名前を指定しているため、世界の文字言語に使用されるすべての文字をエンコードできるだけでなく、英字、漢字、および記号を等しく扱うことができます。扱える文字には、句読記号、数学記号、技術記号、幾何学形状、および飾り活字が含まれます。以下の 3 つのタイプのエンコード方式がサポートされています。

  - UTF-8: Unicode Transformation Format、8 ビット・エンコード方式。既存の ASCII ベースのシステムで簡単に利用できるように設計されたもの。 UTF-8 データは、文字データ・タイプで保管されます。 UTF-8 形式のデータの CCSID 値は 1208 です。

    UTF-8 文字の長さは、1、2、3、または 4 バイトのいずれかにすることもできます。 UTF-8 データ・ストリングには、サロゲートや合成文字を含め、 SBCS および DBCS データを任意に組み合わせて含めることができます。
  - UCS-2: 2 つのオクテットでコード化された汎用文字セット。 1 文字が 16 ビットで表現されることを意味します。 UCS-2 データは、グラフィック・データ・タイプで保管されます。 UCS-2 形式のデータの CCSID 値は 13488 です。

    UCS-2 は UTF-16 のサブセットです。 UTF-16 が合成文字やサロゲートをサポートしていることを除けば、UCS-2 と UTF-16 は同一です。 UCS-2 は UTF-16 を単純化したものであるため、 UTF-16 データに比べて UCS-2 データの方が操作性に優れています。[9]

8. UTF-8 Unicode データも混合データです。しかし本書では、混合 1 バイトおよび 2 バイト・データを指して「混合データ」と呼んでいます。

9. UCS-2 を使ってサロゲートや合成文字を表すこともできますが、それらの文字はその通りには認識されません。 16 ビットごとに 1 文字として認識されてしまいます。

- UTF-16: Unicode Transformation Format、16 ビット・エンコード方式。 100 万を超える文字のコード値を提供できるように設計されたもの。 UTF-16 データは、グラフィック・データ・タイプで保管されます。 UTF-16 形式のデータの CCSID 値は 1200 です。

  UTF-8 データと UTF-16 データにはともに合成文字 が含まれています。合成文字サポートにより、1 つ以上の文字を組み合わせて 1 文字にすることが可能です。データ・ストリングとして、1 文字目の後に、何百文字もの異なる非スペーシング・アクセント文字 (ウムラウト、アクサンなど) を続けることができます。複数の文字を組み合わせてできた文字は、既に文字セットに定義されている場合があります。その場合には、同じ文字に複数の表現方法があるということになります。例えば、UTF-16 では、*é* が X'00E9' (正規化された表現) または X'00650301' (正規化されていない合成文字表現) のいずれかで表現できます。

  同じ文字の複数の表現を等しく比較することはできないので、データベース内に両方の文字形式を保管するのは賢明ではありません。正規化 とは、合成文字のストリングを合成文字を含まない同等の文字で置き換える処理のことです。正規化が行われると、データに存在する特定の文字の表現形式は 1 つになります。例えば、UTF-16 では、X'00650301' (*é* の正規化されていない合成文字表現) が X'00E9' (*é* の正規化された表現) に変換されます。[10]

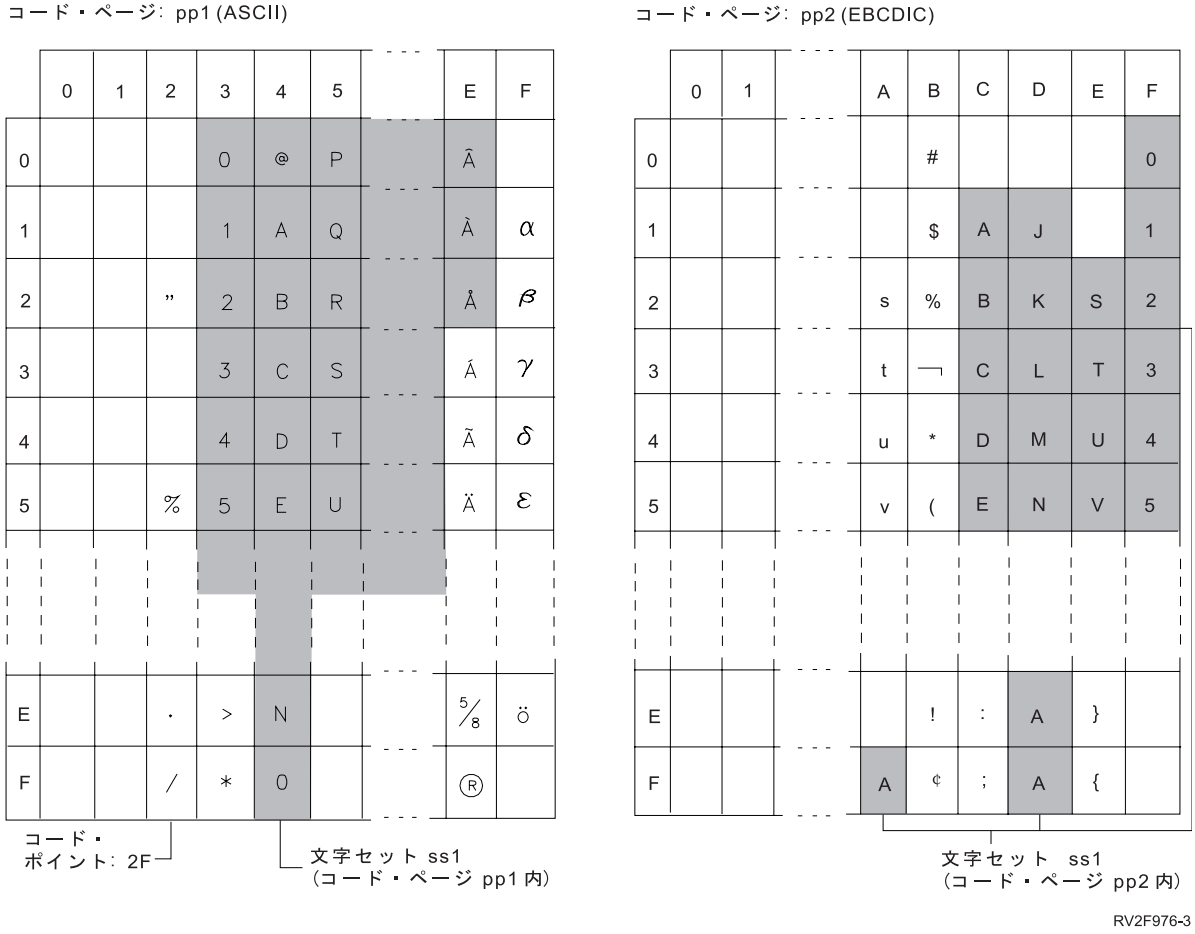
  述部の中で UTF-8 を適正に取り扱うために、正規化が行われることがあります。

  UTF-8 と UTF-16 の両方にはサロゲート と呼ばれる 4 バイト文字を含めることができます。サロゲートとは、2 バイト文字セットで利用できる文字数よりさらに 100 万多い文字数を当てられるようにした 4 バイト・シーケンスです。

文字セットとコード・ページ

次の例は、典型的な文字セットが、2 つの異なるコード・ページの種々のコード・ポイントにどのようにマップされるかを示しています。

10. 正規化がパフォーマンスに与える影響は大きいので (CPU に 2.5 から 25 % の余分な負荷がかかります)、列定義のデフォルトは NOT NORMALIZED です。



同一のエンコード・スキームを使用している場合でも、異なるコード化文字セットが数多くあります。また、同一のコード・ポイントが別のコード化文字セットでは異なる文字を表すことがあります。さらに、文字ストリング中の 1 バイトが、必ずしも 1 バイト文字セット (SBCS) にある 1 文字を表すとは限りません。さらに、文字ストリングは、混合データ (1 バイト文字と 2 バイト文字の混合) や、どのような文字セットにも関連しないデータ (ビット・データと呼ばれる) にも使用されます。これはグラフィック・ストリングの場合には該当しません。その理由は、データベース・マネージャーでは、グラフィック・ストリングの場合には常に、そのバイトの対のすべてが、2 バイト文字セット (DBCS) または汎用コード化文字セット (UCS-2 または UTF-16) の文字を表しているものと想定しているためです。

固有エンコード・スキームのコード化文字セット ID (CCSID) は、データをそのサイトで保管できるコード化文字セットの 1 つです。外部エンコード・スキームの CCSID は、データをそのサイトで保管できないコード化文字セットの 1 つです。例えば、Db2 for i は、データを EBCDIC エンコード・スキームを持つ CCSID には保管できますが、ASCII エンコード・スキームには保管できません。

外部エンコード・スキーム (Unicode 以外) 内のデータを含む変数は、関数または選択リスト の中で使用されるときは、常に固有エンコード・スキームの CCSID に変換されます。また、外部エンコード・スキームのデータを含む変数は、比較またはストリングを組み合わせる操作の中で使用される場合も、固有エンコード・スキ

ームの CCSID に効果的に変換されます。データが固有エンコード・スキームのどの CCSID に変換されるかは、外部 CCSID およびデフォルト CCSID によって決まります。

文字変換の詳細については、以下を参照してください。

- 120 ページの『割り当ての際の変換規則』
- 129 ページの『比較の際の変換規則:』
- 139 ページの『ストリングを結合する演算に適用される変換規則』
- 55 ページの『データ表現に関する考慮事項』.

コード化文字セットと CCSID

IBM の Character Data Representation Architecture (CDRA) は、ストリングの表現とエンコードの違いを解消します。この体系のキー・エレメントとして、コード化文字セット *ID (CCSID)* があります。 CCSID は、2 バイト (無符号) の 2 進数で、文字セットとコード・ページの 1 つ以上の対およびエンコード・スキームを固有のものとして識別します。

長さがストリングの属性であるのとまったく同じように、CCSID はストリングの属性です。 1 つのストリング列にあるすべての値は、同一の CCSID を持ちます。

文字変換は、ソースとターゲットの CCSID を基準にして行われます。各データベース・マネージャーには、ソースとターゲットの有効な組み合わせを識別し、コード化文字セット間の変換を行うためのサポートが用意されています。場合によっては、関係するストリングの CCSID が異なっていても、変換が必要ないことがあります。

異なるタイプの変換が、データベース・マネージャーによってサポートされている場合があります。往復変換は、ターゲット CCSID に定義されていない別の CCSID で文字を保存しようとします。その際、続けてデータが元の CCSID に逆変換される場合に、その結果が元の同じ文字になるようにします。サブセット一致変換を強制的に行っても、元の文字で保存されることはありません。詳しくは、IBM の文字データ表現体系 (CDRA) を参照してください。

デフォルトの CCSID

すべてのアプリケーション・サーバーおよびアプリケーション・リクエスターには、デフォルト CCSID が 1 つあります (DBCS データをサポートするシステムには、複数のデフォルト CCSID があります)。

現行のサーバーでは、以下のタイプのストリングの CCSID が定められています。

---

11. デフォルト CCSID が 65535 である場合、使用されるこの CCSID が DFTCCSID ジョブ属性の値 (または DFTCCSID の関連 CCSID) になります。関連の混合データ CCSID がない場合は、次のようになります。
    - FOR MIXED DATA が指定されている場合に使用される CCSID は、65535 です。
    - GRAPHIC および VARGRAPHIC の場合の CCSID は、65535 です。
    - DBCLOB の場合の CCSID は、1200 です。

- ソースの CCSID が外部エンコード・スキームである場合のストリング定数 (日付/時刻の値を表すストリング定数を含む)
- ストリングの値を持つ特殊レジスター (USER や CURRENT SERVER など)
- CAST の指定。結果は文字またはグラフィック・ストリング
- CHAR、DATAPARTITIONNAME、DAYNAME、DBPARTITIONNAME、DIGITS、HEX、MONTHNAME、SOUNDEX、SPACE、 VARCHAR_FORMAT の各スカラー関数の結果
- CCSID が引数として指定されていない場合の DECRYPT_CHAR、DECRYPT_DB、CHAR、GRAPHIC、VARCHAR、および VARGRAPHIC スカラー関数の結果
- CCSID が引数として指定されていない場合の CLOB および DBCLOB スカラー関数の結果[11]
- CREATE TABLE または ALTER TABLE ステートメントで定義されているストリング列で、列について CCSID が明示的に指定されていない場合[11]
- 列で明示的な CCSID が指定されていない場合に DECLARE GLOBAL TEMPORARY TABLE ステートメントで定義するストリング列[11]
- ソース・タイプが文字ストリング・タイプまたはグラフィック・ストリング・タイプの場合の特殊タイプ
- CREATE FUNCTION または CREATE PROCEDURE ステートメントで定義されているストリング・パラメーターで、パラメーターについて CCSID が明示的に指定されていない場合[11]

上記のストリング・タイプの 1 つが CREATE VIEW ステートメントで使用される場合、デフォルト CCSID はビューの作成時に決定されます。

分散アプリケーションでは、アプリケーション要求側によって変数のデフォルト CCSID が決まります。非分散アプリケーションでは、変数のデフォルト CCSID はアプリケーション・サーバーによって決定されます。IBM i オペレーティング・システムでは、CCSID ジョブ属性によってデフォルトの CCSID が決まります。CCSID について詳しくは、「グローバリゼーション」トピック集の「CCSID の処理」トピックを参照してください。

照合順序

照合順序 (ソート順序ともいいます) は、文字セット内の文字の比較や順序付けを行うときに、文字が互いにどのように関係し合うかを定義します。

ある特定の言語にしたがってデータの順序付けを行う場合は、別の照合順序を使用するのが便利です。例えば、リストをある特定の言語で通常見られる順序でリストすることができます。照合順序を使用して、ある文字 (例えば、**a** と **A**) を同等として扱うこともできます。照合順序は、以下のものを含むすべての比較で機能します。

- SBCS 文字データ (ビット・データを含む)
- 混合データの SBCS 部分
- Unicode データ (UTF-8、UCS-2、または UTF-16)

SBCS 照合順序は、256 バイトの表を使用してサポートされています。この表では、それぞれのバイトが 1 つのコード・ポイントまたは SBCS コード・ページ内の文字に対応しています。照合順序は文字データに適用されるので、表には CCSID を関連付けておかなければなりません。照合順序表中のバイトは、各コード・ポイントとそのコード・ページ内の他のコード・ポイントとの対比に基づいて設定されています。例えば、文字 **a** と **A** を比較の際に同等として扱いたい場合には、照合順序表のこれらのコード・ポイントに対応するバイトには同じ値、すなわち、同じ重みが入ります。

UCS-2 照合順序は、マルチバイトの表を使用してサポートされています。表内の一対のバイトが、UCS-2 コード・ページの 1 文字に対応します。 UCS-2 の数千ある文字の 1 つのサブセットだけが、代表して表に表されます。比較して異なる文字だけが (おそらく、同じ区分内の他の文字も)、表に表されます。照合順序表中のバイトは、それぞれの文字と UCS-2 内の他の文字との対比に基づいて設定されます。

照合順序表の複数のバイト (あるいは、UCS-2 の場合は一対のバイト) に同じ値が入っている場合、その照合順序は共用重みソート順序です。照合順序表中のすべてのバイト (あるいは、UCS-2 の場合は一対のバイト) が固有の値を持つ照合順序は、固有重み照合順序です。システムでは、多くの言語に対応する固有重みおよび共用重みの照合順序が、オペレーティング・システムの一部として出荷されています。他の言語や要件に対応する照合順序が必要な場合には、表作成 (CRTTBL) コマンドを用いてそのソート順序を定義してください。

UTF-8 および UTF-16 照合順序サポートは、 ICU (International Components for Unicode) を使用してインプリメントされています。 ICU は Unicode をソートする標準 API です。この API は正規化および非正規化データに対して同じ結果を生成し、言語固有の規則に基づいてソートの順番を戻します。 IBM i オペレーティング・システムは ICU 2.3.1、ICU 3.4、および ICU 4.0 照合順序をサポートしていますが、ICU 3.4 または ICU 4.0 を使用するようにしてください。 ICU 照合順序表 I34en_us (米国ロケール) は、I34fr_FR (フランス・ロケール) とはデータのソート方法が異なります。

ICU を使用する場合、LIKE 述部および
LOCATE、POSITION、POSSTR、POSITION の各スカラー関数はサポートされません。

ICU 2.3.1 を使用する場合は、照会に以下のものを含めることはできません。

- EXCEPT または INTERSECT 操作
- 全選択の中の VALUES
- OLAP の指定
- 反復共通表式
- ORDER OF
- スカラー全選択 (スカラー副選択はサポートされています)
- 全外部結合
- GROUP BY の中の LOB、
- グループ化集合またはスーパー・グループ
- 副選択の中の ORDER BY または FETCH 節

- OFFSET 節、または N 行の変数を指定した FETCH 節
- CORRELATION、COVARIANCE、COVARIANCE_SAMP、LISTAGG、MEDIAN、PERCENTILE_CONT、 PERCENTILE_DISC、または回帰集約関数
- VERIFY_GROUP_FOR_USER、LOCATE_IN_STRING、2 つの引数を指定した LTRIM または RTRIM、EPOCH を指定した EXTRACT 関数、 HASH 関数
- BSON_TO_JSON、JSON_ARRAY、 JSON_ARRAYAGG、 JSON_OBJECT、JSON_OBJECTAGG、JSON_QUERY、 JSON_TABLE、JSON_TO_BSON、および JSON_VALUE 関数と、IS JSON および JSON_EXISTS 述部
- CONTAINS または SCORE 関数
- XMLAGG、XMLATTRIBUTES、XMLCOMMENT、XMLCONCAT、XMLDOCUMENT、XMLELEMENT、XMLFOREST、XMLGROUP、XMLNAMESPACES、XMLPI、XMLROW、または XMLTEXT 関数
- ユーザー定義関数のデフォルト値
- グローバル変数、または
- 配列の参照

ICU 照合順序表は、一般には言語使用の観点からより正確な結果を生成しますが、以下の状況が観察されています。

- ICU 照合順序表を使用する SQL ステートメントのパフォーマンスは、一般に SBCS または UCS-2 のいずれかの照合順序表を使用した場合に比べて劣ります。ただし、ICU 照合順序表とともに索引を作成して、パフォーマンスを改善することができます。この場合、索引キー値には ICU の重み付けされた値が含まれ、この値によってシステムの ICU サポートを呼び出す回数が非常に少なくて済みます。
- ICU 照合順序表を使用する索引に必要な記憶域は、一般に SBCS または UCS-2 のいずれかの照合順序表を使用した場合に比べて優れています。キー値は、キーを生成するために使用される SBCS データの長さの 3 倍、およびキーを生成するために使用される DBCS データの長さの 6 倍までが可能です。

照合順序によってデータ自体が変わるわけではないことを覚えておいてください。比較には、代わりにデータの重み付け表現が使用されます。SQL では、照合順序は CRTSQLxxx、STRSQL、および RUNSQLSTM コマンドで指定します。 SET OPTION ステートメントを使用して、組み込み SQL を含むプログラムのソース内に照合順序を指定することができます。指定された照合順序は、SQL ステートメントで実行されるすべての文字比較に適用されます。システムのデフォルトの照合順序は、文字の 16 進表示を使用する際に生じる内部順序です。これは、SRTSEQ(*HEX) を指定した場合の順序です。バージョン 2 リリース 3 より前のプロダクト・リリースでプリコンパイルされたプログラムの場合、照合順序は *HEX です。

照合順序は、FOR BIT DATA 列やバイナリー・ストリング列には適用されません。

照合順序は、フィールド・プロシージャーのあるキー列が含まれる索引またはユニーク制約には使用できません。

照合順序 は、以下のインターフェースを使用して明示的に指定されます。

表 3. 照合順序インターフェース

SQL インターフェース	指定
組み込み SQL	「SQL プログラムの作成」(CRTSQLxxx) コマンドの SRTSEQ パラメーター。SRTSEQ の値の設定に SET OPTION ステートメントも使用可能。 (CRTSQLxxx コマンドについて詳しくは、「組み込み SQL プログラミング」を参照してください。)
SQL ステートメント実行	SQL ステートメント実行 (RUNSQLSTM) コマンドの SRTSEQ パラメーター。 (RUNSQLSTM コマンドの詳細については、「SQL プログラミング」を参照。)
サーバー上の呼び出しレベル・インターフェース (CLI)	SQL_ATTR_JOB_SORT_SEQUENCE 環境変数 (CLI について詳しくは、「SQL 呼び出しレベル・インターフェース (ODBC)」を参照してください。)
IBM IBM Developer Kit for Java を使用したサーバーの JDBC または SQLJ	job.sort.sequence プロパティー・オブジェクト (JDBC および SQLJ について詳しくは、「IBM Developer Kit for Java」を参照してください。)
IBM i Access Family ODBC ドライバーを使用したクライアントの ODBC	ODBC セットアップでのソート・タイプ (ODBC の詳細については、「IBM i Access」を参照。)
IBM Toolbox for Java を使用したクライアントの JDBC	JDBC セットアップでのソート順序表 (JDBC の詳細については、「IBM i Access」を参照。) (IBM Toolbox for Java について詳しくは、「IBM Toolbox for Java」を参照してください。)
IBM i Access Family OLE DB Provider を使用したクライアントの OLE DB	ソート順序接続オブジェクト・プロパティー (OLE DB の詳細については、「IBM i Access」を参照。)
IBM i Access FamilyADO .NET プロバイダーを使用しているクライアントの ADO .NET	接続オブジェクト・プロパティー内の SortSequence (ADO .NET の詳細については、「IBM i Access」を参照。)

CCSID の詳細については、IBM i Information Center のグローバリゼーション・セクションの中の CCSID の処理トピックを参照してください。 照合順序、およびシステムに付属している照合順序について詳しくは、IBM i Information Centerにある Db2 および SQL 照合順序のトピックを参照してください。

分散リレーショナル・データベース

分散リレーショナル・データベース は、一組の表とその他のオブジェクト (別個ではあるが相互に接続されているコンピューター・システムまたは同一のコンピューター・システム上の論理区画にまたがって存在している) で構成されます。各コンピューター・システムには、それぞれその環境で表を管理するリレーショナル・データベース・マネージャーがあります。これらのデータベース・マネージャーは、相互に情報を交換し連携することによって、それぞれのデータベース・マネージャーが別のコンピューター・システムに対して SQL ステートメントを実行することができる仕組みになっています。

分散リレーショナル・データベースは、正式なリクエスター (要求元) とサーバーのプロトコルおよび機能に基づいて構築されます。アプリケーション・リクエスターは、接続のアプリケーション側をサポートします。アプリケーション・リクエスターは、アプリケーションのデータベース要求を、分散データベース・ネットワークによる使用に適した通信プロトコルに変換します。これらの要求は、接続のもう一方の側にあるアプリケーション・サーバー が受信して処理します。[12] アプリケーション・リクエスターとアプリケーション・サーバーが、協調して通信とロケーションに関する処理を行うので、アプリケーションはこうした処理を考慮する必要がなく、ローカル・データベースにアクセスする場合と同様に動作できます。 図 8 は、単純な分散リレーショナル・データベース環境を示しています。

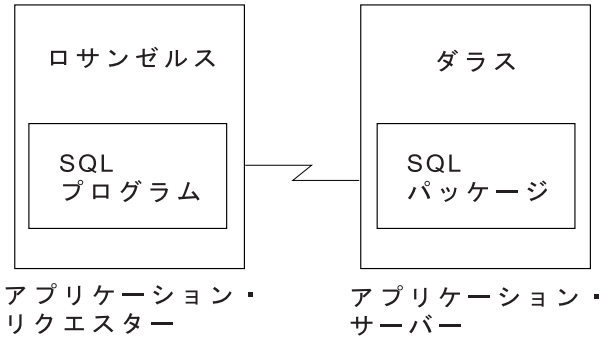


図 *8.* 分散リレーショナル・データベース環境

Distributed Relational Database Architecture™ (DRDA) の通信プロトコルの詳細については、「Open Group Publications: DRDA Vol. 1: Distributed Relational Database Architecture (DRDA)」 を参照してください。

アプリケーション・サーバー

SQL ステートメントを実行できるようにするには、データベース・マネージャーのアプリケーション・サーバーに活動化グループを接続しておくことが必要です。

接続 とは、活動化グループとローカルまたはリモートのアプリケーション・サーバーの間の関連付けのことです。接続は、セッションまたは SQL セッションとも呼ばれます。接続は、アプリケーションにより管理されます。 CONNECT ステートメントを使用することにより、アプリケーション・サーバーへの接続を確立し、そのアプリケーション・サーバーを活動化グループの現行サーバーとすることができます。

暗黙的な CONNECT 操作によってアプリケーション・サーバーとの接続を確立することもできます。

- プログラム、サービス・プログラム、STRSQL コマンドのいずれかを呼び出すと、暗黙的な CONNECT 操作が発生する可能性があります。対象のアプリケーション・サーバーは、CRTSQLxxx コマンドと STRSQL コマンドの RDB パラメーターで指定します。

12. これはアプリケーション・サーバー とも呼ばれます。

そのような場合でも、暗黙的または明示的な CONNECT 操作が既に活動化グループで発生していれば、それが成功したか失敗したかにかかわりなく、暗黙的な CONNECT 操作が発生することはありません。したがって、活動化グループは、1 つのアプリケーション・サーバーに複数回にわたり暗黙に接続することはできません。

- 3 部構成のオブジェクト名、あるいは表またはビューの 3 部構成の名前を参照するために定義した別名を使用すると、暗黙的な CONNECT 操作が発生する可能性があります。

  そのような場合に暗黙的な CONNECT 操作が可能なのは、SQL ステートメントで参照するすべてのオブジェクトが同じリレーショナル・データベースを参照する場合に限られます。例外は次の場合のみです。

  – INSERT ステートメントのターゲット表が、あるリレーショナル・データベース内にあり、INSERT の *select-statement* 内で参照されている表が別のリレーショナル・データベース内にある。

  – CREATE TABLE ステートメントまたは DECLARE GLOBAL TEMPORARY TABLE ステートメントの新規表が、あるリレーショナル・データベース内にあり、 *select-statement* 内で参照されている表が別のリレーショナル・データベース内にある。

  暗黙的な CONNECT が発生すると、ステートメントの現行サーバーが変わります。ステートメントの最後の時点で、前の現行サーバーに接続が再び切り替わります。

- 3 つの部分で修飾された SQL ストアード・プロシージャーまたは SQL 関数を作成する場合、プロシージャー本体内のオブジェクトは、DRDA クライアントと DRDA サーバーの両方に存在していなければなりません。プロシージャー本体のオブジェクトが存在しない場合、オブジェクトが見つからないというエラーが通知される可能性があります。

アプリケーション・サーバーは、活動化グループが開始される環境にとってローカルでもリモートでも構いません。 (アプリケーション・サーバーは、分散リレーショナル・データベースを使用しない場合でも存在します。) この環境には、CONNECT ステートメントに指定されるアプリケーション・サーバーを記述するローカル・ディレクトリーが含まれています。このディレクトリーについて詳しくは、System i ナビゲーターのリレーショナル・データベース・フォルダー、あるいは以下の IBM i Information Center トピック内のディレクトリー・コマンド (ADDRDBDIRE、CHGRDBDIRE、DSPRDBDIRE、RMVRDBDIRE、および WRKRDBDIRE) を参照してください。

- SQL プログラミング
- 分散データベース・プログラミング
- CL 解説書

表またはビューを参照する静的 SQL ステートメントを実行するときは、アプリケーション・サーバーはバインド済み形式のステートメントを使用します。このバインド済みのステートメントは、前もってデータベース・マネージャーがバインド操作によって作成したパッケージから得られます。以下の組み合わせによって適切なパッケージが決定されます。

- CRTSQLxxx コマンドの SQLPKG パラメーターによって指定されたパッケージの名前。 CRTSQLxxx コマンドの説明については、「組み込み SQL プログラミング」を参照してください。
- 内部の整合性トークン (パッケージおよびプログラムが、同時に同じソースから作成されたことを確認する)。

Db2 リレーショナル・データベース製品は、アプリケーション・サーバーに接続されている Db2 製品のバージョンではサポートされていない機能をサポートしている場合があります。このような機能は製品固有であり、複数の製品に共通している場合もあります。

アプリケーションは、そのアプリケーションが現在接続されているアプリケーション・サーバーのデータベース・マネージャーがサポートしているステートメントおよび文節のほとんどを使用することができます。これは、このアプリケーションが、これらのステートメントおよび文節の一部をサポートしていないデータベース・マネージャーのアプリケーション・リクエスターを使用して実行されている場合にも言えることです。 制限については、 1855 ページの『付録 B. SQL ステートメントの特性』に示されています。

CONNECT (タイプ 1) と CONNECT (タイプ 2)

CONNECT ステートメントには、構文は同じで意味体系が異なる 2 つのタイプがあります。 CONNECT (タイプ 1) は、リモート作業単位に対して使用されます。 CONNECT (タイプ 2) は、分散作業単位に対して使用されます。

その相違点の要約については、 1865 ページの『CONNECT (タイプ 1) と CONNECT (タイプ 2) の相違点』を参照してください。

リモート作業単位

リモート作業単位 機能を使用することにより、リモートでの SQL ステートメントの準備と実行が可能になります。コンピューター・システム A にある活動化グループは、コンピューター・システム B にあるアプリケーション・サーバーに接続することができます。さらに、その活動化グループは、1 つ以上の作業単位の中で、B にあるオブジェクトを参照する静的または動的 SQL ステートメントをいくつでも実行することができます。B にある作業単位の処理が終わると、活動化グループは、コンピューター・システム C にあるアプリケーション・サーバーに接続することができます (このようにして次々に処理を続けることができます)。

ほとんどの SQL ステートメントは、以下の制約を伴うものの、リモートで準備し実行することができます。

- 単一の SQL ステートメントで参照されるオブジェクトは、すべて同一のアプリケーション・サーバーによって管理される必要があります。
- ある作業単位内の SQL ステートメントは、すべて同一のアプリケーション・サーバーによって実行される必要があります。

リモート作業単位の接続管理

活動化グループは、必ず以下の 3 つの状態のいずれかになります。

- 接続可能/接続状態

- 接続不能/接続状態
- 接続可能/未接続状態

次の図は、状態の推移を示しています。

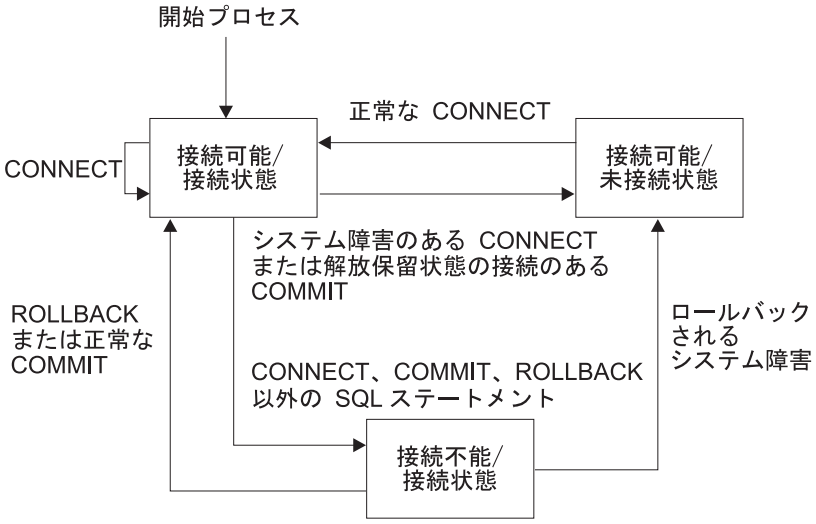


図 *9.* リモート作業単位の活動化グループの接続状態の推移

活動化グループの初期状態は、接続可能/接続状態 です。

接続可能/接続状態

活動化グループはアプリケーション・サーバーに接続されており、CONNECT ステートメントが実行可能です。活動化グループがこの状態に入るのは、活動化グループが接続不能/接続状態からロールバックまたは正常なコミットを完了したとき、または CONNECT ステートメントが接続可能/未接続状態から正常に実行されたときです。

接続不能/接続状態

活動化グループはアプリケーション・サーバーに接続されていますが、CONNECT ステートメントを正常に実行できないためアプリケーション・サーバーを変更できません。活動化グループは、CONNECT、COMMIT、または ROLLBACK 以外の SQL ステートメントを実行すると、接続可能/接続状態からこの状態に入ります。

接続可能/未接続状態

活動化グループはアプリケーション・サーバーに接続されていません。この状態で実行できる SQL ステートメントは、CONNECT だけです。

活動化グループは、以下の場合にこの状態に入ります。

- 前もって接続が解除されており、正常な COMMIT が実行される。
- SQL DISCONNECT ステートメントを使用して、接続が切り離される。
- 接続が接続可能状態であったが CONNECT ステートメントの実行が失敗した。

CONNECT ステートメントは、連続して使用しても正常に実行されます。これは、CONNECT ステートメントは接続可能状態からその活動化グループを除去しないからです。活動化グループが現在接続されているアプリケーション・サーバーに対する CONNECT は、他の CONNECT ステートメントと同様に実行されます。CONNECT、 COMMIT、 DISCONNECT、 SET CONNECTION、 RELEASE、または ROLLBACK (COMMIT(*NC) を指定して実行する場合を除く) 以外の SQL ステートメントが前に実行されていると、CONNECT は正常に実行できません。エラーを避けるために、CONNECT ステートメントを実行する前に、コミットまたはロールバック操作を実行してください。

アプリケーション指向の分散作業単位

アプリケーション指向の分散作業単位機能 は、リモート作業単位の場合と同じように、リモートでの SQL ステートメントの準備と実行を可能にします。リモート作業単位の場合と同様に、コンピューター・システム A の活動化グループは、コンピューター・システム B のアプリケーション・サーバーに接続することができます。また、作業単位が終了する前に、B にあるオブジェクトを参照する静的または動的 SQL ステートメントをいくつでも実行することができます。単一の SQL ステートメントで参照されるオブジェクトは、すべて同一のアプリケーション・サーバーによって管理される必要があります。ただし、リモート作業単位の場合とは異なり、同じ作業単位に任意の数のアプリケーション・サーバーが関与することができます。コミット、またはロールバックの操作により、作業単位は終了します。

分散作業単位は APPC および TCP/IP 接続用に完全サポートされています。

アプリケーション指向の分散作業単位接続管理

どの時点でも、以下のことが当てはまります。

- 活動化グループは、必ず接続状態 または未接続状態 にあり、ゼロまたはそれ以上の接続からなる一組の接続を持っています。活動化グループの個々の接続は、その接続のアプリケーション・サーバーの名前によって一意的に識別されます。
- SQL 接続は、常に以下の状態のいずれかです。
  - 現行/保留
  - 現行/解除保留
  - 休止/保留
  - 休止/解除保留

活動化グループの初期状態**:**

活動化グループは最初は接続状態にあり、接続は 1 つだけです。接続の初期状態は、現行/保留 です。

次の図は、状態の推移を示しています。

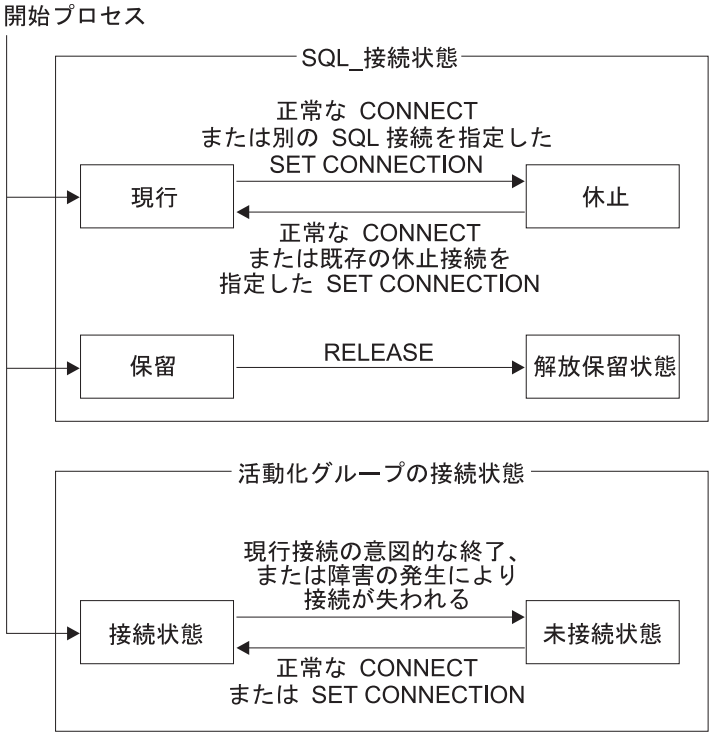


図 *10.* アプリケーション指向の分散作業単位の接続および活動化グループの接続状態の推移

接続状態

アプリケーション・プロセスが CONNECT ステートメントを正常に実行した場合:

- 現行接続が休止/保留状態になる。
- そのサーバー名が接続の組に追加され、その新たな接続が現行/保留状態になる。

該当のサーバー名が、活動化グループの既存の接続のセットに既に存在する場合には、エラーが戻されます。

休止状態の接続は、SET CONNECTION ステートメントの使用により現行状態になります。ある接続が現行状態になると、それ以前の現行接続 (存在する場合) は休止状態になります。どのような時点でも、活動化グループの既存の接続のセットの複数の接続が、現行状態になることはありません。接続の状態を現行から休止へ、または休止から現行へ変更しても、その保留状態や解除保留状態には影響しません。

接続は、RELEASE ステートメントによって解除保留状態になります。活動化グループがコミット操作を実行すると、その活動化グループの解除保留状態の接続はすべて終了します。接続の状態を保留から解除保留へ変更しても、その現行状態や休止状態には影響しません。したがって、解除保留状態の接続は、次のコミット操作まで引き続き使用できます。接続の状態を解除保留から保留へ変更する手段はありません。

活動化グループの接続状態

CONNECT ステートメントの明示的または暗黙的な実行によって、異なるアプリケーション・サーバーを確立することができます。以下の規則が適用されます。

- 1 つの活動化グループが、同じアプリケーション・サーバーに対して同時に複数の接続を持つことはできません。
- 活動化グループが SET CONNECTION ステートメントを実行する場合、指定するロケーション名は、その活動化グループの既存の接続のセットに存在する既存の接続でなければなりません。
- 活動化グループが CONNECT ステートメントを実行する場合、指定するサーバー名は、その活動化グループの既存の接続のセットに存在する既存の接続であってはなりません。

活動化グループが現行接続を持っている場合は、その活動化グループは接続 状態にあります。 CURRENT SERVER 特殊レジスターには、現行接続のアプリケーション・サーバーの名前が入っています。活動化グループは、そのアプリケーション・サーバーが管理するオブジェクトを参照する SQL ステートメントを実行することができます。

未接続状態の活動化グループが CONNECT または SET CONNECTION ステートメントを実行し、成功すると、接続状態になります。

活動化グループが現行接続を持っていない場合は、その活動化グループは未接続 状態にあります。特殊レジスター CURRENT SERVER の内容は、ブランクに等しくなります。実行できる SQL ステートメントは、CONNECT、DISCONNECT、SET CONNECTION、RELEASE、COMMIT、および ROLLBACK のみです。

接続状態の活動化グループが未接続状態に入るのは、その現行接続を意図的に終了させた場合、または現行サーバーのロールバック操作または接続の消失の原因となる障害のために、SQL ステートメントの実行が正常になされなかった場合です。接続を意図的に終了させるのは、活動化グループがコミット操作を正常に実行し、しかもその接続が解除保留状態にある場合、またはアプリケーション・プロセスが DISCONNECT ステートメントを正常に実行した場合です。

接続の終了時

接続が終了すると、その接続を介して活動化グループが獲得していたすべてのリソース、およびその接続の確立や維持に使用されていたすべてのリソースが割り振り解除されます。例えば、アプリケーション・プロセス P がアプリケーション・サーバー X への接続をアプリケーション・サーバー状態にした場合は、次回のコミット操作時に接続が終了した時点で、X にある P のすべてのカーソルがクローズされ、割り振り解除されます。

また、接続は通信障害の結果として終了することもあり、その場合、該当の活動化グループは、未接続状態になります。活動化グループが終了すると、その活動化グループの接続はすべて終了します。

データ表現に関する考慮事項

システムが異なれば、データの表現方法も異なります。あるシステムから別のシステムにデータを移す場合に、データ変換が必要になることがあります。DRDA をサポートするプロダクトでは、データ変換が必要な場合、その変換は受信側システムで自動的に行われます。

数字データの変換を行うために必要な情報は、データ・タイプと送信側システムの環境タイプです。例えば、z/OS® アプリケーション・サーバーで、表の列に Db2 for i アプリケーション・リクエスターからの浮動小数点変数が割り当てられている場合は、その数値は IEEE 形式から System/370* (システム/370) 形式に変換されます。

文字データや図形データの場合には、データ・タイプと送信側システムの環境タイプだけでは十分ではありません。文字や図形のストリングを変換するには、さらに情報が必要になります。ストリングの変換は、データのコード化文字セットおよびそのデータに対して行われる操作の両方に基づいて行われます。ストリングの変換は、IBM 文字データ表現体系 (CDRA) に従って行われます。文字変換の詳細については、「*Character Data Representation Architecture Reference and Registry*」(SC09-2190) という解説書を参照してください。

第 2 章 言語エレメント

このセクションでは、SQL の基本構文と、多くの SQL ステートメントに共通する言語エレメントについて定義します。

文字

SQL 言語のキーワードおよび演算子で使用する基本的な記号は、IBM リレーショナル・データベース製品がサポートするすべての文字セットに含まれる 1 バイト文字です。

この言語の文字は、文字、数字、または特殊文字に分類されます。[13]

文字 とは、英語のアルファベットの 26 の大文字 (A から Z) と 26 の小文字 (a から z) の任意の文字を指します。[14]

数字 は、0 から 9 までのいずれかです。

特殊文字 は、以下に示す文字のいずれかです。

	スペースまたはブランク	-	負符号
"	引用符または二重引用符	.	ピリオド
%	パーセント	/	斜線 (スラッシュ)
&	アンパーサンド	:	コロン
'	アポストロフィまたは単一引用符	;	セミコロン
(	左括弧	<	より小さい
)	右括弧	=	等号
*	アスタリスク	>	より大きい
+	正符号	?	疑問符
,	コンマ	_	下線
|	縦線[16]	^	脱字記号
!	感嘆符[15]	[	左大括弧
{	左中括弧	]	右大括弧
}	右中括弧	¬	否定[15]

13. SQL ステートメントが Unicode データとしてエンコードされている場合は、ストリング定数を除き、ステートメントのすべての文字が処理の前に 1 バイト文字に変換されるという点に注意してください。ストリング定数を表すトークンは、1 バイトに変換せずに UTF-16 グラフィック・ストリングとして処理することができます。

14. 文字には、各国言語用にアルファベットの拡張として予約された 3 つのコード・ポイントが含まれています (米国の場合、#、@、および $)。この 3 つのコード・ポイントは CCSID によって異なる文字を表すので、使用しないようにしてください。

15. 否定記号 (¬) および感嘆符 (!) を使用すると、IBM リレーショナル・データベース製品間のコードの移植性が阻害されることがあります。これらの記号は可変文字なので、使用を避けてください。 ¬= または != の代わりに、<> を使用してください。 ¬> または !> の代わりに、<= を使用してください。¬< または !< の代わりに、>= を使用してください。

16. 縦線 (|) 文字を使用すると、IBM リレーショナル・データベース製品間のコードの移植性が阻害されることがあります。連結記号 (||) の代わりに、CONCAT 演算子を使用してください。

トークン

言語の基本的な構文単位のことを、トークン と呼びます。 1 つのトークンは、1 つ以上の文字から構成されます。ただし、この文字にはブランクや制御文字は入りません。また、ストリング定数または区切り文字付き ID 内の文字も除きます。(これらの用語については後述します。)

トークンは、通常トークン と区切りトークン に分類されます。

- 通常トークン とは、数値定数、通常 ID、ホスト ID、またはキーワードです。

  例

  ```
  1       .1       +2       SELECT       E       3
  ```

- 区切りトークン とは、ストリング定数、区切り文字付き ID、演算記号、または構文図に示される任意の特殊文字を指します。疑問符 (?) も、 1603 ページの『PREPARE』で説明しているようなパラメーター・マーカーとして使用される場合は、区切りトークンとなります。

  例

  ```
  ,     'Myst Island'       "fld1"       =       .
  ```

スペース:

スペース は、1 つ以上のブランク文字を並べたものです。

制御文字:

制御文字 は、ストリングの位置合わせに使用される特殊文字です。次の表は、データベース・マネージャーが取り扱う制御文字を示しています。

表 4. 制御文字

制御文字	**EBCDIC 16** 進値	**UTF-8** の **16** 進値	**UCS-2** および **UTF-16** の **16** 進値
タブ	05	09	U+0009
用紙送り	0C	0C	U+000C
復帰	0D	0D	U+000D
改行	15	C285	U+0085
行送り (改行)	25	0A	U+000A
DBCS スペース	—	—	U+3000

ストリング定数および特定の区切り文字付き ID 以外のトークンには、制御文字またはスペースを含めてはなりません。制御文字またはスペースは、トークンの後ろに続けることができます。区切りトークン、制御文字、またはスペースが、すべての通常トークンの後に続かなければなりません。構文上、通常トークンの後に区切りトークンを続けることが許されない場合には、その通常トークンの後に制御文字またはスペースを続ける必要があります。ここで述べた規則について、以下に例を示します。

上記の通常トークンをいくつか組み合わせると、トークンが事実上変化してしまいます。その例を次に示します。

```
1.1        .1+2        SELECTE        .1E        E3        SELECT1
```

このため、通常トークンの後には必ず区切りトークンまたはスペースを置かなければなりません。

上記の通常トークンと区切りトークンを組み合わせると、トークンが事実上変化してしまうことがあります。この例を次に示します。

```
1.        .3
```

名前の修飾でピリオド (.) を 区切り記号として使用すると、そのピリオドは区切りトークンになります。上記の例では、ピリオドを通常トークンの数値定数と組み合わせて使用しています。このため、通常トークンの後に区切りトークンを置くことは構文上許されません。このような場合は、通常トークンの後に、区切りトークンではなくスペースを置かなければなりません。

147 ページの『小数点』で説明するように、小数点がコンマとして定義されている場合は、コンマが数値定数の小数点として解釈されます。この場合の数値定数の例を、次に示します。

```
1,2        ,1        1,        1,e1
```

'1,2' および '1,e1' がそれぞれ 2 つの項目を表す場合には、コンマが小数点として解釈されるのを防ぐために、通常トークン (1) の後と区切りトークン (,) の後の両方にスペースを置かなければなりません。コンマは、通常は区切りトークンですが、小数点として解釈される場合には数値の一部となります。したがって、通常トークン (1) の後に区切りトークン (,) を続けることは構文上許されません。このような場合は、通常トークンの後に区切りトークンではなくスペースを置く必要があります。

コメント:

動的 SQL ステートメントの中に SQL コメントを組み込むことができます。静的 SQL ステートメントの中では、ホスト言語コメントまたは SQL コメントを組み込むことができます。コメントは、スペースを指定できる場所であればどこでも指定できますが、区切りトークンの中またはキーワードの EXEC と SQL の間は例外です。Java では、組み込み Java 式内での SQL コメントは許可されていません。SQL コメントには、次の 2 つのタイプがあります。

単純コメント
: 単純コメントは、2 つの連続するハイフン (--) で始まります。単純コメントは、その行の終わりを超えて続けることはできません。詳しくは、 878 ページの『SQL のコメント』を参照してください。

ブラケット付きのコメント
: ブラケット付きコメントは、/* と */ で囲みます。括弧付きのコメントは、その行の終わりを超えて続けることができます。詳しくは、 878 ページの『SQL のコメント』を参照してください。

大文字と小文字:

SQL ステートメントで使用するトークンには、小文字を含めることもできますが、通常トークンの小文字は大文字に変換されます (ただし、ID に大/小文字の区別がある C 言語と Java 言語の変数は例外です)。区切りトークンが大文字に変換され

ることはありません。したがって、次のステートメントの場合、

```
select * from EMP where lastname = 'Smith';
```

変換後は、以下のステートメントと同等になります。

```
SELECT * FROM EMP WHERE LASTNAME = 'Smith';
```

ID

ID は、名前を形成するのに使われるトークンです。SQL ステートメントで使用する ID は、SQL ID、システム ID、ホスト ID のいずれかです。

注**:** $、@、#、および他のすべての可変文字は、それらの文字を表すのに使用するコード・ポイントがそれらの文字を含むストリングの CCSID によって変わるため、ID で使用してはなりません。これらの文字を使用すると、予期しない結果が起こる可能性があります。可変文字について詳しくは、「Db2 and SQL sort sequence」トピックを参照してください。

SQL ID

SQL ID には、通常 *ID* と区切り文字付き *ID* の 2 つのタイプがあります。

- 通常 *ID* は大文字で、後にゼロ個またはそれ以上の英大文字、数字、または下線文字が続いています。通常 ID は、大文字に変換されるので注意してください。通常 ID は、予約語であってはなりません。予約語のリストについては、 2201 ページの『付録 I. 予約済みスキーマ名と予約語』を参照してください。予約語を SQL における ID として使用する場合は、大文字で指定するべきです。また、区切り文字付き ID であるか、変数に指定する必要があります。
- 区切り文字付き *ID* は、1 つ以上の文字の並びを SQL エスケープ文字で囲んだものです。このシーケンスは、1 つ以上の文字で構成されていなければなりません。シーケンスの先行ブランクは、意味を持ちます。シーケンスの末尾のブランクは、意味を持ちません。 2 つの SQL エスケープ文字は、区切り文字付き ID の長さには含まれません。区切り文字付き ID は、大文字に変換されないので注意してください。エスケープ文字には引用符 (") を使用します。ただし、次のような場合は例外で、アポストロフィ (') をエスケープ文字として使用します。
  - 対話式 SQL で、SQL ストリング区切り文字が、COBOL 構文検査ステートメント・モードで引用符に設定されている場合。
  - COBOL プログラムにおける動的 SQL で、CRTSQLCBL または CRTSQLCBLI のパラメーター OPTION(*QUOTESQL) が、ストリング区切り文字を引用符 (") として指定している場合。
  - COBOL のアプリケーション・プログラムで、CRTSQLCBL または CRTSQLCBLI のパラメーター OPTION(*QUOTESQL) が、ストリング区切り文字を引用符 (") として指定している場合。

  区切り文字付き ID の中では、以下の文字は使用することはできません。
  - X'00' から X'3F'、および X'FF'

システム ID

IBM i オペレーティング・システムでシステム・オブジェクトの名前を形成するときに、システム ID を使用します。2 つのタイプのシステム ID があります。すなわち、通常 ID と区切り文字付き ID です。

- 通常システム ID の形成に関する規則は、SQL 通常 ID の場合の規則と同じです。
- 区切り文字付きシステム ID の形成に関する規則は、以下の点を除き、SQL 区切り文字付き ID の場合の規則と同じです。
  - 次の特殊文字は、区切り文字付きシステム ID には使用できません。
    - ブランク (X'40')
    - アスタリスク (X'5C')
    - アポストロフィ (X'7D')
    - 疑問符 (X'6F')
    - 引用符 (X'7F')
  - エスケープ文字用に必要なバイト数は、その区切り文字内の文字が通常 ID を形成する場合を除き、その ID の長さに含まれます。

    例えば、"PRIVILEGES" は大文字であり、区切り文字で囲まれている文字が通常 ID を形成するので、長さが 10 バイトで、列の有効なシステム名になります。または、"privileges" は小文字であり、区切り文字に必要なバイト数を ID の長さに含めなければならないため、長さが 12 バイトになり、列の有効なシステム名にはなりません。

例

```
WKLYSAL     WKLY_SAL     "WKLY_SAL"     "UNION"     "wkly_sal"
```

ID の最大長については、 62 ページの『命名規則』を参照してください。

ホスト ID

ホスト *ID* は、ホスト・プログラムで宣言された名前です。

ホスト ID を形成する際の規則は、ホスト言語の規則に従います。ただし、ホスト ID には、DBCS 文字は使用できません。例えば、COBOL プログラムでホスト ID を形成する際の規則は、COBOL プログラムでユーザー定義語を形成する際の規則と同じです。プリコンパイラーでは、'SQ'、'SQL'、'sql'、'RDI'、または 'DSN' という文字で始まるホスト変数を生成するため、これらの文字で始まる名前を使用してはなりません。 Java では、'__sJT_' で始まる名前を使用しないでください。

Db2 for i が課しているホスト ID 名の最大サイズ制限については、 70 ページの表 5 を参照してください。

17. 'SQ' は、C、COBOL、および PL/I では使用できますが、RPG では使用できません。

命名規則

名前を形成する規則は、その名前によって指定されるオブジェクトのタイプと命名オプション (*SQL または *SYS) に依存します。命名オプションは、CRTSQLxxx、RUNSQLSTM、および STRSQL コマンドに指定します。SET OPTION ステートメントを使用して、組み込み SQL を含むプログラムのソース内に命名オプションを指定することができます。構文図では、名前のタイプに応じてさまざまな用語が使用されています。

以下にそれらの用語の定義を示します。

別名　別名を示す修飾名または非修飾名です。別名 の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名 の後にピリオド (.) と SQL ID が続きます。システム命名の場合、修飾形式は、スキーマ名 の後にスラッシュ (/) と SQL ID が続きます[18]。

非修飾形式の別名 は、SQL ID です。非修飾形式は、 72 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

別名 では、別名の名前、または別名のシステム・オブジェクトの名前のいずれかを指定することができます。

配列タイプ名

配列タイプを示す修飾名または非修飾名。配列タイプ名 の修飾形式は、命名オプションに依存します。SQL 命名規則の場合、修飾形式は、スキーマ名 の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名 の後にスラッシュ (/) と SQL ID [18] が続きます。

非修飾形式の配列タイプ名 は、SQL ID です。非修飾形式は、 72 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

システム命名規則では、配列タイプ名 は、SQL ルーチンのパラメーター・データ・タイプ内で使用されている場合、あるいは SQL プロシージャーの SQL 変数宣言内で使用されている場合は、修飾することはできません。

権限名

ユーザー、またはユーザーのグループを指定するシステム ID。権限名 は、サーバー上のユーザー・プロファイル名です。この名前は、小文字または特殊文字を含む区切り文字付き ID であってはなりません。権限名 と権限 ID の相違については、 78 ページの『権限 ID と権限名』を参照してください。

列名　表またはビューの列を示す修飾名または非修飾名です。非修飾形式の列名は、SQL ID です。修飾形式の列名は、修飾子の後にピリオドと SQL ID を付けたものになります。この場合の修飾子は、表名、ビュー名、または相関名です。

システム命名では、列名は、COMMENT および LABEL ステートメント内で使用される場合は、スキーマ名/表名.列名 の形式を使用して修飾できま

18. システム命名の場合、ピリオドを使用する修飾形式も受け入れられます。

す。列名を修飾する必要があり、ステートメント内で相関名が許可される場合、相関名を使用して列を修飾できます。ピリオドを使用する修飾形式も使用可能です。

列名 は、表、またはビューの列名、あるいはシステム列名を指定します。列名 が区切り文字で区切られている場合、名前の長さを判別するときに、区切り文字は名前の一部と見なされます。

制約名

表についての制約を指定する修飾または非修飾の名前です。制約名 の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名 の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名 の後にスラッシュ (/) と SQL ID [18] が続きます。

非修飾形式の制約名 は、SQL ID です。非修飾形式は、 72 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

暗黙的または明示的な修飾子は、表のスキーマ名と同じでなければなりません。

相関名

表、ビュー、または表やビューの個々の行を示す SQL ID です。

カーソル名

SQL カーソルを示す SQL ID です。

記述子名

SQL 記述子域 (SQLDA) を指定する変数名またはストリング定数です。SQL 記述子域を示す変数には、標識変数を指定してはなりません。:ホスト変数:標識変数 という形式は使用できません。変数の説明については、172 ページの『ホスト変数に対する参照』を参照してください。

特殊タイプ名

特殊タイプ名を示す修飾名または非修飾名です。特殊タイプ名 の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名 の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名 の後にスラッシュ (/) と SQL ID [18] が続きます。

非修飾形式の特殊タイプ名 は、SQL ID です。非修飾形式は、 72 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

システム命名規則では、特殊タイプ名 は、 SQL ルーチンのパラメーター・データ・タイプ内で使用されている場合、あるいは SQL 関数、SQL プロシージャー、またはトリガーの SQL 変数宣言内で使用されている場合は、修飾することはできません。

外部プログラム名

これは、外部プログラムを指す修飾名、非修飾名、または文字ストリングです。外部プログラム名 の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、システム・スキーマ名 のピリオド (.) とシ

ステム ID が続きます。システム命名規則の場合、修飾形式は、システム・スキーマ名 の後にスラッシュ (/) とシステム ID [18] が続きます。

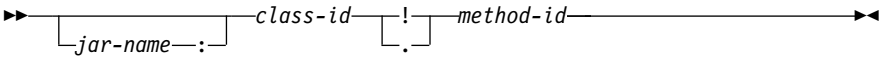
非修飾形式の外部プログラム名 は、システム ID です。非修飾形式は、72 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

サービス・プログラム名の場合、修飾形式は、命名オプションによって異なります。SQL 命名オプションの場合の修飾形式は、*system-schema-name* の後にピリオド (.)、その後にサービス・プログラム名のシステム ID、その後に左括弧、その後に IBM i のエントリー・ポイント名、その後に右括弧 (library-name.service-program-name(entry-point-name)) という形式になります。システム命名オプションの場合の修飾形式は、*system-schema-name* の後にスラッシュ(/)、その後にサービス・プログラム名のシステム ID、その後に左括弧、その後に IBM i のエントリー・ポイント名、その後に右括弧 (library-name/service-program-name(entry-point-name)) [18] という形式になります。エントリー・ポイント名に小文字が含まれている場合は、引用符で囲む必要があります。

サービス・プログラム名の非修飾形式は、システム ID の後に左括弧、その後に IBM i のエントリー・ポイント名、その後に右括弧という形式になります。非修飾形式は、 72 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

文字ストリングの形式は、次のいずれかです。

- IBM i の修飾プログラム名 (「ライブラリー名/プログラム名」)。
- 後に括弧で囲まれた IBM i メンバー名を伴う IBM i 修飾ソース・ファイル名 (「ライブラリー名/ソース・ファイル名 (メンバー名)」)。この形式は、REXX プロシージャーを呼び出す場合にのみ有効です。
- 後に括弧で囲まれた IBM i エントリー・ポイント名を伴う IBM i 修飾サービス・プログラム名 (「ライブラリー名/サービス・プログラム名 (エントリー・ポイント名)」)。
- Java では、オプションの *jar* 名 の後に、クラス ID、その後に感嘆符またはピリオド、その後にメソッド ID を付けたものです (「クラス ID!メソッド ID」または「クラス ID.メソッド ID」)。



jar 名 *jar* 名 は、データベースにインストールされたときの jar スキーマを識別するストリング (大文字小文字の区別あり) です。これは、単純な ID またはスキーマ修飾 ID のどちらも可能です。例えば、'myJar' や 'myCollection.myJar' のようになります。

クラス **ID**

クラス *ID* は、 Java オブジェクトのクラス ID を識別します。クラスが Java パッケージの一部である場合は、クラス ID には完全な Java パッケージ・プレフィックスが含まれていなければなりません。例えば、クラス ID が 'myPackage.StoredProcs' で

ある場合、 Java 仮想計算機は、以下のディレクトリー内で StoredProcs クラスを検索します。

```
'/QIBM/UserData/OS400/SQLLib/
  Function/myPackage/StoredProcs/'
```

メソッド **ID**

メソッド *ID* は、呼び出される公開静的 Java メソッドのメソッド名を識別します。

この形式は、Java プロシージャーおよび Java 関数の場合にのみ有効です。

関数名

ユーザー定義の関数、特殊タイプが作成されたときに生成されたキャスト関数、または組み込み関数を指す修飾名または非修飾名です。関数名 の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名 の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名 の後にスラッシュ (/) と SQL ID [18] が続きます。

CREATE、COMMENT、DROP、GRANT、または REVOKE ステートメントでは、*schema-name* を *server-name* で修飾することができます。他のすべてのコンテキストでは、*server-name* を使用できません。

非修飾形式の関数名 は、SQL ID です。非修飾形式は、 72 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

システム命名規則の場合、CREATE、COMMENT、DROP、GRANT、または REVOKE ステートメントで名前を使用する場合に、関数名をスキーマ名/関数名 の形式でのみ、修飾することができます。式の中では、ピリオドを使用する修飾形式を使用できます。

ホスト・ラベル

ホスト・プログラム内のラベルを指定するトークン。

ホスト変数

ホスト変数を示す一連のトークンです。 172 ページの『ホスト変数に対する参照』で説明されているように、 1 つのホスト変数 は少なくとも 1 つのホスト *ID* を持ちます。

索引名

索引を示す修飾名または非修飾名。索引名 の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名 の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名 の後にスラッシュ (/) と SQL ID [18] が続きます。

非修飾形式の索引名 は、SQL ID です。非修飾形式は、 72 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

mask-name

列マスクを示す修飾名または非修飾名。マスク名 の修飾形式は、命名オプションに依存します。SQL 命名規則の場合、修飾形式は、スキーマ名 の後

にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名 の後にスラッシュ (/) と SQL ID [18] が続きます。

非修飾形式のマスク名 は SQL ID です。非修飾形式は、 72 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

メンバー名
データベース・ファイルのメンバーを指定する非修飾 ID です。メンバーは、パーティション化された表のパーティションでもあります。

ノード・グループ名
ノード・グループを示す修飾名または非修飾名です。ノード・グループは、表が分散される IBM i 製品のグループです。分散表およびノード・グループについての詳細は、「Db2 マルチシステム」を参照してください。

ノード・グループ名 の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名 のピリオド (.) とシステム ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名 の後にスラッシュ (/) とシステム ID [18] が続きます。

非修飾形式のノード・グループ名 は、システム ID です。非修飾形式は、72 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

パッケージ名
パッケージを示す修飾名または非修飾名です。パッケージ名 の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名 のピリオド (.) とシステム ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名 の後にスラッシュ (/) とシステム ID [18] が続きます。

非修飾形式のパッケージ名 は、システム ID です。非修飾形式は、 72 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

パラメーター名
関数またはプロシージャーのパラメーターを示す SQL ID です。プロシージャー用のパラメーター名 の場合、 ID はコロンの後に続くことがあります。

パーティション名
パーティション表のパーティションを示す非修飾 ID です。

permission-name
行の許可を示す修飾名または非修飾名です。許可名 の修飾形式は、命名オプションに依存します。SQL 命名規則の場合、修飾形式は、スキーマ名 の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名 の後にスラッシュ (/) と SQL ID [18] が続きます。

非修飾形式の許可名 は SQL ID です。非修飾形式は、 72 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

プロシージャー名
プロシージャーを指す修飾名または非修飾名です。プロシージャー名 の修

飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名 の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名 の後にスラッシュ (/) と SQL ID [18] が続きます。

非修飾形式のプロシージャー名 は、SQL ID です。非修飾形式は、 72 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

セーブポイント名
: セーブポイントを指す非修飾 ID です。

スキーマ名
: SQL オブジェクトを論理的にグループ化するための修飾名または非修飾名です。スキーマは、表、ビュー、索引、プロシージャー、関数、トリガー、シーケンス、変数、制約、別名、タイプ、またはパッケージの修飾子として使用されます。非修飾形式のスキーマ名 は、システム ID です。スキーマ名 の修飾形式は、命名オプションに依存します。

  SQL 名を使用している場合、SQL ステートメント内の非修飾のスキーマ名は、サーバー名 によって暗黙のうちに修飾されます。修飾形式は、サーバー名 の後にピリオド (.) とシステム ID が続きます。

  システム名を使用している場合、SQL ステートメント内の非修飾のスキーマ名は、サーバー名 によって暗黙のうちに修飾されます。修飾形式は、サーバー名 の後にスラッシュ (/) と SQL ID [18] が続きます。

  server-name を使用してスキーマ名を修飾する場合は、*server-name* でサポート対象のリモート・サーバーを指定することもできます。そうでない場合は、現行サーバーによってスキーマ名が暗黙的に修飾されます。

  注**:** スキーマ名 は、 CREATE SCHEMA ステートメントによって作成されたスキーマ、あるいは IBM i ライブラリーのいずれかを指します。

シーケンス名
: シーケンスを示す修飾名または非修飾名です。シーケンス名 の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名 の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名 の後にスラッシュ (/) と SQL ID [18] が続きます。システム命名では、シーケンス名 は、NEXT VALUE または PREVIOUS VALUE 式で使用されるときにはスラッシュを使用して修飾することはできません (スラッシュを使用する修飾形式は SQL スキーマ・ステートメント内でのみ許可されます)。NEXT VALUE 式または PREVIOUS VALUE 式の中では、ピリオドを使用する修飾形式を使用できます。

  非修飾形式のシーケンス名 は、SQL ID です。非修飾形式は、 72 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

  シーケンス名 では、シーケンスの名前、またはシーケンスのシステム・オブジェクトの名前のいずれかを指定することができます。

サーバー名
: アプリケーション・サーバーを指定する SQL ID。この ID は、文字で開始しなければならず、小文字または特殊文字を含めてはなりません。

サーバー名 は、リレーショナル・データベースの実際の名前か、または、リレーショナル・データベース別名にすることができます。詳しくは、RDB ディレクトリー項目の追加 (ADDRDBDIRE) CL コマンドを参照してください。3 部構成の名前が SQL ステートメント内に直接指定される場合 (CREATE ALIAS ステートメント内に指定される基本表を除く)、その名前は、実際のリレーショナル・データベース名またはリレーショナル・データベース別名のいずれかを使用できます。

例えば、リレーショナル・データベースの実際の名前が ABC で、リレーショナル・データベース別名 MYABC も ABC を参照している場合、次のようになります。

```
SELECT * FROM ABC.SCHEMA1.T1   -- This is valid.

SELECT * FROM MYABC.SCHEMA1.T1 -- This is also valid.
```

特定名

プロシージャーまたは関数を一意的に識別する修飾名または非修飾名です。特定名 の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名 の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名 の後にスラッシュ (/) と SQL ID [18] が続きます。

非修飾形式の特定名 は、SQL ID です。非修飾形式は、 72 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

SQL-condition-name

SQL プロシージャー、SQL 関数、またはトリガー本体内の条件を指定する SQL ID。

SQL 記述子名

ALLOCATE DESCRIPTOR ステートメントを使用して割り振られた SQL 記述子を指定する変数名、文字、またはグラフィック・ストリング定数です。

SQL 記述子の指定に変数が使用されている場合:

- 変数は CLOB または DBCLOB であってはなりません。
- 変数がグラフィック・ストリングである場合は、Unicode グラフィック・ストリングでなければなりません。
- 変数の内容の長さは、*SQL* 記述子名 の最大長を超えることはできません。
- この変数に、標識変数を指定してはなりません。:ホスト変数:標識変数という形式は使用できません。
- 変数の内容には大/小文字の区別があり、大文字に変換されることはありません。

先行ブランクと末尾ブランクは変数またはストリングから削除されます。変数の説明については、 172 ページの『ホスト変数に対する参照』を参照してください。

SQL 記述子の指定にストリング定数が使用されている場合、定数の長さは *SQL* 記述子名 の最大長を超えることはできません。

SQL ラベル
SQL プロシージャー、SQL 関数、またはトリガー本体のラベルを示す非修飾名です。*SQL* ラベル は、SQL ID です。

SQL パラメーター名
SQL ルーチン本体のパラメーターを示す修飾名または非修飾名です。非修飾形式の *SQL* パラメーター名 は、SQL ID です。修飾形式は、プロシージャー名 の後にピリオド (.) と SQL ID が続きます。

SQL 変数名
SQL ルーチン本体の変数を示す修飾名または非修飾名です。非修飾形式の *SQL* 変数名 は、SQL ID です。修飾形式は、 *SQL* ラベル の後にピリオド (.) と SQL ID が続きます。

ステートメント名
準備済み SQL ステートメントを示す SQL ID です。

システム列名
表またはビューの IBM i 列名を示す非修飾名です。システム列名 は、システム ID です。システム列名 は、区切り文字付き ID でも構いませんが、区切り文字で囲まれた内部に小文字や特殊文字が入っていてはなりません。

システム・オブジェクト名
表、ビュー、索引、シーケンス、変数、または別名の IBM i 名を示す非修飾名です。システム・オブジェクト名 は、システム ID です。

表、ビュー、索引、シーケンス、変数、または別名の非修飾名が有効なシステム ID である場合、システム・オブジェクト名 は、表、ビュー、索引、シーケンス、変数、または別名の非修飾名になります。

システム・スキーマ名
スキーマの IBM i 名を指定する非修飾名です。システム・スキーマ名 は、システム ID です。

スキーマの非修飾名が有効なシステム ID である場合、システム・スキーマ名 は、スキーマの非修飾名になります。

表名
表を示す修飾名または非修飾名です。表名 の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名 の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名 の後にスラッシュ (/) と SQL ID [18] が続きます。

非修飾形式の表名 は、SQL ID です。非修飾形式は、 72 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

表名 では、表の名前、または表のシステム・オブジェクトの名前のいずれかを指定することができます。

トリガー名
表に対するトリガーを指定する修飾または非修飾の名前です。トリガー名の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名 のピリオド (.) とシステム ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名 の後にスラッシュ (/) と SQL ID [18] が続きます。

非修飾形式のトリガー名 は、SQL ID です。非修飾形式は、 72 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

変数名

グローバル変数を指定する修飾名または非修飾名。変数名 の修飾形式は、命名オプションに依存します。SQL 命名規則の場合、修飾形式は、スキーマ名 の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名 の後にスラッシュ (/) と SQL ID [18] が続きます。システム命名では、変数名 は、式の中で使用されるときにはスラッシュを使用して修飾することはできません (スラッシュを使用する修飾形式は SQL スキーマ・ステートメント内でのみ許可されます)。式の中では、ピリオドを使用する修飾形式を使用できます。

非修飾形式の変数名 は、SQL ID です。非修飾形式は、 72 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

変数名 では、変数の名前、または変数のシステム・オブジェクトの名前のいずれかを指定することができます。

バージョン **ID**

パッケージの作成時にパッケージに割り当てる 1 文字から 64 文字の ID です。バージョン *ID* は、Db2 for i 以外のサーバーからパッケージを作成する場合にのみ割り当てられます。

ビュー名

ビューを示す修飾名または非修飾名です。ビュー名 の修飾形式は、命名オプションに依存します。 SQL 命名規則の場合、修飾形式は、スキーマ名 の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名 の後にスラッシュ (/) と SQL ID [18] が続きます。

非修飾形式のビュー名 は、SQL ID です。非修飾形式は、 72 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

ビュー名 では、ビューの名前、またはビューのシステム・オブジェクトの名前のいずれかを指定することができます。

XSR オブジェクト名

XML スキーマ・リポジトリーに含まれているオブジェクトを指定する修飾名または非修飾名。*xsrobject-name* の修飾形式は、命名オプションによって異なります。SQL 命名規則の場合、修飾形式は、スキーマ名 の後にピリオド (.) と SQL ID が続きます。システム命名規則の場合、修飾形式は、スキーマ名 の後にスラッシュ (/) と SQL ID [18] が続きます。

xsrobject-name の非修飾形式は、SQL ID です。非修飾形式は、 72 ページの『非修飾オブジェクト名の修飾』で指定された規則に基づいて暗黙的に修飾されます。

表 *5. ID* の長さの制約 *(*バイト数*)*

ID のタイプ	最大の長さ
権限名の最大長[19]	10
相関名の最大長	128

表 *5. ID* の長さの制約 *(*バイト数*) (*続き*)*

ID のタイプ	最大の長さ
カーソル名の最大長	128
外部プログラム名の最大長 (ストリング形式)	279
外部プログラム名の最大長 (非修飾形式)[20]	10
ホスト ID の最大長	64
パッケージ・バージョン ID の最大長	64
パーティション名の最大長	10
セーブポイント名の最大長	128
スキーマ名の最大長	128
サーバー名の最大長	18
SQL 条件名の最大長	128
SQL 記述子名の最大長	128
SQL ラベルの最大長	128
ステートメント名の最大長	128
非修飾の別名の最大長	128
非修飾の配列タイプ名の最大長	128
非修飾の列名の最大長	128
非修飾の制約名の最大長	128
非修飾の特殊タイプ名の最大長	128
非修飾の関数名の最大長	128
非修飾の索引名の最大長	128
非修飾のマスク名の最大長	128
非修飾のノード・グループ名の最大長	10
非修飾のパッケージ名の最大長	10
非修飾の許可名の最大長	128
非修飾のプロシージャー名の最大長	128
非修飾のシーケンス名の最大長	128
非修飾の特定名の最大長	128
非修飾の SQL パラメーター名の最大長	128
非修飾の SQL 変数名の最大長	128
非修飾のシステム列名の最大長	10
非修飾のシステム・オブジェクト名の最大長	10
非修飾のシステム・スキーマ名の最大長	10
非修飾の表名およびビュー名の最大長	128
非修飾のトリガー名の最大長	128
非修飾の変数名の最大長	128
非修飾の XSR オブジェクト名の最大長	128

19. アプリケーション・リクエスターとして、システムは最大 255 バイトまでの権限名を送信できます。

SQL パス

SQL パス とは、スキーマ名が順に並べられたリストです。データベース・マネージャーは、パスを使用して、
ALTER、CREATE、DROP、COMMENT、LABEL、GRANT、または REVOKE ステートメントのメイン・オブジェクト以外として任意のコンテキストに出現する、非修飾のタイプ名 (組み込みタイプ、特殊タイプ、および配列タイプ)、関数名、変数名、およびプロシージャー名のスキーマ名を解決します。

例えば、SQL パスが SMITH、XGRAPHIC、QSYS、QSYS2 で、非修飾の特殊タイプ名 MYTYPE が指定された場合、データベース・マネージャーは MYTYPE を最初にスキーマ SMITH で、次に XGRAPHIC で、その次に QSYS と QSYS2 で探します。

使用される SQL パスは、以下のように SQL ステートメントによって異なります。

- 静的 SQL ステートメント (CALL 変数 ステートメントを除く) の場合、使用されるパスは CRTSQLxxx コマンドの SQLPATH パラメーターの値です。
  SQLPATH は、SET OPTION ステートメントを使用しても設定することができます。

  プログラム、モジュール、サービス・プログラム、ルーチン、およびトリガーに格納されるパスは、そのパス内のスキーマ名と関連する *system-schema-name* 全体によって構成されています。スキーマの *system-schema-name* が名前変更されると、そのパスに依存する SQL ステートメントを使用するオブジェクトを再作成する必要がある場合があります。
- 動的 SQL ステートメントの場合 (さらには、 CALL 変数 ステートメントの場合)、使用されるパスは CURRENT PATH 特殊レジスターの値です。
  CURRENT PATH 特殊レジスターの詳細については、 156 ページの『CURRENT PATH』 を参照してください。

SQL パスを明示的に指定しない場合、 SQL パスは、システム・パスの後にステートメントの実行時権限 ID を付けたものになります。

動的 SQL の SQL パスの詳細については、 156 ページの『CURRENT PATH』を参照してください。

非修飾オブジェクト名の修飾

非修飾オブジェクト名は暗黙的に修飾されます。名前を修飾するための規則は、その名前が識別するオブジェクトのタイプによって異なります。

非修飾の別名、制約名、外部プログラム名、索引名、マスク名、ノード・グループ名、パッケージ名、許可名、シーケンス名、表名、トリガー名、ビュー名、および XSR オブジェクト名

非修飾の別名、制約名、外部プログラム名、索引名、マスク名、ノード・グループ名、パッケージ名、許可名、シーケンス名、表名、トリガー名、ビュー名、および XSR オブジェクト名は、デフォルトのスキーマ によって暗黙的に修飾されます。

20. REXX プロシージャーの場合、その制限は 33 です。

デフォルトのスキーマ は、以下のようにして決まります。

- 静的 SQL ステートメントの場合
  - CRTSQLxxx コマンド (または SET OPTION ステートメント) で DFTRDBCOL パラメーターを指定する場合、デフォルトのスキーマ は、そのパラメーターに指定したスキーマ名 になります。
  - その他の場合のデフォルトのスキーマ は、命名規則に基づきます。
    - SQL 命名規則の場合、デフォルトのスキーマは、そのステートメントの権限 ID になります。
    - システム命名規則の場合、デフォルトのスキーマ は、ジョブ・ライブラリー・リスト (*LIBL) になります。
- 動的 SQL ステートメントの場合、デフォルトのスキーマ は、デフォルトのスキーマ が明示的に指定されているかどうかによって異なります。明示的にこれを指定するメカニズムは、SQL ステートメントを動的に作成し、実行するために使用されるインターフェースにより異なります。
  - デフォルトのスキーマ が明示的に指定されていない場合
    - SQL 命名規則の場合、デフォルトのスキーマは、実行時の権限 ID になります。
    - システム命名規則の場合、デフォルトのスキーマ は、ジョブ・ライブラリー・リスト (*LIBL) になります。
  - デフォルトのスキーマ は、以下のインターフェースによって明示的に指定します。

表 *6.* デフォルトのスキーマ・インターフェース

SQL インターフェース	指定
組み込み SQL	SQL プログラム作成 (CRTSQLxxx) コマンドおよび SQL パッケージ作成 (CRTSQLPKG) コマンドの DFTRDBCOL パラメーターおよび DYNDFTCOL(*YES)。 DFTRDBCOL および DYNDFTCOL の値の設定に SET OPTION ステートメントも使用可能。 (CRTSQLxxx コマンドについて詳しくは、「組み込み SQL プログラミング」を参照してください。)
SQL ステートメント実行	SQL ステートメント実行 (RUNSQLSTM) コマンドの DFTRDBCOL パラメーター。 (RUNSQLSTM コマンドの詳細については、「SQL プログラミング」を参照。)
サーバー上の呼び出しレベル・インターフェース (CLI)	SQL_ATTR_DEFAULT_LIB または SQL_ATTR_DBC_DEFAULT_LIB 環境変数または接続変数。 (CLI について詳しくは、「SQL 呼び出しレベル・インターフェース (ODBC)」を参照してください。)
IBM IBM Developer Kit for Java を使用したサーバーの JDBC または SQLJ	ライブラリー特性オブジェクト (JDBC および SQLJ について詳しくは、「IBM Developer Kit for Java」を参照してください。)

表 6. デフォルトのスキーマ・インターフェース (続き)

SQL インターフェース	指定
IBM i Access Family ODBC ドライバーを使用したクライアントの ODBC	ODBC セットアップ内の SQL デフォルト・スキーマ (ODBC の詳細については、「IBM i Access」を参照。)
IBM Toolbox for Java を使用したクライアントの JDBC	JDBC セットアップ内の SQL デフォルト・スキーマ (JDBC の詳細については、「IBM i Access」を参照。) (IBM Toolbox for Java について詳しくは、「IBM Toolbox for Java」を参照してください。)
IBM i Access Family OLE DB Provider を使用したクライアントの OLE DB	接続オブジェクト・プロパティー内の DefaultCollection (OLE DB の詳細については、「IBM i Access」を参照。)
IBM i Access FamilyADO .NET プロバイダーを使用しているクライアントの ADO .NET	接続オブジェクト・プロパティー内の DefaultCollection (ADO .NET の詳細については、「IBM i Access」を参照。)
すべてのインターフェース	SET SCHEMA または QSQCHGDC (動的デフォルト・コレクション変更) API (QSQCHGDC の詳細については、ファイル API のカテゴリーを参照。)

非修飾の関数、プロシージャー、特定名、タイプ、および変数

関数、プロシージャー、特定名、タイプ (組み込みタイプ、特殊タイプ、および配列タイプ)、および変数の修飾は、非修飾の名前が使われている SQL ステートメントによって異なります。

- 非修飾名が CREATE、COMMENT、LABEL、DROP、GRANT、または REVOKE ステートメントのメイン・オブジェクトの場合は、非修飾の表名の修飾と同じ規則を使用して暗黙的に名前が修飾されます (72 ページの『非修飾の別名、制約名、外部プログラム名、索引名、マスク名、ノード・グループ名、パッケージ名、許可名、シーケンス名、表名、トリガー名、ビュー名、および XSR オブジェクト名』を参照してください)。
- それ以外の場合は、暗黙的なスキーマ名は以下のようにして決められます。
  - タイプ名の場合、データベース・マネージャーは SQL パスを検索し、そのデータ・タイプが存在するような、パス上の最初のスキーマを選択する。
  - 変数名の場合、データベース・マネージャーは SQL パスを検索し、同じ名前を持つ、権限が与えられた変数が含まれるような、パス内の最初のスキーマを選択する。
  - プロシージャー名の場合、データベース・マネージャーは SQL パスを検索し、同じ名前とパラメーター数を持つ、権限が与えられたプロシージャーが含まれるような、パス内の最初のスキーマを選択する。
  - 関数名の場合、データベース・マネージャーは、 185 ページの『関数解決』で説明しているように、関数解決と連携して SQL パスを使用する。
  - ソースとなる関数に指定された特定名については、 1122 ページの『CREATE FUNCTION (ソース派生)』を参照してください。

SQL 名とシステム名: 特殊な考慮事項

CL コマンドのデータベース・ファイル一時変更 (OVRDBF) を指定すると、ローカル・データ操作用の SQL ステートメントについて、SQL 名またはシステム名を他のオブジェクト名に一時変更することができます。この一時変更は、データ定義用の SQL ステートメントおよびリモートのリレーショナル・データベースで実行されるデータ操作用 SQL ステートメントについては、無視されます。

一時変更関数の詳細については、データベース・ファイル管理のトピックを参照してください。

別名

別名 は、表、表のパーティション、ビュー、またはデータベース・ファイルのメンバーの代替名と考えてください。名前または別名によって、SQL ステートメントで表やビューを参照することができます。別名は、同一またはリモートのリレーショナル・データベース内の表、表のパーティション、ビュー、またはデータベース・ファイルのメンバーを参照することができます。

別名は、表名やビュー名を使用できる場所であれば基本的にどこでも使用できますが、以下の例外があります。

- CREATE TABLE または CREATE VIEW ステートメントのように新規の表名またはビュー名の場合は、別名を使用しないでください。例えば、PERSONNEL という別名が作成された場合、CREATE TABLE PERSONNEL のような後続のステートメントはエラーになります。
- 表の個々のパーティションまたはデータベース・ファイルのメンバーを参照している別名は、選択ステートメント、CREATE INDEX、DELETE、INSERT、MERGE、SELECT INTO、SET 変数、UPDATE、または VALUES INTO ステートメントでのみ使用することができます。

別名によって、ファイルの一時変更を避けることもできます。別名の方が一時変更よりも都合がよいだけではなく、別名は一度だけ作成すればよい永続オブジェクトでもあります。

別名が参照するオブジェクトが存在しない場合でも、別名を作成することはできます。ただし、別名を参照するステートメントが実行される時点では、そのオブジェクトは存在している必要があります。オブジェクトが存在しない場合に別名を作成すると、警告が戻されます。ある別名が別の別名を参照することはできません。

3 部構成の名前を使用し、分散データを参照するステートメントは、結果として、リモート・リレーショナル・データベースへの DRDA アクセスを行います。アプリケーション・プログラムがリモート・オブジェクトと DRDA アクセスに 3 部構成の名前の別名を使用する場合、アプリケーション・プログラムは、3 部構成の名前に指定されているロケーションごとにバインドする必要があります。また、各別名はローカル・サイトで定義する必要があります。リモート・サイトでの別名は、参照される別名が結果的に表またはビューを参照している限り、さらに別のサーバーを参照することができます。

別名によって表、表のパーティション、ビュー、またはデータベース・ファイルのメンバーを参照するというオプションは、明示的に構文図に示されることはありません。また、SQL ステートメントの説明で記述されることもありません。

新規の別名は、既存の表、ビュー、索引、ファイル、または別名と同じ完全修飾名を持つことはできません。

SQL ステートメントで別名を使用する効果は、テキスト置換の効果と似ています。別名は、SQL ステートメントの実行前に定義しておく必要がありますが、修飾された基本表名、表のパーティション名、ビュー名、またはデータベース・ファイルのメンバー名によってステートメントの準備時間に置き換えられます。例えば、PBIRD.SALES が DSPN014.DIST4_SALES_148 の別名である場合、次のステートメントの実行時には、

```
SELECT * FROM PBIRD.SALES
```

次のようになります。

```
SELECT * FROM DSPN014.DIST4_SALES_148
```

別名をいったん除去して、別の表を参照する別名を再作成するときの効果は、その別名を参照するステートメントによって異なります。

- その別名を参照している SQL データ・ステートメントまたは SQL データ変更ステートメントは、次の実行時に暗黙的に再バインドされます。
- その別名を参照する索引は影響を受けません。
- その別名を参照するマテリアライズ照会表またはビューは影響を受けません。

既存の Db2 for z/OS アプリケーションの構文を使えるようにするには、CREATE ALIAS および DROP ALIAS ステートメントで、ALIAS の代わりに SYNONYM を使用することができます。

権限 ID と権限名

許可 *ID* とは、データベース・マネージャーとアプリケーション・プロセスとの間、またはデータベース・マネージャーとプログラム準備処理との間の接続が確立されるときに、データベース・マネージャーが獲得する文字ストリングのことです。権限 ID は、特権の集合を示します。権限 ID がユーザーまたはユーザーのグループを示すこともありますが、データベース・マネージャーでは、権限 ID のこの特性は管理しません。

接続が確立された後で、SET SESSION AUTHORIZATION ステートメントを使用して権限 ID を変更することができます。

権限 ID は、データベース・マネージャーで SQL ステートメントの権限検査に使用されます。

ステートメントの権限 ID がプログラムまたはサービス・プログラムの所有者である場合、これは借用権限と呼ばれます。この権限 ID は、実行時の権限 ID に加えて、プログラム所有者の権限を借用し、使用します。

ステートメント権限 ID としてプログラムの所有者を使用するプログラムまたはサービス・プログラムのネストされた呼び出しが起こる場合、複数レベルの借用権限が可能です。借用権限はスレッド・レベルでトラッキングされます。CURRENT USER 特殊レジスターを使用して、最新の借用権限の権限 ID を戻すことができます。ステートメント権限要件の適用では、すべてのレベルの借用権限が考慮されます。DYNUSRPRF 値を *USER に設定して動的 SQL ステートメントが実行されるときには、すべてのレベルの借用権限が抑止されます。

権限 ID は、すべての SQL ステートメントに適用されます。静的 SQL ステートメントの許可検査に使用される権限 ID は、プリコンパイラーのコマンドで指定された USRPRF の値によって、以下のように異なります。

- USRPRF(*OWNER) が指定される場合、または USRPRF(*NAMING) が指定され、SQL 命名モードが使用される場合は、ステートメントの権限 ID は、非分散 SQL プログラムの所有者です。分散 SQL プログラムの場合は、SQL パッケージの所有者です。
- USRPRF(*USER) が指定される場合、または USRPRF(*NAMING) が指定され、システム命名モードが使用される場合は、ステートメントの権限 ID は、非分散 SQL プログラムを実行するユーザーの権限 ID です。分散 SQL プログラムの場合は、現行サーバーのユーザーの権限 ID です。

動的 SQL ステートメントの許可検査に使用される権限 ID も、そのステートメントの実行される場所と方法によって次のように異なります。

- 非分散プログラムで準備され実行されるステートメントの場合
  - そのプログラムの USRPRF の値が *USER で、DYNUSRPRF の値が *USER である場合は、適用される権限 ID は、その非分散プログラムを実行するユーザーの権限 ID。これは、実行時権限 *ID* と呼ばれる。
  - そのプログラムの USRPRF の値が *OWNER で、DYNUSRPRF の値が *USER である場合は、適用される権限 ID は、その非分散プログラムを実行するユーザーの権限 ID。

  - そのプログラムの USRPRF の値が *OWNER で、DYNUSRPRF の値が *OWNER である場合は、適用される権限 ID は、その非分散プログラムの所有者の権限 ID。
- 分散プログラムで準備および実行されるステートメントの場合
  - その SQL パッケージの USRPRF の値が *USER で、DYNUSRPRF の値が *USER である場合は、適用される権限 ID は、現行サーバーでその SQL パッケージを実行するユーザーの権限 ID。この権限 ID も、実行時権限 ID と呼ばれる。
  - その SQL パッケージの USRPRF の値が *OWNER で、DYNUSRPRF の値が *USER である場合は、適用される権限 ID は、現行サーバーでその SQL パッケージを実行するユーザーの権限 ID。
  - その SQL パッケージの USRPRF の値が *OWNER で、DYNUSRPRF の値が *OWNER である場合は、適用される権限 ID は、現行サーバーのその SQL パッケージの所有者の権限 ID。
- 対話式で出されるステートメントの場合、SQL 開始 (STRSQL) コマンドを出したユーザーの ID が、適用される権限 ID になります。
- RUNSQLSTM コマンドにより実行されるステートメントの場合、RUNSQLSTM コマンドを出したユーザーの ID が、適用される権限 ID になります。
- ステートメントが REXX から実行される場合、適用される権限 ID は、STRREXPRC コマンドを出したユーザーの ID です。

IBM i オペレーティング・システムでは、実行時権限 ID はスレッドのユーザー・プロファイルです。

SQL ステートメントで指定される 権限名 とステートメントの権限 ID を混同してはなりません。権限名は、GRANT や REVOKE ステートメントで使用される ID で、認可や取り消しの対象を指します。*X* に特権を付与する前提は、 *X* がそれらの特権を必要とするステートメントの権限 ID であることです。 SQL ステートメントに関する権限を検査するときには、グループ・ユーザー・プロファイルが使用されることもあります。グループ・ユーザー・プロファイルについては、「機密保護解説書」を参照してください。

例

SMITH というユーザー ID を持つユーザーがいるとします。このユーザーが次のようなステートメントを対話式で実行する場合は、SMITH が権限 ID となります。

```
GRANT SELECT ON TDEPT TO KEENE
```

SMITH は、このステートメントの権限 ID です。したがって、このステートメントを実行する権限が SMITH にあるかどうかが検査されます。

KEENE は、ステートメントに指定されている権限名です。KEENE には、SMITH.TDEPT に対する SELECT 特権が与えられます。

プロシージャー解決

プロシージャー呼び出しがあると、Db2 は同じ名前を持つプロシージャーのうちのどれを実行するのかを決定する必要があります。

- プロシージャー呼び出しの引数の数を *A* とします。
- プロシージャー・シグニチャー中のパラメーターの数を *P* とします。
- デフォルトを持たないパラメーターの数を *N* とします。

プロシージャー呼び出しの解決に対する候補プロシージャーは、以下の基準で選択されます。

- 各候補プロシージャーは、一致する名前と適用可能なパラメーター数を持つ。適用可能なパラメーター数とは、$N \le A \le P$ という条件を満たすパラメーター数のことです。
- 各候補プロシージャーには、CALL ステートメントに含まれる名前付き引数ごとに、名前が一致して定位置 (つまり名前なし) 引数にまだ合致していないパラメーターが存在する。
- 候補プロシージャーのパラメーターのうち、対応引数が CALL ステートメントで位置でも名前でも指定されていない各パラメーターは、デフォルトを使用して定義される。
- 1 つ以上のスキーマからなる集合からの各候補プロシージャーは、 CALL ステートメントの権限 ID と関連付けられた EXECUTE 特権を持つ。デフォルト式の中で参照されるオブジェクトの権限は考慮されません。

さらに、候補プロシージャーのセットは、プロシージャー名がどのように修飾されているのかに基づきます。

- プロシージャー名が修飾されていない場合、プロシージャー解決は次のように行われます。

  SQL パス内のスキーマを持つすべてのプロシージャーから候補プロシージャーを検索します。 SQL パスのスキーマで 1 つ以上の候補プロシージャーが見つかると、それらの候補プロシージャーが候補リストに入れられます。リスト中の候補プロシージャーが 1 つの場合、解決は完了します。複数の候補プロシージャーがある場合、 SQL パス内で最も早く出現するスキーマを持つプロシージャーを選択します。まだ複数の候補プロシージャーがある場合、パラメーター数が最も少ない候補プロシージャーを選択します。

  候補プロシージャーがない場合は、エラーが戻されます。
- プロシージャー名が修飾されている場合、プロシージャー解決は次のように行われます。

  修飾子によって指定されているスキーマ内で候補プロシージャーを検索します。単一の候補プロシージャーが存在すれば、解決が完了します。複数の候補プロシージャーがある場合、パラメーター数が最も少ない候補プロシージャーを選択し、解決は完了します。スキーマが存在しないか、権限がある候補プロシージャーがない場合、エラーが戻されます。

例 *1:* 4 つの異なるスキーマに 6 個の FOO プロシージャーがあり、以下のように登録されているとします (必須キーワードの一部は省略されています)。

```
CREATE PROCEDURE AUGUSTUS.FOO (INT) SPECIFIC FOO_1 ...
CREATE PROCEDURE AUGUSTUS.FOO (DOUBLE, DECIMAL(15, 3)) SPECIFIC FOO_2 ...
CREATE PROCEDURE JULIUS.FOO (INT) SPECIFIC FOO_3 ...
CREATE PROCEDURE JULIUS.FOO (INT, INT, INT) SPECIFIC FOO_4 ...
CREATE PROCEDURE CAESAR.FOO (INT, INT) SPECIFIC FOO_5 ...
CREATE PROCEDURE NERO.FOO (INT,INT) SPECIFIC FOO_6 ...
```

以下のようにプロシージャーが参照されるとします (I1 および I2 は INTEGER 値です)。

```
CALL FOO(I1, I2)
```

この参照を行うアプリケーションの SQL パスが次のようになっているとします。

```
"JULIUS", "AUGUSTUS", "CAESAR"
```

スキーマ「NERO」は SQL パスに含まれていないため、アルゴリズムに従って、特定名 FOO_6 のプロシージャーは候補から除かれます。パラメーターの数が違うため、FOO_1、FOO_3、および FOO_4 は候補から除かれます。残った候補は順番に考慮され、SQL パスにより判別します。引数およびパラメーターのタイプは無視されることに注意してください。 FOO_5 のパラメーターは CALL の引数と正確に一致しますが、 SQL パスで "CAESAR" の前に "AUGUSTUS" が現れるため FOO_2 が選ばれます。

例 *2:* 次の例は、 CALL ステートメントで名前付きパラメーターを使用するプロシージャー解決を示します。

```
CREATE PROCEDURE p1(i1 INT)...
CREATE PROCEDURE p1(i1 INT DEFAULT 0, i2 INT DEFAULT 0)...

CALL p1(i2=>1)
```

候補選択プロセスでは引数の名前が考慮に入れられるため、2 番目のバージョンの p1 のみが候補であると見なされます。さらに、これは正常に呼び出されます。このバージョンの p1 の i1 はデフォルトを指定して定義されているので、p1 の呼び出しで i2 のみを指定しても有効だからです。このプロシージャーのパラメーター i1 には値 0 が渡されます。

```
CREATE PROCEDURE p2(i1 INT, i2 INT DEFAULT 0)...
CREATE PROCEDURE p2(i1 INT DEFAULT 0, i2 INT DEFAULT 0, i3 INT DEFAULT 0)...

CALL p2(i2=>1)
```

CALL ステートメントに (位置または名前で指定された) 対応する引数がないプロシージャー・パラメーターに関する基準の 1 つは、そのパラメーターがデフォルト値を指定して定義されることです。したがって、最初のバージョンの p2 は、パラメーター i1 にデフォルト値が定義されていないため、候補と見なされません。2 番目のバージョンの p2 が選択され、第 1 パラメーターおよび第 3 パラメーターのデフォルト値が渡されます。

データ・タイプ

SQL で操作できるデータの最小単位を値 と呼びます。

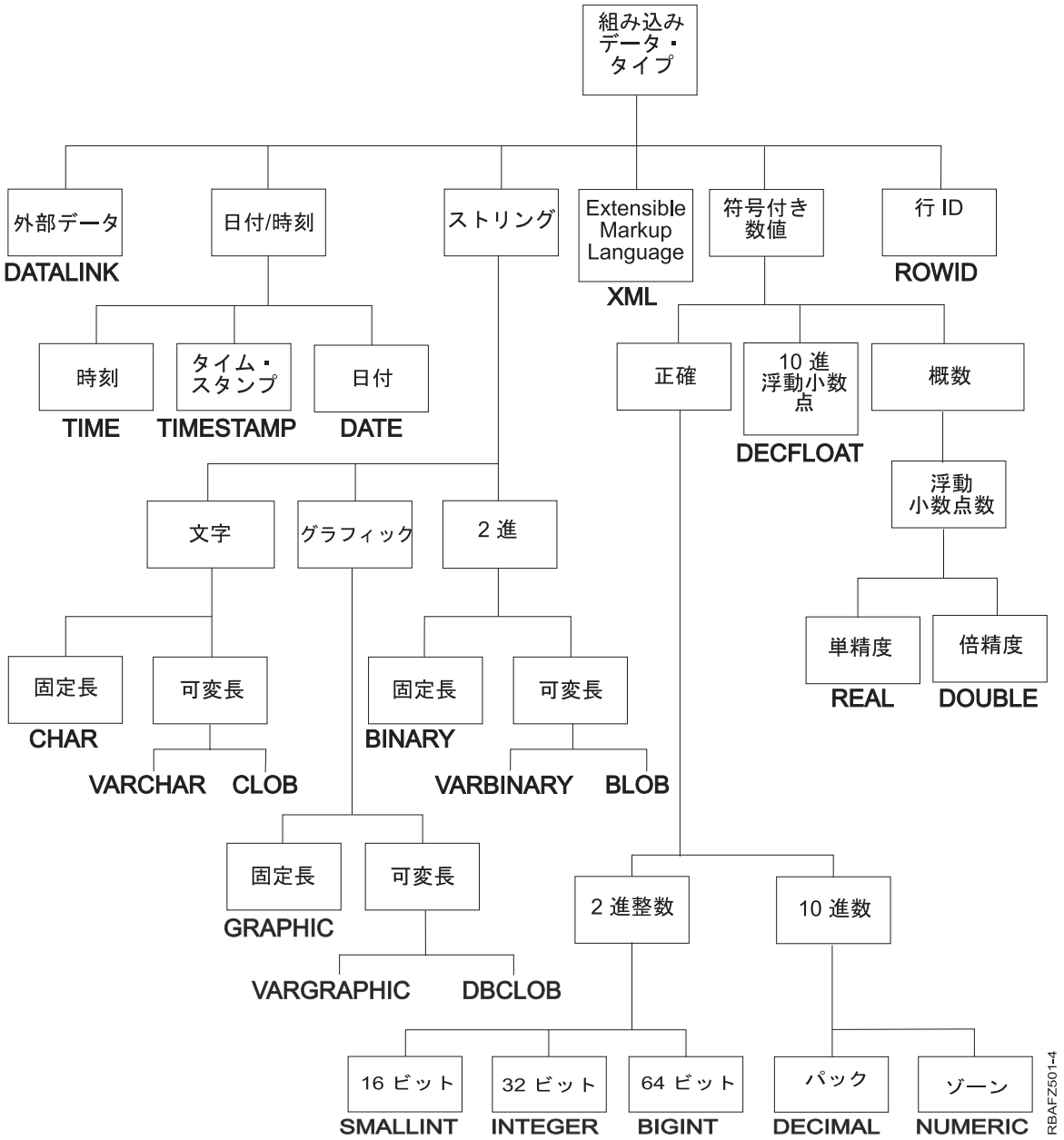
値の解釈の方法は、値のソースの属性 (データ・タイプ、長さ、精度、位取り、CCSID など) によって異なります。値のソースには、以下のものがあります。

- 列

- 定数
- 式
- 関数
- 特殊レジスター
- 変数 (ホスト変数、SQL 変数、グローバル変数、ルーチンのパラメーター・マーカー、パラメーターなど)

Db2 リレーショナル・データベース製品は、組み込みデータ・タイプとユーザー定義データ・タイプの両方をサポートしています。このセクションでは、組み込みデータ・タイプについて説明します。特殊タイプの説明については、 104 ページの『ユーザー定義タイプ』を参照してください。

次の図には、Db2 for i でサポートされる種々の組み込みデータ・タイプを示してあります。



列のデータ・タイプの指定に関する詳しい説明は、 1238 ページの『CREATE TABLE』を参照してください。

NULL

どのようなデータ・タイプにも NULL 値が含まれます。 NULL 値は、NULL 以外のすべての値から NULL を区別する特殊な値であり、それによって値 (NULL 以外の) の不在を示します。

グループ化操作以外では、NULL 値は他の NULL 値からも区別されます。 NULL 値はすべてのデータ・タイプに含まれますが、値のソースによっては NULL 値を提供できないものがあります。例えば、NOT NULL として定義されている定数と列には、NULL 値を含めることはできません。また、COUNT 関数および

COUNT_BIG 関数は NULL 値を戻すことはできません。さらに、照会の結果として ROWID 列に NULL 値が戻されることがありますが、ROWID 列には NULL 値を保管することはできません。

数値

数値データ・タイプは、2 進整数、10 進数、10 進浮動小数点数、および浮動小数点数です。

数値データ・タイプは次のように分類されます。

- 厳密な数値: 2 進整数および 10 進数
- 10 進浮動小数点数
- 近似の数値: 浮動小数点

2 進整数には、短整数、長整数、64 ビット整数が含まれます。 2 進数は、整数の厳密な表現です。 10 進数は、精度と位取りを固定して厳密に表記した数です。2 進数と 10 進数は、厳密な数値タイプと見なされます。

10 進浮動小数点数の精度は 16 または 34 にすることができます。10 進浮動小数点数は、実数の厳密な表記値と実数の近似値の両方をサポートするので、厳密な数値タイプとも近似の数値タイプとも見なされません。

浮動小数点数には、単精度と倍精度があります。 浮動小数点数は実数の近似値であり、近似の数値タイプと見なされます。

すべての数値には、符号、精度、位取り があります。 10 進浮動小数点数を除くどの数値でも、列または式がゼロであれば、符号は正になります。10 進浮動小数点数には、負と正のゼロが含まれます。10 進浮動小数点数には、数の特殊値があり、各種指数と同じ数 (例えば、0.0、0.00、0.0E5、1.0、1.00、1.0000) があります。精度は、符号を除いた総桁数です。位取りは、小数点の右側の総桁数です。小数点がない場合は、位取りはゼロになります。

短整数

短整数 は、5 桁の精度を持つ 2 バイトで構成される 2 進数です。短精度整数の範囲は、-32 768 から +32 767 までです。

短整数では、10 進数の精度と位取りは、COBOL、RPG、および IBM i のシステム・ファイルによってサポートされます。2 進整数の精度と位取りに関しては、「DDS 解説書」トピックを参照してください。

大整数

長整数 は、10 桁の精度を持つ 4 バイトで構成される 2 進数です。長精度整数の範囲は、-2 147 483 648 から +2 147 483 647 までです。

長整数では、10 進数の精度と位取りは、COBOL、RPG、および IBM i のシステム・ファイルによってサポートされます。2 進整数の精度と位取りに関しては、「DDS 解説書」トピックを参照してください。

64 ビット整数

64 ビット整数 は、19 桁の精度を持つ 8 バイトで構成される 2 進数です。 64 ビット整数の範囲は、-9 223 372 036 854 775 808 から +9 223 372 036 854 775 807 までです。

10 進数

10 進数 の値は、暗黙の小数点を持つパック 10 進数またはゾーン 10 進数です。小数点の位置は、その数値の精度および位取りによって決まります。位取り (数値の小数部の桁数) を負の数にしたり、精度より大きい数にすることはできません。最大精度は 63 桁です。

1 つの 10 進数の列にある値は、すべて同一の精度および位取りを持ちます。 10 進変数や 10 進数列の中の数値の範囲は、*-n* から *+n* までです。ここで、*n* の絶対値は、適用可能な精度および範囲で表すことができる最大の数値です。

この場合、最大の範囲は、$-10^{63}+1$ から $10^{63}-1$ です。

浮動小数点

単精度浮動小数点 数は、実数を 32 ビットの概数で表したものです。絶対値の範囲は、およそ $1.17549436 \times 10^{-38}$ から $3.40282356 \times 10^{38}$ までです。

倍精度浮動小数点 数は、実数を IEEE 64 ビットの概数で表したものです。絶対値の範囲は、およそ $2.2250738585072014 \times 10^{-308}$ から $1.7976931348623158 \times 10^{308}$ までです。

単精度浮動小数点は、一般に 7 桁の精度です。倍精度浮動小数点は、一般に 15 桁の精度です。

10 進浮動小数点数

10 進浮動小数点 数は、小数点を持つ IEEE 754R 数です。小数点の位置は、各 10 進浮動小数点値に保管されます。最大精度は 34 桁です。 10 進浮動小数点数の範囲は、16 または 34 桁の精度であり、それぞれ 10^{-383} から 10^{384}、または 10^{-6143} から 10^{6144} までの指数範囲です。

DECFLOAT 値の最小指数 E_{min} は、DECFLOAT(16) の場合は -383、DECFLOAT(34) の場合は -6143 です。DECFLOAT 値の最大指数 E_{max} は、DECFLOAT(16) の場合は 384、DECFLOAT(34) の場合は 6144 です。

有限数に加えて、10 進浮動小数点数は、次の 3 つの特殊値を表すこともできます (詳しくは、 142 ページの『10 進浮動小数点定数』を参照)。

- 無限大 - 絶対値が無限に大きい数を表す値。
- 静止 NaN - 無効数警告を生じない、未定義結果を表す値。
- シグナル NaN - 数値操作で使用される場合に無効数警告が生じる未定義結果を表す値。[21]

21. この警告が戻されるのは、SQL_DECFLOAT_WARNINGS 照会オプションに対して *YES が指定されている場合のみです。

これらの特殊値のいずれかが数値にある場合、その係数と指数は未定義です。無限大の記号が重要です (つまり、正の無限大と負の無限大の両方がある可能性があります)。算術演算にとって、NaN の符号には意味はありません。

詳しくは、 1848 ページの表 120を参照してください。

数値変数

2 進短整数および 2 進長整数の変数は、すべてのホスト言語で使用することができます。64 ビット整数の変数は、C、C++、ILE COBOL、および ILE RPG のみで使用することができます。浮動小数点変数は、RPG/400® および COBOL/400 を除くすべてのホスト言語で使用することができます。10 進変数は、サポートされているすべてのホスト言語で使用することができます。10 進浮動小数点変数は C のみで使用できます。

数値のストリング表現

10 進数、10 進浮動小数点数、または浮動小数点数を (CAST 指定などによって) ストリングにキャストする場合は、暗黙の小数点が、そのステートメントの作成時に有効になっていたデフォルト小数点文字に置き換えられます。ストリングを (CAST 指定などによって) 10 進数、10 進浮動小数点数、または浮動小数点数にキャストする場合は、そのステートメントの作成時に有効になっていたデフォルト小数点文字に基づいて、ストリングが解釈されます。

非正規化数とアンダーフロー

10 進浮動小数点データ・タイプには、通常の 10 進浮動小数点値の範囲外にある、1 組の非ゼロ数値があります。これらの数値は非正規化数と呼ばれます。

調整された指数が E_{min} (DECFLOAT(34) の場合は -6143、DECFLOAT(16) の場合は -383) より小さい非ゼロ数は、非正規化数と呼ばれます。これらの非正規化数は、すべての演算のオペランドとして受け入れられ、どの演算からも生じる可能性があります。丸めの前の結果が非正規化数である場合、非正規化警告が戻されます。[22]

正規化されていない結果の場合、指数の最小値は E_{min}- (precision - 1) になり、E_{tiny} と呼ばれます。この場合、「precision」は 10 進浮動小数点数の精度です。したがって、指数 E_{tiny} の最小値は、-6176 (DECFLOAT(34) の場合) および -398 (DECFLOAT(16) の場合) になります。指数 E_{tiny} が小さくなると、小数部で使用可能な桁数が減少します。非正規化数の小数部で使用可能な桁数は、precision - (-E_{tiny} + E_{min}) です。

指数が E_{tiny} より小さくならないように、必要に応じて結果が丸められます。この丸め時に結果が不正確になる場合、アンダーフロー警告が戻されます。[22]非正規化の結果は、常にアンダーフロー警告を戻すのではなく、常に非正規化警告を戻します。

計算中に数値がゼロにアンダーフローすると、その指数は E_{tiny} になります。指数の最大値は影響を受けません。

22. この警告が戻されるのは、SQL_DECFLOAT_WARNINGS 照会オプションに *YES が指定されている場合のみです。

非正規化数の指数の最大値は、結果が非正規化数にならない演算時に発生する可能性がある指数の最小値と同じです。これが生じるのは、10 進数内の係数の長さが精度と等しい場合です。

文字ストリング

文字ストリング は、一連のバイトです。ストリングの長さとは、そのストリングのバイト数を指します。長さがゼロの場合、その値は空ストリング と呼ばれます。この空ストリングと NULL 値を混同しないように注意してください。

固定長文字ストリング

固定長文字ストリングの特殊タイプ、列、および変数を定義するときには、長さ属性が指定され、すべての値が同じ長さになります。固定長文字ストリングの長さ属性の範囲は 1 から 32766 です。詳しくは、 1845 ページの『付録 A. SQL の制約』を参照してください。

可変長文字ストリング

可変長文字ストリングのタイプは以下のとおりです。

- VARCHAR
- CLOB

文字ラージ・オブジェクト (CLOB) 列は、大容量の文字データ (単一の文字セットを使用して作成された文書など) の格納に便利です。

特殊タイプ、列、変数には、いずれも長さ属性があります。可変長文字ストリングの特殊タイプ、列、および変数を定義するときには、最大長が指定され、これが長さ属性になります。実際の値は、これより小さな長さになる場合があります。可変長文字ストリングの長さ属性の範囲は 1 から 32 740 です。CLOB ストリングの場合、長さ属性の範囲は 1 から 2 147 483 647 までででなければなりません。詳しくは、 1845 ページの『付録 A. SQL の制約』を参照してください。

長い可変長ストリングを使用する際の制約事項については、 93 ページの『ストリングの使用に関する制限』を参照してください。

文字ストリング変数

- 固定長文字ストリング変数は、REXX および Java を除くすべてのホスト言語で使用することができます。(C または C++ 言語では、固定長文字ストリング変数は、その長さが 1 に限定されます。)
- VARCHAR 可変長文字ストリング変数は、C、C++、COBOL、PL/I、REXX、および RPG で使用することができます。
  - PL/I、REXX、および ILE RPG では、可変長文字ストリングのデータ・タイプがあります。
  - COBOL、C、および C++ では、可変長文字ストリングを構造体として表します。
  - C および C++ では、可変長文字ストリング変数は、NUL 終了ストリングによって表すこともできます。

  - RPG/400 では、可変長文字ストリング変数は、外部記述データ構造の結果として組み込まれる VARCHAR 列によってしか表すことができません。
- CLOB 可変長文字ストリング変数は、REXX、RPG/400、および COBOL/400 を除くすべてのホスト言語で定義することができます。
  - ILE RPG では、CLOB 可変長文字ストリングは SQLTYPE キーワードを使用して宣言されます。
  - その他の言語ではすべて、SQL TYPE IS CLOB 文節が使用されます。

文字コード化スキーム

それぞれの文字ストリングは、さらに以下の 4 つのタイプのいずれかとして定義されます。

ビット・データ
: コード化文字セットに関連付けられていないデータ (したがって、変換が行われることのないデータ)。ビット・データの CCSID は 65535 です。

  注: ビット・データは、文字データの形式です。ビット・データへの割り当ての場合、埋め込み文字はブランクです。2 進データへの割り当ての場合、埋め込み文字は X'00' です。ビット・データには、文字の代わりに、2 進データ・タイプの使用をお勧めします。

SBCS データ
: すべての文字が単一バイトで表現されているデータ。SBCS データ文字ストリングはそれぞれ、関連する CCSID を持っています。 SBCS ストリングは、演算での使用に先立って、必要に応じて異なる CCSID を持つ文字ストリングに変換されます。

MIXED (混合**)** データ
: 1 バイト文字セット (SBCS) の文字と 2 バイト文字セット (DBCS) の文字を混用することができるデータ。混合ストリングはそれぞれ、関連の CCSID を持っています。混合データ文字ストリングは、演算に先立って、必要に応じて異なる CCSID を持つ文字ストリングに変換されます。混合データに DBCS 文字が含まれている場合は、SBCS データに変換することはできません。

Unicode データ
: 1 バイト以上で表現される文字を含んだデータ。各 Unicode 文字ストリングは、UTF-8 でコード化されています。 UTF-8 の CCSID は 1208 です。

データベース・マネージャーは、2 バイト文字のサブクラスを認識しません。また、個々の 2 バイト・コードに特定の意味を割り当てることもありません。ただし、混合データの中では、次に示す 2 つの 1 バイト EBCDIC コードに特殊な意味が割り当てられています。

- X'0E' (「シフトアウト」文字)。一連の 2 バイト・コードの始めを示すのに使用されます。
- X'0F' (「シフトイン」文字)。一連の 2 バイト・コードの終わりを示すのに使用されます。

以下の条件に該当する場合に、データベース・マネージャーは、混合データ文字ストリングの 2 バイト文字を認識します。

- ストリング中の 2 バイト文字は、シフトアウト文字とシフトイン文字の対で囲まれていなければなりません。

  シフトアウト文字とシフトイン文字の対は、ストリングを左から右に読み取ったときに検出されます。X'0E' というコードは、その後に X'0F' が見つかった場合は、シフトアウト文字として認識されますが、見つからない場合は、そのコードは無効です。X'0E' の後で 2 バイト境界上で見つかった最初の X'0F' を、対のシフトイン文字として扱います。2 バイト境界上にない X'0F' は、認識されません。

  シフトアウト文字とシフトイン文字の間にあるバイトの数は偶数でなければなりません。バイトの各対 (2 バイト) が、それぞれ 1 つの 2 バイト文字であると見なされます。ストリング中には、シフトアウト文字とシフトイン文字の対が複数存在しても構いません。

混合データ文字ストリングの長さとは、その合計バイト数です。2 バイト文字については、それぞれ 2 バイトとして数え、シフトアウト文字またはシフトイン文字については、それぞれ 1 バイトとして数えます。

ジョブの CCSID が、DBCS が使用可能であることを示しており、しかも FOR BIT DATA、FOR SBCS DATA、または SBCS CCSID が指定されていない場合は、CREATE TABLE は、文字の列を DBCS 混用フィールドとして作成します。このような列は、SQL ユーザーからは文字フィールドのように見えますが、システム・データベースのサポートは、DBCS 混用フィールドとして扱います。 DBCS 混用フィールドの定義については、「データベース・プログラミング」トピック集を参照してください。

グラフィック・ストリング

グラフィック・ストリング は、2 バイト文字のシーケンスです。このストリングの長さは、その文字の数になります。文字ストリングと同様に、グラフィック・ストリングも空でも構いません。

固定長グラフィック・ストリング

固定長グラフィック・ストリングの特殊タイプ、列、変数を定義するときには、長さ属性を指定します。すべての値は同じ長さになります。固定長グラフィック・ストリングの長さ属性は、1 から 16 383 の範囲でなければなりません。詳しくは、1845 ページの『付録 A. SQL の制約』を参照してください。

可変長 GRAPHIC ストリング

- 可変長グラフィック・ストリングのタイプは以下のとおりです。
- VARGRAPHIC
- DBCLOB

2 バイト文字ラージ・オブジェクト (DBCLOB) 列は、2 バイト文字データ (2 バイト文字セットで記述した文書など) を大量に格納するために使用できます。

特殊タイプ、列、変数には、いずれも長さ属性があります。可変長グラフィック・ストリングの特殊タイプ、列、変数を定義するときには、最大長を指定します。その最大長が長さ属性になります。実際の値は、これより小さな長さになる場合があ

ります。可変長グラフィック・ストリングの長さ属性は、1 から 16 370 の範囲でなければなりません。 DBCLOB ストリングの長さ属性は、1 から 1 073 741 823 の範囲でなければなりません。詳しくは、 1845 ページの『付録 A. SQL の制約』を参照してください。

長い可変長ストリングを使用する際の制約事項については、 93 ページの『ストリングの使用に関する制限』を参照してください。

グラフィック・ストリング変数

- 固定長グラフィック・ストリング変数は、C、C++、ILE COBOL、および ILE RPG で定義することができます。 (C および C++ では、固定長グラフィック・ストリング変数の長さは、1 に限定されます。)

  固定長グラフィック・ストリングの変数は、PL/I、COBOL/400、および RPG/400 では定義できませんが、文字ストリングの変数がファイルの外部定義の GRAPHIC 列からソースに生成された場合は、その文字ストリングの変数は、固定長グラフィック・ストリングの変数と同様に扱われます。
- VARGRAPHIC 可変長グラフィック・ストリング変数は、C、C++、ILE COBOL、REXX、および ILE RPG で定義することができます。
  - REXX および ILE RPG には、可変長グラフィック・ストリングのデータ・タイプがあります。
  - C、C++、および ILE COBOL では、可変長グラフィック・ストリングは構造体として表されます。
  - C および C++ では、可変長グラフィック・ストリング変数は、NUL 終了グラフィック・ストリングによって表すこともできます。
  - 可変長グラフィック・ストリングの変数は、PL/I、COBOL/400、および RPG/400 では定義できませんが、文字ストリングの変数がファイルの外部記述の VARGRAPHIC 列からソースに生成された場合は、その文字ストリングの変数は可変長グラフィック・ストリングの変数と同様に扱われます。
- DBCLOB 可変長文字ストリング変数は、REXX、RPG/400、および COBOL/400 を除くすべてのホスト言語で定義することができます。
  - ILE RPG では、DBCLOB 可変長文字ストリングは SQLTYPE キーワードを使用して宣言されます。
  - その他の言語ではすべて、SQL TYPE IS DBCLOB 文節が使用されます。

グラフィック・コード化スキーム

それぞれのグラフィック・ストリングは、さらに以下の 2 つのタイプのいずれかとして定義されます。

DBCS データ
: すべての文字がそれぞれ、シフトアウト文字もシフトイン文字も含まない 2 バイト文字セット (DBCS) の文字で表されるデータ。

  すべての DBCS グラフィック・ストリングには、2 バイトのコード化文字セットを識別する CCSID があります。DBCS グラフィック・ストリングは、演算での使用に先立って、必要に応じて異なる DBCS CCSID を持つ DBCS グラフィック・ストリングに変換されます。

Unicode データ

2 バイト以上で表現される文字を含んだデータ。各 Unicode グラフィック・ストリングは、UCS-2 または UTF-16 のいずれかでコード化されています。 UCS-2 は UTF-16 のサブセットです。 UCS-2 の CCSID は 13488 です。 UTF-16 の CCSID は 1200 です。

NCHAR、NVARCHAR、および NCLOB は、CCSID 1200 を持つ Unicode グラフィック・データの同義語です。

グラフィック・ストリングの変数が明示的に CCSID のタグを付けられていない場合は、ジョブの CCSID の関連する DBCS CCSID が使用されます。関連する DBCS CCSID が存在しない場合には、変数に 65535 のタグが付けられます。グラフィック・ストリングの変数に暗黙に UTF-16 または UCS-2 の CCSID のタグが付けられることはありません。グラフィックの変数に CCSID のタグを付ける方法については、DECLARE VARIABLE ステートメントを参照してください。

2 進ストリング

2 進ストリング は、一連のバイトです。テキスト・データが通常含まれている文字ストリングとは異なり、2 進ストリングは、ピクチャーのような従来とは異なるデータの保持に使用されます。2 進ストリングの長さとは、そのストリングのバイト数を指します。2 進ストリングは、65535 の CCSID を持っています。2 進ストリングとの互換性があるのは、文字ストリング FOR BIT DATA のみです。

固定長 2 進ストリング

固定長バイナリー・ストリングの特殊タイプ、列、および変数を定義するときには、長さ属性が指定され、すべての値が同じ長さになります。固定長バイナリー・ストリングの長さ属性の範囲は 1 から 32 766 です。詳しくは、 1845 ページの『付録 A. SQL の制約』を参照してください。

可変長 2 進ストリング

可変長 2 進ストリングのタイプは以下のとおりです。

- VARBINARY
- BLOB

バイナリー・ラージ・オブジェクト (BLOB) 列は、文字以外のデータ (画像、音声、混合メディアなど) を大量に格納するために使用できます。もう 1 つの用途は、特殊タイプおよびユーザー定義関数によって使用される構造化データの保管です。

特殊タイプ、列、変数には、いずれも長さ属性があります。可変長バイナリー・ストリングの特殊タイプ、列、および変数を定義するときには、最大長が指定され、これが長さ属性になります。実際の値は、これより小さな長さになる場合があります。可変長バイナリー・ストリングの長さ属性の範囲は 1 から 32 740 バイトです。BLOB ストリングの長さ属性は、1 から 2 147 483 647 までの範囲でなければなりません。詳しくは、 1845 ページの『付録 A. SQL の制約』を参照してください。

2 進ストリング変数

2 進ストリング・タイプを持つ変数は、REXX、RPG/400、および COBOL/400 を除くすべてのホスト言語で定義することができます。

- BINARY 固定長 2 進ストリング変数は、REXX、RPG/400、および COBOL/400 を除くすべてのホスト言語で定義することができます。
  - ILE RPG では、BINARY 固定長 2 進ストリング変数は SQLTYPE キーワードを使用して宣言されます。
  - その他の言語ではすべて、SQL TYPE IS BINARY 文節が使用されます。
- VARBINARY 可変長 2 進ストリング変数は、REXX、RPG/400、および COBOL/400 を除くすべてのホスト言語で定義することができます。
  - ILE RPG では、VARBINARY 可変長 2 進ストリング変数は SQLTYPE キーワードを使用して宣言されます。
  - その他の言語ではすべて、SQL TYPE IS VARBINARY 文節が使用されます。
- BLOB 可変長 2 進ストリング変数は、REXX、RPG/400、および COBOL/400 を除くすべてのホスト言語で定義することができます。
  - ILE RPG では、BLOB 可変長 2 進ストリング変数は SQLTYPE キーワードを使用して宣言されます。
  - その他の言語ではすべて、SQL TYPE IS BLOB 文節が使用されます。

ラージ・オブジェクト

ラージ・オブジェクト という語と総称的な頭字語である *LOB* は、CLOB、DBCLOB、BLOB の各データ・タイプを指す総称です。

ロケーターを用いたラージ・オブジェクトの操作

LOB 値は非常に大きい場合があるので、データベース・サーバーからクライアント・アプリケーション・プログラムの変数に LOB 値を転送する処理には、かなりの時間がかかることがあります。また、アプリケーション・プログラムでは、 LOB 値をまとめて処理するのではなく 1 つずつ処理するのが一般的です。そのような場合、アプリケーションでは、ラージ・オブジェクト・ロケーター (LOB ロケーター) によって LOB 値を参照できます。[23]

ラージ・オブジェクト・ロケーター、略して LOB ロケーターは、データベース・サーバーにおける単一の LOB 値を表す値を持った変数です。 LOB ロケーターが開発されたことによって、アプリケーション・プログラムを実行できるクライアント・マシンに LOB 値全体を格納しなくても、非常に大きなオブジェクトをアプリケーション・プログラムで簡単に取り扱うことができるようなメカニズムが可能になります。

例えば、LOB 値を選択する場合、アプリケーション・プログラムは、可能であれば LOB 値全体を選択し、それを同じ大きさの変数に入れる (アプリケーション・プログラムが LOB 値全体を一度で処理するのであれば受け入れ可能) か、または、そ

23. Java アプリケーションには、LOB ロケーターによって記述した CLOB または BLOB と、そうでない CLOB または BLOB とを区別する機能がありません。

の代わりに LOB 値を選択して LOB ロケーターに入れます。その後、LOB ロケーターを使用すれば、アプリケーション・プログラムはロケーター値を入力として与えることにより、LOB 値上での後続のデータベース操作を出すことができます。したがって、例えば、クライアント変数に割り当てられたデータ量などのロケーター演算の結果の出力は、一般的には、入力 LOB 値の小さなサブセットになります。

LOB ロケーターでは基本値を表せるだけでなく、LOB 式に関連付けられた値も表せます。例えば、LOB ロケーターでは以下の式に関連付けられた値を表すこともできます。

```
SUBSTR(lob_value_1 CONCAT lob_value_2 CONCAT lob_value_3, 42, 6000000)
```

アプリケーション・プログラムにおける非ロケーター・ベースのホスト変数の場合、NULL 値がホスト変数に選択されると、標識変数は -1 に設定され、値が NULL であることを示します。しかしながら、LOB ロケーターの場合は、標識変数の意味は若干異なっています。 LOB ロケーター・ホスト変数自体は決して NULL になることはないため、負の標識変数値は LOB ロケーターにより表される LOB 値が NULL であることを示しています。標識変数値によって、クライアントに対して NULL 情報がローカルに保持されます。サーバーは、有効な LOB ロケーターによって NULL 値を追跡しません。

LOB ロケーターは値を表しているのであり、行またはデータベースの位置を表しているのではないということを理解することが重要です。いったん LOB ロケーターに値が選択されてしまうと、 LOB ロケーターが参照している値に影響を及ぼすことになるオリジナルの行や表で実行できる演算はありません。 LOB ロケーターに関連した値は、トランザクションが終了するか、あるいは LOB ロケーターが明示的に解放されるか、そのいずれかが先に起こるまで有効です。

LOB ロケーターは、トランザクション中に LOB 値を参照するためのメカニズムに過ぎません。したがって、ロケーターが作成されたときのトランザクションを超えてまでも存続することはありません。また、LOB ロケーターはデータベース・タイプでもありません。したがって、データベースに保管されることはなく、その結果、ビュー制約や検査制約に関与することも不可能です。しかしながら、ロケーターは LOB タイプを表しているため、FETCH、OPEN、CALL、および EXECUTE ステートメントで使用される SQLDA 構造内で記述できるような LOB タイプ用の SQLTYPE があります。

LOB ストリングを使用する際の制約事項については、 『ストリングの使用に関する制限』を参照してください。

ストリングの使用に関する制限

一部の可変長ストリング・データ・タイプは、特定のコンテキストでは参照できません。

以下の可変長ストリング・データ・タイプは、特定のコンテキストでは参照できません。

- 文字ストリングでは CLOB ストリング
- グラフィック・ストリングでは DBCLOB ストリング
- 2 進ストリングでは BLOB ストリング

表 7. 可変長ストリングの使用が制限されるコンテキスト

使用のコンテキスト	**LOB (CLOB、DBCLOB、または BLOB)**
CREATE INDEX ステートメント	使用できません
主キー、固有キー、および外部キーの定義	使用できません
組み込み関数のパラメーター	可変長文字ストリングと可変長グラフィック・ストリングのいずれかまたは両方を入力引数として使用できる一部の関数では、CLOB ストリングと DBCLOB ストリングのいずれかまたは両方を入力としてサポートしていません。各関数の入力として使用できるデータ・タイプについては、 291 ページの『第 4 章 組み込み関数』の個々の関数の説明を参照してください。

日付/時刻の値

日付/時刻の値は、特定の算術演算やストリング演算で使用することができ、特定のストリングと互換性がありますが、ストリングでも数値でもありません。

ただし、ストリングで日時値を記述することもできます。 95 ページの『日付/時刻の値のストリング表記』を参照してください。

日付

日付 は、グレゴリオ暦を使用して 1 つの時点を示す 3 部構成の値 (年、月、日) です。グレゴリオ暦は西暦 1 年から有効であったものとします。

年の部分の範囲は 0001 から 9999 までです。[24] 日付の形式の *JUL、*MDY、*DMY、および *YMD は、年が 1940 から 2039 までの範囲の日付を表すことができます。月の部分の範囲は、1 から 12 までです。日の部分の範囲は、1 から *x* までです。ここで *x* は、年および月に応じて 28、29、30、または 31 になります。

日付の内部表現は、1 つの整数を含む 4 バイトのストリングです。この整数 (スカリジェル数と呼ばれます) によって、日付を表します。

日付 (DATE) の列の長さは、使用される形式によって、6、8、または 10 バイトのいずれかになります (SQLDA に記述されています)。これらの長さは、値をストリングで表現するのに適した長さです。

時刻

時刻 は、3 つの部分 (時、分、秒) からなる値で、24 時間制を使用して時刻を表します。

時の部分の範囲は 0 から 24 まで、分および秒の部分の範囲は 0 から 59 までです。時の値が 24 の場合、分および秒の値は両方ともゼロになります。

24. 歴史上の日付は、必ずしもグレゴリオ暦に従うとは限らないことに注意してください。1582-10-04 から 1582-10-15 までの日付は、グレゴリオ暦には存在していませんが、有効な日付として受け入れられます。

時刻の内部表現は、3 バイトのストリングです。それぞれのバイトが、2 つのパック 10 進桁から構成されます。最初のバイトが時、2 番目のバイトが分、3 番目のバイトが秒をそれぞれ表します。

時刻 (TIME) の列の長さは 8 バイトで (SQLDA に記述されています)、この長さは、時刻のストリング表現に適した長さです。

タイム・スタンプ

タイム・スタンプ は、日付と時刻を表す、6 または 7 つの部分 (年、月、日、時、分、秒、およびオプションの小数秒) からなる値です。

タイム・スタンプ値の時刻部分には、端数秒の指定を含めることができます。小数秒の桁数は、0 から 12 までの範囲の属性 (デフォルトは 6) を使用して指定します。

タイム・スタンプの内部表示は、7 バイトから 13 バイトのストリングです。最初の 4 バイトは日付、次の 3 バイトは時刻、最後の 0 から 6 バイトは小数秒です。

日時変数

日付、時刻、タイム・スタンプの値を格納するために、通常は文字ストリング変数を使用します。

ILE RPG プリコンパイラーと ILE COBOL プリコンパイラーは、日時変数に対応しています。

Java では、日付、時刻、タイム・スタンプの変数を、java.sql.Date、java.sql.Time、java.sql.Timestamp としてそれぞれ指定することもできます。

日付/時刻の値のストリング表記

データ・タイプが DATE (日付)、TIME (時刻)、または TIMESTAMP (タイム・スタンプ) である値は内部形式で表されますが、SQL ユーザーは、この内部形式を意識する必要はありません。日付、時刻、およびタイム・スタンプは、文字ストリングや Unicode グラフィック・ストリングで表すこともできます。

検索するには、日付/時刻の値をストリング変数に割り当てることができます。結果のストリングの形式は、ステートメントが準備された時点で有効だったデフォルトの日付形式 とデフォルトの時刻形式 によって異なります。デフォルトの日付および時刻の形式は、日付形式 (DATFMT)、日付区切り文字 (DATSEP)、時刻形式 (TIMFMT)、および時刻区切り文字 (TIMSEP) パラメーターに基づいて設定されます。

日付/時刻の値の有効なストリング表現が内部の日付/時刻の値による演算で使用される場合は、その演算が行われる前に、ストリング表現が日付、時刻、またはタイム・スタンプの内部形式に変換されます。デフォルトの日付形式 とデフォルトの時刻形式 では、ストリングの解釈のために使用する日時形式を指定します。ストリングの CCSID が外部のコード化体系を表している場合 (例えば、ASCII)、そのストリングは、日付/時刻の値の内部形式への変換に先立って、そのデフォルトの CCSID によって示されたコード化文字セットにまず変換されます。

以下の各項では、日付/時刻の値の有効なストリング表現を定義しています。

日付ストリング:

日付のストリング表現は、数字で始まる文字ストリングまたは Unicode グラフィック・ストリングで、最低 6 文字の長さを持ちます。このストリングには、末尾ブランクを付けることができます。IBM SQL の標準形式を使用する場合は、月および日の部分から先行ゼロを省略することができます。IBM SQL 標準形式のそれぞれは、名前によって識別され、関連する省略形 (CHAR 関数で使用される) を含みます。それ以外の形式は、CHAR 関数で使用される省略形を持ちません。年が 2 桁の形式の区切り文字は、日付区切り文字 (DATSEP) パラメーターにより制御されます。

日付の有効なストリング形式は、表 8 に示されています。

データベース・マネージャーは、次のいずれかの形式のストリングを日付として認識します。

- デフォルトの日付形式で指定されている形式
- IBM SQL のいずれかの標準日付形式
- 不定様式の年間通算日形式

表 *8.* 日付のストリング表現で使用する形式

形式の名前	省略形	日付形式	例
国際標準化機構 (*ISO)	ISO	'yyyy-mm-dd'	'1987-10-12'
IBM USA 標準 (*USA)		'mm/dd/yyyy'	'10/12/1987'
IBM 欧州標準 (*EUR)	EUR	'dd.mm.yyyy'	'12.10.1987'
日本工業規格の西暦 (*JIS)	JIS	'yyyy-mm-dd'	'1987-10-12'
不定様式の年間通算日	–	'yyyyddd'	'1987285'
年間通算日形式 (*JUL)	–	'yy/ddd'	'87/285'
月、日、年 (*MDY)	–	'mm/dd/yy'	'10/12/87'
日、月、年 (*DMY)	–	'dd/mm/yy'	'12/10/87'
年、月、日 (*YMD)	–	'yy/mm/dd'	'87/12/10'

デフォルトの日付形式は、以下のインターフェースを使用して指定できます。

表 *9.* デフォルト日付形式インターフェース

SQL インターフェース	指定
組み込み SQL	DATFMT および DATSEP パラメーターは、SQL プログラム作成 (CRTSQLxxx) コマンドに指定することができます。 SQL を組み込むプログラム・ソースに DATFMT および DATSEP パラメーターを指定するためには、SET OPTION ステートメントを使用することもできます。 (CRTSQLxxx コマンドについて詳しくは、「組み込み SQL プログラミング」トピック集を参照してください。)

表 9. デフォルト日付形式インターフェース (続き)

SQL インターフェース	指定
対話式 SQL および SQL 実行ステートメント	SQL 開始 (STRSQL) コマンドで DATFMT および DATSEP パラメーターを指定するか、セッション属性を変更します。あるいは、SQL 実行 (RUNSQLSTM) ステートメントで DATFMT および DATSEP パラメーターを使用します。 (STRSQL コマンドと RUNSQLSTM コマンドについて詳しくは、「SQL プログラミング」を参照。)
サーバー上の呼び出しレベル・インターフェース (CLI)	SQL_ATTR_DATE_FMT および SQL_ATTR_DATE_SEP 環境変数または接続変数。 (CLI について詳しくは、「SQL 呼び出しレベル・インターフェース (ODBC)」を参照してください。)
IBM IBM Developer Kit for Java を使用したサーバーの JDBC または SQLJ	「日付形式 (Date Format)」および「日付区切り記号 (Date Separator)」接続プロパティー。 (JDBC および SQLJ について詳しくは、「IBM Developer Kit for Java」を参照してください。)
IBM i Access Family ODBC ドライバーを使用したクライアントの ODBC	ODBC セットアップでの「アドバンスト・サーバー・オプション (Advanced Server Options)」の中の「日付形式 (Date Format)」および「日付区切り記号 (Date Separator)」。 (ODBC の詳細については、「IBM i Access」を参照。)
IBM Toolbox for Java を使用したクライアントの JDBC	JDBC セットアップの中の「形式 (Format)」。 (JDBC の詳細については、「IBM i Access」を参照。) (IBM Toolbox for Java について詳しくは、「IBM Toolbox for Java」を参照してください。)

時刻ストリング**:**

時刻のストリング表現は、数字で始まる文字ストリングまたは Unicode グラフィック・ストリングで、最低 4 文字の長さです。このストリングには末尾ブランクを付けることができます。時刻の時の部分から先行ゼロを除去することができ、秒の部分全体を除去することができます。ユーザーが秒の除去を選択すると、暗黙にゼロ秒が指定されたことになります。したがって、13.30 は 13.30.00 に等しいことになります。

時刻の有効なストリング形式は、 98 ページの表 10 に示されています。IBM SQL 標準形式のそれぞれは、名前によって識別され、関連する省略形 (CHAR 関数で使用される) を含みます。それ以外の形式 (*HMS) は、CHAR 関数によって使用される省略形を持っていません。*HMS 形式の区切り文字は、時刻区切り文字 (TIMSEP) パラメーターにより制御されます。

データベース・マネージャーは、形式が次のいずれかの場合に、ストリングを時刻として認識します。

- デフォルトの時刻形式に指定されている形式
- IBM SQL 標準時刻形式のいずれか

表 *10.* 時刻のストリング表現で使用する形式

形式の名前	省略形	時刻の形式	例
国際標準化機構 (*ISO)	ISO	'hh.mm.ss' [25]	'13.30.05'
IBM USA 標準 (*USA)		'hh:mm AM' (or PM)	'1:30 PM'
IBM 欧州標準 (*EUR)	EUR	'hh.mm.ss'	'13.30.05'
日本工業規格の西暦 (*JIS)	JIS	'hh:mm:ss'	'13:30:05'
時、分、秒 (*HMS)	–	'hh:mm:ss'	'13:30:05'

以下の追加の規則は USA 時刻形式に適用されます。

- 時は 12 を超えてはならず、00:00 AM という特殊な場合を除いて、0 にすることはできません。
- 時刻の分の部分と AM または PM との間に 1 つのスペース文字があります。
- 分の部分は省略することができます。ユーザーが分の除去を選択すると、暗黙にゼロ分が指定されたことになります。

USA 時刻形式で 24 時間時計の ISO 形式を使用する場合、USA 形式と 24 時間時計の間の対応を示すと、次のようになります。

表 *11. USA* 時刻形式

USA 形式	**24** 時間時計
12:01 AM から 12:59 AM	00.01.00 から 00.59.00
01:00 AM から 11:59 AM	01:00.00 から 11:59.00
12:00 PM (正午) から 11:59 PM	12:00.00 から 23.59.00
12:00 AM (午前零時)	24.00.00
00:00 AM (午前零時)	00.00.00

デフォルトの時刻形式は、以下のインターフェースを使用して指定できます。

表 *12.* デフォルト時刻形式インターフェース

SQL インターフェース	指定
組み込み SQL	SQL プログラム作成 (CRTSQLxxx) コマンドで、TIMFMT および TIMSEP パラメーターを指定します。 SQL を組み込むプログラム・ソースに TIMFMT および TIMSEP パラメーターを指定するには、SET OPTION ステートメントを使用することもできます。(CRTSQLxxx コマンドについて詳しくは、「組み込み SQL プログラミング」トピック集を参照してください。)

25. これは、以前のバージョンの ISO 形式です。JIS を使用すると、現行の ISO 形式になります。

表 12. デフォルト時刻形式インターフェース (続き)

SQL インターフェース	指定
対話式 SQL および SQL 実行ステートメント	SQL 開始 (STRSQL) コマンドで TIMFMT および TIMSEP パラメーターを指定するか、セッション属性を変更します。あるいは、SQL 実行 (RUNSQLSTM) ステートメントで TIMFMT および TIMSEP パラメーターを使用します。 (STRSQL コマンドおよび RUNSQLSTM コマンドの詳細については、「SQL プログラミング」トピック集を参照。)
サーバー上の呼び出しレベル・インターフェース (CLI)	SQL_ATTR_TIME_FMT および SQL_ATTR_TIME_SEP 環境変数または接続変数。 (CLI の詳細については、「SQL 呼び出しレベル・インターフェース (ODBC)」トピック集を参照。)
IBM IBM Developer Kit for Java を使用したサーバーの JDBC または SQLJ	「時刻形式 (Time Format)」および「時刻区切り記号 (Time Separator)」接続プロパティー・オブジェクト。 (JDBC および SQLJ の詳細については、「IBM Developer Kit for Java」トピック集を参照。)
IBM i Access Family ODBC ドライバーを使用したクライアントの ODBC	ODBC セットアップでの「アドバンスト・サーバー・オプション (Advanced Server Options)」の中の「時刻形式 (Time Format)」および「時刻区切り記号 (Time Separator)」。 (ODBC の詳細については、「IBM i Access Family」トピック集を参照。)
IBM Toolbox for Java を使用したクライアントの JDBC	JDBC セットアップの中の「形式 (Format)」。 (IBM Toolbox for Java の詳細については、「IBM Toolbox for Java」トピック集を参照してください。)

タイム・スタンプ・ストリング**:**

タイム・スタンプのストリング表現は、数字で始まる文字ストリングまたは Unicode グラフィック・ストリングで、最低 14 文字の長さです。

タイム・スタンプの完全なストリング表現は、以下のいずれかの形式になります。

表 13. タイム・スタンプのストリング表現で使用する形式

形式の名前	時刻の形式	例
ISO タイム・スタンプ	'yyyy-mm-dd hh:mm:ss.nnnnnnnnnnnn'	'1990-03-02 08:30:00.010000000000'
IBM SQL	'yyyy-mm-dd-hh.mm.ss.nnnnnnnnnnnn'	'1990-03-02-08.30.00.010000000000'
14-26 文字形式	'yyyymmddhhmmssnnnnnnnnnnnn'	'19900302083000'

以下の規則が適用されます。

- 先行ブランクは使用できません。
- このストリングには、末尾ブランクを付けることができます。
- 区切り記号付きのタイム・スタンプ形式を使用しているときは、タイム・スタンプの月、日、時、分、および秒の部分の先行ゼロを省略することができます。省略されている桁に対しては、暗黙的な 0 の指定が想定されます。

- 時を 24 にすることができるのは、分、秒、および端数秒がすべてゼロである場合です。
- 端数秒の部分の後続ゼロは、切り捨てたり、全部を除去したりすることができます。
- 端数秒の桁数は、0 から 12 までの範囲で変わります。端数秒が省略されている場合、暗黙的な 0 の指定が想定されます。
- 秒エレメントの後の区切り文字は、端数秒が含まれない場合は省略できます。

タイム・スタンプのストリング表現が暗黙的に TIMESTAMP データ・タイプの値にキャストされる場合、キャストの結果のタイム・スタンプ精度はそのキャスト操作によって決まります。キャストのタイム・スタンプ精度を超えるストリング中の桁が切り捨てられるか、または、キャストのタイム・スタンプ精度に合わせるために必要な不足桁がゼロと想定されます。例えば、1990-3-2-8.30.00 は 1990-03-02-8.30.00.000000000000 と等価です。タイム・スタンプのストリング表記は、この値を指定した精度でタイム・スタンプに明示的にキャストすることによって、異なるタイム・スタンプの精度を指定することができます。定数のキャストでは、ストリングの前に TIMESTAMP キーワードを置くことによって精度を維持できます。例えば、TIMESTAMP '2007-03-28-14.50.35.123'は TIMESTAMP(3) のデータ・タイプになります。

XML 値

XML 値は、XML 文書、XML コンテンツ、または XML シーケンスの形式の適格な XML を表します。

XML データ・タイプで定義された列の値として表に格納される XML 値は、整形式 XML 文書でなければなりません。XML 値は、別の XML 値を含む、どのストリング値にも比較できない内部表記で処理されます。XML データ・タイプに適用できる唯一の述部は、IS NULL 述部です。

XML 値は、XML 文書を表す直列化されたストリング値に変換することができます。これを行うには、XMLSERIALIZE 関数を使用します。同様に、XML 文書を表すストリング値は、XML 値に変換することができます。これを行うには、XMLPARSE 関数を使用します。XML 値は、アプリケーションでストリングやバイナリーのデータ・タイプを変換する際に暗黙的に構文解析または直列化できます。

XML データ・タイプには、最大長は定義されていません。XML を表す直列化されたストリング値として扱われる場合は、有効な最大長に関する制約があります。その場合の限度は、LOB データ・タイプと同じです。LOB と同様、XML ロケーターと XML ファイルは変数を参照します。

XML 値を使用する場合の制約事項**:** いくつかの例外を除き、他のデータ・タイプを使用可能な同一コンテキストの中で XML 値を使用できます。XML 値は、以下において有効です。

- パラメーター・マーカー、XML、または NULL から XML への CAST
- パラメーター・マーカー、XML、または NULL から XML への XMLCAST
- IS NULL 述部
- COUNT および COUNT_BIG 集約関数
- COALESCE、IFNULL、HEX、LENGTH、CONTAINS、および SCORE スカラー関数
- XML スカラー関数
- DISTINCT が指定されていない SELECT リスト
- INSERT VALUES 文節、UPDATE SET 文節、および MERGE
- SET および VALUES INTO
- プロシージャー・パラメーター
- ユーザー定義関数の引数および結果
- トリガー相関変数
- 動的準備済みステートメントのパラメーター・マーカー値

XML 値は、以下の箇所では直接使用できません。 XMLSERIALIZE の引数など、式が使用できる場所では XML 値も使用できます。

- DISTINCT キーワードが含まれる SELECT リスト
- GROUP BY 文節
- ORDER BY 文節
- UNION ALL ではない全選択の副選択
- 基本述部、多値比較述部、BETWEEN、DISTINCT、IN、または LIKE 述部

- DISTINCT キーワードが指定された集約関数
- 主キー、ユニーク・キー、または外部キー
- チェック制約
- 索引列

XML データ・タイプの組み込みデータ・タイプを持つホスト言語はありません。

XML データ・モデルおよび XML 値については、SQL XML プログラミングを参照してください。

XML の CCSID の決定

XML データは、任意の 1 バイトの EBCDIC、混合 CCSID、または Unicode CCSID 1208、1200、13488 を使用して定義できます。XML データの CCSID として、65535 は使用できません。CCSID は、XML データ・タイプを定義する際に明示的に指定できます。明示的に指定しない場合、SQL_XML_DATA_CCSID QAQQINI ファイル・オプションの値を使用して、CCSID が割り当てられます。この値が設定されないと、デフォルトは 1208 (UTF-8) です。

ステートメントが実行されると、SQL スキーマ・ステートメントで使用される XML データ・タイプの CCSID が設定されます。

CCSID を割り当てる DECLARE VARIABLE がない XML ホスト変数の場合、以下のようにして CCSID が割り当てられます。

- XML AS DBCLOB の場合、CCSID は 1200 です。
- XML AS CLOB および SQL_XML_DATA_CCSID QAQQINI 値が 1200 または 13488 の場合、CCSID は 1208 になります。
- その他の場合、SQL_XML_DATA_CCSID QAQQINI 値が CCSID として使用されます。

暗黙的であれ明示的であれすべての XMLPARSE 関数は UTF-8 (1208) を使用して実行されるので、この CCSID でデータを定義するとデータを UTF-8 に変換する必要がなくなります。

データ・リンク値

データ・リンク値とは、データベースからデータベースの外部に保管されたファイルへの論理的な参照を含むカプセル化された値です。

このカプセル化された値の属性は、以下のとおりです。

リンク・タイプ
: 現在サポートされているリンクのタイプは、URL (Uniform Resource Locator) です。

スキーム
: URL の場合、これは HTTP または FILE などの値です。この値は、何が入力されている場合でも、大文字でデータベースに保管されます。

ファイル・サーバー名
: ファイル・サーバーの完全なアドレス。この値は、何が入力されている場合でも、大文字でデータベースに保管されます。

ファイル・パス
: サーバー内のファイルの識別。この値は、大文字小文字の区別があります。したがって、データベースに保管する場合にも大文字に変換されることはありません。

アクセス制御トークン
: 必要に応じて、アクセス・トークンがファイル・パスに組み込まれます。これは、動的に生成されるため、データベースに保管されるデータ・リンク値の永続部分ではありません。

コメント
: 254 バイトまでの記述情報。これは、データのある場所についての詳細や代替の情報など、アプリケーション固有の使用を意図しています。

データ・リンク値で使用する文字は、URL 用に定義されたセットに限定されます。これらの文字には、英大文字 (A から Z) と英小文字 (a から z)、数字 (0 から 9) と特殊文字のサブセット ($、-、_、@、.、 &、+、!、*、"、'、(、)、=、 ;、 /、 #、 ?、 :、スペース、およびコンマ) が含まれます。

最初の 4 つの属性はまとめて、リンケージ属性とも言われます。データ・リンク値が、コメントの属性だけを持ち、リンケージ属性をまったく持たないということもあり得ます。そのような値でも列に保管されますが、当然のことながら、そのような列にリンクされるようなファイルはありません。

このようなファイルに対するデータ・リンク参照と、 178 ページの『LOB または XML ファイル参照変数の参照』で説明されているような LOB ファイル参照変数とを識別することが重要です。両方とも、ファイルを表している点は似ています。しかしながら、

- データ・リンクはデータベースに保存されており、リンクされたファイルのリンクとデータの両方とも、データベースにおけるデータの自然な拡張と考えられます。
- ファイル参照変数は一時的に存在しており、ホスト・プログラム・バッファーの代替と考えられます。

データ・リンク値を構築 (DLVALUE) し、データ・リンク値からカプセル化された値を抽出 (DLCOMMENT、DLLINKTYPE、DLURLCOMPLETE、DLURLPATH、DLURLPATHONLY、 DLURLSCHEME、DLURLSERVER) するために、組み込みスカラー関数が用意されています。

行 ID 値

行 *ID* は、表の中の各行を一意的に識別する値です。特定の列または変数に、行 ID データ・タイプを与えることができます。 ROWID 列を使用することにより、表の中の特定の行まで直接ナビゲートする照会を書くことができます。

ROWID 列の中の値はそれぞれ固有のものでなければなりません。表を再編成しても、データベース・マネージャーは永続的にこれらの値を保持します。表に行を挿入するときに、ROWID 列の値を指定しなければ、データベース・マネージャーがその値を生成します。値を指定する場合は、Db2 for z/OS または Db2 for i により既に生成されている有効な行 ID 値でなければなりません。

ユーザーは、行 ID 値の内部表現を意識する必要はありません。この値は BIT データを含むものと見なされるため、CCSID 変換を受けることはありません。 ROWID 列の長さ属性は 40 です。

ユーザー定義タイプ

ユーザー定義タイプ は、CREATE TYPE ステートメントを使用してデータベースに定義されるデータ・タイプです。ユーザー定義タイプには、特殊タイプと配列タイプという 2 つのタイプがあります。

特殊タイプ

特殊タイプ は、その内部表現を組み込みのデータ・タイプ (その「ソース・タイプ」) と共用するユーザー定義のデータ・タイプですが、大部分の演算では別のタイプ、および、互換性のないタイプと考えられます。例えば、ピクチャー・タイプ、テキスト・タイプ、およびオーディオ・タイプはすべて、その内部表現に組み込みのデータ・タイプ BLOB を使用していますが、意味体系はまったく異なります。特殊タイプは、 1328 ページの『CREATE TYPE (特殊)』を使用して作成します。

例えば、次のステートメントによって、AUDIO という名前の特殊タイプが作成されます。

```
CREATE TYPE AUDIO AS BLOB (1M)
```

AUDIO は、組み込みデータ・タイプ BLOB と同じ内部表現を持っていますが、BLOB や他のいずれのタイプとも比較できない、別のタイプと考えられます。このように、AUDIO を他のデータ・タイプと比較することはできないために、AUDIO 専用の関数を作成することが可能になり、そのような関数は他のデータ・タイプ (ピクチャーやテキストなど) には適用不能ということが保証されることになります。

特殊タイプの名前は、スキーマ名で修飾されます。非修飾名の暗黙的なスキーマ名は、特殊タイプが現れる文脈によって異なります。非修飾の特殊タイプ名を使用する場合、次のとおりです。

- CREATE TYPE ステートメント、あるいは
  DROP、COMMENT、LABEL、GRANT、または REVOKE ステートメントのオブジェクトでは、データベース・マネージャーは権限 ID による修飾上の通常のプロセスを使用して、スキーマ名を決めます。修飾の規則についての詳細は、74 ページの『非修飾の関数、プロシージャー、特定名、タイプ、および変数』を参照してください。
- その他の文脈では、データベース・マネージャーは SQL パスを使用して、スキーマ名を決めます。データベース・マネージャーはパス上を順番に検索し、一致した特殊タイプを持つ最初のスキーマを選択します。 SQL パスの説明については、 72 ページの『SQL パス』を参照してください。

特殊タイプは、そのソース・タイプの関数と演算子が意味のあるものとは限らないため、それらを自動的に獲得することはありません。(例えば、AUDIO タイプの LENGTH 関数は、オブジェクトの長さをバイトではなく、秒で戻すかもしれません。) その代わりに、特殊タイプは強力タイプ をサポートしています。強力タイプでは、特殊タイプ用に明示的に定義された関数と演算子だけを、その特殊タイプに

適用することができるようにしています。ただし、ソース・タイプの関数や演算子は、適切なユーザー定義の関数を作成することによって適用することができます。ユーザー定義の関数は、ソース・タイプをパラメーターとして持つ既存の関数に基づいている必要があります。例えば、以下の一連の SQL ステートメントでは、データ・タイプ DECIMAL(9,2) に基づいて特殊タイプ MONEY を作成する方法、その特殊タイプの + 演算子を定義する方法、その演算子を特殊タイプに提供する方法を示します。

```
CREATE TYPE MONEY AS DECIMAL(9,2) WITH COMPARISONS
CREATE FUNCTION "+"(MONEY,MONEY)
   RETURNS MONEY
   SOURCE "+"(DECIMAL(9,2),DECIMAL(9,2))
CREATE TABLE SALARY_TABLE
   (SALARY MONEY,
    COMMISSION MONEY)
SELECT "+"(SALARY, COMMISSION) FROM SALARY_TABLE
```

特殊タイプは、そのソース・タイプと同じ制約事項に従う必要があります。例えば、1 つの表が持つことができる ROWID 列は 1 つだけです。したがって、ROWID 列を持つ表が、行 ID をソースとする特殊タイプの列を同時に持つことはできません。

データ・リンクに基づいた特殊タイプを除いて、比較演算子が特殊タイプ用に自動的に生成されます。さらに、データベース・マネージャーは、ソース・タイプから特殊タイプへ、および特殊タイプからソース・タイプへのキャストをサポートする特殊タイプ用の関数を自動的に生成します。例えば、前に作成された AUDIO タイプの場合、次のようなキャスト関数が作成されます。

生成されるキャスト関数の名前	パラメーター・リスト	戻されるデータ・タイプ
schema-name.BLOB	schema-name.AUDIO	BLOB
schema-name.AUDIO	BLOB	schema-name.AUDIO

配列タイプ

配列 は、順序付けられたデータ・エレメントの集合が含まれる構造体です。配列内のすべてのエレメントのデータ・タイプは同じです。配列のカーディナリティーは、配列のエレメントの数と同じです。

配列全体を参照するか、または、配列の個々のエレメントを集合内の順序位置によって参照することができます。配列のカーディナリティーを *N* とすれば、各エレメントに関連した順序位置は、1 以上 *N* 以下の整数値になります。

配列タイプ は、配列として定義されたユーザー定義のデータ・タイプです。 SQL 変数または SQL パラメーターは、ユーザー定義配列データ・タイプとして定義できます。さらに、TRIM_ARRAY 関数の呼び出しの結果または CAST 指定の結果も、ユーザー定義配列データ・タイプにすることができます。ユーザー定義配列タイプのエレメントは、その配列のエレメントと同じデータ・タイプを戻す式を使用できる場所であれば、どこからでも参照できます。

無名配列タイプは、ユーザー定義データ・タイプが関連付けられていない配列です。 ARRAY_AGG 集約関数または ARRAY コンストラクターの呼び出しの結果

は、ユーザー定義のデータ・タイプが関連付けられていない配列です。ユーザー定義の配列タイプが関連付けられていない配列のエレメントは、直接参照できません。

配列値は、空 (カーディナリティーがゼロ)、ヌルにすることができ、配列の個々のエレメントは、ヌルまたはヌル以外にすることができます。空の配列は、ヌルの配列値や、すべてのエレメントがヌル値の配列とは異なります。

配列値をデータベースに保管することはできず、Java 以外の外部アプリケーションに戻すこともできません。

データ・タイプのプロモーション

データ・タイプは、関連したデータ・タイプに分類することができます。そのようなグループ内では、あるデータ・タイプが別のデータ・タイプに優先すると考えられるような優先順位が存在します。この優先順位によって、データベース・マネージャーは、1 つのデータ・タイプの、優先順位の低い別のデータ・タイプへのプロモーション をサポートできます。例えば、データ・タイプ CHAR は VARCHAR へのプロモーションが可能であり、 INTEGER は DOUBLE PRECISION へのプロモーションが可能ですが、 CLOB は VARCHAR へのプロモーションができません。

データベース・マネージャーは、以下のようなときにプロモーションを検討します。

- 関数解決を実行するとき (185 ページの『関数解決』を参照)
- 特殊タイプをキャストするとき (109 ページの『データ・タイプ間のキャスト』を参照)
- 特殊タイプを組み込みデータ・タイプに割り当てるとき (131 ページの『特殊タイプの比較』を参照)

表 14 は、データベース・マネージャーが各データ・タイプをプロモートできるデータ・タイプを判定するために使用するデータ・タイプごとの優先順位リストを示しています。この表は、最良の選択は同一データ・タイプであり、別のデータ・タイプにプロモーションしないことであることを示しています。また、この表は、プロモーション・プロセスでは同等であると考えられるデータ・タイプも示していることに注意してください。例えば、CHARACTER と GRAPHIC は同等のデータ・タイプであると考えられます。

表 *14.* データ・タイプの優先順位表

データ・タイプ	データ・タイプ優先順位リスト **(**高いものから低いものへの順**)**
SMALLINT	SMALLINT、INTEGER、BIGINT、decimal、real、double、DECFLOAT
INTEGER	INTEGER、BIGINT、decimal、real、double、DECFLOAT
BIGINT	BIGINT、decimal、real、double、DECFLOAT
10 進数	decimal、real、double、DECFLOAT
real	real、double、DECFLOAT
double	double、DECFLOAT
DECFLOAT	DECFLOAT
CHAR または GRAPHIC	CHAR または GRAPHIC、VARCHAR または VARGRAPHIC、CLOB または DBCLOB
VARCHAR または VARGRAPHIC	VARCHAR または VARGRAPHIC、CLOB または DBCLOB
CLOB または DBCLOB	CLOB または DBCLOB
CHAR FOR BIT DATA	CHAR、VARCHAR、CLOB、BINARY、VARBINARY、BLOB
VARCHAR FOR BIT DATA	VARCHAR、CLOB、VARBINARY、BLOB

表 *14.* データ・タイプの優先順位表 *(*続き*)*

データ・タイプ	データ・タイプ優先順位リスト **(**高いものから低いものへの順**)**
BINARY	BINARY、VARBINARY、BLOB、CHAR FOR BIT DATA、VARCHAR FOR BIT DATA
VARBINARY	VARBINARY、BLOB、VARCHAR FOR BIT DATA
BLOB	BLOB
DATE	DATE、TIMESTAMP
TIME	TIME
TIMESTAMP	TIMESTAMP
DATALINK	DATALINK
ROWID	ROWID
XML	XML
ARRAY	ARRAY
udt	同じ udt

注:

小文字で示したタイプの定義は、以下のとおりです。

10 進数
= DECIMAL(p,s) または NUMERIC(p,s)

real = REAL または FLOAT(*n*)。この場合の *n* は短精度浮動小数点数の指定

double
= DOUBLE、DOUBLE PRECISION、FLOAT または FLOAT(*n*)。この場合の *n* は倍精度浮動小数点数の指定

udt = ユーザー定義タイプ

リストされているデータ・タイプの短形式と長形式の同義語は、リストされている同義語と同じと見なされます。

文字ストリングとグラフィック・ストリングは、Unicode データの場合にのみ互換性があります。

データ・タイプ間のキャスト

所定のデータ・タイプを持つ値を、別のデータ・タイプに、あるいは異なる長さ、精度、位取りを持つ同じデータ・タイプにキャストする (変更する) 必要が生じる場合がしばしばあります。

107 ページの『データ・タイプのプロモーション』で説明したデータ・タイプ・プロモーションは、あるデータ・タイプの値を新しいデータ・タイプにキャストする必要がある場合の例です。別のデータ・タイプに変更できるデータ・タイプは、ソース・データ・タイプからターゲット・データ・タイプへとキャスト可能 です。

あるデータ・タイプから別のデータ・タイプへのキャストは、明示的にも、暗黙的にも起こり得ます。キャスト関数または CAST 指定 (218 ページの『CAST の指定』を参照) は、データ・タイプを明示的に変更するために使用できます。データベース・マネージャーは、特殊タイプを含む割り当ての際に、暗黙的にデータ・タイプをキャストします (124 ページの『特殊タイプの割り当て』を参照してください)。さらに、ソースに基づくユーザー定義関数を作成する場合、ソース関数のパラメーターのデータ・タイプは、作成する関数のデータ・タイプに対してキャスト可能である必要があります (1122 ページの『CREATE FUNCTION (ソース派生)』を参照してください)。

文字ストリングまたはグラフィック・ストリングを別のデータ・タイプにキャストする際に切り捨てが生じた場合は、非ブランクの文字が切り捨てられた場合には、警告が出されます。この切り捨ての動作は、文字ストリングまたはグラフィック・ストリングの検索割り当ての場合と同様です (119 ページの『検索割り当て:』を参照)。

バイナリー・ストリングを別のデータ・タイプにキャストするときに切り捨てが発生すると、警告が生成されます。この切り捨ての動作は、バイナリー・ストリングの取得割り当ての場合とよく似ています (118 ページの『検索割り当て』を参照してください)。

配列タイプが関係するキャストでは、ソースのデータ・タイプとターゲットのデータ・タイプの両方が同じ配列タイプでなければなりません。

キャストする先、あるいはキャストする元のいずれかデータ・タイプとして特殊タイプを含むキャストについて、表 15 はサポートされるキャストを示しています。110 ページの表 16 は、組み込みデータ・タイプ間のキャストの場合にサポートされるキャストを示しています。

表 *15.* 特殊タイプが含まれる場合のサポートされるキャスト

データ・タイプ ...	キャスト可能な相手先のデータ・タイプ ...
特殊タイプ *DT*	特殊タイプ *DT* のソース・データ・タイプ
特殊タイプ *DT* のソース・データ・タイプ	特殊タイプ *DT*
特殊タイプ *DT*	特殊タイプ *DT*
データ・タイプ *A*	特殊タイプ *DT* (*A* が特殊タイプ *DT* のソース・データ・タイプにプロモーション可能な場合) (107 ページの『データ・タイプのプロモーション』を参照)

表 15. 特殊タイプが含まれる場合のサポートされるキャスト (続き)

データ・タイプ ...	キャスト可能な相手先のデータ・タイプ ...
INTEGER	特殊タイプ *DT* (*DT* のソース・タイプが SMALLINT の場合)
DOUBLE	特殊タイプ *DT* (*DT* のソース・データ・タイプが REAL の場合)
VARCHAR	特殊タイプ *DT* (*DT* のソース・データ・タイプが CHAR または GRAPHIC の場合)
VARGRAPHIC	特殊タイプ *DT* (*DT* のソース・データ・タイプが GRAPHIC または CHAR の場合)
VARBINARY	特殊タイプ *DT* (*DT* のソース・データ・タイプが BINARY の場合)

文字ストリングとグラフィック・ストリングは、Unicode データの場合にのみ互換性があります。文字ビット・データとグラフィック・ストリングには、互換性がありません。

特殊タイプがキャストに関係している場合は、特殊タイプの作成時に生成されたキャスト関数が使用されます。データベース・マネージャーがどの関数を選択するかは、関数表記を使用するか、CAST 指定構文を使用するかによって異なります。詳細については、 185 ページの『関数解決』および 218 ページの『CAST の指定』を参照してください。関数解決は、両方の場合に使用されます。ただし、CAST 指定の場合、ターゲット・データ・タイプとして非修飾の特殊タイプを指定すると、データベース・マネージャーは、特殊タイプのスキーマ名を解決してから、そのスキーマ名を使用してキャスト関数の位置を判別します。

次の表は、組み込みデータ・タイプ間でサポートされるキャストを示しています。

表 16. 組み込みデータ・タイプ間でサポートされるキャスト

ソース・データ・タイプ ↓ / ターゲット・データ・タイプ →	SMALLINT	INTEGER	BIGINT	DECIMAL NUMERIC	REAL	DOUBLE	DECFLOAT	CHAR	VARCHAR	CLOB	GRAPHIC	VARGRAPHIC	DBCLOB	BINARY	VARBINARY	BLOB	DATE	TIME	TIMESTAMP	ROWID	DATALINK	XML
SMALLINT	Y	Y	Y	Y	Y	Y	Y	Y[1]						—			—	—	—	—	—	—
INTEGER	Y	Y	Y	Y	Y	Y	Y	Y[1]						—			—	—	—	—	—	—
BIGINT	Y	Y	Y	Y	Y	Y	Y	Y[1]						—			—	—	—	—	—	—
DECIMAL	Y	Y	Y	Y	Y	Y	Y	Y[1]						—			—	—	—	—	—	—
NUMERIC	Y	Y	Y	Y	Y	Y	Y	Y[1]						—			—	—	—	—	—	—
REAL	Y	Y	Y	Y	Y	Y	Y	Y[1]						—			—	—	—	—	—	—
DOUBLE	Y	Y	Y	Y	Y	Y	Y	Y[1]						—			—	—	—	—	—	—
DECFLOAT	Y	Y	Y	Y	Y	Y	Y	Y[1]						—			—	—	—	—	—	—
CHAR	Y	Y	Y	Y	Y	Y	Y	Y[1]						Y			Y	Y	Y	Y	—	—
VARCHAR	Y	Y	Y	Y	Y	Y	Y	Y[1]						Y			Y	Y	Y	Y	—	—
CLOB	Y	Y	Y	Y	Y	Y	Y	Y[1]						Y			Y	Y	Y	Y	—	—
GRAPHIC	Y[1]	Y[1]	Y[1]	Y[1]	Y[1]	Y[1]	Y[1]	Y						Y			Y[1]	Y[1]	Y[1]	—	—	—
VARGRAPHIC	Y[1]	Y[1]	Y[1]	Y[1]	Y[1]	Y[1]	Y[1]	Y						Y			Y[1]	Y[1]	Y[1]	—	—	—

表 16. 組み込みデータ・タイプ間でサポートされるキャスト (続き)

ターゲット・データ・タイプ → / ソース・データ・タイプ ↓	SMALLINT	INTEGER	BIGINT	DECIMAL NUMERIC	REAL DOUBLE DECFLOAT	CHAR VARCHAR CLOB	GRAPHIC VARGRAPHIC DBCLOB	BINARY VARBINARY BLOB	DATE	TIME	TIMESTAMP	ROWID	DATALINK	XML
DBCLOB	Y[1]	Y[1]	Y[1]	Y[1]	Y[1]	Y[1]	Y	Y	Y[1]	Y[1]	Y[1]	—	—	—
BINARY	—	—	—	—	—	Y	—	Y	—	—	—	—	—	—
VARBINARY	—	—	—	—	—	Y	—	Y	—	—	—	—	—	—
BLOB	—	—	—	—	—	Y	—	Y	—	—	—	—	—	—
DATE	—	Y	Y	Y	—	Y	Y[1]	—	Y	—	Y	—	—	—
TIME	—	Y	Y	Y	—	Y	Y[1]	—	—	Y	Y	—	—	—
TIMESTAMP	—	—	Y	Y	—	Y	Y[1]	—	Y	Y	Y	—	—	—
ROWID	—	—	—	—	—	Y	—	Y	—	—	—	Y	—	—
DATALINK	—	—	—	—	—	—	—	—	—	—	—	—	Y	—
XML	—	—	—	—	—	—	—	—	—	—	—	—	—	Y

注:

[1] 変換は、ユニコード・グラフィックの場合にのみサポートされます。他方のデータ・タイプが FOR BIT DATA の場合は、変換できません。

次の表は、データ・タイプへのキャストの規則を示しています。

表 17. データ・タイプへのキャストの規則

ターゲット・データ・タイプ	規則
SMALLINT	628 ページの『SMALLINT』を参照してください。
INTEGER	496 ページの『INTEGER または INT』を参照してください。
BIGINT	363 ページの『BIGINT』を参照してください。
DECIMAL	429 ページの『DECIMAL または DEC』を参照してください。
NUMERIC	738 ページの『ZONED』を参照してください。
REAL	587 ページの『REAL』を参照してください。
DOUBLE	448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。
DECFLOAT	423 ページの『DECFLOAT』を参照してください。
CHAR	375 ページの『CHAR』を参照してください。
VARCHAR	672 ページの『VARCHAR』を参照してください。
CLOB	384 ページの『CLOB』を参照してください。
GRAPHIC	ソース・データ・タイプが文字ストリングの場合は、 113 ページの『割り当ておよび比較』にある変数に対するストリング割り当ての規則を参照してください。 その他の場合は、 474 ページの『GRAPHIC』を参照してください。
VARGRAPHIC	ソース・データ・タイプが文字ストリングの場合は、 113 ページの『割り当ておよび比較』にある変数に対するストリング割り当ての規則を参照してください。 その他の場合は、 689 ページの『VARGRAPHIC』を参照してください。

表 17. データ・タイプへのキャストの規則 (続き)

ターゲット・データ・タイプ	規則
DBCLOB	414 ページの『DBCLOB』を参照してください。
BINARY	365 ページの『BINARY』を参照してください。
VARBINARY	669 ページの『VARBINARY』を参照してください。
BLOB	369 ページの『BLOB』を参照してください。
DATE	405 ページの『DATE』を参照してください。
TIME	643 ページの『TIME』を参照してください。
TIMESTAMP	644 ページの『TIMESTAMP』 で、2 番目の引数としてターゲット・データ・タイプの精度を使用する場合を参照してください。 ソース・データ・タイプが DATE の場合は、指定の日付と 00:00:00 という時刻でタイム・スタンプが構成されます。 ソース・データ・タイプが TIME の場合は、 CURRENT_DATE と指定の時刻でタイム・スタンプが構成されます。
DATALINK	113 ページの『割り当ておよび比較』にあるデータ・リンク割り当ての規則を参照してください。
ROWID	612 ページの『ROWID』を参照してください。
XML	113 ページの『割り当ておよび比較』にある XML 割り当ての規則を参照してください。

割り当ておよび比較

SQL の基本演算は、割り当てと比較です。割り当て演算は、CALL、INSERT、UPDATE、FETCH、SELECT、SET 変数、および VALUE INTO ステートメントの実行時に行われます。比較演算は、述部や他の言語エレメント (MAX、MIN、DISTINCT、GROUP BY、ORDER BY など) が入っているステートメントを実行する過程で行われます。

これらの演算はいずれも、演算に使用するオペランド間でデータ・タイプに互換性がなければならないという基本的な規則があります。この互換性の規則は、UNION、EXCEPT、INTERSECT、連結、CASE 式、および CONCAT、VALUE、COALESCE、IFNULL、MIN、MAX スカラー関数にも適用されます。互換性マトリックスは、次のとおりです。

表 *18.* データ・タイプの互換性

オペランド	**2** 進整数	**10** 進数	浮動小数点数	**10** 進 浮動小数点数	文字ストリング	グラフィック・ストリング	**2** 進ストリング	日付	時刻	タイム・スタンプ	データ・リンク	行 **ID**	**XML** [7]	ユーザー定義タイプ
2 進整数	Y	Y	Y	Y	Y	1	—	—	—	—	—	—	—	3
10 進数[4]	Y	Y	Y	Y	Y	1	—	—	—	—	—	—	—	3
浮動小数点数	Y	Y	Y	Y	Y	1	—	—	—	—	—	—	—	3
10 進浮動小数点数	Y	Y	Y	Y	Y	1	—	—	—	—	—	—	—	3
文字ストリング	Y	Y	Y	Y	Y	1	2	—	—	—	—	—	—	3
グラフィック・ストリング	1	1	1	1	1	Y	—	1	1	1	—	—	—	3
2 進ストリング	—	—	—	—	2	—	Y	—	—	—	—	—	—	3
日付	—	—	—	—	—	1	—	Y	—	Y	—	—	—	3
時刻	—	—	—	—	—	1	—	—	Y	—	—	—	—	3
タイム・スタンプ	—	—	—	—	—	1	—	Y	—	Y	—	—	—	3
データ・リンク	—	—	—	—	—	—	—	—	—	—	5	—	—	3
行 ID	—	—	—	—	—	—	—	—	—	—	—	6	—	3
XML [7]	—	—	—	—	—	—	—	—	—	—	—	—	Y	3
ユーザー定義タイプ	3	3	3	3	3	3	3	3	3	3	3	3	3	3

表 18. データ・タイプの互換性 (続き)

オペランド	**2** 進整数	**10** 進数	浮動小数点数	**10** 進 浮動小数点数	文字ストリング	グラフィック・ストリング	**2** 進ストリング	日付	時刻	タイム・スタンプ	データ・リンク	行 **ID**	**XML** [7]	ユーザー定義タイプ

注:

1. 互換性があるのは Unicode グラフィック・ストリングのみです。Unicode と FOR BIT DATA には、互換性がありません。
2. FOR BIT DATA を持つものを除いて、文字ストリングは 2 進ストリングとの互換性がありません。FOR BIT DATA 文字ストリングと 2 進ストリングは互換性があると見なされ、ターゲットのデータ・タイプに基づいて埋め込みが実行されます。例えば、FOR BIT DATA 列値を固定長 2 進変数に割り当てる場合、必要な埋め込みでは埋め込みバイトとして X'00' を使用します。
3. 特殊タイプの値は、同じ特殊タイプで定義された値とのみ比較が可能です。データベース・マネージャーは、一般に、特殊タイプの値とそのソース・データ・タイプ間の割り当てをサポートしています。追加情報については、124 ページの『特殊タイプの割り当て』を参照してください。

   配列タイプの値は、同じ配列タイプで定義された値とのみ比較が可能です。配列タイプの値は、同じタイプの配列に割り当てることができます。追加情報については、126 ページの『配列タイプの割り当て』を参照してください。
4. 10 進数とは、パック 10 進数とゾーン 10 進数の両方を指します。
5. データ・リンク・オペランドは、別のデータ・リンク・オペランドに対してのみ割り当てることができ、どんなデータ・タイプについても比較はできません。
6. ROWID オペランドは、別の ROWID オペランドに対してのみ割り当てることができ、どんなデータ・タイプについても比較はできません。
7. 文字ストリングとグラフィック・ストリングを XML 列に割り当てることはできます。ただし、XML は、文字ストリング列またはグラフィック・ストリング列に割り当てることはできません。比較の場合は、IS NULL 述部で比較するというのが、XML を比較する唯一の方法です。

割り当て演算には、以下のものに NULL 値を割り当てることはできないという基本的な規則があります。

- NULL 値を組み込めない列
- 関連する標識変数を持たないホスト変数
- プリミティブ・タイプの Java ホスト変数

標識変数については、172 ページの『ホスト変数に対する参照』を参照してください。

NULL 値が関係する比較での NULL 値の具体的な処理については、比較演算の説明を参照してください。

数値割り当て

数値割り当てでは、オーバーフローは認められません。

- 厳密な数値データ・タイプへの割り当て時には、その数値の整数部分のいずれかの桁が除去されるとオーバーフローが発生します。数値の小数部分は、必要があれば切り捨てることができます。
- 近似の数値データ・タイプまたは 10 進浮動小数点数への割り当て時には、その数値の整数部分の最上位桁が除去されるとオーバーフローが起こります。浮動小数点数および 10 進浮動小数点数の場合、その数値の整数部分は、浮動小数点数または 10 進浮動小数点数が、無制限の精度を持つ 10 進数に変換された場合の結果として得られる数値となります。必要な場合に丸めを行うと、その数値の最下位桁が除去される場合があります。

  10 進浮動小数点数の場合、その数値の整数部分の切り捨ては認められており、SQL_DECFLOAT_WARNINGS 照会オプションに *YES を指定すると警告が出され、結果が無限大となります。

浮動小数点数の場合は、アンダーフローも認められません。アンダーフローは、ゼロ以外の最上位桁が除去される場合に 1 と -1 の間の数値で起こります。 10 進浮動小数点数ではアンダーフローが認められており、丸めモードに応じて、結果がゼロ、または表現可能な最小の正数または最大の負数となります。
SQL_DECFLOAT_WARNINGS 照会オプションに *YES を指定すると、警告が返されます。

10 進浮動小数点数の丸めモードについては、 154 ページの『CURRENT DECFLOAT ROUNDING MODE』を参照してください。

標識変数を持つホスト変数への割り当て時にオーバーフローまたはアンダーフローが起こると、エラーの代わりにオーバーフロー警告またはアンダーフロー警告が返されます。この場合、数値はホスト変数に割り当てられず、標識変数は -2 にセットされます。

整数に対する割り当て

10 進数、浮動小数点数、または 10 進浮動小数点数を 2 進整数の列または変数に割り当てると、その数値の小数部分が除去されます。その結果、1 と -1 の間の数値が 0 に減らされます。

10 進数に対する割り当て

整数を 10 進数の列または変数に割り当てると、最初にその整数が一時的に 10 進数に変換され、次に、必要があれば、その一時的な 10 進数の精度と位取りがターゲットの精度と位取りに変換されます。一時的な 10 進数の精度と位取りは、短精度整数では 5,0、長精度整数では 11,0、 64 ビット整数では 19,0 です。

10 進数を 10 進数の列または変数に割り当てると、必要に応じて、数値の精度および位取りが、ターゲットの精度や位取りに変換されます。先行ゼロが必要な数だけ追加されます。また、10 進数の小数部分では、必要な数のゼロが末尾に追加されるか、末尾桁が必要な数だけ除去されます。

浮動小数点数を 10 進数の列または変数に割り当てると、その数値は、まず一時的に精度 63、位取り 63 - (*p*-*s*) の 10 進数に変換されます (*p* および *s* は 10 進数の列または変数の精度および位取りです)。その後に、必要な場合は、その一時的な 10 進数がターゲットの精度と位取りまで切り捨てられます。その結果、その 10 進

数の列または変数内の表現可能な最小の正数より小さい、または最大の負数より大きい、1 と -1 の間の数値が 0 に減らされます。

10 進浮動小数点数を 10 進数の列または変数に割り当てると、その数値はその 10 進数の列または変数の精度と位取りに丸められます。その結果、その 10 進数の列または変数内の表現可能な最小の正数より小さい、または最大の負数より大きい 1 と -1 の間の数値が 0 に減らされるか、または丸めモードに応じて、その 10 進数の列または変数内の表現可能な最小の正数または最大の負数に丸められます。

注: 10 進数とは、パック 10 進数とゾーン 10 進数の両方を指します。位取りがある 2 進整数は、10 進数への割り当ての規則に従います。

注: ファイルから取り出す 10 進数データが SQL CREATE TABLE ステートメントで作成されたものではない 場合は、10 進数フィールドに無効なデータが含まれていることがあります。この場合、そのデータは保管されるのと同じように戻され、警告あるいはエラー・メッセージは出されません。SQL CREATE TABLE ステートメントによって作成された表には、無効な 10 進数データが含まれることを許しません。

浮動小数点数への割り当て

浮動小数点数は、実数の近似値です。したがって、整数、10 進数、浮動小数点数、または 10 進浮動小数点数を浮動小数点数の列または変数に割り当てると、その結果が元の数値と異なる場合があります。この数値は浮動小数点数の算術計算を使用して、その浮動小数点数の列または変数の精度に丸められます。

10 進浮動小数点数への割り当て

整数を 10 進浮動小数点数の列または変数に割り当てると、最初にその数値は一時的に 10 進数に変換され、次に 10 進浮動小数点数に変換されます。一時的な 10 進数の精度と位取りは、短精度整数では 5,0、長精度整数では 11,0、 64 ビット整数では 19,0 です。丸めは、BIGINT を DECFLOAT(16) の列または変数に割り当てるときに行われる場合があります。

10 進数を 10 進浮動小数点数の列または変数に割り当てると、その数値はターゲットの精度 (16 または 34) に変換されます。先行ゼロは除去されます。10 進数の精度と位取りおよびターゲットの精度によっては、値が丸められる可能性があります。

浮動小数点数を 10 進浮動小数点数の列または変数に割り当てると、まずその数値は一時的に浮動小数点数のストリング表現に変換されます。次にそのストリング表現が 10 進浮動小数点数に変換されます。

DECFLOAT(16) の数値を DECFLOAT(34) の列または変数に割り当てると、結果として得られる値は DECFLOAT(16) の数値と同一です。

DECFLOAT(34) の数値を DECFLOAT(16) の列または変数に割り当てると、ソースの指数が結果の形式の対応する指数に変換されます。 DECFLOAT(34) の数値の仮数は、ターゲットの精度に丸められます。 10 進浮動小数点数の丸めモードについて詳しくは、 154 ページの『CURRENT DECFLOAT ROUNDING MODE』を参照してください。

COBOL および RPG の整数に対する割り当て

COBOL および RPG の短整数または長整数のホスト変数への割り当てでは、そのホスト変数に指定された位取りが考慮されます。ただし、整数のホスト変数への割り当てでは、整数の全桁が使用されます。したがって、COBOL のデータ項目や RPG のフィールドに入っている値が、ホスト変数に関して指定された最大の精度を超える場合があります。

例:

- COBOL において、COL1 に値 12345 が入っているとします。以下の SQL ステートメントによって、A が 4 桁しか定義されていなくても、結果として A には値 12345 が入ります。

```
01  A  PIC  S9999  BINARY.
EXEC SQL SELECT COL1
         INTO :A
         FROM TABLEX
END-EXEC.
```

- ただし、以下の COBOL ステートメントでは (12345 ではなく) 2345 が A に入ります。

```
MOVE 12345 TO A.
```

ストリングから数値への代入

ストリングを数値データ・タイプに割り当てると、CAST 指定の規則に基づいて、ストリングがターゲットの数値データ・タイプに変換されます。詳しくは、 218 ページの『CAST の指定』を参照してください。

ストリング割り当て

ストリングの割り当てには、次の 2 つのタイプがあります。

- 記憶域割り当て は、列、関数またはプロシージャーのパラメーター、または遷移変数に値が割り当てられるときに行われます。
- 検索割り当て は、変数 (パラメーターおよび遷移変数ではない) に値が割り当てられるときに行われます。[26]

2 進ストリングの割り当て

割り当てのターゲットがバイナリー・ストリングの場合には、以下の規則が当てはまります。

ストレージ割り当て

基本規則は次のとおりです。すなわち、列、関数またはプロシージャーのパラメーター、および遷移変数に割り当てられるストリングの長さは、その列またはパラメーターの長さ属性を超えてはならないというものです。ストリングがその列、パラメーター、または遷移変数の長さ属性よりも長い場合は、エラーが戻されます。後続 16 進ゼロ (X'00') は通常、ストリングの長さに含まれます。ただし、記憶域割り当ての場合は、後続 16 進ゼロはそのストリングの長さに含まれません。

---

26. SQL 変数や SQL パラメーターに値が割り当てられ、標準オプションが指定された場合は、記憶域割り当て規則が適用されます。 標準オプションについては、 xi ページの『標準への準拠』を参照してください。

固定長 2 進ストリングの列、パラメーター、または遷移変数にストリングを割り当てるときに、ストリングの長さがターゲットの長さ属性よりも短い場合は、そのストリングの右側に必要な数の 16 進ゼロが埋め込まれます。

検索割り当て

変数 (パラメーターおよび遷移変数ではない) に割り当てるストリングの長さは、その変数の長さ属性を超えても構いません。変数へのストリングの割り当てで、ストリングの長さがその変数の長さ属性よりも長い場合は、必要な数のバイトだけを残してストリングの右側が切り捨てられます。この切り捨てが行われると、 SQL 診断領域の RETURNED_SQLSTATE 条件領域項目に '01004' という SQLSTATE が割り当てられます (または、SQLCA の SQLWARN1 フィールドに値として 'W' が割り当てられます)。

固定長 2 進ストリングの変数にストリングを割り当てるときに、ストリングの長さがターゲットの長さ属性よりも短い場合は、そのストリングの右側に必要な数の 16 進ゼロが埋め込まれます。

長さ *n* のストリングを、*n* より大きい最大長を持つ可変長ストリングの変数に割り当てた場合は、変数の *n* 番目のバイト以後のバイトは未定義になります。

文字ストリングとグラフィック・ストリングの割り当て

割り当てのターゲットがストリングの場合には、以下の規則が当てはまります。

日付/時刻のデータ・タイプについては、 120 ページの『日時割り当て』を参照してください。割り当て (特に変数への割り当て) に特殊タイプが関係している場合の特別の考慮事項については、 124 ページの『特殊タイプの割り当て』を参照してください。

数値からストリングへの代入

数値をストリング・データ・タイプに割り当てると、CAST 指定の規則に基づいて、数値がターゲットのストリング・データ・タイプに変換されます。詳しくは、218 ページの『CAST の指定』を参照してください。

記憶域割り当て:

基本規則は次のとおりです。すなわち、列、関数またはプロシージャーのパラメーター、および遷移変数に割り当てられるストリングの長さは、その列またはパラメーターの長さ属性を超えてはならないというものです。ストリングがその列、パラメーター、または遷移変数の長さ属性よりも長い場合は、エラーが戻されます。末尾ブランクは通常、ストリングの長さに含まれます。ただし、記憶域割り当ての場合は、末尾ブランクはそのストリングの長さに含まれません。

固定長ストリングの列、パラメーター、または遷移変数にストリングを割り当てる際に、ストリングの長さがターゲットの長さ属性よりも短い場合には、そのストリングの右側に必要な数の 1 バイト、2 バイト、 UTF-16 または UCS-2 のブランク

27. UTF-16 または UCS-2 では、コード・ポイント X'0020' および X'3000' でブランク文字を定義しています。データベース・マネージャーは、コード・ポイント X'0020' の位置にあるブランクを埋め込みに使用します。データベース・マネージャーは、UTF-8 でコード・ポイント X'20' のブランクを埋め込みます。

が埋め込まれます。[27] 埋め込み文字は、ビット・データの場合でも、常にブランクです。

検索割り当て:

変数 (パラメーターおよび遷移変数ではない) に割り当てるストリングの長さは、その変数の長さ属性を超えても構いません。変数へのストリングの割り当てで、ストリングの長さがその変数の長さ属性よりも大きい場合、必要な数の文字だけを残してストリングの右側が切り捨てられます。この切り捨てが行われると、 SQL 診断領域の RETURNED_SQLSTATE 条件領域項目に '01004' という SQLSTATE が割り当てられます (または、SQLCA の SQLWARN1 フィールドに値として 'W' が割り当てられます)。さらに、変数に標識変数がある場合は、その標識変数にストリングの元の長さがセットされます。 C の NUL 終了ホスト変数で NUL 終了文字だけが切り捨てられ、*NOCNULRQD オプションが、 CRTSQLCI または CRTSQLCPPI コマンド (あるいは、SET OPTION ステートメントでの CNULRQD(*NO)) で指定されていた場合は、 SQL 診断領域の RETURNED_SQLSTATE 条件領域項目に '01004' という SQLSTATE が割り当てられ (または、値として 'N' が SQLCA の SQLWARN1 フィールドに割り当てられ)、 NUL 終了文字は変数には入りません。

固定長変数へストリングを割り当てる際に、ストリングの長さがターゲットの長さ属性よりも短い場合には、そのストリングの右側に必要な数の 1 バイト、2 バイト、 UTF-16 または UCS-2 のブランクが埋め込まれます。[27] 埋め込み文字は、ビット・データの場合でも、常にブランクです。

長さ *n* のストリングを、*n* より大きい最大長を持つ可変長ストリングの変数に割り当てた場合は、変数の *n* 番目以後の文字は未定義になります。

混合ストリングへの割り当て

ストリングに混合データが入っている場合は、割り当て規則によって、一連の 2 バイト・コードの途中で切り捨てが必要になることがあります。一連の 2 バイト文字の終わりを示すシフトイン文字の消失を防ぐために、ストリングの終わりから追加の文字が切り捨てられ、シフトイン文字が追加されます。この切り捨ての結果、シフトアウト文字とそれに対応するシフトイン文字で囲まれている文字数は、常に偶数バイトになります。

C の NUL 終了ストリングへの割り当て

長さ *n* のストリングが、*n+1* を超える長さの C の NUL 終了ストリング変数に割り当てられる場合、

- *CNULRQD オプションが、CRTSQLCI または CRTSQLCPPI コマンド (あるいは、SET OPTION ステートメントでの CNULRQD(*YES)) で指定されていた場合は、そのストリングの右側が x-n-1 個のブランクで埋め込まれます。ここで、x は変数の長さです。埋め込みが行われたストリングが変数に割り当てられ、NUL 終了文字が次の文字位置に入れられます。
- *NOCNULRQD プリコンパイラー・オプションが、CRTSQLCI または CRTSQLCPPI コマンド (あるいは、SET OPTION ステートメントでの

CNULRQD(*NO)) で指定されていた場合は、ストリングの右側の埋め込みはありません。そのストリングが変数に割り当てられ、NUL 終了文字が次の文字位置に入れられます。

割り当ての際の変換規則

列、変数、パラメーターのいずれかの割り当てられるストリングは、必要に応じて、最初にターゲットのコード化文字セットに変換されます。文字変換が必要になるのは、以下の条件にすべて該当する場合だけです。

- 両者の CCSID が異なっている。
- どちらの CCSID も 65535 ではない。
- ストリングが NULL でなく、空でもない。
- 2 つの CCSID 間の変換が必要。詳しくは、 43 ページの『コード化文字セットと CCSID』を参照してください。

次のような場合には、エラーが生じます。

- CCSID ペア間の変換が定義されていない。詳しくは、 43 ページの『コード化文字セットと CCSID』を参照してください。
- 列への割り当てまたは標識変数を持たないホスト変数への割り当て演算で、ストリングの文字を変換できなかった。例えば、2 バイト文字 (DBCS) を、1 バイト文字 (SBCS) の CCSID を持つ列やホスト変数に変換できなかったような場合。

次のような場合には、警告が出されます。

- ストリングの文字が、置換文字に変換された。
- 列への割り当てでも、標識変数を持たないホスト変数への割り当てでもない割り当てで、ストリングの文字を変換できなかった。例えば、DBCS 文字は、SBCS CCSID を持つホスト変数には変換できません。この場合は、ホスト変数にはストリングが割り当てられず、標識変数に -2 がセットされます。

日時割り当て

日付、時刻、またはタイム・スタンプの列または変数に割り当てる値は、日付、時刻、またはタイム・スタンプであるか、あるいは、日付、時刻、またはタイム・スタンプの有効なストリング表現である必要があります。

DATE 列または DATE 変数に割り当てられる値は、日付またはタイム・スタンプか、日付またはタイム・スタンプの有効なストリング表現でなければなりません。タイム・スタンプ値を DATE データ・タイプに割り当てると、日付の部分が抜き出され、時刻の部分は無視されます。 日付を割り当てることができるのは、DATE または TIMESTAMP 列、ストリング列、DATE または TIMESTAMP 変数、ストリング変数に限られます。

TIME 列または TIME 変数に割り当てられる値は、時刻またはタイム・スタンプか、時刻またはタイム・スタンプの有効なストリング表現でなければなりません。タイム・スタンプ値を TIME データ・タイプに割り当てると、日付の部分は無視され、時刻の部分は小数秒を切り捨てて抜き出されます。 時刻を割り当てることができるのは、TIME 列、ストリング列、TIME 変数、ストリング変数に限られます。

TIMESTAMP 列または TIMESTAMP 変数に割り当てられる値は、日付またはタイム・スタンプか、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付またはタイム・スタンプのストリング表現が使用される場合、ターゲットのタイム・スタンプと同じ精度のタイム・スタンプに暗黙的にキャストされます。日付値を TIMESTAMP データ・タイプに割り当てると、不足する時刻情報はすべてゼロであると想定されます。 タイム・スタンプを割り当てることができるのは、DATE 列、TIME 列、TIMESTAMP 列、ストリング列、DATE 変数、TIME 変数、TIMESTAMP 変数、ストリング変数に限られます。タイム・スタンプ値をより精度が低いタイム・スタンプに代入すると、余分の小数秒は切り捨てられます。タイム・スタンプ値をより精度が高いタイム・スタンプに代入すると、不足する桁はゼロであるとみなされます。

日付/時刻の値がストリングの変数や列に割り当てられる場合、その値はストリング表現に変換されます。日付、時刻、タイム・スタンプのどの部分からも、先行ゼロは除去されません。ターゲットの必要な長さは、ストリング表現の形式に応じて変わります。ターゲットの長さが必要な長さよりも長い場合は、変換結果の右側にブランクが埋め込まれます。ターゲットの長さが必要な長さよりも短い場合は、その結果は、関与する日付/時刻の値のタイプとターゲットのタイプによって変わります。

- ターゲットがストリングの列である場合、切り捨ては許されません。以下の規則が適用されます。

  DATE

  – 日付の形式が、*ISO、USA、*EUR、または *JIS の場合、列の長さ属性は 10 以上でなければなりません。日付の形式が *YMD、*MDY、または *DMY の場合、列の長さ属性は 8 以上でなければなりません。日付の形式が *JUL の場合、変数の長さは 6 以上でなければなりません。

  TIME

  – ターゲットの列の長さ属性は、8 以上でなければなりません。

  TIMESTAMP

  – 列の長さ属性は、少なくとも、 TIMESTAMP(0) の場合は 19、TIMESTAMP(p) の場合は 20+p でなければなりません。

- ターゲットが変数である場合は、次の規則が適用されます。

  DATE

  – 日付の形式が *ISO、*USA、*EUR、または *JIS の場合、変数の長さは 10 以上でなければなりません。日付の形式が *YMD、*MDY、または *DMY の場合、変数の長さは 8 以上でなければなりません。日付の形式が *JUL の場合、変数の長さは 6 以上でなければなりません。

  TIME

  – *USA 形式を使用するときは、変数の長さは 8 以上でなければなりません。この形式には、秒は含まれていません。

  – *ISO、*EUR、*JIS、または *HMS の時刻形式を使用している場合は、変数の長さは、5 以上でなければなりません。変数の長さが 5、6、または 7 の場合、時刻の秒の部分が結果から除去され、SQLWARN1 に 'W' がセットされ

ます。この場合、標識変数があれば、時刻の秒の部分はその標識変数に割り当てられます。また、長さが 6 または 7 の場合には、変数の値が時刻の有効なストリング表現になるように、ブランクの埋め込みが行われます。

TIMESTAMP

– 変数の長さは、19 以上でなければなりません。長さが 19 から 31 までの場合、タイム・スタンプはストリングと同様に切り捨てられ、小数秒の 1 つ以上の桁が脱落します。 長さが 20 の場合、値が精度 0 のタイム・スタンプを表す有効なストリング表現になるように、末尾の小数点はブランクで置き換えられます。

XML の割り当て

割り当てのターゲットが XML の場合には、以下の規則が当てはまります。

XML 割り当ての一般規則として、XML 値だけを XML 列または XML 変数に割り当てることができます。その規則の例外は次のとおりです。

- 入力 **XML** 変数を処理する場合**:** これは XML 割り当て規則の特殊ケースです。変数はストリング値を基にしているためです。SQL 内で XML への割り当てを行うため、ストリング値が暗黙で解析されて XML 値になります。その際、CURRENT IMPLICIT XMLPARSE OPTION 特殊レジスターの設定が使用されます。これにより、空白を残すが除去するかが決まります。ただし、変数が XMLVALIDATE 関数の引数である場合は例外で、その場合は不要な空白が常に除去されます。
- データ・タイプ **XML** の入力パラメーター・マーカーにストリングを割り当てる場合**:** 入力パラメーター・マーカーの暗黙的または明示的なデータ・タイプが XML の場合、そのパラメーター・マーカーに割り当てる値は、文字ストリング変数、グラフィック・ストリング変数、またはバイナリー・ストリング変数のいずれでも構いません。その場合、ストリング値が暗黙で解析されて XML 値になります。その際、CURRENT IMPLICIT XMLPARSE OPTION 特殊レジスターの設定が使用されます。これにより、空白を残すが除去するかが決まります。ただし、パラメーター・マーカーが XMLVALIDATE 関数の引数である場合は例外で、その場合は不要な空白が常に除去されます。
- データ変更ステートメントで **XML** 列にストリングを直接割り当てる場合**:** データ変更ステートメントでタイプが XML の列に直接割り当てを行う場合、割り当てられる式は文字ストリング、グラフィック・ストリング、またはバイナリー・ストリングにもできます。その場合、XMLPARSE (DOCUMENT *expression* STRIP WHITESPACE) の結果が、ターゲット列に割り当てられます。サポートされるストリングのデータ・タイプは、XMLPARSE 関数用のサポートされている引数で定義されます。この例外は、XML タイプの SQL パラメーターにも適用されます。
- 取り出し時に **XML** をストリングに割り当てる場合**:** 組み込み SQL 内で FETCH または SELECT INTO を使用して、XML 値を変数に取り出す場合、その変数のデータ・タイプは、CLOB、DBCLOB、または BLOB のいずれでも構いません。XML 値は、変数の CCSID でエンコードされたストリングに暗黙的に直列化されます。他のアプリケーション・プログラミング・インターフェース (CLI、JDBC、または .NET など) を使用する場合、該当するアプリケーション・プログラミング・インターフェースでサポートされている文字、グラフィッ

ク、またはバイナリー・ストリングのタイプで XML 値を取り出すことができます。これらのどの場合でも、XML 値は、 101 ページの『XML 値』で説明されているように、QAQQINI ファイルによって判別される CCSID でエンコードされたストリングに暗黙的に直列化されます。

FETCH、SELECT INTO、SET、および VALUES INTO ステートメントの場合、文字ストリング、グラフィック・ストリング、またはバイナリー・ストリングの値を XML 変数に取り出すことはできません。
INSERT、UPDATE、MERGE、SET、VALUES INTO、および CALL ステートメントの場合には、XML 変数内の値を、データ・タイプが文字、グラフィック、またはバイナリー・ストリングの列、SQL 変数、または SQL パラメーターに割り当てることはできません。

データ・リンクの割り当て

値をデータ・リンク列に割り当てると、値のリンケージ属性が空であるか、列が NO LINK CONTROL で定義されているのではない限り、ファイルへのリンクが確立します。リンクされた値が既に列に存在する場合は、そのファイルはリンク解除されます。また、リンクされた値が既に存在する場合に NULL 値を割り当てても、古い値に関連したファイルはリンク解除されます。

列に既に存在するものと同じデータ位置を、アプリケーションが与えると、そのリンクは保存されます。このことが生じる理由は 2 つあります。

- コメントが変更された。
- 表がリンク保留状態にある場合は、列にあるものと同じリンク属性を与えることによって、表のリンクを復元することができます。

データ・リンク値は、DLVALUE スカラー関数を使用することによって、列に割り当てることができます。 DLVALUE スカラー関数は、後で列に割り当てることができる新しいデータ・リンク値を作成します。値がコメントしか含んでいないか、URL がまったく同じではない限り、割り当ての行動によってファイルへリンクします。

値をデータ・リンク列へ割り当てる際に、次のエラー状態が起こる可能性があります。

- データ・ロケーション (URL) 形式が無効。
- ファイル・サーバーがこのデータベースに登録されていない。
- 無効なリンク・タイプが指定された。
- コメントまたは URL の長さが無効。

  URL パラメーターまたは関数結果のサイズは、入力と出力の両方で同じであり、データ・リンク列の長さに制限されることに注意してください。ただし、場合によっては、戻される URL 値にアクセス・トークンが付加されることがあります。このような可能性がある状態では、出力の場所ではアクセス・トークン用に十分な記憶域とデータ・リンク列に十分な長さが必要になります。したがって、入力で用意される完全に拡張された形式でのコメントと URL の実際の長さを、出力の記憶域に適応するように制限する必要があります。制限された長さを超えた場合には、このエラーが起こります。

割り当てでリンクも作成する場合は、以下のエラーが生じる可能性があります。

- 現在、ファイル・サーバーが使用不能。
- ファイルが存在しない。
- 参照したファイルがリンク用にアクセスできない。
- ファイルが既に別の列へリンクされている。

  このエラーは、異なるリレーショナル・データベースへのリンクの場合でも生じることに注意してください。

さらに、割り当てで既存のリンクを取り除く場合には、以下のエラーが生じる可能性があります。

- 現在、ファイル・サーバーが使用不能。
- 参照保全制御を伴うファイルが、Db2 UDB データ・リンク・ファイル・マネージャーに従った正しい状態にない。

データ・リンク値は、スカラー関数 (DLLINKTYPE および DLURLPATH など) を使用することにより、データベースから検索することができます。その後で、これらのスカラー関数の結果を変数に割り当てることができます。

通常は、検索時にファイル・サーバーをアクセスすることは行われないということに注意してください。[28] したがって、後でファイル・システム・コマンドを使用してファイル・サーバーにアクセスしようとすると、失敗する可能性があります。

表はリンク保留状態にあるため、データ・リンク値を検索すると警告が戻されることがあります。

行 ID の割り当て

行 ID の値を割り当てることができるのは、行 ID データ・タイプを持つ列、パラメーター、または変数のみです。 ROWID 列の値が有効であるためには、その列が GENERATED BY DEFAULT として定義されているか、OVERRIDING SYSTEM VALUE が指定されていることが必要です。 ROWID 列のある各表には、各 ROWID 値をすべて固有の値にするための固有制約が暗黙的に追加されます。この列に指定する値は、Db2 for z/OS または Db2 for i によって既に生成されている有効な行 ID でなければなりません。

特殊タイプの割り当て

特殊タイプの変数への割り当てに適用される規則は、特殊タイプが関係する他のどの割り当てについての規則とも異なります。

28. パスに関連したプレフィックス名を決めるためには、ファイル・サーバーへのアクセスが必要になることがあります。これは、ファイル・サーバーのマウント・ポイントを移動するとファイル・サーバー側で変更することができます。サーバーのファイルを最初にアクセスすると、必要とする値をファイル・サーバーから検索し、そのファイル・サーバーのデータ・リンク値を後で検索するために、データベース・サーバーでキャッシュに入れます。ファイル・サーバーにアクセスできない場合は、エラーが戻されます。

変数への割り当て

特殊タイプの変数への割り当ては、特殊タイプのソース・データ・タイプに基づいています。したがって、特殊タイプのソース・データ・タイプを変数へ割り当て可能な場合にのみ、特殊タイプの値は変数に割り当てることができます。

例:

特殊タイプ AGE が以下の SQL ステートメントを使用して作成されたとします。

```
CREATE TYPE AGE AS SMALLINT WITH COMPARISONS
```

このステートメントを実行すると、次のキャスト関数も生成されます。

```
AGE (SMALLINT) RETURNS AGE
AGE (INTEGER) RETURNS AGE
SMALLINT (AGE) RETURNS SMALLINT
```

次に、表 STUDENTS の STU_AGE 列に特殊タイプ AGE が定義されていると想定します。生徒の年齢を、INTEGER データ・タイプを持つホスト変数 HV_AGE に次のように有効に割り当てます。

```
SELECT  STU_AGE INTO :HV_AGE FROM STUDENTS WHERE STU_NUMBER = 200
```

特殊タイプ (SMALLINT) のソース・データ・タイプがホスト変数 (INTEGER) に割り当て可能なため、特殊タイプの値はホスト変数 HV_AGE に割り当てることができます。

変数以外への割り当て

特殊タイプは、割り当てのソースあるいはターゲットのいずれにもなり得ます。割り当ては、割り当てられる値のデータ・タイプが、ターゲットのデータ・タイプにキャスト可能であるかどうかに基づいています。 109 ページの『データ・タイプ間のキャスト』は、特殊タイプが含まれる場合にサポートされるキャストを示しています。そのため、特殊タイプの値は、以下の場合に変数以外のいずれのターゲットにも割り当てることができます。

- 割り当てのターゲットが同じ特殊タイプを持っている、あるいは
- 特殊タイプはターゲットのデータ・タイプにキャスト可能である。

以下の場合には、いずれの値も特殊タイプに割り当てることができます。

- 割り当てられる値がターゲットと同じ特殊タイプを持っている、あるいは
- 割り当てられる値のデータ・タイプは、ターゲットの特殊タイプにキャスト可能である。

例:

特殊タイプ AGE のソース・データ・タイプが SMALLINT であるとします。

```
CREATE TYPE AGE AS SMALLINT WITH COMPARISONS
```

次に、2 つの表 TABLE1 および TABLE2 が 4 つの同一の列記述で作成されたとします。

```
AGECOL    AGE
SMINTCOL  SMALLINT
INTCOL    INTEGER
DECCOL    DEC(6,2)
```

次の SQL ステートメントを使用し、X と Y にいろいろな値を代入して、TABLE1 のいろいろな列に TABLE2 から値を挿入する場合、表 19 はその割り当てが有効であるかどうかを示しています。

```
INSERT INTO  TABLE1 (Y) SELECT X FROM TABLE2
```

表 *19.* いろいろな割り当ての評価 *(*例えば、*INSERT* で*)*

TABLE2.X	**TABLE1.Y**	有効	理由
AGECOL	AGECOL	はい	ソースとターゲットが同じ特殊タイプである
SMINTCOL	AGECOL	はい	SMALLINT は AGE にキャスト可能 (AGE のソース・タイプは SMALLINT のため)
INTCOL	AGECOL	はい	INTEGER を AGE にキャストできる (AGE のソース・タイプは SMALLINT であるため)
DECCOL	AGECOL	いいえ	DECIMAL は AGE にキャスト不可能
AGECOL	SMINTCOL	はい	AGE はそのソース・タイプである SMALLINT にキャスト可能
AGECOL	INTCOL	いいえ	AGE は INTEGER にキャスト不可能
AGECOL	DECCOL	いいえ	AGE は DECIMAL にキャスト不可能

配列タイプの割り当て

SQL の配列変数または配列パラメーターに対する配列タイプの割り当ての妥当性は、以下の規則によって決まります。

- 割り当ての右側が SQL 配列変数またはパラメーター、 TRIM_ARRAY 関数、または CAST 式のいずれかである場合、そのタイプは割り当ての左側の SQL 配列変数またはパラメーターと同じタイプでなければなりません。
- 割り当ての右側が配列コンストラクターまたは ARRAY_AGG 関数であれば、配列タイプは、左側にある SQL の配列変数または配列パラメーターのタイプに暗黙的にキャストされます。

LOB ロケーターへの割り当て

LOB ロケーターを使用すれば、どんなストリング・データでも参照できます。リモート・サーバー上にあるカーソルの最初のフェッチに LOB ロケーターを使用する場合は、それ以降のすべてのフェッチにも LOB ロケーターを使用する必要があります。ただし、*NOOPTLOB プリコンパイル・オプションを使用する場合は別です。

数値比較

数値は代数の場合と同じように比較されます。つまり、符号が考慮されます。例えば、–2 は +1 より小さくなります。

一方の数値が整数で、もう一方の数値が 10 進数の場合、その整数を 10 進数に変換した整数の一時的なコピーを使用して比較が行われます。

10 進数または位取りがゼロ以外の 2 進数を比較するときに、比較する数値の位取りが異なる場合は、一方の数値の一時的なコピー (両者の小数部が同じ桁数になるように、一方の数値の小数部の桁数を後続ゼロによって増やしたもの) を使用して比較が行われます。

一方の数値が浮動小数点数で、もう一方の数値が整数、10 進数、または単精度の浮動小数点数である場合は、2 番目の数値を倍精度の浮動小数点数に変換し、その一時的なコピーを使用して比較が行われます。ただし、単精度浮動小数点の列を定数と比較するときに、その定数が単精度の浮動小数点数で表される場合は、定数の単精度形式を使用して比較が行われます。

2 つの浮動小数点数は、両者の正規形のビット構成が同一である場合にのみ等しくなります。

一方の数値が DECFLOAT で、もう一方の数値が整数、10 進数、単精度の浮動小数点数、または倍精度の浮動小数点数である場合、2 番目の数値を DECFLOAT に変換し、その一時的なコピーを使用して比較が行われます。

一方の数値が DECFLOAT(16) で、もう一方の数値が DECFLOAT(34) の場合、比較が行われる前に DECFLOAT(16) の値が DECFLOAT(34) に変換されます。

DECFLOAT データ・タイプは正のゼロと負のゼロを両方ともサポートします。正のゼロと負のゼロは異なる 2 進数表現で表されますが、等号 (=) 述部では正のゼロと負のゼロの比較に対して真が返されます。

DECFLOAT データ・タイプでは、同じ数値に対して複数のビット表現が可能です。例えば、2.00 と 2.0 は数値的には同一でもビット表現が異なる 2 つの数値です。 = (等号) 述部では、比較 2.0 = 2.00 に対して真が返されます。2.0 = 2.00 が真であれば、 2.0 < 2.00 は偽です。ここで説明する動作は、DECFLOAT 値のすべての比較で真を保持します (例えば、UNION、SELECT DISTINCT、COUNT DISTINCT、基本述部、IN 述部、MIN、MAX などの場合)。例えば、

```
SELECT 2.0 FROM SYSIBM.SYSDUMMY
UNION
SELECT 2.00 FROM SYSIBM.SYSDUMMY
```

上記により、1 行のデータが生成されます。この照会では、返される値 (2.0 または 2.00) は不定です。

2 進レベルで比較を実行する場合は、関数 COMPARE_DECFLOAT および TOTALORDER を使用することができます。例えば、2.0<>2.00 の比較の場合などです。こうした関数を使用する場合における 10 進浮動小数点値の比較順序は、-NaN < -sNaN < -Infinity < -0.10 <-0.100 < -0 < 0 < 0.100 < 0.10 < Infinity < sNaN < NaN となります。

DECFLOAT データ・タイプは、正と負の NaN (Quiet および Signaling) の指定、および正と負のInfinityの指定もサポートします。 SQL の観点から見ると、無限大 = 無限大、NaN = NaN、および sNaN = sNaN です。

DECFLOAT データ・タイプは、正と負の NaN (静止とシグナル) の指定、および正と負の無限大の指定もサポートします。

これらの特殊値の場合、以下の規則が比較の規則となります。

- 無限大の比較は、同じ符号 (正または負) の無限大のみに対して等しくなります。
- NaN の比較は、同じ符号 (正または負) の NaN のみに対して等しくなります。
- sNaN の比較は、同じ符号 (正または負) の sNaN のみに対して等しくなります。

ストリング・データ・タイプと数値データ・タイプを比較する場合は、ストリングが、同じ精度と位取りの数値データ・タイプに変換されるので、ストリングの内容は数字を表す有効なストリング表現である必要があります。

ストリングの比較

ストリング比較には 2 つのタイプがあります。

2 進ストリングの比較

2 進ストリングの比較では、照合順序は常に *HEX を使用し、各ストリングの対応するバイトが比較されます。さらに、2 つの 2 進ストリングの長さが同一の場合にのみ、その 2 つの 2 進ストリングは等しいと言えます。長いストリングと短いストリングがあって、短いほうのストリングの長さまで両方のストリングが等しい場合は、たとえ長いほうのストリングの残りのバイトが 16 進ゼロであっても、短いほうのストリングが長いほうのストリングよりも小さいと見なされます。また、2 進ストリングと文字ストリングの比較は、文字ストリングを 2 進ストリングにキャストしない限りできません。

文字ストリングとグラフィック・ストリングの比較

文字ストリングと Unicode グラフィック・ストリングの比較では、すべての SBCS データと混合データの単一バイト部分についてステートメントが実行される時点で有効な照合順序が使用されます。照合順序が *HEX の場合、各ストリングの対応するバイトが比較されます。その他の照合順序ではすべて、各ストリングの対応するバイトの重み付けされた値が比較されます。

ストリングの長さが異なる場合は、短い方のストリングの右側にブランクを埋め込んだ一時的コピーを使用して比較します。この埋め込みを行うのは、両方のストリングの長さを等しくするためです。埋め込み文字は、照合順序に関係なく、常にブランクです。ビット・データの場合も、ブランクを使用します。DBCS グラフィッ

ク・データの場合は、埋め込み文字は DBCS のブランク (x'4040') になります。Unicode グラフィック・データの場合、埋め込み文字は UTF-16 のブランクになります。[29]

2 つのストリングが等しくなるのは、次のいずれかに該当する場合です。

- 両方のストリングが空である。
- *HEX の照合順序を使用した場合、対応するバイトがすべて等しい。
- *HEX 以外の照合順序を使用した場合、対応するバイトの重み付けの値がすべて等しい。

空のストリングは、ブランクのストリングと同じです。等しくない 2 つのストリング間の関係は、それらのストリングの左端から調べて、最初に見つかった等しくないバイトの対の比較によって決定されます。この比較は、そのステートメントが実行される時点で有効な照合順序に従って行われます。

複数の環境でアプリケーションを実行する場合は、それぞれの環境で同じ結果を得るために同じ照合順序を使用する必要があります (照合順序は、それぞれの環境の CCSID に依存します)。 EBCDIC、ASCII、Db2 LUW について、米国英語の場合のデフォルトの照合順序の違いを以下の表にまとめます (以下の表は、それぞれの照合順序に基づいてソートしたリストになっています)。

表 *20.* 照合順序の違い

ASCII と **Unicode**	**EBCDIC**	**Db2 LUW** のデフォルト
0000	@@@@	0000
9999	co-op	9999
@@@@	coop	@@@@
COOP	piano forte	co-op
PIANO-FORTE	piano-forte	COOP
co-op	COOP	coop
coop	PIANO-FORTE	piano forte
piano forte	0000	PIANO-FORTE
piano-forte	9999	piano-forte

異なる長さを持つ 2 つの可変長ストリングが末尾ブランクの数だけが異なる場合には、等しくなります。そのような値の集合から 1 つの値を選択する演算では、選択される値は不定のものになります。このような不定な選択を伴う演算には、DISTINCT、MAX、MIN、UNION、 EXCEPT、INTERSECT、およびグループ化列の参照があります。グループ化列の参照に伴う不定な選択については、 818 ページの『GROUP-BY 文節』を参照してください。

比較の際の変換規則:

2 つのストリングを比較する場合、必要があれば、最初に一方のストリングがもう一方のストリングのコード化文字セットに変換されます。文字変換が必要になるのは、以下の条件にすべて該当する場合だけです。

- 2 つのストリングの CCSID が異なっている。

29. UTF-16 では、コード・ポイント X'0020' および X'3000' でブランク文字を定義しています。データベース・マネージャーは、コード・ポイント X'0020' の位置にあるブランクを埋め込みに使用します。

- どちらの CCSID も 65535 ではない。
- 変換する側として選択されたストリングが、NULL でも空でもない。
- 2 つの CCSID 間の変換が必要。詳しくは、 43 ページの『コード化文字セットと CCSID』を参照してください。

コード化体系が異なる 2 つのストリングを比較する場合、以下のように、必要な変換がストリングに適用されます。

表 21. 文字変換のための結果エンコード化体系の選択

	第 **2** オペランド			
第 **1** オペランド	**SBCS** データ	**DBCS** データ	混合データ	**Unicode** グラフィック・データ
SBCS データ	下記参照	第 2	第 2	第 2
DBCS データ	第 1	下記参照	第 2	第 2
混合データ	第 1	第 1	下記参照	第 2
Unicode グラフィック・データ	第 1	第 1	第 1	下記参照

それ以外の場合は、それぞれのオペランドのタイプに応じて、変換用に選択されるオペランドが決まります。以下の表は、オペランドのタイプに応じて、どちらのオペランドが変換されるオペランドとして選択されるかを示しています。

表 22. 文字変換するオペランドの選択

	第 **2** オペランド				
第 **1** オペランド	列値	演算による値	特殊レジスター	定数	変数
列値	第 2	第 2	第 2	第 2	第 2
演算による値	第 1	第 2	第 2	第 2	第 2
特殊レジスター	第 1	第 1	第 2	第 2	第 2
定数	第 1	第 1	第 1	第 2	第 2
変数	第 1	第 1	第 1	第 1	第 2

外部コード化体系のデータを含む変数は、何らかの演算で使用される前に、必ず固有のコード化体系へ有効に変換されます。上記の規則は、このような変換が既に行われていることを前提にしています。

ストリングの文字が変換できない場合や、CCSID 間の変換が定義されていない場合は、エラーが戻されます。詳しくは、 43 ページの『コード化文字セットと CCSID』を参照してください。 ストリングの文字が置換文字に変換された場合は、警告が出されます。

日付/時刻の比較

DATE、TIME、または TIMESTAMP 値は、同じデータ・タイプの別の値、同じデータ・タイプの日時定数、または、同じデータ・タイプのストリング表現と比較することができます。また、日付の値または日付のストリング表記を TIMESTAMP

と比較することもできます。この場合、日付の値にない時刻情報はすべてゼロであるとみなされます。 比較はすべて日時順に行われます。つまり、0001 年 1 月 1 日からの経過日数 (時間) が多ければ多いほど、より大きな値になります。

TIME の値および時刻の値のストリング表現を含む比較では、必ず秒も比較されます。ストリング表現で秒の部分を除去している場合は、その部分にゼロ秒があるものとして扱われます。時刻 24:00:00 は、時刻 00:00:00 より大きいものとして比較します。

TIMESTAMP 値を扱う比較は、次の規則に従って評価されます。

- 精度が異なるタイム・スタンプ値を比較する場合、より高い精度を使用して比較が行われます。秒の小数部分の足りない数字は、ゼロであると見なされます。
- タイム・スタンプ値と、タイム・スタンプ値のストリング表記とを比較する場合、まず、ストリング表記が TIMESTAMP(12) に変換されます。
- タイム・スタンプの比較は日時順に行われます。等しいと見なしてよい表記については考慮されません。したがって、次のような述部が成り立ちます。

```
TIMESTAMP('1990-02-23-00.00.00') > '1990-02-22-24.00.00'
```

XML 比較

XML データ・タイプは、XML データ・タイプを含めてどのデータ・タイプとも比較することができません。

データ・リンクの比較

DATALINK オペランドは、どのデータ・タイプについても直接の比較ができません。 DLCOMMENT、DLLINKTYPE、DLURLCOMPLETE、DLURLPATH、DLURLPATHONLY、 DLURLSCHEME、および DLURLSERVER スカラー関数を使用して、データ・リンクから文字ストリング値を取り出すことができます。これによって、そのデータ・リンクは他のストリングと比較することができるようになります。

行 ID の比較

ROWID オペランドは、どのデータ・タイプについても直接の比較ができません。ROWID のビット表現を比較するには、まず ROWID を文字ストリングにキャストしてください。

特殊タイプの比較

特殊タイプの値は、同じ特殊タイプの値とのみ比較が可能です。

例えば、特殊タイプ YOUTH および表 CAMP_DB2_ROSTER が、次の SQL ステートメントを使用して作成されると仮定します。

```
CREATE TYPE YOUTH AS INTEGER WITH COMPARISONS

CREATE TABLE CAMP_DB2_ROSTER
  ( NAME                VARCHAR(20),
    ATTENDEE_NUMBER     INTEGER NOT NULL,
    AGE                 YOUTH,
    HIGH_SCHOOL_LEVEL   YOUTH)
```

以下の比較は、AGE および HIGH_SCHOOL_LEVEL が同じ特殊タイプを持っているので有効です。

```
SELECT * FROM CAMP_DB2_ROSTER
  WHERE AGE > HIGH_SCHOOL_LEVEL
```

以下の比較は無効です。

```
SELECT * FROM CAMP_DB2_ROSTER                  ***INCORRECT***
  WHERE AGE > ATTENDEE_NUMBER
```

しかし、AGE と ATTENDEE_NUMBER の比較は、キャスト関数または CAST 指定を使用して、特殊タイプとソース・タイプの間でキャストすることによって、可能になります。以下の比較はすべて有効です。

```
SELECT * FROM CAMP_DB2_ROSTER
  WHERE AGE > YOUTH(ATTENDEE_NUMBER)

SELECT * FROM CAMP_DB2_ROSTER
  WHERE AGE > CAST( ATTENDEE_NUMBER AS YOUTH)

SELECT * FROM CAMP_DB2_ROSTER
  WHERE INTEGER(AGE) > ATTENDEE_NUMBER

SELECT * FROM CAMP_DB2_ROSTER
  WHERE CAST(AGE AS INTEGER) > ATTENDEE_NUMBER
```

配列タイプの比較

配列タイプ値の比較はサポートされていません。

配列のエレメントは、エレメントのデータ・タイプの比較規則に基づいて比較できます。

結果のデータ・タイプに関する規則

結果のデータ・タイプは、演算のオペランドに適用される規則によって決まります。このセクションでは、この規則について説明します。

この規則は、以下のものに適用されます。

- UNION、UNION ALL、EXCEPT、INTERSECT の各演算の対応する列
- CASE 式の結果の式
- スカラー関数 COALESCE、IFNULL、MAX、MIN、および VALUE の引数
- IN 述部の IN リストの式の値
- OLAP 指定での集約グループ範囲のための引数

演算子 /、*、+、および - を含む式の結果のデータ・タイプについては、 197 ページの『算術演算子を使用する式』を参照してください。CONCAT 演算子を含む式の結果のデータ・タイプについては、 202 ページの『連結演算子』を参照してください。

結果のデータ・タイプは、オペランドのデータ・タイプによって決まります。最初の 2 つのオペランドのデータ・タイプによって、中間結果のデータ・タイプが決まり、こうして決まったデータ・タイプと次のオペランドのデータ・タイプによって、新しい中間結果のデータ・タイプが決まり、さらに以下同様に続きます。こうして、最後の中間結果のデータ・タイプと最後のオペランドのデータ・タイプによって、結果のデータ・タイプが決まります。データ・タイプの各対ごとに、次の表に要約されている規則を順次適用することによって、結果のデータ・タイプが決まります。

どちらのオペランド列にも NULL が許されない場合は、結果列にも NULL は許されません。それ以外の場合は、結果列に NULL が許されます。

オペランド列のデータ・タイプと属性が結果列の記述と異なる場合は、オペランド列の値が結果列のデータ・タイプと属性に適合するように変換されます。この変換操作は、値を結果列に割り当てる場合とまったく同じです。例えば、

- 一方のオペランド列が CHAR(10) で、もう一方のオペランド列が CHAR(5) の場合は、結果列は CHAR(10) になり、CHAR(5) の列から得られた値の右側に 5 個のブランクが埋め込まれます。
- 数値の整数部分を保持できない場合は、エラーが戻されます。

数値オペランド

数値タイプは、他の数値データ・タイプ、文字ストリング・データ・タイプ、グラフィック・ストリング・データ・タイプと互換性があります。

一方のオペランド列	他方のオペランド	結果列のデータ・タイプ
SMALLINT	SMALLINT またはストリング	SMALLINT
INTEGER	SMALLINT、INTEGER、またはストリング	INTEGER
BIGINT	SMALLINT、INTEGER、BIGINT、またはストリング	BIGINT

一方のオペランド列	他方のオペランド	結果列のデータ・タイプ
DECIMAL(w,x)	SMALLINT	DECIMAL(p,x) (ただし、 p = min(mp, x+max(w-x,5)) mp = 31 または 63) (注 1 を参照)
DECIMAL(w,x)	INTEGER	DECIMAL(p,x) (ただし、 p = min(mp, x+max(w-x,11)) mp = 31 または 63) (注 1 を参照)
DECIMAL(w,x)	BIGINT	DECIMAL(p,x) (ただし、 p = min(mp, x+max(w-x,19)) mp = 31 または 63) (注 1 を参照)
DECIMAL(w,x)	DECIMAL(y,z) または NUMERIC(y,z,)	DECIMAL(p,s) (ただし、 p = min(mp, max(x,z)+max(w-x,y-z)) s = max(x,z) mp = 31 または 63) (注 1 を参照)
DECIMAL(w,x)	ストリング	DECIMAL(w,x)
NUMERIC(w,x)	SMALLINT	NUMERIC(p,x) (ただし、 p = min(mp, x + max(w-x,5)) mp = 31 または 63) (注 1 を参照)
NUMERIC(w,x)	INTEGER	NUMERIC(p,x) (ただし、 p = min(mp, x + max(w-x,11)) mp = 31 または 63) (注 1 を参照)
NUMERIC(w,x)	BIGINT	NUMERIC(p,x) (ただし、 p = min(mp, x + max(w-x,19)) mp = 31 または 63) (注 1 を参照)
NUMERIC(w,x)	NUMERIC(y,z)	NUMERIC(p,s) (ただし、 p = min(mp, max(x,z) + max(w-x, y-z)) s = max(x,z) mp = 31 または 63) (注 1 を参照)
NUMERIC(w,x)	ストリング	NUMERIC(w,x)
NONZERO SCALE BINARY	NONZERO SCALE BINARY	NONZERO SCALE BINARY (どちらかのオペランドが非ゼロの位取りの 2 進数である 場合、両方のオペランドとも同じ位取りの 2 進数でなければならない。)
REAL	REAL	REAL
REAL	SMALLINT、INTEGER、 BIGINT、DECIMAL、 NUMERIC、またはストリン グ	DOUBLE
DOUBLE	SMALLINT、INTEGER、 BIGINT、DECIMAL、 NUMERIC、REAL、 DOUBLE、またはストリン グ	DOUBLE
DECFLOAT(n)	REAL、DOUBLE、 INTEGER、または SMALLINT	DECFLOAT(n)

一方のオペランド列	他方のオペランド	結果列のデータ・タイプ
DECFLOAT(n)	DECIMAL(p<=16,s) または NUMERIC(p<=16,s)	DECFLOAT(n)
DECFLOAT(n)	BIGINT、DECIMAL(p>16,s)、または NUMERIC(p>16,s)	DECFLOAT(34)
DECFLOAT(n)	DECFLOAT(m)	DECFLOAT(max(n,m))
DECFLOAT(n)	ストリング	DECFLOAT(34)

注:

1. mp の値は、次のような場合に 63 になります。
   - w または y が 31 より大きい
   - CRTSQLxxx コマンド、RUNSQLSTM コマンド、または SET OPTION ステートメントの DECRESULT パラメーターの最大精度に値 63 が指定されている

   それ以外の場合、mp の値は 31 です。

文字ストリングとグラフィック・ストリングのオペランド

文字およびグラフィック・ストリングは、対応する CCSID 間で変換が決められている場合、他の文字およびグラフィック・ストリングと互換性があります。文字ストリングとグラフィック・ストリングに互換性があるのは、グラフィック・ストリング・データ・タイプのコード化スキームが Unicode で、文字ストリング・データ・タイプがビット・データではない場合です。

一方のオペランド列	他方のオペランド	結果列のデータ・タイプ
CHAR(x)	CHAR(y)	CHAR(z) (ただし、z = max(x,y))
GRAPHIC(x)	CHAR(y) または GRAPHIC(y)	GRAPHIC(z) (ただし、z = max(x,y))
VARCHAR(x)	CHAR(y) または VARCHAR(y)	VARCHAR(z) (ただし、z = max(x,y))
VARCHAR(x)	GRAPHIC(y)	VARGRAPHIC(z) (ただし、z = max(x,y))
VARGRAPHIC(x)	CHAR(y)、VARCHAR(y)、GRAPHIC(y)、または VARGRAPHIC(y)	VARGRAPHIC(z) (ただし、z = max(x,y))
CLOB(x)	CHAR(y)、VARCHAR(y)、または CLOB(y)	CLOB(z) (ただし、z = max(x,y))
CLOB(x)	GRAPHIC(y) または VARGRAPHIC(y)	DBCLOB(z) (ただし、z = max(x,y))
DBCLOB(x)	CHAR(y)、VARCHAR(y)、CLOB(y)、GRAPHIC(y)、VARGRAPHIC(y)、または DBCLOB(y)	DBCLOB(z) (ただし、z = max(x,y))

オペランドの 1 つの CCSID が 1208 の場合、結果の長さを決定するため、CCSID が 1208 ではないすべてのオペランドの長さ (x または y 値) は 2 倍にされます。

結果のグラフィック・ストリングの CCSID は、 139 ページの『ストリングを結合する演算に適用される変換規則』に基づいて取り込まれます。

2 進ストリングのオペランド

2 進ストリングは、他の 2 進ストリング、または文字ストリング FOR BIT DATA とのみ互換性があります。その他のデータ・タイプは、BINARY、VARBINARY、BLOB のいずれかのスカラー関数を使用してデータ・タイプを 2 進ストリングにキャストすることによって 2 進ストリング・データ・タイプとして扱うことができます。

一方のオペランド列	他方のオペランド	結果列のデータ・タイプ
BINARY(x)	BINARY(y) または CHAR(y) FOR BIT DATA	BINARY(z) (ただし、z = max(x,y))
VARBINARY(x)	BINARY(y)、VARBINARY(y)、CHAR(y) FOR BIT DATA、または VARCHAR(y) FOR BIT DATA	VARBINARY(z) (ただし、z = max(x,y))
VARCHAR(x) FOR BIT DATA	BINARY(y)	VARBINARY(z) (ただし、z = max(x,y))
BLOB(x)	BINARY(y)、VARBINARY(y)、BLOB(y)、CHAR(y) FOR BIT DATA、または VARCHAR(y) FOR BIT DATA	BLOB(z) (ただし、z = max(x,y))

日付/時刻のオペランド

DATE タイプは、別の DATE タイプ、または有効な日付のストリング表現を含む文字ストリング式または Unicode グラフィック・ストリング式と互換性があります。 結果のデータ・タイプは、DATE です。

TIME タイプは、別の TIME タイプ、または有効な時刻のストリング表現を含む文字ストリング式または Unicode グラフィック・ストリング式と互換性があります。結果のデータ・タイプは、TIME です。

TIMESTAMP タイプは、別の TIMESTAMP タイプ、 DATE タイプ、または、日付またはタイム・スタンプの有効なストリング表現を含む文字ストリング式または Unicode グラフィック・ストリング式と互換性があります。結果のデータ・タイプは、TIMESTAMP です。

一方のオペランド列	他方のオペランド	結果列のデータ・タイプ
DATE	DATE、 CHAR(y)、 VARCHAR(y)、 CLOB(y)、 GRAPHIC(y)、 VARGRAPHIC(y)、または DBCLOB(y)	DATE
TIME	TIME、 CHAR(y)、 VARCHAR(y)、 CLOB(y)、 GRAPHIC(y)、 VARGRAPHIC(y)、または DBCLOB(y)	TIME
TIMESTAMP(x)	TIMESTAMP(y)	TIMESTAMP(max(x,y))
TIMESTAMP(x)	DATE、 CHAR(y)、 VARCHAR(y)、 CLOB(y)、 GRAPHIC(y)、 VARGRAPHIC(y)、または DBCLOB(y)	TIMESTAMP(x)

データ・リンクのオペランド

データ・リンクは、別のデータ・リンクと互換性があります。ただし、NO LINK CONTROL を伴うデータ・リンクは、他の NO LINK CONTROL を伴うデータ・リンクとのみ互換性があります。FILE LINK CONTROL READ PERMISSION FS を伴うデータ・リンクは、他の FILE LINK CONTROL READ PERMISSION FS を伴うデータ・リンクとのみ互換性があります。FILE LINK CONTROL READ PERMISSION DB を伴うデータ・リンクは、他の FILE LINK CONTROL READ PERMISSION DB を伴うデータ・リンクとのみ互換性があります。結果のデータ・タイプは、DATALINK です。結果の DATALINK の長さは、すべてのデータ・タイプの中で、最も長いものです。

一方のオペランド列	他方のオペランド	結果列のデータ・タイプ
DATALINK(x)	DATALINK(y)	DATALINK(z) (ただし、z = max(x,y))

ROWID のオペランド

ROWID は、別の ROWID と互換性があります。結果のデータ・タイプは、ROWID です。

XML オペランド

XML データ・タイプは、他の XML データ・タイプとのみ互換性があります。結果のデータ・タイプは XML です。

結果の CCSID は、 101 ページの『XML 値』 で説明されている SQL_XML_DATA_CCSID QAQQINI オプション設定に由来する値です。

特殊タイプのオペランド

ユーザー定義特殊タイプは、同じユーザー定義特殊タイプとのみ互換性があります。結果のデータ・タイプは、そのユーザー定義特殊タイプです。

一方のオペランド列	他方のオペランド	結果列のデータ・タイプ
特殊タイプ	特殊タイプ	特殊タイプ

ストリングを結合する演算に適用される変換規則

ストリングを結合する演算には、連結、UNION、UNION ALL、EXCEPT、および INTERSECT があります。 (これらの規則は、MAX、MIN、VALUE、COALESCE、IFNULL、および CONCAT スカラー関数と CASE 式にも適用されます。) いずれの場合も、結果の CCSID はバインド時に決まり、演算を実行する際には、その CCSID によって識別されるコード化文字セットにストリングを変換する処理を伴うことがあります。

結果の CCSID は、オペランドの CCSID によって決まります。最初の 2 つのオペランドの CCSID によって中間結果の CCSID が決まり、その中間結果の CCSID と次のオペランドの CCSID によって新しい中間結果の CCSID が決まるというように、常にオペランドの CCSID の組み合わせによって結果の CCSID が決まります。結果のストリングまたは列の CCSID は、その 1 つ前の中間結果の CCSID と最後のオペランドの CCSID によって決まります。以下の規則を順番に適用することによって、CCSID の組み合わせごとに結果の CCSID を判別することができます。

- 両者の CCSID が同じ場合、結果の CCSID もそれと同じになります。
- どちらか一方の CCSID が 65535 である場合、結果の CCSID は 65535 になります。[30]
- 一方の CCSID が、他方の CCSID とは異なるコード化体系でデータを表している場合、結果は次の表によって決まります。

表 *23.* 中間結果のコード化体系の選択

	第 **2** オペランド			
第 **1** オペランド	**SBCS** データ	**DBCS** データ	混合データ	**Unicode** グラフィック・データ
SBCS データ	下記参照	第 2	第 2	第 2
DBCS データ	第 1	下記参照	第 2	第 2
混合データ	第 1	第 1	下記参照	第 2
Unicode グラフィック・データ	第 1	第 1	第 1	下記参照

- それ以外の場合、結果の CCSID は次の表によって決定されます。

表 *24.* 中間結果の *CCSID* の選択

	第 **2** オペランド				
第 **1** オペランド	列値	演算による値	定数	特殊レジスター	変数
列値	第 1	第 1	第 1	第 1	第 1
演算による値	第 2	第 1	第 1	第 1	第 1
定数	第 2	第 2	第 1	第 1	第 1

30. どちらかのオペランドが CLOB または DBCLOB の場合、結果の CCSID は、そのジョブのデフォルト CCSID になります。

表 24. 中間結果の *CCSID* の選択 (続き)

第 **1** オペランド	第 **2** オペランド				
	列値	演算による値	定数	特殊レジスター	変数
特殊レジスター	第 2	第 2	第 1	第 1	第 1
変数	第 2	第 2	第 2	第 2	第 1

外部コード化体系のデータを含む変数は、何らかの演算で使用される前に、固有のコード化体系に変換されます。上記の規則は、このような変換が既に行われていることを前提にしています。

中間結果は、演算による値のオペランドであると見なされることに注意してください。例えば、COLA、 COLB、および COLC の各列の CCSID が、それぞれ 37、278、および 500 であるとします。 COLA CONCAT COLB CONCAT COLC の結果の CCSID は、次のように決められます。

1. 最初に COLA CONCAT COLB の結果の CCSID が 37 であると判別されます。これは、両方のオペランドが列なので、第 1 オペランドの CCSID が選択されるからです。
2. 中間結果 CONCAT COLC の結果の CCSID が 500 になるのは、第 1 オペランドが演算によって得られた値であり、第 2 オペランドが列であるので、第 2 オペランドの CCSID が選択されるからです。

連結演算のオペランド、または CASE 式の結果式、または IN 述部のオペランド、または MAX、 MIN、VALUE、COALESCE、IFNULL、または CONCAT スカラー関数の選択された引数は、必要ならば、結果のストリングのコード化文字セットに変換されます。 UNION、UNION ALL、EXCEPT、または INTERSECT のオペランドの各ストリングは、必要に応じて、結果列のコード化文字セットに変換されます。文字変換が必要になるのは、以下の条件にすべて該当する場合だけです。

- 両者の CCSID が異なっている。
- どちらの CCSID も 65535 ではない。
- ストリングが NULL でなく、空でもない。
- 2 つの CCSID 間の変換が必要。詳しくは、 43 ページの『コード化文字セットと CCSID』を参照してください。

ストリングの文字が変換できない場合や、CCSID 間の変換が定義されていない場合は、エラーが戻されます。詳しくは、 43 ページの『コード化文字セットと CCSID』を参照してください。また、ストリングの文字が置換文字に変換された場合には、警告が出されます。

定数

定数 (リテラル ともいう) では ある 1 つの値を指定します。定数は、ストリング定数と数値定数に分類されます。ストリング定数は、さらに文字ストリング定数とグラフィック・ストリング定数に類別されます。数値定数は、さらに整数、浮動小数点数、および 10 進数に類別されます。

定数は、すべて NOT NULL の属性を持ちます。値がゼロの数値定数にある負符号は、無視されます。

整数定数

整数定数 は、小数点を除き最大 19 桁の符号付きまたは符号なしの整数を指定します。整数定数のデータ・タイプは、その値が長整数の範囲内であれば、長整数です。整数定数のデータ・タイプは、その値が長整数の範囲外であっても、64 ビット整数の範囲内であれば、64 ビット整数です。64 ビット整数値の範囲外で定義された定数は、10 進定数と見なされます。

例

```
64       -15      +100         32767     720176     12345678901
```

構文図では、符号を付けてはならない長整数の定数を指す場合に、整数 という用語を使用しています。

10 進定数

10 進定数 では、小数点を含むか、あるいは 2 進整数の範囲内にない、63 桁以下の符号付きまたは符号なしの数値として 10 進数を指定します。

精度は、数字の全桁数 (先行ゼロおよび後書きゼロも含む) であり、小数点の右側にある桁数 (後書きゼロも含む) が、位取りです。10 進定数の精度が最大 10 進精度よりも大きく、位取りが最大 10 進精度よりも大きくない場合、小数点の左側にある先行ゼロは、最大 10 進精度に合わせて除去されます。

例

```
25.5     1000.     -15.     +37589.3333333333
```

浮動小数点定数

浮動小数点定数 は、倍精度浮動小数点数を E で区切られた 2 つの数値として指定します。最初の数値には、符号および小数点を付けることができます。2 番目の数値には、符号を付けることはできますが、小数点を付けることはできません。この定数の値は、2 番目の数値によって指定された 10 の累乗に最初の数値を掛けた積です。この値は、浮動小数点数の範囲内でなければなりません。また、この定数内の文字数は、24 文字以下でなければなりません。先行ゼロを含めずに数えて、最初の数値の桁数は 17 桁以下、2 番目の数値の桁数は 3 桁以下でなければなりません。

例

```
15E1     2.E5     2.2E-1      +5.E+2
```

10 進浮動小数点定数

10 進浮動小数点定数 は、10 進浮動小数点数を E で区切られた 2 つの数値として指定します。最初の数値には、符号および小数点を付けることができます。2 番目の数値には、符号を付けることはできますが、小数点を付けることはできません。この定数の値は、2 番目の数値によって指定された 10 の累乗に最初の数値を掛けた積です。この値は、DECFLOAT(34) の範囲内でなければなりません。また、この定数内の文字数は、42 文字以下でなければなりません。先行ゼロを含めずに数えて、最初の数値の桁数は 34 桁以下、2 番目の数値の桁数は 4 桁以下でなければなりません。

E で区切られた 2 つの数値として指定された定数が 10 進浮動小数点定数であるのは、次の場合のみです。

- 先行ゼロを含めずに数えて、最初の数値の桁数が 17 (精度) を超える場合
- 指数が、倍精度浮動小数点数の範囲外 (-308 より小さいか、308 より大きい) である場合

数値定数に加えて、以下の予約済みキーワードを使用して、10 進浮動小数点特殊値を指定することもできます。これらの特殊値は INFINITY、NAN、および SNAN です。INFINITY は、無限大、つまり絶対値が無限に大きい数を表します。INFINITY の前に、オプションの符号を付けることができます。 INFINITY の代わりに INF を指定できます。NAN は、Not a Number (NaN) を表し、静止 NaN と呼ばれる場合があります。警告または例外を生じない、未定義結果を表す値です。SNAN はシグナル NaN (sNaN) を表します。任意の数値演算で定義される任意の演算で使用される場合、警告または例外を生じる、未定義結果を表す値です。NAN と SNAN はどちらも先頭にオプションの符号を追加できますが、その符号は、算術演算では意味を持ちません。SNAN は、警告または例外を出すことなく、非数値演算で使用できます。例えば、INSERT の VALUES リストで、または述部で比較される定数として使用できます。

特殊値 (INFINITY、INF、NAN、および SNAN) のいずれかが、列名などの識別子として解釈される可能性のあるコンテキストで使用される場合は、その特殊値を表すストリング定数を 10 進浮動小数点にキャストしてください。

```
CAST('snan' AS DECFLOAT(34))
CAST('INF' AS DECFLOAT(34))
CAST('Nan' AS DECFLOAT(34))
```

例

```
1.8E308    -1.23456789012345678E-2   SNAN     -INFINITY
```

文字ストリング定数

文字ストリング定数 は、可変長の文字ストリングを指定します。

文字ストリング定数には、次のように 2 つの形式があります。

- ストリング区切り文字で始まり、また終了する一連の文字。2 つのストリング区切り文字の間のバイト数は、32740 を超えてはなりません。2 つの連続するストリング区切り文字は、文字ストリング内で 1 つのストリング区切り文字を表す場合に使用します。ストリング内に含まれていない連続する 2 つのストリング区切り文字は、空のストリングを表します。

- X の後に、始めと終わりをストリング区切り文字で囲んだ一連の文字。ストリング区切り文字で囲まれている文字は、偶数桁数の 16 進数字でなければなりません。ストリング区切り文字の間のブランクは無視されます。16 進数字の桁数は、32762 以下でなければなりません。16 進数字とは、数字または A から F までの任意の文字 (大文字または小文字) です。16 進数の表記規則により、1 対の 16 進数字がそれぞれ 1 文字を表します。この形式のストリング定数を使うと、キーボード上にはない文字を指定することができます。

文字ストリング定数として、混合データを使用することができます。ジョブの CCSID が混合データをサポートする場合は、文字ストリング定数に DBCS サブストリングが含まれていれば、その文字ストリング定数は混合データとして扱われます。それ以外の場合はすべて、文字ストリング定数は SBCS データとして分類されます。

定数に割り当てられる CCSID は、SQL ステートメントが Unicode や外部エンコード・スキーム (ASCII など) でエンコードされない限り、定数が入っている SQL ステートメント・テキストの CCSID です。この場合、SQL ステートメント・テキストは Unicode または外部エンコード・スキームから現行サーバーのデフォルト CCSID に変換されます。定数に割り当てられた CCSID が、現行サーバーのデフォルト CCSID になります[31]。

以下の表に、SQL ステートメント・テキストの CCSID の定義を示します。

表 *25.* さまざまなインターフェースの *SQL* ステートメント・テキストの *CCSID*

SQL インターフェース	**SQL** ステートメント・テキストの **CCSID**。
組み込み SQL	• 静的ステートメントの場合、CRTSQLxxx コマンドのソース・ファイルの CCSID です。 • 動的ステートメントの場合、PREPARE ステートメントで指定された変数の CCSID。あるいは、PREPARE ステートメントでストリング定数が指定された場合には、CRTSQLxxx コマンドのソース・ファイルの CCSID。
RUNSQLSTM	指定されたソース・ファイルの CCSID。
STRSQL および RUNSQL	現行サーバーのデフォルト CCSID。
サーバー上の呼び出しレベル・インターフェース (CLI)	SQL_ATTR_UCS2、SQL_ATTR_UTF8、またはワイド API が指定された場合、Unicode。 それ以外の場合、現行サーバーのデフォルト CCSID。代わりに、SQL_ATTR_NON_HEXCCSID 環境変数を使用して DFTCCSID ジョブ属性を指定できます。
IBM i Access Family ODBC ドライバーを使用したクライアントの ODBC	UNICODESQL が指定された場合、Unicode。 それ以外の場合、現行サーバーのデフォルト CCSID。デフォルト CCSID が 65535 である場合、使用される CCSID は DFTCCSID ジョブ属性の値になります。

31. デフォルト CCSID が 65535 である場合、使用される CCSID は DFTCCSID ジョブ属性の値になります。

表 25. さまざまなインターフェースの *SQL* ステートメント・テキストの *CCSID* (続き)

SQL インターフェース	**SQL** ステートメント・テキストの **CCSID**。
IBM Developer Kit for Java を使用したサーバーの JDBC または SQLJ	Unicode
IBM Toolbox for Java を使用したクライアントの JDBC	Unicode
IBM i Access Family OLE DB Provider を使用したクライアントの OLE DB	Unicode
IBM i Access Family ODO .NET Provider を使用したクライアントの .ADO .NET	Unicode

文字ストリング定数は、割り当てや比較で日付/時刻の定数値を表すために使用します。詳しくは、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

例

```
'Peggy'    '14.12.1990'    '32'    'DON''T CHANGE'    ''    X'FFFF'
```

グラフィック・ストリング定数

グラフィック・ストリング定数には、2 つのタイプ (DBCS と Unicode グラフィック・ストリング定数) があります。

DBCS グラフィック・ストリング定数

グラフィック・ストリング定数 は、可変長のグラフィック・ストリングです。指定するストリングの長さは、16370 を超えることはできません。DBCS グラフィック・ストリング定数には、次の形式があります。

コンテキスト	グラフィック・ストリング定数	空ストリング	例
すべてのコンテキスト	G ' SO DBCS ストリング SI '	G ' SO SI ' G '' g ' SO SI ' g ''	G ' SO 元気 SI '
PL/I	SO / DBCS ストリング / G SI	SO / / G SI	SO / 元気 / G SI

RV3F000-1

正規形式では、SQL 区切り文字と G は、SBCS 文字です。SBCS の ' は、EBCDIC のアポストロフィ (X'7D') です。

PL/I 形式では、アポストロフィと G は DBCS 文字です。ストリング内で 1 つのストリング区切り文字を表す場合は、2 つの連続した DBCS ストリング区切り文字を使用します。この PL/I 形式は、PL/I プログラムに組み込まれている静的ステートメントの場合にのみ有効であることに注意してください。

16 進の DBCS グラフィック定数もサポートされます。16 進 DBCS グラフィック定数の形式は、次のとおりです。

GX'ssss'

この定数において、**ssss** は、0 から 32760 までの 16 進数字のストリングを表します。ストリング区切り文字に囲まれた文字の数は、4 の偶数倍でなければなりません。ストリング区切り文字間のブランクは無視されます。4 つの数字のグループのそれぞれが 1 つの DBCS グラフィック文字を表します。シフトインおよびシフトアウトに対応する 16 進数 ('0E'X および '0F'X) は、ストリングの中に含めません。

定数に割り当てられる CCSID は、そのソースが外部コード化体系 (ASCII など) でコード化されている場合を除き、ソースの CCSID に関連する DBCS CCSID です。この場合、定数に割り当てられる CCSID は、その定数を含む SQL ステートメントが準備される時点の現行サーバーのデフォルトの CCSID に関連する DBCS CCSID です。ソースの CCSID に関連する DBCS CCSID がない場合、CCSID は 65535 になります。

関連する DBCS CCSID に関する説明については、IBM i Information Center の「Use DBCS CCSIDs」トピックを参照してください。ソースの CCSID については、「文字ストリング定数」を参照してください。

Unicode グラフィック・ストリング定数

Unicode グラフィック・ストリング定数には、2 つのタイプ (N と UX) があります。Unicode グラフィック定数の形式は、次のとおりです。

N'ssss'

この定数では、**ssss** は 16370 文字のストリングです。これらの文字は、処理中にソース CCSID から Unicode CCSID に変換されます。

16 進 Unicode グラフィック定数の形式は、次のとおりです。

UX'ssss'

この定数において、**ssss** は、0 から 32760 までの 16 進数字のストリングを表します。ストリング区切り文字に囲まれた文字の数は、4 の偶数倍でなければなりません。ストリング区切り文字間のブランクは無視されます。4 つ以上の数字のグループのそれぞれが、1 つの Unicode グラフィック文字を表します。

Unicode グラフィック・ストリング定数の CCSID は 13488 (UCS-2) です。標準オプションが指定される場合、この CCSID は 1200 (UTF-16) です。標準オプションについては、 xi ページの『標準への準拠』を参照してください。

2 進ストリング定数

2 進ストリング定数 は、可変長の 2 進ストリングを指定します。

2 進ストリング定数には、次のように 2 つの形式があります。

- BX の後に、始めと終わりをストリング区切り文字で囲んだ一連の文字。ストリング区切り文字で囲まれている文字は、偶数桁数の 16 進数字でなければなりません。ストリング区切り文字の間のブランクは無視されます。16 進数字の桁数は、32740 以下でなければなりません。16 進数字とは、数字または A から F までの任意の文字 (大文字または小文字) です。
- X の後に、始めと終わりをストリング区切り文字で囲んだ一連の文字。ストリング区切り文字で囲まれている文字は、偶数桁数の 16 進数字でなければなりません。ストリング区切り文字の間のブランクは無視されます。16 進数字の桁数は、32740 以下でなければなりません。16 進数字とは、数字または A から F までの任意の文字 (大文字または小文字) です。

定数に割り当てられる CCSID は 65535 です。

2 番目の形式の 2 進ストリング定数の構文は、文字定数の 2 番目の形式に等しいことに注意してください。この形式の定数が 2 進ストリング定数として扱われるのは、標準オプションが指定される場合のみです。標準オプションについては、 xi ページの『標準への準拠』を参照してください。

例

```
BX'FFFF'
X'FFFF'
```

日付/時刻定数

日付/時刻定数 は、日付、時刻、またはタイム・スタンプを指定します。

通常、文字ストリング定数は、割り当てや比較で日付/時刻の定数値を表すために使用します。日時値のストリング表記については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。ただし、ANSI/ISO SQL 標準形式の日時定数を使用して、その定数がストリング定数ではなく日時定数 であることを明示することもできます。

ANSI/ISO SQL 標準の日時定数の 3 つの形式を以下に示します。

- DATE 'yyyy-mm-dd'

  値のデータ・タイプは DATE です。
- TIME 'hh:mm:ss'

  値のデータ・タイプは TIME です。
- TIMESTAMP 'yyyy-mm-dd hh:mm:ss.nnnnnnnnnnnn'

  値のデータ・タイプは TIMESTAMP(*p*) です。*p* は小数秒の桁数です。

  端数秒の部分の後続ゼロは、切り捨てたり、全部を除去したりすることができます。

標準日時定数のどの部分でも、先行ゼロは省略しないでください。

例

```
DATE '2003-09-03'
```

小数点

デフォルトの小数点 を指定できます。

デフォルトの小数点 は、以下の目的のために指定できます。

- 数値定数を解釈するため。
- 文字ストリングを数値にキャストするときに使用する小数点文字を決定するため (例えば、DECFLOAT、DECIMAL、 DOUBLE_PRECISION、FLOAT、および REAL スカラー関数および CAST 指定で使用される小数点)。
- 数値をストリングにキャストするときに結果の中で使用する小数点文字を決定するため (例えば、CHAR、VARCHAR、CLOB、GRAPHIC、VARGRAPHIC の各スカラー関数や CAST 指定で使用する小数点)

デフォルトの小数点は、以下のインターフェースを使用して指定できます。

表 *26.* デフォルト小数点インターフェース

SQL インターフェース	指定
組み込み SQL	SQL プログラム作成 (CRTSQLxxx) コマンドで、OPTION パラメーターに *JOB、 *PERIOD、*COMMA、または *SYSVAL のいずれかの値を指定します。組み込み SQL を含むプログラムのソースの中で DECMPT パラメーターを指定するには、SET OPTION ステートメントも使用できます。 (CRTSQLxxx コマンドについて詳しくは、「組み込み SQL プログラミング」を参照してください。)
対話式 SQL および SQL 実行ステートメント	SQL 開始 (STRSQL) コマンドで DECPNT パラメーターを指定するか、セッション属性を変更します。あるいは、SQL 実行 (RUNSQLSTM) ステートメントで DECMPT パラメーターを使用します。 (STRSQL コマンドと RUNSQLSTM コマンドについて詳しくは、「SQL プログラミング」を参照。)
サーバー上の呼び出しレベル・インターフェース (CLI)	SQL_ATTR_DECIMAL_SEP 環境変数または接続変数 (CLI について詳しくは、「SQL 呼び出しレベル・インターフェース (ODBC)」を参照してください。)
IBM IBM Developer Kit for Java を使用したサーバーの JDBC または SQLJ	「小数点 (Decimal Separator)」接続プロパティー。 (JDBC および SQLJ について詳しくは、「IBM Developer Kit for Java」を参照してください。)
IBM i Access Family ODBC ドライバーを使用したクライアントの ODBC	ODBC セットアップでの「アドバンスト・サーバー・オプション (Advanced Server Options)」の中の「小数点 (Decimal Separator)」。 (ODBC の詳細については、「IBM i Access」を参照。)
IBM Toolbox for Java を使用したクライアントの JDBC	JDBC セットアップの中の「形式 (Format)」。 (ODBC の詳細については、「IBM i Access」を参照。) (IBM Toolbox for Java について詳しくは、「IBM Toolbox for Java」を参照してください。)

小数点がコンマの場合は、以下の規則が適用されます。

- ピリオドは、小数点としても使用できます。
- 数値定数の区切り記号として使用するコンマの後にはスペースを 1 つ付けなければなりません。
- 小数点として使用するコンマの後にスペースを付けてはなりません。

したがって、小数部を持たない 10 進定数を指定するときには、定数の最後に付くコンマの後に、ブランク以外の文字を入れておく必要があります。 このブランク以外の文字には、次のように、区切り記号のコンマを使用することができます。

```
VALUES(9999999999,, 111)
```

区切り文字

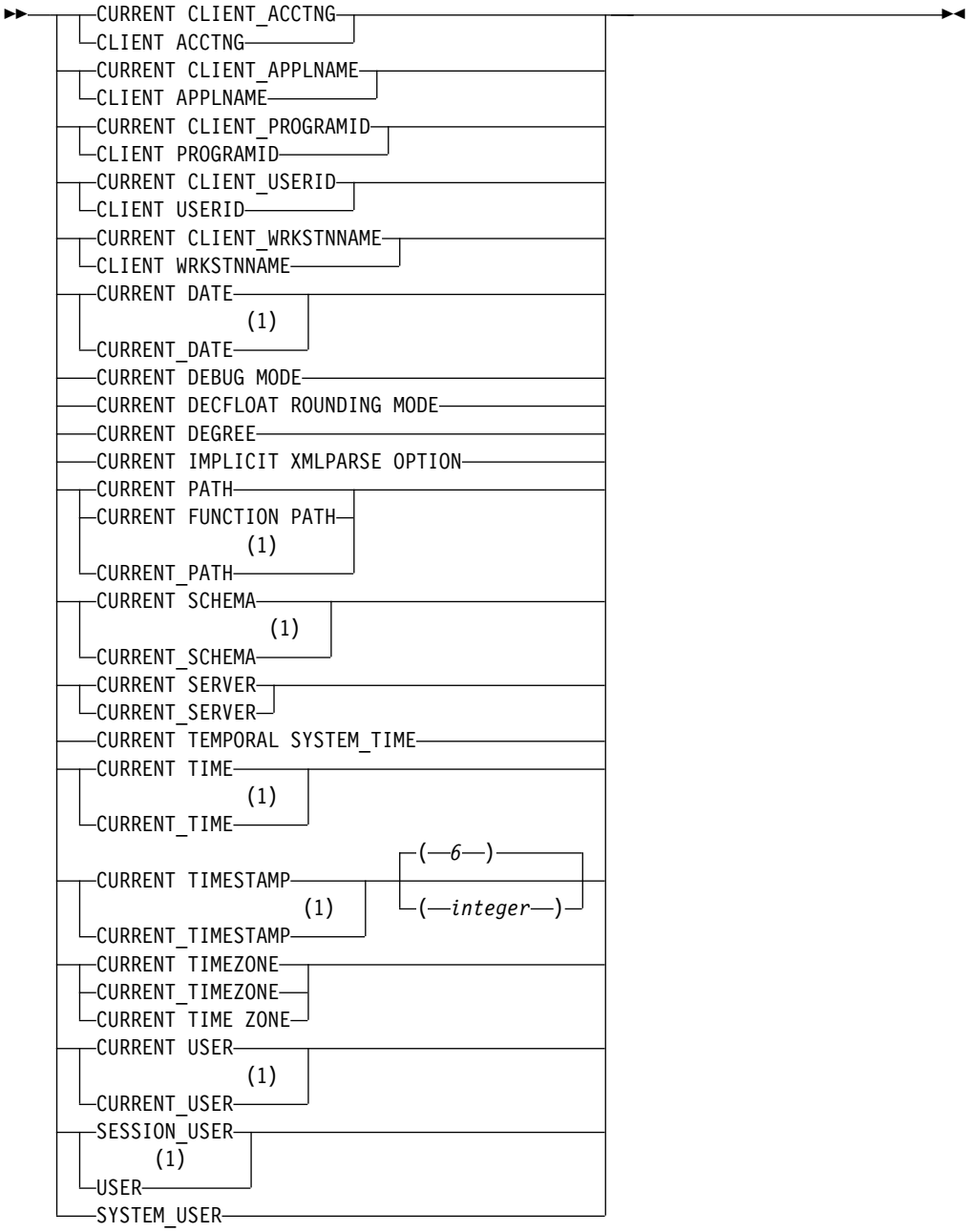
*APOST および *QUOTE は、COBOL ステートメント内でストリング区切り文字を指定する COBOL プリコンパイラー・オプションですが、両方を同時に使用することはできません。*APOST は、アポストロフィ (') をストリング区切り文字として指定し、*QUOTE は、引用符 (") を指定します。*APOST および *QUOTE は、COBOL プログラムに組み込まれた SQL ステートメントに対して同様の役割を果たすプリコンパイラー・オプションですが、両方を同時に使用することはできません。*APOST は、アポストロフィ (') を SQL ストリング区切り文字として指定します。このオプションを使用すると、引用符 (") が SQL エスケープ文字になります。*QUOTE は、引用符を SQL ストリング区切り文字として指定します。このオプションを使用すると、アポストロフィが SQL エスケープ文字になります。
*APOST および *QUOTE の値は、それぞれ *APOST および *QUOTE の値と同じです。

COBOL 以外のホスト言語では、ストリング区切り文字の用法が固定されています。すなわち、ホスト言語および静的 SQL ステートメントのストリング区切り文字にはアポストロフィ (') が使用され、SQL エスケープ文字には引用符 (") が使用されます。

特殊レジスター

特殊レジスター は、データベース・マネージャーによって定義されるアプリケーション・プロセスのための域であり、そこに保管される情報は、SQL ステートメントで参照することができます。特殊レジスターに対する参照は、現行サーバーによって与えられた値に対する参照となります。参照する値がストリングの場合は、そのCCSID は、現行サーバーのデフォルトの CCSID となります。

特殊レジスターは、次のようにも参照できます。



注:

1 SQL 2003 Core 標準では、下線付きの書式が使用されます。

これらの特殊レジスターの値は、CURRENT TEMPORAL SYSTEM_TIME を除いて、NULL にすることはできません。

CURRENT CLIENT_ACCTNG

CURRENT CLIENT_ACCTNG 特殊レジスターでは、現行接続で指定されているクライアント情報からアカウンティング・ストリングの値を取り込む VARCHAR(255) 値を指定します。

このレジスターのデフォルト値は空ストリングです。アカウンティング・ストリングの値は、これらのインターフェースを使用して変更できます。

- Set Client Information (SQLESETI) API は、クライアント特殊レジスターを変更できます。
- SYSPROC.WLM_SET_CLIENT_INFO プロシージャーは、クライアント特殊レジスターを変更できます。
- CLI では、SQLSetConnectAttr() を使用して SQL_ATTR_INFO_ACCTSTR 接続属性を設定できます。
- ODBC では、SQLSetConnectAttr() を使用して ODBC_ATTR_INFO_ACCTSTR 接続属性を設定できます。
- JDBC では、setClientInfo 接続方式を使用して、ClientAccounting 接続プロパティーを設定できます。

例

この接続のアカウンティング・ストリングの現行値を入手します。

```
VALUES CURRENT CLIENT_ACCTNG
  INTO :ACCT_STRING
```

CURRENT CLIENT_APPLNAME

CURRENT CLIENT_APPLNAME 特殊レジスターでは、現行接続で指定されているクライアント情報からアプリケーション名の値を取り込む VARCHAR(255) 値を指定します。

このレジスターのデフォルト値は空ストリングです。アプリケーション名の値は、これらのインターフェースを使用して変更できます。

- Set Client Information (SQLESETI) API は、クライアント特殊レジスターを変更できます。
- SYSPROC.WLM_SET_CLIENT_INFO プロシージャーは、クライアント特殊レジスターを変更できます。
- CLI では、SQLSetConnectAttr() を使用して SQL_ATTR_INFO_APPLNAME 接続属性を設定できます。
- ODBC では、SQLSetConnectAttr() を使用して ODBC_ATTR_INFO_APPLNAME 接続属性を設定できます。
- JDBC では、setClientInfo 接続方式を使用して、ApplicationName 接続プロパティーを設定できます。

例

この接続に使用されているアプリケーションを使用できる部門を選択します。

```
SELECT  DEPT
FROM DEPT_APPL_MAP
WHERE APPL_NAME = CURRENT CLIENT_APPLNAME
```

CURRENT CLIENT_PROGRAMID

CURRENT CLIENT_PROGRAMID 特殊レジスターでは、現行接続で指定されているクライアント情報からクライアント・プログラム ID の値を取り込む VARCHAR(255) 値を指定します。

このレジスターのデフォルト値は空ストリングです。クライアント・プログラム ID の値は、これらのインターフェースを使用して変更できます。

- Set Client Information (SQLESETI) API は、クライアント特殊レジスターを変更できます。
- SYSPROC.WLM_SET_CLIENT_INFO プロシージャーは、クライアント特殊レジスターを変更できます。
- CLI では、SQLSetConnectAttr() を使用して SQL_ATTR_INFO_PROGRAMID 接続属性を設定できます。
- ODBC では、SQLSetConnectAttr() を使用して ODBC_ATTR_INFO_PROGRAMID 接続属性を設定できます。
- JDBC では、setClientInfo 接続方式を使用して、ClientProgramID 接続プロパティーを設定できます。

例

この接続に使用されるプログラム ID を入手します。

```
VALUES CURRENT CLIENT_PROGRAMID
INTO :PGM_ID
```

CURRENT CLIENT_USERID

CURRENT CLIENT_USERID 特殊レジスターでは、現行接続で指定されているクライアント情報からクライアント・ユーザー ID の値を取り込む VARCHAR(255) 値を指定します。

このレジスターのデフォルト値は空ストリングです。クライアント・ユーザー ID の値は、これらのインターフェースを使用して変更できます。

- Set Client Information (SQLESETI) API は、クライアント特殊レジスターを変更できます。
- SYSPROC.WLM_SET_CLIENT_INFO プロシージャーは、クライアント特殊レジスターを変更できます。
- CLI では、SQLSetConnectAttr() を使用して SQL_ATTR_INFO_USERID 接続属性を設定できます。
- ODBC では、SQLSetConnectAttr() を使用して ODBC_ATTR_INFO_USERID 接続属性を設定できます。

- JDBC では、setClientInfo 接続方式を使用して、ClientUser 接続プロパティーを設定できます。

例

現行クライアント・ユーザー ID の所属する部門を検索します。

```
SELECT  DEPT
FROM DEPT_USERID_MAP
WHERE USER_ID = CURRENT CLIENT_USERID
```

CURRENT CLIENT_WRKSTNNAME

CURRENT CLIENT_WRKSTNNAME 特殊レジスターでは、現行接続で指定されているクライアント情報からワークステーション名の値を取り込む VARCHAR(255) 値を指定します。

このレジスターのデフォルト値は空ストリングです。ワークステーション名の値は、これらのインターフェースを使用して変更できます。

- Set Client Information (SQLESETI) API は、クライアント特殊レジスターを変更できます。
- SYSPROC.WLM_SET_CLIENT_INFO プロシージャーは、クライアント特殊レジスターを変更できます。
- CLI では、SQLSetConnectAttr() を使用して SQL_ATTR_INFO_WRKSTNNAME 接続属性を設定できます。
- ODBC では、SQLSetConnectAttr() を使用して ODBC_ATTR_INFO_WRKSTNNAME 接続属性を設定できます。
- JDBC では、setClientInfo 接続方式を使用して、ClientHostName 接続プロパティーを設定できます。

例

この接続に使用されているワークステーション名を入手します。

```
VALUES CURRENT CLIENT_WRKSTNNAME
INTO :WS_NAME
```

CURRENT DATE

CURRENT DATE (現在の日付) 特殊レジスターは、現行サーバーで SQL ステートメントが実行される時点の時刻機構の読み取りに基づく日付を指定します。

この特殊レジスターが 1 つの SQL ステートメント内で複数回使用される場合、または 1 つのステートメント内で CURRENT TIME、CURRENT TIMESTAMP とともに、または CURDATE、CURTIME、または NOW スカラー関数とともに使用される場合は、値はすべて 1 回の刻時機構読み取りに基づきます。[32]

32. CURRENT_DATE の同義語として LOCALDATE を指定できます。

例

以下のステートメントは、表 PROJECT を使用して、MA2111 プロジェクト (PROJNO) のプロジェクト終了日付 (PRENDATE) に CURRENT DATE をセットしています。

```
UPDATE PROJECT
  SET PRENDATE = CURRENT DATE
  WHERE PROJNO = 'MA2111'
```

CURRENT DEBUG MODE

CURRENT DEBUG MODE 特殊レジスターは、Unified Debugger でデバッグできるよう、SQL または Java プロシージャーを作成または変更するかどうかを指定します。

DEBUG MODE の明示的な指定、または CREATE PROCEDURE か ALTER PROCEDURE ステートメントにある SET OPTION ステートメント中の DBGVIEW オプションは、CURRENT DEBUG MODE 特殊レジスターの値をオーバーライドします。 CURRENT DEBUG MODE は、静的および動的 SQL ステートメントに影響します。このレジスターのデータ・タイプは VARCHAR(8) です。有効な値は以下のとおりです。

DISALLOW
Unified Debugger がデバッグできないようにプロシージャーが作成されます。プロシージャーの DEBUG MODE 属性が DISALLOW である場合、後にそのプロシージャーを変更して DEBUG MODE 属性を変更することができます。

ALLOW
Unified Debugger がデバッグできるようにプロシージャーが作成されます。プロシージャーの DEBUG MODE 属性が ALLOW である場合、後にそのプロシージャーを変更して DEBUG MODE 属性を変更することができます。

DISABLE
Unified Debugger がデバッグできないようにプロシージャーが作成されます。プロシージャーの DEBUG MODE 属性が DISABLE である場合、後にそのプロシージャーを変更して DEBUG MODE 属性を変更することはできません。

この値は SET CURRENT DEBUG MODE ステートメントを呼び出して変更することができます。このステートメントの詳細については、 1677 ページの『SET CURRENT DEBUG MODE』を参照してください。

CURRENT DEBUG MODE の初期値は DISALLOW です。

例

以下のステートメントは、SQL または Java プロシージャーの以降の作成または変更がデバッグ可能にならないようにします。

```
SET CURRENT DEBUG MODE = DISALLOW
```

CURRENT DECFLOAT ROUNDING MODE

CURRENT DECFLOAT ROUNDING MODE 特殊レジスターでは、動的に準備する SQL ステートメントで DECFLOAT 値を操作するときに使用する丸め方式を指定します。

このレジスターのデータ・タイプは VARCHAR(128) です。サポートされる丸めモードは次のとおりです。

ROUND_CEILING
正の無限大の方向に丸めます。廃棄される桁がすべてゼロであるか、符号が負である場合、廃棄される桁が除去されることを除いて結果は変わりません。それ以外の場合、結果の係数は 1 だけ増分されます (切り上げられます)。

ROUND_DOWN
ゼロの方向に丸めます (切り捨て)。廃棄される桁は無視されます。

ROUND_FLOOR
負の無限大の方向に丸めます。廃棄される桁がすべてゼロであるか、符号が正である場合、廃棄される桁が除去されることを除いて結果は変わりません。それ以外の場合は、符号が負になり、結果の係数は 1 増分されます。

ROUND_HALF_DOWN
最も近い値に丸めます。等距離である場合、切り捨てます。廃棄される桁が左隣の桁の値の半分 (0.5) より大きい値を表す場合、結果の係数は 1 増分されます (切り上げ)。そうでない場合、廃棄される桁は無視されます。

ROUND_HALF_EVEN
最も近い値に丸めます。等距離である場合、最後の桁が偶数になるように丸めます。廃棄される桁が左隣の桁の値の半分 (0.5) より大きい値を表す場合、結果の係数は 1 増分されます (切り上げ)。半分未満の場合、結果の係数は調整されません (つまり、廃棄される桁は無視されます)。それ以外の場合 (ちょうど半分である場合)、右端の数字が偶数である場合は結果の係数は変更されません。右端の数字が奇数である場合は、(偶数にするために) 1 増分されます (切り上げられます)。

ROUND_HALF_UP
最も近い値に丸めます。等距離である場合、切り上げます。廃棄される桁が左隣の桁の値の半分 (0.5) 以上の値を表す場合、結果の係数は 1 増分されます (切り上げられます)。そうでない場合、廃棄される桁は無視されます。

ROUND_UP
ゼロから離れる方向に丸めます。廃棄される桁がすべてゼロである場合、廃棄される桁が除去されることを除いて結果は変わりません。それ以外の場合、結果の係数は 1 だけ増分されます (切り上げられます)。

活動化グループにおける CURRENT DECFLOAT ROUNDING MODE の初期値は、活動化グループで実行される最初の SQL ステートメントによって設定されます。

- 活動化グループにおける最初の SQL ステートメントが、SQL プログラムまたは SQL パッケージから実行される場合、CURRENT DECFLOAT ROUNDING MODE 特殊レジスターは DECFLTRND パラメーターの値に設定されます。
- それ以外の場合、初期値は ROUND_HALF_EVEN です。

CRTSQLxxx コマンドの DECFLTRND パラメーターまたは SET OPTION は、静的 SQL ステートメントに使用されます。

例

ホスト変数 APPL_ROUND (VARCHAR(128)) を現在の丸め方式に設定します。

```
SELECT CURRENT DECFLOAT ROUNDING MODE
  INTO :APPL_ROUND
  FROM SYSIBM.SYSDUMMY1
```

CURRENT DEGREE

CURRENT DEGREE 特殊レジスターは、照会、索引作成、索引再ビルド、索引の保守、および再編成実行のための I/O または Symmetric MultiProcessing (SMP) 並列処理の度合いを指定します。

CURRENT DEGREE は、静的および動的 SQL ステートメントに影響します。このレジスターのデータ・タイプは CHAR(5) です。有効な値は以下のとおりです。

1 並列処理は許可されません。

2 から **32767**
使用する並列処理の度合いを指定します。

ANY
I/O か SMP のいずれの並列処理の場合でも、データベース・マネージャーが任意の数のタスクを選択できることを指定します。

並列処理の使用および使用されるタスク数は、システムで使用可能なプロセッサーの数、ジョブが実行されているプール内の使用可能なアクティブ・メモリーに関するジョブの割り当て分、および操作に予想される経過時間が CPU 処理または I/O リソースによって限定されるかどうかに基づいて決定されます。データベース・マネージャーは、プール内のメモリーのこのジョブが占める割合に基づいて、経過時間を最小化するインプリメンテーションを選択します。

NONE
並列処理は許可されません。

MAX
I/O か SMP のいずれの並列処理の場合でも、データベース・マネージャーは任意の数のタスクを選択できます。MAX は、プール内のすべてのアクティブ・メモリーを使用できることをデータベース・マネージャーが前提とする点を除いて、ANY と類似しています。

IO データベース・マネージャーが照会に I/O 並列処理の使用を選択した場合、任意の数のタスクを使用できます。SMP は許可されません。

CURRENT DEGREE の初期値は、CHGQRYA CL コマンドによる現行の度合い、現行の照会オプション・ファイル (QAQQINI) 内の PARALLEL_DEGREE パラメーター、または QQRYDEGREE システム値によって決定されます。

この値は SET CURRENT DEGREE ステートメントを呼び出して変更することができます。このステートメントの詳細については、1682 ページの『SET CURRENT DEGREE』を参照してください。

例

次のステートメントは、並列処理を禁止します。

```
SET CURRENT DEGREE  = '1'
```

CURRENT IMPLICIT XMLPARSE OPTION

CURRENT IMPLICIT XMLPARSE OPTION 特殊レジスターでは、直列化した XML データを Db2 サーバーによって検証なしで暗黙的に解析するときに使用する空白文字処理オプションを指定します。

検証なしの暗黙的な解析操作が発生するのは、SQL ステートメントで XML ホスト変数を処理する場合や、XMLVALIDATE 関数の引数ではない暗黙的タイプ付きまたは明示的タイプ付きの XML パラメーター・マーカーを処理する場合です。

このレジスターのデータ・タイプは VARCHAR(128) です。サポートされる値は以下のとおりです。

STRIP WHITESPACE
文書を読みやすくするための空白文字を削除します。空白文字に相当するのは、ブランク、復帰、改行、タブです。境界としての役割を果たしている空白文字 (エレメント間の空白文字) をすべて削除します。

PRESERVE WHITESPACE
空白文字を削除しません。

CURRENT IMPLICIT XMLPARSE OPTION の初期値は、STRIP WHITESPACE です。

例

CURRENT IMPLICIT XMLPARSE OPTION を PRESERVE WHITESPACE に設定します。

```
SET CURRENT IMPLICIT XMLPARSE OPTION = PRESERVE WHITESPACE
```

CURRENT PATH

CURRENT PATH 特殊レジスターは、動的に準備された SQL ステートメントにおける非修飾のタイプ名、関数名、およびプロシージャー名を解決するために使用する SQL パスを指定します。

また、これは、SQL CALL ステートメントの変数 (CALL 変数) として指定された非修飾のプロシージャー名を解決するためにも使用されます。データ・タイプは VARCHAR(8843) です。

CURRENT PATH 特殊レジスターには、SQL パスの値 (1 つ以上のスキーマ名のリスト) を組み込みます。各スキーマ名を区切り文字で囲み、後のスキーマから分離するためにコンマを使用します (ストリング内の区切り文字は、すべての区切り文字付き ID 内の場合と同じ要領で繰り返します)。区切り文字とコンマは、特殊レジスターの長さに含まれています。パス内のスキーマ名の最大数は 268 です。

スキーマ名に別のシステム名がある場合に、Set Path ステートメントでそのシステム名を明示的に指定したとしても、CURRENT PATH 特殊レジスターではスキーマ名が返されます。

動的および静的の両方の SQL ステートメントでの非修飾名を解決するために SQL パスを使用する時点、およびその値の効果についての説明は、 74 ページの『非修飾の関数、プロシージャー、特定名、タイプ、および変数』を参照してください。

活動化グループにおける CURRENT PATH 特殊レジスターの初期値は、実行される最初の SQL ステートメントによって設定されます。

- 活動化グループにおける最初の SQL ステートメントが、SQL プログラムまたは SQL パッケージから実行され、SQLPATH パラメーターが CRTSQLxxx コマンドで指定されている場合には、そのパスは SQLPATH パラメーターで指定された値になります。また、SQLPATH 値は、SET OPTION ステートメントを使用しても指定することができます。
- その他の場合は、
  - SQL 命名規則の場合、"QSYS"、"QSYS2"、"SYSPROC"、"SYSIBMADM"、"ステートメントの実行時権限 *ID* の値"。
  - システム命名規則の場合、"*LIBL"。

特殊レジスターの値は、SET PATH ステートメントの実行によって変更できます。このステートメントの詳細については、 1720 ページの『SET PATH』を参照してください。 プラットフォーム間の移植性を確保するために、アプリケーションの先頭で SET PATH ステートメントを実行することをお勧めします。

例

スキーマ QSYS および QSYS2 (SYSTEM PATH) の前に、スキーマ SMITH を検索するように特殊レジスターを設定します。

```
SET CURRENT PATH SMITH, SYSTEM PATH
```

CURRENT SCHEMA

CURRENT SCHEMA 特殊レジスターは、動的に準備された SQL ステートメントの中で、該当する無修飾のデータベース・オブジェクト参照の修飾に使用されるスキーマ名を識別する VARCHAR(128) 値を指定します。

CURRENT SCHEMA は、DYNDFTCOL が指定されているプログラムの中の名前を修飾するためには使用されません。プログラム内で DYNDFTCOL が指定されている場合は、そのスキーマ名が CURRENT SCHEMA のスキーマ名の代わりに使用されます。[33]

CURRENT SCHEMA の初期値は、現行セッション・ユーザーの権限 ID です。

特殊レジスターの値は、SET SCHEMA ステートメントを実行して変更できます。詳しくは、 1727 ページの『SET SCHEMA』を参照してください。

33. Db2 for z/OS との互換性を維持するために、特殊レジスター CURRENT SQLID は CURRENT SCHEMA の同義語として扱われます。

静的 SQL ステートメントの場合は、無修飾のデータベース・オブジェクト参照を修飾するために使用するスキーマ名は、DFTRDBCOL キーワードにより制御されます。

例

オブジェクト修飾用のスキーマを 'D123' に設定します。

```
SET CURRENT SCHEMA = 'D123'
```

CURRENT SERVER

CURRENT SERVER 特殊レジスターは、現行のアプリケーション・サーバーを識別する VARCHAR(18) 値を指定します。

CURRENT SERVER は、CONNECT (タイプ 1)、CONNECT (タイプ 2)、または SET CONNECTION ステートメントによって変更できますが、それができるのは特定の条件のもとだけに限られます。 1053 ページの『CONNECT (タイプ 1)』、1059 ページの『CONNECT (タイプ 2)』、および 1673 ページの『SET CONNECTION』の説明を参照してください。

CURRENT SERVER を指定できるようにするには、ADDRDBDIRE コマンドまたは WRKRDBDIRE コマンドを使用してリレーショナル・データベース・ディレクトリーに項目を追加することによって、ローカル・リレーショナル・データベースに名前を付けなければなりません。

例

ホスト変数の APPL_SERVE (VARCHAR(18)) を現行サーバーの名前にセットします。

```
SELECT CURRENT SERVER
  INTO :APPL_SERVE
  FROM SYSIBM.SYSDUMMY1
```

CURRENT TEMPORAL SYSTEM_TIME

CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターは、システム期間テンポラル表への参照のためのデフォルトの SYSTEM_TIME 期間指定で使用される TIMESTAMP(12) 値を指定します。

システム期間テンポラル表が参照され、CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターに有効な値が NULL 値でない場合、以下の期間が暗黙的に指定されます。

```
FOR SYSTEM_TIME AS OF CURRENT TEMPORAL SYSTEM_TIME
```

ユーザー定義の関数、プロシージャー、またはトリガー内の特殊レジスターの初期値は、呼び出し側アプリケーションから継承されます。その他のコンテキストでは、特殊レジスターの初期値は NULL 値です。

SET CURRENT TEMPORAL SYSTEM_TIME ステートメントを実行することにより、この特殊レジスターの値を変更できます。ルーチン内で特殊レジスターの値が変更されると、その新規の値は呼び出し側アプリケーションに戻されません。

SYSTIME オプションの設定は、静的 SQL ステートメントと動的 SQL ステートメントでのシステム期間テンポラル表への参照が CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターの値の影響を受けるかどうかを決定します。オプションは、YES または NO に設定できます。

CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターの値が NULL 値ではなく、かつ、 SYSTIME オプションが YES に設定されている場合は、select-statement に FOR SYSTEM_TIME を明示的に指定できません。

例

以下の例では、表 IN_TRAY がシステム期間テンポラル表であると想定します。

例 **1**: CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターによって指定された日付現在の、IN_TRAY にあるメッセージの状態に基づき、ユーザー ID とサブジェクト行をリストします。

```
SELECT SOURCE, SUBJECT
 FROM IN_TRAY
```

特殊レジスターが NULL 以外の値に設定されている場合、上記のステートメントは、以下のステートメントと同等です。

```
SELECT SOURCE, SUBJECT
    FROM IN_TRAY
    FOR SYSTEM_TIME AS OF CURRENT TEMPORAL SYSTEM_TIME
```

例 **2**: 値 '2011-01-01-00.00.00.000000' に設定された CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターによって指定された値より前に送信された、IN_TRAY 内のメッセージのユーザー ID とサブジェクト行をリストします。

```
SELECT SOURCE, SUBJECT
    FROM IN_TRAY
    WHERE RECEIVED < CURRENT TEMPORAL SYSTEM_TIME
```

CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターが NULL 値に設定されている場合、次のステートメントは同じ結果を戻します。

```
SELECT SOURCE, SUBJECT
    FROM IN_TRAY
    FOR SYSTEM_TIME AS OF '2011-01-01-00.00.00.000000'
    WHERE DATE(RECEIVED) < DATE('2011-01-01-00.00.00.000000')
```

CURRENT TIME

CURRENT TIME (現在の時刻) 特殊レジスターは、SQL ステートメントが現行サーバーで実行される時点での刻時機構の読み取りに基づく時刻を指定します。

この特殊レジスターが 1 つの SQL ステートメント内で複数回使用される場合、または 1 つのステートメント内で CURRENT DATE、CURRENT TIMESTAMP、または CURDATE、CURTIME、または NOW スカラー関数とともに使用される場合は、値はすべて 1 回の刻時機構読み取りに基づきます。[34]

34. LOCALTIME および LOCALTIME(0) は CURRENT_TIME の同義語として指定できます。

例

表 CL_SCHED を使用して、当日後刻に開始される (STARTING) すべてのクラス (CLASS_CODE) を選択します。当日のクラスは、DAY 列に 3 の値を持っています。

```
SELECT CLASS_CODE FROM CL_SCHED
  WHERE STARTING > CURRENT TIME  AND DAY = 3
```

CURRENT TIMESTAMP

CURRENT TIMESTAMP (現在のタイム・スタンプ) 特殊レジスターは、SQL ステートメントが現行サーバーで実行される刻時機構の読み取りに基づくタイム・スタンプを指定します。

この特殊レジスターが 1 つの SQL ステートメント内で複数回使用される場合、または 1 つのステートメント内で CURRENT DATE、CURRENT TIME、または CURDATE、CURTIME、または NOW スカラー関数とともに使用される場合は、値はすべて 1 回の刻時機構読み取りに基づきます。[35]

指定する精度のタイム・スタンプが必要な場合は、特殊レジスターを CURRENT TIMESTAMP(整数) で参照することができます (整数 は 0 から 12 までの範囲の整数)。デフォルトの精度は 6 です。

例

以下のステートメントでは、サンプル表 IN_TRAY に行を挿入しています。列 RECEIVED には、行が挿入された日時を示すタイム・スタンプの値が入ります。その他の 3 つの列には、ホスト変数 SRC(CHAR(8))、SUB(CHAR(64))、および TXT (VARCHAR(200)) の値が入ります。

```
INSERT INTO IN_TRAY
  VALUES (CURRENT TIMESTAMP, :SRC, :SUB, :TXT)
```

CURRENT USER

CURRENT USER 特殊レジスターは、ステートメント権限に使用される 1 次権限 ID を指定します。この特殊レジスターのデータ・タイプは VARCHAR(128) です。

1 つのスレッド内で複数の権限 ID が借用された場合、スレッド内で最後に借用された権限 ID の値が戻されます。

例

現行のステートメント権限 ID によって所有されている表を検出します。

```
SELECT  TABLE_SCHEMA, TABLE_NAME FROM QSYS2.SYSTABLES
  WHERE TABLE_OWNER = CURRENT USER AND TABLE_TYPE = 'T'
```

35. LOCALTIMESTAMP および LOCALTIMESTAMP(6) は CURRENT_TIMESTAMP の同義語として指定できます。LOCALTIMESTAMP(*n*) は CURRENT_TIMESTAMP(*n*) の同義語として指定できます。

CURRENT TIMEZONE

CURRENT TIMEZONE 特殊レジスターは、UTC と現行サーバーでの地方時との差を指定します。

この時差は時刻期間 (最初の 2 桁が時、次の 2 桁が分、最後の 2 桁が秒を示す 10 進数) で表されます。[36] 時を示す数値は、-23 から 24 までです。地方時から CURRENT TIMEZONE を引くと、その地方時が UTC に変換されます。

例

以下のステートメントでは、表の IN_TRAY からすべての行を選択して、その値を UTC に合わせます。

```
SELECT RECEIVED - CURRENT TIMEZONE, SOURCE,
      SUBJECT, NOTE_TEXT FROM IN_TRAY
```

SESSION_USER

特殊レジスター SESSION_USER は、現行サーバー側の実行時権限 ID を指定します。この特殊レジスターのデータ・タイプは VARCHAR(128) です。

新規接続の SESSION_USER の初期値は、SYSTEM_USER 特殊レジスターの値と同じです。

SET SESSION AUTHORIZATION ステートメントを実行すれば、その値を変更できます。詳しくは、 1730 ページの『SET SESSION AUTHORIZATION』を参照してください。

例

このステートメントは、ユーザーが自分でそこに置いた表 IN_TRAY から、すべての行を選択しています。

```
SELECT * FROM IN_TRAY
  WHERE SOURCE = SESSION_USER
```

SYSTEM_USER

特殊レジスター SYSTEM_USER は、現行サーバーに接続する権限 ID を指定します。この特殊レジスターのデータ・タイプは VARCHAR(128) です。

例

このステートメントは、ユーザーが自分でそこに置いた表 IN_TRAY から、すべての行を選択しています。

```
SELECT * FROM IN_TRAY
  WHERE SOURCE = SYSTEM_USER
```

USER

特殊レジスター USER は、現行サーバー側の実行時権限 ID を指定します。この特殊レジスターのデータ・タイプは VARCHAR(18) です。

---

36. 協定世界時。以前は GMT (グリニッジ標準時) と呼ばれていました。

新規接続の USER の初期値は、SYSTEM_USER 特殊レジスターの値と同じです。

SET SESSION AUTHORIZATION ステートメントを実行すれば、その値を変更できます。詳しくは、 1730 ページの『SET SESSION AUTHORIZATION』を参照してください。

例

このステートメントは、ユーザーが自分でそこに置いた表 IN_TRAY から、すべての行を選択しています。

```
SELECT * FROM IN_TRAY
  WHERE SOURCE = USER
```

列名

列名の持つ意味は、その列名が使用されている文脈によって異なります。

列名は、次のように使用されることがあります。

- CREATE TABLE ステートメントの中などで列の名前を宣言する。
- CREATE INDEX ステートメントの中などで列を識別する。
- 以下に示すような文脈で列の値を指定する。
  - 集約関数 において、列名は、関数が適用されるグループまたは中間結果表の列のすべての値を指定します。グループと中間結果表については、 787 ページの『第 6 章 照会』の項で説明しています。例えば、MAX(SALARY) は、グループ内の列 SALARY のすべての値に関数 MAX を適用します。
  - *GROUP BY* または *ORDER BY* 文節 の中では、その文節が適用される中間結果表の中のすべての値を、列名によって指定します。例えば、ORDER BY DEPT を指定すると、DEPT という列の値によって中間結果表が順序付けられます。
  - 式、検索条件、またはスカラー関数 では、列名によって、その構造を適用する各行またはグループに対して値を指定します。例えば、検索条件 CODE = 20 がある行に適用される場合、列名 CODE によって指定される値は、それらの行にある列 CODE の値を指定しています。
- FROM 文節の表参照 の中の相関文節 や、選択文節 の AS 文節の中などで、式に列名を指定し、列名を一時的に変更する。

修飾列名

列名の修飾子には、表名、ビュー名、別名、または相関名が可能です。

列名を修飾できるかどうかは、その列名が使用されている文脈によって決まります。

- COMMENT および LABEL ステートメントでは、列名を必ず修飾しなければなりません。
- 列名によって列の値を指定しているところでは、列名を修飾することができます。
- UPDATE ステートメントの割り当て文節 では、列名を修飾することができます。
- INSERT ステートメントの *column-name-list* では、列名を修飾できます。
- 上記以外の文脈では、列名を修飾してはなりません。

修飾子が任意指定の個所では、修飾子は 2 つの役目を果たします。詳しくは、166 ページの『あいまいさを避けるための列名修飾子』および 168 ページの『相関参照内の列名修飾子』を参照してください。

相関名

相関名 は、照会の FROM 文節、および UPDATE または DELETE ステートメントのターゲット表名 またはビュー名 の後に定義できます。

例えば、以下の文節では、X.MYTABLE の相関名として Z を設定しています。

```
FROM X.MYTABLE Z
```

相関名が、表またはビューに関連付けられるのは、その相関名が定義されている文脈の中だけです。したがって、同一の相関名を、別のステートメント内で別の目的のために定義したり、同一のステートメント内の別の文節で定義したりすることができます。

相関名を修飾子として使用することによって、あいまいさを避けたり、相関参照を設定したりすることができます。相関名は、表またはビューの短縮名として使用することも可能です。上記の例では、単に X.MYTABLE を何度も入力するのを避けるために、相関名 Z を使用しても構いません。

表またはビューに対して相関名が指定されている場合、その表またはビューのそのインスタンスの列に対する修飾付き参照では、表名やビュー名ではなく、必ず相関名を使用しなければなりません。例えば、以下の例では、EMPLOYEE に対する相関名が指定されているので、EMPLOYEE.PROJECT への参照は誤りとなります。

```
FROM EMPLOYEE E                          ***INCORRECT***
  WHERE EMPLOYEE.PROJECT='ABC'
```

PROJECT に対する修飾付き参照では、以下のように、代わりに相関名 "E" を使用しなければなりません。

```
FROM EMPLOYEE E
  WHERE E.PROJECT='ABC'
```

FROM 節で指定する名前は、直接的 か間接的 のどちらかです。 相関名は、常に直接的な名前です。相関名が指定されていない場合、表名またはビュー名は、その FROM 文節の中で直接的 であると言われます。例えば、以下の FROM 文節では、EMPLOYEE には相関名が指定され、DEPARTMENT には相関名が指定されていません。したがって、DEPARTMENT は直接的な名前であり、EMPLOYEE は間接的な名前となります。

```
FROM EMPLOYEE E, DEPARTMENT
```

FROM 文節で直接的な表名またはビュー名は、その FROM 文節で直接的な他のいかなる表名またはビュー名とも異なっていなければなりません。また、その FROM 文節のいかなる相関名とも異なっていなければなりません。これらの名前は、修飾のない表名またはビュー名を修飾した後で比較されます。

以下に示す最初の 2 つの FROM 文節では、直接的な名前である EMPLOYEE への参照がそれぞれに 1 つしか入っていないので、正しい FROM 文節となります。

1. 次の FROM 文節では、

   ```
   FROM EMPLOYEE E1, EMPLOYEE
   ```

   FROM 文節の 2 番目の EMPLOYEE にある列は、EMPLOYEE.PROJECT などの修飾付き参照によって指示されます。EMPLOYEE の 1 番目のインスタンスに対する修飾子付き参照では、相関名「E1」を使用する (E1.PROJECT) 必要があります。
2. 次の FROM 文節では、

   ```
   FROM EMPLOYEE, EMPLOYEE E2
   ```

FROM 文節の最初の EMPLOYEE にある列は、EMPLOYEE.PROJECT などの修飾付き参照によって指示されます。EMPLOYEE の 2 番目のインスタンスに対する修飾子付き参照では、相関名"E2"を使用する (E2.PROJECT) 必要があります。

3. 次の FROM 文節では、

```
FROM EMPLOYEE, EMPLOYEE                    ***INCORRECT***
```

この文節に含まれている 2 つの表名 (EMPLOYEE と EMPLOYEE) は同じなので、これは許されません。

4. 次のステートメントでは、

```
SELECT *
  FROM EMPLOYEE E1, EMPLOYEE E2            ***INCORRECT***
  WHERE EMPLOYEE.PROJECT='ABC'
```

FROM 文節内にある 2 つの EMPLOYEE が、両方とも相関名を持っているので、EMPLOYEE.PROJECT という修飾付き参照は誤りです。その代わりに、PROJECT に対する参照は、どちらかの相関名を使って、E1.PROJECT または E2.PROJECT と修飾しなければなりません。

5. 次の FROM 文節では、

```
FROM EMPLOYEE, X.EMPLOYEE
```

2 番目の EMPLOYEE にある列に対する参照では、X.EMPLOYEE (X.EMPLOYEE.PROJECT) を使用しなければなりません。この FROM 文節は、ステートメントの権限 ID が X ではない場合に限って有効です。

FROM 文節で指定する相関名は、以下の名前とは異なっていなければなりません。

- 同じ FROM 文節の他の相関名
- 同じ FROM 文節で直接的な修飾のない表名またはビュー名
- 同じ FROM 文節で直接的な修飾のある表名またはビュー名の 2 番目の SQL ID

例えば、以下のような FROM 文節は誤りです。

```
FROM EMPLOYEE E, EMPLOYEE E
FROM EMPLOYEE DEPARTMENT, DEPARTMENT       ***INCORRECT***
FROM X.T1, EMPLOYEE T1
```

以下のような FROM 文節は、参照が混同される恐れがありますが、構文上は正しい FROM 文節です。

```
FROM EMPLOYEE DEPARTMENT, DEPARTMENT EMPLOYEE
```

また、FROM 文節で相関名を使用すると、結果表の列に関連付ける列名リストを指定する選択も可能になります。相関名と同様に、これらのリストされた列名は、照会の全体にわたって列の参照で使う必要がある直接的な名前になります。列名リストが指定されている場合は、基本表の列名は間接的なものになります。

次の FROM 文節では、

```
FROM DEPARTMENT D (NUM,NAME,MGR,ANUM,LOC)
```

D.NUM などの修飾付きの参照は、DEPTNO として表で定義されている DEPARTMENT 表の最初の列を表します。この FROM 文節を用いた D.DEPTNO への参照は、列名 DEPTNO が間接的な列名であるため、誤りです。

列のリストを指定する場合は、*table-reference* にある列数と同じ数の名前をそのリストに指定する必要があります。それぞれの列名は固有であり、修飾されていない名前でなければなりません。

あいまいさを避けるための列名修飾子

関数、GROUP BY 文節、ORDER BY 文節、式、または検索条件の文脈では、DELETE または UPDATE ステートメントに指定された特定のターゲット表またはビュー内の列にある値、または FROM 文節の表参照 を列名によって参照します。

列を含んでいる可能性のある表、ビュー、および *table-reference* [37]は、そのコンテキストのオブジェクト・テーブル と呼ばれます。同じ名前の列が、複数のオブジェクト・テーブルに入っていることもあります。列名を修飾する理由の 1 つは、その列がどのオブジェクト・テーブルの列であるかを指示するためです。 SQL パラメーター、変数、および列名の間でのあいまいさを避ける方法については、 1773 ページの『SQL パラメーターおよび変数の参照』を参照してください。

LATERAL または TABLE キーワードに続くネストされた表の式は、FROM 文節内でその前にある表参照 をオブジェクト・テーブルと見なします。ネストされた表の式の後に続く表参照 はオブジェクト・テーブルと見なされません。

表指定子

特定のオブジェクト表を指定する修飾子は、表指定子 と呼ばれます。オブジェクト・テーブルを識別する文節では、そのオブジェクト・テーブルを指示する表指定子も設定します。

例えば、SELECT 文節の式で使用するオブジェクト・テーブルは、次のように SELECT 文節の後の FROM 文節で指定します。

```
SELECT CORZ.COLA, OWNY.MYTABLE.COLA
  FROM OWNX.MYTABLE CORZ, OWNY.MYTABLE
```

FROM 文節の表指定子は、次の方法で設定します。

- 表名またはビュー名の後に付く名前は、相関名であると同時に表指定子でもあります。したがって、CORZ は表指定子です。CORZ は、選択リスト内の最初の列名を修飾するために使用されています。
- SQL 命名規則では、直接表名やビュー名は表指定子です。したがって、OWNY.MYTABLE は表指定子です。OWNY.MYTABLE は、選択リスト内の 2 番目の列名を修飾するために使用されています。
- システム命名規則では、直接表名や直接ビュー名の表指定子は、修飾のない表名またはビュー名です。次の例で、MYTABLE は、OWNY/MYTABLE の表指定子です。

```
SELECT CORZ.COLA, MYTABLE.COLA
  FROM OWNX/MYTABLE CORZ, OWNY/MYTABLE
```

37. *joined-table* の場合は、*joined-table* 内の各 *table-reference* がオブジェクト・テーブルです。

文脈内のオブジェクト・テーブルとして、同一の表が何度も指定されることがあります。この場合は、文脈内に現れる個々の表を明確に指示するために、表ごとに別々の相関名を使用しなければなりません。例えば、以下の FROM 文節では、最初の表 EMPLOYEE を参照するために X を定義し、 2 番目の表 EMPLOYEE を参照するために Y を定義しています。

```
SELECT * FROM EMPLOYEE X,EMPLOYEE Y
```

未定義またはあいまいな参照の回避

列名によって列の値を参照する場合は、その列名が、必ず 1 つのオブジェクト・テーブルで解決できなければなりません。

次のような場合は、エラーと見なされます。

- 指定した名前を持つ列が入っているオブジェクト・テーブルが存在しない。この参照は未定義になります。
- 列名が表指定子によって修飾されているときに、指定した名前を持つ列が、指定した表に存在しない。この参照も未定義になります。
- 列名が修飾されていないときに、その名前を持つ列が、複数のオブジェクト・テーブルに存在する。この参照はあいまいになります。
- 列名が表指定子によって修飾されているときに、指定した表が FROM 文節内で固有ではなく、指定した表が出現する 2 回とも、表に列が含まれている。この参照はあいまいになります。
- 列名がネストされた表の式にあり、この表式が LATERAL または TABLE キーワードの後にはないか、列名がある表関数またはネストされた表の式が右外部結合、全外部結合または右例外結合の右オペランドではなく、列名がネストされた表式の全選択内の表参照 の列を参照していない。この参照は未定義になります。

あいまいな参照を避けるには、固有のものとして定義されている表指定子によって列名を修飾します。同じ名前の列が、異なる名前を持ついくつかのオブジェクト・テーブルに入っている場合は、オブジェクト・テーブルの名前を表指定子として使うことができます。相関名に続けて列名リストを使用して、1 つのオブジェクト・テーブルの列に固有の名前を付けることにより、表指定子を使用しなくても、あいまいな参照を避けることができます。

直接的な表名による表指定子で列を修飾する場合は、直接的な表名の修飾形式と非修飾形式のいずれかを使用します。ただし、使用する修飾子および表は、表名またはビュー名と表指定子を完全に修飾したものと同じでなければなりません。

1. デフォルトのスキーマが CORPDATA である場合は、次のようになります。

   ```
   SELECT CORPDATA.EMPLOYEE.WORKDEPT
     FROM EMPLOYEE
   ```

   このステートメントは、有効なステートメントです。

2. デフォルトのスキーマが REGION である場合は、次のようになります。

   ```
   SELECT CORPDATA.EMPLOYEE.WORKDEPT
     FROM EMPLOYEE                          ***INCORRECT***
   ```

このステートメントは無効です。EMPLOYEE は REGION.EMPLOYEE という表を表していますが、WORKDEPT の修飾子は、CORPDATA.EMPLOYEE という異なる表を表しているからです。

3. デフォルトのスキーマが REGION である場合は、次のようになります。

```
SELECT EMPLOYEE.WORKDEPT
  FROM CORPDATA.EMPLOYEE                          ***INCORRECT***
```

このステートメントは無効です。選択リスト内の EMPLOYEE は REGION.EMPLOYEE という表を表していますが、 FROM 文節で明示的に修飾されている表名は、CORPDATA.EMPLOYEE という異なる表を表しているからです。この場合、選択リストからその表修飾子を省略するか、または FROM 文節内の表指定子に相関名を定義して、その相関名をステートメント内で列名の指定子を使用します。

相関参照内の列名修飾子

副選択 は、照会の一形式であり、各種の SQL ステートメントのコンポーネントとして使用できます。

副照会の詳細については、 787 ページの『第 6 章 照会』を参照してください。副照会 は全選択 1 つで、括弧で囲んだ形式をとります。例えば、副照会 は検索条件の中で使用することができます。照会の FROM 節で使用される全選択は、ネストされた表の式 と呼びます。

副照会は、独自の検索条件を含むことができ、これらの検索条件は逆に副照会を含むことができます。したがって、SQL ステートメントは副照会の階層を含むことができます。副照会を含んでいる階層のエレメントは、その階層に含まれている副照会より上位レベルにあるといわれます。

階層の各エレメントは、1 つの文節を持ち、その文節によって 1 つ以上の表指定子を設定します。階層の最上位レベルにある UPDATE または DELETE ステートメントを除いて、この文節は FROM 文節になります。副照会に含まれる検索条件、選択リスト、JOIN 文節、表関数の引数、または LATERAL キーワードの後に来るネストされた表の式 では、その階層自身の要素である FROM 文節によって識別されている表の列を参照できるだけでなく、その階層自身の要素から階層の最上位レベルまでのパスに沿って、どのレベルで識別されている表の列も参照することができます。上位のレベルで指定される表の列への参照は、相関参照 と呼ばれます。LATERAL キーワードを使ってネストされた表の式 と同じレベルで識別されている表の列への参照を、水平相関 と呼びます。

Q が表 T に対して定義されている相関名である場合、表 T の列 C に対する相関参照は、C、T.C、または Q.C という形式をとります。以下の説明は、相関参照が常に修飾付きの列名の形式をとり、その修飾子が相関名であることを前提にしています。

Q.C が相関参照となるのは、次の 3 つの条件が満たされている場合だけです。

- 副照会に含まれる検索条件、選択リスト、JOIN 文節、または表関数の引数で Q.C が使用されている。
- Q が、その副照会の FROM 文節、選択リスト、JOIN 文節、または表関数の引数で使用されている表を指示していない。

- Q が、上位レベルで使用されている表を指示している。

Q.C によって参照される列 C は、Q を表またはビューの表指定子として使用しているレベルの表またはビューにある列です。同一の表またはビューを複数のレベルから識別する可能性があるので、表指定子には、固有の相関名を使用するようにしてください。Q を使用して複数のレベルから表を指示した場合、Q.C は、Q.C を使用している副照会の上位にある階層のうち、最下位レベルの階層を参照します。

以下のステートメントでは、Q を T1 および T2 に対する相関名として使用していますが、Q.C は T2 に関連付けられている相関名を参照します。これは、Q.C を使用している副照会の上位にある階層の最下位レベルで、Q が T2 に関連付けられているためです。

```
SELECT *
  FROM T1 Q
  WHERE A < ALL (SELECT B
                 FROM T2 Q
                 WHERE B < ANY (SELECT D
                                FROM T3
                                WHERE D = Q.C))
```

相関参照における修飾されていない列名

修飾されていない列名も相関参照となります。

それは次のような場合です。

- その列が、副照会の検索条件で使用されている。
- その列が、副照会の FROM 文節で使用されている表に含まれていない。
- その列が、上位レベルで使用されている表に含まれている。

修飾されていない相関参照は、SQL ステートメントが分かりにくくなるので、なるべく使用しないでください。列は、ステートメントが準備される時点で、その列が見つかった表によって暗黙的に修飾されます。この暗黙の修飾は、いったん行われると、ステートメントが準備し直されるまで変更されることはありません。修飾されていない相関参照を持つ SQL ステートメントが準備または作成されている場合は、警告が戻されます (SQLSTATE 01545)。

変数

SQL ステートメントの変数 は、SQL ステートメントの実行時に変更が可能な値を指定します。

SQL ステートメントで使用される変数 には幾つかのタイプがあります。

グローバル変数
: グローバル変数は、組み込みグローバル変数の場合もあれば、ユーザー定義グローバル変数の場合もあります。グローバル変数を参照する方法については、『グローバル変数』を参照してください。

ホスト変数
: ホスト変数は、ホスト言語のステートメントによって定義されます。ホスト変数を参照する方法については、 172 ページの『ホスト変数に対する参照』を参照してください。

遷移変数
: 遷移変数はトリガーの中で定義されるもので、列の古い値または新しい値を参照します。遷移変数を参照する方法については、 1302 ページの『CREATE TRIGGER』を参照してください。

SQL 変数
: SQL 変数は、SQL 関数、SQL プロシージャー、またはトリガー内の SQL 複合ステートメントによって定義されます。 SQL 変数について詳しくは、1773 ページの『SQL パラメーターおよび変数の参照』を参照してください。

SQL パラメーター
: SQL パラメーターは、CREATE FUNCTION (SQL スカラー)、CREATE FUNCTION (SQL 表)、または CREATE PROCEDURE (SQL) ステートメントで定義されます。 SQL パラメーターについて詳しくは、 1773 ページの『SQL パラメーターおよび変数の参照』を参照してください。

パラメーター・マーカー
: 動的 SQL ステートメントでは、変数を参照することはできません。代わりに、SQL 記述子の中でパラメーター・マーカーを定義して、使用します。パラメーター・マーカーについて詳しくは、 1603 ページの『PREPARE』のパラメーター・マーカーを参照してください。

グローバル変数

グローバル変数は、名前付きのメモリー変数であり、SQL ステートメントでアクセスして変更できます。

Db2 データベース管理システムでは、以下のタイプのグローバル変数をサポートしています。

組み込みグローバル変数
: 組み込みグローバル変数は、データベース管理システムの一部であり、データベース・マネージャーで実行される SQL ステートメントで使用できます。組み込みグローバル変数のリストと詳細については、 277 ページの『第 3 章 組み込みグローバル変数』を参照してください。

ユーザー定義グローバル変数

ユーザー定義グローバル変数を使用すれば、SQL ステートメント間でリレーショナル・データを共有できます。アプリケーション・ロジックでそのデータ転送をサポートする必要はありません。

ユーザー定義グローバル変数は、特定のセッションに関連付けられていて、そのセッションに固有の値を持っています。ユーザー定義セッション・グローバル変数は、その変数が定義されているデータベースに対して実行されるすべてのアクティブ SQL ステートメントで使用できます。ユーザー定義グローバル変数は複数のセッションに関連付けることができますが、その値はセッションごとに固有になります。ユーザー定義グローバル変数はシステム・カタログで定義されます。

グローバル変数に対するアクセスを制御するには、GRANT (グローバル変数特権) ステートメントと REVOKE (グローバル変数特権) ステートメントを使用します。

グローバル変数の名前は、修飾名です。非修飾グローバル変数名が ALTER、CREATE、COMMENT、DROP、GRANT、または REVOKE ステートメントのメイン・オブジェクトの場合は、非修飾の表名の修飾と同じ規則を使用して暗黙的に名前が修飾されます。それ以外の場合は、SQL パスが名前の解決に使用されます。

静的 SQL ステートメントと静的 SQL ルーチンの場合は、すべての表参照が初めて解決されるときに、ステートメントのグローバル変数も解決されます。ビュー、トリガー、他のグローバル変数の場合は、オブジェクトの作成時に、グローバル変数が解決されます。グローバル変数の名前を再度解決しなければならないときに、例えば、別のスキーマで追加された同じ名前の新しいグローバル変数がその SQL パスに存在すると、動作が変わってしまう可能性もあります。

動的ステートメントで参照するグローバル変数は、そのステートメントが最初に準備される時点で解決されます。表の変更に伴ってステートメントをリフレッシュしなければならない場合を除き、グローバル変数が再び解決されることはありません。

グローバル変数の名前は、SQL ステートメントで参照する表またはビューの列の名前や、SQL ルーチンで使用する SQL 変数または SQL パラメーターの名前と同じでもかまいません。同一の名前は明示的に修飾する必要があります。名前を修飾していない場合や、修飾していてもまだ不明確な場合は、以下の優先順位の規則に基づいて名前が解決されます。

- ステートメントで参照されている現行サーバーの既存の表またはビューの列の名前と同じかどうかが確認されます。
- SQL ルーチンでグローバル変数を使用する場合は、SQL 変数、SQL パラメーター、遷移変数の名前と同じかどうかが確認されます。
- どちらの規則でも同じ名前が見つからない場合は、グローバル変数と見なされます。
- SQL_GVAR_BUILD_RULE QAQQINI オプションが *EXIST であり、グローバル変数がプリコンパイル時または SQL ルーチン作成時に存在していない場合、エラーが発行されます。

グローバル変数が、トリガー、ビュー、ルーチン、他のグローバル変数で参照されている場合は、そのステートメントまたはオブジェクトで完全修飾グローバル変数名の依存関係が記録されます。さらに、該当する場合、ステートメントに使用されている権限 ID がチェックされ、グローバル変数に対する適切な権限があるかどうか検査されます。

グローバル変数は、変数を使用できる SQL ステートメントであればどのステートメントでも使用できます。式の中でもグローバル変数を参照できますが、以下の例外があります。

- 検査制約
- マテリアライズ照会表 (MQT)
- 索引
- 照会に以下の指定がある場合は、グローバル変数を使用できません。
  - 分散表
  - 読み取りトリガーを指定する表
  - 複数の物理ファイル・メンバー上に構築された論理ファイル

許可**:** グローバル変数がステートメント内で参照される場合、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別されるグローバル変数に対して、
  - グローバル変数が参照される場合、グローバル変数に対する READ 特権
  - グローバル変数に値が割り当てられる場合、グローバル変数に対する WRITE 特権
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

グローバル変数の値は、FETCH、SET、SELECT INTO、VALUES INTO のいずれかのステートメントで変更できます。CALL ステートメントの OUT パラメーターまたは INOUT パラメーターの引数として使用する場合にも、変更が可能です。

ホスト変数に対する参照

ホスト変数 とは、COBOL データ項目、RPG フィールド、または SQL ステートメントで参照される PLI、REXX、C++、C、あるいは Java の変数を指します。ホスト変数は、ホスト言語のステートメントによって定義されます。

動的 SQL ステートメントでは、ホスト変数を参照できません。代わりに、パラメーター・マーカーを使用する必要があります。パラメーター・マーカーについての詳細は、 176 ページの『動的 SQL での変数』を参照してください。

SQL ステートメント中のホスト変数 は、ホスト変数の宣言の規則に従ってプログラム内で記述されたホスト変数でなければなりません。

Java、REXX、および RPG 以外のすべてのホスト言語では、SQL ステートメントで使用されるホスト変数はすべて、SQL 宣言セクションで宣言されていなければなりません。 REXX では、変数は宣言されている必要はありません。Java と RPG には、宣言セクションはありませんが、ホスト変数は、そのプログラム全体にわた

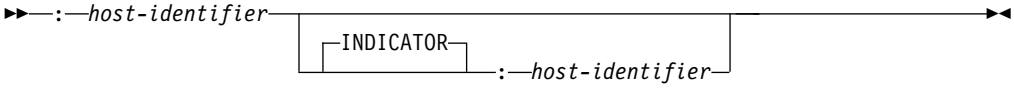
って宣言されます。 SQL 宣言セクションで宣言されている変数と同じ名前を持つ変数を、SQL 宣言セクションの外側で宣言してはなりません。 SQL 宣言セクションは、BEGIN DECLARE SECTION で開始し、END DECLARE SECTION で終了します。

ホスト変数の使用についての詳細は、「組み込み SQL プログラミング」トピックを参照してください。

FETCH、SELECT INTO、SET 変数の INTO 文節、GET DESCRIPTOR ステートメント、または VALUE INTO ステートメントにおける変数は、結果列の値を割り当てるホスト変数を識別します。GET DIAGNOSTICS ステートメント内の変数は、診断値を割り当てるホスト変数を識別します。 CALL または EXECUTE ステートメント内のホスト変数は、プロシージャーの実行後に値を割り当てられる出力引数、プロシージャーに入力値を提供する入力引数、または入力引数と出力引数の両方にすることができます。その他のすべてのコンテキストでの変数は、アプリケーション・プログラムからデータベース・マネージャーに渡される値を指定します。

非 Java 変数参照:

Java 以外のすべての言語では、変数 参照の一般的な形式は次のとおりです。



それぞれのホスト *ID* は、ソース・プログラム内で宣言しておく必要があります。2 番目のホスト *ID* によって指定される変数は標識変数 と呼ばれ、データ・タイプは短整数でなければなりません。標識変数には正常標識変数 と拡張標識変数 の 2 つの形式があります。

正常標識変数は、次のような用途に使用します。

- NULL 以外の値を指定します。標識変数を 0 (ゼロ) または正の値にすると、関連した最初のホスト *ID* がこのホスト変数参照の値を提供することを指定します。
- NULL 値を指示します。標識変数の負の値は、NULL 値を示します。
- 出力で、以下の数値変換エラーのいずれかを示します。
  - 数値変換エラー (アンダーフローまたはオーバーフロー)。
  - 算術式エラー (0 による除算)。
  - 数値が無効。

  標識変数の値 -2 は、これらのいずれかの警告があるので、NULL の結果を示します。
- 出力で、以下のストリング・エラーのいずれかを示します。
  - 文字を変換できなかった。
  - 混合 (MIXED) データが正しく形成されていない。

  標識変数の値 -2 は、これらのいずれかの警告があるので、NULL の結果を示します。

- 出力で、以下の日時エラーのいずれかを示します。
  - 日付またはタイム・スタンプの変換エラー (指定されている日付形式の有効な範囲内にない日付またはタイム・スタンプ)。
  - 日付/時刻の値のストリング表現が正しくない。

  標識変数の値 -2 は、これらのいずれかの警告があるので、NULL の結果を示します。
- 出力で、以下の各種エラーのいずれかを示します。
  - スカラー関数 SUBSTR の引数が範囲外。
  - 暗号化解除関数の引数に無効なデータ・タイプが含まれている。

  標識変数の値 -2 は、これらのいずれかの警告があるので、NULL の結果を示します。
- 出力で、ホスト変数に割り当てたストリングが切り捨てられた場合に、そのストリングの元の長さを記録します。標識変数が用意されていない場合にストリングが切り捨てられても、エラー条件とはなりません。
- 出力で、ホスト変数に割り当てた時刻が切り捨てられた場合に、その時刻の秒の部分を記録します。標識変数が用意されていない場合に時刻が切り捨てられても、エラー条件とはなりません。

例えば、:V1:V2 というホスト変数参照を使用して挿入値または更新値を指定した場合に、V2 の値が負ならば、指定した値が NULL 値であることを示します。V2 が負でない場合、指定した値が V1 の値になります。

同様に、:V1:V2 を CALL、FETCH、SELECT INTO、または VALUES INTO ステートメントで指定した場合に、戻された値が NULL であれば、V1 は未定義であり、V2 に負の値がセットされます。セットされる負の値は、次のとおりです。

- -1、これは選択された値が NULL 値であったことを示します。
- -2、これは、外側の副選択の選択リストのデータ・マッピング・エラーにより NULL 値が戻されたことを示します。[38]

参照によって戻された値が NULL 値でなければ、その値が V1 に割り当てられ、V2 にはゼロがセットされます (ただし、V1 への割り当ての際に切り捨てが必要だった場合は、V2 にそのストリングの元の長さがセットされます)。また、割り当ての際に、時刻の秒の部分を切り捨てる必要があった場合は、切り捨てられた秒数が V2 にセットされます。

2 番目のホスト *ID* が省略された場合は、そのホスト変数 は標識変数をもちません。このような場合、ホスト変数 :V1 によって指定された値は常に V1 の値になり、NULL 値をその変数に割り当てることはできません。したがって、対応する結果列に NULL 値を入れることができない場合以外は、この形式を使用してはなりません。この形式を使用し、しかも列に NULL 値が含まれている場合、データベース・マネージャーは実行時にエラーを戻します (SQLSTATE 23502)。

拡張標識変数は、入力ホスト変数に限定されます。その目的は次のとおりです。

38. 特定のスカラー関数や算術式で、データ・マッピング・エラーのための NULL 値が戻されることがありますが、算術式またはスカラー関数の引数がNULL 可能でない場合は、その結果の列はNULL 可能とは見なされません。

- NULL 以外の値を指定します。0 (ゼロ) または正の値にすると、関連したホスト *ID* がこのホスト変数参照の値を提供することを指定します。
- NULL 値を指示します。値 -1、-2、-3、-4、および -6 は、NULL 値を指定します。
- DEFAULT 値を指定します。値 -5 は、このホスト変数のターゲット列がそのデフォルト値に設定されることを指定します。
- UNASSIGNED 値を指定します。値 -7 は、このホスト変数のターゲット列が、ステートメントで指定されなかったかのように扱われることを指定します。

拡張標識変数は、要求される場合のみ使用可能になります。それ以外の場合、すべての標識変数は正常標識変数です。正常標識変数と比較すると、拡張標識変数には、NULL 値と NULL 以外の値の使用に関する追加の制限がありません。拡張標識は、ホスト構造のある標識構造で使用できます。拡張標識変数値 DEFAULT および UNASSIGNED が使用可能になる場合に関する制限は、それらの値がどのようにホスト・アプリケーションで表示されていても、一様に適用されます。 DEFAULT および UNASSIGNED 拡張標識変数は、ある 1 つの列に割り当てられる単一のホスト・パラメーターを含む式として、または、そういったホスト・パラメーターのキャストとしてのみ使用できます。出力標識変数値は、拡張標識変数ではありません。

拡張標識変数が有効になっている場合は、正の値、ゼロ、上記の 6 つの負の値以外の標識変数値を使用しないでください。DEFAULT 値および UNASSIGNED 値は、それらがサポートされているコンテキスト (INSERT、 MERGE、および UPDATE ステートメント) でのみ指定されなければなりません。これらは CALL ステートメント内ではサポートされません。

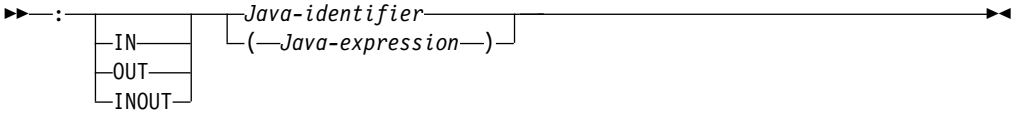
C、C++、ILE RPG、および PL/I でホスト変数を参照する SQL ステートメントは、そのホスト変数の宣言の有効範囲になければなりません。カーソルに対する SELECT ステートメントでホスト変数を参照する場合、そのホスト変数の宣言の有効範囲内になければならないのは、DECLARE CURSOR ステートメントではなく、OPEN ステートメントです。

ストリング・ホスト変数の CCSID は、次のいずれかです。

- DECLARE VARIABLE ステートメントで指定された CCSID、または
- 該当のホスト変数に対して CCSID 文節を伴う DECLARE VARIABLE の指定がない場合には、そのホスト変数を含む SQL ステートメントが実行される時点のアプリケーション・リクエスターのデフォルト CCSID (ただし、ASCII などの、Unicode 以外の外部コード化体系に対する CCSID でない場合のみ)。外部コード体系に対する CCSID である場合には、ホスト変数は、現行サーバーのデフォルトの CCSID に変換されます。

Java 変数参照:

Java の場合、ホスト変数参照の一般形式は次のとおりです。



Java の場合、標識変数は使用されません。その代わりに、Java クラスのインスタンスは NULL 値に設定できます。 Java プリミティブ・タイプとして定義されている変数は、NULL 値に設定できません。

IN、OUT、または INOUT を指定しない場合、変数を使用するコンテキストによってデフォルトが変わります。 Java 変数が INTO 文節で使用される場合、OUT がデフォルトです。それ以外の場合は、IN がデフォルトです。 Java 変数について詳しくは、「IBM Developer Kit for Java」を参照してください。

例

PROJECT 表を使用して、プロジェクト (PROJNO) 'IF1000' について、ホスト変数 PNAME (VARCHAR(26)) をプロジェクト名 (PROJNAME) に、ホスト変数 STAFF (DECIMAL(5,2)) を平均人員レベル (PRSTAFF) に、そしてホスト変数 MAJPROJ (CHAR(6)) を主プロジェクト (MAJPROJ) に設定します。PRSTAFF と MAJPROJ の列には NULL 値が入っている可能性があるので、標識変数の STAFF_IND (SMALLINT) と MAJPROJ_IND (SMALLINT) を指定しています。

```
SELECT PROJNAME, PRSTAFF, MAJPROJ
  INTO :PNAME, :STAFF :STAFF_IND, :MAJPROJ :MAJPROJ_IND
  FROM PROJECT
  WHERE PROJNO = 'IF1000'
```

動的 SQL での変数

動的 SQL ステートメントでは、変数の代わりにパラメーター・マーカーが使用されます。パラメーター・マーカーは疑問符 (?) で表し、アプリケーションが値を提供する動的 SQL ステートメント内の位置、すなわち、ステートメント・ストリングが静的 SQL ステートメントであれば、変数が存在する位置を示します。

次の例は、ホスト変数と、パラメーター・マーカーを使った動的ステートメントを使用する静的 SQL を示しています。

```
INSERT INTO DEPT
  VALUES( :HV_DEPTNO, :HV_DEPTNAME, :HV_MGRNO:IND_MGRNO, :HV_ADMRDEPT)

INSERT INTO DEPT
  VALUES( ?, ?, ?, ? )
```

パラメーター・マーカーについて詳しくは、 1603 ページの『PREPARE』のパラメーター・マーカーを参照してください。

LOB または XML 変数の参照

通常の LOB または XML 変数、LOB または XML ロケーター変数、および LOB または XML ファイル参照変数を定義することができます。

通常の LOB または XML 変数、LOB または XML ロケーター変数、および LOB または XML ファイル参照変数は、以下のホスト言語で定義することができます。

- C
- C++
- ILE RPG
- ILE COBOL
- PL/I (LOB のみ)

LOB または XML が許可されている場合は、構文図における変数 という用語は、通常の変数、ロケーター変数、またはファイル参照変数を意味していることになります。これらの変数はホスト・プログラム言語での固有のデータ・タイプではないため、SQL 拡張子が使用され、プリコンパイラーはそれぞれの変数を表すために必要なホスト言語構成を生成します。

LOB または XML 値全体を収容できるほどの大きな変数を定義することが可能であり、サーバーからのデータ転送の遅れに関するパフォーマンス上の利点が必要ない場合には、LOB または XML ロケーターは不要です。しかしながら、LOB または XML 値全体を一時記憶域に保管するということは、ホスト言語の制約、記憶域の制限、またはパフォーマンス要件により、受け入れられないことがしばしばあります。 LOB または XML 値全体を一時的に保管することが受け入れられない場合、LOB または XML 値を LOB または XML ロケーターによって参照することが可能であり、LOB または XML 値の一部にアクセスすることができます。

LOB または XML ロケーター変数の参照

LOB または XML ロケーター変数 は、アプリケーション・サーバーで LOB または XML 値を表すロケーターの入った変数です。

LOB または XML ロケーター変数は、以下のホスト言語で定義することができます。

- C
- C++
- ILE RPG
- ILE COBOL
- PL/I (LOB のみ)

LOB および XML 値を扱うためのロケーターの使用法の詳細については、 92 ページの『ロケーターを用いたラージ・オブジェクトの操作』を参照してください。

SQL ステートメントのロケーター変数は、ロケーター変数の宣言の規則に従って、プログラムで記述されている LOB または XML ロケーター変数を識別するものでなければなりません。これは常に、SQL ステートメントにより間接的に行われます。例えば、C では以下のようになります。

```
static volatile SQL TYPE IS CLOB_LOCATOR *loc1;
```

他のすべての変数と同様に、LOB または XML ロケーター変数は、関連した標識変数を持つことができます。LOB または XML ロケーター変数の標識変数は、他のデータ・タイプの標識変数と同じような働きをします。 NULL 値がデータベースから戻されると、標識変数が設定されますが変数は変更ありません。LOB または XML ロケーターに関連した標識変数がヌルの場合、参照された LOB または XML はヌルです。これは、ロケーターが NULL 値を指すことはあり得ないということを意味します。

ロケーター変数が、現在、いかなる値も表していない場合は、ロケーター変数が参照されたときに、エラーが起こります。

トランザクション・コミットまたはトランザクション終了の場合、そのトランザクションが獲得したすべての LOB または XML ロケーターは解放されます。

LOB または XML ロケーターが、必ず最初に LOB または XML ロケーターを生成したアプリケーション・サーバーで実行される SQL ステートメント内のみで実行されるようにするのは、アプリケーション・プログラマーの役目です。例えば、LOB ロケーターが 1 つのアプリケーション・サーバーから返され、LOB ロケーター変数に割り当てられたとします。この LOB ロケーター変数が、その後、別のアプリケーション・サーバーで実行される SQL ステートメントで使用されると、予期しない結果が起こります。

LOB または XML ファイル参照変数の参照

LOB または XML ファイル参照変数 は、LOB または XML の直接ファイル入出力に使用されます。

LOB または XML ファイル参照変数 は、以下のホスト言語で定義することができます。

- C
- C++
- ILE RPG
- ILE COBOL
- PL/I (LOB のみ)

これらは固有のデータ・タイプではないため、SQL 拡張子が使用され、プリコンパイラーはそれぞれの変数を表すために必要なホスト言語構成を生成します。

ファイル参照変数は、LOB または XML ロケーターが LOB または XML データを含むのではなく、表すのと同じように、ファイルを (含むのではなく) 表します。データベース照会、更新、および挿入では、ファイル参照変数を使用して、単一の列の値を保管したり、検索します。参照されるファイルはアプリケーション・リクエスター内になければなりません。

他のすべての変数と同様に、ファイル参照変数は、関連した標識変数を持つことができます。ファイル参照変数の標識変数は、他のデータ・タイプの標識変数と同じような働きをします。 NULL 値がデータベースから戻されると、標識変数が設定されますが変数は変更ありません。ファイル参照変数に関連した標識変数がヌルの場合、参照された LOB または XML 値はヌルです。これは、ファイル参照変数が NULL 値を指すことはあり得ないということを意味します。

ファイル参照変数の長さ属性は、LOB または XML の最大長であると想定されます。

ファイル参照変数は、現在、ルート (/)、QOpenSys、および UDFS ファイル・システムでサポートされています。ファイルが作成されると、ファイルに書き込み中のデータの CCSID が与えられます。現在、混合 CCSID はサポートされていません。ファイル参照変数を用いて作成されたファイルを使用するには、ファイルを 2 進モードでオープンする必要があります。

ファイル参照変数の詳細については、「SQL プログラミング」トピック集を参照してください。

XML 変数の参照

XML 変数は、ホスト言語において LOB 変数の一種として定義されます。

LOB 同様、XML 変数はストリング、ロケーター、またはファイル参照変数として定義できます。CLOB、DBCLOB、または BLOB データとして扱うことができます。XML 対応 CCSID を CLOB 変数および DBCLOB 変数に割り当てることが可能です。この値を使用して、LOB 変数内に格納された XML データのエンコードを定義します。BLOB として定義された XML 変数には、エンコード処理を決定する XML 1.0 仕様に従って、データ内の指定どおりにエンコードされるデータが入れられます。例えば、C では、CLOB ストレージ構造を使用する XML 変数を以下のように定義できます。

```
SQL TYPE IS XML AS CLOB(10K);
```

アプリケーションの XML 変数宣言には LOB タイプ指定が含まれていますが、この変数宣言は XML データ・タイプ (宣言で使用される LOB タイプではない) であると見なされます。また、このアプリケーションでは、XML 変数の代わりに、非 XML 変数を使用しても構いません。例えば、準備されたステートメントの実行時、このアプリケーションでは文字変数を使用してそのステートメントにある XML パラメーター・マーカーを置き換えても構いません。

XML データ・タイプは他のすべてのデータ・タイプとは非互換ですが、XML および 非 XML データ・タイプの両方を XML データとの入力と出力に対して使用可能です。アプリケーションでは SQL ステートメントにおける入出力変数として、文字、Unicode グラフィック、または 2 進数の変数を使用できます。

XML 変数を SQL ステートメントに対する入力として使用する場合、値は暗黙的に解析されます。

ホスト構造

ホスト構造 とは、SQL ステートメントで参照する COBOL のグループ、PL/I、C、C++ の構造、RPG のデータ構造のことをいいます。 Java と REXX には、ホスト構造 に相当するものがありません。

ホスト構造は、ホスト言語のステートメントによって定義されます。これについては、「組み込み SQL プログラミング」トピック集で説明しています。ここで使用する "ホスト構造" という用語には、SQLCA や SQLDA は含まれません。

ホスト構造参照の形式は、ホスト変数参照の形式と同じです。:S1:S2 の参照は、S1 がホスト構造を指定している場合は、ホスト構造参照です。S1 がホスト構造を指している場合、S2 は、短整数変数、または短整数変数の配列のいずれかでなければなりません。S1 はホスト構造で、S2 はその標識配列です。

ホスト構造は、ホスト変数のリストを参照できる文脈であれば、どのような文脈でも参照できます。ホスト構造の参照は、その構造に含まれている各ホスト変数を、ホスト言語の構造宣言で定義されている順序にしたがって参照するのと同じことです。標識配列の *n* 番目の変数は、ホスト構造の *n* 番目の変数の標識変数です。

例えば、C で、V1、V2、および V3 が構造 S1 内の変数として宣言されている場合、

```
EXEC SQL FETCH CURSOR1 INTO :S1;
```

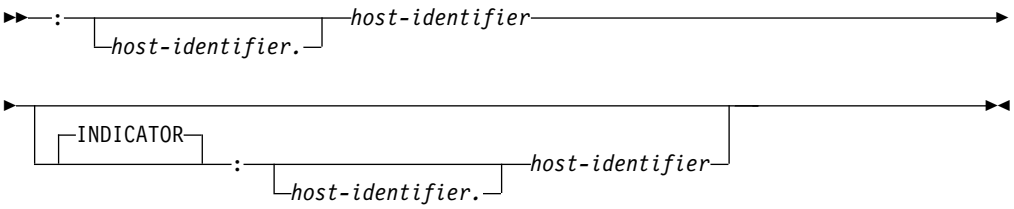
上記のステートメントは、次のステートメントと同等です。

```
EXEC SQL FETCH CURSOR1 INTO :V1, :V2, :V3;
```

ホスト構造に、標識配列より *m* 個 だけ多い変数がある場合、ホスト構造の最後の *m* 個 の変数は、標識変数を持ちません。ホスト構造に、標識配列より *m* 個 だけ少ない変数がある場合、標識配列の最後の *m* 個 の変数は無視されます。上記の規則は、ホスト構造への参照に標識変数が含まれる場合、またはホスト変数への参照に標識配列が含まれる場合にも当てはまります。標識配列または標識変数を指定しない場合、ホスト構造の変数はいずれも標識変数を持たないことになります。

構造参照に加えて、ホスト構造内の個々のホスト変数、または標識配列内の個々の標識変数は、修飾名によって参照することができます。この修飾の形式は、ホスト ID の後にピリオドと他のホスト ID を付けたものです。最初のホスト ID は、ホスト構造を指し、2 番目のホスト ID は、そのホスト構造内のホスト変数を指していなければなりません。

ホスト変数またはホスト構造参照の形式は、以下のようになります。



式の中のホスト変数 は、ホスト変数の宣言に関する規則に従ってプログラムで記述されているホスト変数 (構造ではなく) を識別するものでなければなりません。

次の C サンプルには、ホスト構造、ホスト標識配列、およびホスト変数の参照が示されています。

```
struct { char empno[7];
                struct          { short int firstname_len;
                                  char firstname_text[12];
                                } firstname;
                char midint,
                struct          { short int lastname_len;
                                  char lastname_text[15];
                                } lastname;
                char workdept[4];
       } pemp1;
short ind[14];
short eind
struct { short  ind1;
         short  ind2;
       } indstr;

  .....
strcpy(pemp1.empno,"000220");
.....
EXEC SQL
  SELECT *
    INTO :pemp1:ind
    FROM corpdata.employee
    WHERE empno=:pemp1.empno;
```

上の例では、以下のホスト変数およびホスト構造への参照が有効です。

```
:pemp1   :pemp1.empno   :pemp1.empno:eind   :pemp1.empno:indstr.ind1
```

C、C++、COBOL、PL/I、RPG でホスト構造を参照する方法の詳細については、『組み込み SQL プログラミング』トピック集を参照してください。

ホスト構造配列

PL/I、C++、および C では、ホスト構造配列は、次元属性を持つ構造名です。COBOL の場合は、1 次元の表です。RPG の場合は、オカレンス・データ構造です。 ILE RPG では、ホスト構造配列をキーワード DIM を持つデータ構造にすることもできます。

ホスト構造配列を参照することができるのは、複数行の取り出しを使用する場合の FETCH ステートメント、または複数行挿入を使用する場合の INSERT ステートメントだけです。ホスト構造配列は、ホスト言語のステートメントによって定義されます。これについては、「組み込み SQL プログラミング」トピック集で説明しています。

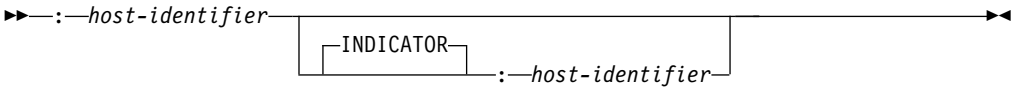
ホスト構造配列の参照の形式は、ホスト変数参照の形式と同じです。:S1:S2 の参照は、S1 がホスト構造配列を指す場合は、ホスト構造配列に対する参照です。S1 がホスト構造を指す場合、S2 は、短整数ホスト変数、短整数ホスト変数の配列、または短整数ホスト変数の 2 次元の配列のいずれかでなければなりません。次の例では、S1 はホスト構造配列であり、S2 はその標識配列です。

```
EXEC  SQL FETCH CURSOR1 FOR 5 ROWS
          INTO :S1:S2;
```

ホスト構造と標識配列の次元は、等しくなければなりません。

ホスト構造に、標識配列より *m* 個 だけ多い変数がある場合、ホスト構造の最後の *m* 個 の変数は、標識変数を持ちません。ホスト構造に、標識配列より *m* 個 だけ少ない変数がある場合、標識配列の最後の *m* 個 の変数は無視されます。標識配列または標識変数の指定がない場合は、ホスト構造配列中の変数には標識変数がありません。

次の図は、ホスト構造の配列に対する参照の構文を示しています。



ホスト構造の配列は、REXX ではサポートされません。

関数

関数 とは、特定の操作に名前、つまり関数名を付けたもので、関数名の後には括弧に囲まれた 1 つ以上のオペランドが続きます。関数は、入力の値のセットと結果の値のセットの間の関係を表しています。関数への入力値は引数 と呼ばれます。例えば、関数に日時のデータ・タイプを持った 2 つの引数を渡し、結果としてタイム・スタンプ・データ・タイプの戻り値を渡すことができます。

関数のタイプ

関数を分類する方法は、いくつかあります。

その 1 つの方法は、組み込み、ユーザー定義、または特殊タイプ用に生成されたユーザー定義関数として分類することです。

- データベース・マネージャーには組み込み関数 が付属しています。 これらの関数は、単一の値の結果を提供します。組み込み関数には、"+" のような演算子関数、AVG のような集約関数、あるいは SUBSTR のようなスカラー関数が含まれています。組み込みの集約およびスカラー関数のリストとこれらの関数についての詳細については、 291 ページの『第 4 章 組み込み関数』を参照してください。

  組み込み関数 はスキーマ QSYS2 の一部です。[39]
- ユーザー定義関数 は、CREATE FUNCTION ステートメントを使用して作成され、カタログ表 QSYS2.SYSROUTINES およびカタログ・ビュー QSYS2.SYSFUNCS でデータベース・マネージャーに登録されます。詳しくは、1070 ページの『CREATE FUNCTION』を参照してください。これらの関数により、ユーザー独自の、あるいはサード・パーティーのベンダーの関数定義を追加することによって、データベース・マネージャーの機能を拡張することができます。

  ユーザー定義関数は、*SQL*、外部、またはソース 関数のいずれかです。SQL 関数は、SQL ステートメントのみを使用して、データベースに定義されます。外部関数は、関数が呼び出されたときに実行される外部プログラムまたはサービス・プログラムへの参照を伴って、データベースに定義されます。ソース関数は、組み込み関数または別のユーザー定義関数への参照を伴って、データベースに定義されます。ソース関数を使用して、特殊タイプで用いる組み込みの集約およびスカラー関数を拡張することができます。

  ユーザー定義関数は、それが作成されたスキーマに常駐します。そのスキーマが、QSYS、QSYS2、QTEMP、SYSIBM、または SYSPROC ということはあり得ません。
- 特殊タイプ用に生成されたユーザー定義関数 とは、 CREATE TYPE ステートメントを使用して特殊タイプが作成されたときに、データベース・マネージャーが自動的に生成する関数です。これらの関数は、特殊タイプからソース・タイプ

39. 組み込み関数 は、データベース・マネージャーによって内部的にインプリメントされており、したがって、関連するプログラムやサービス・プログラム・オブジェクトは、組み込み関数 には存在しません。さらに、カタログには組み込み関数 についての情報は含まれていません。しかしながら、組み込み関数 は、あたかも QSYS2 に存在しているように取り扱うことができ、組み込み関数名は QSYS2 で修飾することができます。

へ、さらにソース・タイプから特殊タイプへのキャストをサポートします。特殊タイプはそれ自体とのみしか互換性がないため、データ・タイプ間のキャストの可能性は重要です。

生成されたキャスト関数は、対象となった特殊タイプと同じスキーマに常駐します。そのスキーマが、QSYS、QSYS2、QTEMP、SYSIBM、または SYSPROC ということはあり得ません。特殊タイプ用に生成される関数の詳細については、1328 ページの『CREATE TYPE (特殊)』を参照してください。

関数を分類するもう 1 つの方法では、入力データの値と結果の値によって、集約関数、スカラー関数、または表関数として分類します。

- 集約関数 は、それぞれの引数ごとに値のセット (列の値など) を受け取り、入力値のセットについて単一の値の結果を戻します。集約関数は、しばしば、列関数と呼ばれます。組み込み関数およびユーザー定義のソース関数は、集約関数になり得ます。
- スカラー関数 は、それぞれの引数ごとに単一の値を受け取り、単一の値の結果を戻します。組み込み関数およびユーザー定義関数は、スカラー関数になり得ます。また、特殊タイプ用に生成されたユーザー定義関数もスカラー関数です。
- 表関数 は、受け取った引数のセットに関する表を戻します。各引数はそれぞれ単一の値です。表関数は、副選択の FROM 文節の中でのみ参照することができます。表関数は、外部関数または SQL 関数として定義できます。ただし、表関数はソース関数となることはできません。

  表関数を使用して、データベースに格納されていないデータに SQL 言語処理能力を適用したり、その種のデータが表に格納されているかのようにアクセスしたりできます。例えば、表関数により、特定のファイルを読み取ったり、Web からデータを取得したり、Lotus Notes® データベースにアクセスして結果表を戻したりすることができます。

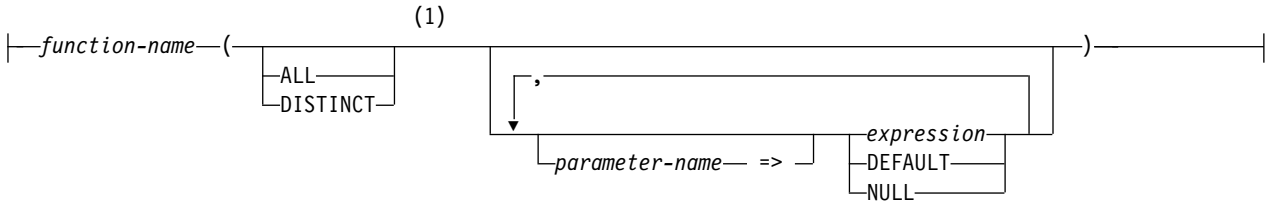
関数呼び出し

スカラー関数または集約関数 (組み込みまたはユーザー定義に関係なく) の各参照は、特定の構文に準拠する必要があります。

構文は次のとおりです。[40]

40. 一部の関数では、式の代わりにキーワードを使用できます。例えば、CHAR 関数では、キーワードのリストを使用して目的の日付形式を指定できます。また、一部の関数では、式のコンマ区切りリストでコンマの代わりにキーワードを使用します。例えば、EXTRACT、TRIM、および POSITION 関数ではキーワードを使用します。

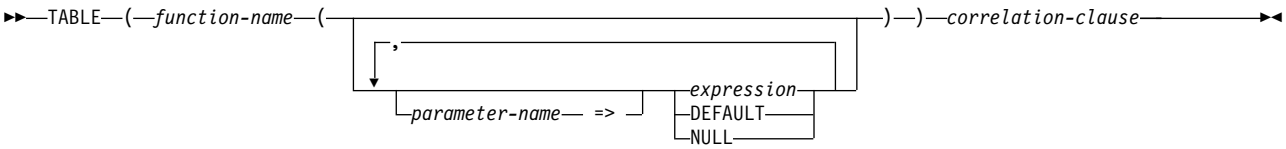
function-invocation:



注:

1 ALL または DISTINCT キーワードは、集約関数または集約関数をソースにしたユーザー定義関数にのみ指定することができます。

表関数に対する各参照は、以下の構文に従います。



上記の構文において、式 はスカラー関数または集約関数の場合と同じです。その他の式 の規則については、 196 ページの『式』を参照してください。

引数とは、呼び出し時に関数に渡される値、または DEFAULT で指定されるデフォルト値です。 SQL の中で呼び出されるとき、関数にはゼロ個以上の引数のリストが渡されます。このような引数は、引数のセマンティクスが引数リスト内の位置によって決定されるという意味で定位置と言えます。パラメーターは、関数への入力の形式上の定義です。データベースに対して内部的に (組み込み関数) またはユーザーによって (ユーザー定義関数) 関数が定義されるときは、関数のパラメーターが (ゼロ個以上) 指定されます。また、パラメーターの定義の順序によって、パラメーターの位置とセマンティクスが定義されることになります。したがって、どのパラメーターも関数の特定の定位置入力です。

呼び出し時に、定位置構文かまたは名前付き構文を使用して引数がパラメーターに代入されます。定位置構文を使用する場合、引数は引数リストにおける位置によって特定のパラメーターに対応します。名前付き構文を使用する場合、引数はパラメーターの名前に基づいて特定のパラメーターに対応します。名前付き構文を使用して、ある引数がパラメーターに代入される場合、それに続く引数もすべて名前付き構文を使用して代入される必要があります 。名前付き引数の名前は、関数呼び出し 1 つにつき一度だけ使用できます。関数呼び出しの引数のデータ・タイプが、選択した関数のパラメーターのデータ・タイプと一致しない場合には、列への値の代入と同じ規則が適用され、引数は実行時にパラメーターのデータ・タイプに変換されます。これには、引数とパラメーターの間で精度、位取り、または長さが異なる場合も含まれます。関数呼び出しの引数が DEFAULT の指定である場合は、引数に使用される実際の値は、関数定義で対応パラメーターのデフォルトに指定された値になります。パラメーターのデフォルト値が定義されていない場合は、NULL 値が使用されます。型なし式 (パラメーター・マーカー、NULL キーワード、または

DEFAULT キーワード) が引数として使用された場合、引数に関連付けられるデータ・タイプは、選択した関数のパラメーターのデータ・タイプによって決まります。

インラインで記述されている SQL 関数と外部関数の場合、引数リストおよびデフォルト式の中にある、日付、時刻、またはタイム・スタンプの特殊レジスター値のすべての参照は 1 つのクロック読み取りを使用します。インラインで記述されていない SQL 関数の場合、引数リストの中にある、日付、時刻、またはタイム・スタンプの特殊レジスター値の参照は、すべてのデフォルト式に対して 1 つのクロック読み取りを使用し、明示的引数内にあるすべての参照には別のクロック読み取りを使用します。

関数が呼び出されると、その各パラメーターの値は、記憶域割り当てを使用して、関数の対応するパラメーターに割り当てられます。制御は、ホスト言語の呼び出し規則に従って、外部関数に渡されます。ユーザー定義の集約またはスカラー関数の実行が完了すると、関数の結果が、記憶域割り当てを使用して、結果のデータ・タイプに渡されます。割り当て規則の詳細については、 113 ページの『割り当ておよび比較』を参照してください。

表関数は、副選択の FROM 文節の中でのみ参照することができます。表関数を参照する方法についての詳細は、 794 ページの『table-reference』の FROM 文節に関する説明を参照してください。

関数解決

関数はその関数名によって呼び出されます。関数名は、暗黙的にまたは明示的にスキーマ名で修飾され、その後に括弧で囲まれた関数への引数が続きます。

データベース内では、それぞれの関数はその関数のシグニチャーによって一意的に識別されます。シグニチャーとは、そのスキーマ名、関数名、パラメーター数、およびパラメーターのデータ・タイプのことです。このため、スキーマでは、関数名が同じでも、それぞれパラメーター数が異なるか、パラメーター・データ・タイプが異なるため、複数の関数を持つことができます。あるいは、名前やパラメーター数、パラメーターのタイプが同じ関数でも、別々のスキーマには存在することができます。

- 関数の多重定義**:** 同じスキーマに同じ数のパラメーターを持つ複数の関数インスタンスのある関数名は、多重定義関数と呼ばれます。
- 関数のオーバーライド**:** 関数は SQL パス内の複数のスキーマにおいてもオーバーライド可能です。その場合、その SQL パス内の異なるスキーマに、同じ数のパラメーターを持つその名前の付いた関数が 2 つ以上あることになります。必ずしもパラメーター・データ・タイプがそれぞれ異なっている必要はありません。

関数が呼び出されると、データベース・マネージャーは実行すべき関数を決定することが必要になります。このプロセスを、関数解決 と呼びます。

一群の候補関数の判別**:**

- 関数呼び出しの引数の数を A とします。
- 関数シグニチャー中のパラメーターの数を P とします。

- 関数シグニチャーでデフォルトが定義されていないパラメーターの数を N とします。

関数呼び出しを解決するための候補関数は、以下の基準で選択されます。

- 各候補関数は、一致する名前と適用可能なパラメーター数を持つ。適用可能なパラメーター数とは、N ≤ A ≤ P という条件を満たすパラメーター数のことです。
- 候補関数のパラメーターのうち、対応引数が関数呼び出しで位置でも名前でも指定されていない各パラメーターは、デフォルトを使用して定義される。
- 1 つ以上のスキーマの集合に含まれる各候補関数は、関数を呼び出しているステートメントの許可 ID に関連付けられた EXECUTE 特権を持つ。

関数解決は、関数を修飾関数名を指定して呼び出した場合と非修飾関数名を指定して呼び出した場合で同じです。ただし、非修飾名を指定した場合、データベース・マネージャーは複数のスキーマを検索する必要があります。

- 修飾関数解決: 関数が関数名とスキーマ名を使用して呼び出されると、データベース・マネージャーは指定されたスキーマのみを検索して候補関数を探します。

  スキーマ内で候補関数が見つからない場合、エラーが戻されます。関数が選択された場合、関数が正常に実行されるかどうかは、その関数が呼び出されたコンテキストで、戻される結果が有効であるかどうかによって左右されます。例えば、関数が文字データ・タイプでなければならないところに整数データ・タイプを戻したり、表が許可されていないのに表を戻したりすると、エラーが返されます。
- 修飾されない関数解決: 関数が関数名だけで呼び出されると、データベース・マネージャーは実行する関数インスタンスを決めるために、複数のスキーマを検索する必要があります。SQL パスは、検索するスキーマのリストを持っています。SQL パス (72 ページの『SQL パス』を参照) 内の各スキーマごとに、データベース・マネージャーは候補となる関数を選択します。

  パス内のどのスキーマ内にも候補関数がない場合、エラーが戻されます。関数が選択された場合、関数が正常に実行されるかどうかは、その関数が呼び出されたコンテキストで、戻される結果が有効であるかどうかによって左右されます。例えば、関数が文字データ・タイプでなければならないところに整数データ・タイプを戻したり、表が許可されていないのに表を戻したりすると、エラーが返されます。

データベース・マネージャーは、候補の関数を識別した後で、実行する関数インスタンスとして最適のものを選択します (187 ページの『最適の判別』を参照してください)。最適 (関数シグニチャーがスキーマ名を除いて同一) の関数インスタンスが、複数のスキーマにある場合は、データベース・マネージャーは SQL パスで一番早いスキーマの関数を選択します。

関数解決は、組み込み関数を含むすべての関数に適用されます。組み込み関数は、スキーマ QSYS2 に論理的に存在します。スキーマ QSYS2 が SQL パスで明示的に指定されていない場合は、スキーマは暗黙的にそのパスの前にあるものと見なされます。非修飾関数名を指定する場合は、対象の関数が選択されるようにするために、各スキーマを正しい検索順序で並べたリストを SQL パスとして設定しなければなりません。

CREATE VIEW または CREATE TABLE ステートメントでは、ビューまたはマテリアライズ照会表の作成時に関数解決が実行されます。その後で同じ名前の他の関数が作成された場合でも、そのビューまたはマテリアライズ照会表は影響を受けません。そのビューまたはマテリアライズ表の作成時に選択された関数よりもその新しい関数のほうが適しているとしても、その動作に変わりはありません。CREATE FUNCTION、 CREATE PROCEDURE、CREATE TRIGGER、CREATE VARIABLE、CREATE MASK、または CREATE PERMISSION ステートメントでは、関数、プロシージャー、トリガー、変数、マスク、または許可が作成されるときに、関数解決が実行されます。選択された関数のスキーマが、トリガー、変数、マスク、および許可で保存されます。また、デフォルト式の関数およびプロシージャーでも保存されます。後で同じ名前の別の関数が作成された場合、関数、プロシージャー、トリガー、変数、マスク、許可、またはルーチン・デフォルトに影響が及ぶのは、オブジェクトが作成された時点で選択された関数よりも新しい関数のほうがより適している場合のみです。

最適の判別

実行の候補となるような同じ名前の関数が、複数存在する場合があります。その場合、候補関数のセットに含まれる各関数のパラメーターのデータ・タイプ、SQL パスでのスキーマの位置、パラメーターの総数を使用して、関数が最適要件を満たすかどうかが判別されます。関数の結果のデータ・タイプ、または現在考慮中の関数のタイプ (集約、スカラー、または表) もこの決定では考慮されないことに注意してください。

候補関数のセットに複数の関数が含まれており、関数呼び出しで名前付き引数が使用されている場合、名前付き引数に対応するパラメーターの順序位置は、すべての候補関数で同じでなければなりません。

パラメーター・セット という用語は、候補関数のセットの (上記のようなパラメーターが存在する) パラメーター・リストで同じ位置にあるすべてのパラメーターを表します。パラメーターの対応引数は、関数呼び出しで引数がどのように指定されているかに基づいて判別されます。定位置引数の場合、パラメーターの対応引数は、関数呼び出しにおいて候補関数のパラメーター・リストでのパラメーターと同じ位置にある引数です。名前付き引数の場合、パラメーターの対応引数は、パラメーターと同じ名前の引数です。この場合、関数呼び出しにおける引数の順序は、最適を判別する際に考慮されません。候補関数のパラメーター数が関数呼び出しでの引数の数よりも多い場合、対応引数のない各パラメーターは、あたかも DEFAULT キーワードを値として持つ対応引数があるかのように処理されます。

最適である関数を判別するために、以下のステップが使用されます。

- ステップ 1: 型付き式である引数の考慮

  データベース・マネージャーは、各パラメーターのデータ・タイプと対応引数のデータ・タイプを比較することによって、呼び出しの最適要件を満たす関数、また関数のセットを判別します。

  パラメーターのデータ・タイプがその対応引数のデータ・タイプと同じかどうかは、以下のように判別されます。

  – データ・タイプのシノニムが一致する場合は、同じであると見なされます。例えば、DOUBLE と FLOAT は同じであると見なされます。

– データ・タイプの属性 (長さ、精度、位取り、CCSID など) は無視されます。したがって、CHAR(8) と CHAR(35) は同じであると見なされ、同様に、 DECIMAL(11,2) と DECIMAL(4,3) は同じであると見なされます。
– 文字タイプとグラフィック・タイプは同じであると見なされます。例えば、以下は同じタイプであると見なされます。 CHAR と GRAPHIC、VARCHAR と VARGRAPHIC、および CLOB と DBCLOB。CHAR(13) と GRAPHIC(8) は同じタイプであると見なされます。

関数呼び出しにおける型なし式でない各引数のデータ・タイプが、関数インスタンスの対応パラメーターのデータ・タイプと一致する関数か、またはそのデータ・タイプにプロモート可能である関数のみを考慮することにより、候補関数のサブセットが取得されます。関数呼び出しの引数が型なし式である場合、対応パラメーターのデータ・タイプは、どのデータ・タイプでも構いません。 107 ページの『データ・タイプのプロモーション』 にあるデータ・タイプのプロモーションの優先順位リストは、各データ・タイプに適合する (プロモーションを考慮) データ・タイプを、適合度が最高から最低の順にリストしています。このサブセットが空でない場合、この候補関数のサブセットについてプロモート可能なプロセスを使用して最適が決定されます。このサブセットが空である場合、候補関数のオリジナルのセットについてキャスト可能なプロセスを使用して最適が決定されます。

プロモート可能プロセス
このプロセスは、関数呼び出しの引数が、関数定義の対応するパラメーターのデータ・タイプと一致するか、またはそのデータ・タイプにプロモートできるかどうかを考慮した場合にのみ、最適を決定します。候補関数のサブセットの場合、パラメーター・リストが左から右へと処理されていきます。つまり、候補関数のサブセットの最初の位置のパラメーター・セットをまず処理してから、2 番目の位置のパラメーター・セットに進むというように処理されます。候補関数のサブセットから候補関数を除去するために、以下のステップが使用されます (プロモーションのみを考慮)。

1. ある候補関数のパラメーターの対応引数のデータ・タイプが、(プロモーションのみを考慮した場合に) 他の候補関数よりもパラメーターのデータ・タイプに適合する場合、関数呼び出しに対する適合度が同等でない候補関数は除去されます。 107 ページの『データ・タイプのプロモーション』 にあるデータ・タイプのプロモーションの優先順位リストは、各データ・タイプに適合する (プロモーションを考慮) データ・タイプを、適合度が最高から最低の順にリストしています。
2. 対応引数のデータ・タイプが型なし式である場合、候補関数は除去されません。
3. これらのステップは、残りの候補関数の次のパラメーター・セットに対しても行われ、パラメーター・セットがそれ以上なくなるまで繰り返されます。

キャスト可能なプロセス
このプロセスではまず、関数呼び出しにおける対応引数のデータ・タイプが、関数定義のパラメーターのデータ・タイプに一致するかどうか、またはそのデータ・タイプにプロモートできるかどうかをパラメーター

ごとに調べて、最適が判別されます。次にデータベース・マネージャーは、プロモート可能だったデータ・タイプを持つ対応引数がない各パラメーター・セットについて、関数解決のために対応引数のデータ・タイプをパラメーターのデータ・タイプに暗黙的にキャストできるかどうかを、パラメーターごとに調べます。

候補関数のセットの場合、パラメーター・リストに含まれるパラメーターが左から右へと処理されていきます。つまり、すべての候補関数の最初の位置のパラメーター・セットをまず処理してから、2 番目の位置のパラメーター・セットに進むというように処理されます。候補関数のセットから候補関数を除去するために、以下のステップが使用されます (プロモーションのみを考慮)。

1. ある候補関数のパラメーターの対応引数のデータ・タイプが、(プロモーションのみを考慮した場合に) 他の候補関数よりもパラメーターのデータ・タイプに適合する場合、関数呼び出しに対する適合度が同等でない候補関数は除去されます。 107 ページの『データ・タイプのプロモーション』 にあるデータ・タイプのプロモーションの優先順位リストは、各データ・タイプに適合する (プロモーションを考慮) データ・タイプを、適合度が最高から最低の順にリストしています。
2. 対応引数のデータ・タイプをいずれの候補関数のパラメーターのデータ・タイプにもプロモートできない場合 (対応引数が型なし式である場合を含む)、候補関数は除去されません。
3. これらのステップは、残りの候補関数の次のパラメーター・セットに対しても行われ、パラメーター・セットがそれ以上なくなるまで繰り返されます。

(プロモーションのみを考慮した場合に) 適合する対応引数が少なくとも 1 つのパラメーター・セットになく、パラメーター・セットの対応引数がデータ・タイプを持つ場合、データベース・マネージャーは、こうしたパラメーター・セットを 1 つずつ左から右へと比較していきます。候補関数のセットから候補関数を除去するために、以下のステップが使用されます (暗黙的キャストを考慮)。

1. 残りのすべての候補関数のパラメーター・セットのすべてのデータ・タイプが、 107 ページの『データ・タイプのプロモーション』に指定されているように同じデータ・タイプ優先順位リストに属していない場合、エラーが戻されます。
2. 対応する引数のデータ・タイプを、 190 ページの『関数解決のための暗黙的キャスト』 に指定されているようにパラメーターのデータ・タイプに暗黙的にキャストできない場合、エラーが戻されます。
3. ある候補関数のパラメーターの対応引数のデータ・タイプが、(暗黙的キャストを考慮した場合に) 他の候補関数よりもパラメーターのデータ・タイプに適合する場合、関数呼び出しに対する適合度が同等でない候補関数は除去されます。 190 ページの『関数解決のための暗黙的キャスト』にあるデータ・タイプ・リストは、(暗黙的キャストを考慮した場合に) より適合するデータ・タイプを示しています。
4. これらのステップは、(プロモーションのみを考慮した場合に) 適合する対応引数がなく、対応引数がデータ・タイプを持つ次のパラメー

ター・セットに対しても行われ、こうしたパラメーター・セットがなくなるかエラーが発生するまで繰り返されます。

- ステップ 2: SQL パスの考慮

  複数の候補関数が残っている場合、データベース・マネージャーは、SQL パスの最初にスキーマがある候補関数を選択します。

- ステップ 3: 関数呼び出しにおける引数の数の考慮

  複数の候補関数が残っており、ある候補関数のパラメーター数が他の候補関数のパラメーター数以下である場合、パラメーター数の多い候補関数は除去されます。

- ステップ 4: 型なし式である引数の考慮

  複数の候補関数が残っており、少なくとも 1 つのパラメーター・セットの対応引数が型なし式である場合、データベース・マネージャーは、こうしたパラメーター・セットを左から右へと比較していきます。候補関数のセットから候補関数を除去するために、以下のステップが使用されます。

  1. 残りのすべての候補関数のパラメーター・セットのすべてのデータ・タイプが、 107 ページの『データ・タイプのプロモーション』に指定されているように同じデータ・タイプ優先順位リストに属していない場合、エラーが戻されます。
  2. ある候補関数のパラメーターのデータ・タイプが、暗黙的キャストのデータ・タイプ順序付けで他の候補関数よりも左にある場合、パラメーターのデータ・タイプがデータ・タイプ順序付けで右にある候補関数は除去されます。 『関数解決のための暗黙的キャスト』にあるデータ・タイプ・リストに、暗黙的キャストのデータ・タイプの順序付けが示されています。

  複数の候補関数がまだ存在する場合は、エラーが戻されます。

関数解決のための暗黙的キャスト

ユーザー定義タイプまたは XML、 ROWID、または DATALINK データ・タイプを持つ引数については、関数解決の暗黙的キャストはサポートされません。組み込み関数またはユーザー定義 cast 関数の場合にもサポートされません。サポートされるのは、以下の場合です。

- DATE、TIME、および TIMESTAMP 以外の組み込みタイプから数値データ・タイプへのサポートされる任意のキャスト。 110 ページの表 16を参照してください。
- 型なし引数をどのデータ・タイプにもキャストできる。

プロモーション用のデータ・タイプ優先順位リストと同様に、暗黙的キャストの場合も関連したデータ・タイプのグループ内のデータ・タイプに対する順序があります。この順序は、暗黙的キャストを考慮する関数解決の実行時に使用されます。 191 ページの表 27 は、関数解決のための暗黙的キャストに使用するデータ・タイプの順序を示しています。データ・タイプは優先順位の高いものから順にリストされます (これは、プロモーション用のデータ・タイプ優先順位リストの順序とは異なることに注意してください)。

表 27. 関数解決のための暗黙的キャストに使用するデータ・タイプの順序

データ・タイプ・グループ	関数解決のための暗黙的キャストに使用するデータ・タイプ・リスト **(**優先順位の高いものから順に**)**
数値データ・タイプ	DECFLOAT、 double、 real、 decimal、 BIGINT、 INTEGER、 SMALLINT
文字データ・タイプおよび GRAPHIC ストリング・データ・タイプ	VARCHAR または VARGRAPHIC、CHAR または GRAPHIC、CLOB または DBCLOB
バイナリー・データ・タイプ	VARBINARY、BINARY、BLOB、 VARCHAR FOR BIT DATA、 CHAR FOR BIT DATA
日時データ・タイプ	TIMESTAMP、DATE、TIME

注**:**

小文字で示したタイプの定義は、以下のとおりです。

10 進数
= DECIMAL(p,s) または NUMERIC(p,s)

real = REAL または FLOAT(*n*)。この場合の *n* は短精度浮動小数点数の指定

double
= DOUBLE、DOUBLE PRECISION、FLOAT または FLOAT(*n*)。この場合の *n* は倍精度浮動小数点数の指定

例

以下に、関数解決の例を示します。 (必要なキーワードがすべて示されているわけではないことに注意してください。)

例 1: 以下に、関数解決での SQL パスの考慮事項を示す例があります。この例では、3 つの異なるスキーマに 8 つの ACT 関数があり、以下のように登録されています。

```
CREATE FUNCTION AUGUSTUS.ACT (CHAR(5), INT, DOUBLE) SPECIFIC ACT_1 ...
CREATE FUNCTION AUGUSTUS.ACT (INT, INT, DOUBLE) SPECIFIC ACT_2 ...
CREATE FUNCTION AUGUSTUS.ACT (INT, INT, DOUBLE, INT) SPECIFIC ACT_3 ...
CREATE FUNCTION JULIUS.ACT (INT, DOUBLE, DOUBLE) SPECIFIC ACT_4 ...
CREATE FUNCTION JULIUS.ACT (INT, INT, DOUBLE) SPECIFIC ACT_5 ...
CREATE FUNCTION JULIUS.ACT (SMALLINT, INT, DOUBLE) SPECIFIC ACT_6 ...
CREATE FUNCTION JULIUS.ACT (INT, INT, DECFLOAT) SPECIFIC ACT_7 ...
CREATE FUNCTION NERO.ACT (INT, INT, DEC(7,2)) SPECIFIC ACT_8 ...
```

以下のように関数が参照されるとします (I1 および I2 は INTEGER 列、 D は DECIMAL 列です)。

```
SELECT ... ACT(I1, I2, D) ...
```

この参照を行うアプリケーションの SQL パスが次のようになっているとします。

```
"JULIUS","AUGUSTUS","CAESAR"
```

関数解決の規則は次のとおりです。

- スキーマ NERO が SQL パスに組み込まれていないため、特定の名前 ACT_8 の付いた関数は候補から除かれます。
- パラメーターの数が違っているため、ACT_3 は候補から除かれます。第 1 引数が第 1 パラメーターのデータ・タイプにプロモートできないため、 ACT_1 と ACT_6 はどちらも候補から除かれます。
- この時点で複数の候補が残っているため、次に引数が順に検討されます。
  - 最初の引数については、残りのすべての関数 ACT_2、ACT_4、ACT_5、および ACT_7 がその引数タイプと完全に一致します。この検討ではどの関数も検討の対象から除かれないため、次の引数を検討する必要があります。
  - 2 番目の引数では、ACT_2、ACT_5、および ACT_7 が完全に一致していますが、 ACT_4 は一致していないため、ACT_4 が検討の対象から除かれます。 ACT_2、ACT_5、および ACT_7 の間の何らかの差異を判別するために、さらに次の引数が検討されます。
  - 第 3 の最後の引数では、ACT_2、ACT_5、ACT_7 のいずれも、引数のタイプと完全には一致していません。 ACT_2 と ACT_5 の適合度は同程度ですが、ACT_7 は他の 2 つよりも適合度が劣ります。タイプ DOUBLE は DECFLOAT よりも、DECIMAL に近いからです。 ACT_7 は除去されます。
- この時点で、パラメーター・シグニチャーが同じである関数として ACT_2 と ACT_5 の 2 つが残っています。最終的な決定要因は、どちらの関数のスキーマが SQL パスで先に出現するかであり、この基準によって ACT_5 が最終的に選択されます。

例 *2:* これは、関数解決がエラーになるというシチュエーションの例です。エラーになる原因は、同程度に呼び出しに適合する候補関数が複数あるが、引数の 1 つの対応パラメーターが同じタイプ優先順位リストに属していないことです。

この例では、以下のように定義された単一のスキーマに 3 つの関数のみ含まれています。

```
CREATE FUNCTION CAESAR.ACT (INT, VARCHAR(5), VARCHAR(5)) SPECIFIC ACT_1 ...
CREATE FUNCTION CAESAR.ACT (INT, INT, DATE) SPECIFIC ACT_2 ...
CREATE FUNCTION CAESAR.ACT (INT, INT, DOUBLE) SPECIFIC ACT_3 ...
```

以下のように関数が参照されるとします (I1 および I2 は INTEGER 列、VC は VARCHAR 列です)。

```
SELECT ... ACT(I1, I2, VC) ...
```

この参照を行うアプリケーションの SQL パスが次のようになっているとします。

```
"CAESAR"
```

関数解決の規則は次のとおりです。

- 関数呼び出しの各入力引数のデータ・タイプが、関数インスタンスの対応するパラメーターのデータ・タイプと一致するか、またはそのデータ・タイプにプロモート可能かどうかを判別するために、それぞれの候補関数が評価されます。
  - 最初の引数については、候補となるすべての関数にこのパラメーター・タイプと完全に一致するデータ・タイプが含まれます。
  - 2 番目の引数については、INTEGER を VARCHAR にプロモートできないため、ACT_1 は除去されます。

  - 3 番目の引数については、VARCHAR を DATE または DOUBLE にプロモートできないため、候補関数が残らないように、ACT_2 と ACT_3 の両方が除去されます。
- 上からの候補関数のサブセットが空のため、候補関数はキャスト可能なプロセスを使用して考慮されます。
  - 最初の引数については、候補となるすべての関数にこのパラメーター・タイプと完全に一致するデータ・タイプが含まれます。
  - 2 番目の引数については、INTEGER を VARCHAR にプロモートできないため、ACT_1 は除去されます。 ACT_2 と ACT_3 が候補として適しています。
  - 3 番目の引数については、ACT_2 および ACT_3 の対応するパラメーターのデータ・タイプが同じデータ・タイプの優先順位リストに属していないため、エラーが戻されます。

例 3: この例では、キャスト可能なプロセスを使用して関数解決が成功する状況を示しています。

この例では、以下のように定義された単一のスキーマに 3 つの関数のみ含まれています。

```
CREATE FUNCTION CAESAR.ACT (INT, VARCHAR(5), VARCHAR(5)) SPECIFIC ACT_1 ...
CREATE FUNCTION CAESAR.ACT (INT, INT, DECFLOAT) SPECIFIC ACT_2 ...
CREATE FUNCTION CAESAR.ACT (INT, INT, DOUBLE) SPECIFIC ACT_3 ...
```

以下のように関数が参照されるとします (I1 および I2 は INTEGER 列、VC は VARCHAR 列です)。

```
SELECT ... ACT(I1, I2, VC) ...
```

この参照を行うアプリケーションの SQL パスが次のようになっているとします。

```
"CAESAR"
```

関数解決の規則は次のとおりです。

- 関数呼び出しの各入力引数のデータ・タイプが、関数インスタンスの対応するパラメーターのデータ・タイプと一致するか、またはそのデータ・タイプにプロモート可能かどうかを判別するために、それぞれの候補関数が評価されます。
  - 最初の引数については、候補となるすべての関数にこのパラメーター・タイプと完全に一致するデータ・タイプが含まれます。
  - 2 番目の引数については、INTEGER を VARCHAR にプロモートできないため、ACT_1 は除去されます。
  - 3 番目の引数については、VARCHAR を DECFLOAT または DOUBLE にプロモートできないため、候補関数が残らないように、ACT_2 と ACT_3 の両方が除去されます。
- 上からの候補関数のサブセットが空のため、候補関数はキャスト可能なプロセスを使用して考慮されます。
  - 最初の引数については、候補となるすべての関数にこのパラメーター・タイプと完全に一致するデータ・タイプが含まれます。

- 2 番目の引数については、INTEGER を VARCHAR にプロモートできないため、ACT_1 は除去されます。 ACT_2 と ACT_3 が候補として適しています。
- 3 番目の引数については、DECFLOAT と DOUBLE の両方が同じデータ・タイプの優先順位リストにあり、VARCHAR は DECFLOAT と DOUBLE の両方を暗黙のうちにキャストできます。DECFLOAT は暗黙的キャストに適しているため、 ACT_2 が最適です。

例 *4:* この例では、キャスト可能なプロセスを使用した関数解決時に、後の引数のプロモーションが暗黙的キャストより優先することを示しています。

この例では、以下のように定義された単一のスキーマに 3 つの関数のみ含まれています。

```
CREATE FUNCTION CAESAR.ACT (INT, INT, VARCHAR(5)) SPECIFIC ACT_1 ...
CREATE FUNCTION CAESAR.ACT (INT, INT, DECFLOAT) SPECIFIC ACT_2 ...
CREATE FUNCTION CAESAR.ACT (INT, INT, DOUBLE) SPECIFIC ACT_3 ...
```

以下のように関数が参照されるとします (I1 は INTEGER 列、VC1 は VARCHAR 列、および C1 は CHAR 列です)。

```
SELECT ... ACT(I1, VC1, C1) ...
```

この参照を行うアプリケーションの SQL パスが次のようになっているとします。

```
"CAESAR"
```

関数解決の規則は次のとおりです。

- 関数呼び出しの各入力引数のデータ・タイプが、関数インスタンスの対応するパラメーターのデータ・タイプと一致するか、またはそのデータ・タイプにプロモート可能かどうかを判別するために、それぞれの候補関数が評価されます。
  - 最初の引数については、候補となるすべての関数にこのパラメーター・タイプと完全に一致するデータ・タイプが含まれます。
  - 2 番目の引数については、VARCHAR を INTEGER にプロモートできないため、候補関数が残らないように、すべての候補関数が除去されます。
- 上からの候補関数のサブセットが空のため、候補関数はキャスト可能なプロセスを使用して考慮されます。
  - 最初の引数については、候補となるすべての関数にこのパラメーター・タイプと完全に一致するデータ・タイプが含まれます。
  - 2 番目の引数については、候補関数のどれにも対応する引数をプロモートできるパラメーターがないため、候補関数は除去されません。
  - 3 番目の引数は ACT_1 のパラメーターにプロモートできますが、ACT_2 または ACT_3 のパラメーターにはプロモートできないため、ACT_1 が最適です。

最適に関する考慮事項

いったん関数が選択されても、まだ、関数の使用が許可されない理由が考えられます。それぞれの関数は、特定のデータ・タイプの結果を戻すように定義されています。結果のデータ・タイプが、関数の呼び出される文脈内で互換性がない場合には、エラーが起こることになります。

例えば、次のように、STEP という名前の 2 つの関数が、結果として別のデータ・タイプで定義されていたとします。

```
STEP(SMALLINT) RETURNS DATE)
STEP(DOUBLE) RETURNS INTEGER
```

そして、次の関数参照がありました (ここで、S は SMALLINT 列)。

```
SELECT ... 3 +STEP(S)
```

次に、引数のタイプが完全に一致するため、最初の STEP が選択されます。加算演算子の引数で要求される数値タイプの代わりに、結果のタイプが DATE であるため、このステートメントでエラーが起こることになります。

関数呼び出しの引数が選択された関数のパラメーターのデータ・タイプに完全に一致しない場合は、列への割り当て (113 ページの『割り当ておよび比較』を参照してください) と同じ規則を使って、引数は実行時にパラメーターのデータ・タイプに変換されます。これには、精度、位取り、長さ、または CCSID が引数とパラメーター間で異なっているケースも含まれます。

以下の例でも、エラーが起きます。

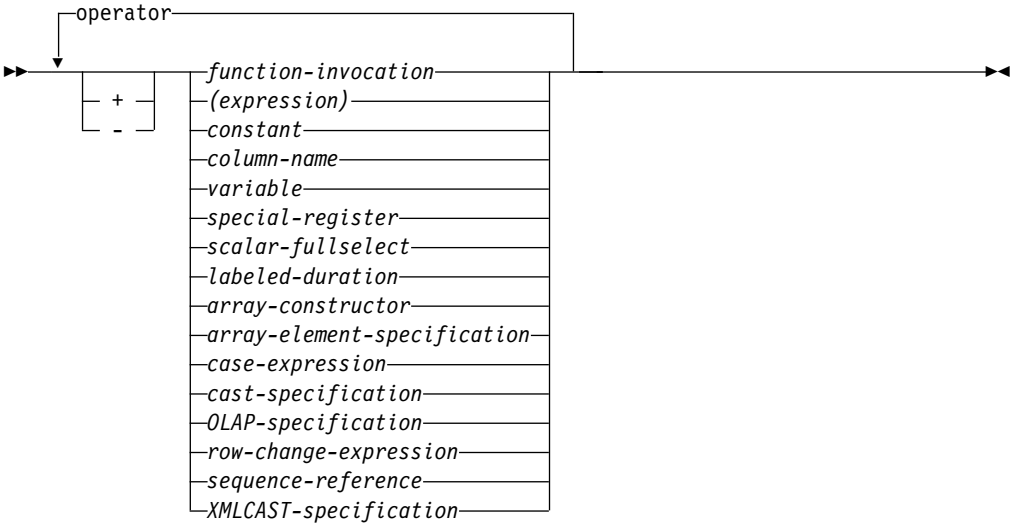
- 関数は FROM 文節の TABLE 文節で参照されているが、関数解決ステップで選択された関数はスカラー関数または集約関数である。
- SQL ステートメントで参照されている関数にはスカラー関数または集約関数が必要であるが、関数解決ステップで選択された関数は表関数である。

式

式では、値を指定します。

権限: *scalar-fullselect*、*sequence-reference*、*function-invocation*、*cast-specification* など、一部の式を使用するには、それぞれに該当する権限が必要になる場合があります。その種の式の場合、ステートメントの権限 ID の特権には、以下の権限が含まれている必要があります。

- グローバル変数。許可の考慮事項について、 170 ページの『グローバル変数』を参照してください。
- *scalar-fullselect*。許可の考慮事項について、 787 ページの『第 6 章 照会』を参照してください。
- *sequence-reference*。シーケンスを参照する権限。許可の考慮事項について、 238 ページの『注』を参照してください。
- *function-invocation*。ユーザー定義関数を実行する権限。許可の考慮事項について、 183 ページの『関数呼び出し』を参照してください。
- *array-element-specification*。配列タイプを参照する権限。許可の考慮事項について、 1542 ページの『GRANT (タイプ特権)』を参照してください。
- *cast-specification*。ユーザー定義タイプを参照する権限。許可の考慮事項について、 218 ページの『CAST の指定』を参照してください。



operator:

演算子を使用しない式

演算子を使用しない式では、指定された値が式の結果になります。

例

```
SALARY     :SALARY     'SALARY'     MAX(SALARY)
```

算術演算子を使用する式

算術演算子を使用すると、その演算子をオペランドの値に適用して得られた数値が式の結果となります。

NULL になる可能性があるオペランドを使用すると、結果も NULL になる可能性があります。どちらか一方のオペランドが NULL 値の場合、式の結果は NULL 値になります。

算術演算子の一方のオペランドが数値である場合には、もう一方のオペランドをストリングにすることができます。このストリングはまず数値オペランドのデータ・タイプに変換されるので、このストリングには数を表す有効なストリング表記が含まれている必要があります。

接頭演算子として + を使用しても、オペランドは変更されません (これを単項プラス と呼びます)。接頭演算子として - を使用すると、ゼロ以外の非 10 進浮動小数点オペランドの符号が反転します (これを単項マイナス と呼びます)。接頭演算子として - (単項マイナス) を使用すると、ゼロおよび特殊値 (すなわち、シグナリングおよび非シグナリング NaNs およびプラスおよびマイナスの無限大) を含むすべての 10 進浮動小数点オペランドの符号が反転します。A のデータ・タイプが短整数の場合、-A のデータ・タイプは長整数になります。接頭演算子の後に続くトークンの最初の文字は、正符号や負符号であってはなりません。

挿入演算子 の +、-、*、/、および ** は、それぞれ加算、減算、乗算、除算、および指数演算を示します。除算の第 2 オペランドの値はゼロであってはなりませんが、10 進浮動小数点演算を使用して計算が実行され、第 1 オペランドが infinity または -infinity の場合は例外です。

COBOL では、負符号 (-) の前後にブランクを入れて、 COBOL ホスト変数名 (ダッシュが使用できる) と混同しないようにすることが必要です。

指数演算子 (**) の結果は、倍精度浮動小数点数になります。その他の演算子の結果は、オペランドのタイプによって決まります。

NUMERIC データ・タイプのオペランドは、算術演算の実行前に DECIMAL オペランドに変換されます。

2 つの整数オペランド

算術演算子のオペランドが両方とも整数で、それぞれの位取りがゼロであれば、いずれかの (あるいは両方の) オペランドが 64 ビット整数でない限り、2 進数の演算が実行され、結果は長整数となります。いずれかの (あるいは両方の) オペランドが 64 ビット整数の場合は、結果は 64 ビット整数となります。この場合は、除算で剰余があっても、その剰余は失われます。整数の算術演算 (単項減算 を含む) の結果は、長整数または 64 ビット整数の範囲内になければなりません。どちらかの整数

オペランドがゼロ以外の位取りを持つ場合は、そのオペランドが同一の精度および位取りの 10 進数オペランドに変換されます。

整数と 10 進数オペランド

一方のオペランドが位取りゼロの整数で、もう一方が 10 進数の場合は、整数の一時的なコピー (以下の精度を持つ 10 進数 (位取りは 0) に変換したもの) を使用して、10 進数の演算が実行されます。

これを以下の表で定義します。

オペランド	**10** 進数のコピーの精度
列または変数 : 64 ビット整数	19
列または変数 : 長整数	11
列または変数 : 短整数	5
定数 (先行ゼロを含む)	定数の桁数と同じ。

一方のオペランドが位取りがゼロ以外の整数の場合、まず、そのオペランドが同一の精度および位取りを持つ 10 進数オペランドに変換されます。

2 つの 10 進数オペランド

オペランドが両方とも 10 進数の場合は、10 進数の演算が実行されます。10 進算術演算の結果は、必ず 10 進数になります。結果の精度および位取りは、実行された演算とオペランドの精度および位取りによって決まります。加算または減算を実行するときに、2 つのオペランドの位取りが異なっている場合は、一方のオペランドの一時的なコピーを使用して演算が実行されます。この一時的なコピーは、2 つのオペランドの小数部分の桁数が同じになるように、位取りが小さい方のオペランドに後書きゼロを付加して、小数部分の桁数を増やしたものです。

特に指定のない限り、10 進数を使用できるすべての関数および演算では、最高 63 桁までの精度が使用できます。 10 進演算の結果の精度は、63 桁以下でなければなりません。

SQL での 10 進数演算

SQL における 10 進数演算の結果の精度および位取りは、以下の各式によって定義されます。記号 *p* と *s* は第 1 オペランドの精度と位取りを表し、記号 *p'* と *s'* は第 2 オペランドの精度と位取りを表します。

記号 *mp* は最大精度を表します。 *mp* の値は、次のような場合に 63 になります。

- *p* または *p'* が 31 より大きい
- 最大精度に値 63 が明示的に指定されている

それ以外の場合、*mp* の値は 31 です。

記号 *ms* は最大スケールを表します。 *ms* のデフォルト値は 31 です。 *ms* は、0 から最大精度の間の任意の数値に明示的に設定できます。

記号 *mds* は最小除算スケールを表します。 *mds* のデフォルト値は 0 です。この 0 は、最小スケールが指定されないことを示します。 *mds* は、1 から min (*ms*, 9) の間の任意の数値に明示的に設定できます。

最大精度、最大スケール、および最小除算スケールは、CRTSQLxxx コマンド、RUNSQLSTM コマンド、または SET OPTION ステートメントの DECRESULT パラメーターに明示指定することができます。これらは、ODBC データ・ソース、JDBC プロパティー、OLE DB プロパティー、および .NET プロパティーにも指定できます。

加算および減算

加算および減算の結果の位取りは max(*s,s'*) です。精度は、min(*mp*,max(*p-s,p'-s'*) +max(*s,s'*)+1) です。

乗算

乗算の結果の精度は min (*mp,p+p'*) であり、位取りは min(*ms,s+s'*) です。

除算

除算の結果の精度は (*p-s+s'*) + max(*mds*, min(*ms*, *mp* - (*p-s+s'*))) です。スケールは max(*mds*, min(*ms*, *mp* - (*p-s+s'*))) です。位取りは、正の数でなければなりません。

浮動小数点数オペランド

算術演算子のオペランドのいずれかが浮動小数点数であり、そのどちらも 10 進浮動小数点数でない場合は、浮動小数点数で演算が行われます。必要であれば、オペランドがまず倍精度浮動小数点数に変換されます。したがって、式のエレメントの中に浮動小数点数がある場合、その式の結果は倍精度の浮動小数点数になります。

浮動小数点数と整数の演算は、その整数を倍精度浮動小数点数に変換した一時的コピーを使用して行われます。浮動小数点数と 10 進数による演算は、倍精度の浮動小数点数に変換されたその 10 進数の一時的なコピーを使用して行われます。浮動小数点数演算の結果は、浮動小数点数の値の範囲内になければなりません。

浮動小数点オペランドは実数の概数を表すものなので、浮動小数点オペランド (または関数への引数) が処理される順序によって、結果が少しずつ変わることがあります。オペランドの処理順序は、最適化プログラムにより暗黙的に変更されることがあるので (例えば、最適化プログラムは、どの程度の並列性を使用するか、およびどのアクセス・プランを使用するかなどを決定します)、浮動小数点オペランドを使用する SQL ステートメントを実行する場合は、結果がいつも正確に同じになるという前提でアプリケーションを使用しないようにしてください。

10 進浮動小数点オペランド

算術演算子のどちらかのオペランドが 10 進浮動小数点数の場合、演算は 10 進浮動小数点数で行われます。

整数オペランドと DECFLOAT(n) オペランド

一方のオペランドが短精度整数または整数で、もう一方のオペランドが DECFLOAT の場合は、DECFLOAT(n) 数に変換された整数の一時コピーに基づいて、DECFLOAT(n) で演算が行われます。一方のオペランドが 64 ビット整数で、もう一方のオペランドが DECFLOAT である場合、64 ビット整数の一時コピーは

DECFLOAT(34) 数に変換されます。その後、2 つの DECFLOAT オペランドの規則が適用されます。

10 進数オペランドと DECFLOAT(n) オペランド

一方のオペランドが 10 進数で、もう一方のオペランドが DECFLOAT である場合、10 進数の精度に基づいて DECFLOAT 数に変換された 10 進数の一時コピーを使用して、DECFLOAT で演算が行われます。10 進数の精度が < 17 である場合、10 進数は DECFLOAT(16) に変換されます。それ以外の場合、10 進数は DECFLOAT(34) 数に変換されます。その後、2 つの DECFLOAT オペランドの規則が適用されます。

浮動小数点オペランドと DECFLOAT(n) オペランド

一方のオペランドが浮動小数点 (REAL または DOUBLE) で、もう一方のオペランドが DECFLOAT である場合、DECFLOAT(n) 数に変換された浮動小数点数の一時コピーを使用して、DECFLOAT(n) で演算が行われます。

2 つの DECFLOAT オペランド

オペランドが両方とも DECFLOAT(n) の場合は、DECFLOAT(n) で演算が行われます。一方のオペランドが DECFLOAT(16) で、もう一方のオペランドが DECFLOAT(34) である場合、DECFLOAT(34) で演算が行われます。

DECFLOAT の一般的な算術演算規則

DECFLOAT データ・タイプのすべての算術演算には、以下の一般的な規則が適用されます。

- 有限数のすべての演算は、該当する場合は係数で整数算術計算を使用して、正確な数学的結果が計算される場合と同じように、(個々の演算で説明されたとおり) 実行されます。

  理論上の正確な結果の係数の桁数が、その精度 (16 または 34) を表す桁数を超えない場合、(アンダーフローまたはオーバーフローがある場合を除いて) その係数が変更なく結果に使用されます。超える場合、その精度 (16 または 34) を表す桁数に正確に丸められ (短縮され)、除去された桁数分、指数が増えます。

  丸めでは、DECFLOAT 丸めモードを使用します。詳しくは、 154 ページの『CURRENT DECFLOAT ROUNDING MODE』を参照してください。

  結果の調整された指数の値が E_{min} より小さい場合、非正規化警告が戻されます。[41]この場合、計算された係数と指数により結果が生成されます。ただし、指数の値が E_{tiny} より小さい場合を除きます。この場合、指数は E_{tiny} に設定され、係数は、指数の調整と一致するように丸められ (おそらくゼロに)、符号は変わりません。この丸めにより、不正確な結果が得られる場合、アンダーフロー警告が戻されます。[41]

41. 警告が戻されるのは、SQL_DECFLOAT_WARNINGS 照会オプションに *YES が指定されている場合のみです。

結果の調整された指数の値が E_{max} より大きい場合、オーバーフロー警告が戻されます。この場合、結果は無限大になる可能性があります。符号は理論上の結果と同じになります。

- 特殊値 Infinity を使用する算術計算は、通常の規則に従います。この規則では、負の Infinity はすべての有限数より小さく、正の Infinity はすべての有限数より大きくなります。これらの規則では、無限の結果が常に正確です。以下の算術演算は警告を返し、NaN の結果になります。[41]
  - 加算または減算演算時に、+無限大を -無限大に加算する
  - 0 または -0 に +無限大または -無限大を乗算する
  - +無限大または -無限大を +無限大または -無限大で除算する
- シグナル NaNs は、算術演算のオペランドとして使用されて NaNs が戻される場合、常に警告またはエラーを出します。
- オペランドが NaN である算術演算の結果は、NaN です。結果の符号は、NaN である第 1 オペランドからコピーされます。結果が NaN である場合は必ず、結果の符号は、コピーされるオペランドのみによって決まります。
- 乗算または除算の結果の符号が負になるのは、オペランドの符号が異なり、いずれも NaN でない場合のみです。
- 加算または減算の結果の符号が負になるのは、結果がゼロより小さく、いずれのオペランドも NaN でない場合のみです。

算術演算と数字関数の結果が負のゼロになる場合もあります。

特殊値を含む例

```
INFINITY + 1              = INFINITY
INFINITY + INFINITY       = INFINITY
INFINITY + -INFINITY      = NAN        -- warning
NAN + 1                   = NAN
NAN + INFINITY            = NAN
1 - INFINITY              = -INFINITY
INFINITY - INFINITY       = NAN        -- warning
-INFINITY - -INFINITY     = NAN        -- warning
-0.0 - 0.0E1              = -0.0
-1.0 * 0.0E1              = -0.0
1.0E1 / 0                 = INFINITY
-1.0E5 / 0.0              = -INFINITY
1.0E5 / -0                = -INFINITY
INFINITY / -INFINITY      = NAN        -- warning
INFINITY / 0              = INFINITY   -- warning
-INFINITY / 0             = -INFINITY  -- warning
-INFINITY / -0            = INFINITY   -- warning
```

特殊タイプのオペランド

特殊タイプのソース・データ・タイプが数値であっても、特殊タイプを算術演算子と組み合わせて使用することはできません。 算術演算を行うには、そのソースとして算術演算子を持つ関数を作成します。例えば、特殊タイプ INCOME および EXPENSES があり、両方とも DECIMAL(8,2) のデータ・タイプを持っている場合、次のユーザー定義関数 REVENUE を使用して、他方から一方を減算することができます。

```
CREATE FUNCTION REVENUE ( INCOME, EXPENSES )
  RETURNS DECIMAL(8,2) SOURCE "-" ( DECIMAL, DECIMAL)
```

別の方法として、新規のデータ・タイプを減算するユーザー定義関数を使用して、- (マイナス) 演算子を多重定義する方法があります。

```
CREATE FUNCTION "-" ( INCOME, EXPENSES )
  RETURNS DECIMAL(8,2) SOURCE "-" ( DECIMAL, DECIMAL)
```

また、特殊タイプを組み込みタイプにキャストして、結果を算術演算子のオペランドとして使用することもできます。

連結演算子

連結演算子 (CONCAT または ||) を使用している場合、式の結果はストリングになります。

連結のオペランドのデータ・タイプは、互換の、ストリング、数値、日付、時刻、またはタイム・スタンプでなければなりません。[42] 連結のオペランドを特殊タイプにすることはできません。数値、日付、時刻、またはタイム・スタンプのオペランドを指定すると、連結前に、等価の文字ストリングにキャストされます。バイナリー・ストリングは、FOR BIT DATA として定義されていない文字ストリングとは連結できないことに注意してください。

結果のデータ・タイプは、オペランドのデータ・タイプによって決まります。結果のデータ・タイプは、次の表に要約されています。

表 *28.* 連結を用いた結果のデータ・タイプ

一方のオペランド列	他方のオペランド	結果列のデータ・タイプ
DBCLOB(x)	CHAR(y)* または VARCHAR(y)* または CLOB(y)* または GRAPHIC(y) または VARGRAPHIC(y) または DBCLOB(y)	DBCLOB(z) (ただし、z = MIN(x + y, DBCLOB の最大長))
VARGRAPHIC(x)	CHAR(y)* または VARCHAR(y)* または GRAPHIC(y) または VARGRAPHIC(y)	VARGRAPHIC(z) (ただし、z = MIN(x + y, VARGRAPHIC の最大長))
GRAPHIC(x)	CHAR(y)* 混合データ	VARGRAPHIC(z) (ただし、z = MIN(x + y, VARGRAPHIC の最大長))
GRAPHIC(x)	CHAR(y)* SBCS データまたは GRAPHIC(y)	GRAPHIC(z) (ただし、z = MIN(x + y, GRAPHIC の最大長))
CLOB(x)*	GRAPHIC(y) または VARGRAPHIC(y)	DBCLOB(z) (ただし、z = MIN(x + y, DBCLOB の最大長))
VARCHAR(x)*	GRAPHIC(y)	VARGRAPHIC(z) (ただし、z = MIN(x + y, VARGRAPHIC の最大長))

42. 縦線文字 (|) を使用すると、リレーショナル・データベース製品間のコードの移植性が抑制される可能性があります。|| 演算子の代わりに CONCAT 演算子を使用してください。または、SQL 2003 コア標準への準拠が最も優先される場合は、|| 演算子を使用してください。

表 28. 連結を用いた結果のデータ・タイプ (続き)

一方のオペランド列	他方のオペランド	結果列のデータ・タイプ
CLOB(x)	CHAR(y) または VARCHAR(y) または CLOB(y)	CLOB(z) (ただし、z = MIN(x + y, CLOB の最大長))
VARCHAR(x)	CHAR(y) または VARCHAR(y)	VARCHAR(z) (ただし、z = MIN(x + y, VARCHAR の最大長))
CHAR(x) 混合データ	CHAR(y)	VARCHAR(z) (ただし、z = MIN(x + y, VARCHAR の最大長))
CHAR(x) SBCS データ	CHAR(y)	CHAR(z) (ただし、z = MIN(x + y, CHAR の最大長))
BLOB(x)	BINARY(y) または VARBINARY(y) または BLOB(y) または CHAR(y) FOR BIT DATA または VARCHAR(y) FOR BIT DATA	BLOB(z) (ただし、z = MIN(x + y, BLOB の最大長))
VARBINARY(x)	BINARY(y) または VARBINARY(y) または CHAR(y) FOR BIT DATA または VARCHAR(y) FOR BIT DATA	VARBINARY(z) (ただし、z = MIN(x + y, VARBINARY の最大長)
BINARY(x)	VARCHAR(y) FOR BIT DATA	VARBINARY(z) (ただし、z = MIN(x + y, VARBINARY の最大長)
BINARY(x)	BINARY(y) または CHAR(y) FOR BIT DATA	BINARY(z) (ただし、z = MIN(x + y, BINARY の最大長)
注: * 他方のオペランドがグラフィック・ストリングであり、そのストリングが Unicode である場合は、文字ストリングのみが許可されます。		

表 29. 連結を用いた結果のコード化スキーム

一方のオペランド列	他方のオペランド	結果列のデータ・タイプ
Unicode データ	Unicode データまたは DBCS または混合または SBCS データ	Unicode データ
DBCS データ	DBCS データ	DBCS データ
ビット・データ	混合または SBCS またはビット・データ	ビット・データ
混合データ	混合または SBCS データ	混合データ
SBCS データ	SBCS データ	SBCS データ

両方のオペランドの合計長が結果のデータ・タイプの最大長属性を超える場合は、次のようになります。

- 結果の長さ属性が結果のデータ・タイプの最大長になります。[43]
- ブランクのみが切り捨てられた場合は、警告もエラーも生じません。
- ブランク以外の文字が切り捨てられた場合は、エラーが起こります。

NULL になる可能性があるオペランドをどちらか一方に使用した場合は、結果も NULL になる可能性があります。また、どちらか一方が NULL ならば、結果は NULL 値になります。それ以外の場合、結果は、第 1 オペランドのストリングの後に、第 2 オペランドのストリングが続いたストリングになります。

混合データを連結する場合は、その結果の 『継ぎ目に』 余分なシフト・コードが入ることはありません。したがって、第 1 オペランドのストリングが「シフトイン文字」文字 (X'0F') で終わり、第 2 オペランドの文字ストリングが「シフトアウト」文字 (X'0E') で始まっていても、第 1 オペランドのシフトイン文字と第 2 オペランドのシフトアウト文字 (合わせて 2 バイト) は結果から除去されます。

余分なシフト文字が除去された場合を除いて、オペランドの長さの合計が実際の結果の長さになります。余分なシフト文字が除去された場合は、実際の結果の長さは、オペランドの長さの合計よりも 2 だけ小さくなります。

結果の CCSID は、オペランドの CCSID によって決定されます (これについては、 139 ページの『ストリングを結合する演算に適用される変換規則』で説明しています)。これらの規則による結果として、以下の点に注意してください。

- ビット・データのオペランドがあれば、結果はビット・データになります。
- 一方のオペランドが混合データで、もう一方が SBCS データの場合、結果は混合データになります。ただし、このことは、その結果が、形式が正しい混合データであることを必ずしも意味するわけではありません。

例

ブランクを間に置いて、列 FIRSTNME と列 LASTNAME を連結します。

```
FIRSTNME CONCAT ' ' CONCAT LASTNAME
```

スカラー全選択

式の中で使用できるスカラー全選択は、括弧で囲んだ全選択で、単一の列値から成る単一の行を戻します。この全選択が行を戻さない場合は、式の結果は NULL 値になります。選択リスト・エレメントが、単なる列名である式の場合は、その列の名前に基づいて結果の列名が決まります。それ以外の場合は、その結果の列は無名となります。

詳しくは、 841 ページの『全選択』を参照してください。

照会に以下の指定がある場合、スカラー全選択は許可されません。

- 分散表

43. 該当の式が選択リストに含まれている場合は、長さ属性は、最大レコード・サイズに収まるようにするためにさらに縮小されることがあります。詳しくは、 1295 ページの『最大行サイズ 』を参照してください。

- 読み取りトリガーを指定する表
- 複数の物理ファイル・メンバー上に構築された論理ファイル

スカラー全選択が副選択の場合、スカラー副選択とも言われます。詳しくは、 788 ページの『副選択』を参照してください。

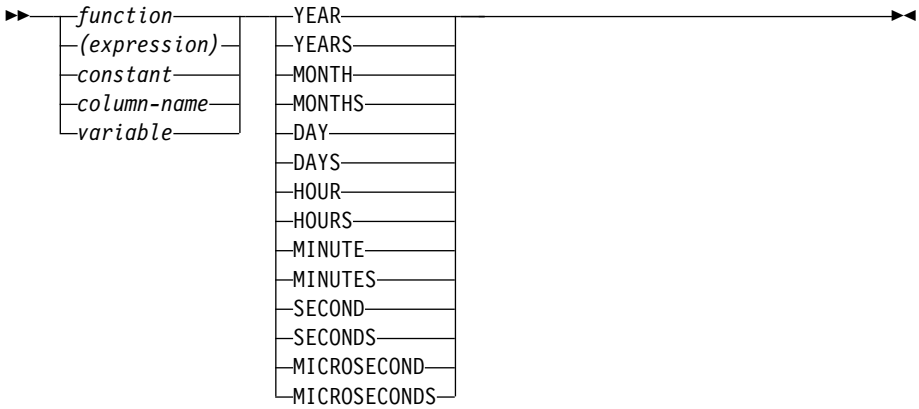
日付/時刻のオペランドと期間

日付/時刻の値に対して、増分や減分、および減算を行うことができます。これらの演算では、期間 と呼ばれる 10 進数を使用することができます。この期間 は、時間間隔を表す正または負の数値です。

期間には、以下の 4 つのタイプがあります。

ラベル付き期間
: ラベル付き期間の形式は、以下のとおりです。

labeled-duration:



ラベル付き期間 (labeled-duration) は、特定の時間単位を表すもので、数値 (式の結果でも可) の後に 7 つの期間キーワード YEARS、MONTHS、DAYS、HOURS、MINUTES、SECONDS、または MICROSECONDS のうちの 1 つを付けたものです。[44]指定した値は、DECIMAL(15,0) の数値へ割り当てられる場合と同様に変換されます。ただし、DECIMAL(27,12) を使用する SECONDS は別で、ここには 0 から 12 桁までの小数秒を組み込むことができます。 MICROSECONDS 期間は、6 桁の小数秒を戻します。

ラベル付き期間は、算術演算子のオペランドの一方がデータ・タイプとして DATE、TIME、または TIMESTAMP を持つ値である場合にのみ、もう一方のオペランドとして使用することができます。したがって、HIREDATE + 2 MONTHS + 14 DAYS という式は有効ですが、HIREDATE + (2 MONTHS + 14 DAYS) という式は無効です。この 2 つの式で、2 MONTHS および 14 DAYS がラベル付き期間です。

日付期間
: 日付期間 は、DECIMAL(8,0) の数値として表現される年数、月数、および

44. これらのキーワードは、単数形 (YEAR、MONTH、DAY、HOUR、MINUTE、SECOND、および MICROSECOND) で指定しても構いません。

日数を表します。この数値が正しく解釈されるためには、*yyyymmdd* という形式にする必要があります (*yyyy* は年数、*mm* は月数、*dd* は日数を表します)。ある日付の値から別の日付の値を引いた結果 (例えば、式 HIREDATE - BRTHDATE の結果) は、日付期間になります。

時刻期間
: 時刻期間 は、DECIMAL(6,0) の数値として表現される時間数、分数、および秒数を表します。この数値が正しく解釈されるためには、*hhmmss* という形式にする必要があります。*hh* は時間数、*mm* は分数、*ss* は秒数を表します。ある時刻の値から別の時刻の値を引いた結果は、時刻期間になります。

タイム・スタンプ期間
: タイム・スタンプ期間 は、DECIMAL(14+*s*,*s*) の数値として表現され (*s* は小数秒の桁数で、0 から 12 の範囲)、年数、月数、日数、時間数、分数、秒数、および小数秒数を表します。この数値が正しく解釈されるためには、*yyyymmddhhmmss.zzzzzzzzzzzz* という形式になっていなければなりません (*yyyy*、*mm*、*dd*、*hh*、*mm*、*ss*、*zzzzzzzzzzzz* は、年数、月数、日数、時間数、分数、秒数、小数秒数にそれぞれ相当します)。 1 つのタイム・スタンプ値を別のタイム・スタンプ値から引いた結果は、タイム・スタンプ・オペランドの最大タイム・スタンプ精度に一致する位取りを持つタイム・スタンプ期間となります。

SQL における日付/時刻の算術演算

日付/時刻の値に対して実行できる算術演算は、加算と減算だけです。日付/時刻の値が加算のオペランドである場合は、もう一方のオペランドは期間でなければなりません。

日付/時刻の値に対する加算の演算子の用法に関する特定の規則は、次のとおりです。

- 一方のオペランドが日付の場合、もう一方のオペランドは日付期間、または年、月、日のラベル付き期間のいずれかでなければなりません。
- 一方のオペランドが時刻の場合、もう一方のオペランドは時刻期間、または時、分、秒のラベル付き期間のいずれかでなければなりません。
- 一方のオペランドがタイム・スタンプの場合、もう一方のオペランドは期間でなければなりません。この場合、どのようなタイプの期間でも使用できます。
- 加算演算子のどちらのオペランドにも、タイプなしパラメーター・マーカーは使用できません。

日付/時刻の値は、期間から減算することはできず、また日付/時刻の 2 つの値の減算の処理は、日付/時刻の値から期間を減算する処理とは異なるので、日付/時刻の値に対する減算演算子の用法に関する規則は、加算の場合と同じではありません。日付/時刻の値に対する減算演算子の用法に関する特定の規則は、次のとおりです。

- 第 1 オペランドが日付の場合、第 2 オペランドは、日付、タイム・スタンプ、日付期間、日付のストリング表現、タイム・スタンプのストリング表現、または、年、月、または日のラベル付き期間でなければなりません。
- 第 2 オペランドが日付の場合、第 1 オペランドは、日付、タイム・スタンプ、日付のストリング表記、またはタイム・スタンプのストリング表記であることが必要です。

- 第 1 オペランドが時刻の場合、第 2 オペランドは時刻、時刻期間、時刻のストリング表現、または時、分、秒のラベル付き期間のいずれかでなければなりません。
- 第 2 オペランドが時刻の場合、第 1 オペランドは時刻、または時刻のストリング表現のいずれかでなければなりません。
- 第 1 オペランドがタイム・スタンプの場合、第 2 オペランドは、日付、タイム・スタンプ、日付のストリング表記、タイム・スタンプのストリング表記、または期間であることが必要です。第 2 オペランドがタイム・スタンプのストリング表現の場合、第 1 オペランドと同じ精度のタイム・スタンプに暗黙的に変換されます。
- 第 2 オペランドがタイム・スタンプの場合、第 1 オペランドは、日付、タイム・スタンプ、日付のストリング表記、またはタイム・スタンプのストリング表記であることが必要です。第 1 オペランドがタイム・スタンプのストリング表現の場合、第 2 オペランドと同じ精度のタイム・スタンプに暗黙的に変換されます。
- 減算演算子のどちらのオペランドにも、タイプなしパラメーター・マーカーは使用できません。

日付の算術演算

日付は、減算を行えるほかに、増やしたり減らしたりすることができます。

日付の減算

ある日付 (DATE1) から別の日付 (DATE2) を引いた結果は、その 2 つの日付の間にある年、月および日の数を示す日付期間になります。この結果のデータ・タイプは、10 進数 (8,0) です。 DATE1 が DATE2 より大きいか、または両者が等しい場合、DATE1 から DATE2 が引かれます。 DATE1 が DATE2 より小さい場合でも、DATE2 から DATE1 が引かれ、結果の符号が負になります。以下の手順型の記述は、RESULT = DATE1 - DATE2 という演算に伴う各ステップを説明したものです。

```
DAY(DATE2) <= DAY(DATE1) の場合 :
    then DAY(RESULT) = DAY(DATE1) - DAY(DATE2).

If DAY(DATE2) > DAY(DATE1)
    then DAY(RESULT) = N + DAY(DATE1) - DAY(DATE2)
      where N = the last day of MONTH(DATE2).
    MONTH(DATE2) is then incremented by 1.

MONTH(DATE2) <= MONTH(DATE1) の場合 :
    then MONTH(RESULT) = MONTH(DATE1) - MONTH(DATE2).

If MONTH(DATE2) > MONTH(DATE1)
    then MONTH(RESULT) = 12 + MONTH(DATE1) - MONTH(DATE2).
    YEAR(DATE2) is then incremented by 1.

YEAR(RESULT) = YEAR(DATE1) - YEAR(DATE2).
```

例えば、DATE('3/15/2000') - '12/31/1999' は 215 (つまり、0 年、2 月、15 日という期間) になります。

日付の増減

日付に期間を加えた結果や、日付から期間を引いた結果は、それ自身が日付になります。 (この演算の目的に沿って、月はカレンダーのページと同等の意味を持ちます。日付に月を加えるのは、その日付があるカレンダーのページを月の数だけめくることに相当します。) 結果は、0001 年 1 月 1 日から 9999 年 12 月 31 日までの間に含まれる日付でなければなりません。

日付に対して、年の期間を加えたり引いたりした場合、増減されるのはその日付の年の部分だけです。月は未変更のままです。日についても同様です。ただし、結果がうるう年以外の年の 2 月 29 日になってしまった場合は除きます。結果がうるう年以外の年の 2 月 29 日になった場合は、日が 28 に変更され、月の最後の日の調整を行ったことを示す '01506' の SQLSTATE が SQL 診断域の RETURNED_SQLSTATE 条件領域に割り当てられます (または 'W' が SQLCA の SQLWARN6 にセットされます)。

同様に、日付に対して月の期間を加えたり引いたりした場合も、まず日付の月の部分だけが増減され、必要があれば年の部分が増減されます。結果が無効な日付 (例えば、9 月 31 日など) にならない限り、日付の日の部分は変更されません。結果が 9 月 31 日などの無効な日付になった場合は、日の部分が該当する月の最終日に変更され、月の最後の日の調整を行ったことを示す 'W' が SQLCA の SQLWARN6 にセットされます。

日の期間を加えたり引いたりした場合も、まず日付の日の部分が増減され、必要がある場合にだけ月および年の部分が増減されます。DAYS のラベル付き期間を加えた場合は、月の最後の日の調整は行われません。

日付期間 (正または負) も、日付に加えたり、日付から引いたりすることができます。ラベル付き期間を使用すると、結果は有効な日付となります。この場合も、月の終了日の調整が必要ならば、SQLCA 内に警告標識がセットされます。

日付に正の日付期間を加えた場合や、日付から負の日付期間を引いた場合は、日付が年、月、日の順に、指定した数だけ増やされます。DATE1 + X (X は、正の DECIMAL(8,0)) は、次の式と等価です。

- DATE1 + YEAR(X) YEARS + MONTH(X) MONTHS + DAY(X) DAYS

日付から正の日付期間を引いた場合や、日付に負の日付期間を加えた場合は、日付が日、月、年の順に、指定した数だけ減らされます。したがって、DATE1 - X (X は、正の DECIMAL(8,0)) は、次の式と等価です。

- DATE1 - DAY(X) DAYS - MONTH(X) MONTHS - YEAR(X) YEARS

注**:** 所定の日付に 1 つ以上の月を加えた上で、その結果から同数の月を引いても、最終的な日付が元の日付と同じにならない場合があります。

論理的に等価な式であるからといって、同じ結果が生成されるとは限らないことにも注意してください。例えば、次の例があります。

- (DATE('2002–01–31') + 1 MONTH) + 1 MONTH の結果は、2002–03–28 という日付になります。

この結果と次の式の結果は同じではありません。

- DATE('2002–01–31') + 2 MONTHS の結果は、2002–03–31 という日付になります。

日付にラベル付き日付期間を加算および減算する順序が、結果に影響します。日付期間を追加または減算した結果の互換性を保つため、特定の計算順序を使用することが必要です。ラベル付き日付期間を日付に加算する場合は、YEARS + MONTHS + DAYS の順序で指定します。日付からラベル付き日付期間を減算する場合は、DAYS - MONTHS - YEARS の順序で指定します。例えば、ある日付に 1 年と 1 日を加算するには、次のように指定します。

- DATE1 + 1 YEAR + 1 DAY

ある日付から 1 年と 1 月と 1 日を減算するには、次のように指定します。

- DATE1 - 1 DAY - 1 MONTH - 1 YEAR

時刻の算術演算

時刻は、減算に加え、増やしたり減らしたりすることができます。

時刻の減算

ある時刻 (TIME1) から別の時刻 (TIME2) を引いた結果は、その 2 つの時刻の間にある時、分、および秒の数を示す時刻期間となります。この結果のデータ・タイプは 10 進数 (6,0) です。TIME1 が TIME2 より大きいか、または両者が等しい場合、TIME1 から TIME2 が引かれます。 TIME1 が TIME2 より小さい場合でも、TIME2 から TIME1 が引かれ、結果の符号が負になります。以下の手順型の記述は、RESULT = TIME1 - TIME2 という演算に伴う各ステップを説明したものです。

SECOND(TIME2) <= SECOND(TIME1) の場合 :
    then SECOND(RESULT) = SECOND(TIME1) - SECOND(TIME2).

If SECOND(TIME2) > SECOND(TIME1)
    then SECOND(RESULT) = 60 + SECOND(TIME1) - SECOND(TIME2).
    MINUTE(TIME2) is then incremented by 1.

MINUTE(TIME2) <= MINUTE(TIME1) の場合 :
    then MINUTE(RESULT) = MINUTE(TIME1) - MINUTE(TIME2).

If MINUTE(TIME2) > MINUTE(TIME1)
    then MINUTE(RESULT) = 60 + MINUTE(TIME1) - MINUTE(TIME2).
    HOUR(TIME2) is then incremented by 1.

HOUR(RESULT) = HOUR(TIME1) - HOUR(TIME2).

例えば、TIME('11:02:26') - '00:32:56' の結果は、102930 (10 時間、29 分、30 秒という期間) になります。

時刻の増減

時刻に期間を加えた結果や、時刻から期間を引いた結果は、それ自身が時刻になります。時のオーバーフローやアンダーフローは破棄されるので、結果は常に時刻となります。時刻に対して、時の期間を加えたり引いたりした場合は、時刻の時の部分だけが増減されます。分および秒の部分は変更されません。

同様に、時刻に対して分の期間を加えたり引いたりした場合は、まず分の部分だけが増減され、必要があれば時の部分が増減されます。時刻の秒の部分は変更されません。

秒の期間を加えたり引いたりした場合も、時刻の秒の部分が増減され、必要があるときだけ分および時の部分が増減されます。

時刻期間 (正または負) を時刻に加えたり、時刻から引いたりすることもできます。結果は、指定した時間数、分数、秒数分だけこの順序で増減させた時刻になります。TIME1 + X (「X」は DECIMAL(6,0)) は次の式と同等です。

```
TIME1 + HOUR(X) HOURS + MINUTE(X) MINUTES + SECOND(X) SECONDS
```

タイム・スタンプの算術演算

タイム・スタンプは、減算が行えるほかに、増やしたり減らしたりすることができます。

タイム・スタンプの減算

あるタイム・スタンプ (TS1) から別のタイム・スタンプ (TS2) を引いた結果は、その 2 つのタイム・スタンプの間にある年、月、日、時、分、秒および少数秒の数を示すタイム・スタンプ期間になります。結果のデータ・タイプは DECIMAL(14+*s*,*s*) です。*s* は、TS1 と TS2 の最大タイム・スタンプ精度です。 TS1 が TS2 より大きいか、または両者が等しければ、TS1 から TS2 が引かれます。TS1 が TS2 より小さい場合でも、TS2 から TS1 が引かれ、結果の符号が負になります。以下の手順型の記述は、RESULT = TS1 - TS2 という演算に伴う各ステップを説明したものです。

SECOND(TS2,s) <= SECOND(TS1,s) の場合:
SECOND(RESULT,s) = SECOND(TS1,s) - SECOND(TS2,s)

SECOND(TS2,s) > SECOND(TS1,s) の場合:
SECOND(RESULT,s) = 60 + SECOND(TS1,s) - SECOND(TS2,s,)
MINUTE(TS2) は 1 だけ増やされる

タイム・スタンプの分の部分は、時刻の減算規則で
指定されているように減算されます。
HOUR(TS2) <= HOUR(TS1) の場合 :
HOUR(RESULT) = HOUR(TS1) - HOUR(TS2)

If HOUR(TS2) > HOUR(TS1)
HOUR(RESULT) = 24 + HOUR(TS1) - HOUR(TS2)
DAY(TS2) は 1 だけ増やされる

タイム・スタンプの日の部分の減算は、日付の減算に適用される規則に従って行われます。

日付 (D1) をタイム・スタンプ (TS1) から減算した結果は、TIMESTAMP(D1) を TS1 から減算した結果と同じです。同様に、あるタイム・スタンプ (TS1) を日付 (D2) から減算した結果は、TS1 を TIMESTAMP(D2) から減算した結果と同じです。

タイム・スタンプの増減

タイム・スタンプに期間を加えた結果や、タイム・スタンプから期間を引いた結果は、それ自身がタイム・スタンプになります。結果タイム・スタンプの精度は、タイム・スタンプ・オペランドの精度に一致します。 時のオーバーフローまたはアンダーフローが、結果の日付部分に桁送りされることを除いて、以前の項で述べたとおりの日付および時刻の算術演算が行われます。この結果は、有効な日付の範囲内になければなりません。小数秒のオーバーフローは秒に桁送りされます。したがって、タイム・スタンプ TIMESTAMP1 からの期間 X (X は DECIMAL(14+s,s) の数値) の減算は、以下の式と同等です。

```
TIMESTAMP1 - YEAR(X) YEARS - MONTH(X) MONTHS - DAY(X) DAYS
           - HOUR(X) HOURS - MINUTE(X) MINUTES
           - SECOND(X,s) SECONDS
```

ゼロ以外の位取りの期間、MICROSECOND または MICROSECONDS のラベル付き期間、または、値に小数秒が含まれる SECOND または SECONDS のラベル付き期間を減算する場合、その減算は、最大 12 桁の小数秒が含まれるタイム・スタンプ値であるかのように実行されます。結果値は、タイム・スタンプ・オペランドのタイム・スタンプの精度を持つタイム・スタンプ値に割り当てられ、その結果として秒の小数部分の桁は切り捨てられます。

演算の優先順位

括弧の中にある式は最初に計算されます。括弧によって計算の順序が指定されていないときは、接頭演算子 (-、単項減算など) の後、かつ乗算および除算の前に、累乗演算を行います。乗算および除算は、加算および減算の前に行います。同じ優先レベルにある演算子は、左から右の順に処理されます。

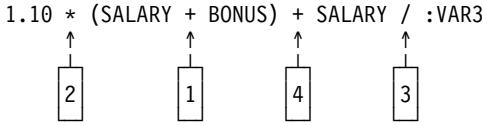
次の表は、すべての演算子の優先順位を示しています。

優先順位	演算子
1	+、- (符号付数値に使用する場合)
2	**
3	*、/、CONCAT、||
4	+、- (2 つのオペランドの間で使用する場合)

例 1:

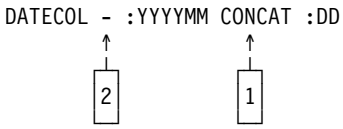
この例では、括弧で囲まれている (SALARY + BONUS) の加算が最初の演算です。 2 つ目の加算演算子よりも優先順位が高く、除算演算子より左に位置する乗算が、2 番目の演算です。 2 つ目の加算演算子よりも優先順位が高い除算が、3 番目

の演算です。最後に、残った加算が実行されます。



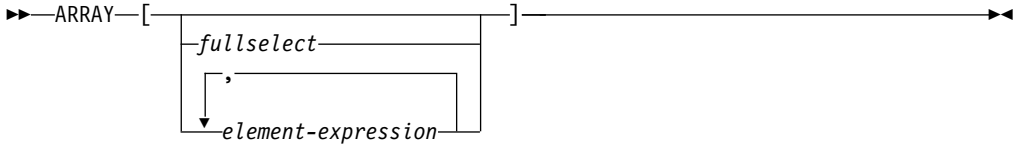
例 2:

この例では、最初の演算 (CONCAT) で変数 YYYYMM と DD の文字ストリングが日付を表す 1 つのストリングに結合されます。次に 2 番目の演算では、結合された日付を DATECOL で処理された日付から減算します。その計算結果が、2 つの日付の間で経過した時間を表します。



ARRAY コンストラクター

ARRAY コンストラクターからは、式のリストまたは全選択で指定した配列が返されます。



fullselect
1 つの列を返す全選択。全選択によって返される値は、配列の各エレメントです。配列のカーディナリティーは、全選択によって返される行の数と等しくなります。全選択では、ORDER BY 文節を使用して、配列の各エレメントの順序を指定することもできます。指定しなければ、順不同になります。配列の基本タイプの属性は、全選択の結果列のデータ・タイプと同じです。

element-expression
配列エレメントの値を定義する式。配列のカーディナリティーは、エレメント式の数と等しくなります。最初の *element-expression* は配列指標 1 を持つ配列エレメントに割り当てられます。2 番目の *element-expression* は配列指標 2 を持つ配列エレメントに割り当てられ、それ以降も同様になります。エレメントの式のデータ・タイプはすべて互換性がなければなりません。配列の基本タイプの属性は、 133 ページの『結果のデータ・タイプに関する規則』で説明されているように、すべてのオペランドによって決まります。

大括弧の中に式を入れない場合の結果は、空の配列です。空の配列のカーディナリティーは、0 です。

ARRAY コンストラクターは SET 変数または割り当てステートメント の右側にのみ指定できます。

例

- 配列タイプ PHONENUMBERS の配列変数 RECENT_CALLS を固定番号の配列に設定します。

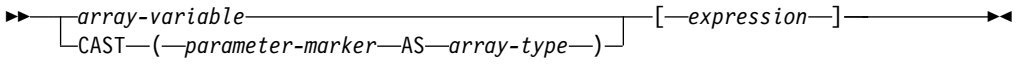
```
SET RECENT_CALLS = ARRAY[9055553907, 4165554213, 4085553678]
```

- 配列タイプ PHONENUMBERS の配列変数 DEPT_PHONES を、DEPARTMENT_INFO 表から取得される電話番号の配列に設定します。

```
SET DEPT_PHONES = ARRAY[SELECT DECIMAL(AREA_CODE CONCAT '555' CONCAT EXTENSION,16)
                        FROM DEPARTMENT_INFO
                        WHERE DEPTID = 624]
```

ARRAY エレメントの指定

ARRAY エレメントの指定からは、*expression* で指定する配列のエレメントが返されます。



array-variable
SQL プロシージャーまたは SQL 関数のタイプ配列の変数またはパラメーターを指定します。

CAST(*parameter-marker* **AS** *array-type***)**
パラメーター・マーカーで使用する配列データ・タイプを指定します。パラメーター・マーカー値に渡す配列データ・タイプは、この配列データ・タイプと完全に一致している必要があります。

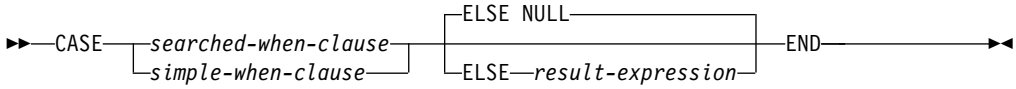
[*expression***]**
配列から抽出されるエレメントの配列指標を指定します。配列指標では、ゼロのスケールまたは DECFLOAT の厳密な数値を返す必要があります。値は、1 から配列のカーディナリティーまでの範囲内でなければなりません。

結果のデータ・タイプは、CREATE TYPE (配列) ステートメントの配列で指定したデータ・タイプになります。

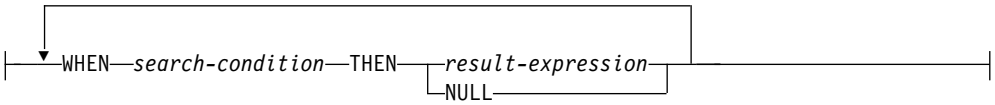
expression が NULL の場合や配列が NULL の場合は、NULL 値が返されます。

CASE 式

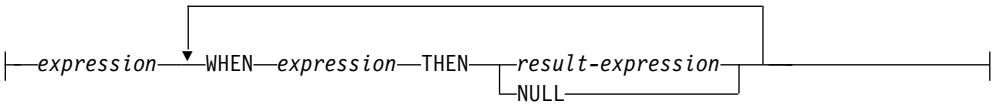
CASE 式により、1 つ以上の条件の評価に基づいて式を選択することができます。



searched-when-clause:



simple-when-clause:



一般に、*case* 式 の値は、真であると評価される最初の (左端の) *when* 文節 に続く結果式 の値です。どの *when* 文節 も真であると評価されず、かつ ELSE キーワードが存在する場合は、結果は ELSE 結果式 の値または NULL になります。どの *when* 文節 も真であると評価せず、かつ ELSE キーワードが存在しない場合は、結果は NULL になります。 *when* 文節 が不明 (NULL のため) と評価された場合は、 *when* 文節 は真ではなく、したがって、偽であると評価される *when* 文節 と同じ方法で扱われます。

searched-when-clause
評価のために提供される行または表データのグループのそれぞれに適用される検索条件 と、その条件が真の場合の結果を指定します。

simple-when-clause
最初の WHEN キーワードの前の式 の値が、それぞれの WHEN キーワードの後の式 の値と等しいかどうかテストされることを指定します。また、その条件が真の場合の結果も指定します。

最初の WHEN キーワードの前にある *expression* のデータ・タイプは、それぞれの WHEN キーワードの後ろにある *expression* のデータ・タイプと互換であることが必要です。

結果式 または **NULL**
THEN キーワードおよび ELSE キーワードに続く値を指定します。 CASE 式では、定義済みのデータ・タイプを持つ少なくとも 1 つの結果式 がなければなりません。すべてのケースに NULL を指定することはできません

すべての 結果式 は互換データ・タイプを持っている必要があり、結果の属性は 133 ページの『結果のデータ・タイプに関する規則』に基づいて決まります。

search-condition
行または表データのグループについて、真、偽、または不明の条件を指定します。

検索条件 には、EXISTS または IN 述部に副照会を組み込むことはできません。

CASE 式が、選択リスト、UPDATE または MERGE ステートメントの SET 節、または、 INSERT または MERGE ステートメントの VALUES 節に指定され、かつ、*simple-when-clause* または *searched-when-clause* が、列アクセス制御がアクティブになっている列を参照している場合、列値の代わりに、マスクされた値が使用されます。

CASE の持つ機能のサブセットを扱うように特化された 2 つのスカラー関数、NULLIF および COALESCE があります。次の表は、CASE またはこれらの関数を使用した同等の式を示しています。

表 *30.* 同等の *CASE* 式

CASE 式	同等の式
CASE WHEN e1=e2 THEN NULL ELSE e1 END	NULLIF(e1,e2)
CASE WHEN e1 IS NOT NULL THEN e1 ELSE e2 END	COALESCE(e1,e2)
CASE WHEN e1 IS NOT NULL THEN e1 ELSE COALESCE(e2,...,eN) END	COALESCE(e1,e2,...,eN)

例

- 部門番号の先頭文字が組織上の部である場合には、CASE 式を使用して、それぞれの従業員が所属する部門の完全な名前をリストすることができます。

```
SELECT EMPNO, LASTNAME,
  CASE SUBSTR(WORKDEPT,1,1)
  WHEN 'A' THEN 'Administration'
  WHEN 'B' THEN 'Human Resources'
  WHEN 'C' THEN 'Accounting'
  WHEN 'D' THEN 'Design'
  WHEN 'E' THEN 'Operations'
  END
FROM EMPLOYEE
```

- 教育年数が EMPLOYEE 表で使用されており、教育のレベルを示します。CASE 式を使用して、これらをグループ化し、教育のレベルを示します。

```
SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME,
  CASE
  WHEN EDLEVEL < 15 THEN 'SECONDARY'
  WHEN EDLEVEL < 19 THEN 'COLLEGE'
  ELSE 'POST GRADUATE'
  END
FROM EMPLOYEE
```

- CASE ステートメントを使用した別の興味ある例として、0 による割り算のエラーの保護があります。例えば、次のコードでは、歩合で収入の 25% 以上を稼ぎながら、歩合の全額を支払われていない従業員を見つけだすものです。

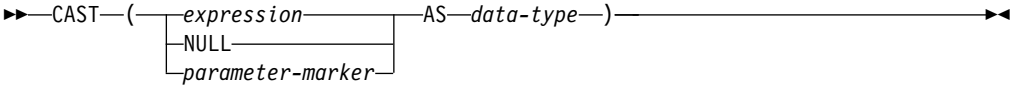
```
SELECT EMPNO, WORKDEPT, SALARY+COMM
FROM EMPLOYEE
WHERE (CASE WHEN SALARY=0 THEN NULL
            ELSE COMM/SALARY
       END) > 0.25
```

- 次の CASE 式は同等のものです。

```
SELECT LASTNAME,
 CASE
 WHEN LASTNAME = 'Haas' THEN 'President'
 ...
 ELSE 'Unknown'
 END
FROM EMPLOYEE

SELECT LASTNAME,
 CASE LASTNAME
 WHEN 'Haas' THEN 'President'
 ...
 ELSE 'Unknown'
 END
FROM EMPLOYEE
```

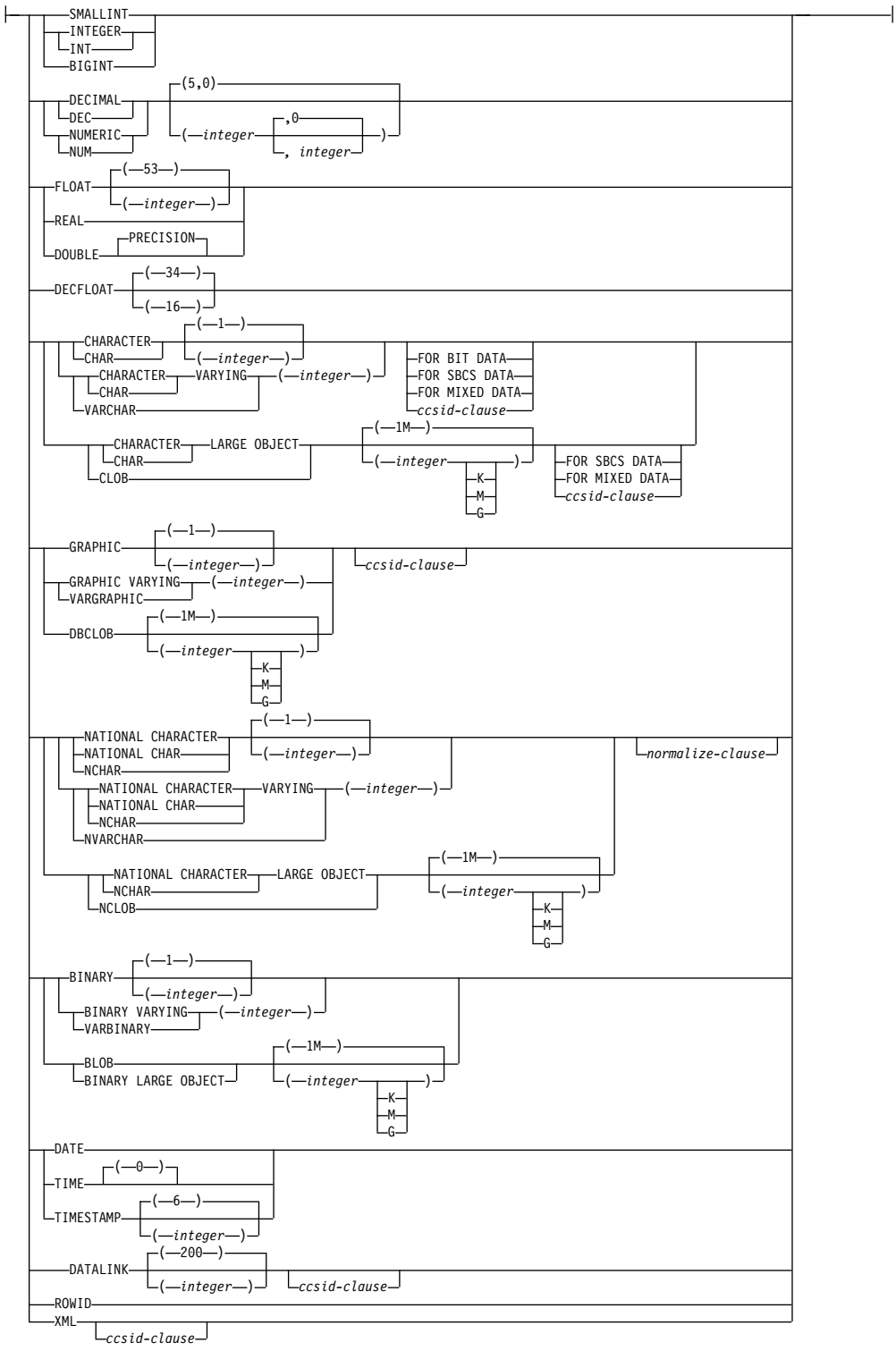
CAST の指定

CAST 指定は、*data-type* によって指定されたタイプにキャストされたキャスト・オペランド (第 1 オペランド) を戻します。



data-type:

built-in-type:



ccsid-clause:

normalize-clause:

CAST 指定は、*data-type* によって指定されたタイプにキャストされたキャスト・オペランド (第 1 オペランド) を戻します。いずれかのオペランドのデータ・タイプが特殊タイプの場合は、ステートメントの権限 ID によって保持される特権には、特殊タイプの USAGE 特権が含まれている必要があります。

expression
キャスト・オペランドが、NULL でもパラメーター・マーカーでもない式であることを指定します。結果は指定したターゲット・データ・タイプに変換される引数値です。

サポートされているキャストについては、 109 ページの『データ・タイプ間のキャスト』 に示されています。そこでは、最初の列がキャスト・オペランドのデータ・タイプ (ソース・データ・タイプ) を、そして上段のデータ・タイプが CAST 指定のターゲット・データ・タイプを表しています。キャストがサポートされていない場合は、エラーが戻されます。

文字またはグラフィック・ストリングを、長さが異なる文字またはグラフィック・ストリングにキャストすると、末尾ブランク以外の切り捨てが生じた場合には、警告が戻されます。

NULL
キャスト・オペランドが NULL 値であることを指定します。結果は、指定されたデータ・タイプ を持つ NULL 値です。

parameter-marker
パラメーター・マーカー (疑問符として指定) は、一般には式であると考えられますが、特別な意味を持つのでこのケースは別に記しています。キャスト・オペランドがパラメーター・マーカー の場合、指定したデータ・タイプ は、置き換えが指定したデータ・タイプ に割り当て可能であることの保証であると考えられます (記憶域割り当ての規則を使用します。 113 ページの『割り当ておよび比較』を参照)。 そのようなパラメーター・マーカーを、型付きパラメーター・マーカー と言います。タイプ・パラメーター・マーカーは、選択リストの DESCRIBE または列の割り当てのために、他のタイプ値と同様に取り扱われることになります。

data-type
結果のデータ・タイプを指定します。データ・タイプが修飾されていない場合は、適切なデータ・タイプを見つけるために SQL パスを使用します。詳しくは、 74 ページの『非修飾の関数、プロシージャー、特定名、タイプ、および変数』を参照してください。 *data-type* の説明は、 1238 ページの『CREATE TABLE』を参照してください。(オペレーティング・システム間の移植性のために、浮動小数点データ・タイプを使用する場合は、FLOAT の代わりに REAL または DOUBLE を使用してください。)

サポートされるデータ・タイプに関する制限は、指定されたキャストのオペランドに基づきます。

- キャストのオペランドが *expression* の場合、キャストのオペランドのデータ・タイプに基づいてサポートされているターゲット・データ・タイプに関しては、 110 ページの表 16 を参照してください。
- キャストのオペランドがキーワード NULL の場合は、ターゲット・データ・タイプはどのデータ・タイプでも構いません。
- キャストのオペランドがパラメーター・マーカーの場合は、ターゲット・データ・タイプはどのデータ・タイプでも構いません。データ・タイプが特殊タイプの場合、パラメーター・マーカーを使用するアプリケーションは、特殊タイプのソース・データ・タイプを使用することになります。データ・タイプが配列タイプの場合は、パラメーター・マーカーを使用して、ターゲットの配列データ・タイプの最大カーディナリティー以下のカーディナリティーで配列を記述する必要があります。パラメーター・マーカーのデータ・タイプは、ターゲットの配列データ・タイプのデータ・タイプと完全に一致していなければなりません。

CCSID 属性が指定されていない場合は、次のようになります。

- データ・タイプ が BINARY、VARBINARY、または BLOB の場合、 65535 の CCSID が使用されます。
- FOR BIT DATA が指定された場合、65535 の CCSID が使用されます。
- *expression* が文字ストリングで、データ・タイプ が CHAR、VARCHAR、または CLOB の場合は、次のようになります。
  - FOR SBCS DATA が指定されている場合
    - *expression* の CCSID が Unicode CCSID の場合は、ジョブのデフォルト CCSID に関連した 1 バイト CCSID が使用されます。
    - それ以外の場合は、*expression* の CCSID に関連した 1 バイト CCSID が使用されます。[45]
  - FOR MIXED DATA が指定されている場合
    - *expression* の CCSID が Unicode CCSID の場合は、ジョブのデフォルト CCSID に関連した混合バイト CCSID が使用されます。
    - それ以外の場合は、*expression* の CCSID に関連した混合バイト CCSID が使用されます。[45]
  - その他の場合は、
    - *expression* が SBCS データの場合、*expression* の CCSID が使用されます。[4645]
    - *expression* が混合データで、結果の長さ属性が 4 以上の場合、*expression* の CCSID が使用されます。
    - *expression* が DBCS 混用または DBCS 択一の混合データで、結果の長さ属性が 4 未満である場合、結果の CCSID は、混合データ CCSID に関連した SBCS CCSID です。
- *expression* がグラフィック・ストリングの場合や、*expression* がパラメーター・マーカーの場合に、*data-type* が CHAR、VARCHAR、CLOB のいずれかであれば、以下のようになります。

45. XSLTRANSFORM の場合は、*expression* の CCSID が 65535 であれば、ジョブのデフォルト CCSID が使用されます。

46. *expression* の CCSID が 65535 であれば、CLOB にキャストすることはできません。

- FOR SBCS DATA が指定された場合、ジョブのデフォルト CCSID に関連した単一バイト CCSID が使用されます。
- FOR MIXED DATA が指定された場合、ジョブのデフォルト CCSID に関連した混合バイト CCSID が使用されます。
- その他の場合は、
  - ジョブのデフォルト CCSID が SBCS データの場合、ジョブのデフォルト CCSID が使用されます。
  - ジョブのデフォルト CCSID が混合データで、結果の長さ属性が 4 以上の場合、ジョブのデフォルト CCSID が使用されます。
  - ジョブのデフォルト CCSID が DBCS 混用または DBCS 択一の混合データで、結果の長さ属性が 4 未満である場合、結果の CCSID は、混合データ CCSID に関連した SBCS CCSID です。

- *expression* が文字ストリングまたはパラメーター・マーカーで、*data-type* が GRAPHIC、VARGRAPHIC、DBCLOB、DATE、TIME、または TIMESTAMP の場合、CCSID 1200 が使用されます。
- *expression* がグラフィック・ストリングで、データ・タイプ が GRAPHIC、VARGRAPHIC、または DBCLOB の場合、*expression* の CCSID が使用されます。
- *data-type* が XML であれば、SQL_XML_DATA_CCSID QAQQINI 設定で指定されている CCSID 値が使用されます。詳しくは、 101 ページの『XML 値』を参照してください。
- それ以外の場合は、ジョブのデフォルト CCSID が使用されます。

CCSID 属性が指定された場合、データはその CCSID に変換されます。 NORMALIZED が指定された場合、データは正規化されます。 FOR MIXED DATA または混合 CCSID が指定されている場合には、結果の長さが 4 未満であってはなりません。

データ・タイプ間でサポートされるキャスト、およびデータ・タイプへキャストする際の規則についての詳細は、 109 ページの『データ・タイプ間のキャスト』を参照してください。

例

- アプリケーションでは、EMPLOYEE 表の SALARY 列 (DECIMAL(9,2) として定義) の整数部分にのみ関与します。次の CAST 指定は、SALARY 列を INTEGER に変換します。

```
SELECT EMPNO, CAST(SALARY AS INTEGER)
  FROM EMPLOYEE
```

- 2 つの特殊タイプが存在するものとします。 T_AGE は、SMALLINT をソースとしており、PERSONNEL 表の AGE 列のデータ・タイプです。 R_YEAR は、INTEGER をソースとしており、同じ表の RETIRE_YEAR 列のデータ・タイプです。次の UPDATE ステートメントを用意しました。

```
UPDATE PERSONNEL SET RETIRE_YEAR = ?
  WHERE AGE = CAST( ? AS T_AGE )
```

最初のパラメーターはタイプなしパラメーター・マーカーで、そのデータ・タイプは R_YEAR になります。この場合、パラメーター・マーカー値が特殊タイプに割り当てられるので、明示的な CAST 指定は必要ありません。

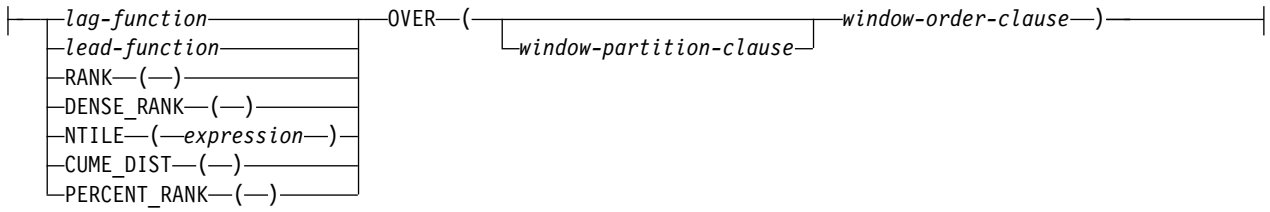
2 番目のパラメーター・マーカーは型付きパラメーター・マーカーで、それは特殊タイプ T_AGE にキャストされます。この場合、パラメーター・マーカー値が特殊タイプと比較されるので、明示的な CAST 指定が必要です。

OLAP 指定

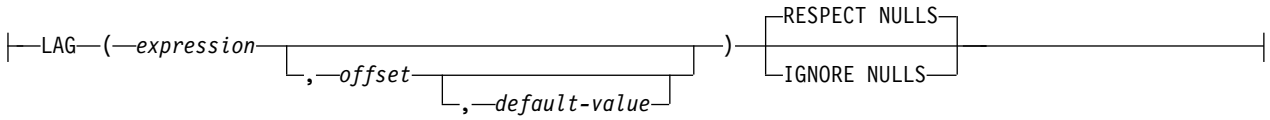
On-Line Analytical Processing (OLAP) 指定を使用すれば、ランキング、行番号付け、その他の集約関数情報を照会結果の中でスカラー値として戻すことができます。

OLAP-specification

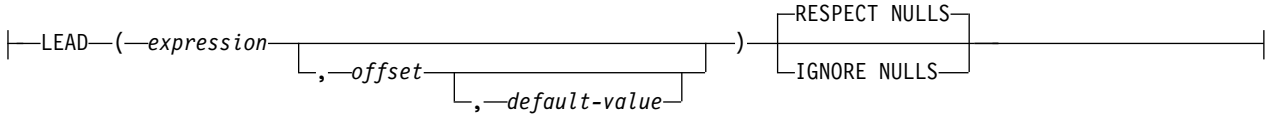
ordered-OLAP-specification:



lag-function:

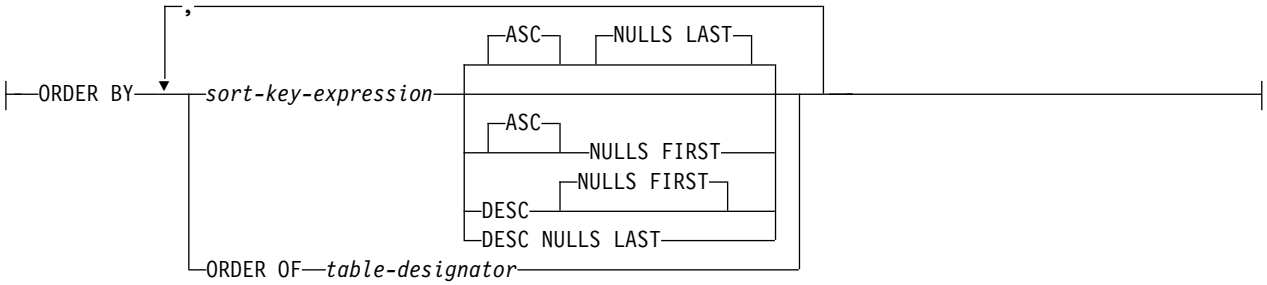


lead-function:

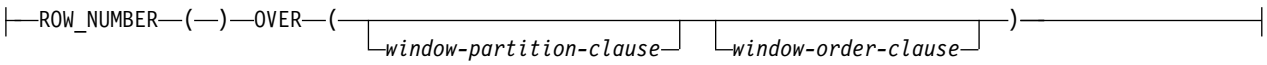


window-partition-clause:

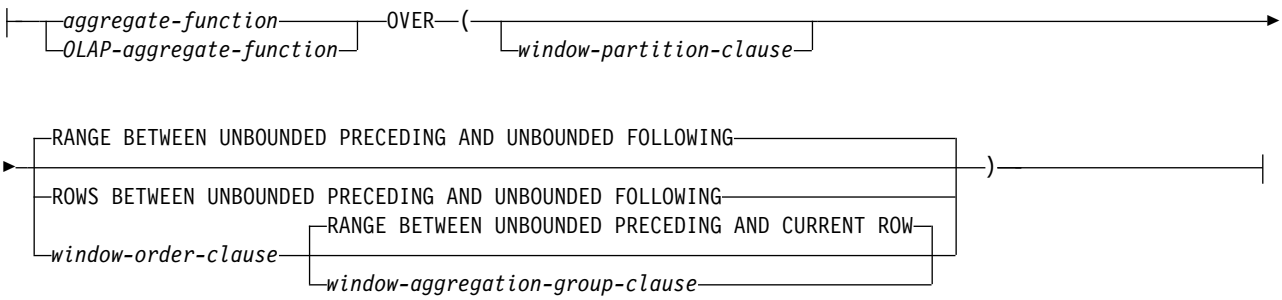
window-order-clause:



numbering-specification:



aggregation-specification:

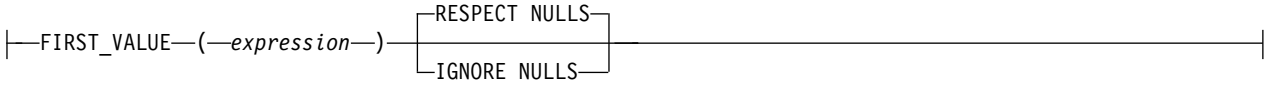


aggregate-function:

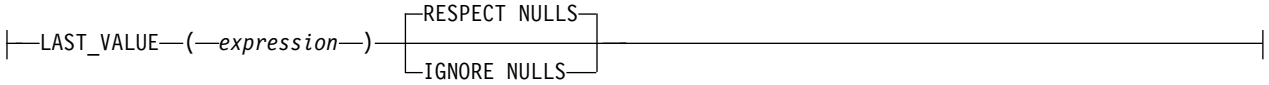
式

OLAP-aggregate-function:

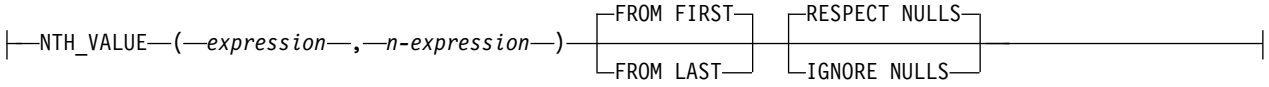
first-value-function:



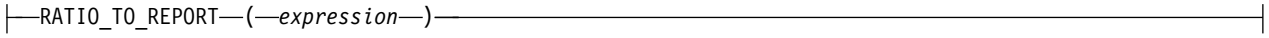
last-value-function:



nth-value-function:



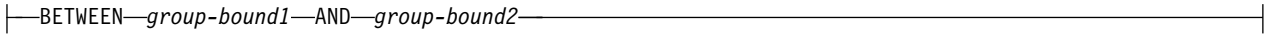
ratio-to-report-function:



window-aggregation-group-clause:

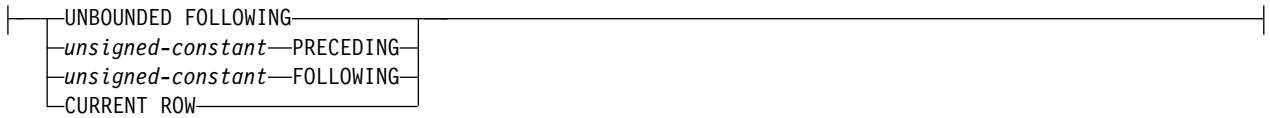
group-start:

group-between:

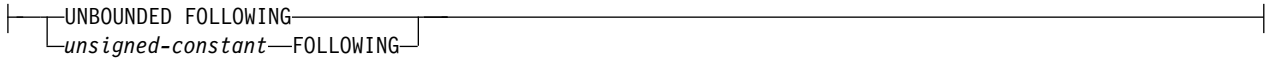


group-bound1:

group-bound2:



group-end:



OLAP の指定は、*select-clause* 内の式、または *select-statement* の ORDER BY 文節の中に含めることができます。OLAP の指定が適用される照会結果は、OLAP の指定を含む最内部の副選択の結果表です。OLAP の指定は、ウィンドウ関数 としばしば呼ばれます。

OLAP の指定は WHERE、VALUES、GROUP BY、HAVING、または SET 文節では無効であり、中間結合表の ON 文節における *join-condition* でも無効です。OLAP の指定は、*select-clause* で集約関数の引数として使用することはできません。

OLAP の指定の呼び出し時には、関数が適用される行とどの順序で適用されるかを定義するウィンドウが指定されます。

ordered-OLAP-specification
window-order-clause を必要とする OLAP 操作を指定します。

LAG または **LEAD**
現在行の前または後の行を使用して計算された *expression* 値を戻します。

offset は正の整数または正の bigint 定数でなければなりません。 *offset* が指定されていない場合、値 1 が使用されます。

default-value は、*expression* のタイプにキャスト可能な式でなければなりません。 *default-value* が指定されない場合、デフォルト値は NULL 値になります。

IGNORE NULLS が指定された場合は、*expression* 値が NULL 値である行はすべて計算時に考慮されません。

LAG LAG 関数は、現在の行から *offset* 行前にある行の式 の値を戻します。*window-partition-clause* が指定されている場合、 *offset* とは現在のパーティションに含まれる、現在の行から *offset* 行前のことです。

LEAD
LEAD 関数は、現在の行から *offset* 行後にある行の式 の値を戻します。*window-partition-clause* が指定されている場合、 *offset* とは現在のパーティションに含まれる、現在の行から *offset* 行後のことです。

offset が現在のパーティションの有効範囲を超える場合は、*default-value* が戻されます。

結果のデータ・タイプは、*expression* のデータ・タイプになります。結果が、NULL になることもあります。 IGNORE NULLS が指定されていて、ウィンドウ内のすべての値が NULL の場合、結果は NULL 値になります。

RANK または DENSE_RANK
ウィンドウ内の行の順序ランクを計算することを指定します。ウィンドウ内での順序に関しては区別できない行には同位が割り当てられます。ランキングの結果については、重複する値の結果の数値間に抜けが生じる状態と生じない状態で定義できます。

結果のデータ・タイプは BIGINT です。結果が NULL になることはありません。

RANK
行のランクが、その行の前にある厳密な行数に 1 を加算したものと定義されることを指定します。したがって、順番に関して 2 行以上の行が区別できない場合、順次のランクの番号付けには 1 つ以上の抜けが生じることになります。

DENSE_RANK
行のランクが、その行の前にある、順序に関して区別できる行の数に 1 を加算したものと定義されることを指定します。したがって、順次のランク番号付けには抜けはありません。

NTILE
ウィンドウ内の行の分位数ランクを計算することを指定します。

引数は、BIGINT にキャスト可能でなければなりません。 *expression* が SMALLINT、INTEGER、および BIGINT を戻さない場合は、関数を評価する前に BIGINT にキャストされます。ゼロより大きい値でなければなりません。 *expression* に *scalar-fullselect*、列参照、およびユーザー定義関数参照を含めてはなりません。

結果は、現在行の分位数ランクになります。結果内の分位数の数は引数によって決定され、分位数の数はウィンドウ内の行数を引数の値で割ることによって決定されます。ウィンドウ内の行数が引数で割り切れない場合、各分位数は少なくとも *n* 行を含み、分位数 1 から *m* はそれぞれ、*n+1* 行を含みます。

- *r* は、ウィンドウの行数です。
- *q* は、引数の値です。
- *m* = MOD (*r* , *q*)
- *n* = TRUNC (*r* , *q*)

結果のデータ・タイプは、引数の値に基づいて SMALLINT、INTEGER、または BIGINT になります。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL の場合、結果は NULL 値になります。

CUME_DIST
各行のパーセンタイル順位 (0 から 1 までの小数として表される) を決定する累積分布関数。デフォルトの行の昇順であれば、CUME_DIST は、現在行以下にランクする行の数 (現在行も含める) をパーティション内の合計行

数で割った値を計算します。 *window_order_clause* で降順を指定した場合、CUME_DIST は、現在行以上にランクする行の数をパーティション内の合計行数で割った値を計算します。

例えば、デフォルトの順序で、パーティション内に 10 行あり、それらの中の 6 行より下に現在行がランクする場合、CUME_DIST の結果は 0.7 (6 行 + 現在行 = 7 を 10 で割った値) になります。パーティション内の最下位行の CUME_DIST 値は、デフォルトの昇順の場合、1.0 になります。パーティション内に行が 1 つだけある場合も、CUME_DIST 値は 1.0 になります。

結果のデータ・タイプは DECFLOAT(34) です。結果が NULL 値になることはありません。

PERCENT_RANK

PERCENT_RANK 関数は、OLAP ウィンドウ内の行の相対パーセンタイル・ランク (0.0 から 1.0 までの値) を戻す分布関数です。 OLAP ウィンドウ内の行数が 1 より大きい場合、結果は次のように計算されます。

- OLAP ウィンドウの現在行の RANK から 1 を引いたものを、OLAP ウィンドウ内の行数から 1 を引いたもので除算する。

それ以外の場合、結果は 0.0 です。

結果のデータ・タイプは DECFLOAT(34) です。結果が NULL 値になることはありません。

numbering-specification

行ごとに連続番号を戻す OLAP 操作を指定します。

ROW_NUMBER

順序付けで定義されたウィンドウ内の行に関して、最初の行を 1 として始め、順番の行番号が計算されることを指定します。ORDER BY 文節がウィンドウ内で指定されていない場合、行番号は、副選択で戻される任意の順序で (*select-statement* 内の ORDER BY 文節によってではなく) 行に割り当てられます。

結果のデータ・タイプは BIGINT です。結果が NULL になることはありません。

window-partition-clause

その中で OLAP 演算が適用される区分を定義します。

PARTITION BY *(partitioning-expression,...)*

その中で OLAP 演算が適用される区分を定義します。区分化式 は、結果セットの区分化の定義に使用される式です。*partitioning-expression* で参照する各列名は、OLAP 指定を含んでいる副選択の結果表の列を明確に参照する必要があります。区分化式 にはスカラー全選択 あるいはどんな非 deterministic 関数や外部アクションを持つ関数も含めることはできません。

window-order-clause

OLAP の指定の値を決定するために使用されるパーティション内での行の順序を定義します。これによって結果表の順序は定義されません。

ORDER BY *(sort-key-expression,...)*

sort-key-expression は、ウィンドウ・パーティション内の行の順序付けを定

義するために使用する式です。*sort-key-expression* 内で参照される各列名は、OLAP の指定を含む副選択の結果表の列を明確に参照していなければなりません。ソート・キー式 にはスカラー全選択 あるいはどんな非 deterministic 関数や外部アクションのある関数も含めることはできません。

ソート・キー式 の長さ属性の合計は 3.5 ギガバイトを超えてはなりません。

ASC
sort-key-expression の値を昇順に使用することを指定します。

DESC
sort-key-expression の値を降順に使用することを指定します。

NULLS FIRST
ウィンドウの順序付けが、ソート順序の中で NULL 値をすべての非 NULL 値より先に考慮することを指定します。

NULLS LAST
ウィンドウの順序付けが、ソート順序の中で NULL 値をすべての非 NULL 値より後に考慮することを指定します。

ORDER OF *table-designator*
表指定子 で使用されているのと同じ順序付けを、副選択の結果表にも適用することを指定します。表指定子 に一致する表参照が FROM 文節を指定する副選択のその文節内になければならず、その表参照はネストされた表の式 または 共通表式 を識別するものでなければなりません。指定の表指定子 に対応する副選択 (または全選択) には、データに従属する ORDER BY 文節が含まれている必要があります。適用される順序付けは、ネストされた副選択 (または全選択) 内の ORDER BY 文節の列が外側の副選択 (全選択) に含まれており、それらの列が ORDER OF 文節の代わりに指定されている場合と同じです。

OLAP-aggregate-function
OLAP ウィンドウから単一値を計算する関数を指定します。

FIRST_VALUE または **LAST_VALUE**
OLAP ウィンドウ内の最初または最後の値を戻します。

IGNORE NULLS が指定された場合は、*expression* 値が NULL 値である行はすべて計算時に考慮されません。

FIRST_VALUE
結果は、OLAP ウィンドウ内の最初の行の *expression* 値です。

LAST_VALUE
結果は、OLAP ウィンドウ内の最後の行の *expression* 値です。

結果のデータ・タイプは、*expression* のデータ・タイプになります。結果が、NULL になることもあります。IGNORE NULLS が指定されていて、ウィンドウ内のすべての値が NULL の場合、結果は NULL 値になります。

NTH_VALUE
OLAP ウィンドウ内の *n* 番目の行の *expression* 値を戻します。

n-expression は、ゼロより大きい値を持つ整数定数または整数変数でなければなりません。

IGNORE NULLS が指定された場合は、*expression* 値が NULL 値である行はすべて計算時に考慮されません。

FROM FIRST が指定されている場合、*n* 番目の値は、OLAP ウィンドウの先頭から順に数えて計算されます。

FROM LAST が指定されている場合、*n* 番目の値は、OLAP ウィンドウの末尾から逆に数えて計算されます。

結果は、*n-expression* で判別された、OLAP ウィンドウ内の *n* 番目の値になります。

結果のデータ・タイプは、*expression* のデータ・タイプになります。結果が、NULL になることもあります。 *n-expression* が NULL の場合、結果は NULL 値になります。 IGNORE NULLS が指定されていて、ウィンドウ内のすべての値が NULL の場合、結果は NULL 値になります。

FIRST_VALUE(*expression*) は、NTH_VALUE(*expression*, 1) FROM FIRST と同等です。

LAST_VALUE(*expression*) は、NTH_VALUE(*expression*, 1) FROM LAST と同等です。

RATIO_TO_REPORT

OLAP ウィンドウにおける引数の合計に対する 1 つの引数の比率を戻します。例えば、以下の関数は同じことを意味します。

```
RATIO_TO_REPORT(expression) OVER (...)
CAST(expression AS DECFLOAT(34)) /
     SUM(CAST(expression as DECFLOAT(34))) OVER(...)
```

引数は、DECFLOAT(34) にキャスト可能な式でなければなりません。割り算は、DECFLOAT(34) を使用して実行されます。

結果のデータ・タイプは DECFLOAT(34) です。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

window-aggregation-group-clause

所定の行の集約グループとは、所定の行に対する関係に定義されている行のセットです (その所定の行のパーティションにおける行の順序を継承)。

window-aggregation-group-clause は集約グループを指定します。この文節が指定されておらず、*window-order-clause* も指定されていない場合、集約グループは、ウィンドウ・パーティションの全行で構成されます。ウィンドウ・パーティションの全行による集約グループは、RANGE 文節または ROWS 文節を使用して明示的に指定できます。

window-order-clause が指定されている一方で *window-aggregation-group-clause* が指定されていない場合、ウィンドウ集約グループは、所定の行のパーティションで所定の行に先行するすべての行か、*window-order-clause* によって定義されるウィンドウ・パーティションのウィンドウの順序付けで所定の行と同位であるすべての行で構成されます。

ROW
行をカウントすることによって集約グループが定義されることを指定します。

RANGE
ソート・キーからのオフセットによって集約グループが定義されることを指定します。

group-start
集約グループの開始点を指定します。集約グループの終わりは CURRENT ROW です。*group-start* を指定することは、BETWEEN *group-start* AND CURRENT ROW として *group-between* を指定することに相当します。

group-between
集約グループが ROWS または RANGE のいずれかに基づいて開始および終了することを指定します。

group-end
集約グループの終了点を指定します。集約グループの開始は CURRENT ROW です。*group-end* を指定することは、*group-between* を BETWEEN CURRENT ROW AND *group-end* として指定することと等価です。

UNBOUNDED PRECEDING
現在行に先行するパーティション全体を集約グループに含めることを指定します。ROWS 文節または RANGE 文節のいずれかと組み合わせて指定できます。現在行に先行するパーティション全体を集約グループに含める場合は、*window-order-clause* で *sort-key-expressions* を複数使用することができます。

UNBOUNDED FOLLOWING
現在行に続くパーティション全体を集約グループに含めることを指定します。ROWS 文節または RANGE 文節のいずれかと組み合わせて指定できます。現在行に続くパーティション全体を集約グループに含める場合は、*window-order-clause* で *sort-key-expressions* を複数使用することができます。

CURRENT ROW
集約グループが現在行に基づいて開始または終了することを指定します。ROWS が指定された場合、current row が集約グループ境界です。RANGE を指定すると、*sort-key-expression* に指定した値を現在行とする行のセットが集約グループ境界に含まれます。 *group-bound-1* で *unsigned-constant* FOLLOWING が指定されている場合は、この文節を *group-bound-2* に指定できません。

unsigned-constant **PRECEDING**
現在行に先行する範囲または行数のいずれかを指定します。ROWS が指定されている場合、*unsigned-constant* は、ゼロ、あるいは行数を示す正整数または正の bigint である必要があります。 RANGE が指定されている場合、*unsigned-constant* のデータ・タイプは *window-order-clause* の *sort-key-expression* のデータ・タイプと比較可能である必要があります。許可される *sort-key-expression* は 1 つのみであり、 *sort-key-expression* のデータ・タイプは減算できるデータ・タイプである必要があります。*group-bound-1* が CURRENT ROW または *unsigned-constant* FOLLOWING の場合、この文節を *group-bound-2* で指定することはできません。

unsigned-constant **FOLLOWING**
現在行の後続の範囲または行数のいずれかを指定します。ROWS が指定されている場合、*unsigned-constant* は、ゼロ、あるいは行数を示す正整数または正の bigint である必要があります。 RANGE が指定されている場合、*unsigned-constant* のデータ・タイプは *window-order-clause* の *sort-key-expression* のデータ・タイプと比較可能である必要があります。許可される *sort-key-expression* は 1 つのみであり、 *sort-key-expression* のデータ・タイプは加算できるデータ・タイプである必要があります。

注

比較: パーティション化および順序付けは、 113 ページの『割り当ておよび比較』で説明されている比較規則にしたがって行われます。

照合順序: OLAP 式を含むステートメントの実行時に *HEX 以外の照合順序が有効な場合で、しかも区分化式 またはソート・キー式 が SBCS データ、混合データ、または Unicode データの場合、結果は重み付けされた値を使用して決定されます。この重み付けされた値は、区分化式 およびソート・キー式 に照合順序を適用して得られます。

列マスク: OLAP 指定の *partitioning-expression* または *sort-key-expression* で参照されている列が、列マスクを持つように定義されている場合、その列マスクは適用されません。

制約事項: 照会に以下の指定がある場合、OLAP の指定は許可されません。

- 分散表
- 読み取りトリガーを指定する表
- 複数の物理ファイル・メンバー上に構築された論理ファイル

決定論: OLAP の指定は、非決定的です。

代替構文:

- DENSE_RANK の代わりに DENSERANK を指定できます。
- ROW_NUMBER の代わりに ROWNUMBER を指定できます。
- LAG、LEAD、FIRST_VALUE、および LAST_VALUE への *string-constant* の最後の引数として IGNORE NULLS または RESPECT NULLS を指定できます。

例

- 給与の合計が $30,000 を超える (給与にボーナスを足したものに基づいて) 従業員のランキングを姓の順番で表示します。

```
SELECT EMPNO, LASTNAME, FIRSTNME, SALARY+BONUS AS TOTAL_SALARY,
    RANK() OVER (ORDER BY SALARY+BONUS DESC) AS RANK_SALARY
  FROM EMPLOYEE
  WHERE SALARY+BONUS > 30000
  ORDER BY LASTNAME
```

結果を給与のランキングで順序付けするには、ORDER BY LASTNAME を以下で置き換えることに注意してください。

```
ORDER BY RANK_SALARY
```

または

```
ORDER BY RANK() OVER (ORDER BY SALARY+BONUS DESC)
```

- 部門ごとの給与合計の平均によって部門をランク付けします。

```
SELECT WORKDEPT, AVG(SALARY+BONUS) AS AVG_TOTAL_SALARY,
    RANK() OVER (ORDER BY AVG( SALARY+BONUS) DESC) AS RANK_AVG_SAL
  FROM EMPLOYEE
  GROUP BY WORKDEPT
  ORDER BY RANK_AVG_SAL
```

- 部門内の従業員を教育レベルによってランク付けします。部門内に同位の従業員が複数いても、その次のランキング値が増加することにはなりません。

```
SELECT WORKDEPT, EMPNO, LASTNAME, FIRSTNME, EDLEVEL,
    DENSE_RANK() OVER (PARTITION BY WORKDEPT ORDER BY EDLEVEL DESC)
      AS RANK_EDLEVEL
  FROM EMPLOYEE
  ORDER BY WORKDEPT, LASTNAME
```

- 照会の結果に行番号を含めます。

```
SELECT ROW_NUMBER() OVER (ORDER BY WORKDEPT, LASTNAME ) AS NUMBER,
    LASTNAME, SALARY
  FROM EMPLOYEE
  ORDER BY WORKDEPT, LASTNAME
```

- 給与の高い上位 5 人の従業員をリストします。

```
SELECT EMPNO, LASTNAME, FIRSTNME, TOTAL_SALARY, RANK_SALARY
  FROM (SELECT EMPNO, LASTNAME, FIRSTNME, SALARY+BONUS AS TOTAL_SALARY,
          RANK() OVER (ORDER BY SALARY+BONUS DESC) AS RANK_SALARY
        FROM EMPLOYEE) AS RANKED_EMPLOYEE
  WHERE RANK_SALARY < 6
  ORDER BY RANK_SALARY
```

ランクが WHERE 文節で使用される前に、ランキングを含む結果の最初の計算にはネストされた表の式が使用されたことに注意してください。共通表式を使うこともできます。

- NTH_VALUE を使用して、株 ABC の上位 3 件の株価を求めます。

```
SELECT Symbol, StockDate, Price,
      FIRST_VALUE(Price) OVER (PARTITION BY Symbol ORDER BY StockDate) AS FIRST_PRICE,
      NTH_VALUE(Price, 2) OVER (PARTITION BY Symbol ORDER BY StockDate) AS SECOND_PRICE,
      NTH_VALUE(Price, 3) OVER (PARTITION BY Symbol ORDER BY StockDate) AS THIRD_PRICE
FROM DailyStockData
WHERE StockDate BETWEEN CURRENT DATE - 1 MONTH AND CURRENT DATE
      AND Symbol = 'ABC'
```

- NTILE を使用して、四分位数ランクを計算します。

```
SELECT proc_id, total_sales,
       NTILE(4) OVER (ORDER BY total_sales DESC) AS Quartile
FROM Sales
```

- 2005 年における株「ABC」および「XYZ」の 30 日間移動平均を計算します。

```
WITH V1(SYMBOL, TRADINGDATE, MOVINGAVG30DAY) AS
(
 SELECT SYMBOL, TRADINGDATE,
    AVG(CLOSINGPRICE) OVER (PARTITION BY SYMBOL
                ORDER BY TRADINGDATE
                ROWS BETWEEN 29 PRECEDING AND CURRENT ROW)
  FROM DAILYSTOCKDATA
  WHERE SYMBOL IN ('ABC', 'XYZ')
      AND TRADINGDATE BETWEEN DATE('2005-01-01') - 2 MONTHS AND '2005-12-31'
)
```

```
SELECT SYMBOL, TRADINGDATE, MOVINGAVG30DAY
 FROM V1
 WHERE TRADINGDATE BETWEEN '2005-01-01' AND '2005-12-31'
 ORDER BY SYMBOL, TRADINGDATE
```

- 各従業員の給与と、その従業員の部門の給与の中央値との差を表示します。

```
SELECT EMPNO, WORKDEPT, SALARY,
       SALARY - (MEDIAN(SALARY) OVER (PARTITION BY WORKDEPT))
FROM EMPLOYEE
ORDER BY WORKDEPT
```

- 各従業員の給与と、その従業員の部門内の給与の 90 番目のパーセンタイルとの差を表示します。

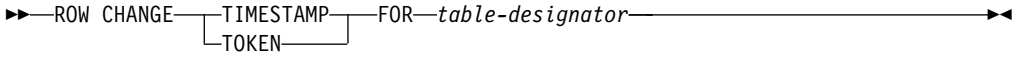
```
SELECT EMPNO, WORKDEPT, SALARY,
       SALARY - (PERCENTILE_CONT(0.9) WITHIN GROUP (ORDER BY SALARY)
                                      OVER (PARTITION BY WORKDEPT))
FROM EMPLOYEE
ORDER BY WORKDEPT
```

- 部門内の各従業員の給与の累積分布と相対パーセンタイル・ランクを求めます。

```
SELECT EMPNO, WORKDEPT, SALARY,
       CAST(CUME_DIST() OVER (PARTITION BY WORKDEPT ORDER BY SALARY) AS DECIMAL(4,3))
         AS CUME_DIST,
       CAST(PERCENT_RANK() OVER (PARTITION BY WORKDEPT ORDER BY SALARY)
            AS DECIMAL(4,3))
         AS PERCENT_RANK
FROM EMP
ORDER BY WORKDEPT, SALARY
```

ROW CHANGE 式

ROW CHANGE 式は、行に対する最終変更を表すトークンまたはタイム・スタンプを戻します。



ROW CHANGE TIMESTAMP

行が最後に変更された時刻を表すタイム・スタンプを返すことを指定します。行が変更されていない場合、結果は初期値が挿入された時刻となります。表に行変更タイム・スタンプがない場合、この式は使用できません。

ROW CHANGE TOKEN

行の変更シーケンス内での相対点を表す BIGINT 値であるトークンを返すことを指定します。行が変更されていない場合、結果は初期値が挿入されたときを表すトークンとなります。

FOR*table-designator*

副選択の表指定子を指定します。表指定子の詳細については、 166 ページの『表指定子』を参照してください。SQL 命名規則では、表名は修飾できます。システム命名規則では、表名は修飾できません。表指定子は、表関数またはデータ変更表参照 を識別することはできません。表指定子がビューまたはネストされた表の式を識別する場合、この式はその基本表の ROW CHANGE TOKEN または ROW CHANGE TIMESTAMP を返します。

結果が NULL 値になる場合もあります。これらの式は deterministic ではありません。

例

- 前日に変更されたすべての行を検出します。

```
SELECT *
  FROM  ORDERS
  WHERE ROW CHANGE TIMESTAMP FOR  ORDERS > CURRENT TIMESTAMP  - 24 HOURS
```

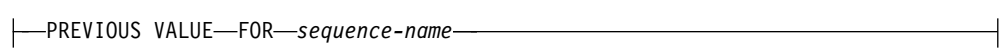
シーケンス参照

シーケンスは、NEXT VALUE および PREVIOUS VALUE 式を使用し、シーケンスの名前を指定することによって参照されます。

sequence-reference:

nextval-expression:

prevval-expression:



シーケンスは、NEXT VALUE および PREVIOUS VALUE 式を使用し、シーケンスの名前を指定することによって参照されます。

nextval-expression
NEXT VALUE 式は、指定したシーケンスの次の値を生成して戻します。NEXT VALUE 式でシーケンスの名前を指定すると、シーケンスの新しい値が生成されます。ただし、照会内で同じシーケンス名を指定する NEXT VALUE 式の複数インスタンスがある場合は、結果の各行につき一度だけシーケンス値が増分され、 NEXT VALUE のすべてのインスタンスが、結果の行に対して同じ値を戻します。 NEXT VALUE は、シーケンス値の増分を生じさせるので、それは外部アクションを伴う、非 deterministic 式です。

シーケンスの次の値が生成されたときに、シーケンスの論理範囲に対して、昇順シーケンスの最大値または降順シーケンスの最小値が超過しており、 NO CYCLE オプションが有効である場合、エラーが戻されます。

NEXT VALUE 式の結果のデータ・タイプおよび長さ属性は、指定したシーケンスのデータ・タイプおよび長さ属性と同じです。結果が NULL になることはありません。

prevval-expression
PREVIOUS VALUE 式は、現行アプリケーション・プロセス内での直前のステートメントについて、指定したシーケンスに最近生成された値を戻します。この値は、PREVIOUS VALUE 式を使用し、シーケンスの名前を指定することによって、繰り返し参照することができます。単一ステートメント内で同じシーケンス名を指定している PREVIOUS VALUE 式の複数インスタンスがある場合もあり、それらはすべて同じ値を戻します。

PREVIOUS VALUE 式を使用できるのは、同じシーケンス名を指定している NEXT VALUE が、現行アプリケーション・プロセス内で既に参照されている場合だけです。

PREVIOUS VALUE 式の結果のデータ・タイプおよび長さ属性は、指定したシーケンスのデータ・タイプおよび長さ属性と同じです。結果が NULL になることはありません。

sequence-name
参照されるシーケンスを識別します。シーケンス名 は、現行サーバーに存在するシーケンスを示すものでなければなりません。

注

許可**:** シーケンスがステートメント内で参照される場合、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別されるシーケンスに対して、
  - シーケンスでの USAGE 特権、および
  - シーケンスが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

SQL 特権に対応するシステム権限については、『シーケンスへの権限を検査する際の対応するシステム権限』を参照してください。

NEXT VALUE による値の生成**:** シーケンスに値が生成される際に、その値は消費され、次回に値が要求されるときには、新しい値が生成されます。このことは、NEXT VALUE 式を含むステートメントが失敗したりロールバックされた場合にも同様です。

PREVIOUS VALUE の有効範囲**:** PREVIOUS VALUE 値は、現行セッションでシーケンスに次の値が生成されるか、シーケンスがドロップまたは変更されるか、またはアプリケーション・セッションが終了するまで存続します。この値は COMMIT または ROLLBACK ステートメントの影響を受けません。PREVIOUS VALUE の値を直接設定することはできず、この値はシーケンスに NEXT VALUE 式を実行した結果として得られます。

アプリケーションや製品で特にパフォーマンス上の理由から通常使用されるのは、一群の接続を管理し、トランザクションを任意の接続に経路指定するという技法です。こうした状況では、シーケンスに対する PREVIOUS VALUE はそのトランザクションが終了するまでの間しか使用できません。

ユニーク・キー値としての使用**:** 2 つの別々の表内のユニーク・キー値として、同じシーケンス番号を使用することができます。そのためには、最初の行の NEXT VALUE 式と (これでシーケンス値が生成される)、他の行の PREVIOUS VALUE 式でシーケンス番号を参照します。 (PREVIOUS VALUE のインスタンスは、現行セッションで最近生成されたシーケンス値を参照します。) この点を以下に示します。

```
INSERT INTO ORDER (ORDERNO, CUSTNO)
  VALUES (NEXT VALUE FOR ORDER_SEQ, 123456)

INSERT INTO LINE_ITEM (ORDERNO, PARTNO, QUANTITY)
  VALUES (PREVIOUS VALUE FOR ORDER_SEQ, 987654, 1)
```

NEXT VALUE および **PREVIOUS VALUE** の使用許可**:** NEXT VALUE および PREVIOUS VALUE 式は、以下の場所に指定することができます。

- ステートメントに DISTINCT キーワード、GROUP BY 文節、ORDER BY 文節、UNION キーワード、INTERSECT キーワード、または EXCEPT キーワードが含まれていない場合、 SELECT ステートメントまたは SELECT INTO ステートメントの選択文節 内
- 全選択 の VALUES 文節内 (NEXT VALUE は使用できません)
- INSERT ステートメントの VALUES 文節内。
- INSERT ステートメントの全選択の選択文節 内。
- 検索または定位置の UPDATE ステートメントの SET 文節内。ただし、NEXT VALUE は、SET 文節内の式の副選択の選択文節 には指定できません。

  PREVIOUS VALUE 式は、UPDATE ステートメント内の SET 文節内の任意の場所に指定できますが、 NEXT VALUE 式を指定できるのは、それが式の全選択の選択文節 内にあるのでなければ、SET 文節内だけです。例えば、シーケンス式の次のような使用はサポートされます。

  ```
  UPDATE T SET C1 = (SELECT PREVIOUS VALUE FOR S1 FROM T)

  UPDATE T SET C1 = PREVIOUS VALUE FOR S1

  UPDATE T SET C1 = NEXT VALUE FOR S1
  ```

  シーケンス式の次のような使用はサポートされません。

  ```
  UPDATE T SET C1 = (SELECT NEXT VALUE FOR S1 FROM T)
  ```
- 割り当てステートメント 内。ただし、式の全選択の選択文節 内を除きます。シーケンス式の次のような使用はサポートされます。

  ```
  SET :ORDERNUM = NEXT VALUE FOR INVOICE

  SET :ORDERNUM = PREVIOUS VALUE FOR INVOICE
  ```

  シーケンス式の次のような使用はサポートされません。

  ```
  SET :X = (SELECT NEXT VALUE FOR S1 FROM T)

  SET :X = (SELECT PREVIOUS VALUE FOR S1 FROM T)
  ```
- VALUES または VALUES INTO ステートメント内。ただし、式の全選択の選択文節 内を除きます。
- CREATE PROCEDURE ステートメントの *SQL* ルーチン本体 内。
- CREATE TRIGGER ステートメントの *SQL* トリガー本体 内 (PREVIOUS VALUE は許可されません)。
- CALL ステートメントの引数リスト内。
- CREATE PROCEDURE または CREATE FUNCTION のデフォルト式内。NEXT VALUE または PREVIOUS VALUE を含んでいるデフォルトのある関数は、NEXT VALUE または PREVIOUS VALUE を直接指定できる場所でのみ使用できます。

NEXT VALUE および **PREVIOUS VALUE** の使用上の制限**:** NEXT VALUE および PREVIOUS VALUE 式は、以下の場所には指定できません。

- CREATE TABLE または ALTER TABLE ステートメント内のマテリアライズ照会表の定義内。
- CHECK 制約内。
- ビュー定義内。
- CREATE INDEX ステートメント内。
- CREATE FUNCTION ステートメントの *SQL* ルーチン本体 内。

加えて、NEXT VALUE 式は、以下の場所には指定できません。

- CASE 式
- 集約関数のパラメーター・リスト
- 明示的に許可されている場合を除き、コンテキスト内の副照会
- 外側の SELECT に DISTINCT 演算子または GROUP BY 文節が含まれている場合の SELECT ステートメント
- 外側の SELECT に、UNION、INTERSECT、または EXCEPT 演算子を使った別の SELECT ステートメントが結合している場合の SELECT ステートメント
- OFFSET 文節を含む SELECT ステートメント。
- 結合の結合条件
- ネストされた表の式
- 表関数のパラメーター・リスト
- UPDATE ステートメントの SET 文節内の式の全選択の選択文節
- 最外部の SELECT ステートメントか、DELETE または UPDATE ステートメントの WHERE 文節
- 最外部の SELECT ステートメントの ORDER BY 文節
- SQL ルーチン内の IF、WHILE、DO . . . UNTIL、または CASE ステートメント

カーソルを使ったシーケンス式の使用**:** 通常は、 SELECT NEXT VALUE FOR ORDER_SEQ FROM T1 が生成する結果表には、シーケンス ORDER_SEQ から生成される値が、T1 から検索される行の数と同数、含まれます。カーソルの SELECT ステートメント内の NEXT VALUE 式への参照は、結果表の行に対して生成される値を参照します。行が検索されるたびに、NEXT VALUE 式に対してシーケンス値が生成されます。

DRDA 環境においてブロッキングがクライアントで行われる場合、アプリケーションの FETCH ステートメントの処理の前に、 Db2 サーバーでシーケンス値が生成される可能性があります。クライアント・アプリケーションが、データベースから検索されたすべての行を明示的に FETCH するのでない場合、そのアプリケーションは、シーケンスの生成される値すべてを確認することはできません (FETCH されなかった行と同数分)。それらの値はシーケンス内のギャップを構成することになります。

カーソルの SELECT ステートメント内の PREVIOUS VALUE 式への参照は、OPEN の際に評価されます。言い換えると、カーソルの SELECT ステートメント内の PREVIOUS VALUE 式への参照は、カーソルのオープンよりも前に、指定されたシーケンスに対してこのアプリケーション・プロセスにより生成された最後の値を参照するということです。OPEN の際にいったん評価されると、カーソルの本

体の中で PREVIOUS VALUE によって戻される値は、FETCH ごとに変化することはありません。そのことは、カーソルの本体の中で NEXT VALUE が呼び出されても変りません。カーソルが閉じられた後は、PREVIOUS VALUE の値は、アプリケーション・プロセスにより生成された最後の NEXT VALUE になります。

代案の構文: キーワード NEXTVAL および PREVVAL を、それぞれ NEXT VALUE および PREVIOUS VALUE の代わりに使用することができます。

例

- ORDER という表があり、ORDER_SEQ というシーケンスが次のように作成されると想定します。

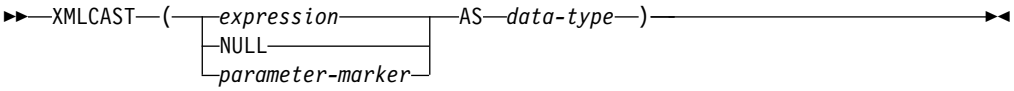
```
CREATE SEQUENCE ORDER_SEQ
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO CYCLE
  CACHE 24
```

NEXT VALUE 式を使って ORDER_SEQ シーケンス番号を生成する方法を、以下のいくつかの例で示します。

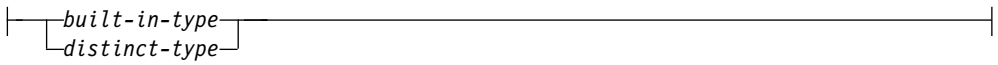
```
INSERT INTO ORDER (ORDERNO, CUSTNO)
  VALUES (NEXT VALUE FOR ORDER_SEQ, 123456)

UPDATE ORDER
  SET ORDERNO = NEXT VALUE FOR ORDER_SEQ
  WHERE CUSTNO = 123456

VALUES NEXT VALUE FOR ORDER
  INTO :HV_SEQ
```

XMLCAST 指定

XMLCAST 指定は、XML *data-type* にキャストされたキャスト・オペランド (第 1 オペランド) を戻します。

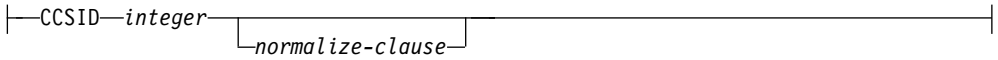


data-type:

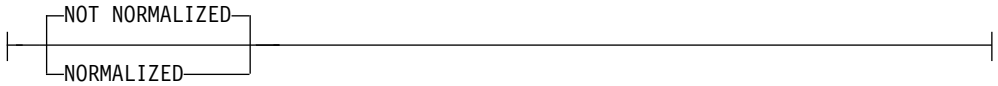


built-in-type:

ccsid-clause:



normalize-clause:



expression
キャスト・オペランドが、NULL でもパラメーター・マーカーでもない式であることを指定します。 XML データ・タイプでなければなりません。結果は、指定した XML ターゲット・データ・タイプと CCSID に変換された引数値です。

NULL
キャスト・オペランドが NULL 値であることを指定します。結果は、XML データ・タイプの NULL 値です。

parameter-marker
パラメーター・マーカー (疑問符として指定) は、一般には式であると考えられますが、特別な意味を持つのでこのケースは別に記しています。キャスト・オペランドが *parameter-marker* の場合、XML データ・タイプは、置換値が XML データ・タイプに割り当て可能であることの保証と考えられます。そのようなパラメーター・マーカーを、型付きパラメーター・マーカー と言います。タイプ・パラメーター・マーカーは、選択リストの DESCRIBE または列の割り当てのために、他のタイプ値と同様に取り扱われることになります。

data-type
結果のデータ・タイプを指定します。このデータ・タイプは、XML であるか、XML に基づく特殊タイプでなければなりません。

CCSID 属性を指定しないと、SQL_XML_DATA_CCSID QAQQINI 設定で指定された CCSID 値が使用されます。詳しくは、 101 ページの『XML 値』を参照してください。

CCSID 属性が指定された場合、データはその CCSID に変換されます。NORMALIZED が指定された場合、データは正規化されます。

例

- NULL の XML 値を作成します。

```
VALUES(XMLCAST(NULL AS XML))
```

述部

述部 では、特定の値、行、グループについて、真、偽、不明のいずれかになる条件を指定します。

以下の規則は、すべてのタイプの述部に適用されます。

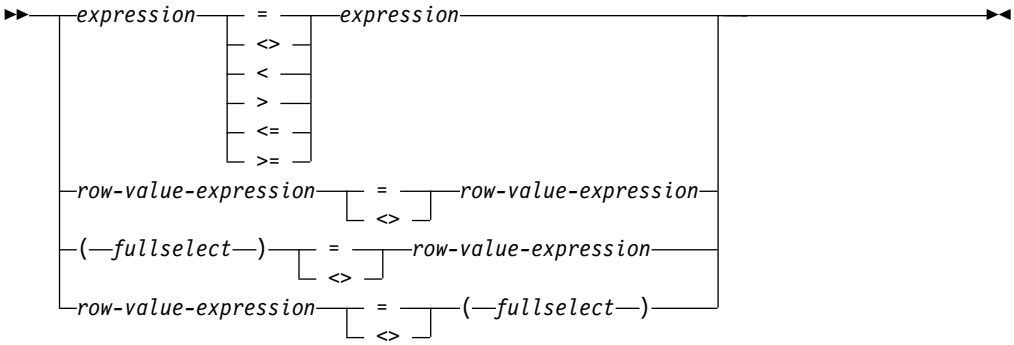
- 述部のオペランドである式の後に述部が評価されます。
- 同じ述部に指定する値には、すべて互換性がなければなりません。
- 変数の値は NULL の可能性があります (すなわち、変数は負の標識変数を持つ可能性があります)。
- 複数のオペランドを持つ述部のオペランドの CCSID 変換は、 129 ページの『比較の際の変換規則:』に従って行われます。
- データ・リンク値の使用は、NULL 述部に限定されています。

行値式: いくつかの述部 (基本、多値比較、および IN) のオペランドを行値式 にできます。

行値式 は 1 つ以上の列値からなる単一行を戻します。値は、式のリストとして指定することができます。行値式 によって戻される列の数は、そのリストで指定する式の数と同じです。

基本述部

基本述部 は 2 つの値を比較、または値の集合を別の値の集合と比較します。



注:

[1] 他の比較演算子もサポートされています。[47]

単一の式 が演算子の左側に指定されている場合、別の式 が右側に指定されていなければなりません。対応する式のデータ・タイプは互換性がなければなりません。左側の式の値は右側の式の値と比較されます。一方のオペランドの値が NULL の場合は、述部の結果は未知になります。 それ以外の場合、結果は真または偽のいずれかになります。

行値式 が演算子 (= または <>) の左側に指定されており、別の行値式 が演算子の右側に指定されている場合、両方の行値式 に同じ数の値式がなければなりません。行値式 の対応する式のデータ・タイプには互換性がある必要があります。左側の各式の値は、対応する右側の式の値と比較されます。

行値式 が指定されており、全選択 も指定されている場合:

- SELECT * は、全選択 の最外部の選択リスト内には許可されません。
- 全選択 の結果表は、行値式 と同じ数の列を持たなければなりません。行値式 と全選択 の対応する式のデータ・タイプには互換性がなければなりません。左側の各式の値は、対応する右側の式の値と比較されます。

この述部の結果は、以下のように演算子によって異なります。

- 演算子が = である場合、述部の結果は次のようになります。
  - 対応する値式のすべての対が真と評価される場合、結果は真。

47. 基本述部と比較述部では、次の形式の比較演算子もサポートされます。つまり、!=、!<、!>、¬=、¬<、および ¬> がサポートされます。これらのプロダクト特定の形式の比較演算子は、こうした演算子を使用している既存の SQL ステートメントをサポートすることだけが目的であり、新規の SQL ステートメントを作成するときに使用することはお勧めできません。

    キーボードによっては、NOT (¬) の記号の代わりに 16 進数値を使用しなければなりません。この 16 進数値は、使用するキーボードによって異なります。NOT 記号 (¬) または特定の国または地域でその場所に使う必要がある文字は、あるデータベース・サーバーから別のデータベース・サーバーに渡されるステートメントで構文解析エラーを引き起こすことがあります。問題が起きるのは、ステートメントがある種の組み合わせのソース CCSID とターゲット CCSID を使って文字変換を行う場合です。この問題を避けるために、NOT 記号を含む演算子は、同等の演算子で置き換えてください。例えば、「¬=」は「<>」に、「¬>」は「<=」に、「¬<」は「>=」に置き換えます。

  - 対応する値式の対で偽であると評価されるものが 1 対でもある場合、結果は偽。
  - それ以外の場合、結果は不明 (これは、対応する値式の少なくとも 1 つの比較が NULL 値のために不明であり、対応する値式の対で偽と評価されるものがない場合)。
- 演算子が <> である場合、述部の結果は次のようになります。
  - 対応する値式の対で真であると評価されるものが 1 対でもある場合、結果は偽。
  - 対応する値式のすべての対が偽と評価される場合、結果は真。
  - それ以外の場合、結果は不明 (これは、対応する値式の少なくとも 1 つの比較が NULL 値のために不明であり、対応する値式の対で真と評価されるものがない場合)。

述部の対応するオペランドが SBCS データ、混合データ、または Unicode データであり、そのステートメントが実行される時点で有効な照合順序が *HEX でない場合は、そのオペランドの比較は、オペランドの重み付けされた値を使用して行われます。値の重み付けは、該当の照合順序に基づいています。

値 *x* および *y* について、次のような関係が成り立ちます。

述部	下記の場合にのみ真となる
x **=** *y*	*x* は *y* に等しい
x **<>** *y*	*x* は *y* に等しくない
x **<** *y*	*x* は *y* より小さい
x **>** *y*	*x* は *y* より大きい
x **>=** *y*	*x* は *y* より大きいか、または等しい
x **<=** *y*	*x* は *y* より小さいか、または等しい

例

例 1

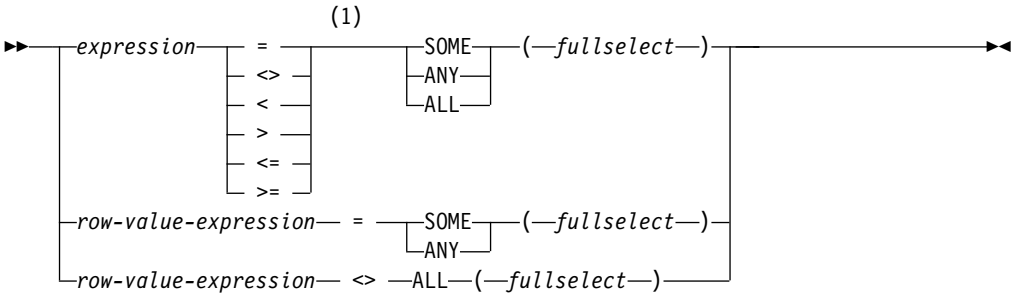
```
EMPNO = '528671'

PRTSTAFF <> :VAR1

SALARY + BONUS + COMM < 20000

SALARY > (SELECT AVG(SALARY)
          FROM EMPLOYEE)
```

例 2: 'OP1000' プロジェクトに責任を持つ従業員の姓、名、および給与をリストします。

```
SELECT  LASTNAME, FIRSTNME, SALARY
  FROM  EMPLOYEE X
  WHERE  EMPNO = ( SELECT  RESPEMP
                   FROM PROJA1 Y
                   WHERE MAJPROJ = 'OP1000' )
```

多値比較述部

多値比較述部 は、ある値または複数の値を値の集合と比較します。



注:

1 他の比較演算子もサポートされています。[47]

式 が指定されている場合、全選択は単一の結果列を戻す必要があります。全選択は、NULL か否かに関係なくいくつかの値を戻すことがあります。結果は、以下のように指定される演算子によって異なります。

- ALL を指定すると、述部の結果は次のようになります。
  - 全選択の結果が空の場合、または指定された関係が全選択によって戻されたすべての値について真である場合は、結果は真。
  - 全選択によって戻された値の少なくとも 1 つについて指定された関係が偽である場合、結果は偽。
  - 全選択によって戻された値の中に指定された関係が偽でないものがあり、しかも、少なくとも 1 つの比較が NULL 値により不明になった場合、結果は不明。
- SOME または ANY が指定された場合、述部の結果は次のようになります。
  - 全選択によって戻された値の少なくとも 1 つについて、指定された関係が真である場合、結果は真。
  - 全選択の結果が空の場合、または全選択によって戻されたすべての値について、指定された関係が偽である場合は、結果は偽。
  - 全選択によって戻された値の中に指定された関係が真でないものがあり、しかも少なくとも 1 つの比較が NULL 値により不明であった場合、結果は不明。

行値式 が指定されている場合、全選択から戻される結果列の数は、行値式 によって指定される値式の数と同じでなければなりません。全選択によって、いくつかの値の行が戻されることがあります。行値式の対応する式のデータ・タイプには互換性がなければなりません。行値式 からの各式の値は、全選択からの対応する結果列の値と比較されます。SELECT * は、全選択 の最外部の選択リスト内には許可されません。この述部の値は、以下のように指定された演算子によって異なります。

- ALL を指定すると、述部の結果は次のようになります。
  - 全選択の結果が空の場合、または指定された関係が全選択によって戻されたすべての行について真である場合は、結果は真。

  - 全選択によって戻された行の少なくとも 1 つについて指定された関係が偽である場合、結果は偽。
  - 全選択によって戻された行の中に指定された関係が偽でないものがあり、しかも、少なくとも 1 つの比較が NULL 値により不明になった場合、結果は不明。
- SOME または ANY が指定された場合、述部の結果は次のようになります。
  - 全選択によって戻された行の少なくとも 1 つについて、指定された関係が真である場合、結果は真。
  - 全選択の結果が空の場合、または全選択によって戻されたすべての行について、指定された関係が偽である場合は、結果は偽。
  - 全選択によって戻された行の中に指定された関係が真でないものがあり、しかも少なくとも 1 つの比較が NULL 値により不明であった場合、結果は不明。

述部の対応するオペランドが SBCS データ、混合データ、または Unicode データであり、そのステートメントが実行される時点で有効な照合順序が *HEX でない場合は、そのオペランドの比較は、オペランドの重み付けされた値を使用して行われます。値の重み付けは、該当の照合順序に基づいています。

例

表 **TBLA**

```
COLA
-----
    1
    2
    3
    4
 NULL
```

表 **TBLB**

```
COLB
-----
    2
    3
```

例 1

```
SELECT * FROM TBLA WHERE COLA =  ANY(SELECT COLB FROM TBLB)
```

結果は 2、3 になります。全選択は (2,3) を戻します。行 2 および 3 の COLA が、これらの値の少なくとも 1 つと等しくなります。

例 2

```
SELECT * FROM TBLA WHERE COLA > ANY(SELECT COLB FROM TBLB)
```

結果は 3、4 になります。全選択は (2,3) を戻します。行 3 および 4 の COLA が、これらの値の少なくとも 1 つよりも大きくなります。

例 3

```
SELECT * FROM TBLA WHERE COLA > ALL(SELECT COLB FROM TBLB)
```

結果は 4 になります。全選択は (2,3) を戻します。これらの両方の値よりも大きいのは、行 4 の COLA だけです。

例 *4*

```
SELECT * FROM TBLA WHERE COLA > ALL(SELECT COLB FROM TBLB WHERE COLB<0)
```

結果は 1、2、3、4、およびヌルになります。全選択は値を戻しません。したがって、述部の結果は TBLA のすべての行について真となります。

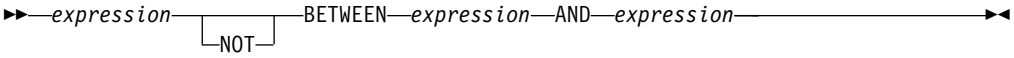
例 *5*

```
SELECT * FROM TBLA WHERE COLA > ANY(SELECT COLB FROM TBLB WHERE COLB<0)
```

結果は空の結果表です。全選択は値を戻しません。したがって、述部の結果は TBLA のすべての行について偽となります。

BETWEEN 述部

BETWEEN 述部は、ある値を値の範囲と比較します。



各オペランドのデータ・タイプが同じでない場合は、すべての値が、 133 ページの『結果のデータ・タイプに関する規則』の適用の結果として得られるデータ・タイプに変換されます。

BETWEEN 述部の各オペランドが CCSID の異なるストリングになっている場合は、以下のような論理的に等価な検索条件を指定した場合と同じように各オペランドが変換されます。

次の BETWEEN 述部は、

```
value1 BETWEEN value2 AND value3
```

論理的に、次の検索条件と等価です。

```
value1 >= value2 AND value1 <= value3
```

次の BETWEEN 述部は、

```
value1 NOT BETWEEN value2 AND value3
```

論理的に、次の検索条件と等価です。

```
NOT(value1 BETWEEN value2 AND value3)
```

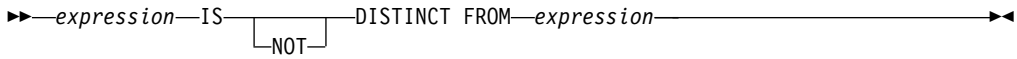
述部のオペランドが SBCS データ、混合データ、または Unicode データであり、そのステートメントが実行される時点で有効な照合順序が *HEX でない場合は、そのオペランドの比較は、オペランドの重み付けされた値を使用して行われます。値の重み付けは、該当の照合順序に基づいています。

例

```
EMPLOYEE.SALARY BETWEEN 20000 AND 40000

SALARY NOT BETWEEN 20000 + :HV1 AND 40000
```

DISTINCT 述部

DISTINCT 述部は、値を別の値と比較します。



述部が IS DISTINCT であるときに、式の比較が真と評価される場合にこの述部の結果は真です。その他の場合、述部の結果は偽です。結果が不明になることはありません。

述部が IS NOT DISTINCT FROM であるときに、式の比較が真と評価される場合 (NULL 値は NULL 値に等しいと見なされます)、この述部の結果は真です。その他の場合、述部の結果は偽です。結果が不明になることはありません。

次の DISTINCT 述部は、

```
value1 IS NOT DISTINCT FROM value2
```

論理的に、次の検索条件と等価です。

```
( value1 IS NOT NULL AND value2 IS NOT NULL AND value1 = value2 )
      OR
( value1 IS NULL AND value2 IS NULL )
```

次の DISTINCT 述部は、

```
value1 IS DISTINCT FROM value2
```

論理的に、次の検索条件と等価です。

```
NOT (value1 IS NOT DISTINCT FROM  value2)
```

DISTINCT 述部のオペランドがそれぞれ異なる CCSID を持つストリングである場合、オペランドは上記の論理的に等価な検索条件が指定されている場合と同様に変換されます。

述部のオペランドが SBCS データ、混合データ、または Unicode データであり、そのステートメントが実行される時点で有効な照合順序が *HEX でない場合は、そのオペランドの比較は、オペランドの重み付けされた値を使用して行われます。値の重み付けは、該当の照合順序に基づいています。

例

表 T1 が存在し、単一の列 C1 を持っていて、 C1 に次のような値を持つ 3 つの行があると想定します: 1、2、NULL 。次の照会は、下記の結果を生成します。

```
SELECT * FROM T1
   WHERE C1 IS DISTINCT FROM :HV
```

C1	:HV	結果
1	2	真
2	2	偽
1	NULL	真
NULL	NULL	偽

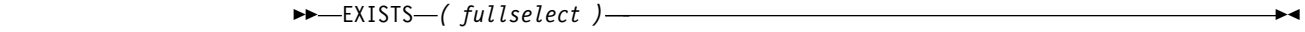
次の照会は、下記の結果を生成します。

```
SELECT * FROM T1
   WHERE C1 IS NOT DISTINCT FROM :HV
```

C1	:HV	結果
1	2	偽
2	2	真
1	NULL	偽
NULL	NULL	真

EXISTS 述部

EXISTS 述部は、特定の行が存在するかどうかを検査します。



全選択には、列をいくつでも指定できます。

- 結果が真になるのは、全選択によって指定された行の数がゼロでなかった場合だけです。
- 結果が偽になるのは、全選択によって指定された行の数がゼロだった場合だけです。
- 結果が不明になることはありません。

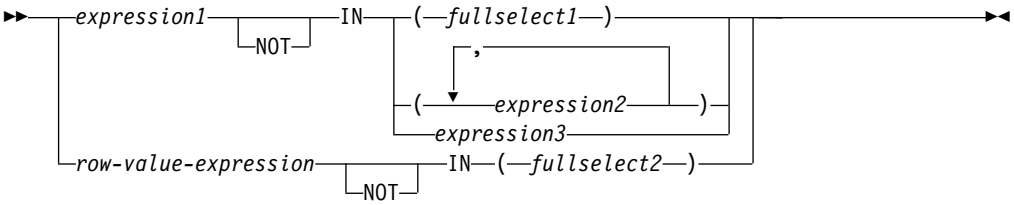
全選択によって戻された値は無視されます。

例

```
EXISTS (SELECT *
        FROM EMPLOYEE WHERE SALARY > 60000)
```

IN 述部

IN 述部は、ある値または複数の値を値の集合と比較します。



expression1 を指定した場合、IN 述部は 1 つの値を値のセットと比較します。*fullselect1* を指定した場合、全選択は単一の結果列を戻す必要があり、NULL か非 NULL かは関係なく、戻す値の数は任意です。*expression1* のデータ・タイプと、*fullselect1*、*expression2*、*expression3* の結果列のデータ・タイプとの間には互換性がなければなりません。各変数は、ホスト構造や変数の宣言規則に従って記述されている構造や変数を識別していなければなりません。

行値式 が指定されている場合、IN 述部は値を値の集合と比較します。

- *fullselect2* の最外部の選択リストでは、SELECT * を使用できません。
- *fullselect2* の結果表の列の数は、*row-value-expression* の数と同じでなければなりません。*row-value-expression* の対応する式のデータ・タイプと、*fullselect2* の対応する結果列のデータ・タイプとの間には、互換性がなければなりません。*row-value-expression* の各式の値が、*fullselect2* の対応する結果列の値と比較されます。

この述部の値は、以下のように指定された演算子によって異なります。

- 演算子が IN である場合、述部の結果は次のようになります。
  - *fullselect2* から返される行のうち、少なくとも 1 つが *row-value-expression* と等しい場合は、真。
  - *fullselect2* の結果が空の場合や、*fullselect2* から返される行のうち、*row-value-expression* と等しい行がまったくない場合は、偽。
  - それ以外の場合は、不明 (つまり、*fullselect2* から戻される行の少なくとも 1 つに NULL 値があるために、*row-value-expression* と *fullselect2* から戻される行の比較が不明と評価され、かつ *fullselect2* から戻された行がどれも *row-value-expression* と等しくない場合)。
- 演算子が NOT IN である場合、述部の結果は次のようになります。
  - *fullselect2* の結果が空の場合、または *row-value-expression* が *fullselect2* から戻されるどの行とも等しくない場合は、真。
  - *row-value-expression* が *fullselect2* から戻される行の少なくとも 1 つと等しい場合は、偽。
  - その他の場合は、不明 (*fullselect2* から返される行のうち、少なくとも 1 つの行が NULL 値になっていることが原因で、*row-value-expression* と *fullselect2* から返される行との比較が、*fullselect2* から返されるどの行についても真にならなければ、*row-value-expression* と *fullselect2* から返される行との比較は不明と評価されます)。

IN 述部は、以下のように他の述部と同等です。

IN 述部	同等述部
expression **IN (**expression**)**	expression **=** expression
expression **IN (**fullselect**)**	expression **= ANY (**fullselect**)**
expression **NOT IN (**fullselect**)**	expression **<> ALL (**fullselect**)**
expression **IN (**expression1, expression2, ..., expressionn**)**	expression **IN (SELECT** * **FROM** R**)** ここで T は単一の行を持つ表で、R は次の全選択によって形成された一時表です。 **SELECT** expression1 **FROM** T **UNION** **SELECT** expression2 **FROM** T **UNION** . . . **UNION** **SELECT** expressionn **FROM** T
row-value-expression **IN (**fullselect**)**	row-value-expression **= SOME (** fullselect**)**
row-value-expression **IN (**fullselect**)**	row-value-expression **= ANY (** fullselect**)**
row-value-expression **NOT IN (**fullselect**)**	row-value-expression **<> ALL (** fullselect**)**

IN 述部のオペランドが異なるデータ・タイプまたは異なる属性を持つ場合は、 IN 述部の演算のデータ・タイプの決定に使用される規則は、 UNION、UNION ALL、EXCEPT、および INTERSECT の場合に使用される規則です。説明は、133 ページの『結果のデータ・タイプに関する規則』を参照してください。

IN 述部のオペランドが異なる CCSID を持つストリングである場合に、変換するオペランドの決定に使用される規則は、ストリングの結合の演算で使用される規則です。説明については、 139 ページの『ストリングを結合する演算に適用される変換規則』を参照してください。

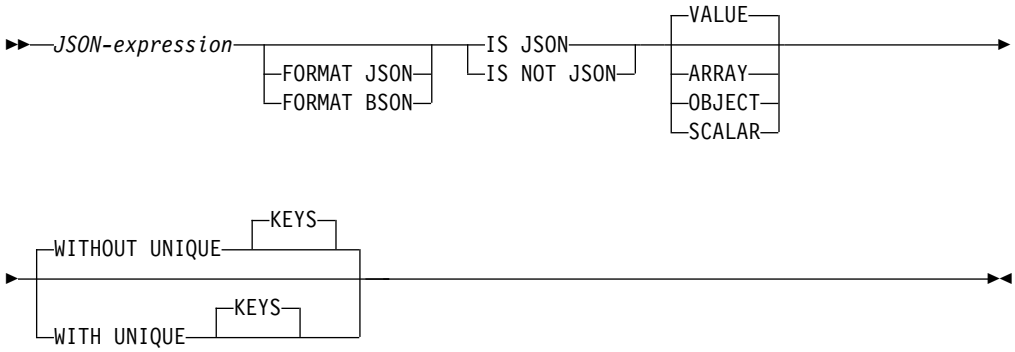
述部の対応するオペランドが SBCS データ、混合データ、または Unicode データであり、そのステートメントが実行される時点で有効な照合順序が *HEX でない場合は、そのオペランドの比較は、オペランドの重み付けされた値を使用して行われます。値の重み付けは、該当の照合順序に基づいています。

例

```
DEPTNO IN ('D01', 'B01', 'C01')

EMPNO IN(SELECT EMPNO FROM EMPLOYEE WHERE WORKDEPT = 'E11')
```

IS JSON 述部

IS JSON 述部は、指定された属性を持つ JSON 形式の値かどうかを判別します。



JSON-expression が、項目タイプおよび UNIQUE KEYS 節によって指定された JSON 形式に合致している場合、IS JSON 述部の結果は真となります。*JSON-expression* が、項目タイプおよび UNIQUE KEYS 節によって指定された JSON 形式に合致していない場合、または *JSON-expression* が空ストリングの場合、結果は偽となります。NOT が指定されている場合、結果は逆になります。*JSON-expression* の値が NULL の場合、結果は不明です。

JSON-expression
組み込みストリング・データ・タイプの値を戻す式。

FORMAT JSON または **FORMAT BSON**
JSON-expression の解釈方法を指定します。

FORMAT JSON
JSON-expression には JSON データが含まれています。*JSON-expression* がバイナリー・データの場合、データは UTF-8 または UTF-16 として解釈されます。EBCDIC CCSID を使用してバイナリー・データをエンコードすることはできません。

FORMAT BSON
JSON-expression には、JSON データの BSON 表現が含まれています。FORMAT BSON を指定した場合、*JSON-expression* はバイナリー・ストリング・データ・タイプでなければなりません。

FORMAT 節を指定しなかった場合、*JSON-expression* が文字ストリングまたはグラフィック・ストリングであれば、*JSON-expression* は JSON として扱われます。 FORMAT 節を指定しなかった場合、*JSON-expression* がバイナリー・ストリングであれば、*JSON-expression* は BSON として扱われます。

VALUE、**ARRAY**、**OBJECT**、または **SCALAR**
JSON-expression の内容と比較する JSON 項目のタイプを指定します。

VALUE
タイプが ARRAY、OBJECT、または SCALAR のいずれかである有効な JSON の値。

ARRAY
有効な JSON 配列。これは、コンマで区切った値のリストを大括弧で囲んだものです。

OBJECT
有効な JSON オブジェクト。これは、コンマで区切ったキー:値 のペアのリストを左中括弧と右中括弧で囲んだものです。

SCALAR
JSON 配列でも JSON オブジェクトでもない有効な JSON 値。スカラー値は、文字ストリング、数値、または JSON リテラル (NULL、true、または false) にすることができます。

WITHOUT UNIQUE KEYS または **WITH UNIQUE KEYS**
キーがユニークでない場合に *JSON-expression* を有効な JSON と見なすかどうかを指定します。 JSON 項目が JSON オブジェクトではない場合、この節は無視されます。

WITHOUT UNIQUE KEYS
キーがユニークでない JSON オブジェクトは、有効な JSON です。

WITH UNIQUE KEYS
キーがユニークでない JSON オブジェクトは、無効な JSON です。

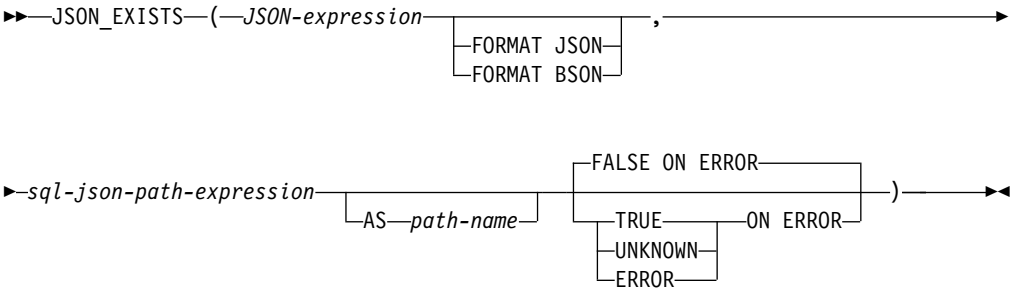
例

- 有効な JSON オブジェクトのみが JSON_DATA 列に挿入されるようにするトリガーを作成します。

```
CREATE TRIGGER VALIDATE_JSON BEFORE INSERT ON T
  REFERENCING NEW AS N
  FOR EACH ROW
  IF N.JSON_DATA IS NOT JSON OBJECT THEN
    SIGNAL SQLSTATE '75007' SET MESSAGE_TEXT = 'Input is not valid JSON');
  END IF
```

JSON_EXISTS 述部

JSON_EXISTS 述部は、指定された *sql-json-path-expression* を使用して検出される JSON 値が、JSON データに含まれているかどうかを判別します。



sql-json-path-expression を使用して、少なくとも 1 つの値が *JSON-expression* で見つかった場合、JSON_EXISTS 述部の結果は真になります。*sql-json-path-expression* で strict モードを使用してエラーが発生した場合、述部の結果は ON ERROR 節によって決まります。*JSON-expression* が NULL 値の場合、JSON_EXISTS 述部の結果は不明です。

JSON-expression
組み込みストリング・データ・タイプを返す式。文字またはグラフィックのデータ・タイプの場合は、正しい形式の JSON データが含まれている必要があります。バイナリー・データ・タイプの場合は、明示的または暗黙的な FORMAT 節に従って解釈されます。

FORMAT JSON または **FORMAT BSON**
JSON-expression の解釈方法を指定します。

FORMAT JSON
JSON-expression には JSON データが含まれています。*JSON-expression* がバイナリー・データの場合、データは UTF-8 または UTF-16 として解釈されます。EBCDIC CCSID を使用してバイナリー・データをエンコードすることはできません。

FORMAT BSON
JSON-expression には、JSON データの BSON 表現が含まれています。FORMAT BSON を指定した場合、*JSON-expression* はバイナリー・ストリング・データ・タイプでなければなりません。

FORMAT 節を指定しなかった場合、*JSON-expression* が文字ストリングまたはグラフィック・ストリングであれば、*JSON-expression* は JSON として扱われます。 *JSON-expression* がバイナリー・ストリングの場合、*JSON-expression* は BSON として扱われます。

sql-json-path-expression
組み込み文字またはグラフィック・ストリング・データ・タイプの値を戻す式。このストリングは SQL/JSON パス式として解釈され、*JSON-expression* で指定された JSON データ内で JSON 値を見つけるために使用されます。SQL/JSON パス式の内容については、 260 ページの『sql-json-path-expression』を参照してください。

sql-json-path-expression の値が空ストリング、すべてブランクのストリング、または NULL 値の場合、値が見つからないため、述部の結果は偽になります。

AS *path-name*
sql-json-path-expression の識別に使用する名前を指定します。

FALSE ON ERROR、**TRUE ON ERROR**、**UNKNOWN ON ERROR**、または **ERROR ON ERROR**
エラーが発生した場合の述部の結果を指定します。

FALSE ON ERROR
エラーが発生した場合、結果は偽になります。これはデフォルトです。

TRUE ON ERROR
エラーが発生した場合、結果は真になります。

UNKNOWN ON ERROR
エラーが発生した場合、結果は UNKNOWN になります。

ERROR ON ERROR
エラーが発生した場合、エラーが戻されます。

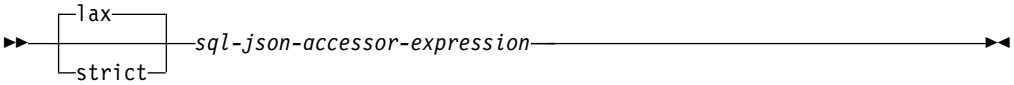
例

- JSON_DATA 列に緊急時連絡先がない社員の行を戻します。 COALESCE を指定することで、NULL 値が空ストリングとして扱われます。 emergency 値を含まないすべての行を戻すために、FALSE ON ERROR 節を使用しています。

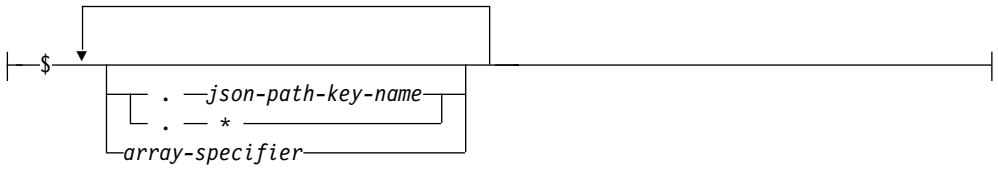
```
SELECT empno, lastname FROM employee
  WHERE NOT JSON_EXISTS(COALESCE(json_data, ''), 'strict $.emergency' FALSE ON ERROR);
```

sql-json-path-expression

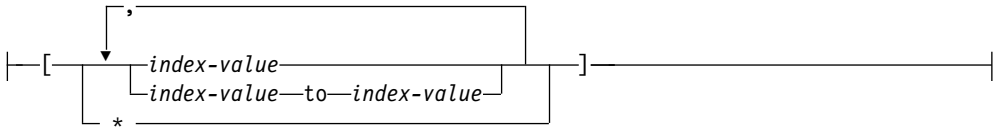
SQL/JSON パス式は、JSON テキストのエレメントへのアクセスを定義します。



sql-json-accessor-expression:



array-specifier:



index-value:

lax または **strict**
JSON パス・モードを指定します。

lax
現在の JSON テキストのナビゲート時に、特定の構造エラーを許容することを指定します。これには、以下が含まれます。

- 配列のネストを自動で解除
- 配列として参照された場合に、自動ラップによってスカラー値を 1 つのエレメント配列に変換。
- 存在しない項目 (範囲外の配列添字値を含む) の指定。

項目が存在しない場合、SQL/JSON パス式は空ストリングを戻します。空ストリングは、現行の ON EMPTY 節に従って処理されます。

strict
指定されたパス式を使用して現在の JSON テキストをナビゲートできない場合に、エラーを報告することを指定します。エラーは、現在の ON ERROR 節に従って処理されます。

sql-json-accessor-expression

$ SQL/JSON パス式の残りの部分が適用される、コンテキスト項目の開始点を指定します。

json-path-key-name
JSON テキスト内の キー、値 のペアのキー名を指定します。名前に特殊文字が含まれる場合は、" 文字で区切る必要があります。

***** すべてのキーの値が SQL/JSON シーケンスとして戻されることを指定します。

array-specifier
配列に適用する 1 つ以上の配列添字値のリストを指定します。値は、個別の数値、または範囲として指定できます。これらは任意の順番で指定でき、重複が含まれてもかまいませんが、結果は重複なく文書順で戻されます。添字値の範囲が指定され、開始値と終了値が順序が正しくない場合、 lax モードでは範囲内のすべての添字値が使用されます。 strict モードではエラーになります。

index-value
配列添字値を指定します。

number
配列エレメントを表す、符号なし整数定数。配列の最初のエレメントの添字は 0 です。

last
配列の最後のエレメントを示します。この値は、添字範囲内の最初の値として指定できません。

last - *number*
配列の最後のエレメントに対する相対位置を示します。

***** すべての配列エレメントが選択されることを示します。

例

- 以下のテキストについて検討します。

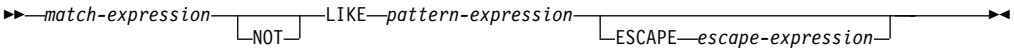
```
{ "isbn": "123-456-222", "author": [ { "name":"Jones"},{"name","Smith"}]}
```

以下は、さまざまな SQL/JSON パス式を使用して JSON テキスト内の項目にアクセスした結果を示しています。

パス	値
$.isbn	"123-456-222"
$.author[0].name	"Jones"
$.author[1].name	"Smith"

LIKE 述部

LIKE 述部は、特定のパターンを持つストリングを検索します。検索するパターンはストリングで指定します。下線およびパーセント記号は、パターンを指定するストリングの中で特別な意味を持ちます。パターン内の末尾ブランクは、パターンの一部です。



いずれかの引数の値が NULL の場合、LIKE 述部の結果は未知になります。

一致式、パターン式、およびエスケープ式 は、ストリングまたは数値を識別しなければなりません。数値引数は、述部の評価の前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。一致式、パターン式、および エスケープ式 の値は、すべてが 2 進ストリングであるか、どれも 2 進ストリングでないか、のどちらかでなければなりません。 この 3 つの引数には、文字ストリングとグラフィック・ストリングを混合して含めることができます。

どの式も特殊タイプを生成することはできませんが、特殊タイプをソース・タイプにキャストする関数を使用することは可能です。

述部のオペランドが SBCS データ、混合データ、または Unicode データであり、そのステートメントが実行される時点で有効な照合順序が *HEX でない場合は、そのオペランドの比較は、オペランドの重み付けされた値を使用して行われます。値の重み付けは、該当の照合順序に基づいています。 ICU 照合順序を LIKE 述部と一緒に使用することはできません。

文字ストリングの場合は、以下の説明における文字、パーセント記号、および下線という用語は、1 バイト文字を指しています。 グラフィック・ストリングの場合は、この用語は 2 バイトまたは Unicode 文字を指しています。 2 進ストリングの場合、この用語は、これらの 1 バイト文字のコード・ポイントを指しています。

match-expression
所定の文字パターンに適合するかどうかを調べるストリングを示す式。

LIKE *pattern-expression*
突き合わせに使用するストリングを示す式。

簡単な説明: LIKE パターンについて、以下に簡単に説明します。

- 下線記号 (_) は、任意の 1 文字を表します。
- パーセント記号 (%) は、ゼロまたは 1 つ以上の文字から構成されるストリングを表します。
- 上記以外の文字は、すべてその文字自身を表します。

pattern-expression に下線文字またはパーセント文字を含める必要がある場合、*escape-expression* を使用して、パターン内の下線文字またはパーセント文字のいずれかの前に置く文字を指定します。

厳密な説明: x は一致式 の値を表し、 y はパターン式 の値を表すものとします。

ストリング *y* は、最小数の一連のサブストリング指定子として解釈されるので、 *y* の各文字は、厳密に 1 つのサブストリング指定子の一部となります。サブストリング指定子とは、下線、パーセント記号、または下線とパーセント記号以外の空でない一連の任意の文字を指します。

x または *y* が NULL 値の場合は、述部の結果は不明です。それ以外の場合、結果は真または偽のいずれかになります。 *x* と *y* の両方が空ストリングの場合、あるいは以下のようなサブストリングに *x* を区分化できる場合、結果は真になります。

- *x* のサブストリングがゼロ個または 1 個以上の連続する一連の文字である場合に、 *x* の各文字が厳密に 1 つのサブストリングの一部である。
- *n* 番目のサブストリング指定子が下線の場合、 *x* の *n* 番目のサブストリング指定子は任意の 1 文字である。
- *n* 番目のサブストリング指定子がパーセント記号の場合、 *x* の *n* 番目のサブストリング指定子は 0 個以上の文字の並びである。
- *n* 番目のサブストリング指定子が下線でもパーセント記号でもない場合に、*x* の *n* 番目のサブストリングが、対応するサブストリング指定子と等しく、かつ対応するサブストリング指定子と同じ長さである。
- *x* のサブストリングの数が、サブストリング指定子の数と同じである。

したがって、*y* が空のストリングで、 *x* が空のストリングでない場合は、結果が偽になります。同様に、*x* が空ストリングで、*y* が空ストリングでない (% 記号以外が含まれている) 場合も、結果は偽になります。

x `NOT LIKE` *y* という述部は、 `NOT(`*x* `LIKE` *y*`)` という検索条件と同等です。

必要な場合、一致式、パターン式、およびエスケープ式 の CCSID は、一致式とパターン式 の間の互換性のある CCSID に変換されます。

混合データ: 列が混合データである場合、パターンには SBCS 文字と DBCS 文字の両方を含めることができます。パターン内の特殊文字は次のように解釈されます。

- SBCS の下線は、1 つの SBCS 文字を示します。
- DBCS の下線は、1 つの DBCS 文字を示します。
- パーセント記号 (SBCS または DBCS) は、任意のタイプ (SBCS または DBCS) の任意の個数の文字を示します。
- 一致式 とパターン式 の余分なシフト文字は無視されます。[48]

Unicode データ: Unicode では、パターン内の特殊文字は次のように解釈されます。

- SBCS または DBCS の下線は 1 文字を表します (1 文字は 1 バイトまたは複数バイト)。
- パーセント記号 (SBCS または DBCS) は、ゼロ個または 1 つ以上の文字のストリングを表します (1 文字は 1 バイトまたは複数バイト)。

48. 余分なシフト文字は通常は無視されます。しかし、それらが確実に無視されるようにするには、IGNORE_LIKE_REDUNDANT_SHIFTS 照会属性を指定してください。照会属性の設定の詳細については、 データベース・パフォーマンスおよび Query 最適化を参照してください。

LIKE 述部を Unicode データで使用する場合、 Unicode の % 記号と下線は、次の表に示されているコード・ポイントを使用します。

文字	**UTF-8**	**UTF-16** または **UCS-2**
半角 %	X'25'	X'0025'
全角 %	X'EFBC85'	X'FF05'
半角 _	X'5F'	X'005F'
全角 _	X'EFBCBF'	X'FF3F'

全角または半角の % はゼロ個以上の文字にマッチングします。全角または半角の _ 文字は厳密に 1 つの文字にマッチングします。 (EBCDIC データの場合、全角の _ 文字は 1 つの DBCS 文字にマッチングします。)

2 進データ**:** 列が 2 進データである場合、パターンにはバイトが含まれます。パターン内の特殊バイトは次のように解釈されます。

- SBCS 下線のコード・ポイント (X'6D') は、1 バイトを参照します。
- SBCS パーセントのコード・ポイント (X'6C') は、任意数のバイトを参照します。

パラメーター・マーカー:

LIKE 述部で指定したパターンがパラメーター・マーカーであり、固定長文字の変数を使用してそのパラメーター・マーカーを置き換えるときは、その変数の値に正しい長さを指定します。正しい長さが指定されない場合、意図した結果が選択で戻されないことになります。

例えば、変数が CHAR(10) として定義されている場合、その変数に WYSE% という値を割り当てると、余白にはブランクが埋め込まれます。使用されるパターンは、次のとおりです。

```
'WYSE%     '
```

このパターンは、WYSE で始まり、5 つのブランク・スペースで終わるすべての値の検索をデータベース・マネージャーに要求します。'WYSE' で始まる値だけを検索したい場合は、変数に 'WYSE%%%%%%' の値を割り当てる必要があります。

ESCAPE *escape-expression*

パターン式の下線 (_) 文字とパーセント (%) 文字の特殊な意味を変更するのに使用する文字を指定する式。これにより、LIKE 述部を使用して、実際にパーセント文字や下線文字を含んでいる値を突き合わせることが可能になります。
ESCAPE 文節と エスケープ式 の使用には、次の規則が適用されます。

- エスケープ式 は、長さが 1 のストリングでなければならない。[49]
- パターン式 には、エスケープ文字の後にエスケープ文字、パーセント記号、または下線が続く場合を除き、エスケープ文字を含めてはならない。

  例えば、「+」が拡張文字の場合、「++」、「+_」、または「+%」以外の「+」がパターン式に現れると、エラーになります。
- エスケープ式 は、パラメーター・マーカーでもかまいません。

49. NULL で終了する場合は、長さが 2 の C 文字ストリング変数を指定できます。

次の例は、連続したエスケープ文字 (この場合は、正符号 (+)) の効果を示しています。

パターン・ストリング	実際のパターン
+%	パーセント記号
++%	正符号の後にゼロ個以上の任意の文字が続く
+++%	正符号の後にパーセント記号が続く

例

例 **1**

PROJECT 表の PROJNAME 列で「SYSTEMS」というストリングを検索します。

```
SELECT PROJNAME
  FROM PROJECT
  WHERE PROJECT.PROJNAME LIKE '%SYSTEMS%'
```

例 **2**

EMPLOYEE 表の FIRSTNME 列で、先頭の文字が「J」で長さがちょうど 2 文字のストリングを検索します。

```
SELECT FIRSTNME
     FROM EMPLOYEE
     WHERE EMPLOYEE.FIRSTNME LIKE 'J_'
```

例 **3**

この例で、

```
SELECT *
     FROM TABLEY
     WHERE C1 LIKE 'AAAA+%BBB%' ESCAPE '+'
```

'+' はエスケープ文字であり、この検索が、'AAAA%BBB' で始まるストリングの検索であることを示しています。'+%' は、パターンの '%' の 1 つのオカレンスとして解釈されます。

例 **4**

次の EBCDIC における例で、COL1 は混合データであると想定しています。この表は、2 列目からの COL1 値を使用して 1 列目の述部を評価した場合の結果を示しています。

述部	COL1 値	結果
WHERE COL1 LIKE 'aaa %AB%C%'	'aaa %ABDZC%'	真
	'aaa %AB% dzx %C%'	真
WHERE COL1 LIKE 'a% %C%'	'a %C%'	真
	'ax %C%'	真
	'ab %DE% fg %C%'	真
WHERE COL1 LIKE 'a _%C%'	'a% %C%'	真
	'a %XC%'	偽
WHERE COL1 LIKE 'a %__C%'	'a %XC%'	真
	'ax %C%'	偽
WHERE COL1 LIKE '%%'	空ストリング	真
WHERE COL1 LIKE 'ab %C% _'	'ab %C% d'	真
	'ab %%%C% d'	真

RV3F001-0

例 **5**

ソース・データ・タイプが CHAR(5) の ZIP_TYPE という名前の特殊タイプが存在し、ある表 TABLEY にデータ・タイプが ZIP_TYPE の ADDRZIP 列が存在するものと想定します。次のステートメントは、ZIP コード (ADDRZIP) が '9555' で始まっている行を選択します。

```
SELECT *
  FROM TABLEY
  WHERE CHAR(ADDRZIP) LIKE '9555%'
```

例 **6**

サンプル表 EMP_RESUME の RESUME 列は、CLOB として定義されています。変数 LASTNAME の値が 'JONES' である場合、次のステートメントは、列内にストリング JONES が見つかった場合、RESUME 列を選択します。

```
SELECT RESUME
  FROM EMP_RESUME
  WHERE RESUME LIKE '%'||LASTNAME||'%'
```

NULL 述部

NULL 述部は、NULL 値かどうかを検査します。

NULL 述部の結果が不明になることはありません。式の値が NULL であれば、結果は真になります。値が NULL でなければ、結果は偽になります。

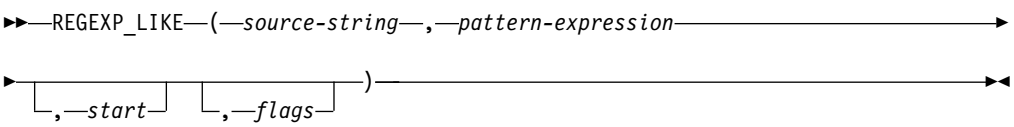
NOT を指定すると、結果が逆になります。

例

```
EMPLOYEE.PHONE IS NULL

SALARY IS NOT NULL
```

REGEXP_LIKE 述部

REGEXP_LIKE 述部は、ストリングで正規表現パターンを検索します。



pattern-expression が見つかった場合、結果は真です。 *pattern-expression* が見つからない場合、結果は偽です。 *source-string* と *pattern-expression* が空ストリングの場合は、結果は真です。 *source-string* または *pattern-expression* のいずれか一方が空ストリングの場合、結果は偽です。いずれかの引数の値が NULL の場合、結果は未知になります。

source-string
その中で検索が行われるストリングを指定する式。式は、組み込み文字ストリング、グラフィック・ストリング、数値、日時のいずれかのデータ・タイプの値を戻す必要があります。値が UTF-16 DBCLOB ではない場合、正規表現パターンを検索する前に、暗黙的に UTF-16 DBCLOB にキャストされます。 FOR BIT DATA 属性の文字ストリングと、バイナリー・ストリングはサポートされません。ストリングの長さが 1 GB を超えてはなりません。

pattern-expression
検索パターンの正規表現ストリングを指定する式。式は、組み込み文字ストリング、グラフィック・ストリング、数値、日時のいずれかのデータ・タイプの値を戻す必要があります。値が UTF-16 DBCLOB ではない場合、正規表現パターンを検索する前に、暗黙的に UTF-16 DBCLOB にキャストされます。 FOR BIT DATA 属性の文字ストリングと、バイナリー・ストリングはサポートされません。ストリングの長さが 32K を超えてはなりません。

有効な *pattern-expression* は、検索のパターンを記述する文字および制御文字のセットで構成されます。有効な制御文字の説明については、 269 ページの『正規表現の制御文字』 を参照してください。

start
検索が開始される *source-string* 内の位置を指定する式。式は任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す必要があります。引数は、正規表現パターンを検索する前に INTEGER にキャストされます。INTEGER への変換について詳しくは、 496 ページの『INTEGER または INT』を参照してください。整数の値は、1 以上でなければなりません。整数の値が *source-string* の実際の長さより大きい場合、述部の結果は偽です。

flags
パターン・マッチングの特性を制御するフラグを指定する式。式は、組み込み文字ストリングまたはグラフィック・ストリングのいずれかのデータ・タイプの値を戻す必要があります。FOR BIT DATA 属性の文字ストリングと、バイナリー・ストリングはサポートされません。このストリングには、1 つ以上の有効なフラグ値を含めることができます。フラグ値の組み合わせは有効でなければなりません。空ストリングは、値「c」と同じです。

有効なフラグ文字の説明については、『正規表現のフラグ値』 を参照してください。

正規表現のフラグ値

次の表では、サポートされるフラグ値について説明します。

表 31. フラグ値

フラグ値	説明
c	マッチングで大文字小文字が区別されることを指定します。これは、「c」および「i」のいずれも指定されていない場合のデフォルト値です。この値を値「i」と一緒に指定してはなりません。
i	マッチングで大文字小文字が区別されないことを指定します。この値を値「c」と一緒に指定してはなりません。
m	入力データが複数の行を含む可能性があることを指定します。デフォルトでは、パターン内の「^」および「$」はそれぞれ、入力ストリングの開始と終了にのみ一致します。このフラグが設定されている場合、「^」と「$」は入力ストリングの各行の開始と終了にも一致します。
n	パターン内の文字「.」は入力ストリングの行終了文字に一致することを指定します。デフォルトでは、パターン内の「.」は行終了文字に一致しません。入力ストリング内の復帰と改行 (LF) のペアは、単一の行終了文字として振る舞い、パターン内の単一の「.」と一致します。
s	パターン内の文字「.」は入力ストリングの行終了文字に一致することを指定します。これは、値「n」の同義語です。
x	エスケープされない限り、パターン内の空白文字は無視されることを指定します。

正規表現の制御文字

正規表現の処理は、 International Components for Unicode (ICU) 正規表現 API を使用して、以下にリストした制御文字を使用した正規表現パターンを含む Unicode データに対して実行されます。

半角文字のみが認識されることに注意してください。これらの表にある文字に対応する全角文字はどれも認識されません。

表 32. 正規表現のメタキャラクター

文字	セットの外部	[セットの内部]	説明
¥a	✓	✓	ベル文字、¥u0007 にマッチします。
¥A	✓		入力の先頭にマッチします。¥A は、入力内の改行の後にはマッチしない点で、^ と異なります。
\b	✓		現在位置がワード境界の場合にマッチします。境界は、単語文字 (¥w) と非単語文字 (¥W) の間の移行部で発生し、結合文字は無視されます。
¥B	✓		現在位置がワード境界ではない場合にマッチします。
¥cX	✓	✓	CTRL-X 文字にマッチします。
¥d	✓	✓	Unicode 一般カテゴリーが Nd (数字、10 進数字) である任意の文字にマッチします。
¥D	✓	✓	10 進数字ではない任意の文字にマッチします。
¥e	✓	✓	エスケープ文字、¥u001B にマッチします。
¥E	✓	✓	¥Q ... ¥E の引用符で囲まれたシーケンスを終了します。

表 32. 正規表現のメタキャラクター (続き)

文字	セットの外部	[セットの内部]	説明
¥f	✓	✓	用紙送り、¥u000C にマッチします。
¥G	✓		現在位置が前回のマッチの終わりである場合にマッチします。
¥n	✓	✓	改行、¥u000A にマッチします。
¥N{UNICODE CHARACTER NAME}	✓	✓	名前付き文字にマッチします。
¥p{UNICODE PROPERTY NAME}	✓	✓	指定された Unicode プロパティーを持つ任意の文字にマッチします。
¥P{UNICODE PROPERTY NAME}	✓	✓	指定された Unicode プロパティーを持たない任意の文字にマッチします。
¥Q	✓	✓	¥E までの後続のすべての文字を引用符で囲みます。
¥r	✓	✓	復帰、¥u000D にマッチします。
¥s	✓	✓	空白文字にマッチします。空白文字は、[¥t¥n¥f¥r¥p{Z}] として定義されます。
¥S	✓	✓	非空白文字にマッチします。
¥t	✓	✓	水平タブ、¥u0009 にマッチします。
¥uhhhh	✓	✓	16 進値が hhhh である文字にマッチします。
¥Uhhhhhhhh	✓	✓	16 進値が hhhhhhhh である文字にマッチします。最大 Unicode コード・ポイントが ¥U0010ffff である場合でも、正確に 8 桁の 16 進数字を指定する必要があります。
¥w	✓	✓	単語文字にマッチします。単語文字は、[¥p{Alphabetic}¥p{Mark}¥p{Decimal_Number} ¥p{Connector_Punctuation}¥u200c¥u200d] です。
¥W	✓	✓	非単語文字にマッチします。
¥x{hhhh}	✓	✓	16 進値が hhhh である文字にマッチします。 1 桁から 6 桁の 16 進数字を指定できます。
¥xhh	✓	✓	2 桁の 16 進値が hh である文字にマッチします。
¥X	✓		書記素クラスターにマッチします。
\Z	✓		現在位置が入力の終わりであるが、最後の行終了文字 (存在する場合) の前である場合にマッチします。
¥z	✓		現在位置が入力の終わりである場合にマッチします。
¥n	✓		後方参照。n 番目のキャプチャー・グループがマッチするたびにマッチします。 n は 1 より大きく、かつパターン内のキャプチャー・グループの総数より小さい数でなければなりません。
¥0ooo	✓	✓	8 進文字にマッチします。 'ooo' は 1 桁から 3 桁の 8 進数字です。 0377 が、許容される最大の 8 進文字です。先行ゼロが必要です。これにより、 8 進定数と後方参照を区別します。
[pattern]	✓	✓	セットからの任意の 1 文字にマッチします。
.	✓		任意の文字にマッチします。
^	✓		行の先頭にマッチします。

表 32. 正規表現のメタキャラクター (続き)

文字	セットの外部	[セットの内部]	説明
$	✓		行の終わりにマッチします。
\	✓		後続文字を引用符で囲みます。リテラルとして処理するために引用符で囲む必要がある文字は、以下のものです。 * ? + [( ) { } ^ $ | ¥ .
\		✓	後続文字を引用符で囲みます。リテラルとして処理するために引用符で囲む必要がある文字は、[ ] ¥ です。状況によっては引用符で囲まなければならない場合がある文字は、- & です。

表 33. 正規表現の演算子

演算子	説明
|	代替。A|B は、A または B のどちらかにマッチします。
*	0 回以上、マッチします。できる限り多くの回数のマッチが行われます。
+	1 回以上、マッチします。できる限り多くの回数のマッチが行われます。
?	0 回または 1 回、マッチします。1 回が優先されます。
{n}	正確に n 回、マッチします。
{n,}	少なくとも n 回、マッチします。できる限り多くの回数のマッチが行われます。
{n,m}	n 回から m 回までマッチします。できる限り多くの回数のマッチが行われますが、m 回を超えることはありません。
*?	0 回以上、マッチします。できる限り少ない回数のマッチが行われます。
+?	1 回以上、マッチします。できる限り少ない回数のマッチが行われます。
??	0 回または 1 回、マッチします。0 回が優先されます。
{n}?	正確に n 回、マッチします。
{n,}?	少なくとも n 回のマッチが行われますが、パターン全体のマッチに必要な回数は超えません。
{n,m}?	n 回から m 回までマッチします。できる限り少ない回数のマッチが行われますが、n 回より少ないことはありません。
*+	0 回以上、マッチします。1 つ目の検出時にできる限り多くの回数のマッチを行い、全体としてのマッチに失敗してもマッチ回数が少なくなる再試行は行われません (強欲なマッチ)。
++	1 回以上、マッチします。強欲なマッチです。
?+	0 回または 1 回、マッチします。強欲なマッチです。
{n}+	正確に n 回、マッチします。
{n,}+	少なくとも n 回、マッチします。強欲なマッチです。
{n,m}+	n 回から m 回までマッチします。強欲なマッチです。
(...)	キャプチャー括弧。括弧で囲まれた副次式にマッチした入力の範囲が、マッチ後に使用可能になります。
(?: ...)	非キャプチャー括弧。含まれるパターンをグループ化しますが、マッチするテキストのキャプチャーは行われません。キャプチャー括弧よりも、いくらか効率が良くなります。
(?> ...)	アトミック・マッチ括弧。括弧で囲まれた副次式の 1 つ目のマッチが、試行される唯一のマッチです。それによりパターン全体のマッチが発生しない場合は、「(?>」の前の位置にマッチの検索をバックアップします。
(?# ...)	フリー形式のコメント (?# コメント)。

表 33. 正規表現の演算子 (続き)

演算子	説明
(?= ...)	先読みアサーション。括弧で囲まれたパターンが現在の入力位置にマッチする場合に真ですが、入力位置を前に移動しません。
(?! ...)	負の先読みアサーション。括弧で囲まれたパターンが現在の入力位置にマッチしない場合に真です。入力位置を前に移動しません。
(?<= ...)	後読みアサーション。括弧で囲まれたパターンが現在の入力位置より前のテキストにマッチし、マッチの最後の文字が現在位置の直前の入力文字である場合に真です。入力位置を変更しません。後読みパターンがマッチする有効なストリングの長さは、無制限であってはなりません (* 演算子も + 演算子もないこと)。
(?<! ...)	負の後読みアサーション。括弧で囲まれたパターンが現在の入力位置より前のテキストにマッチせず、マッチの最後の文字が現在位置の直前の入力文字である場合に真です。入力位置を変更しません。後読みパターンがマッチする有効なストリングの長さは、無制限であってはなりません (* 演算子も + 演算子もないこと)。
(?ismwx-ismwx: ...)	フラグの設定。指定されたフラグの有効化または無効化によって、括弧で囲んだ式を評価します。
(?ismwx-ismwx)	フラグの設定。フラグの設定を変更します。変更は、設定の後ろのパターンの一部分に適用されます。例えば、(?i) は、大/小文字を区別しないマッチに変更します。

表 34. セット式 (文字クラス)

例	説明
[abc]	文字 a、b、または c のどれにもマッチします。
[^abc]	否定 - a、b、および c を除く任意の文字にマッチします。
[A-M]	範囲 - A から M までの任意の文字にマッチします。含まれる文字は、Unicode コード・ポイントの順序によって決まります。
[¥u0000-¥U0010ffff]	範囲 - すべての文字にマッチします。
[¥p{Letter}] [¥p{General_Category=Letter}] [\p{L}]	Unicode カテゴリー = Letter の文字。示されたすべての形式は同等です。
[¥P{Letter}]	否定プロパティー。(大文字 ¥P) Letter 以外のすべてにマッチします。
[¥p{numeric_value=9}]	数値 9 を含むすべての数にマッチします。セット式で任意の Unicode プロパティーを使用できます。
[¥p{Letter}&&¥p{script=cyrillic}]	論理 AND または論理積。すべてのキリル文字のセットにマッチします。
[¥p{Letter}--¥p{script=latin}]	減算。ラテン以外のすべての文字にマッチします。
[[a-z][A-Z][0-9]] [[a-zA-Z0-9]]	暗黙的論理 OR または和集合。これらの例は、ASCII 文字と数字にマッチします。2 つの形式は同等です。
[:script=Greek:]	プロパティーの POSIX に似た代替構文。 ¥p{script=Greek} と同等です。

注

前提条件**:** REGEXP_LIKE 述部を使用するには、 International Components for Unicode (ICU) オプションがインストールされていなければなりません。

処理**:** 正規表現の処理は、 International Components for Unicode (ICU) 正規表現インターフェースを使用して行われます。詳しくは、http://userguide.icu-project.org/strings/regexp を参照してください。

3 つの引数のみが指定された場合、 3 番目の引数は、*start* 引数または *flags* 引数の可能性があります。 3 番目の引数がストリングの場合は、*flags* 引数として解釈されます。それ以外の場合は、*start* 引数として解釈されます。

例

- 例 *1*: EMPLOYEE 表から姓のスペルが LUCCHESSI、LUCHESSI、または LUCHESI である従業員番号を選択します。大文字小文字は考慮しません。

```
SELECT EMPNO FROM EMPLOYEE
 WHERE REGEXP_LIKE(LASTNAME,'luc+?hes+?i','i')
```

  結果は、EMPNO 値 '000110' の 1 行です。

- 例 *2*: PRODUCT 表からすべての無効なプロダクト ID 値を選択します。予期される形式は 'nnn-nnn-nn' で、'n' は 0 から 9 の数字です。

```
SELECT PID FROM PRODUCT
 WHERE NOT REGEXP_LIKE(pid,'[0-9]{3}-[0-9]{3}-[0-9]{2}')
```

  すべてのプロダクト ID がパターンにマッチするため、結果は 0 行です。

トリガー・イベント述部

トリガー・イベント述部は、トリガーをアクティブにしたイベントをテストするためのトリガー・アクションで使用されます。これは CREATE TRIGGER ステートメントのトリガー・アクション内でのみ使用できます。

DELETING
トリガーが削除操作でアクティブになった場合に TRUE になります。それ以外の場合は FALSE になります。

INSERTING
トリガーが挿入操作でアクティブになった場合に TRUE になります。それ以外の場合は FALSE になります。

UPDATING
トリガーが更新操作でアクティブになった場合に TRUE になります。それ以外の場合は FALSE になります。

注

トリガー・イベント述部は、CREATE TRIGGER ステートメントのトリガー・アクション内のどこでも使用できます。

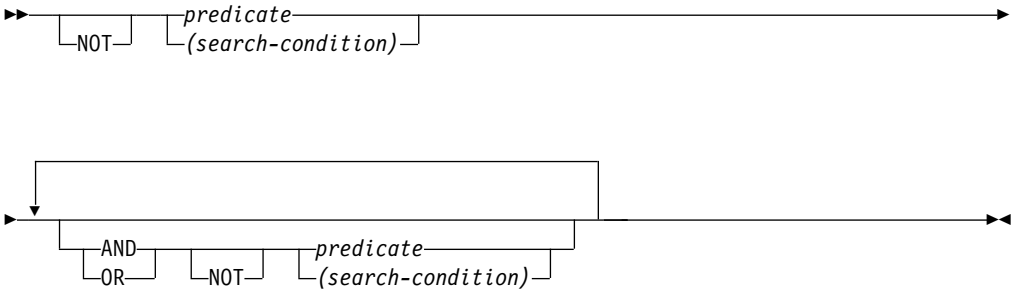
例

以下のトリガーは、ルーチン本体のトリガー・イベント述部で使用され、新しい従業員が採用されるたびに (つまり、EMPLOYEE 表に新しい行が挿入されるたびに) 従業員数を増やし、従業員が退職するたびに従業員数を減らします。また、昇給額が現在の給与の 10 パーセントを超えることになるような更新が起こった場合はエラーを出します。

```
CREATE TRIGGER HIRED
 AFTER INSERT OR DELETE OR UPDATE OF SALARY ON EMPLOYEE
 REFERENCING NEW AS N OLD AS O FOR EACH ROW
   BEGIN
     IF INSERTING
       THEN UPDATE COMPANY_STATS SET NBREMP = NBREMP + 1;
     END IF;
     IF DELETING
       THEN UPDATE COMPANY_STATS SET NBREMP = NBREMP - 1;
     END IF;
     IF UPDATING AND (N.SALARY > 1.1 * O.SALARY)
        THEN SIGNAL SQLSTATE '75000' SET MESSAGE_TEXT = 'Salary increase > 10%'
     END IF;
   END
```

検索条件

検索条件 は、ある所定の行またはグループについて、真、偽、または不明の条件を示します。



検索条件の結果は、指定した論理演算子 (AND、OR、NOT) を、指定したそれぞれの述部の結果に適用することによって求められます。論理演算子を指定しないと、指定した述部の結果が検索条件の結果となります。

AND および OR の定義を以下の表に示します。表の中の P および Q は、任意の述部を示します。

表 *35. AND* と *OR* の真理値表

P	**Q**	**P AND Q**	**P OR Q**
真	真	真	真
真	偽	偽	真
真	不明	不明	真
偽	真	偽	真
偽	偽	偽	偽
偽	不明	偽	不明
不明	真	不明	真
不明	偽	偽	不明
不明	不明	不明	不明

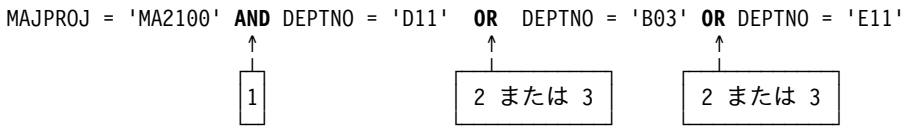
NOT(真) は偽、NOT(偽) は真、NOT(不明) は不明になります。

括弧内の検索条件が先に評価されます。評価の順序を括弧で指定しなかった場合は、NOT が処理されてから AND が処理され、AND が処理されてから OR が処理されます。同じ優先順位レベルにある演算子が評価されるときの順序は、検索条件が最適化されるため、決まっていません。

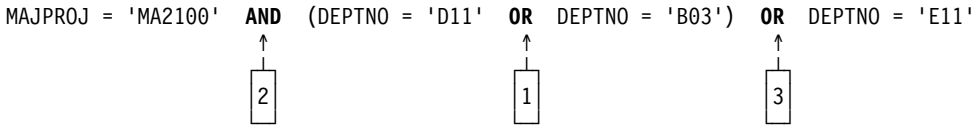
例

以下の例では、2 行目の番号が演算子の評価される順序を示しています。

例 **1**


MAJPROJ = 'MA2100' AND DEPTNO = 'D11' OR DEPTNO = 'B03' OR DEPTNO = 'E11'
1
2 または 3
2 または 3

例 **2**


MAJPROJ = 'MA2100' AND (DEPTNO = 'D11' OR DEPTNO = 'B03') OR DEPTNO = 'E11'
2
1
3

第 3 章 組み込みグローバル変数

この章では、組み込みグローバル変数について、意味の説明、規則、および使用例を示します。

組み込みグローバル変数はデータベース・マネージャーに用意されているもので、その変数に関連付けられたスカラー値を取得するために SQL ステートメントで使用します。

例えば、ROUTINE_TYPE グローバル変数を SQL ステートメント内で参照すれば、現行ルーチンのタイプを取得できます。

グローバル変数の値を取得するステートメントの権限 ID には、グローバル変数に対する READ 特権と、グローバル変数を含むスキーマに対する USAGE 特権が必要です。

例

グローバル変数 CLIENT_HOST にアクセスするには、以下の照会を実行します。

```
SELECT SYSIBM.CLIENT_HOST
FROM SYSIBM.SYSDUMMY1
```

照会は、現行クライアントのホスト名を返します。

```
hotellnx93
```

CLIENT_HOST

このグローバル変数には、システムによって戻される現行クライアントのホスト名が入ります。

このグローバル変数には、以下の特性があります。

- 読み取り専用で、値はシステムによって保守されます。
- タイプは VARCHAR(255) です。
- スキーマは SYSIBM です。
- このグローバル変数の有効範囲はセッションです。

クライアント接続がローカル・システムで実行されているアプリケーションからの接続の場合、このグローバル変数の値は NULL です。サーバーは、接続が受け入れられるときに、ネットワークからクライアント IP アドレスを取得します。TCP/IP を使用するリモート・システムからのプロセスではない場合、このグローバル変数の値は NULL です。

CLIENT_IPADDR

このグローバル変数には、システムによって戻される現行クライアントの IP アドレスが入ります。

このグローバル変数には、以下の特性があります。

- 読み取り専用で、値はシステムによって保守されます。
- タイプは VARCHAR(128) です。
- スキーマは SYSIBM です。
- このグローバル変数の有効範囲はセッションです。

クライアントが TCP/IP または SSL プロトコルを使用して接続しなかった場合、このグローバル変数の値は NULL です。

CLIENT_PORT

このグローバル変数は、現行クライアントがサーバーと通信するために使用するポート番号を含みます。

このグローバル変数には、以下の特性があります。

- 読み取り専用で、値はシステムによって保守されます。
- タイプは INTEGER です。
- スキーマは SYSIBM です。
- このグローバル変数の有効範囲はセッションです。

クライアントが TCP/IP プロトコルを使用して接続しなかった場合、このグローバル変数の値は NULL です。

JOB_NAME

このグローバル変数には、現行ジョブの名前が入ります。

このグローバル変数には、以下の特性があります。

- 読み取り専用で、値はシステムによって保守されます。
- タイプは VARCHAR(28) です。
- スキーマは QSYS2 です。
- このグローバル変数の有効範囲はセッションです。

PACKAGE_NAME

このグローバル変数には、DRDA 接続に現在使用されているパッケージの名前が入ります。

このグローバル変数には、以下の特性があります。

- 読み取り専用で、値はシステムによって保守されます。
- タイプは VARCHAR(128) です。
- スキーマは SYSIBM です。
- このグローバル変数の有効範囲はセッションです。

現在実行中のパッケージがない場合、値は NULL です。

PACKAGE_SCHEMA

このグローバル変数には、DRDA 接続に現在使用されているパッケージのスキーマ名が入ります。

このグローバル変数には、以下の特性があります。

- 読み取り専用で、値はシステムによって保守されます。
- タイプは VARCHAR(128) です。
- スキーマは SYSIBM です。
- このグローバル変数の有効範囲はセッションです。

現在実行中のパッケージがない場合、値は NULL です。

PACKAGE_VERSION

このグローバル変数には、DRDA 接続に現在使用されているパッケージのバージョン ID が入ります。

このグローバル変数には、以下の特性があります。

- 読み取り専用で、値はシステムによって保守されます。
- タイプは VARCHAR(64) です。
- スキーマは SYSIBM です。
- このグローバル変数の有効範囲はセッションです。

現在実行中のパッケージがないか、現在実行中のパッケージにバージョン ID がない場合、値は NULL です。パッケージは、Db2 for i 以外のサーバーから作成されている場合にのみ、バージョン ID があります。

PROCESS_ID

このグローバル変数には、現行ジョブのプロセス ID が入ります。

このグローバル変数には、以下の特性があります。

- 読み取り専用で、値はシステムによって保守されます。
- タイプは INTEGER です。
- スキーマは QSYS2 です。
- このグローバル変数の有効範囲はセッションです。

例

このジョブの詳細を確認します。

```
SELECT USER, CURRENT SERVER, QSYS2.JOB_NAME, QSYS2.PROCESS_ID, QSYS2.THREAD_ID
  FROM SYSIBM.SYSDUMMY1
```

ROUTINE_SCHEMA

このグローバル変数には、現在実行中のルーチンのスキーマ名が入ります。

このグローバル変数には、以下の特性があります。

- 読み取り専用で、値はシステムによって保守されます。
- タイプは VARCHAR(128) です。
- スキーマは SYSIBM です。
- このグローバル変数の有効範囲はセッションです。

現在実行中のルーチンがない場合、値は NULL です。

ROUTINE_SCHEMA グローバル変数の値は、プロシージャーおよび関数についてのみ設定されます。値は、現在実行中のルーチンのスキーマ名を常に反映します。

インライン記述されている関数については値は変更されません。インライン関数が呼び出されたときと同じ値のままになります。

ROUTINE_SPECIFIC_NAME

このグローバル変数には、現在実行中のルーチンの名前が入ります。

このグローバル変数には、以下の特性があります。

- 読み取り専用で、値はシステムによって保守されます。
- タイプは VARCHAR(128) です。
- スキーマは SYSIBM です。
- このグローバル変数の有効範囲はセッションです。

現在実行中のルーチンがない場合、値は NULL です。

ROUTINE_SPECIFIC_NAME グローバル変数の値は、プロシージャーおよび関数についてのみ設定されます。値は、現在実行中のルーチンの名前を常に反映します。

インライン記述されている関数については値は変更されません。インライン関数が呼び出されたときと同じ値のままになります。

ROUTINE_TYPE

このグローバル変数には、現在実行中のルーチンのタイプが入ります。

このグローバル変数には、以下の特性があります。

- 読み取り専用で、値はシステムによって保守されます。
- タイプは CHAR(1) です。
- スキーマは SYSIBM です。
- このグローバル変数の有効範囲はセッションです。

このグローバル変数の値は、プロシージャーの場合は「P」、関数の場合は「F」です。現在実行中のルーチンがない場合、値は NULL です。

ROUTINE_TYPE グローバル変数の値は、プロシージャーおよび関数についてのみ設定されます。値は、現在実行中のルーチンのタイプを常に反映します。

インライン記述されている関数については値は変更されません。インライン関数が呼び出されたときと同じ値のままになります。

SERVER_MODE_JOB_NAME

このグローバル変数には、SQL サーバー・モード接続を確立したジョブの名前が入ります。

このグローバル変数には、以下の特性があります。

- 読み取り専用で、値はシステムによって保守されます。
- タイプは VARCHAR(28) です。
- スキーマは QSYS2 です。
- このグローバル変数の有効範囲はセッションです。

サーバー・モード接続がない場合、値は NULL です。

THREAD_ID

このグローバル変数には、現行スレッドのスレッド ID が入ります。

このグローバル変数には、以下の特性があります。

- 読み取り専用で、値はシステムによって保守されます。
- タイプは BIGINT です。
- スキーマは QSYS2 です。
- このグローバル変数の有効範囲はセッションです。

例

現行スレッドによって保持され、SALES 表に対する、有効範囲がスレッドのレコード・ロックを収集します。

```
SELECT * FROM QSYS2.RECORD_LOCK_INFO L
  WHERE L.TABLE_NAME = 'SALES' AND
        L.JOB_NAME = QSYS2.JOB_NAME AND
        L.THREAD_ID = QSYS2.THREAD_ID
```

第 4 章 組み込み関数

この章には、以下の表にリストしている 組み込み関数 の構文図、意味の説明、規則、および使用例を示しています。

機能について詳しくは、 182 ページの『関数』を参照してください。

表 *36.* 集約関数

関数	説明	参照
ARRAY_AGG	一連のエレメントを配列として集約します。	304 ページの『ARRAY_AGG』
AVG	数値の集合の平均を戻します。	306 ページの『AVG』
CORR または CORRELATION	数値ペア集合の相関係数を戻します。	308 ページの『CORR または CORRELATION』
COUNT	行または値の集合の中にある行の数または値の数を戻します。	310 ページの『COUNT』
COUNT_BIG	行または値の集合の中にある行の数または値の数を戻します (COUNT_BIG は COUNT 関数に類似していますが、COUNT 関数とは異なり、結果の値は整数の最大値より大きい値をとることができます)。	311 ページの『COUNT_BIG』
COVARIANCE または COVAR	一組の数値のペアの (母集団の) 共分散係数を戻します。	312 ページの『COVARIANCE または COVAR』
COVAR_SAMP または COVARIANCE_SAMP	数値ペア集合の不偏標本共分散 (n-1) を戻します。	314 ページの『COVAR_SAMP または COVARIANCE_SAMP』
GROUPING	行がグループ化集合の要約行として生成されたかどうかを示す値を戻します。	316 ページの『GROUPING』
JSON_ARRAYAGG	JSON 値または SQL 値の集合の各値に対応した配列エレメントを含む JSON 配列を戻します。	318 ページの『JSON_ARRAYAGG』
JSON_OBJECTAGG	SQL 値の集合の個々の各キーおよび値に対応した *key:value* のペアを含む JSON オブジェクトを戻します。	322 ページの『JSON_OBJECTAGG』
LISTAGG	ストリングを連結することにより、ストリング・エレメントの集合を 1 つのストリングに集約します。	327 ページの『LISTAGG』
MAX	あるグループの中の値の集合の最大値を戻します。	331 ページの『MAX』
MEDIAN	数値集合の中央値を戻します。	332 ページの『MEDIAN』
MIN	あるグループの中の値の集合の最小値を戻します。	333 ページの『MIN』
PERCENTILE_CONT	連続分散モデルを使用して、ソート指定が与えられた指定されたパーセンタイルに該当する値を戻します。	335 ページの『PERCENTILE_CONT』
PERCENTILE_DISC	離散分散モデルを使用して、ソート指定が与えられた指定されたパーセンタイルに該当する値を戻します。	337 ページの『PERCENTILE_DISC』

表 36. 集約関数 (続き)

関数	説明	参照
回帰関数	回帰関数は、通常の最小二乗法による回帰直線 (形式 y = a * x + b) を数値ペアの集合に当てはめることをサポートします。	339 ページの『回帰関数』
STDDEV	数値の集合のバイアス標準偏差を戻します。	342 ページの『STDDEV_POP または STDDEV』
STDDEV_SAMP	数値の集合の標本標準偏差を戻します。	343 ページの『STDDEV_SAMP』
SUM	数値の集合の合計を戻します。	344 ページの『SUM』
VARIANCE または VAR	数値の集合のバイアス偏差を戻します。	345 ページの『VAR_POP または VARIANCE または VAR』
VAR_SAMP または VARIANCE_SAMP	数値の集合の標本分散を戻します。	346 ページの『VAR_SAMP または VARIANCE_SAMP』
XMLAGG	1 組の XML 値の中の NULL 以外の値ごとに 1 つの項目を含む XML シーケンスを戻します。	347 ページの『XMLAGG』
XMLGROUP	整形式 XML 文書である XML 値を戻します。	349 ページの『XMLGROUP』

表 37. キャスト・スカラー関数

関数	説明	参照
BIGINT	数値の 64 ビット整数表現を戻します。	363 ページの『BIGINT』
BINARY	任意のタイプのストリングの BINARY 表現を戻します。	365 ページの『BINARY』
BLOB	任意のタイプのストリングの BLOB 表現を戻します。	369 ページの『BLOB』
CHAR	値の CHARACTER 表現を戻します。	375 ページの『CHAR』
CLOB	値の CLOB 表現を戻します。	384 ページの『CLOB』
DATE	値から DATE を戻します。	405 ページの『DATE』
DBCLOB	ストリングの DBCLOB 表現を戻します。	414 ページの『DBCLOB』
DECFLOAT	数値の DECFLOAT 表現を戻します。	423 ページの『DECFLOAT』
DECIMAL	数値の DECIMAL 表現を戻します。	429 ページの『DECIMAL または DEC』
DOUBLE_PRECISION または DOUBLE	数値の DOUBLE_PRECISION 表現を戻します。	448 ページの『DOUBLE_PRECISION または DOUBLE』
FLOAT	数値の FLOAT 表現を戻します。	465 ページの『FLOAT』
GRAPHIC	ストリングの GRAPHIC 表現を戻します。	474 ページの『GRAPHIC』
INTEGER または INT	数値の INTEGER 表現を戻します。	496 ページの『INTEGER または INT』
REAL	数値の REAL 表現を戻します。	587 ページの『REAL』
ROWID	値から行 ID を戻します。	612 ページの『ROWID』
SMALLINT	数値の SMALLINT 表現を戻します。	628 ページの『SMALLINT』

表 37. キャスト・スカラー関数 (続き)

関数	説明	参照
TIME	値から TIME を戻します。	643 ページの『TIME』
TIMESTAMP	1 つまたは 1 対の値から TIMESTAMP を戻します。	644 ページの『TIMESTAMP』
TIMESTAMP_ISO	日時値からタイム・スタンプ値を戻します。	652 ページの『TIMESTAMP_ISO』
VARBINARY	任意のタイプのストリングの VARBINARY 表現を戻します。	669 ページの『VARBINARY』
VARCHAR	値の VARCHAR 表現を戻します。	672 ページの『VARCHAR』
VARGRAPHIC	値の VARGRAPHIC 表現を戻します。	689 ページの『VARGRAPHIC』
ZONED	数値のゾーン 10 進数表現を戻します。	738 ページの『ZONED』

表 38. データ・リンク・スカラー関数

関数	説明	参照
DLCOMMENT	データ・リンク値からコメント値を戻します。	439 ページの『DLCOMMENT』
DLLINKTYPE	データ・リンク値からリンク・タイプ値を戻します。	440 ページの『DLLINKTYPE』
DLURLCOMPLETE	リンク・タイプ URL のデータ・リンク値から完全な URL 値を戻します。	441 ページの『DLURLCOMPLETE』
DLURLPATH	リンク・タイプ URL のデータ・リンク値から、あるサーバー内のファイルにアクセスするのに必要なパスとファイル名を戻します。	442 ページの『DLURLPATH』
DLURLPATHONLY	リンク・タイプ URL のデータ・リンク値から、あるサーバー内のファイルにファイル・アクセス・トークンなしでアクセスするのに必要なパスとファイル名を戻します。	443 ページの『DLURLPATHONLY』
DLURLSCHEME	リンク・タイプ URL のデータ・リンク値からそのスキーマを戻します。	444 ページの『DLURLSCHEME』
DLURLSERVER	リンク・タイプ URL のデータ・リンク値から、ファイル・サーバーを戻します。	445 ページの『DLURLSERVER』
DLVALUE	データ・リンク値を戻します。	446 ページの『DLVALUE』

表 39. 日付時刻スカラー関数

関数	説明	参照
ADD_MONTHS	日付引数に月数の引数を足したものを表す日付を戻します。	355 ページの『ADD_MONTHS』
CURDATE	時点の刻時機構の読み取りに基づく日付を戻します。	400 ページの『CURDATE』
CURTIME	時点の刻時機構の読み取りに基づく時刻を戻します。	401 ページの『CURTIME』
DAY	値の日付の部分を戻します。	407 ページの『DAY』
DAYNAME	値の曜日の名前の部分を戻します。	408 ページの『DAYNAME』

表 39. 日付時刻スカラー関数 (続き)

関数	説明	参照
DAYOFMONTH	その月の日付を表す整数を戻します。	409 ページの『DAYOFMONTH』
DAYOFWEEK	値から曜日 (1 は日曜日を表し、7 は土曜日を表す) を戻します。	410 ページの『DAYOFWEEK』
DAYOFWEEK_ISO	値から曜日 (1 は月曜日を表し、7 は日曜日を表す) を戻します。	411 ページの『DAYOFWEEK_ISO』
DAYOFYEAR	値から年間通算日を戻します。	412 ページの『DAYOFYEAR』
DAYS	日付の整数表現を戻します。	413 ページの『DAYS』
EXTRACT	日時値の指定された部分を戻します。	460 ページの『EXTRACT』
HOUR	値の時間の部分を戻します。	486 ページの『HOUR』
JULIAN_DAY	紀元前 4712 年 1 月 1 日から引数で指定された日付までの日数を表す整数値を戻します。	516 ページの『JULIAN_DAY』
LAST_DAY	引数の月の最後の日を表す日付またはタイム・スタンプを戻します。	518 ページの『LAST_DAY』
MICROSECOND	値のマイクロ秒の部分を戻します。	543 ページの『MICROSECOND』
MIDNIGHT_SECONDS	真夜中から指定の時刻値までの間の秒数を表す整数値を戻します。	544 ページの『MIDNIGHT_SECONDS』
MINUTE	値の分の部分を戻します。	546 ページの『MINUTE』
MONTH	値の月の部分を戻します。	549 ページの『MONTH』
MONTHNAME	値の月の名前の部分を戻します。	550 ページの『MONTHNAME』
MONTHS_BETWEEN	2 つの日付間の月数の見積もりを戻します。	551 ページの『MONTHS_BETWEEN』
NEXT_DAY	2 番目の引数で指定された日より後の最初の平日を表す日付またはタイム・スタンプ値を戻します。	565 ページの『NEXT_DAY』
NOW	時点の刻時機構の読み取りに基づくタイム・スタンプを戻します。	568 ページの『NOW』
QUARTER	日付が存在する四半期を表す整数を戻します。	583 ページの『QUARTER』
ROUND_TIMESTAMP	指定された単位に丸められたタイム・スタンプを戻します。	609 ページの『ROUND_TIMESTAMP』
SECOND	値の秒の部分を戻します。	623 ページの『SECOND』
TIMESTAMP_FORMAT	指定されたストリングの形式に応じて、タイム・スタンプを文字ストリングで表現したものからタイム・スタンプを戻します。	646 ページの『TIMESTAMP_FORMAT』
TIMESTAMPDIFF	2 つのタイム・スタンプの差に基づいて、間隔の見積数を戻します。	653 ページの『TIMESTAMPDIFF』
TRUNC_TIMESTAMP	指定された単位に切り捨てられたタイム・スタンプを戻します。	665 ページの『TRUNC_TIMESTAMP』
WEEK	値から年間通算週を戻します。(週は日曜日から始まる)	698 ページの『WEEK』

表 39. 日付時刻スカラー関数 (続き)

関数	説明	参照
WEEK_ISO	値から年間通算週を戻します。(週は月曜日から始まる)	699 ページの『WEEK_ISO』
YEAR	値の年の部分を戻します。	737 ページの『YEAR』

表 40. JSON スカラー関数

関数	説明	参照
BSON_TO_JSON	形式設定された BSON データを含むストリングを、JSON として形式設定されたデータを含む文字ストリングに変換します。	371 ページの『BSON_TO_JSON』
JSON_ARRAY	配列エレメントを明示的にリストするか、または照会を使用することで、JSON 配列を生成します。	498 ページの『JSON_ARRAY』
JSON_OBJECT	指定されたキー:値 のペアを使用して、JSON オブジェクトを生成します。	502 ページの『JSON_OBJECT』
JSON_QUERY	SQL/JSON パス式を使用して、指定された JSON テキストから SQL/JSON 値を戻します。	506 ページの『JSON_QUERY』
JSON_TO_BSON	形式設定された JSON データを含むストリングを、BSON として形式設定されたデータを含むバイナリー・ストリングに変換します。	511 ページの『JSON_TO_BSON』
JSON_VALUE	SQL/JSON パス式を使用して、JSON テキストから SQL スカラー値を戻します。	512 ページの『JSON_VALUE』

表 41. その他のスカラー関数

関数	説明	参照
CARDINALITY	配列のエレメント数に相当する値を返します。	372 ページの『CARDINALITY』
COALESCE	NULL でない最初の引数を戻します。	390 ページの『COALESCE』
CONTAINS	テキスト索引で一致項目が見つかったかどうかを示す標識を返します。	394 ページの『CONTAINS』
DATABASE	現行サーバーを戻します。	402 ページの『DATABASE』
GENERATE_UNIQUE	関数の他の実行と比較して固有であるビット文字ストリングを戻します。	362 ページの『ATAN2』
GET_BLOB_FROM_FILE	ソース・ストリーム・ファイルまたはソース物理ファイルのメンバーのデータを組み込んだ BLOB ロケーターを返します。	469 ページの『GET_BLOB_FROM_FILE』
GET_CLOB_FROM_FILE	ソース・ストリーム・ファイルまたはソース物理ファイルのメンバーのデータを組み込んだ CLOB ロケーターを返します。	470 ページの『GET_CLOB_FROM_FILE』
GET_DBCLOB_FROM_FILE	ソース・ストリーム・ファイルまたはソース物理ファイルのメンバーのデータを組み込んだ DBCLOB ロケーターを返します。	471 ページの『GET_DBCLOB_FROM_FILE』

表 41. その他のスカラー関数 (続き)

関数	説明	参照
GET_XML_FILE	ソース・ストリーム・ファイルまたはソース物理ファイルのメンバーのデータを組み込んだ BLOB ロケーターを返します (データを UTF-8 に変換します)。	472 ページの『GET_XML_FILE』
HASH_MD5、HASH_SHA1、HASH_SHA256、HASH_SHA512	ハッシュ関数は、入力データの 128 ビット、160 ビット、256 ビット、または 512 ビットのハッシュを返します。	480 ページの『HASH_MD5、HASH_SHA1、HASH_SHA256、および HASH_SHA512』
HEX	値の 16 進数表現を戻します。	484 ページの『HEX』
IDENTITY_VAL_LOCAL	識別列の最新割り当て値を戻します。	487 ページの『IDENTITY_VAL_LOCAL』
IFNULL	NULL でない最初の引数を戻します。	492 ページの『IFNULL』
LENGTH	値の長さを戻します。	522 ページの『LENGTH』
MAX	値の集合の最大値を戻します。	541 ページの『MAX』
MAX_CARDINALITY	配列に組み込めるエレメントの最大数に相当する値を返します。	542 ページの『MAX_CARDINALITY』
MIN	値の集合の最小値を戻します。	545 ページの『MIN』
NULLIF	引数が等しい場合は NULL を戻します。等しくない場合は、最初の引数の値を戻します。	569 ページの『NULLIF』
RAISE_ERROR	指定の SQLSTATE およびメッセージ・テキストでエラーを通知します。	585 ページの『RAISE_ERROR』
RID	行の相対レコード番号を BIGINT として戻します。	604 ページの『RID』
RRN	行の相対レコード番号を DECIMAL(15,0) として戻します。	617 ページの『RRN』
SCORE	テキスト索引で一致項目が見つかった頻度を示す標識を返します。	620 ページの『SCORE』
SYS_CONNECT_BY_PATH	階層照会のためのルート行から現在の行へのパスを表すストリングを戻します。	815 ページの『SYS_CONNECT_BY_PATH』
TABLE_NAME	別名に対して検出されたオブジェクトの非修飾名を戻します。	639 ページの『TABLE_NAME』
TABLE_SCHEMA	別名に対して検出されたオブジェクトのスキーマ名を戻します。	640 ページの『TABLE_SCHEMA』
TRIM_ARRAY	末尾からいくつかのエレメントを削除した配列のコピーを返します。	662 ページの『TRIM_ARRAY』
VALUE	NULL でない最初の引数を戻します。	668 ページの『VALUE』
VERIFY_GROUP_FOR_USER	指定されたプロファイルのリスト内にユーザーがあるかどうかを示す標識を戻します。	696 ページの『VERIFY_GROUP_FOR_USER』
WRAP	難読化された DDL ステートメント・テキスト	700 ページの『WRAP』

表 42. *MQSeries* スカラー関数

関数	説明	参照
MQREAD	キューからメッセージを除去せずに、指定された MQSeries® ロケーション (VARCHAR の戻り値) からメッセージを戻します。	553 ページの『MQREAD』
MQREADCLOB	キューからメッセージを除去せずに、指定された MQSeries ロケーション (CLOB の戻り値) からメッセージを戻します。	555 ページの『MQREADCLOB』
MQRECEIVE	キューからメッセージを除去して、指定された MQSeries ロケーション (VARCHAR の戻り値) からメッセージを戻します。	557 ページの『MQRECEIVE』
MQRECEIVECLOB	キューからメッセージを除去して、指定された MQSeries ロケーション (CLOB の戻り値) からメッセージを戻します。	559 ページの『MQRECEIVECLOB』
MQSEND	メッセージを指定された MQSeries ロケーションに送信します。	561 ページの『MQSEND』

表 43. 数値スカラー関数

関数	説明	参照
ABS	数値の絶対値を戻します。	353 ページの『ABS』
ACOS	数値のアークコサイン (逆余弦) をラジアンで戻します。	354 ページの『ACOS』
ANTILOG	数値の逆対数 (底 10) を戻します。	357 ページの『ANTILOG』
ASIN	数値のアークサイン (逆正弦) をラジアンで戻します。	359 ページの『ASIN』
ATAN	数値のアークタンジェント (逆正接) をラジアンで戻します。	360 ページの『ATAN』
ATANH	数値の双曲線アークタンジェント (双曲線逆正接) をラジアンで戻します。	361 ページの『ATANH』
ATAN2	x 座標と y 座標の逆正接を、ラジアンで表された角度として戻します。	362 ページの『ATAN2』
BITAND、BITANDNOT、BITOR、BITXOR、BITNOT	入力引数の 2 の補数表記を使用して、ビット単位演算に基づく底 10 の値を返します。	366 ページの『BITAND、BITANDNOT、BITOR、BITXOR、および BITNOT』
CEILING	数値より大きいか等しい最小の整数値を戻します。	373 ページの『CEILING または CEIL』
COMPARE_DECFLOAT	2 つの 10 進浮動小数点値の比較方法の指示を戻します。	391 ページの『COMPARE_DECFLOAT』
COS	数値のコサイン (余弦) を戻します。	397 ページの『COS』
COSH	数値の双曲線コサイン (双曲線余弦) を戻します。	398 ページの『COSH』
DECFLOAT_FORMAT	指定された形式を使用した入力ストリングの変換処理に基づく DECFLOAT(34) 値を戻す	425 ページの『DECFLOAT_FORMAT』
DECFLOAT_SORTKEY	10 進浮動小数点値のソートに使用できる整数値を戻します。	428 ページの『DECFLOAT_SORTKEY』

表 43. 数値スカラー関数 (続き)

関数	説明	参照
DEGREES	角度の度数を戻します。	436 ページの『DEGREES』
DIGITS	数値の絶対値の文字ストリング表現を戻します。	438 ページの『DIGITS』
EXP	自然対数の底 (e) を引数の指定だけ累乗した値を戻します。	459 ページの『EXP』
FLOOR	数値より小さいまたは等しい、最大の整数値を戻します。	466 ページの『FLOOR』
LN	数値の自然対数を戻します。	524 ページの『LN』
LOG10	数値の共通対数 (底 10) を戻します。	531 ページの『LOG10』
MOD	最初の引数を 2 番目の引数で除算した剰余を戻します。	547 ページの『MOD』
MULTIPLY_ALT	最初の引数と 2 番目の引数を乗算して、その積を戻します。	563 ページの『MULTIPLY_ALT』
NORMALIZE_DECFLOAT	10 進浮動小数点値を最も単純な形式で戻します。	567 ページの『NORMALIZE_DECFLOAT』
PI	π の値を戻します。	574 ページの『PI』
POWER	最初の引数を 2 番目の引数だけ累乗した結果を戻します。	579 ページの『POWER』
QUANTIZE	提供された値に応じてフォーマットされた 10 進浮動小数点値を戻します。	581 ページの『QUANTIZE』
RADIANS	度で表された引数に対してラジアン数を戻します。	584 ページの『RADIANS』
RAND	乱数を戻します。	586 ページの『RAND』
ROUND	指定の小数部の桁数まで丸められた数値を戻します。	607 ページの『ROUND』
SIGN	数値の符号を戻します。	625 ページの『SIGN』
SIN	数値のサイン (正弦) を戻します。	626 ページの『SIN』
SINH	数値の双曲線サイン (双曲線正弦) を戻します。	627 ページの『SINH』
SQRT	数値の平方根を戻します。	632 ページの『SQRT』
TAN	数値のタンジェント (正接) を戻します。	641 ページの『TAN』
TANH	数値の双曲線タンジェント (双曲線正接) を戻します。	642 ページの『TANH』
TOTALORDER	2 つの 10 進浮動小数点値の配列指示を戻します。	656 ページの『TOTALORDER』
TRUNCATE または TRUNC	指定の小数部の桁数で切り捨てられた数値を戻します。	663 ページの『TRUNCATE または TRUNC』

表 44. パーティション化スカラー関数

関数	説明	参照
DATAPARTITIONNAME	行が置かれているパーティションの名前を戻します。	403 ページの『DATAPARTITIONNAME』
DATAPARTITIONNUM	行のパーティション番号を戻します。	404 ページの『DATAPARTITIONNUM』

表 44. パーティション化スカラー関数 (続き)

関数	説明	参照
DBPARTITIONNAME	行が置かれているリレーショナル・データベースの名前を戻します。	421 ページの『DBPARTITIONNAME』
DBPARTITIONNUM	行のノード番号を戻します。	422 ページの『DBPARTITIONNUM』
HASH_VALUES	値の集合のパーティション番号を戻します。	482 ページの『HASH_VALUES』
HASHED_VALUE	行のパーティション・マップ索引番号を戻します。	483 ページの『HASHED_VALUE』

表 45. ストリング・スカラー関数

関数	説明	参照
ASCII	引数の左端の文字の ASCII コード値を 1 つの整数として戻します。	358 ページの『ASCII』
BIT_LENGTH	ストリング式の長さをビット数で戻します。	368 ページの『BIT_LENGTH』
CHARACTER_LENGTH	ストリング式の長さを戻します。	382 ページの『CHARACTER_LENGTH』
CHR	引数で指定された ASCII コード値をもつ EBCDIC 文字を戻します。	383 ページの『CHR』
CONCAT	2 つのストリングの連結であるストリングを戻します。	393 ページの『CONCAT』
DECRYPT_BIT、DECRYPT_BINARY、DECRYPT_CHAR、および DECRYPT_DB	暗号化されたストリングを暗号化解除します。	432 ページの『DECRYPT_BIT、DECRYPT_BINARY、DECRYPT_CHAR、および DECRYPT_DB』
DIFFERENCE	2 つのストリングの音の相違を表す値を戻します。	437 ページの『DIFFERENCE』
ENCRYPT および ENCRYPT_RC2	RC2 暗号化アルゴリズムを使用してストリングを暗号化します。	453 ページの『ENCRYPT_RC2』
ENCRYPT_AES	AES 暗号化アルゴリズムを使用してストリングを暗号化します。	450 ページの『ENCRYPT_AES』
ENCRYPT_TDES	Triple-DES 暗号化アルゴリズムを使用してストリングを暗号化します。	456 ページの『ENCRYPT_TDES』
GETHINT	暗号化されたストリングからヒントを戻します。	473 ページの『GETHINT』
INSERT	サブストリングが削除され、代わりに新しいストリングが挿入されたストリングを戻します。	493 ページの『INSERT』
LAND	引数である 2 つのストリングの論理 AND であるストリングを戻します。	517 ページの『LAND』
LCASE	すべての文字を小文字に変換したストリングを戻します。	519 ページの『LCASE』
LEFT	ストリングから左端の文字を戻します。	520 ページの『LEFT』
LNOT	引数ストリングの論理否定 (論理 NOT) であるストリングを戻します。	525 ページの『LNOT』

表 45. ストリング・スカラー関数 (続き)

関数	説明	参照
LOCATE	別のストリング内のあるストリングの開始位置を戻します。	526 ページの『LOCATE』
LOCATE_IN_STRING	LOCATE_IN_STRING 関数は、あるストリングの、別のストリング内での開始位置を戻します。	528 ページの『LOCATE_IN_STRING』
LOR	引数である 2 つのストリングの論理 OR であるストリングを戻します。	532 ページの『LOR』
LOWER	すべての文字を小文字に変換したストリングを戻します。	533 ページの『LOWER』
LPAD	左側に埋め込んだストリングを戻します。	534 ページの『LPAD』
LTRIM	別のストリングの先頭からブランクまたは 16 進数のゼロが削除されたストリングを戻します。	538 ページの『LTRIM』
OCTET_LENGTH	ストリング式の長さをオクテット数で戻します。	570 ページの『OCTET_LENGTH』
OVERLAY	元のストリングから多くの文字が削除され、新しいストリングが挿入されたストリングを戻します。	571 ページの『OVERLAY』
POSITION	別のストリング内のあるストリングの開始位置を戻します。	575 ページの『POSITION』
POSSTR	別のストリング内のあるストリングの開始位置を戻します。	577 ページの『POSSTR』
REGEXP_COUNT	ストリングで正規表現パターンが一致した回数を戻します。	589 ページの『REGEXP_COUNT』
REGEXP_INSTR	一致したサブストリングの開始位置、または終了後の位置を戻します。	591 ページの『REGEXP_INSTR』
REGEXP_REPLACE	ソース・ストリングで正規表現パターンを置換ストリングに置換して変更したものを戻します。	594 ページの『REGEXP_REPLACE』
REGEXP_SUBSTR	ストリングで正規表現パターンに一致するサブストリングのオカレンスを戻します。	597 ページの『REGEXP_SUBSTR』
REPEAT	何回も反復される別のストリングで構成されるストリングを戻します。	600 ページの『REPEAT』
REPLACE	あるストリングの出現箇所すべてが別のストリングで置き換えられるストリングを戻します。	602 ページの『REPLACE』
RIGHT	ストリングから右端の文字を戻します。	605 ページの『RIGHT』
RPAD	右側に埋め込んだストリングを戻します。	613 ページの『RPAD』
RTRIM	別のストリングの終わりからブランクまたは 16 進数のゼロが削除されたストリングを戻します。	618 ページの『RTRIM』
SOUNDEX	引数内のワードの音を表す文字コードを戻します。	630 ページの『SOUNDEX』
SPACE	指定されたブランク数からなる文字ストリングを戻します。	631 ページの『SPACE』
STRIP	ストリング式の後部または前部から、ブランクまたは指定した文字を除去します。	633 ページの『STRIP』
SUBSTR	ストリングのサブストリングを戻します。	634 ページの『SUBSTR』

表 45. ストリング・スカラー関数 (続き)

関数	説明	参照
SUBSTRING	ストリングのサブストリングを戻します。	637 ページの『SUBSTRING』
TRANSLATE	ストリングの中の 1 つ以上の文字を他の文字に変換したストリングを戻します。	657 ページの『TRANSLATE』
TRIM	ストリング式の先頭、末尾、または両方から、ブランクまたは別の指定された文字を除去します。	660 ページの『TRIM』
UCASE	すべての文字を大文字に変換したストリングを戻します。	666 ページの『UCASE』
UPPER	すべての文字を大文字に変換したストリングを戻します。	667 ページの『UPPER』
VARBINARY_FORMAT	フォーマットされた文字ストリングのバイナリー・ストリング表現を戻します。	670 ページの『VARBINARY_FORMAT』
VARCHAR_FORMAT	最初の引数の文字ストリング表現を、オプションのフォーマット設定ストリングで示されたフォーマットで戻します。	678 ページの『VARCHAR_FORMAT』
VARCHAR_FORMAT_BINARY	フォーマットされたビット・ストリングの文字ストリング表現を戻します。	688 ページの『VARCHAR_FORMAT_BINARY』
XOR	引数である 2 つのストリングの論理 XOR であるストリングを戻します。	732 ページの『XOR』

表 46. XML スカラー関数

関数	説明	参照
XMLATTRIBUTES	引数から XML 属性を構成します。	702 ページの『XMLATTRIBUTES』
XMLCOMMENT	内容として入力引数が含まれている XML 値を返します。	704 ページの『XMLCOMMENT』
XMLCONCAT	可変数の XML 入力引数の連結を含む XML シーケンスを戻します。	705 ページの『XMLCONCAT』
XMLDOCUMENT	整形式 XML 文書である XML 値を戻します。	707 ページの『XMLDOCUMENT』
XMLELEMENT	XML エレメントである XML 値を戻します。	708 ページの『XMLELEMENT』
XMLFOREST	XML エレメントのシーケンスである XML 値を戻します。	712 ページの『XMLFOREST』
XMLNAMESPACES	引数からネーム・スペース宣言を構成します。	715 ページの『XMLNAMESPACES』
XMLPARSE	引数を XML 文書として構文解析した結果の XML 値を戻します。	718 ページの『XMLPARSE』
XMLPI	単一の処理命令を含む XML 値を戻します。	720 ページの『XMLPI』
XMLROW	整形式 XML 文書である XML 値を戻します。	721 ページの『XMLROW』
XMLSERIALIZE	直列化した XML 値を指定のデータ・タイプとして返します。	723 ページの『XMLSERIALIZE』
XMLTEXT	入力引数の値を含む XML 値を戻します。	726 ページの『XMLTEXT』

表 *46. XML* スカラー関数 (続き)

関数	説明	参照
XMLVALIDATE	XML スキーマの検証から得られた情報で補強した XML 値を返します。	727 ページの『XMLVALIDATE』
XSLTRANSFORM	XML 文書を別のデータ・フォーマットに変換します。	733 ページの『XSLTRANSFORM』

表 *47.* 表関数

関数	説明	参照
BASE_TABLE	指定された別名の表名およびスキーマ名を含んでいる 1 つの行がある表を戻します。	742 ページの『BASE_TABLE』
JSON_TABLE	SQL/JSON パス式の評価から表を戻します。	744 ページの『JSON_TABLE』
MQREADALL	キューからメッセージを除去せずに、VARCHAR 列で指定された MQSeries ロケーションからのメッセージおよびメッセージ・メタデータが入った表を戻します。	755 ページの『MQREADALL』
MQREADALLCLOB	キューからメッセージを除去せずに、CLOB 列で指定された MQSeries ロケーションからのメッセージおよびメッセージ・メタデータが入った表を戻します。	757 ページの『MQREADALLCLOB』
MQRECEIVEALL	キューからメッセージを除去して、VARCHAR 列で指定された MQSeries ロケーションからのメッセージおよびメッセージ・メタデータが入った表を戻します。	759 ページの『MQRECEIVEALL』
MQRECEIVEALLCLOB	キューからメッセージを除去して、CLOB 列で指定された MQSeries ロケーションからのメッセージおよびメッセージ・メタデータが入った表を戻します。	762 ページの『MQRECEIVEALLCLOB』
XMLTABLE	XPath 式の評価から表を戻します。	765 ページの『XMLTABLE』

集約関数

集約関数は、(データの列のような) 値の集合を取り、この値の集合から単一値の結果を戻します。

以下の説明は、COUNT(*) および COUNT_BIG(*) を除くすべての集約関数に適用されます。

- 集約関数の引数は、1 つの式から得られた値の集合です。式には列を含めることはできますが、他の集約関数を含めることはできません。この集合の有効範囲は、第 6 章『照会』で説明しているグループまたは中間結果表です。
- 照会で GROUP BY 文節が指定され、FROM、WHERE、GROUP BY、および HAVING 文節の中間結果が空の結果表である場合、集約関数は適用されず、照会の結果は空の表になります。
- 照会で GROUP BY 文節が指定されず、FROM、WHERE、および HAVING 文節の中間結果が空の結果表である場合、空の結果表に集約関数が適用されます。例えば、次の SELECT ステートメントの結果は、部門 D01 には従業員がいないため、空の結果表に適用されます。

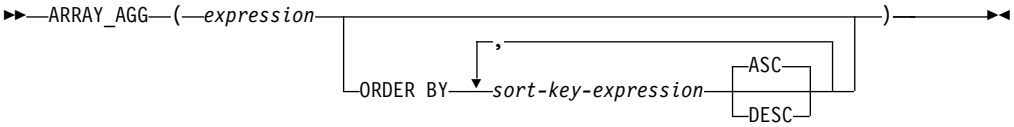
  ```
  SELECT COUNT(DISTINCT JOB)
    FROM EMPLOYEE
    WHERE WORKDEPT = 'D01'
  ```

- キーワード DISTINCT は、この関数の引数ではなく、この関数の適用に先立って決められる演算の仕様です。 DISTINCT が指定される場合は、重複する値は除かれます。 ALL が暗黙、または明示的に指定されている場合には、重複する値は除去されません。

  数の上で等しい 10 進浮動小数点値の DISTINCT 節を解釈する場合、値内の有効数字の桁数は考慮されません。例えば、10 進浮動小数点数 123.00 は、10 進浮動小数点数 123 とは異なりません。照会から戻された数の表記は、検出された表記のいずれか 1 つ (例えば、123.00 か 123 のどちらか) になります。
- WHERE 文節の中で集約関数を使用できるのは、その文節が HAVING 文節の副照会の一部であり、式の中で指定する列名がグループに対する相関参照である場合だけです。式に複数の列名が含まれている場合は、それらの列名は同じグループに対する相関参照でなければなりません。

ARRAY_AGG

ARRAY_AGG 関数は、一連のエレメントを配列として集約します。



expression
CREATE TYPE (配列) ステートメントで指定できるデータ・タイプの値を返す式。

ORDER BY
集約で処理される、同じグループ化セットからの行の順序を指定します。ORDER BY 文節を指定しない場合、または ORDER BY 文節ではソート・キー値の順序を区別できない場合、同じグループ内の行は任意に順序付けられます。

sort-key-expression
列名または式のどちらかのソート・キー値を指定します。列または式のデータ・タイプは、DATALINK 値または XML 値であってはなりません。

集約したエレメントの順序付けは、ソート・キーの値に基づいて行われます。

ソート・キー式 の長さ属性の合計は 3.5 ギガバイトを超えてはなりません。

ARRAY_AGG 関数を含むステートメントの実行時に *HEX 以外の照合順序が有効で、しかもソート・キー式 が SBCS データ、混合データ、または Unicode データの場合、結果は重み付けされた値の比較によって求められます。この重み付けされた値は、ソート・キー式 に照合順序を適用して得られます。

SQL プロシージャーまたは SQL 関数で ARRAY_AGG 関数を指定できるのは、以下の特定のコンテキストに限られます。

- SELECT INTO ステートメントの *select-clause*
- SET ステートメントの右側のスカラー副照会の *select-clause*

ARRAY_AGG を使用する SELECT に DISTINCT 文節を組み込むことはできません。

例

配列タイプと表を以下のように作成するとします。

```
CREATE TYPE PHONELIST AS DECIMAL(10,0) ARRAY[10]

CREATE TABLE EMPLOYEE (
    ID          INTEGER NOT NULL,
    PRIORITY    INTEGER NOT NULL,
    PHONENUMBER DECIMAL(10,0),
    PRIMARY KEY (ID, PRIORITY) )
```

SELECT INTO ステートメントを使用して従業員の連絡先の電話番号を優先順位に基づいて並べたリストを返すプロシージャーを作成します。

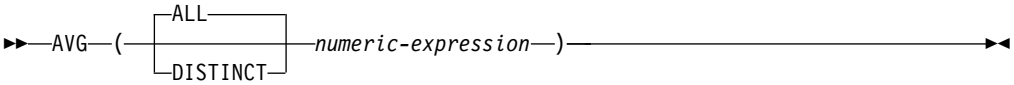
```
CREATE PROCEDURE GETPHONENUMBERS
    (IN EMPID INTEGER,
     OUT NUMBERS PHONELIST)
  BEGIN
    SELECT ARRAY_AGG(PHONENUMBER ORDER BY PRIORITY) INTO NUMBERS
        FROM EMPLOYEE
        WHERE ID = EMPID;
  END
```

SET ステートメントを使用して従業員の連絡先の電場番号を任意の順序で並べたリストを返すプロシージャーを作成します。

```
CREATE PROCEDURE GETPHONENUMBERS
    (IN EMPID INTEGER,
     OUT NUMBERS PHONELIST)
  BEGIN
    SET NUMBERS =
      (SELECT ARRAY_AGG(PHONENUMBER)
        FROM EMPLOYEE
        WHERE ID = EMPID);
  END
```

AVG

AVG 関数は、数値の集合の平均値を戻します。



numeric-expression
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数が文字ストリングまたはグラフィック・ストリングの場合、関数を評価する前に DECFLOAT(34) にキャストされます。引数値の合計は、結果のデータ・タイプの範囲内でなければなりません。

結果のデータ・タイプは、原則として引数の値のデータ・タイプと同じになります。ただし、以下の場合は、結果のデータ・タイプが引数の値のデータ・タイプとは異なるものになります。

- 引数値が DECFLOAT(16) の場合、結果は DECFLOAT(34) になります。
- 引数値が単精度浮動小数点の場合、結果は倍精度浮動小数点になります。
- 引数値が短整数である場合は、結果は長整数になる。
- 結果は、引数が、10 進数または位取りがゼロ以外の 2 進数で、精度が *p*、位取りが *s* である場合は、10 進数になります。結果の精度は、*p-s+* min(*ms, mp-p+s*) です。結果の位取りは、min(*ms, mp-p+s*) です。

p、*s*、*ms*、および *mp* の値については、 198 ページの『SQL での 10 進数演算』を参照してください。

この関数は、引数の値から NULL 値を除いた値の集合に対して適用されます。DISTINCT を使用すると、重複する値は除かれます。

結果が、NULL になることもあります。値の集合が空である場合は、結果は NULL 値です。それ以外の場合は、結果は集合の値の平均値です。

値を集計する順序は定義されていませんが、すべての中間結果は結果のデータ・タイプの範囲内になければなりません。

結果のデータ・タイプが整数である場合、平均値の小数部分は失われます。

注

特殊値 **DECFLOAT** を含む結果**:** 引数のデータ・タイプが 10 進浮動小数点であり、特殊値 sNaN または -sNaN、もしくは +Infinity と -Infinity の両方が集約に含まれる場合、エラーまたは警告が戻されます。それ以外の場合、+NaN または -NaN が検出されれば、結果は +NaN または -NaN になります。+Infinity または -Infinity が検出されれば、結果は +Infinity または -Infinity になります。

例

- PROJECT 表を使用して、ホスト変数 AVERAGE (DECIMAL(5,2)) に、部門 (DEPTNO) D11 のプロジェクトの平均要員水準 (PRSTAFF) を設定します。

```
SELECT AVG(PRSTAFF)
  INTO :AVERAGE
  FROM PROJECT
  WHERE DEPTNO = 'D11'
```

上記の結果、AVERAGE は 4.25 (つまり 17/4) にセットされます。

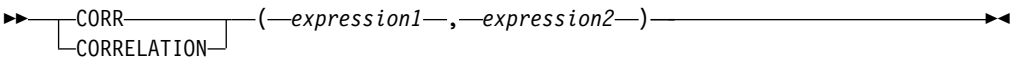
- 表 PROJECT を使用して、ホスト変数 ANY_CALC に、部門 (DEPTNO)「D11」のそれぞれ固有の要員水準の値 (PRSTAFF) の平均値をセットします。

```
SELECT AVG(DISTINCT PRSTAFF)
  INTO :ANY_CALC
  FROM PROJECT
  WHERE DEPTNO = 'D11'
```

上記の結果、ANY_CALC は 4.66 (つまり、14/3) に設定されます。

CORR または CORRELATION

CORRELATION 関数は、数値の組の集合に関する相関係数を戻します。



expression1
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数が文字ストリングまたはグラフィック・ストリングの場合、関数を評価する前に DECFLOAT(34) にキャストされます。

expression2
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数が文字ストリングまたはグラフィック・ストリングの場合、関数を評価する前に DECFLOAT(34) にキャストされます。

いずれかの引数が 10 進浮動小数点数の場合、関数の結果は DECFLOAT(34) となります。それ以外の場合は、関数の結果は倍精度浮動小数点数です。結果が、NULL になることもあります。値が NULL 値でない場合、結果は -1 から 1 になります。

この関数は、引数の値から導出されたペアの集合 (*expression1, expression2*) から、*expression1* または *expression2* のどちらかが NULL であるすべてのペアを除外したものに対して適用されます。

この関数が空のセットに適用された場合や、 STDDEV(*expression1*) または STDDEV(*expression2*) のどちらかがゼロに等しい場合、結果は NULL 値になります。それ以外の場合、結果はその集合内にある値ペアの相関係数になります。結果は、次のようにして割り出されます。

1. sdexp1 を STDDEV(*expression1*) の結果に、sdexp2 を STDDEV(*expression2*) の結果にします。
2. CORRELATION(*expression1, expression2*) の結果は以下になります。

   ```
   COVARIANCE(expression1, expression2) / (sdexp1 * sdexp2)
   ```

値を集計する順序は定義されていませんが、すべての中間結果は結果のデータ・タイプの範囲内になければなりません。

注

特殊値 **DECFLOAT** を含む結果**:** 引数のデータ・タイプが 10 進浮動小数点であり、特殊値 sNaN または -sNaN、もしくは +Infinity と -Infinity の両方が集約に含まれる場合、エラーまたは警告が戻されます。それ以外の場合、+NaN または -NaN が検出されれば、結果は +NaN または -NaN になります。+Infinity または -Infinity が検出されれば、結果は +Infinity または -Infinity になります。

代替構文**:** SQL 規格に準拠して CORR を使用する必要があります。

例

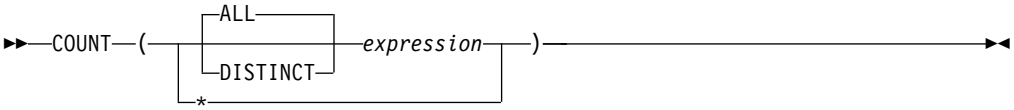
- EMPLOYEE 表を使用して、ホスト変数 CORRLN を、部門 'A00' の従業員の給与と賞与の間に見られる相関に設定します。

```
SELECT CORRELATION(SALARY, BONUS)
  INTO :CORRLN
  FROM EMPLOYEE
  WHERE WORKDEPT = 'A00';
```

サンプル表を使用した場合、CORRLN は 0.9760236077658643 に設定されます。

COUNT

COUNT 関数は、行または値の集合の中にある行の数または値の数を戻します。



expression
引数の値には、データ・リンクを除く任意の組み込みデータ・タイプを指定できます。 XML は、COUNT (DISTINCT *expression*) で使用できません。

この関数の結果は長整数であり、結果の値は長整数の値の範囲内になければなりません。結果が NULL になることはありません。表が分散表の場合には、結果は DECIMAL(15,0) になります。分散表の詳細については、「Db2 UDB for iSeries マルチ・システム」トピック集を参照してください。

COUNT(*) の引数は、行の集合です。この結果は、その集合の行の数になります。この行の数には、NULL 値しか入っていない行も含まれます。

COUNT(式) または COUNT(ALL 式) の引数は、値の集合です。この関数は、引数の値から NULL 値を除いた値の集合に適用されます。結果はその集合の非 NULL 値の個数です。これには重複した個数も含まれます。

COUNT(DISTINCT 式) の引数は、値の集合です。この関数は、引数値から NULL 値および重複する値を除いて得られる値の集合に適用されます。結果はその集合の値の個数です。

COUNT(DISTINCT 式) を含むステートメントの実行時に *HEX 以外の照合順序が有効で、しかも引数が SBCS データ、混合データ、または Unicode データの場合、結果は、集合の各値の重み付けされた値の比較によって求められます。値の重み付けは、該当の照合順序に基づいています。

例

- EMPLOYEE 表を使用して、SEX 列の値が「F」になっている行の数をホスト変数 FEMALE (INTEGER) に設定します。

```
SELECT COUNT(*)
  INTO :FEMALE
  FROM EMPLOYEE
  WHERE SEX = 'F'
```

  上記の結果、FEMALE が 19 にセットされます。

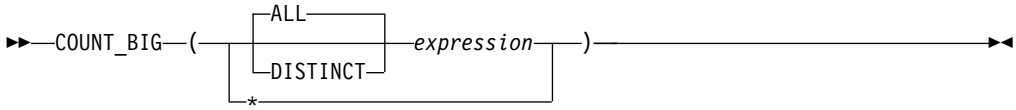
- 表 EMPLOYEE を使用して、ホスト変数 FEMALE_IN_DEPT (INTEGER) に、少なくとも一人の女性がいる部門 (WORKDEPT) の数をセットします。

```
SELECT COUNT(DISTINCT WORKDEPT)
  INTO :FEMALE_IN_DEPT
  FROM EMPLOYEE
  WHERE SEX='F'
```

  上記の結果、FEMALE_IN_DEPT は 6 にセットされます (部門 A00、 C01、 D11、 D21、 E11、および E21 に、それぞれ女性が少なくとも 1 人はいます)。

COUNT_BIG

COUNT_BIG 関数は、行または値の集合の中にある行の数または値の数を戻します。この関数は COUNT 関数に類似していますが、COUNT 関数とは異なり、結果の値は整数の最大値より大きい値をとることができます。



expression
引数の値には、データ・リンクを除く任意の組み込みデータ・タイプを指定できます。 XML は、COUNT_BIG(DISTINCT *expression*) で使用できません。

関数の結果は、精度が 31 で位取りが 0 である 10 進数です。結果は NULL であってはなりません。

COUNT_BIG(*) の引数は、行の集合です。この結果は、その集合の行の数になります。この行の数には、NULL 値しか入っていない行も含まれます。

COUNT_BIG(*expression*) の引数は、値の集合です。この関数は、引数の値から NULL 値を除いた値の集合に適用されます。結果はその集合の値の個数です。

COUNT_BIG(DISTINCT *expression*) を含むステートメントの実行時に *HEX 以外の照合順序が有効で、しかも引数が SBCS データ、混合データ、または Unicode データの場合、結果は、集合の各値の重み付けされた値の比較によって求められます。値の重み付けは、該当の照合順序に基づいています。

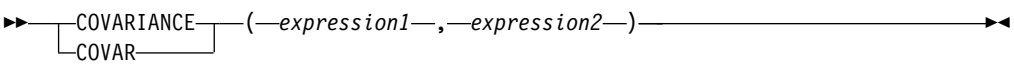
例

- COUNT の例を参照して、COUNT を COUNT_BIG と読み替えてください。結果のデータ・タイプを除いて、結果は同じです。
- 特定の列上でカウントするためには、ソース化関数は列のタイプを指定しなければなりません。この例では、CREATE FUNCTION ステートメントが、CHAR として定義された任意の列を取るソース化関数を作成し、COUNT_BIG を使用してカウントを実行し、その結果を倍精度の浮動小数点数として戻します。以下の照会は、サンプルの従業員表内で固有の部門数をカウントします。

```
CREATE FUNCTION RICK.COUNT(CHAR(19)) RETURNS DOUBLE
  SOURCE QSYS2.COUNT_BIG(CHAR());

SET CURRENT PATH RICK, SYSTEM PATH

SELECT COUNT(DISTINCT WORKDEPT) FROM EMPLOYEE;
```

COVARIANCE または COVAR

COVARIANCE 関数は、数値の組の集合に関する (集団) 共分散を戻します。



expression1
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数が文字ストリングまたはグラフィック・ストリングの場合、関数を評価する前に DECFLOAT(34) にキャストされます。

expression2
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数が文字ストリングまたはグラフィック・ストリングの場合、関数を評価する前に DECFLOAT(34) にキャストされます。

いずれかの引数が 10 進浮動小数点数の場合、関数の結果は DECFLOAT(34) となります。それ以外の場合は、関数の結果は倍精度浮動小数点数です。結果が、NULL になることもあります。

この関数は、引数の値から導出されたペアの集合 (*expression1, expression2*) から、*expression1* または *expression2* のどちらかが NULL であるすべてのペアを除外したものに対して適用されます。

この関数が空のセットに適用されると、結果は NULL 値になります。それ以外の場合、結果はそのセット内の値ペアの共分散になります。結果は、次のようにして割り出されます。

1. avgexp1 を AVG(*expression1*) の結果に、 avgexp2 を AVG(*expression2*) の結果にします。
2. COVARIANCE(*expression1, expression2*) の結果は以下になります。

   ```
   AVG( (expression1 - avgexp1) * (expression2 - avgexp2) )
   ```

値を集計する順序は定義されていませんが、すべての中間結果は結果のデータ・タイプの範囲内になければなりません。

注

特殊値 **DECFLOAT** を含む結果**:** 引数のデータ・タイプが 10 進浮動小数点であり、特殊値 sNaN または -sNaN、もしくは +Infinity と -Infinity の両方が集約に含まれる場合、エラーまたは警告が戻されます。それ以外の場合、+NaN または -NaN が検出されれば、結果は +NaN または -NaN になります。+Infinity または -Infinity が検出されれば、結果は +Infinity または -Infinity になります。

代替構文**:** COVAR_POP は COVARIANCE のシノニムとして指定できます。

例

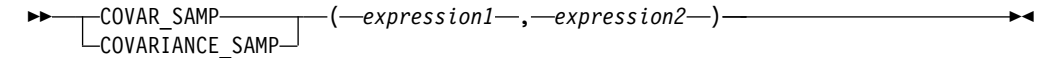
- EMPLOYEE 表を使用して、ホスト変数 COVARNCE を、部門 'A00' の従業員の給与と賞与の間に見られる共分散に設定します。

```
SELECT COVARIANCE(SALARY, BONUS)
  INTO :COVARNCE
  FROM EMPLOYEE
  WHERE WORKDEPT = 'A00';
```

サンプル表を使用した場合、COVARNCE は 1743000.0000 に設定されます。

COVAR_SAMP または COVARIANCE_SAMP

COVARIANCE_SAMP 関数は、数値ペア集合の不偏標本共分散 (n-1) を戻します。



expression1
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数が文字ストリングまたはグラフィック・ストリングの場合、関数を評価する前に DECFLOAT(34) にキャストされます。

expression2
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数が文字ストリングまたはグラフィック・ストリングの場合、関数を評価する前に DECFLOAT(34) にキャストされます。

いずれかの引数が 10 進浮動小数点数の場合、関数の結果は DECFLOAT(34) となります。それ以外の場合は、関数の結果は倍精度浮動小数点数です。結果が、NULL になることもあります。

この関数は、引数の値から導出されたペアの集合 (*expression1, expression2*) から、*expression1* または *expression2* のどちらかが NULL であるすべてのペアを除外したものに対して適用されます。

この関数を空集合または 1 行のみの集合に対して適用すると、結果は NULL 値になります。それ以外の場合、結果はその集合内にある値ペアの標本共分散になります。結果は、次のようにして割り出されます。

1. avgexp1 を AVG(*expression1*) の結果に、 avgexp2 を AVG(*expression2*) の結果にします。
2. COVARIANCE_SAMP(*expression1, expression2*) の結果は以下になります。

```
SUM( (expression1 - avgexp1) * (expression2 - avgexp2) ) /
     ( COUNT(expression1) – 1 )
```

値を集計する順序は定義されていませんが、すべての中間結果は結果のデータ・タイプの範囲内になければなりません。

注

特殊値 **DECFLOAT** を含む結果**:** 引数のデータ・タイプが 10 進浮動小数点であり、特殊値 sNaN または -sNaN、もしくは +Infinity と -Infinity の両方が集約に含まれる場合、エラーまたは警告が戻されます。それ以外の場合、+NaN または -NaN が検出されれば、結果は +NaN または -NaN になります。+Infinity または -Infinity が検出されれば、結果は +Infinity または -Infinity になります。

代替構文**:** SQL 規格に準拠して COVAR_SAMP を使用する必要があります。

例

- ホスト変数 COVARNCE_S を、EMPLOYEE 表の部門 'A00' の従業員の給与と賞与の間に見られる標本共分散に設定します。ホスト変数 COVARNCE_S のデータ・タイプは、倍精度の浮動小数点数です。

```
SELECT COVARIANCE_SAMP(SALARY, BONUS)
  INTO :COVARNCE_S
  FROM EMPLOYEE
  WHERE WORKDEPT = 'A00'
```

  サンプル表を使用した場合、COVARNCE_S は 2178750.0000 に設定されます。

GROUPING

GROUPING 集約関数は、グループ化集合およびスーパー・グループに関連して使用され、GROUP BY 応答セットで戻された行が、*expression* によって表される列を除外し、グループ化集合によって生成された行であるか否かを示す値を戻します。

expression
引数値はどの組み込みデータ・タイプでも構いませんが、GROUP BY 文節の項目でなければなりません。

結果のデータ・タイプは、短精度整数です。結果は以下のいずれかの値に設定されます。

1 戻された行の *expression* の値は NULL 値であり、しかもその行はスーパー・グループによって生成されました。生成されたその行は、GROUP BY 式の小計の値を求めるのに使用することができます。

0 値は上記以外です。

例

以下の照会は、

```
SELECT SALES_DATE, SALES_PERSON,
      SUM(SALES) AS UNITS_SOLD,
      GROUPING(SALES_DATE) AS DATE_GROUP,
      GROUPING(SALES_PERSON) AS SALES_GROUP
FROM SALES
GROUP BY CUBE( SALES_DATE, SALES_PERSON)
ORDER BY SALES_DATE, SALES_PERSON
```

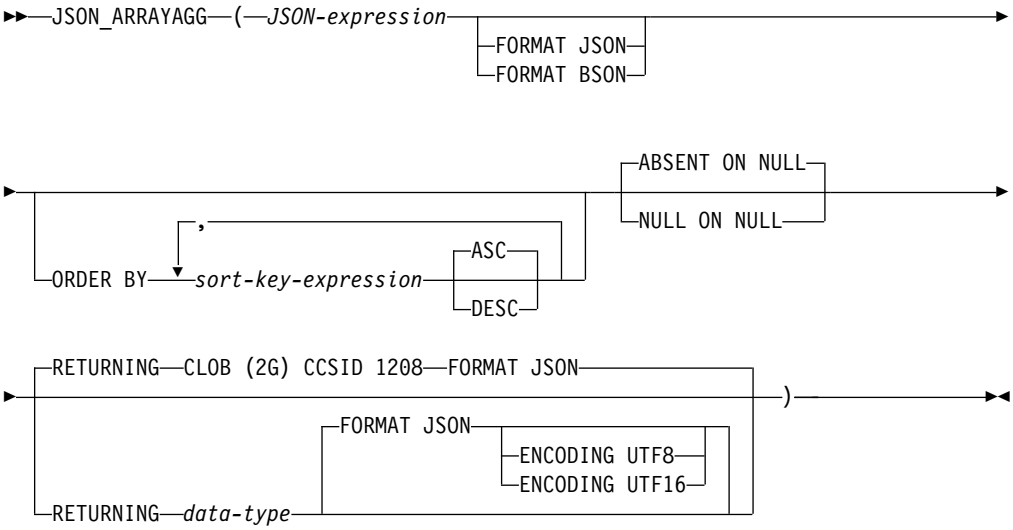
結果は、以下のようになります。

```
SALES_DATE SALES_PERSON UNITS_SOLD DATE_GROUP SALES_GROUP
---------- ------------ ---------- ---------- -----------
12/31/1995 GOUNOT                1          0           0
12/31/1995 LEE                   6          0           0
12/31/1995 LUCCHESSI             1          0           0
12/31/1995 -                     8          0           1
03/29/1996 GOUNOT               11          0           0
03/29/1996 LEE                  12          0           0
03/29/1996 LUCCHESSI             4          0           0
03/29/1996 -                    27          0           1
03/30/1996 GOUNOT               21          0           0
03/30/1996 LEE                  21          0           0
03/30/1996 LUCCHESSI             4          0           0
03/30/1996 -                    46          0           1
03/31/1996 GOUNOT                3          0           0
03/31/1996 LEE                  27          0           0
03/31/1996 LUCCHESSI             1          0           0
03/31/1996 -                    31          0           1
04/01/1996 GOUNOT               14          0           0
04/01/1996 LEE                  25          0           0
04/01/1996 LUCCHESSI             4          0           0
04/01/1996 -                    43          0           1
-          GOUNOT               50          1           0
-          LEE                  91          1           0
-          LUCCHESSI            14          1           0
-          -                   155          1           1
```

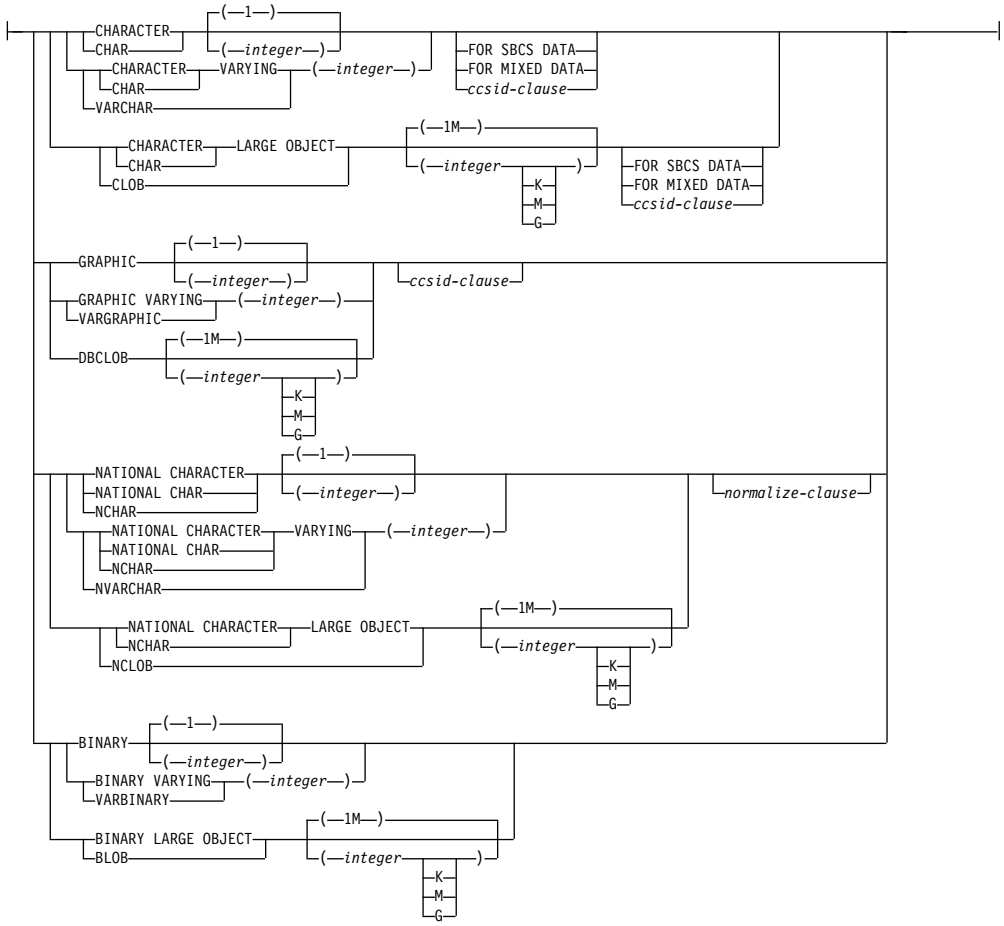
アプリケーションは、DATE_GROUP の値が 0 であり、SALES_GROUP の値が 1 であるという事実に基づいて、SALES_DATE の小計行を認識できます。SALES_PERSON の小計行は、DATE_GROUP の値が 1 であり、SALES_GROUP の値が 0 であるという事実に基づいて認識できます。合計行は、DATE_GROUP と SALES_GROUP の両方の値が 1 であるという事実に基づいて認識できます。

JSON_ARRAYAGG

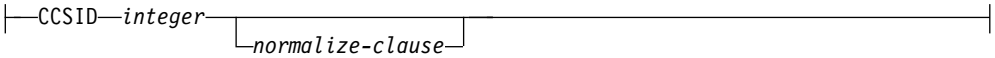
JSON_ARRAYAGG 関数は、JSON 値または SQL 値の集合の各値に対応した配列エレメントを含む JSON 配列を戻します。



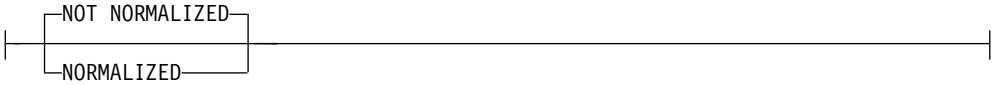
data-type:



ccsid-clause:



normalize-clause:



JSON-expression
JSON 配列の値を生成するために使用する式。この式の結果タイプには、XML、ROWID、および DATALINK を除く、任意の組み込みデータ・タイプが可能です。 CHAR または VARCHAR ビット・データであってはなりません。これらのいずれかのデータ・タイプをソースとするユーザー定義タイプにすることはできません。

FORMAT JSON または **FORMAT BSON**
JSON-expression が既に形式設定されたデータであるかどうかを指定します。

FORMAT JSON
JSON-expression は JSON データとして形式設定されています。*JSON-expression* が文字またはグラフィック・ストリング・データ・タイプであれば、JSON データとして扱われます。バイナリー・ストリング・データ・タイプの *JSON-expression* は、UTF-8 または UTF-16 データとして解釈されます。

FORMAT BSON
JSON-expression は JSON データの BSON 表現として形式設定されています。これは、バイナリー・ストリング・データ・タイプでなければなりません。

FORMAT JSON も FORMAT BSON も指定されない場合:

- *JSON-expression* が、組み込み関数 JSON_ARRAY、JSON_OBJECT、JSON_QUERY、 JSON_ARRAYAGG、または JSON_OBJECTAGG のいずれかである場合、関数の RETURNING 節の明示的または暗黙的な FORMAT 値が *JSON-expression* の形式を決定します。
- バイナリー・ストリング・タイプの *JSON-expression* は、FORMAT BSON として解釈されます。
- それ以外の場合、*JSON-expression* は不定形式データと見なされます。生成値が数値以外の場合、結果ストリングは、引用符で囲んだストリングで構成され、特殊文字はエスケープされます。有効な JSON 数値ではない数値 (INFINITY や NAN など) は、エラーになります。

ORDER BY
集約で処理される、同じグループ化セットからの行の順序を指定します。ORDER BY 文節を指定しない場合、または ORDER BY 文節ではソート・キー値の順序を区別できない場合、同じグループ内の行は任意に順序付けられます。

sort-key-expression
列名または式のどちらかのソート・キー値を指定します。列または式のデータ・タイプは、DATALINK 値または XML 値であってはなりません。

ASC
sort-key-expression を昇順で処理します。これはデフォルトです。

DESC
sort-key-expression を降順で処理します。

順序付けはソート・キーの値に基づいて行われ、これは *JSON-expression* の中で使用される場合もされない場合もあります。

sort-key-expression の長さ属性の合計は 3.5 ギガバイトを超えてはなりません。

JSON_ARRAYAGG 関数を含むステートメントの実行時に *HEX 以外の照合順序が有効で、しかも *sort-key-expression* が SBCS データ、混合データ、または Unicode データの場合、結果は重み付けされた値の比較によって求められます。この重み付けされた値は、ソート・キー式 に照合順序を適用して得られます。

ABSENT ON NULL または NULL ON NULL
JSON-expression によって生成された配列エレメントが NULL 値である場合に何を戻すかを指定します。

ABSENT ON NULL
NULL 配列エレメントは JSON 配列に含められません。これはデフォルトです。

NULL ON NULL
NULL 配列エレメントが JSON 配列に含められます。

RETURNING *data-type*
結果の形式を指定します。

data-type
結果のデータ・タイプです。CHAR 結果および VARCHAR 結果の場合、CCSID を 65535 にすることはできません。デフォルトは CLOB(2G) CCSID 1208 です。

CCSID が指定され、*data-type* が GRAPHIC、VARGRAPHIC、または DBCLOB の場合、CCSID は Unicode CCSID であることが必要です。

CCSID 属性が指定されないと、CCSID は 218 ページの『CAST の指定』に記されているように決定されます。

FORMAT JSON
JSON データは JSON ストリングとして戻されます。

ENCODING UTF8 または ENCODING UTF16
data-type がバイナリー・ストリング・タイプの場合に使用するエンコード方式。この節は、バイナリー・ストリング・タイプの場合にのみ使用できます。バイナリー・ストリングのデフォルトは UTF8 です。

結果が、NULL になることもあります。値の集合が空である場合は、結果は NULL 値です。

例

- すべての部門番号を含む JSON 配列を戻します。

```
SELECT JSON_ARRAYAGG(deptno) AS deptlist FROM dept;
```

結果は、以下の JSON 配列です。

```
["A00","B01","C01","D01","D11","D21","E01","E11","E21","F22","G22","H22","I22","J22"]
```

- 各部門ごとに、その部門に割り当てられた従業員のリストを含む JSON 配列を戻します。

```
SELECT workdept, JSON_ARRAYAGG(lastname ORDER BY lastname) AS dept_employees
FROM emp
WHERE workdept LIKE 'D%'
GROUP BY workdept;
```

結果は、以下の 2 行です。

```
DEPTNO      PROJLIST
 D11        ["ADAMSON","BROWN","JOHN","JONES","LUTZ","PIANKA","SCOUTTEN",
             "STERN","WALKER","YAMAMOTO","YOSHIMURA"]
 D21        ["JEFFERSON","JOHNSON","MARINO","MONTEVERDE","PEREZ","PULASKI","SMITH"]
```

- 部門と各部門の従業員のリストが入った JSON オブジェクトを戻します。

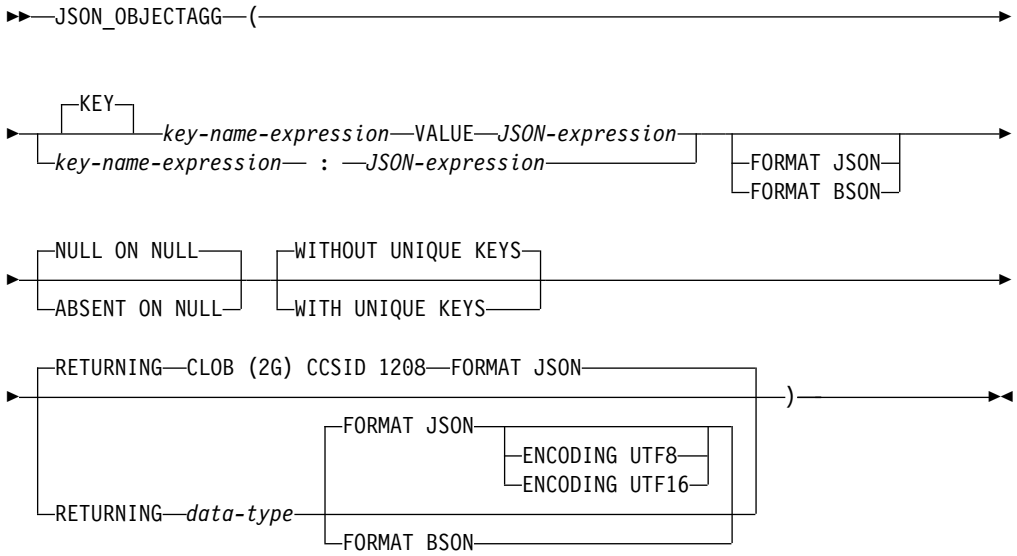
```
SELECT JSON_OBJECT('department number' VALUE deptno,
                   'department name' VALUE deptname,
                   'employee list' VALUE
                        JSON_ARRAYAGG(
                              JSON_OBJECT('last name' VALUE lastname,
                                          'employee id' VALUE empno)
                              ORDER BY lastname))
 FROM dept LEFT OUTER JOIN emp ON deptno = workdept
 WHERE deptno LIKE 'D%'
 GROUP BY deptno, deptname;
```

結果は、以下の 3 行です。

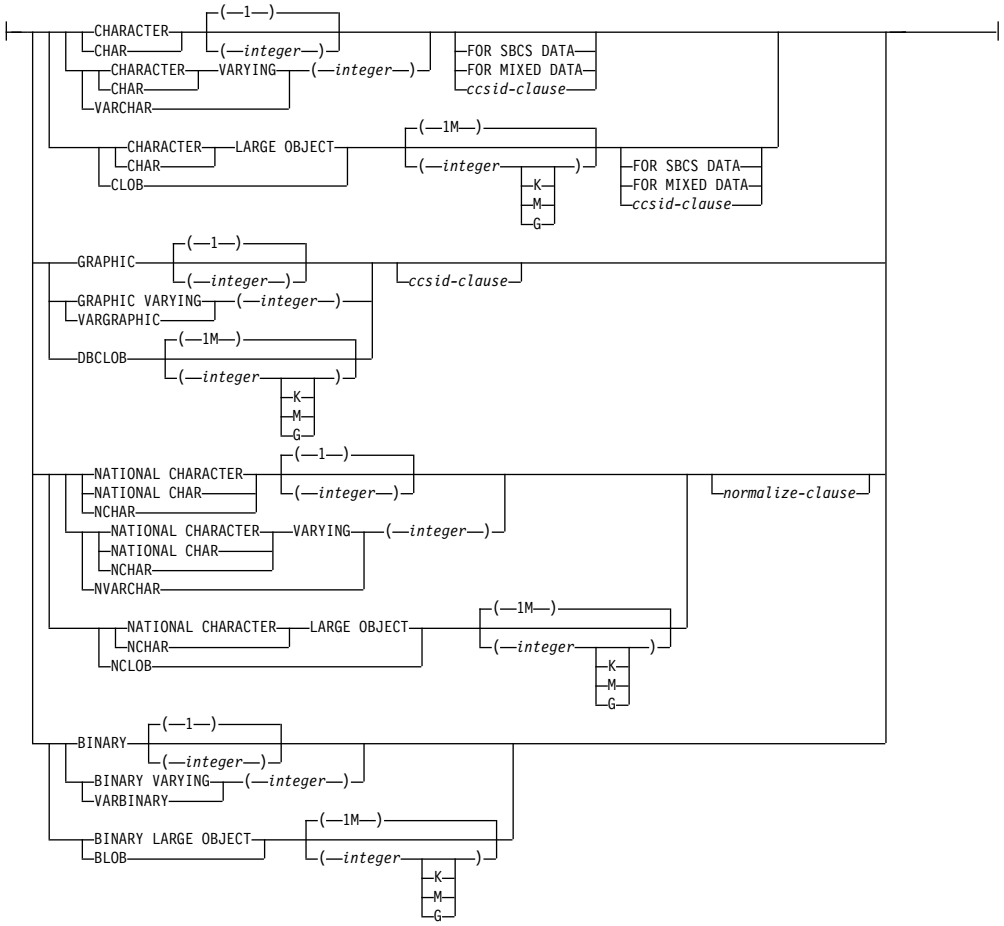
```
{"department number":"D11","department name":"MANUFACTURING SYSTEMS",
    "employee list":[{"last name":"ADAMSON","employee id":"000150"},
                     {"last name":"BROWN","employee id":"000200"},
                     {"last name":"JOHN","employee id":"200220"},
                     {"last name":"JONES","employee id":"000210"},
                     {"last name":"LUTZ","employee id":"000220"},
                     {"last name":"PIANKA","employee id":"000160"},
                     {"last name":"SCOUTTEN","employee id":"000180"},
                     {"last name":"STERN","employee id":"000060"},
                     {"last name":"WALKER","employee id":"000190"},
                     {"last name":"YAMAMOTO","employee id":"200170"},
                     {"last name":"YOSHIMURA","employee id":"000170"}]}

{"department number":"D21","department name":"ADMINISTRATION SYSTEMS",
    "employee list":[{"last name":"JEFFERSON","employee id":"000150"},
                      "last name":"JOHNSON","employee id":"000150"},
                      "last name":"MARINO","employee id":"000150"},
                      "last name":"MONTEVERDE","employee id":"000150"},
                      "last name":"PEREZ","employee id":"000150"},
                      "last name":"PULASKI","employee id":"000150"},
                      "last name":"SMITH","employee id":"000150"}]}

{"department number":"D01","department name":"DEVELOPMENT CENTER"}
```

JSON_OBJECTAGG

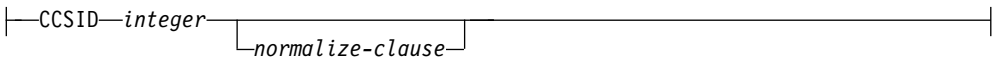
JSON_OBJECTAGG 関数は、SQL 値の集合の個々の各キーおよび値に対応した *key:value* のペアを含む JSON オブジェクトを戻します。



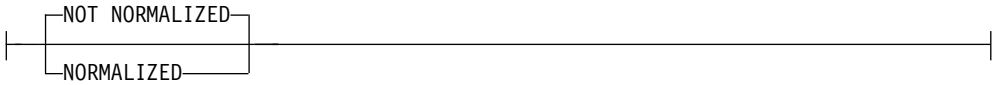
data-type:



ccsid-clause:



normalize-clause:



key-name-expression
JSON キーの名前。この名前は、NULL であってはなりません。 *key:value* のペアの定義にコロン形式を使用する場合、*key-name-expression* は文字ストリング・リテラルでなければなりません。それ以外の場合、*key-name-expression* の結果は、組み込み文字またはグラフィック・ストリング・データ・タイプでなければなりません。 CHAR または VARCHAR ビット・データであってはなりません。

JSON-expression
key-name-expression に関連付けられた JSON 値を生成するために使用する式。

この式の結果タイプには、XML、ROWID、および DATALINK を除く、任意の組み込みデータ・タイプが可能です。 CHAR または VARCHAR ビット・データであってはなりません。これらのいずれかのデータ・タイプをソースとするユーザー定義タイプにすることはできません。

FORMAT JSON または FORMAT BSON
JSON-expression が既に形式設定されたデータであるかどうかを指定します。

FORMAT JSON
JSON-expression は JSON データとして形式設定されています。 *JSON-expression* が文字またはグラフィック・ストリング・データ・タイプであれば、JSON データとして扱われます。バイナリー・ストリング・データ・タイプの *JSON-expression* は、UTF-8 または UTF-16 データとして解釈されます。

FORMAT BSON
JSON-expression は JSON データの BSON 表現として形式設定されています。これは、バイナリー・ストリング・データ・タイプでなければなりません。

FORMAT JSON も FORMAT BSON も指定されない場合:

- *JSON-expression* が、組み込み関数 JSON_ARRAY、JSON_OBJECT、JSON_QUERY、 JSON_ARRAYAGG、または JSON_OBJECTAGG のいずれかである場合、関数の RETURNING 節の明示的または暗黙的な FORMAT 値が *JSON-expression* の形式を決定します。
- バイナリー・ストリング・タイプの *JSON-expression* は、FORMAT BSON として解釈されます。
- それ以外の場合、*JSON-expression* は不定形式データと見なされます。生成値が数値以外の場合、結果ストリングは、引用符で囲んだストリングで構成され、特殊文字はエスケープされます。有効な JSON 数値ではない数値 (INFINITY や NAN など) は、エラーになります。

NULL ON NULL または ABSENT ON NULL
JSON-expression が NULL 値である場合に何を戻すかを指定します。

NULL ON NULL
NULL 値が戻されます。これはデフォルトです。

ABSENT ON NULL
キー:値 のペアを JSON オブジェクトから除外します。

WITHOUT UNIQUE KEYS または WITH UNIQUE KEYS
結果の JSON オブジェクトのキー値をユニークにする必要があるかどうかを指定します。

WITHOUT UNIQUE KEYS
結果の JSON オブジェクトに重複キーがあるかどうかは検査されません。これはデフォルトです。

WITH UNIQUE KEYS
結果の JSON オブジェクトは、ユニーク・キー値を持つ必要があります。重複キーが生成されると、エラーが発行されます。

ユニーク・キーを持つ JSON オブジェクトを生成することが、ベスト・プラクティスと見なされています。*key-name-expression* がユニークなキー名を生成する場合は、WITH UNIQUE KEYS を省いてパフォーマンスを向上させることができます。

RETURNING *data-type*
結果の形式を指定します。

data-type
結果のデータ・タイプです。CHAR 結果および VARCHAR 結果の場合、CCSID を 65535 にすることはできません。デフォルトは CLOB(2G) CCSID 1208 です。

CCSID が指定され、*data-type* が GRAPHIC、VARGRAPHIC、または DBCLOB の場合、CCSID は Unicode CCSID であることが必要です。

CCSID 属性が指定されないと、CCSID は 218 ページの『CAST の指定』に記されているように決定されます。

FORMAT JSON
JSON データは JSON ストリングとして戻されます。

ENCODING UTF8 または **ENCODING UTF16**
data-type がバイナリー・ストリング・タイプの場合に使用するエンコード方式。この節は、バイナリー・ストリング・タイプの場合にのみ使用できます。バイナリー・ストリングのデフォルトは UTF8 です。

FORMAT BSON
JSON データは、BSON 形式で戻されます。FORMAT BSON を指定した場合、*data-type* は VARBINARY または BLOB ストリング・タイプでなければなりません。

結果が、NULL になることもあります。値の集合が空である場合は、結果は NULL 値です。

例

- 各部門の管理者番号を含む JSON オブジェクトを戻します。管理者が割り当てられていない場合、その部門は結果から除外されます。

```
SELECT JSON_OBJECTAGG(deptno VALUE mgrno ABSENT ON NULL) FROM dept;
```

結果は、以下の JSON ストリングです。JSON オブジェクト内の項目の順序は未定義です。

```
{"A00":"000010","B01":"000020","C01":"000030","D11":"000060","D21":"000070",
 "E01":"000050","E11":"000090","E21":"000100"}
```

- 各部門ごとに、その部門に割り当てられたプロジェクトのリストを含む JSON オブジェクトを戻します。

```
SELECT deptno, JSON_OBJECTAGG(projno VALUE projname) AS projlist FROM proj
WHERE deptno LIKE 'D%'
GROUP BY deptno;
```

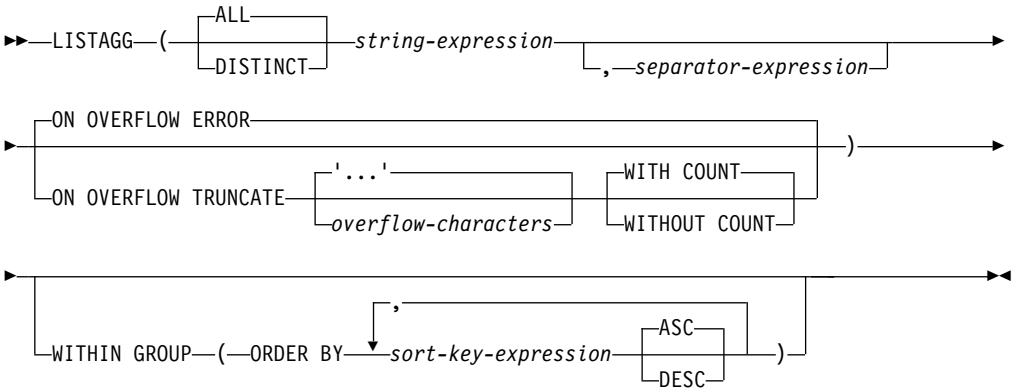
結果は、以下の 3 行です。JSON オブジェクト内の項目の順序は未定義です。

```
DEPTNO      PROJLIST
 D01        {"AD3100":"ADMIN SERVICES","MA2100":"WELD LINE AUTOMATION"}
 D11        {"MA2110":"W L PROGRAMMING","MA2111":"W L PROGRAM DESIGN",
```

```
                 "MA2112":"W L ROBOT DESIGN","MA2113":"W L PROD CONT PROGS"}
D21              {"AD3110":"GENERAL ADMIN SYSTEMS","AD3111":"PAYROLL PROGRAMMING",
                 "AD3112":"PERSONNEL PROGRAMMING","AD3113":"ACCOUNT PROGRAMMING"}
```

LISTAGG

LISTAGG 関数は、ストリングを連結することにより、一連のストリング・エレメントを 1 つのストリングに集約します。オプションで、隣接する入力ストリング同士の間に挿入するセパレーター・ストリングを指定することができます。



LISTAGG 関数は、特定のグループのストリング値の集合を 1 つのストリングに集約します。これは、WITHIN GROUP 節で指定された順序に基づいて *string-expression* の値を追加することによって行います。

この関数は、最初の引数から NULL 値を取り除いて得られた値の集合に対して適用されます。 DISTINCT を指定すると、重複する *string-expression* 値は除去されます。 NULL 値ではない区切り文字引数を指定すると、NULL 以外の *string-expression* のそれぞれの値ペアの間に、その区切り文字の値が挿入されます。

string-expression
集約するストリング値を指定する式。この式は、組み込みデータ・タイプであるストリング、数値、日時のいずれかの値を戻す必要があります。数値または日時のデータ・タイプの値は、関数の評価の前に VARCHAR に暗黙的にキャストされます。

separator-expression
NULL でない *string-expression* 値の間で使用するストリングを定義する式。この式は、組み込みデータ・タイプであるストリング、数値、日時のいずれかの値を戻す必要があります。数値または日時のデータ・タイプの値は、関数の評価の前に VARCHAR に暗黙的にキャストされます。 *separator-expression* にスカラー全選択、列参照、および 非 deterministic 関数および外部アクションを含む関数への参照を含めてはなりません。

separator-expression が指定されない場合や、*separator-expression* が NULL 値の場合、 *string-expression* 値の間の区切りはありません。

ON OVERFLOW ERROR または **ON OVERFLOW TRUNCATE**
集約された結果ストリングの実際の長さが結果の長さを超える場合の動作を指定します。デフォルトは ON OVERFLOW ERROR です。

ON OVERFLOW ERROR
結果ストリングの実際の長さが結果の長さを超える場合、エラーが戻されるように指示します。

ON OVERFLOW TRUNCATE
結果ストリングの実際の長さが結果の長さを超える場合、集約された結果ストリングが切り捨てられるように指示します。切り捨ては、ストリング値の末尾で行われます。結果ストリングは、定義されていれば最後の *separator-expression* を含む、完全な項目を含みます。

string-expression がバイナリー・ストリングまたは非 Unicode グラフィック・ストリングである場合、このオプションは使用できません。

'...' または *overflow-characters*
切り捨てが行われたことを示すために、結果ストリングの末尾に付加される文字ストリング定数を指示します。

'...'
最後の完全な項目のすぐ後に、3 つのピリオド文字が付加されるように指示します。これはデフォルトです。

overflow-characters
最後の完全な項目のすぐ後に付加される文字ストリング定数を指示します。

WITH COUNT または **WITHOUT COUNT**
切り捨てられた値の数を結果ストリングの最後に含めるかどうかを指示します。デフォルトは WITH COUNT です。

WITH COUNT
ストリングから切り捨てられた値の数が、結果ストリングの最後に付加されます。括弧内の数値として形式設定されます。例えば、10 項目が切り捨てられた場合、(10) がストリングの最後に組み込まれます。

WITHOUT COUNT
切り捨てられた項目の数を表すものは何も戻されません。

WITHIN GROUP
集約するときに、グループ化集合の中の指定順序で配列されることを示します。

WITHIN GROUP が指定されていない場合、結果に含まれるストリングの順序付けは一律には決まりません。

ORDER BY
集約で処理される、同じグループ化セットからの行の順序を指定します。ORDER BY 文節を指定しない場合、または ORDER BY 文節ではソート・キー値の順序を区別できない場合、同じグループ内の行は任意に順序付けられます。

sort-key-expression
列名または式のどちらかのソート・キー値を指定します。列または式のデータ・タイプは、DATALINK 値または XML 値であってはなりません。

集約したエレメントの順序付けは、ソート・キーの値に基づいて行われます。

sort-key-expression の長さ属性の合計は 3.5 ギガバイトを超えてはなりません。

ASC
sort-key-expression を昇順で処理します。これはデフォルトです。

DESC
sort-key-expression を降順で処理します。

LISTAGG 関数を含むステートメントの実行時に *HEX 以外の照合順序が有効で、しかも *sort-key-expressions* が SBCS データ、混合データ、または Unicode データの場合、結果は重み付けされた値の比較によって求められます。この重み付けされた値は、ソート・キー式 に照合順序を適用して得られます。

LISTAGG の結果データ・タイプは、*string-expression* のデータ・タイプに基づいて決まります。

表 *48.* 結果のデータ・タイプとデータ長の決定

string-expression のデータ・タイプ	結果のデータ・タイプとデータ長
CHAR(*n*) または VARCHAR(*n*)	VARCHAR(MAX(4000, *n*))
CLOB (*n*)	CLOB(1M)
GRAPHIC(*n*) または VARGRAPHIC(*n*)	VARGRAPHIC(MAX(2000, *n*))
DBCLOB(*n*)	DBCLOB(1M)
BINARY(*n*) または VARBINARY(*n*)	VARBINARY(MAX(4000, *n*))
BLOB(*n*)	BLOB(1M)

派生したサイズを使用して結果のサイズが決まる場合は、結果のデータ・タイプが VARCHAR(4000)、VARBINARY(4000)、または VARGRAPHIC(2000) を超えることがあります。指定可能な最大値は、結果のデータ・タイプの最大値です。以下の例では、戻りデータ・タイプ VARCHAR(10000) が生成されます。

```
LISTAGG(CAST(NAME AS VARCHAR(10000)), ',')
```

集約された結果ストリングの実際の長さが結果の長さを超える場合は、 ON OVERFLOW 節によって動作が決まります。

結果の CCSID は *string-expression* の CCSID です。

LISTAGG 関数を含むステートメントの実行時に *HEX 以外の照合順序が有効で、しかも *sort-key-expression* が SBCS データ、混合データ、または Unicode データの場合、結果は重み付けされた値の比較によって求められます。この重み付けされた値は、*sort-key-expression* に該当の照合順序を適用することにより得られます。

結果が、NULL になることもあります。この関数が空集合に適用されるか、集合に含まれるすべての *string-expression* の値が NULL 値である場合、結果は NULL 値になります。

規則

- LISTAGG を OLAP 指定の一部として使用することはできません。

例

- 部門別にグループ化したコンマ区切りの名前リストをアルファベット順で生成します。

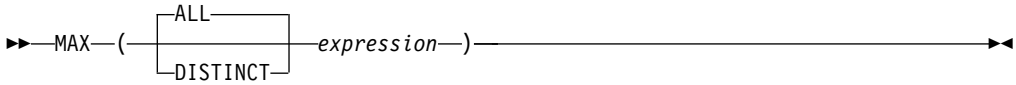
```
SELECT workdept,
       LISTAGG(lastname, ', ') WITHIN GROUP(ORDER BY lastname)
         AS employees
  FROM emp
  GROUP BY workdept
```

以下の結果を生成します。

```
WORKDEPT   EMPLOYEES
A00        HAAS, HEMMINGER, LUCCHESSI, O'CONNELL, ORLANDO
B01        THOMPSON
C01        KWAN, NATZ, NICHOLLS, QUINTANA
D11        ADAMSON, BROWN, JOHN, JONES, LUTZ, PIANKA, SCOUTTEN, STERN, WALKER
                YAMAMOTO, YOSHIMURA
D21        JEFFERSON, JOHNSON, MARINO, MONTEVERDE, PEREZ, PULASKI, SMITH
E01        GEYER
E11        HENDERSON, PARKER, SCHNEIDER, SCHWARTZ, SETRIGHT, SMITH, SPRINGER
E21        ALONZO, GOUNOT, LEE, MEHTA, SPENSER, WONG
```

MAX

MAX 集約関数は、あるグループの中の値の集合の最大値を戻します。



expression
引数値は、データ・リンクまたは XML を除く任意の組み込みデータ・タイプにすることができます。

結果のデータ・タイプおよび長さ属性は、引数の値のデータ・タイプおよび長さ属性と同じになります。引数がストリングの場合、結果は引数と同じ CCSID を持ちます。

MAX 関数を含むステートメントの実行時に *HEX 以外の照合順序が有効で、しかも引数が SBCS データ、混合データ、または Unicode データの場合、結果は、集合の各値の重み付けされた値の比較によって求められます。値の重み付けは、該当の照合順序に基づいています。

この関数は、引数の値から NULL 値を除いた値の集合に対して適用されます。

結果が、NULL になることもあります。値のセットが空の場合、結果は NULL 値になります。それ以外の場合は、結果は集合の中の最大値になります。

DISTINCT を指定しても結果には影響を与えないため、使用しないようにしてください。

注

DECFLOAT 特殊値が存在する場合の結果: 引数のデータ・タイプが 10 進浮動小数点数の場合に、正または負の Infinity、sNaN、NaN が検出されると、10 進浮動小数点数の順序付け規則に基づいて最大数が決定されます。 127 ページの『数値比較』を参照してください。同じ 10 進浮動小数点値に複数の表記が検出される (例えば、2.00 と 2.0) 場合、どの表記が戻されるかは予測不能です。

例

- EMPLOYEE 表を使用して、ホスト変数 MAX_SALARY (DECIMAL(7,2)) を最大月給 (SALARY / 12) の値に設定します。

```
SELECT MAX(SALARY) /12
  INTO :MAX_SALARY
  FROM EMPLOYEE
```

  上記の結果、MAX_SALARY は 4395.83 にセットされます。

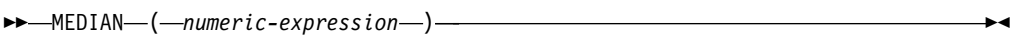
- 表 PROJECT を使用して、ソート順序で最後になるプロジェクト名 (PROJNAME) をホスト変数 LAST_PROJ (CHAR(24)) にセットします。

```
SELECT MAX(PROJNAME)
  INTO :LAST_PROJ
  FROM PROJECT
```

  結果として、LAST_PROJ は 'WELD LINE PLANNING ' に設定されます。

MEDIAN

MEDIAN 関数は、一式の数値の中央値を返します。



numeric-expression
組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプを戻す式。引数が文字ストリングまたはグラフィック・ストリングの場合、関数を評価する前に DECFLOAT(34) にキャストされます。

引数が DECFLOAT(n) である場合、関数の結果は DECFLOAT(34) です。それ以外の場合は、関数の結果は倍精度浮動小数点数です。

この関数は、引数の値から NULL 値を除いた値の集合に対して適用されます。

結果が、NULL になることもあります。*numeric-expression* が NULL である場合、または空の集合に関数が適用された場合、結果は NULL 値になります。

注

OLAP の指定で使用された場合は、*window-partition-clause* のみを指定できます。

例

- EMPLOYEE 表の部門 D11 に所属する従業員の給与の中央値を計算します。

```
SELECT MEDIAN(SALARY)
FROM EMPLOYEE
WHERE WORKDEPT = 'D11';
```

  結果は 24680.00 です。部門 D11 には 11 人の従業員がいます。 11 個の値のグループにおいて、中央の行は 6 行目です。このグループ全体の MEDIAN の結果は、6 行目の値である 24680.00 となります。

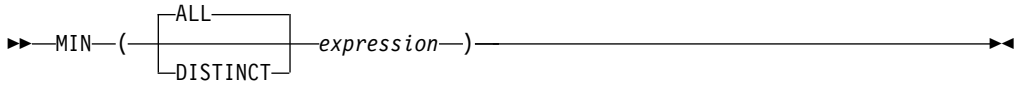
- EMPLOYEE 表の部門 E21 に所属する従業員の給与の中央値を計算します。

```
SELECT MEDIAN(SALARY)
FROM EMPLOYEE
WHERE WORKDEPT = 'E21';
```

  結果は 24605.00 です。部門 E21 には 6 人の従業員がいます。偶数個の行があるため、中央にある 2 つの行の間に値を挿入することで MEDIAN を計算します。中央の 2 つの行は、値が 23840.00 である 3 行目と、値が 25370.00 である 4 行目です。 MEDIAN は、これらの 2 つの値を平均して算出した 24605.00 となります。

MIN

MIN 集約関数は、あるグループの中の値の集合の最小値を戻します。



expression
引数値は、データ・リンクまたは XML を除く任意の組み込みデータ・タイプにすることができます。

結果のデータ・タイプおよび長さ属性は、引数の値のデータ・タイプおよび長さ属性と同じになります。引数がストリングの場合、結果は引数と同じ CCSID を持ちます。結果は NULL になる場合があります。

MIN 関数を含むステートメントの実行時に *HEX 以外の照合順序が有効で、しかも引数が SBCS データ、混合データ、または Unicode データの場合、結果は、集合の各値の重み付けされた値の比較によって求められます。

この関数は、引数の値から NULL 値を除いた値の集合に対して適用されます。

値のセットが空の場合、結果は NULL 値になります。それ以外の場合は、結果はその集合の中の最小値になります。

DISTINCT を指定しても結果には影響を与えないため、使用しないようにしてください。

注

特殊値 **DECFLOAT** を含む結果**:** 引数のデータ・タイプが 10 進浮動小数点であり、正または負の Infinity、sNaN、または NaN が検出される場合、最小値は 10 進浮動小数点の順序付け規則を使用して判別されます。 127 ページの『数値比較』を参照してください。同じ 10 進浮動小数点値に複数の表記が検出される (例えば、2.00 と 2.0) 場合、どの表記が戻されるかは予測不能です。

例

- 表 EMPLOYEE を使用して、部門 (WORKDEPT) 'D11' の社員の手数料 (COMM) の最大値と最小値の差をホスト変数 COMM_SPREAD (DECIMAL(7,2)) にセットします。

```
SELECT MAX(COMM) - MIN(COMM)
  INTO :COMM_SPREAD
  FROM EMPLOYEE
  WHERE WORKDEPT  = 'D11'
```

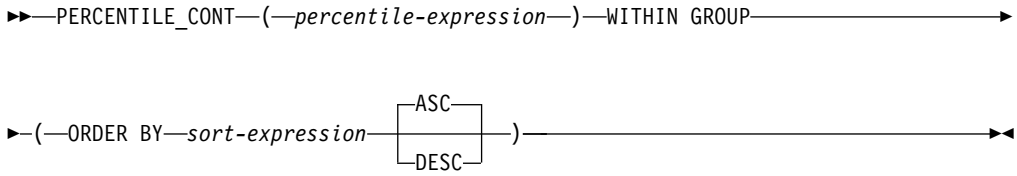
  上記の結果、COMM_SPREAD は 1118 (つまり、2580 - 1462) にセットされます。
- 表 PROJECT を使用して、最も早期に完了が予定されているプロジェクトの予定終了日付 (PRENDATE) を、ホスト変数 FIRST_FINISHED (CHAR(10)) にセットします。

```
SELECT MIN(PRENDATE)
  INTO :FIRST_FINISHED
  FROM PROJECT
```

上記の結果、FIRST_FINISHED は '1982-09-15' にセットされます。

PERCENTILE_CONT

PERCENTILE_CONT 関数は、連続分散モデルを使用して、ソート指定が与えられた指定されたパーセンタイルに該当する値を戻します。



percentile-expression
関数による計算対象のパーセンタイルを指定します。 *percentile-expression* は任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプを戻す必要があります。引数が文字ストリングまたはグラフィック・ストリングの場合、関数を評価する前に DECFLOAT(34) にキャストされます。値の範囲は 0 から 1 までです。*percentile-expression* に *scalar-fullselect*、列参照、およびユーザー定義関数参照を含めてはなりません。

WITHIN GROUP
パーセンタイルはグループ内の識別された行から計算されることを指定します。

sort-expression
パーセンタイルを計算する値の集合と、集合の順序を指定します。 *sort-expression* は、組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプを戻す必要があります。 *sort-expression* が文字ストリングまたはグラフィック・ストリングの場合、関数を評価する前に DECFLOAT(34) にキャストされます。

ASC
sort-expression による値を昇順で使用してパーセンタイルを計算することを指定します。

DESC
sort-expression による値を降順で使用してパーセンタイルを計算することを指定します。

結果は、*sort-expression* で識別された値の集合において、*percentile-expression* で指定されたパーセンタイルに該当する値です。値の集合は、連続分布として扱われます。計算されるパーセンタイルは、入力集合にはなかった可能性がある補間値です。 *sort-expression* のデータ・タイプが DECFLOAT(n) である場合、関数の結果は DECFLOAT(34) です。それ以外の場合、結果のデータ・タイプは、倍精度の浮動小数点数になります。

この関数は、引数の値から NULL 値を除いた値の集合に対して適用されます。

結果は NULL 値の場合もあります。 *percentile-expression* が NULL である場合、または空の集合に関数が適用された場合、結果は NULL 値になります。

注

OLAP の指定で使用された場合は、*window-partition-clause* のみを指定できます。

特殊値 **DECFLOAT** を含む結果**:** 引数のデータ・タイプが 10 進浮動小数点であり、特殊値 sNaN または -sNaN、もしくは +Infinity と -Infinity の両方が集約に含まれる場合、エラーまたは警告が戻されます。それ以外の場合、+NaN または -NaN が検出されれば、結果は +NaN または -NaN になります。+Infinity または -Infinity が検出されれば、結果は +Infinity または -Infinity になります。

列マスク**:** PERCENTILE_CONT 関数の *sort-expression* で参照されている列が、列マスクを持つように定義されている場合、その列マスクは適用されません。

PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY *sort-expression*) を使用した場合の結果は、 MEDIAN(*sort-expression*) を指定した場合と同じになります。

例

- 例 **1**: 部門 D11 に所属する従業員の給与の中央値を計算します。

```
SELECT PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY SALARY) FROM EMPLOYEE
WHERE WORKDEPT = 'D11'
```

  結果は 24680.00 です。部門 D11 には、11 人の従業員がいます。 11 個の値のグループにおいて、中央の行は 6 行目です。奇数個の行があるため、パーセンタイル 0.5 の PERCENTILE_CONT は、6 行目の値の 24680.00 を戻します。

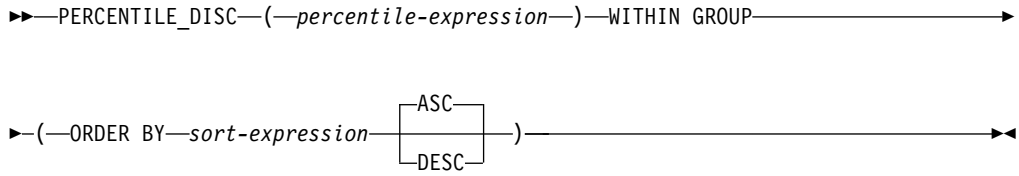
- 例 **2**: 部門 E21 に所属する従業員の歩合給の中央値を計算します。

```
SELECT PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY COMM) FROM EMPLOYEE
WHERE WORKDEPT = 'E21'
```

  結果は 1968.5 です。偶数個の行があるため、中央にある 2 つの行の間に値を挿入することで PERCENTILE_CONT を計算します。中央にある 2 つの行とは、値が 1907.00 の 3 行目と、値が 2030.00 の 4 行目です。
  PERCENTILE_CONT は、これらの 2 つの値を平均して算出した 1968.5 となります。

PERCENTILE_DISC

PERCENTILE_DISC 関数は、離散分布モデルを使用して、ソート指定が与えられた指定されたパーセンタイルに該当する値を戻します。



percentile-expression
関数による計算対象のパーセンタイルを指定します。 *percentile-expression* は任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプを戻す必要があります。引数が文字ストリングまたはグラフィック・ストリングの場合、関数を評価する前に DECFLOAT(34) にキャストされます。値の範囲は 0 から 1 までです。*percentile-expression* に *scalar-fullselect*、列参照、およびユーザー定義関数参照を含めてはなりません。

WITHIN GROUP
パーセンタイルはグループ内の識別された行から計算されることを指定します。

sort-expression
パーセンタイルを計算する値の集合と、集合の順序を指定します。
sort-expression は、組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプを戻す必要があります。

ASC
sort-expression による値を昇順で使用してパーセンタイルを計算することを指定します。

DESC
sort-expression による値を降順で使用してパーセンタイルを計算することを指定します。

結果は、*sort-expression* で識別された値の集合において、*percentile-expression* で指定されたパーセンタイルに該当する値です。入力集合内の各値は、離散値として扱われます。計算されるパーセンタイルは常に、入力集合で出現した値です。結果のデータ・タイプは、*sort-expression* の結果のデータ・タイプと同じです。

PERCENTILE_DISC 関数を含むステートメントの実行時に *HEX 以外の照合順序が有効で、しかも *sort-expression* が SBCS データ、混合データ、または Unicode データの場合、結果は重み付けされた値の比較によって決まります。この重み付けされた値は、*sort-expression* に該当の照合順序を適用することにより得られます。

この関数は、引数の値から NULL 値を除いた値の集合に対して適用されます。

結果が、NULL になることもあります。*percentile-expression* が NULL である場合、または空の集合に関数が適用された場合、結果は NULL 値になります。

注

OLAP の指定で使用された場合は、*window-partition-clause* のみを指定できます。

特殊値 **DECFLOAT** を含む結果**:** 引数のデータ・タイプが 10 進浮動小数点であり、特殊値 sNaN または -sNaN、もしくは +Infinity と -Infinity の両方が集約に含まれる場合、エラーまたは警告が戻されます。それ以外の場合、+NaN または -NaN が検出されれば、結果は +NaN または -NaN になります。+Infinity または -Infinity が検出されれば、結果は +Infinity または -Infinity になります。

列マスク**:** PERCENTILE_DISC 関数の *sort-expression* で参照されている列が、列マスクを持つように定義されている場合、その列マスクは適用されません。

例

- 例 **1**: 部門 D11 に所属する従業員の離散値として、給与の中央値を計算します。

```
SELECT PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY SALARY)
FROM EMPLOYEE
WHERE WORKDEPT = 'D11'
```

  結果は 24680.00 です。部門 D11 には、11 人の従業員がいます。 11 個の値のグループにおいて、中央の行は 6 行目です。奇数個の行があるため、パーセンタイル 0.5 の PERCENTILE_DISC は、6 行目の値の 24680.00 を戻します。

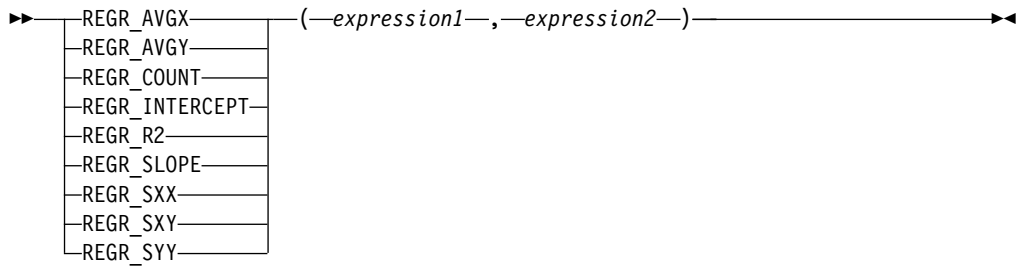
- 例 **2**: 部門 E21 に所属する従業員の離散値として、歩合給の中央値を計算します。

```
SELECT PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY COMM)
FROM EMPLOYEE
WHERE WORKDEPT = 'E21'
```

  結果は 1907.00 です。偶数個の行があるため、中央にある 2 つの行の中で最初の行 (これは値が 1907.00 である 3 行目です) を戻すことで PERCENTILE_DISC が計算されます。

回帰関数

回帰関数は、通常の最小二乗法による回帰直線 (形式 y = a * x + b) を数値ペアの集合に当てはめることをサポートします。各ペアの最初のエレメント *(expression1*) は、従属変数の値 (つまり、「y 値」) と解釈されます。各ペアの 2 番目のエレメント (*expression2*) は、独立変数の値 (つまり、「x 値」) と解釈されます。



expression1
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数が文字ストリングまたはグラフィック・ストリングの場合、関数を評価する前に DECFLOAT(34) にキャストされます。

expression2
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数が文字ストリングまたはグラフィック・ストリングの場合、関数を評価する前に DECFLOAT(34) にキャストされます。

REGR_COUNT 関数は、回帰直線を求めるために使用する NULL ではない数字のペアの数を戻します。

REGR_INTERCEPT 関数は、回帰直線の y 切片 (式 y = a * x + b の「b」) を戻します。

REGR_R2 関数は、回帰に関する判別の係数 ("R 二乗" または "適合度" ともいう) を戻します。

REGR_SLOPE 関数は、直線の傾き (式の y = a * x + b の「a」) を戻します。

REGR_AVGX、REGR_AVGY、REGR_SXX、REGR_SXY、および REGR_SYY 関数は数量を戻します。そのデータを使用すれば、回帰モデルの質と統計としての有効性を評価するために必要な各種の診断統計を計算できます。

REGR_COUNT の結果のデータ・タイプは bigint です。残りの関数の場合、引数のいずれかが DECFLOAT(n) であれば、結果のデータ・タイプは DECFLOAT(34) になります。そうでない場合、結果のデータ・タイプは倍精度浮動小数点になります。いずれかの引数が特殊 10 進浮動小数点値である場合、10 進浮動小数点数の一般算術演算の規則が適用されます。 詳しくは、 200 ページの『DECFLOAT の一般的な算術演算規則』を参照してください。

結果が、NULL になることもあります。値が NULL 値でない場合、REGR_R2 の結果は 0 から 1 になります。 REGR_SXX と REGR_SYY の結果はどちらも非負になります。

各関数は、引数の値から導出された (*expression1, expression2*) ペアの集合から、*expression1* または *expression2* のどちらかが NULL であるすべてのペアを除外したものに対して適用されます。

集合が空の場合は、REGR_COUNT はゼロを戻し、その他の関数は NULL 値を戻します。

集合が空でない場合、関数は、以下に定義された結果を戻します。

REGR_COUNT
集合内の非 NULL ペアの数。

REGR_SLOPE
VARIANCE(*expression2*) が正の場合:

```
REGR_SLOPE(expression1,expression2) =
    COVARIANCE(expression1,expression2)/VARIANCE(expression2)
```

VARIANCE(*expression2*) がゼロの場合、NULL 値を戻します。

REGR_INTERCEPT
VARIANCE(*expression2*) が正の場合:

```
REGR_INTERCEPT(expression1, expression2) =
    AVG(expression1) - REGR_SLOPE(expression1, expression2) * AVG(expression2)
```

VARIANCE(*expression2*) がゼロの場合、NULL 値を戻します。

REGR_R2
VARIANCE(*expression2*) が正の場合:

- VARIANCE(*expression1*) が正の場合:

  ```
  REGR_R2(expression1, expression2) =
      POWER(CORRELATION(expression1, expression2), 2)
  ```

- VARIANCE(*expression1*) がゼロの場合:

  ```
  REGR_R2(expression1, expression2) = 1
  ```

VARIANCE(*expression2*) がゼロの場合、NULL 値を戻します。

REGR_AVGX

```
REGR_AVGX(expression1, expression2) = AVG(expression2)
```

REGR_AVGY

```
REGR_AVGY(expression1, expression2) = AVG(expression1)
```

REGR_SXX

```
REGR_SXX(expression1, expression2) =
      REGR_COUNT(expression1, expression2) * VARIANCE(expression2)
```

REGR_SYY

```
REGR_SYY(expression1, expression2) =
      REGR_COUNT(expression1, expression2) * VARIANCE(expression1)
```

REGR_SXY

```
REGR_SXY(expression1, expression2) =
      REGR_COUNT(expression1, expression2) * COVARIANCE(expression1, expression2)
```

値を集計する順序は定義されていませんが、すべての中間結果は結果のデータ・タイプの範囲内になければなりません。

線形回帰分析を伴う通常の診断統計は上記の関数によって計算できます。例えば、次のようになります。

```
Adjusted R2
    1 - ( (1 - REGR_R2) * ((REGR_COUNT - 1) / (REGR_COUNT - 2)) )
Standard error
    SQRT( (REGR_SYY-(POWER(REGR_SXY,2)/REGR_SXX))/(REGR_COUNT-2) )
Total sum of squares
    REGR_SYY
Regression sum of squares
    POWER(REGR_SXY,2) / REGR_SXX
Residual sum of squares
    (二乗和の合計) - (回帰二乗和)
t statistic for slope
    REGR_SLOPE * SQRT(REGR_SXX) / (標準誤差)
t statistic for y-intercept
   REGR_INTERCEPT/((Standard error) *
      SQRT((1/REGR_COUNT)+(POWER(REGR_AVGX,2)/REGR_SXX)))
```

注記

代替構文: REGR_ICPT は REGR_INTERCEPT のシノニムとして指定できます。

例

- EMPLOYEE 表を使用して、部門 (WORKDEPT) 'A00' の従業員の賞与を表す回帰直線 (通常の最小二乗法による) を、従業員の給与の線形関数として計算します。ホスト変数 SLOPE、ICPT、RSQR をそれぞれ、回帰直線の判別の傾き、切片、係数に設定します。また、ホスト変数 AVGSAL を部門 'A00' の従業員の平均給与、ホスト変数 AVGBONUS を部門 'A00' の従業員の平均賞与に設定します。さらに、ホスト変数 CNT (整数) を、部門 'A00' の従業員のうち、給与データと賞与データが両方とも存在している従業員の数に設定します。その他の回帰統計はホスト変数 SXX、SYY、および SXY に格納します。

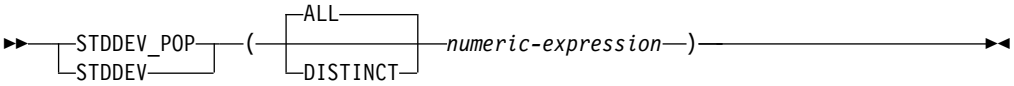
  ```
  SELECT REGR_SLOPE(BONUS,SALARY), REGR_INTERCEPT(BONUS,SALARY),
         REGR_R2(BONUS,SALARY), REGR_COUNT(BONUS,SALARY),
         REGR_AVGX(BONUS,SALARY), REGR_AVGY(BONUS,SALARY),
         REGR_SXX(BONUS,SALARY), REGR_SYY(BONUS,SALARY),
         REGR_SXY(BONUS,SALARY)
    INTO :SLOPE, :ICPT,
         :RSQR, :CNT,
         :AVGSAL, :AVGBONUS,
         :SXX, :SYY,
         :SXY
    FROM EMPLOYEE
    WHERE WORKDEPT = 'A00'
  ```

  サンプル表を使用する場合、ホスト変数は以下の概数値に設定されます。

  ```
  SLOPE:     0.018363799188747826
  ICPT:      69.8388031396513
  RSQR:      0.9526220829162935
  CNT:       5
  AVGSAL:    40850.0
  AVGBONUS:  820.0
  SXX:       4.74575E8
  SYY:       168000.0
  SXY:       8715000.0
  ```

STDDEV_POP または STDDEV

STDDEV_POP 関数は、数値の集合のバイアス標準偏差 (/n) を戻します。



バイアス標準偏差を計算するための数式は以下のとおりです。

```
STDDEV_POP = SQRT(VAR_POP)
```

ここで、SQRT(VAR_POP) は差の平方根です。

numeric-expression
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数が文字ストリングまたはグラフィック・ストリングの場合、関数を評価する前に DECFLOAT(34) にキャストされます。

引数が DECFLOAT(n) である場合、関数の結果は DECFLOAT(34) です。それ以外の場合、結果のデータ・タイプは、倍精度の浮動小数点数になります。

この関数は、引数の値から NULL 値を除いた値の集合に対して適用されます。DISTINCT が指定される場合は、重複する値は除かれます。

結果が、NULL になることもあります。値のセットが空の場合、結果は NULL 値になります。それ以外の場合は、結果は集合の中の値の標準偏差となります。

値が加算される順序は未定義ですが、中間結果はすべて結果のデータ・タイプの範囲内になければなりません。

注

特殊値 **DECFLOAT** を含む結果**:** 引数のデータ・タイプが 10 進浮動小数点であり、特殊値 sNaN または -sNaN、もしくは +Infinity と -Infinity の両方が集約に含まれる場合、エラーまたは警告が戻されます。それ以外の場合、+NaN または -NaN が検出されれば、結果は +NaN または -NaN になります。+Infinity または -Infinity が検出されれば、結果は +Infinity または -Infinity になります。

代替構文**:** SQL 2003 規格に準拠して STDEV_POP を使用する必要があります。

例

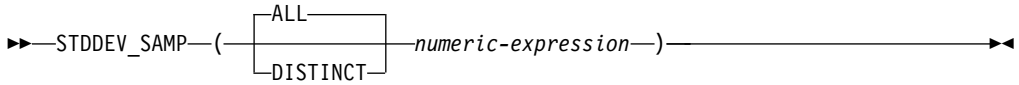
- 表 EMPLOYEE を使用して、部門 A00 の従業員の給与の標準偏差を、ホスト変数 DEV (倍精度浮動小数点数) にセットします。

```
SELECT STDDEV_POP(SALARY)
  INTO :DEV
  FROM EMPLOYEE
  WHERE WORKDEPT = 'A00';
```

上記の結果、DEV は約 9742.43 にセットされます。

STDDEV_SAMP

STDDEV_SAMP 関数は、数値の集合の標本標準偏差 (/n-1) を戻します。



標本標準偏差を計算するための数式は以下のとおりです。

```
STDDEV_SAMP = SQRT(VAR_SAMP)
```

ここで、SQRT(VAR_SAMP) は標本分散の平方根です。

numeric-expression
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数が文字ストリングまたはグラフィック・ストリングの場合、関数を評価する前に DECFLOAT(34) にキャストされます。

引数が DECFLOAT(n) である場合、関数の結果は DECFLOAT(34) です。それ以外の場合、結果のデータ・タイプは、倍精度の浮動小数点数になります。

この関数は、引数の値から NULL 値を除いた値の集合に対して適用されます。DISTINCT が指定される場合は、重複する値は除かれます。

結果が、NULL になることもあります。値の集合が空であるか、1 行のみを含む場合、結果は NULL 値になります。それ以外の場合は、結果は集合の中の値の標準偏差となります。

値が加算される順序は未定義ですが、中間結果はすべて結果のデータ・タイプの範囲内になければなりません。

注

特殊値 **DECFLOAT** を含む結果**:** 引数のデータ・タイプが 10 進浮動小数点であり、特殊値 sNaN または -sNaN、もしくは +Infinity と -Infinity の両方が集約に含まれる場合、エラーまたは警告が戻されます。それ以外の場合、+NaN または -NaN が検出されれば、結果は +NaN または -NaN になります。+Infinity または -Infinity が検出されれば、結果は +Infinity または -Infinity になります。

例

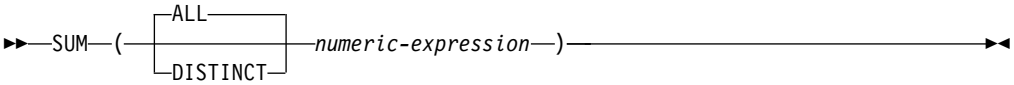
- 表 EMPLOYEE を使用して、部門 A00 の従業員の給与の標本標準偏差を、ホスト変数 DEV (倍精度浮動小数点数) にセットします。

```
SELECT STDDEV_SAMP(SALARY)
  INTO :DEV
  FROM EMPLOYEE
  WHERE WORKDEPT = 'A00';
```

上記の結果、DEV は約 10892.37 にセットされます。

SUM

SUM 関数は、数値の集合の合計値を戻します。



numeric-expression
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数が文字ストリングまたはグラフィック・ストリングの場合、関数を評価する前に DECFLOAT(34) にキャストされます。

結果のデータ・タイプは、以下の場合を除いて、引数値のデータ・タイプと同じです。

- 引数値が DECFLOAT(16) の場合、DECFLOAT(34) になります。
- 引数値が単精度の浮動小数点数の場合は、倍精度の浮動小数点数になります。
- 引数値が短整数である場合、結果は長整数になります。
- 引数が、10 進数または位取りがゼロ以外の 2 進数で、精度が *p*、位取りが *s* である場合は、精度が *mp* で位取りが *s* の 10 進数になります。

p、*s*、および *mp* の値については、 198 ページの『SQL での 10 進数演算』を参照してください。

この関数は、引数の値から NULL 値を除いた値の集合に対して適用されます。DISTINCT が指定される場合は、重複する値は除かれます。

値が加算される順序は未定義ですが、中間結果はすべて結果のデータ・タイプの範囲内になければなりません。

注

DECFLOAT 特殊値が存在する場合の結果: 引数のデータ・タイプが 10 進浮動小数点数の場合に、特殊値 sNaN または -sNaN、あるいは +Infinity と -Infinity の両方が集約の中に含まれていると、エラーが通知されます。それ以外の場合、+NaN または -NaN が検出されれば、結果は +NaN または -NaN になります。+Infinity または -Infinity が検出されれば、結果は +Infinity または -Infinity になります。

例

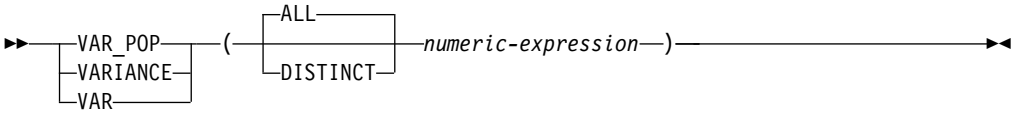
- EMPLOYEE 表を使用して、事務職員 (JOB='CLERK') に支払われるボーナス (BONUS) の総額をホスト変数 JOB_BONUS (DECIMAL(9,2)) に設定します。

```
SELECT SUM(BONUS)
  INTO :JOB_BONUS
  FROM EMPLOYEE
  WHERE JOB = 'CLERK'
```

上記の結果、JOB_BONUS は 4000 にセットされます。

VAR_POP または VARIANCE または VAR

VAR_POP 関数は、数値の集合のバイアス偏差 (/n) を戻します。



バイアス偏差を計算するための数式は以下のとおりです。

```
VAR_POP = SUM(X**2)/COUNT(X) - (SUM(X)/COUNT(X))**2
```

numeric-expression
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数が文字ストリングまたはグラフィック・ストリングの場合、関数を評価する前に DECFLOAT(34) にキャストされます。

引数が DECFLOAT(n) である場合、関数の結果は DECFLOAT(34) です。それ以外の場合、結果のデータ・タイプは、倍精度の浮動小数点数になります。

この関数は、引数の値から NULL 値を除いた値の集合に対して適用されます。DISTINCT が指定される場合は、重複する値は除かれます。

結果が、NULL になることもあります。値のセットが空の場合、結果は NULL 値になります。それ以外の場合は、結果は集合の中の値の偏差になります。

値が加算される順序は未定義ですが、中間結果はすべて結果のデータ・タイプの範囲内になければなりません。

注

特殊値 **DECFLOAT** を含む結果**:** 引数のデータ・タイプが 10 進浮動小数点であり、特殊値 sNaN または -sNaN、もしくは +Infinity と -Infinity の両方が集約に含まれる場合、エラーまたは警告が戻されます。それ以外の場合、+NaN または -NaN が検出されれば、結果は +NaN または -NaN になります。+Infinity または -Infinity が検出されれば、結果は +Infinity または -Infinity になります。

代替構文**:** SQL 2003 規格に準拠して VAR_POP を使用する必要があります。

例

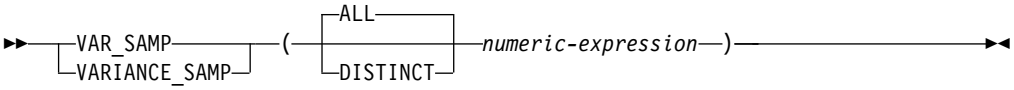
- 表 EMPLOYEE を使用して、部門 A00 の従業員の給与の偏差を、ホスト変数 VARNCE (倍精度浮動小数点数) にセットします。

```
SELECT VAR_POP(SALARY)
  INTO :VARNCE
  FROM EMPLOYEE
  WHERE WORKDEPT = 'A00';
```

上記の結果、VARNCE は約 94 915 000 にセットされます。

VAR_SAMP または VARIANCE_SAMP

VAR_SAMP 関数は、数値の集合の標本分散 (/n-1) を戻します。



標本分散を計算するための数式は以下のとおりです。

```
VAR_SAMP = (SUM(X**2) - ((SUM(X)**2) / (COUNT(X)))) / (COUNT(X) - 1)
```

numeric-expression
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数が文字ストリングまたはグラフィック・ストリングの場合、関数を評価する前に DECFLOAT(34) にキャストされます。

引数が DECFLOAT(n) である場合、関数の結果は DECFLOAT(34) です。それ以外の場合、結果のデータ・タイプは、倍精度の浮動小数点数になります。

この関数は、引数の値から NULL 値を除いた値の集合に対して適用されます。DISTINCT が指定される場合は、重複する値は除かれます。

結果が、NULL になることもあります。値の集合が空であるか、1 行のみを含む場合、結果は NULL 値になります。それ以外の場合は、結果は集合の中の値の偏差になります。

値が加算される順序は未定義ですが、中間結果はすべて結果のデータ・タイプの範囲内になければなりません。

注

特殊値 **DECFLOAT** を含む結果**:** 引数のデータ・タイプが 10 進浮動小数点であり、特殊値 sNaN または -sNaN、もしくは +Infinity と -Infinity の両方が集約に含まれる場合、エラーまたは警告が戻されます。それ以外の場合、+NaN または -NaN が検出されれば、結果は +NaN または -NaN になります。+Infinity または -Infinity が検出されれば、結果は +Infinity または -Infinity になります。

代替構文**:** SQL 2003 規格に準拠して VAR_SAMP を使用する必要があります。

例

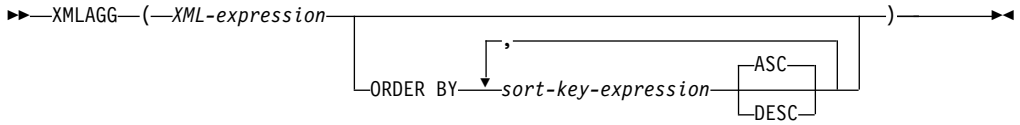
- 表 EMPLOYEE を使用して、部門 A00 の従業員の給与の標本分散を、ホスト変数 VARNCE (倍精度浮動小数点数) にセットします。

```
SELECT VAR_SAMP(SALARY)
  INTO :VARNCE
  FROM EMPLOYEE
  WHERE WORKDEPT = 'A00';
```

上記の結果、VARNCE は約 1 186 437 500 にセットされます。

XMLAGG

XMLAGG 関数は、1 組の XML 値の中の NULL 以外の値ごとに 1 つの項目を含む XML シーケンスを戻します。



XML-expression
XML 値を戻す式。

ORDER BY
集約で処理される、同じグループ化セットからの行の順序を指定します。ORDER BY 文節を指定しない場合、または ORDER BY 文節ではソート・キー値の順序を区別できない場合、同じグループ内の行は任意に順序付けられます。

sort-key-expression
列名または式のどちらかのソート・キー値を指定します。列または式のデータ・タイプは、DATALINK 値または XML 値であってはなりません。

順序付けはソート・キーの値に基づいて行われ、これは *XML-expression* の中で使用される場合もされない場合もあります。

ソート・キー式 の長さ属性の合計は 3.5 ギガバイトを超えてはなりません。

XMLAGG 関数を含むステートメントの実行時に *HEX 以外の照合順序が有効で、しかもソート・キー式 が SBCS データ、混合データ、または Unicode データの場合、結果は重み付けされた値の比較によって求められます。この重み付けされた値は、ソート・キー式 に照合順序を適用して得られます。

この関数は、引数の値から NULL 値を除いた値の集合に対して適用されます。

結果のデータ・タイプは XML です。結果が、NULL になることもあります。値のセットが空の場合、結果は NULL 値になります。それ以外の場合、結果はそのセット内のそれぞれの値ごとに 1 つの項目を含む XML シーケンスです。

例

注: XMLAGG は、出力に空白スペースや改行文字を挿入しません。読みやすくするために、すべての出力例はフォーマット設定されています。

- 従業員を部門別にグループ化し、部門の名前を属性として使用して各部門ごとに「Department」エレメントを生成し、各部門の従業員に対応する「emp」エレメントをすべてネストし、「emp」エレメントを LASTNAME によって順序付けます。

```
SELECT XMLSERIALIZE(XMLDOCUMENT (
            XMLELEMENT(NAME "Department",
                       XMLATTRIBUTES(E.WORKDEPT AS "name"),
                       XMLAGG(XMLELEMENT ( NAME "emp", E.LASTNAME)
                              ORDER BY E.LASTNAME)
                       ))
```

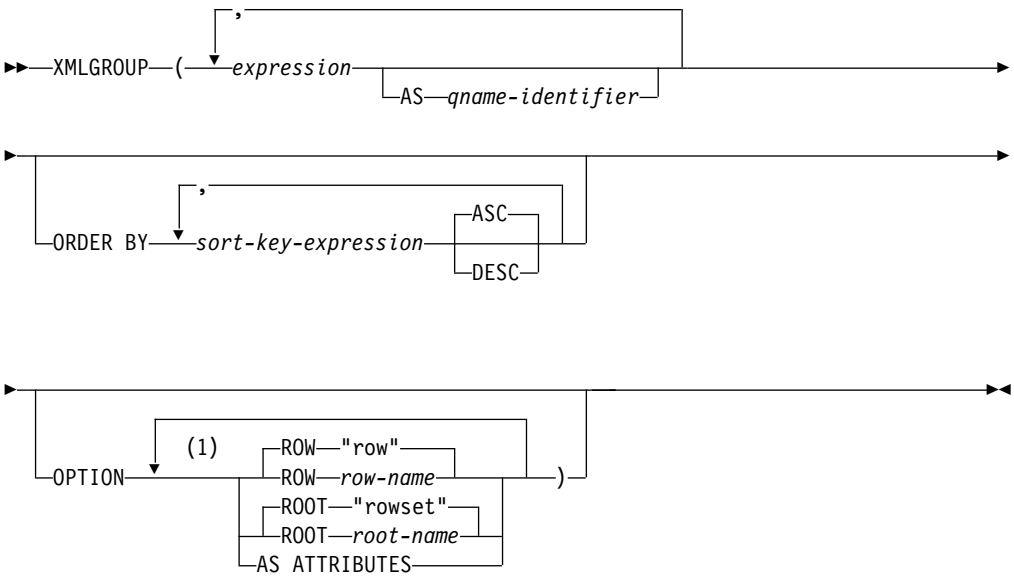
```
                        AS CLOB(200)) AS "dept_list"
  FROM EMPLOYEE E
  WHERE E.WORKDEPT IN ('C01', 'E21')
  GROUP BY WORKDEPT
```

照会の結果は、次の結果のようになります。

```
dept_list
---------------------------------
<Department name="C01">
  <emp>KWAN</emp>
  <emp>NICHOLLS</emp>
  <emp>QUINTANA</emp>
</Department>
<Department name="E21">
  <emp>GOUNOT</emp>
  <emp>LEE</emp>
  <emp>MEHTA</emp>
  <emp>SPENSER</emp>
</Department>
```

XMLGROUP

XMLGROUP 関数は、整形式 XML 文書である XML 値を戻します。



注:

1 同じ文節を複数回指定することはできません。

expression
各 XML エレメントの内容を式で指定します。 *expression* のデータ・タイプは、ROWID または DATALINK、あるいは ROWID または DATALINK に基づく特殊タイプ以外でなければなりません。式として、SQL 式を指定することもできます。式が単純な列参照でない場合は、エレメント名を指定する必要があります。AS ATTRIBUTES を指定する場合、*expression* のデータ・タイプは、XML、または XML に基づく特殊タイプ以外でなければなりません。

AS *qname-identifier*
XML のエレメント名または属性名を SQL ID として指定します。 *qname-identifier* は、XML 修飾名 (つまり QName) の形式でなければなりません。有効な名前についての詳細は、W3C XML ネーム・スペースの指定の項目を参照してください。名前を修飾する場合は、有効範囲内にネーム・スペース接頭部が宣言されている必要があります。*qname-identifier* を指定しない場合は、*expression* を列名にする必要があります。エレメント名または属性名は、列名から QName への完全エスケープ・マッピングを使用して、列名から作成されます。

ORDER BY
集約で処理される、同じグループ化セットからの行の順序を指定します。 ORDER BY 文節を指定しない場合、または ORDER BY 文節ではソート・キー値の順序を区別できない場合、同じグループ内の行は任意に順序付けられます。

sort-key-expression
列名または式のどちらかのソート・キー値を指定します。列または式のデータ・タイプは、DATALINK 値または XML 値であってはなりません。

順序付けはソート・キーの値に基づいて行われ、これは *XML-expression* の中で使用される場合もされない場合もあります。

OPTION
XML 値を構成するための追加オプションを指定します。OPTION 文節を指定しない場合は、デフォルトの動作が適用されます。

ROW *row-name*
各行に対応付けるエレメントの名前を指定します。このオプションを指定しない場合のデフォルトのエレメント名は、「row」です。

ROOT *root-name*
ルート・エレメントの名前を指定します。このオプションを指定しない場合のデフォルトのルート・エレメント名は、「rowset」です。

AS ATTRIBUTES
それぞれの式を属性値に対応付けることを指定します。属性名としての役割を果たすのは、列名または *qname-identifier* です。

XMLGROUP 関数を含むステートメントの実行時に *HEX 以外の照合順序が有効で、しかもソート・キー式 が SBCS データ、混合データ、または Unicode データの場合、結果は重み付けされた値の比較によって求められます。この重み付けされた値は、ソート・キー式 に照合順序を適用して得られます。

この関数の結果は XML です。結果が、NULL になることもあります。値の集合が空である場合は、結果は NULL 値です。それ以外の場合、結果はそのセット内のそれぞれの値ごとに 1 つの項目を含む XML シーケンスです。

注

デフォルトの動作では、結果セットと XML 値の簡単な対応付けを定義するだけですが、関数の動作に当てはまる追加の注意点を以下にまとめます。

- デフォルトでは、各行が「row」という名前の XML エレメントに変換され、各 *expression* が、ネストされたエレメントに変換されます (列名または *qname-identifier* がエレメント名になります)。
- NULL 処理の動作は、NULL ON NULL です。サブエレメントが存在しないという状態に、式の NULL 値が対応付けられます。すべての式値が NULL であれば、行のエレメントは生成されません。行のエレメントが生成されなければ、関数から NULL 値が返されます。
- バイナリー・データ・タイプと FOR BIT DATA データ・タイプのバイナリー・コード化スキームは、base64Binary エンコード方式です。
- ルート・エレメントに組み込まれる行のサブエレメントの順序は、照会の結果セットで返される行の順序と同じです。

例

注: XMLGROUP は、出力に空白スペースや改行文字を挿入しません。読みやすくするために、すべての出力例はフォーマット設定されています。

表 T1 に列 C1 と C2 があるとします。

```
 C1    C2
----  ----
  1     2
  -     2
  1     -
  -     -
```

- デフォルトの動作で実行する XMLGROUP 照会と出力断片の例を示します。1 つの最上位エレメントが表に対応しています。

  ```
  SELECT XMLGROUP(C1, C2) FROM T1
  ```

  単一の結果行に以下の値を戻します。

  ```
  <rowset>
  <row><C1>1</C1><C2>2</C2></row>
  <row><C2>2</C2></row>
  <row><C1>1</C1></row>
  </rowset>
  ```

- 属性中心の対応付けを指定して実行する XMLGROUP 照会と出力断片の例を以下に示します。前の例では、ネストされたエレメントとしてデータを表示しましたが、この例では、データをエレメントの属性に対応付けます。

  ```
  SELECT XMLGROUP(C1, C2 OPTION AS ATTRIBUTES) FROM T1
  ```

  単一の結果行に以下の値を戻します。

  ```
  <rowset>
  <row C1="1" C2="2"/>
  <row C2="2"/>
  <row C1="1"/>
  </rowset>
  ```

- デフォルトの <rowset> ルート・エレメントを <document> に置き換え、デフォルトの <row> エレメントを <entry> に置き換えて実行する XMLGROUP 照会と出力断片の例を以下に示します。<column1> エレメントと <column2> エレメントとして列 C1 と C2 を返し、列 C1 を基準にして戻り値のセットを配列します。

  ```
  SELECT XMLGROUP(C1 AS "column1", C2 AS "column2"
                  ORDER BY C1 OPTION ROW "entry" ROOT "document")
         FROM T1
  ```

  単一の結果行に以下の値を戻します。

  ```
  <document>
  <entry> <column1>1</column1><column2>2</column2></entry>
  <entry> <column1>1</column1></entry>
  <entry> <column2>2</column2></entry>
  </document>
  ```

スカラー関数

スカラー関数 は入力引数をとり、単一値結果を返します。スカラー関数は、式を使える場所であればどこでも使用することができます。

スカラー関数は、値の集合ではなく単一のパラメーター値に適用されるものなので、集約関数を使用するときの制約事項はスカラー変数には適用されません。スカラー関数では、関数を引数として使用できます。ただし、スカラー関数の中で式や集約関数を使用する場合は、それらの式および集約関数の使用法に適用される制約が適用されます。例えば、スカラー関数の引数に集約関数を指定できるのは、そのスカラー関数が使用される文脈で集約関数が許される場合だけです。

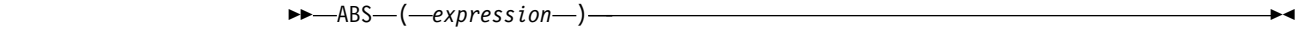
例

次の SELECT ステートメントの結果は、部門 D01 の従業員数と同じ行数になります。

```
SELECT EMPNO, LASTNAME, YEAR(CURRENT DATE - BIRTHDATE)
  FROM EMPLOYEE
  WHERE WORKDEPT = 'D01'
```

ABS

ABS 関数は、数値の絶対値を戻します。



expression
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

結果のデータ・タイプは、以下の場合を除いて、引数値のデータ・タイプと同じです。

- 引数値が単精度の浮動小数点数の場合は、倍精度の浮動小数点数になります。
- 引数値が短整数である場合、結果は長整数になります。
- 引数が、10 進数または位取りがゼロ以外の 2 進数で、精度が *p*、位取りが *s* である場合は、精度が *mp* で位取りが *s* の 10 進数になります。

注記

DECFLOAT 特殊値が関係する場合の結果**:** 10 進浮動小数点値の場合、特殊値は次のように扱われます。

- ABS(NaN) および ABS(-NaN) は NaN を返します。
- ABS(Infinity) および ABS(-Infinity) は Infinity を返します。
- ABS(sNaN) および ABS(-sNaN) は sNaN を返します。

代替構文**:** ABSVAL は ABS の同義語です。これは、Db2 の旧リリースとの互換性を維持するためにのみサポートされています。

例

- ホスト変数 PROFIT は、値が -50000 の長整数であると想定します。

```
SELECT ABS(:PROFIT)
  FROM SYSIBM.SYSDUMMY1
```

値として 50000 が戻されます。

ACOS

ACOS 関数は、引数のアークコサイン (逆余弦) を、ラジアンで表した角度で戻します。 ACOS 関数と COS 関数は、逆の演算です。

expression
任意の組み込み数値データ・タイプ (DECFLOAT を除く)、文字ストリングまたはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。値は、-1 以上で、1 以下でなければなりません。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

結果は、0 以上で、π 以下になります。

例

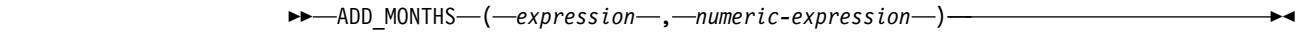
- ホスト変数 ACOSINE は、DECIMAL(10,9) で値が 0.070737202 のホスト変数であると想定します。

```
SELECT ACOS(:ACOSINE)
  FROM SYSIBM.SYSDUMMY1
```

およそ 1.49 の値が戻されます。

ADD_MONTHS

ADD_MONTHS 関数は、*expression* に *numeric-expression* の月数を加えた日付またはタイム・スタンプを戻します。



expression
日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式。

expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

numeric-expression
ゼロのスケールの組み込み数値データ・タイプの値を戻す式。負の数値は許可されます。

expression がタイム・スタンプの場合、この関数の結果は、expression と同じ精度のタイム・スタンプです。そうでない場合、この関数の結果は日付です。引数のどちらかが NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数のどちらかが NULL である場合は、結果は NULL 値になります。

式 が月の最後の日であるか、または結果の月に式 の日のコンポーネントより少ない日数しか含まれない場合、結果は、結果の月の最後の日になります。その他の場合、結果は式 と同じ日のコンポーネントを持ちます。

例

- 今日が 2000 年 1 月 31 日であると想定します。ホスト変数 ADD_MONTH に、1 月の最後の日に 1 カ月を加えたものを設定します。

```
SET :ADD_MONTH = ADD_MONTHS(LAST_DAY(CURRENT_DATE), 1 )
```

  ホスト変数 ADD_MONTH は、2 月の終わりである 2000-02-29 を表す値で設定されます。

- DATE が 1965 年 7 月 27 日の値のホスト変数であると想定します。ホスト変数 ADD_MONTH に、この日の値に 3 カ月を加えたものを設定します。

```
SET :ADD_MONTH = ADD_MONTHS(:DATE, 3)
```

  ホスト変数 ADD_MONTH は、この日に 3 カ月を足した 1965-10-27 を表す値で設定されます。

- ADD_MONTHS 関数と日付の算術計算で、類似した結果を得ることができます。以下の例では、類似点と差異が分かります。

```
SET :DATEHV = DATE('2000-2-28') + 4 MONTHS
SET :DATEHV ADD_MONTHS('2000-2-28', 4)
```

  どちらの場合も、ホスト変数 DATEHV には値「2000–06–28」が設定されます。

  今度は、同じ例で日付「2000–2–29」を引数として使用してみます。

```
SET :DATEHV = DATE('2000-2-29') + 4 MONTHS
```

ホスト変数 DATEHV には値「2000–06–29」が設定されます。

```
SET :DATEHV ADD_MONTHS('2000-2-29', 4)
```

ホスト変数 DATEHV には値「2000–06–30」が設定されます。

この場合、ADD_MONTHS 関数は、2000 年 6 月 29 日ではなく、月の最後の日である 2000 年 6 月 30 日を戻します。理由は、2 月 29 日が月の最後の日だからです。それで、ADD_MONTHS 関数は 6 月の最後の日を戻します。

ANTILOG

ANTILOG 関数は、数値の逆対数 (底 10) を戻します。ANTILOG 関数と LOG 関数は、逆の演算です。

expression
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

引数のデータ・タイプが DECFLOAT(*n*) の場合、結果は DECFLOAT(*n*) です。それ以外の場合、結果のデータ・タイプは、倍精度の浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

注記

DECFLOAT 特殊値が関係する場合の結果**:** 10 進浮動小数点値の場合、特殊値は次のように扱われます。

- ANTILOG(NaN) は NaN を返します。
- ANTILOG(-NaN) は -NaN を返します。
- ANTILOG(Infinity) は Infinity を返します。
- ANTILOG(-Infinity) は 0 を返します。
- ANTILOG(sNaN) および ANTILOG(-sNaN) は、警告またはエラーを返します。[50]

例

- ホスト変数 ALOG は、DECIMAL(10,9) のホスト変数で、値は 1.499961866 であると想定します。

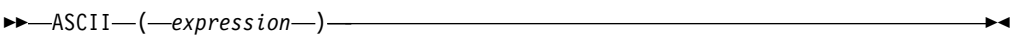
```
SELECT ANTILOG(:ALOG)
  FROM SYSIBM.SYSDUMMY1
```

およそ 31.62 の値が戻されます。

50. SQL_DECFLOAT_WARNINGS 照会オプションに *YES を指定すると、NaN および -NaN がそれぞれ警告と一緒に返されます。

ASCII

ASCII 関数は、引数の左端の文字の ASCII コード値を整数として戻します。



expression
評価するための文字を含むストリングを指定する式。*expression* には、任意の組み込み数値またはストリング・データ・タイプを指定する必要があります。ストリングの最初の文字は、この関数で処理できるように ASCII CCSID 367 に変換されます。

この関数の結果は長精度整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

- 'A' の ASCII 表現を表す整数値を返します。

```
SELECT ASCII('A')
    FROM SYSIBM.SYSDUMMY1
```

値として 65 が返されます。

ASIN

ASIN 関数は、引数のアークサイン (逆正弦) を、ラジアンで表した角度で戻します。 ASIN 関数と SIN 関数は、逆の演算です。

expression
任意の組み込み数値データ・タイプ (DECFLOAT を除く)、文字ストリングまたはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。値は、-1 以上で、1 以下でなければなりません。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

結果は -π /2 以上で、π /2 以下になります。

例

- ホスト変数 ASINE は、DECIMAL(10,9) のホスト変数で値が 0.997494987 であると想定します。

```
SELECT ASIN(:ASINE)
  FROM SYSIBM.SYSDUMMY1
```

およそ 1.50 の値が戻されます。

ATAN

ATAN 関数は、引数のアークタンジェント (逆正接) を、ラジアンで表した角度で戻します。 ATAN 関数と TAN 関数は、逆の演算になります。

expression
任意の組み込み数値データ・タイプ (DECFLOAT を除く)、文字ストリングまたはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

結果は、$-\pi$ /2 以上で、π /2 以下になります。

例

- ホスト変数 ATANGENT は、DECIMAL(10,8) のホスト変数で値が 14.10141995 であると想定します。

```
SELECT ATAN(:ATANGENT)
  FROM SYSIBM.SYSDUMMY1
```

およそ 1.50 の値が戻されます。

ATANH

ATANH 関数は、数値の双曲線アークタンジェント (双曲線逆正接) をラジアンで戻します。 ATANH 関数と TANH 関数は、逆の演算です。

expression
任意の組み込み数値データ・タイプ (DECFLOAT を除く)、文字ストリングまたはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。値は、-1 より大きく 1 より小さくなければなりません。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

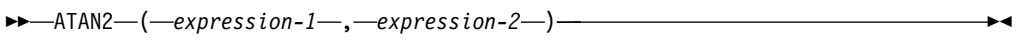
- ホスト変数 HATAN が、DECIMAL(10,9) のホスト変数で 0.905148254 の値を持つものと想定します。

```
SELECT ATANH(:HATAN)
  FROM SYSIBM.SYSDUMMY1
```

およそ 1.50 の値が戻されます。

ATAN2

ATAN2 関数は、x 座標と y 座標のアークタンジェント (逆正接) を、ラジアンで表された角度として戻します。最初の引数と 2 番目の引数は、それぞれ x 座標と y 座標を表します。



expression-1
任意の組み込み数値データ・タイプ (DECFLOAT を除く)、文字ストリングまたはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。一方の引数が 0 の場合は、もう一方の引数は 0 であってはなりません。

expression-2
任意の組み込み数値データ・タイプ (DECFLOAT を除く)、文字ストリングまたはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。一方の引数が 0 の場合は、もう一方の引数は 0 であってはなりません。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

例

- ホスト変数 HATAN2A と HATAN2B は、それぞれ値が 1 と 2 の DOUBLE ホスト変数であるとします。

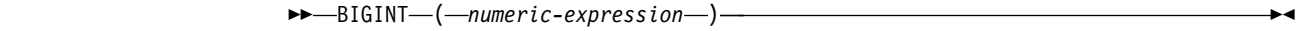
```
SELECT ATAN2(:HATAN2A,:HATAN2B)
  FROM SYSIBM.SYSDUMMY1
```

これは、概略値 1.1071487 の倍精度の浮動小数点数を戻します。

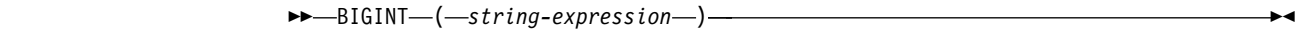
BIGINT

BIGINT 関数は、64 ビット整数表現を返します。

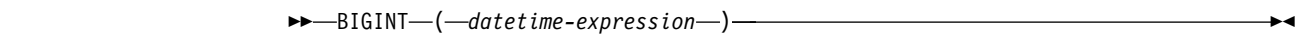
数値から **64** ビット整数へ



ストリングから **64** ビット整数



日時→ **Big Integer**



BIGINT 関数は、次のものの 64 ビット整数表現を戻します。

- 数値
- 10 進数の文字ストリング表現またはグラフィック・ストリング表現
- 整数の文字ストリング表現またはグラフィック・ストリング表現
- 浮動小数点数の文字ストリング表現またはグラフィック・ストリング表現
- 10 進浮動小数点数の文字ストリング表現またはグラフィック・ストリング表現
- 日付
- 時刻
- タイム・スタンプ

数値から 64 ビット整数へ

numeric-expression
任意の組み込み数値データ・タイプの数値を戻す式。

結果は、引数が 64 ビット整数の列または変数に割り当てられたときに得られる数値と同じです。 引数の整数部が、64 ビット整数の範囲内にない場合は、エラーが戻されます。引数の小数部は切り捨てられます。

ストリングから 64 ビット整数へ

string-expression
数値の文字ストリング表現またはグラフィック・ストリング表現の値を戻す式。

結果は、CAST(*string-expression* AS BIGINT) から得られる数値と同じです。先行ブランクと末尾ブランクは除去され、結果のストリングは、浮動小数点数、10 進浮動小数点数、整数、または 10 進数の定数を形成する際の規則に合致している必要があります。 引数の整数部が、64 ビット整数の範囲内にない場合は、エラーが戻されます。引数の小数部は切り捨てられます。

日時→ Big Integer

datetime-expression
次のデータ・タイプのいずれかの式。

- DATE。結果は、日付を *yyyymmdd* で表した BIGINT 値になります。
- TIME。結果は、時間を *hhmmss* で表した BIGINT 値になります。
- TIMESTAMP。結果は、タイム・スタンプを *yyyymmddhhmmss* で表した BIGINT 値になります。タイム・スタンプの小数秒の部分は、結果には入っていません。

この関数の結果は、64 ビット整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

注記

代替構文: アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、 218 ページの『CAST の指定』を参照してください。

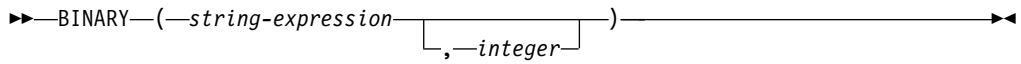
例

- EMPLOYEE 表を使用して、64 ビット整数形式の SALARY 列を選択し、アプリケーション内の処理をさらに進めます。

```
SELECT BIGINT(SALARY)
  FROM EMPLOYEE
```

BINARY

BINARY 関数は、任意のタイプのストリングの BINARY 表現を戻します。



この関数の結果は、固定長 2 進ストリングになります。最初の引数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引数がヌルの場合は、結果は NULL 値になります。

string-expression
値が組み込み文字ストリング、グラフィック・ストリング、2 進ストリング、または行 ID データ・タイプのいずれかでなければならないストリング式。

integer
結果の 2 進ストリングの長さ属性を指定する整数定数。値は 1 から 32766 でなければなりません。

整数 を指定しなかった場合は、以下のようになります。

- ストリング式 が空ストリング定数の場合は、結果の長さ属性は 1。
- 空ストリング定数以外の場合は、最初の引数が漢字ストリングでなければ、結果の長さ属性は最初の引数の長さ属性と同じになる。グラフィック・ストリングの場合は、結果の長さ属性は、引数の長さ属性の 2 倍です。

実際の長さは、結果の長さ属性と同じになります。ストリング式 の長さが結果の長さより小さい場合は、結果は、結果に指定されている長さまで 16 進数のゼロで埋め込まれます。ストリング式 の長さが結果の長さ属性よりも大きい場合は、切り捨てが行われます。最初の入力引数が文字ストリングで、切り捨てられた文字がすべてブランクである場合、最初の入力引数がグラフィック・ストリングで、切り捨てられた文字がすべて 2 バイト・ブランクである場合、または最初の入力引数が 2 バイト・ストリングで、切り捨てられたバイトがすべて 16 進数のゼロである場合以外は、警告 (SQLSTATE 01004) が戻されます。

注記

代替構文**:** 長さを指定する場合、アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、 218 ページの『CAST の指定』を参照してください。

例

- 次の関数は、ストリング「This is a BINARY」の BINARY を戻します。

```
SELECT BINARY('This is a BINARY')
  FROM SYSIBM.SYSDUMMY1
```

BITAND、BITANDNOT、BITOR、BITXOR、および BITNOT

これらのビット関数は、入力引数の整数値の「2 の補数」表現について操作し、入力引数のデータ・タイプに基づくデータ・タイプで、対応する 10 進整数値として結果を戻します。

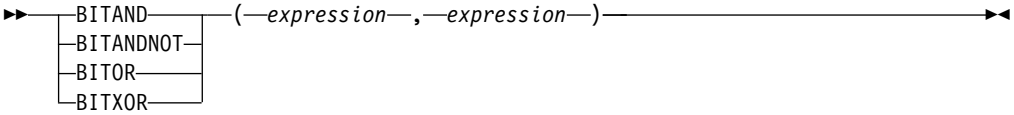


表 49. ビット操作関数

関数	説明	結果における 2 の補数表現のビット
BITAND	ビット単位の AND 演算を実行します。	両方の引数の対応するビットが 1 の場合にのみ 1。
BITANDNOT	2 番目の引数の中にある最初の引数内のすべてのビットをクリアします。	2 番目の引数にある対応するビットが 1 の場合にはゼロ。その他の場合には、最初の引数内の対応ビットから結果がコピーされます。
BITOR	ビット単位の OR 演算を実行します。	両方の引数の対応するビットがゼロでない場合は、1。
BITXOR	ビット単位の排他 OR 演算を実行します。	両方の引数の対応するビットが同じでない場合は、1。
BITNOT	ビット単位の NOT 演算を実行します。	引数内の対応するビットを反転します。

expression
任意の組み込み数値データ・タイプの値を戻す式。タイプ DECIMAL、REAL、または DOUBLE の引数は、DECFLOAT にキャストされます。値は、整数に切り捨てられます。

ビット操作関数が操作できる最大ビット数は、SMALLINT が 16 ビット、INTEGER が 32 ビット、BIGINT が 64 ビット、DECFLOAT が 113 ビットです。DECFLOAT 値のサポート範囲は、-2^{112} から 2^{112} -1 までの整数で、NaN または INFINITY などの特殊値はサポートしていません。2 つの引数のデータ・タイプが異なる場合、サポートしているビット数が少ない引数が、より多くのビットをサポートしている引数のデータ・タイプの値にキャストされます。このキャストは、負の値に設定されているビットに影響を与えます。例えば、SMALLINT 値の -1 は 16 ビットで 1 に設定されていますが、INTEGER 値にキャストされると 32 ビットで 1 となります。

2 つの引数を取るこうした関数の結果は、上位変換のためのデータ・タイプ優先順位リストで上位にある引数のデータ・タイプとなります。どちらかの引数が

DECFLOAT の場合、結果のデータ・タイプは DECFLOAT(34) となります。引数のどちらかが NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数のどちらかが NULL である場合は、結果は NULL 値になります。

BITNOT 関数の結果は、入力引数のデータ・タイプと同じになります。ただし、DECIMAL、REAL、DOUBLE、または DECFLOAT(16) の場合には、DECFLOAT(34) を戻します。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

1 つの値で複数のビットをトグルさせる場合には、BITXOR 関数を使用することをお勧めします。BITANDNOT 関数を使用すると、ビットがクリアされます。BITANDNOT(val, pattern) の方が、BITAND(val, BITNOT(pattern)) よりも効率的に機能します。

例

以下の例は、タイプ INTEGER の PROPERTIES 列が入っている ITEM 表に基づいています。

- 3 番目のプロパティー・ビットが設定されているすべての項目を戻します。

```
SELECT ITEMID FROM ITEM
 WHERE BITAND(PROPERTIES, 4) = 4
```

- 4 番目または 6 番目のプロパティー・ビットが設定されているすべての項目を戻します。

```
SELECT ITEMID FROM ITEM
 WHERE BITAND(PROPERTIES, 40) <> 0
```

- ID が 3412 の項目の 12 番目のプロパティーをクリアします。

```
UPDATE ITEM
 SET PROPERTIES = BITANDNOT(PROPERTIES, 2048)
 WHERE ITEMID = 3412
```

- ID が 3412 の項目の 5 番目のプロパティーを設定します。

```
UPDATE ITEM
 SET PROPERTIES = BITOR(PROPERTIES, 16)
 WHERE ITEMID = 3412
```

- ID が 3412 の項目の 11 番目のプロパティーをトグルします。

```
UPDATE ITEM
 SET PROPERTIES = BITXOR(PROPERTIES, 1024)
 WHERE ITEMID = 3412
```

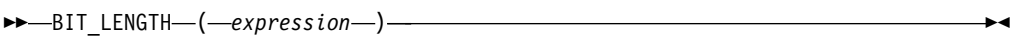
- 2 番目のビットだけがオンである 16 ビット値のすべてのビットを反転させます。

```
VALUES BITNOT(CAST(2 AS SMALLINT))
```

  -3 を戻します (データ・タイプは SMALLINT です)。

BIT_LENGTH

BIT_LENGTH 関数は、ストリング式の長さをビット数で戻します。



類似の関数については、 522 ページの『LENGTH』、 382 ページの『CHARACTER_LENGTH』、および 570 ページの『OCTET_LENGTH』を参照してください。

expression
任意の組み込み数値データ・タイプ、またはストリング・データ・タイプの値を戻す式。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。

この関数の結果は DECIMAL(31) になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

結果はその引数のビット数 (バイト数 * 8) です。ストリングの長さには、末尾ブランクも含まれます。可変長ストリングを指定した場合に戻る長さは、ビット数 (バイト数 * 8) で表した実際の長さであり、最大長ではありません。

例

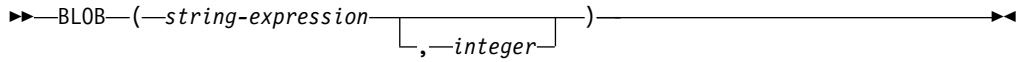
- 表 T1 に C1 という名前の GRAPHIC(10) 列があると想定します。

```
SELECT BIT_LENGTH( C1 )
  FROM T1
```

値として 160 が戻されます。

BLOB

BLOB 関数は、任意のタイプのストリングの BLOB 表現を戻します。



string-expression
値が文字ストリング、グラフィック・ストリング、2 進ストリング、または行 ID のいずれかであるストリング式。

integer
結果の 2 進ストリングの長さ属性を指定する整数定数。値は 1 から 2 147 483 647 でなければなりません。

整数 を指定しなかった場合は、以下のようになります。

- ストリング式 が空ストリング定数の場合は、結果の長さ属性は 1。
- 空ストリング定数以外の場合は、最初の引数が漢字ストリングでなければ、結果の長さ属性は最初の引数の長さ属性と同じになる。グラフィック・ストリングの場合は、結果の長さ属性は、引数の長さ属性の 2 倍です。

結果の実際の長さは、結果の長さ属性と式の実際の長さ (または入力がグラフィック・データの場合は式の長さの 2 倍) のいずれか小さい方となります。ストリング式 の長さが結果の長さ属性よりも大きい場合は、切り捨てが行われます。最初の入力引数が文字ストリングで、切り捨てられた文字がすべてブランクである場合、最初の入力引数がグラフィック・ストリングで、切り捨てられた文字がすべて 2 バイト・ブランクである場合、または最初の入力引数が 2 バイト・ストリングで、切り捨てられたバイトがすべて 16 進数のゼロである場合以外は、警告 (SQLSTATE 01004) が戻されます。

この関数の結果は、BLOB になります。最初の引数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引数がヌルの場合は、結果は NULL 値になります。

注記

代替構文**:** 長さを指定する場合、アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、 218 ページの『CAST の指定』を参照してください。

例

- 次の関数は、ストリング「This is a BLOB」の BLOB を戻します。

```
SELECT BLOB('This is a BLOB')
  FROM SYSIBM.SYSDUMMY1
```

- 次の関数は、ロケーター myclob_locator が識別するラージ・オブジェクトの BLOB を戻します。

```
SELECT BLOB(:myclob_locator)
  FROM SYSIBM.SYSDUMMY1
```

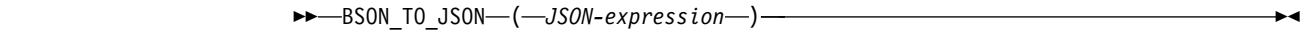
- 表に、TOPOGRAPHIC_MAP という名前の BLOB 列と、MAP_NAME という名前の VARCHAR があるとします。「Pellow Island」というストリングが入っているマップを見つけ、実際のマップの前にマップ名を連結した単一 2 進スト

リングを戻すことにします。次の関数は、ロケーター myclob_locator が識別するラージ・オブジェクトの BLOB を戻します。

```
SELECT BLOB( MAP_NAME CONCAT  ':  ' CONCAT  TOPOGRAPHIC_MAP )
  FROM ONTARIO_SERIES_4
  WHERE TOPOGRAPHIC_MAP LIKE '%Pellow Island%'
```

BSON_TO_JSON

BSON_TO_JSON 関数は、形式設定された BSON データを含むストリングを、JSON として形式設定されたデータを含む文字ストリングに変換します。



JSON-expression

バイナリー・ストリング値を戻す式を指定します。形式設定された BSON データを含まなければなりません。

JSON-expression に重複キーが含まれる場合、その重複キーは結果 JSON ストリングで保持されます。

結果のデータ・タイプは、CCSID が 1208 の CLOB(2G) です。

引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

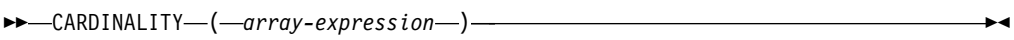
例

- BSON として形式設定されたデータが入った値を、JSON 値を保持する列に挿入します。

```
INSERT INTO TABLE1 (JSON_COLUMN) VALUES (BSON_TO_JSON(:BSON_DATA));
```

CARDINALITY

CARDINALITY 関数は、配列のエレメント数を表わす値を戻します。



array-expression
この式は、配列データ・タイプの SQL 変数またはパラメーターでも、配列データ・タイプに対するパラメーター・マーカーのキャスト仕様であっても構いません。

CARDINALITY 関数によって戻される値は、配列に含まれている割り当て済みエレメントの最も上位の配列指標になります。これには、NULL 値が割り当てられたエレメントも含まれます。

関数の結果は BIGINT です。この関数は、配列が空の場合には 0 を戻します。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

例

配列タイプ PHONENUMBERS および配列変数 RECENT_CALLS が次のように定義されていると想定します。

```
CREATE TYPE PHONENUMBERS AS INTEGER ARRAY[50];
DECLARE RECENT_CALLS PHONENUMBERS;
```

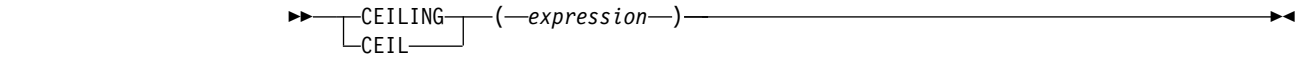
RECENT_CALLS には 3 つのエレメントが入っています。以下の SET ステートメントは、これまでに配列に保存されている通話件数を SQL 変数 HOWMANYCALLS に代入します。

```
SET HOWMANYCALLS = CARDINALITY(RECENT_CALLS)
```

このステートメントを実行した後、HOWMANYCALLS には 3 が入っています。

CEILING または CEIL

CEILING または CEIL 関数は、*expression* より大きいか等しい最小の整数値を戻します。



expression
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

関数の結果のデータ・タイプと長さ属性は引数と同じになります。ただし、引数が 10 進数の場合の位取りは 0 になります。例えば、データ・タイプが DECIMAL(5,5) の引数の場合、結果は DECIMAL(5,0) となります。

引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

注記

DECFLOAT 特殊値が関係する場合の結果**:** 10 進浮動小数点値の場合、特殊値は次のように扱われます。

- CEILING(NaN) は NaN を返します。
- CEILING(-NaN) は -NaN を返します。
- CEILING(Infinity) は Infinity を返します。
- CEILING(-Infinity) は -Infinity を返します。
- CEILING(sNaN) および CEILING(-sNaN) は警告またはエラーを返します。[51]

例

- 全従業員の中で最も高い月収を検索します。結果を次の整数に切り上げます。SALARY 列は、10 進数のデータ・タイプを持っています。

```
SELECT CEIL(MAX(SALARY)/12)
  FROM EMPLOYEE
```

この例では、4396.00 が戻されます。給与が最も高い従業員は Christine Haas であり、その年収は $52750.00 だからです。 CEIL 関数を適用する前の彼女の平均月収は 4395.83 です。

- 正負両方の数値に関して CEILING を使用します。

```
SELECT CEILING( 3.5),
       CEILING( 3.1),
       CEILING(-3.1),
       CEILING(-3.5)
  FROM SYSIBM.SYSDUMMY1
```

51. SQL_DECFLOAT_WARNINGS 照会オプションに *YES を指定すると、NaN および -NaN がそれぞれ警告と一緒に返されます。

この例ではそれぞれ、

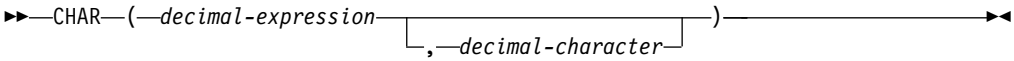
```
04.   04.  -03.  -03.
```

CHAR

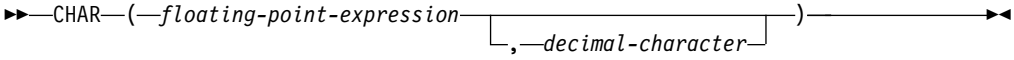
CHAR 関数は、固定長文字ストリング表現を返します。

整数から文字

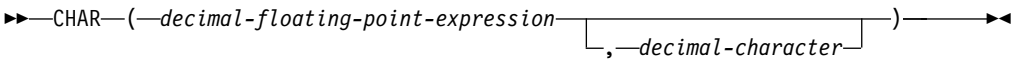
10 進数から文字



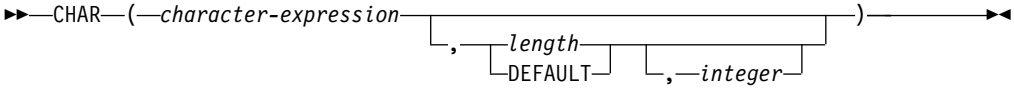
浮動小数点数から文字に



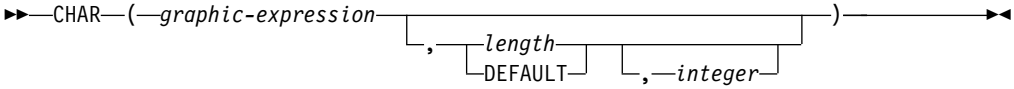
10 進浮動小数点数から文字に



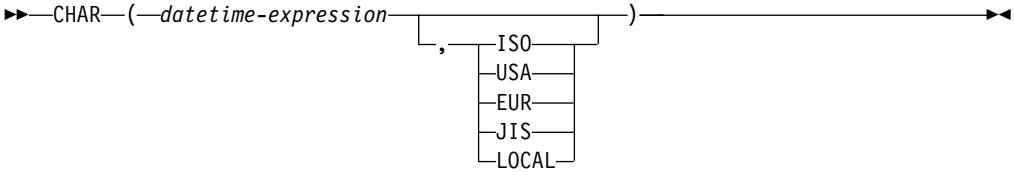
文字から文字に



グラフィックから文字



日付/時刻から文字に



CHAR 関数は、次の値の固定長文字ストリング表現を戻します。

- 整数 (最初の引数が SMALLINT、INTEGER、または BIGINT の場合)
- 10 進数 (最初の引数が 10 進数の場合)
- 倍精度浮動小数点数 (最初の引数が DOUBLE または REAL の場合)

- 最初の引数が DECFLOAT である場合は 10 進浮動小数点数。
- 文字ストリング (最初の引数が任意のタイプの文字ストリングの場合)
- グラフィック・ストリング (最初の引数が任意のタイプのグラフィック・ストリングの場合)
- 日付値 (最初の引数が DATE の場合)
- 時刻値 (最初の引数が TIME の場合)
- タイム・スタンプ値 (最初の引数が TIMESTAMP)
- 行 ID 値 (最初の引数が ROWID の場合)

この関数の結果は、固定長文字ストリングになります。最初の引数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引数がヌルの場合は、結果は NULL 値になります。

整数から文字に

integer-expression
組み込み整数データ・タイプ (SMALLINT、INTEGER、または BIGINT) の値を戻す式。

結果は、引数を SQL 整数定数の形式で表した固定長文字ストリング表現です。結果は、引数の値を表す n 文字の有効数字から成ります。引数が負数の場合は、負符号が前に付きます。結果は左寄せにされます。

- 引数が短整数の場合は、

  結果の長さは 6 です。結果の文字数が 6 文字より少ない場合は、結果は右側がブランクで埋め込まれます。
- 引数が長整数の場合は、

  結果の長さは 11 です。結果の文字数が 11 文字より少ない場合は、結果は右側がブランクで埋め込まれます。
- 引数が 64 ビット整数の場合は、

  結果の長さは 20 です。結果の文字数が 20 文字より少ない場合は、結果は右側がブランクで埋め込まれます。

結果の CCSID は、現行サーバーのデフォルトの SBCS CCSID になります。

10 進数から文字に

decimal-expression
組み込み 10 進数データ・タイプ (DECIMAL または NUMERIC) の値を戻す式。精度や位取りを変えたい場合は、DECIMAL スカラー関数を使用して変更することができます。

decimal-character
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

結果は、引数を固定長の文字ストリングで表現したものになります。この結果には、1 文字の小数点文字と *p* 桁までの数字が含まれます。 *p* は *10* 進数式 の精度で、引数が負数の場合は負符号が先頭に付きます。先行ゼロは戻されません。後続ゼロは戻されます。*decimal-expression* の位取りがゼロの場合、小数点文字は戻されません。

結果の長さは 2+*p* です (*p* は *decimal-expression* の精度)。すなわち、正の値には、1 桁の末尾ブランクが常に含まれることになります。

結果の CCSID は、現行サーバーのデフォルトの SBCS CCSID になります。

浮動小数点数から文字に

浮動小数点数式
組み込み浮動小数点データ・タイプ (DOUBLE または REAL) の値を戻す式。

decimal-character
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

結果は、引数を浮動小数点定数の形式で表した固定長文字ストリング表現です。結果の長さは 24 です。引数が負数の場合は、結果の最初の文字は負符号になります。負符号でなければ、最初の文字は数字または *decimal-character* です。引数がゼロであると、結果は 0E0 になります。それ以外の場合、結果は、ゼロ以外の 1 桁の数字および *decimal-character* の後に一連の数字で小数部が構成されるように、引数の値を表すために使用できる最小の文字数になります。

結果の文字数が 24 文字より少ない場合は、結果は右側がブランクで埋め込まれます。

結果の CCSID は、現行サーバーのデフォルトの SBCS CCSID になります。

10 進浮動小数点数から文字に

decimal-floating-point expression
組み込みの 10 進浮動小数点データ・タイプ (DECFLOAT) の値を返す式。

decimal-character
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

結果は、引数を 10 進浮動小数点定数の形式で表した固定長文字ストリング表現です。

結果の長さ属性は、42 です。 結果の実際の長さは、記号、数字、および *decimal-character* を含む、引数の値を表す最小文字数です。後続ゼロは有効数字です。引数が負数の場合は、結果の最初の文字は負符号になります。負数でなければ、最初の文字は数字または *decimal-character* です。引数がゼロであると、結果は 0 になります。

DECFLOAT 値が Infinity、sNaN、または NaN の場合、それぞれストリング 'INFINITY'、'SNAN'、および 'NAN' が返されます。特殊値が負の場合、負符号 (-) がそのストリングの最初の文字となります。 DECFLOAT 特殊値 sNaN を使用しても、ストリングへの変換時に例外は発生しません。

結果の文字数が 42 文字より少ない場合、結果は右側がブランクで埋め込まれます。

結果の CCSID は、現行サーバーのデフォルトの SBCS CCSID になります。

文字から文字に

character-expression
組み込み文字ストリング・データ・タイプの値を返す式。[52]

length
結果の固定長文字ストリングの長さ属性を指定する整数定数。値は 1 から 32766 (NULL でもよい場合は、32765) までででなければなりません。最初の引数が DBCS 専用混合データである場合は、2 番目の引数は 4 より小さくてはなりません。

2 番目の引数が指定されないか DEFAULT が指定された場合は、次のようになります。

- 文字式 が空ストリング定数の場合は、結果の長さ属性は 1。
- 空ストリング定数でない場合は、結果の長さ属性は、最初の引数の長さ属性と同じ。

実際の長さは、結果の長さ属性と同じになります。 *character-expression* の長さが結果の長さ属性より小さい場合は、結果は、結果に指定されている長さまでブランクで埋め込まれます。*character-expression* の長さが結果の長さ属性よりも大きい場合は、切り捨てが行われます。切り捨てられた文字がすべてブランクであった場合以外は、警告 (SQLSTATE 01004) が戻されます。

integer
結果の CCSID を指定する整数定数。これは有効な SBCS CCSID、混合データ CCSID、または 65535 (ビット・データ) とする必要があります。 3 番目の引数が SBCS CCSID の場合は、結果は SBCS データになります。 3 番目の引数が混合 CCSID の場合、結果は混合データです。結果の長さ属性が 4 より小さくてはなりません。 3 番目の引数が 65535 の場合、結果はビット・データです。 3 番目の引数が SBCS CCSID の場合は、最初の引数が DBCS 択一または DBCS 専用のストリングであることはありません。

3 番目の引数の指定がない場合は、次のようになります。

- 最初の引数が SBCS データであれば、結果は SBCS データになる。結果の CCSID は、最初の引数の CCSID と同一です。
- 最初の引数が混合データで、結果の長さ属性が 4 以上の場合、結果は混合データです。結果の CCSID は、最初の引数の CCSID と同一です。

52. CCSID が指定されていない場合、または CCSID として 65535 が明示的に指定されている場合、バイナリー・ストリングも使用できます。

- 最初の引数が DBCS 混用または DBCS 択一の混合データで、結果の長さ属性が 4 未満である場合、結果の CCSID は、混合データ CCSID に関連した SBCS CCSID です。

グラフィックから文字に

graphic-expression
組み込みグラフィック・ストリング・データ・タイプである値を戻す式。最初の引数は、DBCS グラフィック・データであってはなりません。

length
結果の固定長文字ストリングの長さ属性を指定する整数定数。値は 1 から 32766 (NULL でもよい場合は、32765) までででなければなりません。

2 番目の引数が指定されないか DEFAULT が指定された場合は、次のようになります。

- グラフィック式 が空ストリング定数の場合は、結果の長さ属性は 1。
- 結果が SBCS データの場合、結果の長さは *n* です。
- 結果が混合データであれば、結果の長さは (2.5*(*n* -1)) + 4 になる。

実際の長さは、結果の長さ属性と同じになります。グラフィック式 の長さが結果の長さより小さい場合は、結果は、結果に指定されている長さまでブランクで埋め込まれます。グラフィック式 の長さが結果の長さ属性よりも大きい場合は、切り捨てが行われます。 切り捨てられた文字がすべてブランクであった場合以外は、警告 (SQLSTATE 01004) が戻されます。

integer
結果の CCSID を指定する整数定数。これは有効な SBCS CCSID または混合データ CCSID とする必要があります。 3 番目の引数が SBCS CCSID の場合は、結果は SBCS データになります。 3 番目の引数が混合 CCSID の場合、結果は混合データです。結果の長さ属性が 4 より小さくてはなりません。 3 番目の引数は 65535 であってはなりません。

3 番目の引数が指定されていない場合は、結果の CCSID は現行サーバーのデフォルト CCSID になります。デフォルト CCSID が混合データで、結果の長さ属性が 4 以上の場合、結果は混合データです。それ以外の場合、その結果は SBCS データとなります。

日付/時刻から文字に

datetime-expression
次の 3 つの組み込みデータ・タイプのいずれかである式。

日付　結果は、2 番目の引数によって指定された形式の日付の文字ストリング表現です。 2 番目の引数を指定しなかった場合は、デフォルトの日付形式が使用されます。形式として ISO、USA、 EUR、または JIS を指定すると、結果の長さは 10 になります。その他の場合は、結果の長さはデフォルトの日付形式の長さになります。詳しくは、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

時刻　結果は、2 番目の引数によって指定された形式の時刻の文字ストリング表現です。 2 番目の引数を指定しなかった場合は、デフォルトの時刻形式が使用されます。結果の長さは 8 です。詳しくは、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

timestamp
2 番目の引数は適用されないので、指定してはなりません。

結果は、タイム・スタンプの文字ストリング表現です。*datetime-expression* が TIMESTAMP(0) である場合、結果の長さは 19 になります。*datetime-expression* のデータ・タイプが TIMESTAMP(*n*) の場合、結果の長さは 20+*n* です。これ以外の場合、結果の長さは 26 となります。

結果の CCSID は、現行サーバーのデフォルトの SBCS CCSID になります。

ISO、**EUR**、**USA**、または **JIS**
結果の文字ストリングの日付形式または時刻形式を指定します。詳しくは、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

LOCAL
結果の文字ストリングの日付または時刻の形式を、現行サーバーのジョブの DATFMT、DATSEP、 TIMFMT、および TIMSEP 属性から取る必要があることを指定します。

注記

代替構文**:** 最初の引数がストリングで、長さ引数を指定する場合、アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、 218 ページの『CAST の指定』を参照してください。

例

- 列 PRSTDATE には、1988-12-25 に相当する内部値が入っていると想定します。日付形式は *MDY で、日付区切り記号はスラッシュ (/) です。

```
SELECT CHAR(PRSTDATE, USA)
  FROM PROJECT
```

結果として、'12/25/1988' の値が返されます。

```
SELECT CHAR(PRSTDATE)
  FROM PROJECT
```

結果として、'12/25/88' の値が戻されます。

- 列 STARTING には、17.12.30 に相当する内部値が入っており、ホスト変数 HOUR_DUR (DECIMAL(6,0)) は、050000 (つまり、5 時間) の値を持つ時刻期間であると想定します。

```
SELECT CHAR(STARTING, USA)
  FROM CL_SCHED
```

結果として、'5:12 PM'の値が戻されます。

```
SELECT CHAR(STARTING + :HOUR_DUR, JIS)
  FROM CL_SCHED
```

結果として、'10:12:00'の値が戻されます。

- 列 RECEIVED (タイム・スタンプ) には、列 PRSTDATE と列 STARTING を合わせた値に相当する内部値が入っていると想定します。

```
SELECT CHAR(RECEIVED)
  FROM IN_TRAY
```

この結果、'1988-12-25-17.12.30.000000'の値が返されます。

- CHAR 関数を使用して、タイプを固定長文字にし、表示桁の長さを EMPLOYEE 表 (VARCHAR(15) と定義) の LASTNAME 列 の 10 文字までに減らすには、次のように指定します。

```
SELECT CHAR(LASTNAME,10)
  FROM EMPLOYEE
```

  LASTNAME を持つ行が 10 文字 (後書きブランクを除く) を超える場合は、値が切り捨てられるという警告 (SQLSTATE 01004) が出ます。

- CHAR 関数を使用して、EDLEVEL (SMALLINT と定義) の値を固定長ストリングとして戻します。

```
SELECT CHAR(EDLEVEL)
  FROM EMPLOYEE
```

  EDLEVEL の値が 18 の場合、CHAR(6) では値「18　　」(18 の後ろに 4 つのブランクが続く) が返されます。

- 20000.25 から減算されるのと同じ SALARY がコンマを小数点文字として戻されると想定します。

```
SELECT CHAR(20000.25 - SALARY, ',')
  FROM EMPLOYEE
```

  21150 の SALARY は、値「-1149,75 」(-1149,75 の後に 3 つのブランクを付けたもの) を返します。

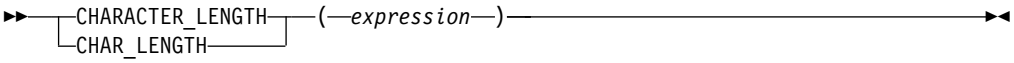
- ホスト変数 DOUBLE_NUM が倍精度浮動小数点データ・タイプで、値が -987.654321E-35 であるとします。

```
SELECT CHAR(:DOUBLE_NUM)
  FROM SYSIBM.SYSDUMMY1
```

  結果は、文字値「-9.8765432100000002E-33 」になります。

CHARACTER_LENGTH

CHARACTER_LENGTH または CHAR_LENGTH 関数は、ストリング式の長さを戻します。



同様な関数として、 522 ページの『LENGTH』を参照してください。

expression
任意の組み込み数値データ・タイプ、またはストリング・データ・タイプの値を戻す式。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。

この関数の結果は長精度整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

式 が文字ストリングまたはグラフィック・ストリングの場合は、結果は、引数の文字数 (バイト数ではない) です。 1 文字は、SBCS、DBCS、またはマルチバイト文字です。 *expression* が 2 進ストリングの場合は、結果は、引数のバイト数です。ストリングの長さには、末尾ブランクまたは 16 進数のゼロも含まれます。可変長ストリングを指定した場合に戻る長さは、実際の長さであり、最大長ではありません。

例

- NAME は VARCHAR(128) の列で Unicode UTF-8 でエンコードされ、値 'Jürgen' を含むものと仮定します。

```
SELECT CHARACTER_LENGTH(NAME), LENGTH(NAME)
  FROM T1
  WHERE NAME = 'Jürgen'
```

CHARACTER_LENGTH には値 6、LENGTH には値 7 が戻ります。

CHR

CHR 関数は、引数で指定された ASCII コード値をもつ EBCDIC 文字を返します。expression が 0 であれば、結果はブランク文字になります。

expression
データ・タイプ BIGINT、INTEGER、または SMALLINT の値を戻す式。引数の値は 0 から 255 の範囲でなければなりません。そうでない場合、戻り値は NULL 値になります。この値は、ASCII CCSID 367 の文字のコード・ポイントとして解釈されます。

関数の結果は CHAR(1) です。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

結果の CCSID は、現行サーバーのデフォルトの SBCS CCSID になります。

例

- ASCII CCSID 367 コード・ポイント 65 に対応する EBCDIC 文字を返します。

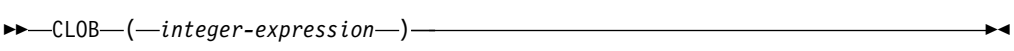
```
SELECT CHR(65)
    FROM SYSIBM.SYSDUMMY1
```

値 'A' が戻されます。

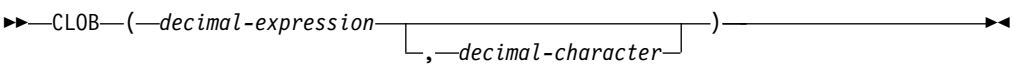
CLOB

CLOB 関数は、文字ストリング表現を返します。

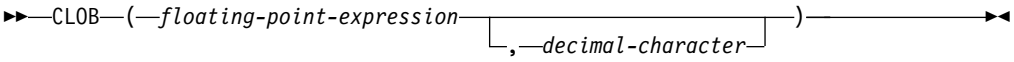
整数から **CLOB** に



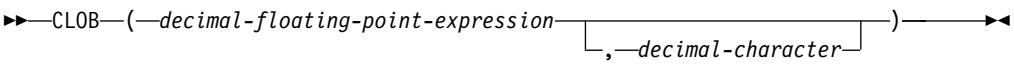
10 進数から **CLOB** に



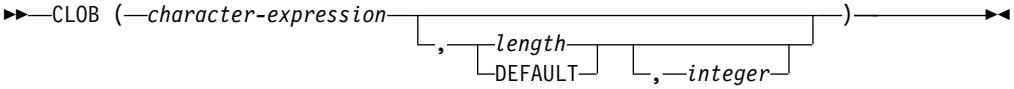
浮動小数点から **CLOB** に



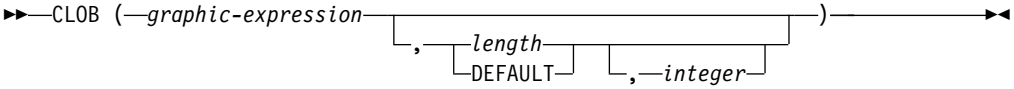
10 進浮動小数点から **CLOB** に



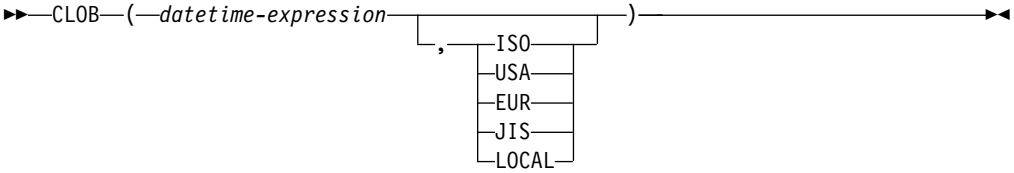
文字から **CLOB**



グラフィックから **CLOB**



日時から **CLOB** に



CLOB 関数は、次のものの文字ストリング表現を戻します。

- 整数 (最初の引数が SMALLINT、INTEGER、または BIGINT の場合)
- 10 進数 (最初の引数がパックまたはゾーン 10 進数の場合)
- 倍精度浮動小数点数 (最初の引数が DOUBLE または REAL の場合)

- 最初の引数が DECFLOAT である場合は 10 進浮動小数点数。
- 文字ストリング (最初の引数が任意のタイプの文字ストリングの場合)
- グラフィック・ストリング (最初の引数が Unicode グラフィック・ストリング・ストリングの場合)
- 日付値 (最初の引数が DATE の場合)
- 時刻値 (最初の引数が TIME の場合)
- タイム・スタンプ値 (最初の引数が TIMESTAMP)

この関数の結果は、CLOB になります。最初の引数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引数がヌルの場合は、結果は NULL 値になります。

整数から CLOB に

integer-expression
組み込み整数データ・タイプ (SMALLINT、INTEGER、または BIGINT) の値を戻す式。

結果は、SQL 整数定数の形式で引数を表現した可変長文字ストリングです。結果は、引数の値を表す n 文字の有効数字から成ります。引数が負数の場合は、負符号が前に付きます。結果は左寄せにされます。

- 引数が短整数の場合は、結果の長さ属性は 6
- 引数が長整数の場合は、結果の長さ属性は 11
- 引数が 64 ビット整数の場合は、結果の長さ属性は 20

結果の実際の長さは、引数の値を表すために使用できる最小文字数です。先行ゼロは含まれません。引数が負数の場合は、結果の最初の文字は負符号になります。負符号でなければ、最初の文字は数字または *decimal-character* です。

結果の CCSID は、現行サーバーのデフォルトの SBCS CCSID になります。

10 進数から CLOB に

decimal-expression
組み込み 10 進数データ・タイプ (DECIMAL または NUMERIC) の値を戻す式。精度や位取りを変えたい場合は、DECIMAL スカラー関数を使用して変更することができます。

decimal-character
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

結果は、引数を可変長文字ストリングで表現したものになります。この結果には、1 文字の小数点文字と *p* 桁までの数字が含まれます。 *p* は *10* 進数式 の精度で、引数が負数の場合は負符号が先頭に付きます。先行ゼロは戻されません。後続ゼロは戻されます。*decimal-expression* の位取りがゼロの場合、小数点文字は戻されません。

結果の長さ属性は 2+*p* です (*p* は *decimal-expression* の精度)。結果の実際の長さは、引数の値を表すために使用できる最小文字数ですが、ただし、後書き文字も含まれます。先行ゼロは含まれません。引数が負数の場合は、結果の最初の文字は負符号になります。負符号でなければ、結果の最初の文字は数字または *decimal-character* になります。

結果の CCSID は、現行サーバーのデフォルトの SBCS CCSID になります。

浮動小数点から CLOB に

浮動小数点数式
組み込み浮動小数点データ・タイプ (DOUBLE または REAL) の値を戻す式。

decimal-character
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

結果は、浮動小数点定数の形式で引数を可変長文字ストリングで表現したものになります。

結果の長さ属性は、24 です。結果の実際の長さは、ゼロ以外の 1 桁の数字、その後ろに 1 つの小数点文字 と一連の数字が続く小数部の引数の値を表す最小文字数です。引数が負数の場合は、結果の最初の文字は負符号になります。負数でなければ、最初の文字は数字または *decimal-character* です。引数がゼロであると、結果は 0E0 になります。

結果の CCSID は、現行サーバーのデフォルトの SBCS CCSID になります。

10 進浮動小数点から CLOB に

decimal floating-point expression
組み込みの 10 進浮動小数点データ・タイプの値を返す式。

decimal-character
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

結果は、10 進浮動小数点定数の形式で引数を可変長文字ストリングで表現したものになります。

結果の長さ属性は、42 です。 結果の実際の長さは、記号、数字、および *decimal-character* を含む、引数の値を表す最小文字数です。後続ゼロは有効数字です。引数が負数の場合は、結果の最初の文字は負符号になります。負数でなければ、最初の文字は数字または *decimal-character* です。引数がゼロであると、結果は 0 になります。

DECFLOAT 値が Infinity、sNaN、または NaN の場合、それぞれストリング 'INFINITY'、'SNAN'、および 'NAN' が返されます。特殊値が負の場合、負符号 (-)

がそのストリングの最初の文字となります。 DECFLOAT 特殊値 sNaN を使用しても、ストリングへの変換時に例外は発生しません。

結果の CCSID は、現行サーバーのデフォルトの SBCS CCSID になります。

文字から CLOB に

character-expression
組み込み文字ストリング・データ・タイプである値を戻す式。

length
結果の可変長文字ストリングの長さ属性を指定する整数定数。値は 1 から 2 147 483 647 でなければなりません。最初の引数が DBCS 専用混合データである場合は、2 番目の引数は 4 より小さくてはなりません。

2 番目の引数が指定されないか DEFAULT が指定された場合は、次のようになります。

- 文字式 が空ストリング定数の場合は、結果の長さ属性は 1。
- 空ストリング定数でない場合は、結果の長さ属性は、最初の引数の長さ属性と同じ。

結果の実際の長さは、結果の長さ属性と *character-expression* の実際の長さのうち、小さい方になります。*character-expression* の長さが結果の長さ属性よりも大きい場合は、切り捨てが行われます。切り捨てられた文字がすべてブランクであった場合以外は、警告 (SQLSTATE 01004) が戻されます。

integer
結果の CCSID を指定する整数定数。これは有効な SBCS CCSID または混合データ CCSID とする必要があります。 3 番目の引数が SBCS CCSID の場合は、結果は SBCS データになります。 3 番目の引数が混合 CCSID の場合、結果は混合データです。結果の長さ属性が 4 より小さくてはなりません。 3 番目の引数が SBCS CCSID の場合、最初の引数は DBCS 択一および DBCS 専用のストリングであってはなりません。 3 番目の引数を 65535 とすることはできません。

3 番目の引数の指定がない場合は、最初の引数の CCSID は 65535 であってはなりません。

- 最初の引数がビット・データの場合は、エラーが戻されます。
- 最初の引数が SBCS データであれば、結果は SBCS データになる。結果の CCSID は、最初の引数の CCSID と同一です。
- 最初の引数が混合データで、結果の長さが 4 以上の場合、結果は混合データです。結果の CCSID は、最初の引数の CCSID と同一です。
- 最初の引数が DBCS 混用または DBCS 択一の混合データで、結果の長さ属性が 4 未満である場合、結果の CCSID は、混合データ CCSID に関連した SBCS CCSID です。

グラフィックから CLOB に

graphic-expression
組み込みグラフィック・ストリング・データ・タイプである値を戻す式。最初の引数は、DBCS グラフィック・データであってはなりません。

length
結果の可変長文字ストリングの長さ属性を指定する整数定数。値は 1 から 2 147 483 647 でなければなりません。結果が混合データの場合は、2 番目の引数は 4 より小さくてはなりません。

2 番目の引数が指定されていないか、または DEFAULT が指定されている場合は、結果の長さ属性は、次のように決まります。(ただし、*n* は最初の引数の長さ属性です。)

- グラフィック式 が空グラフィック・ストリング定数の場合は、結果の長さ属性は 1 になる。
- 結果が SBCS データの場合、結果の長さは *n* です。
- 結果が混合データであれば、結果の長さは (2.5*(*n* -1)) + 4 になる。

結果の実際の長さは、結果の長さ属性と *graphic-expression* の実際の長さのうち、小さい方になります。グラフィック式 の長さが結果の長さ属性よりも大きい場合は、切り捨てが行われます。 切り捨てられた文字がすべてブランクであった場合以外は、警告 (SQLSTATE 01004) が戻されます。

integer
結果の CCSID を指定する整数定数。これは有効な SBCS CCSID または混合データ CCSID とする必要があります。 3 番目の引数が SBCS CCSID の場合は、結果は SBCS データになります。 3 番目の引数が混合 CCSID の場合、結果は混合データです。結果の長さ属性が 4 より小さくてはなりません。 3 番目の引数は 65535 であってはなりません。

3 番目の引数が指定されていない場合は、結果の CCSID は現行サーバーのデフォルト CCSID になります。デフォルト CCSID が混合データで、結果の長さ属性が 4 以上の場合、結果は混合データです。それ以外の場合、その結果は SBCS データとなります。

日時から CLOB に

datetime-expression
次の 3 つの組み込みデータ・タイプのいずれかである式。

日付　結果は、2 番目の引数によって指定された形式の日付の可変長文字文字ストリング表記になります。 2 番目の引数を指定しなかった場合は、デフォルトの日付形式が使用されます。形式として ISO、USA、EUR、または JIS を指定すると、結果の長さは 10 になります。その他の場合は、結果の長さはデフォルトの日付形式の長さになります。詳しくは、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

時刻　結果は、2 番目の引数によって指定された形式の時刻の可変長文字文字ストリング表記になります。 2 番目の引数を指定しなかった場合は、デフォルトの時刻形式が使用されます。結果の長さは 8 です。詳しくは、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

timestamp
2 番目の引数は適用されないので、指定してはなりません。

結果は、そのタイム・スタンプを可変長文字ストリング表現したものとなります。*datetime-expression* が TIMESTAMP(0) である場合、結果の長さは 19 になります。*datetime-expression* のデータ・タイプが TIMESTAMP(*n*) の場合、結果の長さは 20+*n* です。これ以外の場合、結果の長さは 26 となります。

結果の CCSID は、現行サーバーのデフォルトの SBCS CCSID になります。

ISO、EUR、USA、または JIS
結果の文字ストリングの日付形式または時刻形式を指定します。詳しくは、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

LOCAL
結果の文字ストリングの日付または時刻の形式を、現行サーバーのジョブの DATFMT、DATSEP、 TIMFMT、および TIMSEP 属性から取る必要があることを指定します。

注記

代替構文: 最初の引数がストリングで、長さ属性を指定する場合、アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、 218 ページの『CAST の指定』を参照してください。

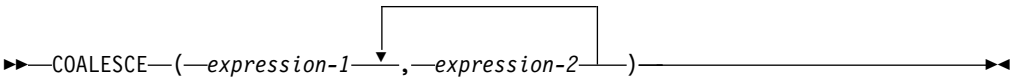
例

- 次の関数は、ストリング「This is a CLOB」の CLOB を戻します。

```
SELECT CLOB('This is a CLOB')
  FROM SYSIBM.SYSDUMMY1
```

COALESCE

COALESCE 関数は、NULL でない最初の式の値を戻します。



各引数には、互換性がなければなりません。文字ストリングの引数は、日付/時刻の値と互換性があります。データ・タイプの互換性についての詳細は、 113 ページの『割り当ておよび比較』を参照してください。

expression-1
任意の組み込みデータ・タイプまたはユーザー定義データ・タイプの値を戻す式。[53]

expression-2
任意の組み込みデータ・タイプまたはユーザー定義データ・タイプの値を戻す式。[53]

引数は、指定されている順序にしたがって評価され、NULL でない最初の引数がこの関数の結果となります。結果が NULL になる可能性があるのは、指定した引数がどれも NULL になる可能性がある場合だけです。また、結果が実際に NULL になるのは、すべての引数が NULL だった場合だけです。

選択された引数は、必要があれば、結果の属性に変換されます。結果の属性は、133 ページの『結果のデータ・タイプに関する規則』で説明しているすべてのオペランドを基にして決められます。

注記

代替構文: VALUE は COALESCE の同義語です。

例

- DEPARTMENT 表のすべての行のすべての値を選択する場合に、部門の管理者 (MGRNO) が欠落しているときには (つまり NULL 値なら)、 'ABSENT' という値を戻すようにします。

```
SELECT DEPTNO, DEPTNAME, COALESCE(MGRNO, 'ABSENT'), ADMRDEPT
  FROM DEPARTMENT
```

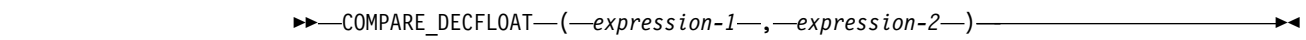
- 表 EMPLOYEE のすべての行から、従業員番号 (EMPNO) および給与 (SALARY) を選択するときに、給与が欠落しているもの (つまり、NULL のもの) については、値としてゼロを戻します。

```
SELECT EMPNO, COALESCE(SALARY,0)
  FROM EMPLOYEE
```

53. この関数は、ユーザー定義の関数を作成するときに、ソース関数として使用することはできません。これは、引数としてどのような互換データ・タイプでも受け入れるので、特殊タイプをサポートするための追加のシグニチャーを作成する必要はありません。

COMPARE_DECFLOAT

COMPARE_DECFLOAT 関数は DECFLOAT 値の順序付けを返します。



COMPARE_DECFLOAT 関数は、*expression-1* と *expression-2* の比較方法を示す短精度整数値を戻します。

expression-1
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数が DECFLOAT(34) でない場合、その引数は処理のために DECFLOAT(34) に論理的に変換されます。

expression-2
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数が DECFLOAT(34) でない場合、その引数は処理のために DECFLOAT(34) に論理的に変換されます。

最初の引数が 2 番目の引数と比較され、結果は以下の規則に従って返されます。

- 両方のオペランドが有限の場合、比較は代数の場合と同様に行われ、DECFLOAT の減算の規則に従います。差がいずれかの符号を持つゼロで、小数点以下のゼロの数が同じである場合、それらの引数は等しくなります。差がゼロ以外で正の場合、最初の引数は 2 番目の引数より大きくなります。差がゼロ以外で負の場合、最初の引数は 2 番目の引数より小さくなります。
- 正のゼロおよび負のゼロは、比較により等しいと判断されます。
- 正の無限大は、比較により負の無限大と等しくなります。
- 正の無限大は、比較によりどの有限数より大きくなります。
- 負の無限大は、比較により負の無限大と等しくなります。
- 負の無限大は、比較によりどの有限数より小さくなります。
- 数値の比較は正確です。有限オペランドの結果は、範囲と精度が無制限の場合と同様に求められます。オーバーフローまたはアンダーフローが起こってはなりません。
- いずれかの引数が NaN または sNaN (正または負) の場合、結果は順序付け不能となります。

結果の値は次のように設定されます。

0	引数が正確に等しい場合。
1	*expression-1* が *expression-2* より小さい場合。
2	*expression-1* が *expression-2* より大きい場合。
3	引数が順序付け不能の場合。

関数の結果は SMALLINT になります。引数のどちらかが NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数のどちらかが NULL である場合は、結果は NULL 値になります。

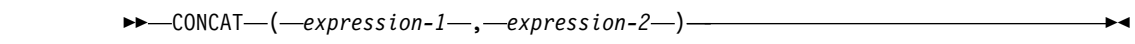
例

次の例に、この関数が使用されたときに返される値を示します。

```
COMPARE_DECFLOAT (DECFLOAT(2.17),  DECFLOAT(2.17))   =  0
COMPARE_DECFLOAT (DECFLOAT(2.17),  DECFLOAT(2.170))  =  2
COMPARE_DECFLOAT (DECFLOAT(2.170), DECFLOAT(2.17))   =  1
COMPARE_DECFLOAT (DECFLOAT(2.17),  DECFLOAT(0.0))    =  2
COMPARE_DECFLOAT (INFINITY,        INFINITY)         =  0
COMPARE_DECFLOAT (INFINITY,        -INFINITY)        =  2
COMPARE_DECFLOAT (DECFLOAT(-2),    INFINITY)         =  1
COMPARE_DECFLOAT (NAN,             NAN)              =  3
COMPARE_DECFLOAT (DECFLOAT(-0.1),  SNAN)             =  3
```

CONCAT

CONCAT 関数は、2 つの引数を結合します。



各引数には、互換性がなければなりません。データ・タイプの互換性についての詳細は、 113 ページの『割り当ておよび比較』を参照してください。

expression-1
任意の組み込み数値データ・タイプ、日時データ・タイプ、またはストリング・データ・タイプの値を戻す式。数値引数または日時引数は、関数を評価する前に文字ストリングにキャストされます。数値または日時から文字ストリングへの変換について詳しくは、 672 ページの『VARCHAR』 を参照してください。

expression-2
任意の組み込み数値データ・タイプ、日時データ・タイプ、またはストリング・データ・タイプの値を戻す式。数値引数または日時引数は、関数を評価する前に文字ストリングにキャストされます。数値または日時から文字ストリングへの変換について詳しくは、 672 ページの『VARCHAR』 を参照してください。

この関数の結果は、第 1 の引数ストリングの後に第 2 の引数ストリングを結合したストリングです。結果のデータ・タイプは、引数のデータ・タイプによって決まります。詳しくは、 202 ページの『連結演算子』を参照してください。引数のどちらかが NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数のどちらかが NULL である場合は、結果は NULL 値になります。

注記

代替構文: CONCAT 関数は CONCAT 演算子と同等です。詳しくは、 202 ページの『連結演算子』を参照してください。

例

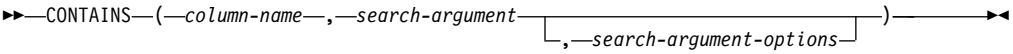
- 列 FIRSTNME と列 LASTNAME を連結します。

```
SELECT CONCAT(FIRSTNME, LASTNAME)
  FROM EMPLOYEE
  WHERE EMPNO ='000010'
```

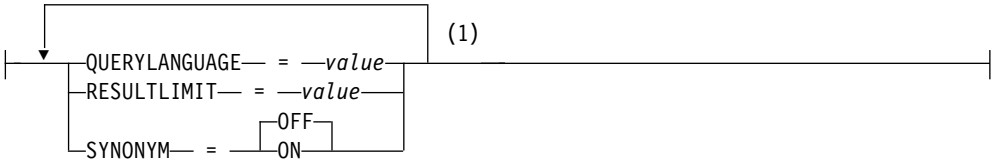
「CHRISTINEHAAS」の値が戻されます。

CONTAINS

CONTAINS 関数は、検索引数で指定した基準を使用してテキスト検索索引を検索し、一致項目が見つかったかどうかについての結果を戻します。



search-argument-options:



注:

1 同じ文節を複数回指定することはできません。

column-name
検索するテキスト検索索引がある列の修飾名または非修飾名を指定します。この列は、ステートメントの FROM 文節で指定されている表またはビューになければならず、表の列、またはビューの基礎となる基本表の列には関連付けられたテキスト検索索引がなければなりません。ビューの列の基礎となる式は、基礎表の列を、直接または他のネストされたビューを介して参照する単純な列参照であることが必要です。

search-argument
検索する用語が含まれる、文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプを戻す式です。空ストリングであったり、すべてを空白にしたりすることはできません。このストリングの実際の長さが Unicode への変換後に 32,740 バイトを超えてはならず、検索引数構文で指定されたテキスト検索限界や用語数を超えてもなりません。*search-argument* 構文については、 2187 ページの『付録 G. テキスト検索引数の構文』を参照してください。

search-argument-options
検索に使用する検索引数オプションが含まれている文字ストリング値またはグラフィック・ストリング値。定数または変数でなければなりません。

search-argument-options の一部として指定できるオプションは、以下のとおりです。

QUERYLANGUAGE = *value*
言語値を指定します。この値は、サポートされている任意の言語コードにすることできます。QUERYLANGUAGE を指定しないと、この関数の呼び出し時に使用されるテキスト検索索引の言語値がデフォルトとなります。テキスト検索索引の言語値が AUTO の場合、QUERYLANGUAGE のデフォルト値は en_US です。照会言語オプションについては、 2197 ページの『テキスト検索の言語オプション』を参照してください。

RESULTLIMIT = *value*
基礎となる検索エンジンから戻される結果の最大数を指定します。*value* は、1 から 2,147,483,647 の整数でなければなりません。RESULTLIMIT を指定しないと、照会での結果に制限がなくなります。

オプティマイザーが選択するプランによって、結果表の各行に対して CONTAINS が呼び出されることもあれば、呼び出されない場合もあります。基礎となる検索エンジンに対する照会で CONTAINS がいったん呼び出されると、検索エンジンから、一致する ROWID または主キーすべての結果セットが戻されます。その後、この結果セットはそうした結果行を識別する列が含まれる表に結合されます。この場合、RESULTLIMIT 値は基礎となるテキスト検索エンジンからの FETCH FIRST *n* ROWS ONLY のような動作になり、最適化のために使用できます。オプティマイザーが最適なプランであると判断し、CONTAINS が結果の各行に対して呼び出される場合には、RESULTLIMIT オプションは無効です。

SYNONYM = OFF または **SYNONYNM = ON**
テキスト検索索引に関連付けられている同義語ディクショナリーを使用するかどうかを指定します。デフォルトは OFF です。

オフ
同義語ディクショナリーを使用しません。

ON テキスト検索索引に関連付けられている同義語ディクショナリーを使用します。

search-argument-options が空ストリングまたは NULL 値の場合、この関数は *search-argument-options* が指定されていない場合のように評価されます。

この関数の結果は長精度整数になります。 *search-argument* を NULL にできる場合には、結果が NULL になる可能性があります。*search-argument* が NULL の場合、結果も NULL 値になります。

この列に、*search-argument* で指定された検索基準と一致する項目が含まれている場合には、結果は 1 になります。それ以外の場合、結果は 0 です。この列に NULL 値が含まれている場合、結果は 0 です。

CONTAINS は非決定性関数です。

注

前提条件**:** CONTAINS および SCORE 関数を使用するためには、OmniFind Text Search Server for Db2 for i がインストールされ、開始されている必要があります。

規則**:** ビュー、ネストされた表の式、または共通表式で、スカラー関数 CONTAINS または SCORE の対象になるテキスト検索列を用意する場合に、その該当するビュー、ネストされた表の式、共通表式の最外部の SELECT ステートメントで DISTINCT 文節を使用するのであれば、SELECT リストに、テキスト検索索引のすべての対応するキー・フィールドを組み込む必要があります。

ビュー、ネストされた表の式、または共通表式で、スカラー関数 CONTAINS または SCORE の対象になるテキスト検索列を用意する場合は、その該当するビュー、

ネストされた表の式、または共通表式の最外部の SELECT で、
UNION、EXCEPT、INTERSECT を使用できません。

共通表式で、スカラー関数 CONTAINS または SCORE の対象になるテキスト検索列を用意する場合に、その後、その共通表式を照会全体の中で再度参照できるのは、その参照によって、スカラー関数 CONTAINS または SCORE の対象になるテキスト検索列を用意しない場合に限られます。

CONTAINS および SCORE スカラー関数は、照会が以下を指定する場合には使用できません。

- 分散表
- 読み取りトリガーを指定する表
- 複数の物理ファイル・メンバー上に構築された論理ファイル

例

- 以下のステートメントは、履歴書に「COBOL」が含まれる従業員すべてを検出します。このテキスト検索引数には、大/小文字の区別はありません。

```
SELECT EMPNO
 FROM EMP_RESUME
 WHERE RESUME_FORMAT = 'ascii'
    AND CONTAINS(RESUME, 'cobol') = 1
```

- オンライン論文にスペイン語の「fossil fuel (combustible fósil)」という句が含まれる生徒をランダムに 10 人検出し、ラジオ・インタビューに招待します。任意の 10 人の生徒を選択できるので、検索の結果数を制限する RESULTLIMIT を使用して、照会を最適化します。

```
SELECT FIRSTNME, LASTNAME
  FROM STUDENT_ESSAYS
  WHERE CONTAINS(TERM_PAPER, 'combustible fósil',
                 'QUERYLANGUAGE = es_ES RESULTLIMIT = 10 SYNONYM = ON') = 1
```

- COMMENT 列で「ate」というストリングを検出します。検索引数を提供するためにホスト変数を使用します。

```
char search_arg[100];
...
EXEC SQL DECLARE C1 CURSOR FOR
  SELECT CUSTKEY
    FROM CUSTOMERS
    WHERE CONTAINS(COMMENT, :search_arg) = 1
    ORDER BY CUSTKEY;
strcpy(search_arg, "ate");
EXEC SQL OPEN C1;
```

COS

COS 関数は、引数のコサイン (余弦) を戻すもので、引数はラジアンで表された角度です。 COS 関数と ACOS 関数は、逆の演算になります。

expression
任意の組み込み数値データ・タイプ (DECFLOAT を除く)、文字ストリングまたはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

- ホスト変数 COSINE は、値が 1.5 の DECIMAL(2,1) のホスト変数であると想定します。

```
SELECT COS(:COSINE)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 0.07 の値が戻されます。

COSH

COSH 関数は、引数の双曲線コサイン (双曲線余弦) を戻すもので、引数はラジアンで表された角度です。

expression
任意の組み込み数値データ・タイプ (DECFLOAT を除く)、文字ストリングまたはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

- ホスト変数 HCOS は、値が 1.5 の DECIMAL(2,1) のホスト変数であると想定します。

```
SELECT COSH(:HCOS)
  FROM SYSIBM.SYSDUMMY1
```

およそ 2.35 の値が戻されます。

COT

COT 関数は引数のコタンジェント (余接) を戻すもので、引数はラジアンで表された角度です。

expression
任意の組み込み数値データ・タイプ (DECFLOAT を除く)、文字ストリングまたはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

- ホスト変数 COTAN は、値が 1.5 の DECIMAL(2,1) のホスト変数であると想定します。

```
SELECT COT(:COTAN)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 0.07 の値が戻されます。

CURDATE

CURDATE 関数は、SQL ステートメントが現行サーバーで実行される時点の刻時機構の読み取りに基づく日付を戻します。 CURDATE 関数によって戻される値は、CURRENT DATE 特殊レジスターによって戻される値と同じです。

結果のデータ・タイプは日付になります。結果が NULL になることはありません。

この関数が 1 つの SQL ステートメント内で複数回使用される場合、または 1 つのステートメント内で CURTIME または NOW スカラー関数、あるいは CURRENT_DATE、CURRENT_TIME、または CURRENT_TIMESTAMP 特殊レジスターとともに使用される場合は、値はすべて 1 回の刻時機構読み取りに基づきます。

注記

代替構文: CURRENT_DATE 特殊レジスターを使用して移植性を最大限に引き出す必要があります。詳しくは、 149 ページの『特殊レジスター』を参照してください。

例

- 刻時機構に基づく現在日付が戻されます。

```
SELECT CURDATE()
  FROM SYSIBM.SYSDUMMY1
```

CURTIME

CURTIME 関数は、SQL ステートメントが現行サーバーで実行される時点の刻時機構の読み取りに基づく時刻を戻します。 CURTIME 関数によって戻される値は、CURRENT TIME 特殊レジスターによって戻される値と同じです。

結果のデータ・タイプは時刻です。結果が NULL になることはありません。

この関数が 1 つの SQL ステートメント内で複数回使用される場合、または 1 つのステートメント内で CURDATE または NOW スカラー関数、あるいは CURRENT_DATE、CURRENT_TIME、または CURRENT_TIMESTAMP 特殊レジスターとともに使用される場合は、値はすべて 1 回の刻時機構読み取りに基づきます。

注記

代替構文: CURRENT_TIME 特殊レジスターを使用して移植性を最大限に引き出す必要があります。詳しくは、 149 ページの『特殊レジスター』を参照してください。

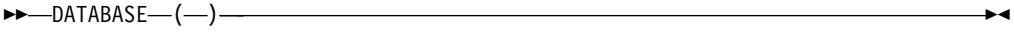
例

- 刻時機構に基づく現在時刻が戻されます。

```
SELECT CURTIME()
  FROM SYSIBM.SYSDUMMY1
```

DATABASE

DATABASE 関数は、現行サーバーを戻します。



関数の結果は VARCHAR(18) になります。結果が NULL になることはありません。

ストリングの CCSID は、現行サーバーにおけるデフォルト SBCS CCSID です。

注記

代替構文: DATABASE 関数は、CURRENT SERVER 特殊レジスターと同じ結果を戻します。

例

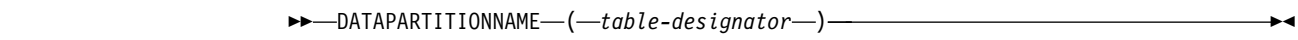
- 現行サーバーが「RCHASGMA」であると想定します。

```
SELECT DATABASE( )
  FROM SYSIBM.SYSDUMMY1
```

この結果、'RCHASGMA' の値が戻されます。

DATAPARTITIONNAME

DATAPARTITIONNAME 関数は、行が置かれているパーティションの名前を戻します。引数がパーティション化されていない表を示している場合は、空ストリングが戻されます。



パーティションの詳細については、「Db2 マルチシステム」トピック集を参照してください。

table-designator

副選択の表指定子。表指定子の詳細については、 166 ページの『表指定子』を参照してください。

SQL 命名規則では、表名は修飾できます。システム命名規則では、表名は修飾できません。

table-designator は、 *collection-derived-table*、VALUES 節、*table-function*、または *data-change-table-reference* を指定するものであってはなりません。

引数がビュー、共通表式、またはネストされた表式を示している場合、この関数はその基本表のパーティション名を返します。引数が、複数の基本表から派生したビュー、共通表式、またはネストされた表式を示している場合、この関数は、そのビュー、共通表式、またはネストされた表式の外側の副選択内にある最初の表のパーティション名を返します。

引数は、外側の全選択に、集約関数、GROUP BY 節、HAVING 節、UNION、INTERSECT、または EXCEPT 節、DISTINCT 節、VALUES 節、または *table-function* が含まれている、ビュー、共通表式、またはネストされた表式を示していてはなりません。全選択が集約関数、GROUP BY 文節、または HAVING 文節を含む場合、SELECT 文節に DATAPARTITIONNAME 関数を指定することはできません。

この結果のデータ・タイプは、VARCHAR(18) です。結果は NULL になる場合があります。

結果の CCSID は、現行サーバーのデフォルトの CCSID になります。

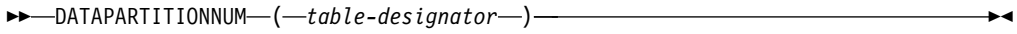
例

- EMPLOYEE 表と DEPARTMENT 表を結合し、社員番号 (EMPNO) を選択して、発生した結合に関連する各行からパーティションを判別します。

```
SELECT EMPNO, DATAPARTITIONNAME(X), DATAPARTITIONNAME(Y)
  FROM EMPLOYEE X, DEPARTMENT Y
  WHERE X.DEPTNO=Y.DEPTNO
```

DATAPARTITIONNUM

DATAPARTITIONNUM 関数は、行のデータ・パーティション番号を戻します。引数がパーティション化されていない表を示している場合は、値 0 が戻されます。



データ・パーティションの詳細については、「Db2 マルチシステム」トピック集を参照してください。

table-designator
副選択の表指定子。表指定子の詳細については、 166 ページの『表指定子』を参照してください。

SQL 命名規則では、表名は修飾できます。システム命名規則では、表名は修飾できません。

table-designator は、 *collection-derived-table*、VALUES 節、*table-function*、または *data-change-table-reference* を指定するものであってはなりません。

引数がビュー、共通表式、またはネストされた表式を示している場合、この関数はその基本表のデータ・パーティション番号を返します。引数が、複数の基本表から派生したビュー、共通表式、またはネストされた表式を示している場合、この関数は、そのビュー、共通表式、またはネストされた表式の外側の副選択内にある最初の表のデータ・パーティション番号を返します。

引数は、外側の全選択の副選択に、集約関数、GROUP BY 節、HAVING 節、UNION、INTERSECT、または EXCEPT 節、DISTINCT 節、VALUES 節、または *table-function* が含まれている、ビュー、共通表式、またはネストされた表式を示していてはなりません。全選択が集約関数、GROUP BY 文節、または HAVING 文節を含む場合、SELECT 文節に DATAPARTITIONNUM 関数を指定することはできません。

結果のデータ・タイプは、長整数です。結果は NULL になる場合があります。

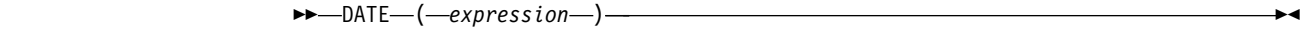
例

- 表 EMPLOYEE の各行についてのパーティション番号と従業員名を判別します。パーティション化された表の場合は、その行が存在するノードの番号が戻されます。

```
SELECT DATAPARTITIONNUM(EMPLOYEE), LASTNAME
  FROM EMPLOYEE
```

DATE

DATE 関数は、指定された値に基づく日付を戻します。



expression
日付、タイム・スタンプ、文字ストリング、グラフィック・ストリング、または数値のいずれかの組み込みデータ・タイプの値を戻す式。

- *expression* が文字ストリングまたはグラフィック・ストリングの場合、値は以下のいずれかでなければなりません。
  - 日付またはタイム・スタンプの有効なストリング表現。日付とタイム・スタンプのストリング表現の有効な形式については、95 ページの『日付/時刻の値のストリング表記』を参照してください。
  - *yyyynnn* の形式で有効な日付を表す、実際長が 7 のストリング。 *yyyy* は年番号を表す数値で、*nnn* は年間通算日を表す 001 から 366 の数値です。
- 式 が数値である場合は、その数値は 3 652 059 以下の正の数でなければなりません。

関数の結果は、日付になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

その他の規則は、引数のデータ・タイプに応じて以下のように異なります。

- 引数がタイム・スタンプの場合 :

  結果はタイム・スタンプの日付の部分になります。
- 引数が日付の場合 :

  結果は、指定された日付になります。
- 引数が数値の場合 :

  結果は、0001 年 1 月 1 日の *n*-1 日後の日付になります (*n* は、指定した数値の整数部です)。
- 引数が文字ストリングまたはグラフィック・ストリングの場合 :

  結果は、ストリングが示す日付か、ストリングが示すタイム・スタンプの日付部分です。

注記

代替構文**:** 引数が日付、タイム・スタンプ、または文字ストリングの場合、アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、218 ページの『CAST の指定』を参照してください。

例

- 列 RECEIVED (TIMESTAMP) には、'1988-12-25-17.12.30.000000' に相当する内部値が入っているものと想定します。

```
SELECT DATE(RECEIVED)
  FROM IN_TRAY
  WHERE SOURCE = 'BADAMSON'
```

結果は '1988-12-25'という値の日付データ・タイプになります。

- 次の例では、DATE スカラー関数が日付の ISO ストリング表現に適用されています。

```
SELECT DATE('1988-12-25')
  FROM SYSIBM.SYSDUMMY1
```

結果は '1988-12-25'という値の日付データ・タイプになります。

- 次の例では、DATE スカラー関数が日付の EUR ストリング表現に適用されています。

```
SELECT DATE('25.12.1988')
  FROM SYSIBM.SYSDUMMY1
```

結果は '1988-12-25'という値の日付データ・タイプになります。

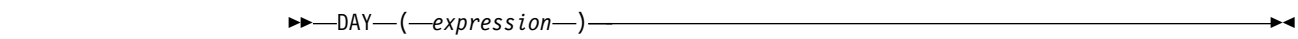
- 次の例では、DATE スカラー関数が正の整数に適用されています。

```
SELECT DATE(35)
  FROM SYSIBM.SYSDUMMY1
```

結果は '0001-02-04'という値の日付データ・タイプになります。

DAY

DAY 関数は、指定した値の日の部分を戻します。



expression
日付、タイム・スタンプ、文字ストリング、グラフィック・ストリング、または数値のいずれかの組み込みデータ・タイプの値を戻す式。

- *expression* が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。
- 式 が数値である場合は、その数値は日付期間またはタイム・スタンプ期間でなければなりません。日時期間の有効な形式については、 205 ページの『日付/時刻のオペランドと期間』を参照してください。

この関数の結果は長精度整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

その他の規則は、引数のデータ・タイプに応じて以下のように異なります。

- 引数が日付、タイム・スタンプ、または、日付またはタイム・スタンプの有効な文字ストリング表現である場合は、次のようになります。

  結果は、指定した値の日の部分 (1 から 31 までの整数) になります。
- 引数が日付期間またはタイム・スタンプ期間の場合 :

  結果は、指定した値の日の部分 (-99 から 99 までの整数) になります。ゼロ以外の結果の符号は、引数と同じになります。

例

- 表 PROJECT を使用して、WELD LINE PLANNING プロジェクト (PROJNAME) の停止が予定されている日付 (PRENDATE) の日の部分を END_DAY (SMALLINT) にセットします。

```
SELECT DAY(PRENDATE)
  INTO :END_DAY
  FROM PROJECT
  WHERE PROJNAME = 'WELD LINE PLANNING'
```

  結果として、END_DAY は 15 にセットされます。
- 2 つの日付間の差の日の部分を戻します。

```
SELECT DAY( DATE('2000-03-15') - DATE('1999-12-31') )
  FROM SYSIBM.SYSDUMMY1
```

  結果として、15 の値が戻されます。

DAYNAME

引数の日の部分の曜日の名前 (例えば、Friday) を含む大/小文字混合の文字ストリングを返します。

expression
日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式。

expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

この関数の結果は VARCHAR(100) になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

結果の CCSID は、現行サーバーのデフォルトの CCSID になります。

注記

各国語の考慮事項**:** 戻される曜日の名前は、ジョブのメッセージに使用される言語に基づいています。曜日の名前は、ライブラリー *LIBL の中のメッセージ・ファイル QCPFMSG のメッセージ CPX9034 から検索されます。

例

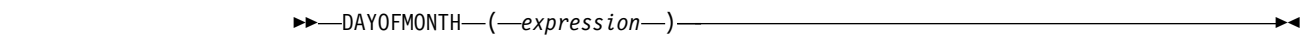
- 使用される言語が米国英語であると想定します。

```
SELECT DAYNAME( '2003-01-02' )
  FROM SYSIBM.SYSDUMMY1
```

結果は「Thursday」になります。

DAYOFMONTH

DAYOFMONTH 関数は、月の中の日付を表す 1 から 31 の整数を戻します。



expression
日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式。

expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

この関数の結果は長精度整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

- 表 PROJECT を使用して、WELD LINE PLANNING プロジェクト (PROJNAME) の停止が予定されている日付 (PRENDATE) の日の部分を END_DAY (SMALLINT) にセットします。

```
SELECT DAYOFMONTH(PRENDATE)
  INTO :END_DAY
  FROM PROJECT
  WHERE PROJNAME = 'WELD LINE PLANNING'
```

  結果として、END_DAY は 15 にセットされます。

DAYOFWEEK

DAYOFWEEK 関数は、曜日を表す 1 から 7 までの整数 (1 は日曜日を表し、7 は土曜日を表す) を戻します。

別の指定方法については、 411 ページの『DAYOFWEEK_ISO』を参照してください。

expression
日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式。

expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

この関数の結果は長精度整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

- 表 EMPLOYEE を使用して、Christine Haas (EMPNO='000010') の雇用が開始した曜日 (HIREDATE) にホスト変数 DAY_OF_WEEK (INTEGER) をセットします。

```
SELECT DAYOFWEEK(HIREDATE)
  INTO :DAY_OF_WEEK
  FROM EMPLOYEE
  WHERE EMPNO = '000010'
```

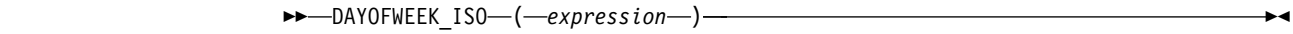
  DAY_OF_WEEK に 6 (金曜日を表す) がセットされる結果になります。

- 次の照会は、4 つの値 (1、2、1、2) を戻します。

```
SELECT DAYOFWEEK(CAST('10/11/1998' AS DATE)),
       DAYOFWEEK(TIMESTAMP('10/12/1998','01.02')),
       DAYOFWEEK(CAST(CAST('10/11/1998' AS DATE) AS CHAR(20))),
       DAYOFWEEK(CAST(TIMESTAMP('10/12/1998','01.02') AS CHAR(26)))
   FROM SYSIBM.SYSDUMMY1
```

DAYOFWEEK_ISO

DAYOFWEEK_ISO 関数は、曜日を表す 1 から 7 までの整数 (1 は月曜日を表し、7 は日曜日を表す) を戻します。



別の指定方法については、 410 ページの『DAYOFWEEK』を参照してください。

expression
日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式。

expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

この関数の結果は長精度整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

- 表 EMPLOYEE を使用して、Christine Haas (EMPNO='000010') の雇用が開始した曜日 (HIREDATE) にホスト変数 DAY_OF_WEEK (INTEGER) をセットします。

```
SELECT DAYOFWEEK_ISO(HIREDATE)
  INTO :DAY_OF_WEEK
  FROM EMPLOYEE
  WHERE EMPNO = '000010'
```

  DAY_OF_WEEK に 5 (金曜日を表す) がセットされる結果になります。

- 次の照会は、4 つの値、つまり 7、1、7、1 を戻します。

```
SELECT DAYOFWEEK_ISO(CAST('10/11/1998' AS DATE)),
       DAYOFWEEK_ISO(TIMESTAMP('10/12/1998','01.02')),
       DAYOFWEEK_ISO(CAST(CAST('10/11/1998' AS DATE) AS CHAR(20))),
       DAYOFWEEK_ISO(CAST(TIMESTAMP('10/12/1998','01.02') AS CHAR(26)))
  FROM SYSIBM.SYSDUMMY1
```

DAYOFYEAR

DAYOFYEAR 関数は、年間通算日を表す 1 から 366 までの整数 (1 は 1 月 1 日を表す) を戻します。

expression
日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式。

expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

この関数の結果は長精度整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

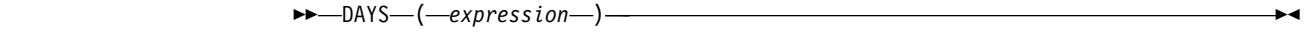
- 表 EMPLOYEE を使用して、従業員の雇用が開始された年間通算日 (HIREDATE) の平均をホスト変数 AVG_DAY_OF_YEAR (INTEGER) にセットします。

```
SELECT AVG(DAYOFYEAR(HIREDATE))
  INTO :AVG_DAY_OF_YEAR
  FROM EMPLOYEE
```

結果として、AVG_DAY_OF_YEAR は 197 にセットされます。

DAYS

DAYS 関数は、日付の整数表現を戻します。



expression
日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式。

expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

この関数の結果は長精度整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

結果は、0001 年 1 月 1 日から *D* までの日数に 1 を加えたものになります。*D* は、引数に DATE 関数を適用した場合に戻される日付です。

例

- 表 PROJECT を使用して、プロジェクト (PROJNO)'IF2000' の終了までに経過する予想日数 (PRENDATE - PRSTDATE) を、ホスト変数 EDUCATION_DAYS (INTEGER) にセットします。

```
SELECT DAYS(PRENDATE) - DAYS(PRSTDATE)
  INTO :EDUCATION_DAYS
  FROM PROJECT
  WHERE PROJNO = 'IF2000'
```

  結果として、EDUCATION_DAYS は 396 にセットされます。

- 表 PROJECT を使用して、部門 (DEPTNO)'E21' のすべてのプロジェクトの終了までに経過する予想日数 (PRENDATE- PRSTDATE) の合計を、ホスト変数 TOTAL_DAYS (INTEGER) にセットします。

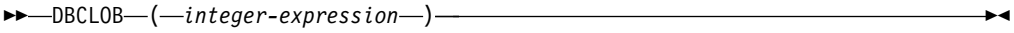
```
SELECT SUM(DAYS(PRENDATE) - DAYS(PRSTDATE))
  INTO :TOTAL_DAYS
  FROM PROJECT
  WHERE DEPTNO = 'E21'
```

  結果として、TOTAL_DAYS は 1584 にセットされます。

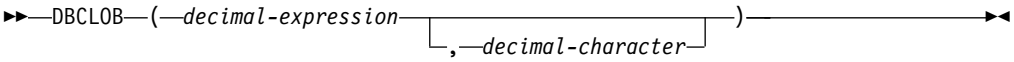
DBCLOB

DBCLOB 関数は、グラフィック・ストリング表現を返します。

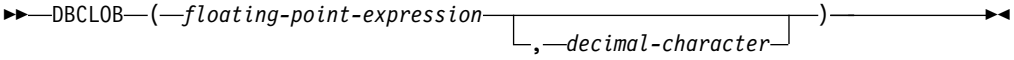
整数から **DBCLOB** に



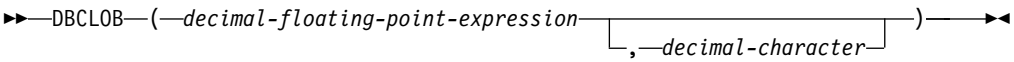
10 進数から **DBCLOB** に



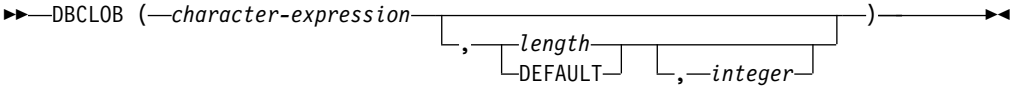
浮動小数点数から **DBCLOB** に



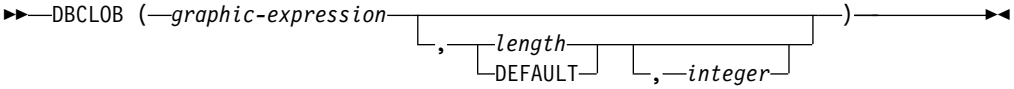
10 進浮動小数点数から **DBCLOB** に



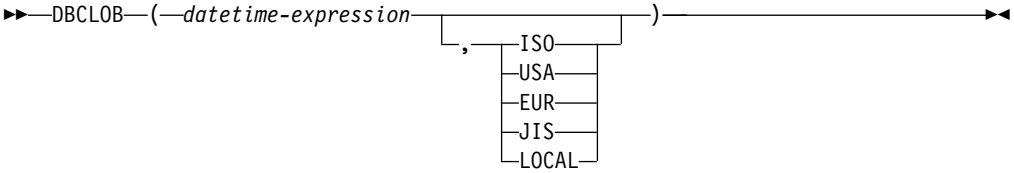
文字から **DBCLOB** に



グラフィックから **DBCLOB**



日時から **DBCLOB**



DBCLOB 関数は、次のもののグラフィック・ストリング表現を戻します。

- 整数 (最初の引数が SMALLINT、INTEGER、または BIGINT の場合)
- 10 進数 (最初の引数がパックまたはゾーン 10 進数の場合)
- 倍精度浮動小数点数 (最初の引数が DOUBLE または REAL の場合)

- 最初の引数が DECFLOAT である場合は 10 進浮動小数点数。
- 文字ストリング (最初の引数が任意のタイプの文字ストリングの場合)
- グラフィック・ストリング (最初の引数が任意のタイプのグラフィック・ストリングの場合)
- 日付値 (最初の引数が DATE の場合)
- 時刻値 (最初の引数が TIME の場合)
- タイム・スタンプ値 (最初の引数が TIMESTAMP)

この関数の結果は、DBCLOB になります。最初の引数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引数がヌルの場合は、結果は NULL 値になります。

整数から DBCLOB に

integer-expression
組み込み整数データ・タイプ (SMALLINT、INTEGER、または BIGINT) の値を戻す式。

結果は、SQL 整数定数の形式で引数を表現した可変長グラフィック・ストリングです。結果は、引数の値を表す n 文字の有効数字から成ります。引数が負数の場合は、負符号が前に付きます。結果は左寄せにされます。

- 引数が短整数の場合は、結果の長さ属性は 6
- 引数が長整数の場合は、結果の長さ属性は 11
- 引数が 64 ビット整数の場合は、結果の長さ属性は 20

結果の実際の長さは、引数の値を表すために使用できる最小文字数です。先行ゼロは含まれません。引数が負数の場合は、結果の最初の文字は負符号になります。負符号でなければ、最初の文字は数字または *decimal-character* です。

結果の CCSID は 1200 (UTF-16) です。

10 進数から DBCLOB に

decimal-expression
組み込み 10 進数データ・タイプ (DECIMAL または NUMERIC) の値を戻す式。精度や位取りを変えたい場合は、DECIMAL スカラー関数を使用して変更することができます。

decimal-character
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

結果は、引数を可変長グラフィック・ストリングで表現したものになります。この結果には、1 文字の小数点文字と *p* 桁までの数字が含まれます。 *p* は *10* 進数式の精度で、引数が負数の場合は負符号が先頭に付きます。先行ゼロは戻されません。後続ゼロは戻されます。*decimal-expression* の位取りがゼロの場合、小数点文字は戻されません。

結果の長さ属性は 2+*p* です (*p* は *decimal-expression* の精度)。結果の実際の長さは、引数の値を表すために使用できる最小文字数ですが、ただし、後書き文字も含まれます。先行ゼロは含まれません。引数が負数の場合は、結果の最初の文字は負符号になります。負符号でなければ、結果の最初の文字は数字または *decimal-character* になります。

結果の CCSID は 1200 (UTF-16) です。

浮動小数点数から DBCLOB に

浮動小数点数式
組み込み浮動小数点データ・タイプ (DOUBLE または REAL) の値を戻す式。

decimal-character
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

結果は、浮動小数点定数の形式で引数を可変長グラフィック・ストリングで表現したものになります。

結果の長さ属性は、24 です。結果の実際の長さは、ゼロ以外の 1 桁の数字、その後ろに 1 つの小数点文字 と一連の数字が続く小数部の引数の値を表す最小文字数です。引数が負数の場合は、結果の最初の文字は負符号になります。負数でなければ、最初の文字は数字または *decimal-character* です。引数がゼロであると、結果は 0E0 になります。

結果の CCSID は 1200 (UTF-16) です。

10 進浮動小数点数から DBCLOB に

decimal floating-point expression
組み込みの 10 進浮動小数点データ・タイプの値を返す式。

decimal-character
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

結果は、10 進浮動小数点定数の形式で引数を可変長グラフィック・ストリングで表現したものになります。

結果の長さ属性は、42 です。 結果の実際の長さは、記号、数字、および *decimal-character* を含む、引数の値を表す最小文字数です。後続ゼロは有効数字です。引数が負数の場合は、結果の最初の文字は負符号になります。負数でなければ、最初の文字は数字または *decimal-character* です。引数がゼロであると、結果は 0 になります。

DECFLOAT 値が Infinity、sNaN、または NaN の場合、それぞれストリング 'INFINITY'、'SNAN'、および 'NAN' が返されます。特殊値が負の場合、負符号 (-)

がそのストリングの最初の文字となります。 DECFLOAT 特殊値 sNaN を使用しても、ストリングへの変換時に例外は発生しません。

結果の CCSID は 1200 (UTF-16) です。

文字から DBCLOB に

character-expression
組み込み文字ストリング・データ・タイプである値を戻す式。 CHAR または VARCHAR ビット・データであってはなりません。式が空ストリング、または EBCDIC ストリング X'0E0F' である場合は、結果は空ストリングになります。

length
結果の可変長文字ストリングの長さ属性を指定する整数定数。値は 1 から 1 073 741 823 でなければなりません。

2 番目の引数が指定されないか DEFAULT が指定された場合は、次のようになります。

- 文字式 が空ストリング定数の場合は、結果の長さ属性は 1。
- 空ストリング定数でない場合は、結果の長さ属性は、最初の引数の長さ属性と同じ。

結果の実際の長さは、結果の長さ属性と *character-expression* の実際の長さのうち、小さい方になります。*character-expression* の長さが結果の長さ属性よりも大きい場合は、切り捨てが行われます。切り捨てられた文字がすべてブランクであった場合以外は、警告 (SQLSTATE 01004) が戻されます。

integer
結果の可変長グラフィック・ストリングの CCSID を指定する整数定数。これは DBCS、UTF-16、または UCS-2 の CCSID でなければなりません。 CCSID が 65535 であることはできません。

以下の規則では、S は次のいずれかを指します。

- ストリング式が外部コード化スキームのデータを含むホスト変数である場合は、データを固有コード化スキームの CCSID に変換した後の式の結果が S。 (詳しくは、 39 ページの『文字変換』を参照してください。)
- ストリング式が固有コード化スキームのデータである場合は、そのストリング式が S。

3 番目の引数の指定がなく、最初の引数が文字である場合は、結果の CCSID は混合 CCSID によって決まります。 M でその混合 CCSID を示すことにします。M は次のように決まります。

- S の CCSID が混合 CCSID である場合は、M はその CCSID になる。
- S の CCSID が SBCS CCSID である場合 :
  - S の CCSID が関連する混合 CCSID を持つ場合は、M はその CCSID になる。
  - それ以外の場合は、演算ができない。

次の表には、M をもとにした結果の CCSID を要約してあります。

M	結果の **CCSID**	説明	**DBCS** 置換文字
930	300	日本語 EBCDIC	X'FEFE'

M	結果の CCSID	説明	DBCS 置換文字
933	834	韓国語 EBCDIC	X'FEFE'
935	837	中国語 (簡体字) EBCDIC	X'FEFE'
937	835	中国語 (繁体字) EBCDIC	X'FEFE'
939	300	日本語 EBCDIC	X'FEFE'
5026	4396	日本語 EBCDIC	X'FEFE'
5035	4396	日本語 EBCDIC	X'FEFE'

結果が DBCS グラフィック・データの場合は、SBCS と DBCS が等価になるかどうかは M によって決まります。 CCSID に関係なく、引数の中の 2 バイトのコード・ポイントはすべて DBCS 文字と見なされ、引数の中の 1 バイトのコード・ポイントはすべて SBCS 文字と見なされます (ただし、EBCDIC 混合データのシフト・コード X'0E' および X'0F' は例外)。

- 引数の n 番目の文字が DBCS 文字である場合は、結果の n 番目の文字はその DBCS になる。
- 引数の n 番目の文字が、等価の DBCS 文字を持つ SBCS 文字である場合は、結果の n 番目の文字は、その等価の DBCS 文字になる。
- 引数の n 番目の文字が、等価の DBCS 文字を持たない SBCS 文字である場合は、結果の n 番目の文字は、DBCS 置換文字になる。

結果が Unicode グラフィック・データである場合は、引数の各文字ごとに結果の 1 文字が決まります。結果の n 番目の文字は、引数の n 番目の文字と等価の UTF-16 または UCS-2 文字になります。

グラフィックから DBCLOB に

graphic-expression
組み込みグラフィック・ストリング・データ・タイプである値を戻す式。

length
結果の可変長文字ストリングの長さ属性を指定する整数定数。値は 1 から 1 073 741 823 でなければなりません。

2 番目の引数が指定されないか DEFAULT が指定された場合は、次のようになります。

- グラフィック式 が空ストリング定数の場合は、結果の長さ属性は 1。
- 空ストリング定数でない場合は、結果の長さ属性は、最初の引数の長さ属性と同じ。

結果の実際の長さは、結果の長さ属性と *graphic-expression* の実際の長さのうち、小さい方になります。グラフィック式 の長さが結果の長さ属性よりも大きい場合は、切り捨てが行われます。 切り捨てられた文字がすべてブランクであった場合以外は、警告 (SQLSTATE 01004) が戻されます。

integer
結果の可変長グラフィック・ストリングの CCSID を指定する整数定数。これは DBCS、UTF-16、または UCS-2 の CCSID でなければなりません。 CCSID が 65535 であることはできません。

以下の規則では、S は次のいずれかを指します。

- ストリング式が外部コード化スキームのデータを含むホスト変数である場合は、データを固有コード化スキームの CCSID に変換した後の式の結果が S。 (詳しくは、 39 ページの『文字変換』を参照してください。)
- ストリング式が固有コード化スキームのデータである場合は、そのストリング式が S。

3 番目の引数の指定がない場合は、結果の CCSID は最初の引数の CCSID と同じになります。

日時から DBCLOB

datetime-expression
次の 3 つの組み込みデータ・タイプのいずれかである式。

日付　結果は、2 番目の引数によって指定された形式の日付の可変長 GRAPHIC ストリング表記になります。 2 番目の引数を指定しなかった場合は、デフォルトの日付形式が使用されます。形式として ISO、USA、 EUR、または JIS を指定すると、長さ属性と結果の実際の長さは 10 になります。その他の場合は、長さ属性と結果の実際の長さはデフォルトの日付形式の長さになります。詳しくは、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

時刻　結果は、2 番目の引数によって指定された形式の時刻の可変長 GRAPHIC ストリング表記になります。 2 番目の引数を指定しなかった場合は、デフォルトの時刻形式が使用されます。長さ属性と結果の実際の長さは 8 になります。詳しくは、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

timestamp
2 番目の引数は適用されないので、指定してはなりません。

結果は、タイム・スタンプの可変長 GRAPHIC ストリング表記になります。 *datetime-expression* が TIMESTAMP(0) の場合、結果の長さ属性および実際の長さは 19 です。*datetime-expression* のデータ・タイプが TIMESTAMP(*n*) の場合、結果の長さ属性および実際の長さは 20+*n* です。それ以外の場合、結果の長さ属性および実際の長さは 26 です。

結果の CCSID は 1200 (UTF-16) です。

ISO、EUR、USA、または JIS
結果のグラフィック・ストリングの日付形式または時刻形式を指定します。詳しくは、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

LOCAL
結果のグラフィック・ストリングの日付または時刻の形式を、現行サーバーのジョブの DATFMT、DATSEP、 TIMFMT、および TIMSEP 属性から取る必要があることを指定します。

注記

代替構文: 最初の引数がストリングで、長さ属性を指定する場合、アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、 218 ページの『CAST の指定』を参照してください。

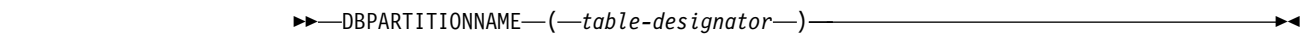
例

- 表 EMPLOYEE を使用して、ホスト変数 VAR_DESC (VARGRAPHIC(24)) を従業員番号 (EMPNO) '000050' に対応する氏名の名 (FIRSTNME) と等価の DBCLOB にセットします。

```
SELECT DBCLOB(VARGRAPHIC(FIRSTNME))
  INTO :VAR_DESC
  FROM EMPLOYEE
  WHERE EMPNO = '000050'
```

DBPARTITIONNAME

DBPARTITIONNAME 関数は、行が置かれているリレーショナル・データベースの名前 (データベース・パーティション名) を戻します。引数が非分散表を示している場合は、現行サーバーが戻されます。



パーティションの詳細については、「Db2 マルチシステム」トピック集を参照してください。

table-designator
副選択の表指定子。表指定子の詳細については、 166 ページの『表指定子』を参照してください。

SQL 命名規則では、表名は修飾できます。システム命名規則では、表名は修飾できません。

table-designator は、 *collection-derived-table*、VALUES 節、*table-function*、または *data-change-table-reference* を指定するものであってはなりません。

引数がビュー、共通表式、またはネストされた表式を示している場合、この関数はその基本表のリレーショナル・データベース名を返します。引数が、複数の基本表から派生したビュー、共通表式、またはネストされた表式を示している場合、この関数は、そのビュー、共通表式、またはネストされた表式の外側の副選択内にある最初の表のパーティション名を返します。

引数は、外側の全選択に、集約関数、GROUP BY 節、HAVING 節、UNION、INTERSECT、または EXCEPT 節、DISTINCT 節、VALUES 節、または *table-function* が含まれている、ビュー、共通表式、またはネストされた表式を示していてはなりません。全選択が集約関数、GROUP BY 文節、または HAVING 文節を含む場合、SELECT 文節に DBPARTITIONNAME 関数を指定することはできません。

この結果のデータ・タイプは、VARCHAR(18) です。結果は NULL になる場合があります。

結果の CCSID は、現行サーバーのデフォルトの CCSID になります。

注記

代替構文: NODENAME は DBPARTITIONNAME の同義語です。

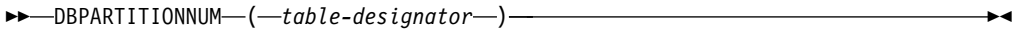
例

- EMPLOYEE 表と DEPARTMENT 表を結合し、社員番号 (EMPNO) を選択して、発生した結合に関連する各行からノードを判別します。

```
SELECT EMPNO, DBPARTITIONNAME(X), DBPARTITIONNAME(Y)
  FROM EMPLOYEE X, DEPARTMENT Y
  WHERE X.DEPTNO=Y.DEPTNO
```

DBPARTITIONNUM

DBPARTITIONNUM 関数は、行のノード番号 (データベース・パーティション名) を戻します。



引数が非分散表を識別している場合、値 0 が戻されます。[54] ノードおよびノード番号の詳細については、「Db2 UDB for iSeries マルチ・システム」を参照してください。

table-designator
副選択の表指定子。表指定子の詳細については、 166 ページの『表指定子』を参照してください。

SQL 命名規則では、表名は修飾できます。システム命名規則では、表名は修飾できません。

table-designator は、 *collection-derived-table*、VALUES 節、*table-function*、または *data-change-table-reference* を指定するものであってはなりません。

引数がビュー、共通表式、またはネストされた表式を示している場合、この関数はその基本表のノード番号を返します。引数が、複数の基本表から派生したビュー、共通表式、またはネストされた表式を示している場合、この関数は、そのビュー、共通表式、またはネストされた表式の外側の副選択内にある最初の表のノード番号を返します。

引数は、外側の全選択の副選択に、集約関数、GROUP BY 節、HAVING 節、UNION、INTERSECT、または EXCEPT 節、DISTINCT 節、VALUES 節、または *table-function* が含まれている、ビュー、共通表式、またはネストされた表式を示していてはなりません。全選択が集約関数、GROUP BY 文節、または HAVING 文節を含む場合、SELECT 文節に DBPARTITIONNUM 関数を指定することはできません。

結果のデータ・タイプは、長整数です。結果は NULL になる場合があります。

注記

代替構文: NODENUMBER は DBPARTITIONNUM の同義語です。

例

- 表 EMPLOYEE の各行についてのノード番号と従業員名を判別します。分散表の場合は、その行が存在するノードの番号が戻されます。

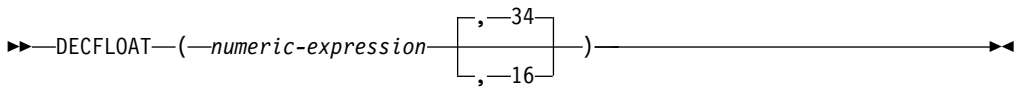
```
SELECT DBPARTITIONNUM(EMPLOYEE), LASTNAME
  FROM EMPLOYEE
```

54. 引数が、複数の論理ファイル番号に基づいて DDS が作成した論理ファイルを識別している場合、DBPARTITIONNUM は 0 を戻さず、代わりに基になる物理ファイル・メンバーの番号を戻します。

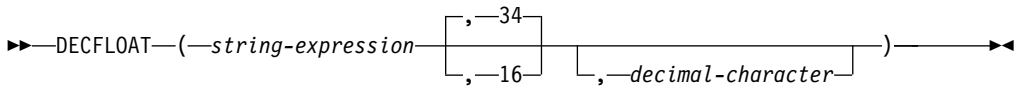
DECFLOAT

DECFLOAT 関数は、数値または数値のストリング表現を 10 進浮動小数点で表現したものを返します。

数値から **DECFLOAT** に



ストリングから **DECFLOAT** に



DECFLOAT 関数は、次のものの 10 進浮動小数点表現を返します。

- 数値
- 10 進数の文字ストリング表現またはグラフィック・ストリング表現
- 整数の文字ストリング表現またはグラフィック・ストリング表現
- 浮動小数点数の文字ストリング表現またはグラフィック・ストリング表現
- 10 進浮動小数点数の文字ストリング表現またはグラフィック・ストリング表現

数値から DECFLOAT に

numeric-expression
任意の組み込み数値データ・タイプの値を戻す式。

34 または *16*
結果の精度の桁数を指定します。デフォルトは 34 です。

結果は、最初の引数が 10 進浮動小数点の列または変数に割り当てられたときに得られる数値と同じです。

ストリングから DECFLOAT に

string-expression
数値の文字ストリング表現またはグラフィック・ストリング表現の値を戻す式。先行ブランクと末尾ブランクは除去され、大文字に変換された結果のストリングは、浮動小数点数、10 進浮動小数点数、整数、または 10 進数の定数を形成する際の規則に合致している必要があります。

34 または *16*
結果の精度の桁数を指定します。デフォルトは 34 です。

decimal-character
数値の整数部分から*string-expression* の小数桁数を区切るために使用される 1 バイトの文字定数を指定します。この文字は、ピリオドかコンマとする必要があります。 *decimal-character* を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

この関数の結果は、(暗黙に、または明示的に) 指定された精度の桁数を持つ DECFLOAT の数値です。最初の引数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引数がヌルの場合は、結果は NULL 値になります。

必要な場合は、ソースがターゲットの精度に丸められます。詳しくは、 154 ページの『CURRENT DECFLOAT ROUNDING MODE』を参照してください。

注記

代替構文: アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、 218 ページの『CAST の指定』を参照してください。

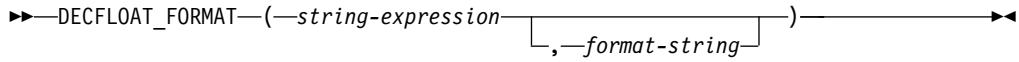
例

- この例では、DECFLOAT 関数を使用して、表 EMPLOYEE の列 EDLEVEL (データ・タイプ = SMALLINT) の選択リストで DECFLOAT データ・タイプが返されるようにしています。*select-clause* には、列 EMPNO も必要です。

```
SELECT EMPNO, DECFLOAT(EDLEVEL,16)
  FROM EMPLOYEE
```

DECFLOAT_FORMAT

DECFLOAT_FORMAT 関数は、指定された形式を使用した入力ストリング解釈に基づく DECFLOAT(34) 値を戻します。



string-expression
組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプを返す式。値がグラフィック・データ・タイプの場合、その値は関数を評価する前に暗黙的に VARCHAR にキャストされます。前後のブランクはストリングから除去されます。*format-string* が指定されていない場合、結果のサブストリングは、SQL の整数定数、10 進定数、浮動小数点定数、または 10 進浮動小数点定数を形成するための規則に準拠し、前後のブランクを除去後に 63 文字以下でなければなりません。そうでない場合、結果のサブストリングに、*format-string* で指定された形式に対応する数値のコンポーネントが含まれている必要があります。

format-string
組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプを返す式。値がグラフィック・データ・タイプの場合、その値は関数を評価する前に暗黙的に VARCHAR にキャストされます。*format-string* には、*string-expression* を DECFLOAT 値に変換処理する方法を示すテンプレートが含まれています。 *format-string* には、以下にリストするフォーマット・エレメントの有効な組み合わせが含まれていなければなりません。有効な組み合わせは、次の規則に従います。

- 少なくとも 1 つの「0」または「9」フォーマット・エレメントを指定する必要があります。
- 符号のフォーマット・エレメント (「S」、「MI」、「PR」) を指定できるのは 1 回だけです。
- 小数点のフォーマット・エレメントを指定できるのは 1 回だけです。
- 英字のフォーマット・エレメントは、大文字で指定する必要があります。
- 接頭部のフォーマット・エレメントを指定できるのは、フォーマット・ストリングの先頭のみです。また、接頭部のフォーマット・エレメントの前に接頭部以外のフォーマット・エレメントがあってはなりません。複数の接頭部のフォーマット・エレメントを指定するときには、どのような順番で指定してもかまいません。
- 接尾部のフォーマット・エレメントを指定できるのは、フォーマット・ストリングの末尾のみです。また、接尾部のフォーマット・エレメントの後に接尾部以外のフォーマット・エレメントがあってはなりません。複数の接尾部のフォーマット・エレメントを指定するときには、どのような順番で指定してもかまいません。
- コンマまたは G のフォーマット・エレメントは、接頭部のフォーマット・エレメント以外の最初のフォーマット・エレメントにすることができます。コンマまたは G のフォーマット・エレメントは、いくつでも使用することができます。

- ブランクをフォーマット・エレメントの間に指定することはできません。前後のブランクを指定することはできますが、それらは無視されます。

表 *50. DECFLOAT_FORMAT* 関数のフォーマット・エレメント

フォーマット・エレメント	説明
0 または 9	指定の場所に含めることができる数字を表します。どちらのフォーマット・エレメントも意味は同じです。
S	接頭部: *string-expression* が負の数値を表す場合、指定の場所に先行の負符号 (–) が必要です。*string-expression* が正の数値を表す場合、指定の場所に先行の正符号 (–) または先行ブランクを含めることができます。
$	接頭部: 指定の場所に、先行のドル記号 (「$」) が必要です。
MI	接尾部: *string-expression* が負の数値を表す場合、指定の場所に末尾の負符号 (-) が必要です。*string-expression* が正の数値を表す場合は、指定の場所に末尾ブランクを含めることができます。
PR	接尾部: *string-expression* が負の数値を表す場合は、先行の「より小さい」文字 (<) と末尾の「より大きい」文字 (>) が期待されます。*string-expression* が正の数値を表す場合は、先行ブランクと末尾ブランクを含めることができます。
,	コンマの位置を指定します。このコンマは、グループ分離文字として使用されます。
.	ピリオドの位置を指定します。このピリオドは小数点として使用されます。
L	接頭部または接尾部: 指定の場所にローカル通貨記号が必要であることを指定します。通貨記号は、ライブラリー *LIBL の中のメッセージ・ファイル QCPFMSG のメッセージ CPX8416 から検索されます。
D	指定の場所にローカルの小数点文字が必要であることを指定します。小数点文字は、ライブラリー *LIBL の中のメッセージ・ファイル QCPFMSG のメッセージ CPX8416 から検索されます。
G	指定の場所にローカルのグループ区切り文字が必要であることを指定します。ライブラリー *LIBL の中のメッセージ・ファイル QCPFMSG のメッセージ CPX8416 から検索されたローカルの小数点文字がピリオドである場合、グループ区切り文字はコンマになります。ローカルの小数点文字がコンマの場合、グループ区切り文字はピリオドになります。

結果は、DECFLOAT(34) です。DECFLOAT_FORMAT 関数のいずれかの引数が NULL 値の可能性がある場合、結果も NULL 値になる可能性があります。いずれかの引数が NULL 値の場合、結果は NULL 値になります。

注記

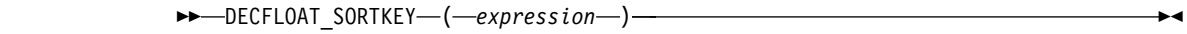
代替構文: TO_NUMBER は DECFLOAT_FORMAT の同義語です。

例

例	結果
DECFLOAT_FORMAT('123.45')	123.45
DECFLOAT_FORMAT('-123456.78')	-123456.78
DECFLOAT_FORMAT('+123456.78')	123456.78
DECFLOAT_FORMAT('1.23E4')	12300
DECFLOAT_FORMAT('123.4', '9999.99')	123.40
DECFLOAT_FORMAT('001,234', '000,000')	1234
DECFLOAT_FORMAT('1234 ', '9999MI')	1234
DECFLOAT_FORMAT('1234-', '9999MI')	-1234
DECFLOAT_FORMAT('+1234', 'S9999')	1234
DECFLOAT_FORMAT('-1234', 'S9999')	-1234
DECFLOAT_FORMAT(' 1234 ', '9999PR')	1234
DECFLOAT_FORMAT('<1234>', '9999PR')	-1234
DECFLOAT_FORMAT('$123,456.78', '$999,999.99')	123456.78

DECFLOAT_SORTKEY

DECFLOAT_SORTKEY 関数は DECFLOAT 値のソートに使用できる 2 進値を返します。



DECFLOAT_SORTKEY 関数は、全順序付けに関する IEEE 754R 仕様に準拠する方法で、10 進浮動小数点値をソートするために使用できる 2 進値を返します。

expression
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。

引数のデータ・タイプが SMALLINT、INTEGER、REAL、DOUBLE、DECIMAL(p,s) (p <=16)、または NUMERIC(p,s) (p <=16) の場合、引数は処理を行うために DECFLOAT(16) に変換されます。これ以外の場合、引数は処理を行うために DECFLOAT(34) に変換されます。

この関数の結果は、引数が DECFLOAT(16) の場合は BINARY(9)、引数が DECFLOAT(34) の場合は BINARY(17) です。

引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL である場合は、結果は NULL 値です。

例

```
CREATE TABLE T1 (D1 DECFLOAT(16));
INSERT INTO T1 VALUES(2.100);
INSERT INTO T1 VALUES(2.10);
INSERT INTO T1 VALUES(2.1000);
INSERT INTO T1 VALUES(2.1);

SELECT D1 FROM T1 ORDER BY D1;
```

```
     D1
----------------
          2.100
           2.10
         2.1000
            2.1
```

この結果セットは不定的であることに注意してください。ORDER BY によってこれらの値の順序付けは影響を受けません。

```
SELECT D1 FROM T1 ORDER BY DECFLOAT_SORTKEY(D1);
```

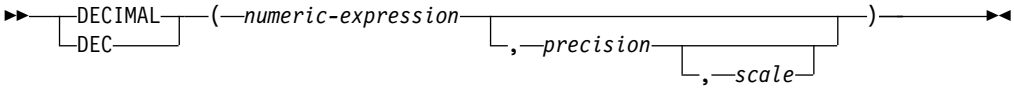
```
     D1
----------------
         2.1000
          2.100
           2.10
            2.1
```

この結果セットは、IEEE 745R の順序付け仕様に従って順序付けされていることに注意してください。

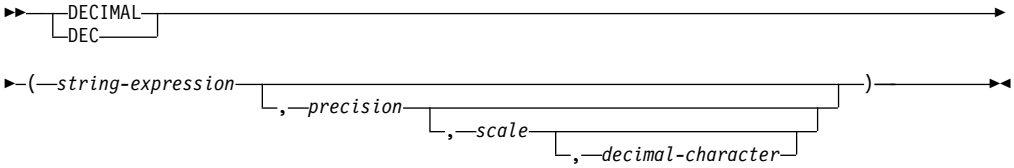
DECIMAL または DEC

DECIMAL 関数は、10 進数表現を返します。

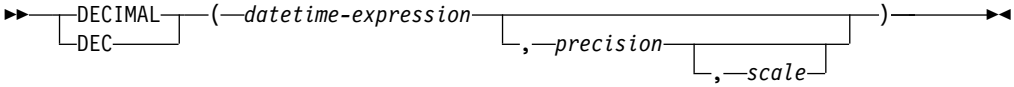
数値から **10** 進数に



ストリングから **10** 進数に



日付/時刻→ **10** 進数



DECIMAL 関数は、次のものの 10 進数表現を戻します。

- 数値
- 10 進数の文字ストリング表現またはグラフィック・ストリング表現
- 整数の文字ストリング表現またはグラフィック・ストリング表現
- 浮動小数点数の文字ストリング表現またはグラフィック・ストリング表現
- 10 進浮動小数点数の文字ストリング表現またはグラフィック・ストリング表現
- 日付
- 時刻
- タイム・スタンプ

数値から 10 進数に

numeric-expression
任意の組み込み数値データ・タイプの値を戻す式。

precision
1 以上で 63 以下の値を持つ整数定数。

デフォルトの精度 は、数値式 のデータ・タイプによって決まります。

- 5 (最初の引数が短整数の場合)
- 11 (最初の引数が長整数の場合)
- 19 (最初の引数が 64 ビット整数の場合)
- 15 (最初の引数が浮動小数点数、10 進数、数字、または位取りがゼロ以外の 2 進数の場合)
- 31 (10 進浮動小数点の場合)

scale
0 以上で精度 以下である整数定数。これを指定しないと、デフォルト値の 0 になります。

結果は、最初の引数が、精度 *precision*、位取り *scale* の 10 進数の列または変数に割り当てられたときに得られる数値と同じです。数値全体を表すのに必要な有効 10 進数字の桁数が *precision-scale* よりも大きい場合は、エラーが戻されます。最初の引数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引数がヌルの場合は、結果は NULL 値になります。

ストリングから 10 進数に

string-expression
数値の文字ストリング表現またはグラフィック・ストリング表現を戻す式。先行ブランクと末尾ブランクは除去され、結果のストリングは、浮動小数点数、10 進浮動小数点数、整数、または 10 進数の定数を形成する際の規則に合致している必要があります。

precision
1 以上で 63 以下である整数定数。これを指定しないと、デフォルト値の 15 になります。

scale
0 以上で精度 以下である整数定数。これを指定しないと、デフォルト値の 0 になります。

decimal-character
数値の整数部分から*string-expression* の小数桁数を区切るために使用される 1 バイトの文字定数を指定します。この文字は、ピリオドかコンマとする必要があります。 *decimal-character* を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

decimal-character の右側の桁数が位取り *s* より大きい場合は、末尾の桁が切り捨てられます。 *string-expression* の中の *decimal-character* より左側にある有効桁数 (数値の整数部分) が *precision-scale* より大きい場合は、エラーが戻されます。小数点文字 引数の指定がある場合は、サブストリング内のデフォルト小数点文字は無効です。

日付/時刻→ 10 進数

datetime-expression
タイプ DATE、TIME、または TIMESTAMP の値を戻す式

precision
結果の精度を指定する、1 以上で 63 以下の整数定数。この指定がない場合、精度および位取りのデフォルトは、以下のように *datetime-expression* のデータ・タイプによって決まります。

- DATE の場合は、精度が 8 で、位取りが 0 です。結果は、日付を *yyyymmdd* で表した DECIMAL(8,0) 値になります。
- TIME の場合は、精度が 6 で、位取りが 0 です。結果は、時間を *hhmmss* で表した DECIMAL(6,0) 値になります。

- TIMESTAMP(*tp*) の場合は、精度が 14+*tp* で、位取りが *tp* です。結果は、タイム・スタンプを *yyyymmddhhmmss.nnnnnnnnnnnn* で表した DECIMAL(14+*tp*、*tp*) 値になります。

scale
0 以上で精度 以下である整数定数。これを指定しないと、デフォルト値の 0 になります。

結果は、CAST(*datetime-expression* AS DECIMAL(*precision, scale*)) の場合の結果と同じ数値になります。小数点文字の右側にある数字の数が *scale* よりも多い場合、末尾から数字が切り捨てられます。 *datetime-expression* の小数点文字の左側にある有効数字 (数値の整数部分) の桁数が *precision - scale* より多い場合、エラーが戻されます。

この関数の結果は、精度が *precision* で位取りが *scale* である 10 進数です。最初の引数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引数がヌルの場合は、結果は NULL 値になります。

注記

代替構文**:** 精度を指定する場合、アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、 218 ページの『CAST の指定』を参照してください。

例

- この例では、DECIMAL 関数を使用して表 EMPLOYEE の列 EDLEVEL (データ・タイプ = SMALLINT) に関する選択リストで DECIMAL データ・タイプ (精度が 5 で、位取りが 2) が戻されるようにしています。選択リストには、列 EMPNO も必要です。

```
SELECT EMPNO, DECIMAL(EDLEVEL,5,2)
  FROM EMPLOYEE
```

- 表 PROJECT を使用して、ホスト変数に指定されている期間だけ延ばされている開始日付 (PRSTDATE) を、すべて選択しています。ホスト変数 PERIOD は INTEGER タイプであると想定します。PERIOD の値を日付期間として使用するためには、PERIOD を「キャスト」して DECIMAL(8,0) にする必要があります。

```
SELECT PRSTDATE + DECIMAL(:PERIOD,8)
  FROM PROJECT
```

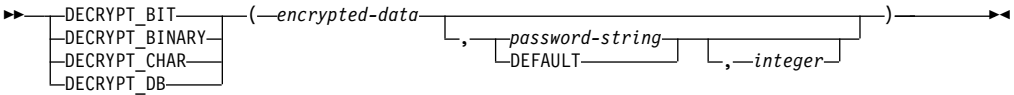
- SALARY 列への更新が、小数点文字をコンマとして、文字ストリングでウィンドウから入力される (例えば、ユーザーが 21400,50 と入力する) とします。アプリケーションで妥当性検査を受けた後、この値は CHAR(10) と定義されているホスト変数 newsalary に割り当てられます。

```
UPDATE STAFF
  SET SALARY = DECIMAL(:newsalary, 9, 2, ',')
  WHERE ID = :empid
```

SALARY の値は、これで 21400.50 になります。

DECRYPT_BIT、DECRYPT_BINARY、DECRYPT_CHAR、および DECRYPT_DB

DECRYPT_BIT、DECRYPT_BINARY、 DECRYPT_CHAR、および DECRYPT_DB 関数は、暗号化されたデータを暗号化解除した結果の値を戻します。暗号化解除に使用されるパスワードは、パスワード・ストリング 値か、 SET ENCRYPTION PASSWORD ステートメントで割り当てられる ENCRYPTION PASSWORD 値 です。



暗号化解除関数で暗号化解除できる値は、ENCRYPT_AES、ENCRYPT_RC2、または ENCRYPT_TDES 関数を使用して暗号化された値だけです。

encrypted-data
CHAR FOR BIT DATA、VARCHAR FOR BIT DATA、BINARY、VARBINARY、または BLOB 組み込みデータ・タイプの完全な暗号化されたデータ値を戻すストリング式。データ・ストリングは、ENCRYPT_AES、ENCRYPT_RC2、または ENCRYPT_TDES 関数を使用して暗号化しておく必要があります。

password-string
6 バイト以上 127 バイト以下の文字ストリング値を戻す式。この式は CLOB であってはなりません。この式は、データの暗号化に使用されたものと同じパスワードでなければならず、そうでなければ、暗号化解除の結果は、暗号化された元の値と異なるものになります。パスワード引数の値が NULL であるか、値を指定しない場合は、データは ENCRYPTION PASSWORD 値を使用して暗号化解除されます。この値は、SET ENCRYPTION PASSWORD ステートメントを使用して設定しておく必要があります。

DEFAULT
データは ENCRYPTION PASSWORD 値を使用して暗号化解除されます。この値は、SET ENCRYPTION PASSWORD ステートメントを使用して設定しておく必要があります。

integer
結果の CCSID を指定する整数定数。 DECRYPT_BIT または DECRYPT_BINARY を指定する場合は、3 番目の引数を指定してはなりません。

DECRYPT_CHAR を指定する場合は、整数 は有効な SBCS CCSID または混合データ CCSID とする必要があります。 65535 (ビット・データ) であってはなりません。 3 番目の引数が SBCS CCSID の場合は、結果は SBCS データになります。 3 番目の引数が混合 CCSID の場合は、結果は混合データになります。 3 番目の引数が指定されていない場合は、結果の CCSID は現行サーバーのデフォルト CCSID になります。

DECRYPT_DB を指定する場合は、整数 は有効な DBCS CCSID とする必要があります。 3 番目の引数が指定されていない場合は、結果の CCSID は現行サーバーのデフォルト CCSID に関連付けられた DBCS CCSID になります。

結果のデータ・タイプは、以下の表に示すとおり、指定された関数と最初の引数のデータ・タイプによって決まります。暗号化されたデータの実際のタイプから関数の結果へのキャストがサポートされない場合は、警告またはエラーが戻されます。

関数	最初の引数のデータ・タイプ	暗号化されたデータの実際のデータ・タイプ	結果
DECRYPT_BIT	CHAR FOR BIT DATA、VARCHAR FOR BIT DATA、BINARY、または VARBINARY	文字ストリング	VARCHAR FOR BIT DATA
DECRYPT_BIT	CHAR FOR BIT DATA、VARCHAR FOR BIT DATA、BINARY、または VARBINARY	グラフィック・ストリング	エラーまたは警告 **
DECRYPT_BIT	CHAR FOR BIT DATA、VARCHAR FOR BIT DATA、BINARY、または VARBINARY	2 進ストリング	エラーまたは警告 **
DECRYPT_BIT	BLOB	任意のストリング	エラー
DECRYPT_BINARY	CHAR FOR BIT DATA、VARCHAR FOR BIT DATA、BINARY、または VARBINARY	任意のストリング	VARBINARY
DECRYPT_BINARY	BLOB	任意のストリング	BLOB
DECRYPT_CHAR	CHAR FOR BIT DATA、VARCHAR FOR BIT DATA、BINARY、または VARBINARY	文字ストリング	VARCHAR
DECRYPT_CHAR	CHAR FOR BIT DATA、VARCHAR FOR BIT DATA、BINARY、または VARBINARY	Unicode グラフィック・ストリング	VARCHAR
DECRYPT_CHAR	CHAR FOR BIT DATA、VARCHAR FOR BIT DATA、BINARY、または VARBINARY	非 Unicode グラフィック・ストリング	エラーまたは警告 **
DECRYPT_CHAR	CHAR FOR BIT DATA、VARCHAR FOR BIT DATA、BINARY、または VARBINARY	2 進ストリング	エラーまたは警告 **
DECRYPT_CHAR	BLOB	文字ストリング	CLOB
DECRYPT_CHAR	BLOB	Unicode グラフィック・ストリング	CLOB

関数	最初の引数のデータ・タイプ	暗号化されたデータの実際のデータ・タイプ	結果
DECRYPT_CHAR	BLOB	非 Unicode グラフィック・ストリング	エラーまたは警告 **
DECRYPT_CHAR	BLOB	2 進ストリング	エラーまたは警告 **
DECRYPT_DB	CHAR FOR BIT DATA、VARCHAR FOR BIT DATA、BINARY、または VARBINARY	UTF-8 文字ストリングまたはグラフィック・ストリング	VARGRAPHIC
DECRYPT_DB	CHAR FOR BIT DATA、VARCHAR FOR BIT DATA、BINARY、または VARBINARY	非 UTF-8 文字ストリング	エラーまたは警告 **
DECRYPT_DB	CHAR FOR BIT DATA、VARCHAR FOR BIT DATA、BINARY、または VARBINARY	2 進ストリング	エラーまたは警告 **
DECRYPT_DB	BLOB	UTF-8 文字ストリングまたはグラフィック・ストリング	DBCLOB
DECRYPT_DB	BLOB	非 UTF-8 文字ストリング	エラーまたは警告 **
DECRYPT_DB	BLOB	2 進ストリング	エラーまたは警告 **

注:

** 暗号化解除関数が外側の副選択の選択リストに存在する場合は、データ・マッピング警告が戻されます。存在しない場合は、エラーが戻されます。データ・マッピング警告についての詳細は、 113 ページの『割り当ておよび比較』を参照してください。

暗号化されたデータ にヒントが組み込まれている場合は、関数によってヒントが戻されることはありません。結果の長さ属性は、暗号化されたデータ より 8 バイト小さいデータ・タイプの長さ属性になります。結果の実際の長さは、暗号化された元のストリングの長さです。暗号化されたデータ に暗号化されたストリングを超えるバイトが組み込まれている場合は、関数によってこれらのバイトが戻されることはありません。

引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

最初に暗号化された値とは異なる CCSID を使用してデータを暗号化解除すると、暗号化解除された値をこの CCSID に変換するときにバイト数の拡張が生じる場合があります。このような場合は、暗号化されたデータ をバイト数のより大きい可変長ストリングにキャストする必要があります。

注記

パスワード保護**:** 暗号化パスワードへの不用意なアクセスを避けるため、プログラム、プロシージャー、または関数のソースにパスワード・ストリング をストリング

定数として指定しないでください。代わりに、ENCRYPTION PASSWORD 特殊レジスターまたはホスト変数を使用してください。

リモート・リレーショナル・データベースに接続しているとき、指定されたパスワード自体は「平文で」送信されます。つまり、パスワード自体は暗号化されません。このようなケースでパスワードを保護するには、 IPSEC (または IBM i 製品同士の接続の場合は SSL) などの通信暗号化メカニズムを使用することを考慮してください。

代替構文: 旧バージョンの Db2 との互換性を確保するために、 DECRYPT_BIT の代わりに DECRYPT_BIN を指定することもできます。

例

- 表 EMP1 に SSN という社会保障の列があると想定します。この例では、暗号化パスワードを保持するために ENCRYPTION PASSWORD 値を使用します。

```
SET ENCRYPTION PASSWORD = :pw

INSERT INTO EMP1 (SSN) VALUES ENCRYPT_RC2( '289-46-8832' )

SELECT DECRYPT_CHAR( SSN)
  FROM EMP1
```

  DECRYPT_CHAR 関数は、元の値「289-46-8832」を戻します。

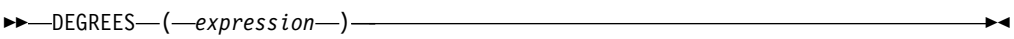
- この例では、変数 pw にセットされた暗号化パスワードを明示的に受け渡しします。

```
INSERT INTO EMP1 (SSN) VALUES ENCRYPT_TDES( '289-46-8832', :pw)

SELECT DECRYPT_CHAR( SSN, :pw)
  FROM EMP1
```

  DECRYPT_CHAR 関数は、元の値「289-46-8832」を戻します。

DEGREES

DEGREES 関数は、引数の度数をラジアンで表した角度で戻します。



expression
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

引数のデータ・タイプが DECFLOAT(*n*) の場合、結果は DECFLOAT(*n*) です。それ以外の場合、結果のデータ・タイプは、倍精度の浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

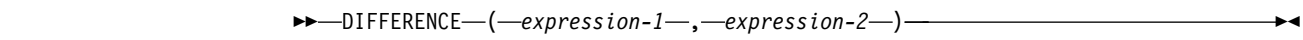
- ホスト変数 RAD は、値が 3.142 の DECIMAL(4,3) のホスト変数であると想定します。

```
SELECT DEGREES(:RAD)
  FROM SYSIBM.SYSDUMMY1
```

およそ 180.0 の値が戻されます。

DIFFERENCE

DIFFERENCE 関数は、ストリングに SOUNDEX 関数を適用し、2 つのストリングの音の相違を表す 0 から 4 の値を戻します。値 4 が、音が一致する可能性が最も高くなります。



expression-1
CLOB または DBCLOB を除く、組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプを戻す式。引数を 2 進ストリングとすることはできません。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。

expression-2
CLOB または DBCLOB を除く、組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプを戻す式。引数を 2 進ストリングとすることはできません。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。

結果のデータ・タイプは、INTEGER です。引数のどちらかが NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数のどちらかが NULL である場合は、結果は NULL 値になります。

例

- 次のステートメントで、

```
SELECT DIFFERENCE('CONSTRAINT','CONSTANT'),
       SOUNDEX('CONSTRAINT'),
       SOUNDEX('CONSTANT')
  FROM SYSIBM.SYSDUMMY1
```

  4、C523、および C523 が戻されたとします。 2 つのストリングが同じ SOUNDEX 値を戻しているので、相違は 4 (可能な最高値) になります。

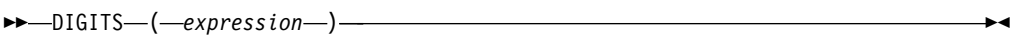
- 次のステートメントで、

```
SELECT DIFFERENCE('CONSTRAINT','CONTRITE'),
       SOUNDEX('CONSTRAINT'),
       SOUNDEX('CONTRITE')
  FROM SYSIBM.SYSDUMMY1
```

  2、C523、C536 が戻されたとします。この場合、2 つのストリングが異なる SOUNDEX 値を戻しているので、相違値は低くなります。

DIGITS

DIGITS 関数は、数値の絶対値の文字ストリング表現を戻します。



expression
組み込み SMALLINT、INTEGER、BIGINT、DECIMAL、NUMERIC、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に DECIMAL(63,31) にキャストされます。ストリングを 10 進数に変換する方法については、 429 ページの『DECIMAL または DEC』を参照してください。

引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

この関数の結果は、その位取りには関係なく、引数の絶対値を表現する固定長の文字ストリングになります。結果には、符号や小数点は含まれません。文字ストリングは数字だけから構成され、必要に応じてストリングは、先行ゼロによって埋め込まれます。ストリングの長さは、次のとおりです。

- 5 (引数が位取りゼロの短精度整数の場合)
- 10 (引数が位取りゼロの長精度整数の場合)
- 19 (引数が 64 ビット整数の場合)
- *p* (引数が 10 進数 (または位取りがゼロより大きい整数) で精度が *p* の場合)

文字ストリングの CCSID は、現行サーバーにおけるデフォルト SBCS CCSID です。

例

- 表 TABLEX に 10 桁の整数の数値を含む列 INTCOL があるものと想定します。次の例は、列 INTCOL に入っている数値の最初の 4 桁の数字の組み合わせすべてをリストしています。

```
SELECT DISTINCT SUBSTR(DIGITS(INTCOL),1,4)
  FROM TABLEX
```

- COLUMNX が DECIMAL(6,2) のデータ・タイプを持ち、その値の 1 つが -6.28 であると想定します。

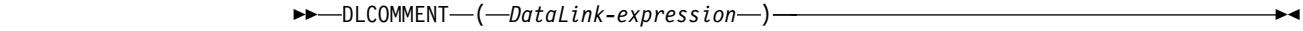
```
SELECT DIGITS(COLUMNX)
  FROM TABLEX
```

値として「000628」が返されます。

結果は、長さ 6 (該当の列の精度) のストリングになります。この長さになるように先行ゼロが埋め込まれます。符号も小数点も、結果には含まれません。

DLCOMMENT

DLCOMMENT 関数は、データ・リンク値からコメント値を (それが存在する場合) を戻します。



DataLink-expression
結果が DataLink 組み込みデータ・タイプの値になる式。

この関数の結果は VARCHAR(254) になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

文字ストリングの CCSID は、データ・リンク式 のものと同じになります。

例

- HOCKEY_GOALS 表の ARTICLES 列へのリンクから、日付、記述、およびコメントを選択するステートメントを準備します。選択する行は、Richard 兄弟 (Maurice か Henri) のいずれかが点を入れたゴールの行です。

```
stmtvar = "SELECT DATE_OF_GOAL, DESCRIPTION, DLCOMMENT(ARTICLES)
                 FROM HOCKEY_GOALS
                 WHERE BY_PLAYER = 'Maurice Richard'
                 OR BY_PLAYER = 'Henri Richard' ";
EXEC SQL PREPARE HOCKEY_STMT FROM :stmtvar;
```

- スカラー関数を使用して表 TBLA のある行の列 COLA に挿入されているデータ・リンク値があるとします。

```
INSERT INTO TBLA
  VALUES (DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment'))
```

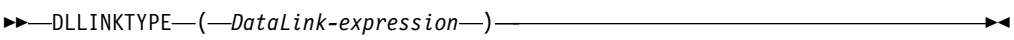
次の関数がこの値に対して実行されると、

```
SELECT DLCOMMENT(COLA)
  FROM TBLA
```

「A comment」という値が戻されます。

DLLINKTYPE

DLLINKTYPE 関数は、データ・リンク値からリンク・タイプ値を戻します。



DataLink-expression
結果が DataLink 組み込みデータ・タイプの値になる式。

この関数の結果は VARCHAR(4) になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

文字ストリングの CCSID は、データ・リンク式 のものと同じになります。

例

- スカラー関数を使用して表 TBLA のある行の列 COLA に挿入されているデータ・リンク値があるとします。

```
INSERT INTO TABLA
  VALUES( DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment') )
```

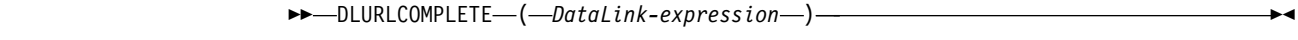
次の関数がこの値に対して実行されると、

```
SELECT DLLINKTYPE(COLA)
  FROM TBLA
```

「URL」という値が戻されます。

DLURLCOMPLETE

DLURLCOMPLETE 関数は、リンク・タイプ URL のデータ・リンク値から完全な URL 値を戻します。この値は、DLURLSCHEME を '://' と、次に DLURLSERVER と、さらに DLURLPATH と連結した結果と同じになります。データ・リンクの属性が FILE LINK CONTROL で、しかも READ PERMISSION DB である場合、値にはファイル・アクセス・トークンが含まれます。



DataLink-expression
結果が DataLink 組み込みデータ・タイプの値になる式。

引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

この関数の結果は可変長ストリングです。属性の長さは、データ・リンクの属性によって次のように異なります。

- データ・リンクの属性が FILE LINK CONTROL でかつ READ PERMISSION DB の場合は、結果の長さ属性は引数の長さ属性に 19 を加えたもの
- それ以外の場合は、結果の長さ属性は、引数の長さ属性

データ・リンク値にコメントしか含まれていない場合は、戻る結果は長さゼロのストリングです。

文字ストリングの CCSID は、データ・リンク式 のものと同じになります。

例

- スカラー関数を使用して表 TBLA のある行の列 COLA に (FILE LINK CONTROL および READ PERMISSION DB という属性で) 挿入されているデータ・リンク値があるとします。

```
INSERT INTO TABLA
  VALUES( DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment') )
```

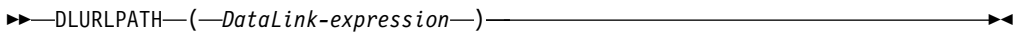
次の関数がこの値に対して実行されると、

```
SELECT DLURLCOMPLETE(COLA)
  FROM TBLA
```

「HTTP://DLFS.ALMADEN.IBM.COM/x/y/**********;a.b」という値が戻されます。********** はアクセス・トークンを表します。

DLURLPATH

DLURLPATH 関数は、リンク・タイプ URL のデータ・リンク値から、あるサーバー内のファイルにアクセスするのに必要なパスとファイル名を戻します。該当する場合は、この値にはファイル・アクセス・トークンが含まれます。



DataLink-expression
結果が DataLink 組み込みデータ・タイプの値になる式。

引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

この関数の結果は可変長ストリングです。属性の長さは、データ・リンクの属性によって次のように異なります。

- データ・リンクの属性が FILE LINK CONTROL でかつ READ PERMISSION DB の場合は、結果の長さ属性は引数の長さ属性に 19 を加えたもの
- それ以外の場合は、結果の長さ属性は、引数の長さ属性

データ・リンク値にコメントしか含まれていない場合は、戻る結果は長さゼロのストリングです。

文字ストリングの CCSID は、データ・リンク式 のものと同じになります。

例

- スカラー関数を使用して表 TBLA のある行の列 COLA に (FILE LINK CONTROL および READ PERMISSION DB という属性で) 挿入されているデータ・リンク値があるとします。

```
INSERT INTO TABLA
  VALUES( DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment') )
```

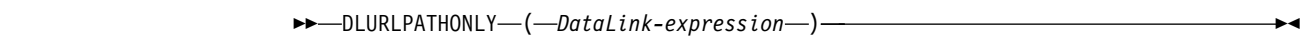
次の関数がこの値に対して実行されると、

```
SELECT DLURLPATH(COLA)
  FROM TBLA
```

「/x/y/**********;a.b」という値が戻されます。 ********** はアクセス・トークンを表します。

DLURLPATHONLY

DLURLPATHONLY 関数は、リンク・タイプ URL のデータ・リンク値から、あるサーバー内のファイルにアクセスするのに必要なパスとファイル名を戻します。戻される値には、ファイル・アクセス・トークンは含まれていません。



DataLink-expression
結果が DataLink 組み込みデータ・タイプの値になる式。

引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

この関数の結果は、引数の長さ属性に等しい長さ属性を持つ可変長ストリングになります。

データ・リンク値にコメントしか含まれていない場合は、戻る結果は長さゼロのストリングです。

文字ストリングの CCSID は、データ・リンク式 のものと同じになります。

例

- スカラー関数を使用して表 TBLA のある行の列 COLA に挿入されているデータ・リンク値があるとします。

```
INSERT INTO TABLA
  VALUES( DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment') )
```

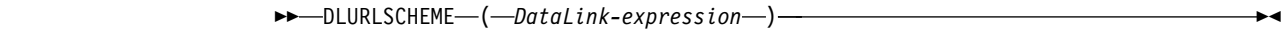
次の関数がこの値に対して実行されると、

```
SELECT DLURLPATHONLY(COLA)
  FROM TBLA
```

「/x/y/a.b」という値が戻されます。

DLURLSCHEME

DLURLSCHEME 関数は、リンク・タイプ URL のデータ・リンク値からそのスキームを戻します。この値は常に大文字です。



DataLink-expression
結果が DataLink 組み込みデータ・タイプの値になる式。

この関数の結果は VARCHAR(20) になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

データ・リンク値にコメントしか含まれていない場合は、戻る結果は長さゼロのストリングです。

文字ストリングの CCSID は、データ・リンク式 のものと同じになります。

例

- スカラー関数を使用して表 TBLA のある行の列 COLA に挿入されているデータ・リンク値があるとします。

```
INSERT INTO TABLA
  VALUES( DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment') )
```

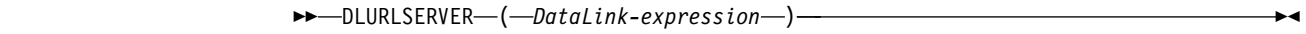
次の関数がこの値に対して実行されると、

```
SELECT DLURLSCHEME(COLA)
  FROM TBLA
```

「HTTP」という値が戻されます。

DLURLSERVER

DLURLSERVER 関数は、リンク・タイプ URL のデータ・リンク値から、ファイル・サーバーを戻します。この値は常に大文字です。



DataLink-expression
結果が DataLink 組み込みデータ・タイプの値になる式。

引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

この関数の結果は、引数の長さ属性に等しい長さ属性を持つ可変長ストリングになります。

データ・リンク値にコメントしか含まれていない場合は、戻る結果は長さゼロのストリングです。

文字ストリングの CCSID は、データ・リンク式 のものと同じになります。

例

- スカラー関数を使用して表 TBLA のある行の列 COLA に挿入されているデータ・リンク値があるとします。

```
INSERT INTO TABLA
  VALUES( DLVALUE('http://dlfs.almaden.ibm.com/x/y/a.b','URL','A comment') )
```

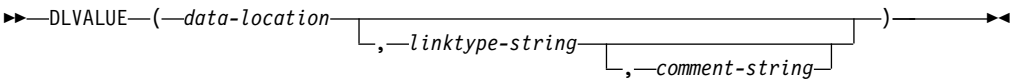
次の関数がこの値に対して実行されると、

```
SELECT DLURLSERVER(COLA)
  FROM TBLA
```

「DLFS.ALMADEN.IBM.COM」という値が戻されます。

DLVALUE

DLVALUE 関数は、データ・リンク値を戻します。この関数を UPDATE ステートメントの SET 文節の右側または INSERT ステートメントの VALUES 文節で使用した場合は、通常はファイルに対するリンクも作成されます。ただし、コメントだけを指定すると (この場合は、データ・ロケーション は長さゼロのストリング)、データ・リンク値は空のリンク属性を使用して作成され、したがってファイル・リンクにはなりません。



data-location
リンク・タイプが URL の場合は、これは完全な URL 値を含む文字ストリング式です。式が空ストリングではない場合は、この中に URL スキームと URL サーバーを入れる必要があります。文字ストリング式の実際の長さは、32718 文字以下でなければなりません。

linktype-string
データ・リンク値のリンク・タイプを指定するオプションの文字ストリング式。有効な値は「URL」だけです。

comment-string
コメント、または追加のロケーション情報を提供するオプションの文字ストリング式。この文字ストリング式の実際の長さは、254 文字以下でなければなりません。

コメント・ストリング は、NULL 値であってはなりません。コメント・ストリング が指定されない場合は、コメント・ストリング は空ストリングになります。

最初の引数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引数がヌルの場合は、結果は NULL 値になります。

この関数の結果は、データ・リンク値になります。

データ・リンクの CCSID は、次の場合を除き、データ・ロケーション のものと同じになります。

- コメント・ストリング が混合データで、データ・ロケーション が混合データでない場合、結果の CCSID は コメント・ストリング の CCSID になります。 [55]
- データ・ロケーション が CCSID としてビット・データ (65535)、 UTF-16 グラフィック・データ (1200)、UCS-2 グラフィック・データ (13488)、トルコ語データ (905 または 1026)、あるいは日本語データ (290、930、または 5026) を持っている場合は、結果の CCSID は次の表のようになります。

データ・ロケーションの **CCSID**	コメント・ストリングの **CCSID**	結果の **CCSID**
65535	65535	ジョブ・デフォルト CCSID

55. コメント・ストリング の CCSID が 5026 または 930 の場合、結果の CCSID は 939 になります。

データ・ロケーションの **CCSID**	コメント・ストリングの **CCSID**	結果の **CCSID**
65535	65535 以外	コメント・ストリング CCSID (CCSID が 290、930、 5026、905、1026 または 13488 の場合を除く。これらの場合は、CCSID は以下に示すように修正される。)
290	任意	4396
930 または 5026	任意	939
905 または 1026	任意	500
1200	任意	500
13488	任意	500

この関数を使用してデータ・リンク値を定義するときは、値のターゲットの最大長を考慮してください。例えば、DataLink(200) と定義されている列では、データ・ロケーション の最大長にコメントを加えたものが 200 バイトになります。

例

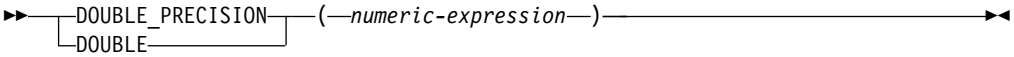
- 表に 1 行を挿入するとします。最初の 2 つのリンクの URL 値は、url_article と url_snapshot という名前の変数に入っています。また、url_snapshot_comment という名前の変数には、スナップショット・リンクに付随するコメントが入っています。ただし、movie のためのリンクはまだありません。url_movie_comment という名前の変数にコメントが入っているだけです。

```
INSERT INTO HOCKEY_GOALS
  VALUES('Maurice Richard',
         'Montreal canadian',
         '?',
         'Boston Bruins,
         '1952-04-24',
         'Winning goal in game 7 of Stanley Cup final',
         DLVALUE(:url_article),
         DLVALUE(:url_snapshot, 'URL', :url_snapshot_comment),
         DLVALUE('', 'URL', :url_movie_comment) )
```

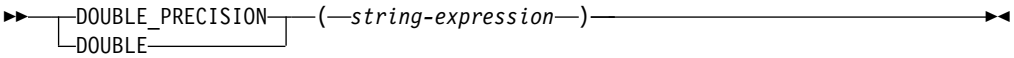
DOUBLE_PRECISION または DOUBLE

DOUBLE_PRECISION と DOUBLE の関数は、浮動小数点表現を返します。

数値から倍精度に



ストリングから倍精度に



DOUBLE_PRECISION と DOUBLE の関数は、次のものの浮動小数点表現を戻します。

- 数値
- 10 進数の文字ストリング表現またはグラフィック・ストリング表現
- 整数の文字ストリング表現またはグラフィック・ストリング表現
- 浮動小数点数の文字ストリング表現またはグラフィック・ストリング表現
- 10 進浮動小数点数の文字ストリング表現またはグラフィック・ストリング表現

数値から倍精度に

numeric-expression
任意の組み込み数値データ・タイプの値を戻す式。

結果は、式が倍精度浮動小数点数の列または変数に割り当てられていた場合に得られるものと同じ数値になります。

ストリングから倍精度に

string-expression
数値の文字ストリング表現またはグラフィック・ストリング表現の値を戻す式。

引数がストリング式 の場合、結果は、CAST(ストリング式 AS DOUBLE PRECISION で得られる数値と同じです。先行ブランクと末尾ブランクは除去され、結果のストリングは、浮動小数点数、10 進浮動小数点数、整数、または 10 進数の定数を形成する際の規則に合致している必要があります。

数値の整数部分からストリング式 の小数桁数を区切るために使用する必要のある 1 バイトの文字定数は、デフォルトの小数点文字です。詳しくは、 147 ページの『小数点』を参照してください。

この関数の結果は、倍精度浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

注記

代替構文: FLOAT は、DOUBLE_PRECISION および DOUBLE の同義語です。

アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、218 ページの『CAST の指定』を参照してください。

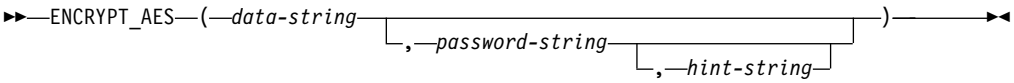
例

- 表 EMPLOYEE を使用して、何らかの手数料を得ている社員について、給与に占める手数料の割合を求めます。給与 (列 SALARY) および手数料 (列 COMM) のデータ・タイプは、DECIMAL (10 進数) です。範囲外の結果が生じる可能性を避けるために、除算が浮動小数点数で行われるように、SALARY に対して DOUBLE-PRECISION が使用されます。

```
SELECT EMPNO, DOUBLE_PRECISION(SALARY)/COMM
  FROM EMPLOYEE
  WHERE COMM > 0
```

ENCRYPT_AES

ENCRYPT_ 関数は、AES 暗号化アルゴリズムを使用してデータ・ストリング を暗号化した結果の値を返します。暗号化解除に使用されるパスワードは、パスワード・ストリング 値か、暗号化パスワードの値 (SET ENCRYPTION PASSWORD ステートメントで割り当てられる) です。



data-string
暗号化するストリング値を戻す式。ストリング式は、組み込みストリング・データ・タイプでなければなりません。

データ・ストリング のデータ・タイプの長さ属性は、ヒント・ストリング 引数を指定しない場合、結果のデータ・タイプの最大長より 24 バイト (または 32 バイト) 短い長さに、また、ヒント・ストリング 引数を指定する場合は結果のデータ・タイプの最大長より 56 バイト (または 64 バイト) 短い長さに制限されています。

password-string
6 バイト以上 127 バイト以下の文字ストリング値を戻す式。この式は CLOB であってはならず、式の CCSID は 65535 であってはなりません。この値は、データ・ストリングを暗号化するために使用したパスワードを表します。パスワード引数の値が NULL であるか、値を指定しない場合は、データは ENCRYPTION PASSWORD 値を使用して暗号化されます。この値は、SET ENCRYPTION PASSWORD ステートメントを使用して設定しておく必要があります。

hint-string
データ所有者がパスワードを思い出すのに役立つ最大 32 バイトの文字ストリング値を戻す式 (例えば、「Ocean」は「Pacific」を思い出すためのヒント)。この式は CLOB であってはならず、式の CCSID は 65535 であってはなりません。ヒント値を指定すると、ヒントは結果に埋め込まれ、GETHINT 関数を使用して検索できます。パスワード・ストリング が指定されており、この引数が NULL 値であるか提供されない場合は、ヒントは結果に埋め込まれません。パスワード・ストリング が指定されていない場合は、SET ENCRYPTION PASSWORD ステートメントを使用してヒントを指定できます。

結果のデータ・タイプは、以下の表に示すとおり、最初の引数によって決まります。

最初の引数のデータ・タイプ	結果のデータ・タイプ
BINARY または VARBINARY	VARBINARY
CHAR、VARCHAR、GRAPHIC、または VARGRAPHIC	VARCHAR FOR BIT DATA
BLOB、CLOB、または DBCLOB	BLOB

結果の長さ属性は、指定される引数によって次のように異なります。

- パスワード・ストリング を指定し、ヒント・ストリング を指定しない場合、データ・ストリング の長さ属性に 24 を足し、それに 16 バイト境界までのバイト数を足したものになります。
- それ以外の場合は、データ・ストリング の長さ属性に 64 を足し、それに 16 バイト境界までのバイト数を足したものになります。

結果の実際の長さは、以下の合計になります。

- *data-string* の実際の長さに、16 バイト境界までのバイト数を足したもの。
- ヒントの実際の長さ。

  ヒント・ストリング が関数の引数として、または SET ENCRYPTION PASSWORD ステートメントに指定されていない場合、ヒントの実際の長さはゼロです。
- *n* - ここで *n* (値を暗号化するのに必要なオーバーヘッドの量) は 24 バイト (データ・ストリング が LOB である場合、またはデータ・ストリング、パスワード、またはヒントに別の CCSID 値を使用する場合は、32 バイト) です。

最初の引数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引数がヌルの場合は、結果は NULL 値になります。

暗号化された結果は、データ・ストリング 値より長くなることに注意してください。したがって、暗号化された値を割り当てる場合は、暗号化された値の全体を入れるのに十分なサイズでターゲットを宣言するようにしてください。

注

パスワード保護**:** 暗号化パスワードへの不用意なアクセスを避けるため、プログラム、プロシージャー、または関数のソースにパスワード・ストリング をストリング定数として指定しないでください。代わりに、SET ENCRYPTION PASSWORD ステートメントまたはホスト変数を使用してください。

リモート・リレーショナル・データベースに接続しているとき、指定されたパスワード自体は「平文で」送信されます。つまり、パスワード自体は暗号化されません。このようなケースでパスワードを保護するには、 IPSEC (または IBM i 製品同士の接続の場合は SSL) などの通信暗号化メカニズムを使用することを考慮してください。

暗号化アルゴリズム**:** 使用される内部暗号化アルゴリズムは、IBM Research の CLiC Toolkit のものです。128 ビット暗号鍵が SHA1 メッセージ要約を使用してパスワードから引き出されます。

暗号化パスワードおよびデータ**:** パスワードは、ユーザーが責任を持って管理します。データを暗号化すると、データを暗号化解除するのに使用できるのは暗号化に使用したパスワードだけです。 CHAR 変数を使用してパスワード値を設定する場合は、パスワード値にブランクが埋め込まれることがあるので注意してください。暗号化された結果には、NULL 終止符や他の印刷できない文字が含まれる場合があります。

表列の定義**:** 列および特殊タイプに暗号化されたデータが入るように定義する場合:

- CHAR FOR BIT DATA、VARCHAR FOR BIT DATA、BINARY、VARBINARY、または BLOB のデータ・タイプを使用して列を定義する必要があります。
- 列の長さ属性に追加の *n* バイトが含まれている必要があります。 *n* は、前述のとおり、データを暗号化するのに必要なオーバーヘッドです。

これらのデータ・タイプの 1 つがない列、または提案されたデータ長より短い長さの列に割り当てまたはキャストを行うと、割り当てエラーになる場合があります。あるいは、割り当てが成功すると、その後データを暗号化解除するときに失敗してデータが失われます。ブランクは暗号化されたデータとして有効ですが、短い列に保管される場合は切り捨てられます。

列の長さの計算例を以下に示します。

```
暗号化されていないデータの最大長          6 バイト
次の 16 バイト境界までのバイト数          10 バイト
オーバーヘッド                            24 バイト (または 32 バイト)
                                          --------
暗号化されたデータの列の長さ              40 バイト (または 48 バイト)


暗号化されていないデータの最大長          32 バイト
16 バイト境界までのバイト数               0 バイト
オーバーヘッド                            24 バイト (または 32 バイト)
                                          --------
暗号化されたデータ列の長さ                56 バイト
```

暗号化されたデータの管理**:** 暗号化されたデータは、ENCRYPT_AES 関数に対応する暗号化解除関数をサポートするサーバーのみで暗号化解除できます。したがって、暗号化されたデータを含む列のレプリケーションは、暗号化解除関数をサポートするサーバーに対してのみ行う必要があります。

例

- 表 EMP1 に SSN という社会保障の列があると想定します。この例では、暗号化パスワードを保持するために ENCRYPTION PASSWORD 値を使用します。

```
SET ENCRYPTION PASSWORD = 'Ben123'

INSERT INTO EMP1 (SSN) VALUES ENCRYPT_AES( '289-46-8832' )
```

- この例は、暗号化パスワードを明示的に受け渡しします。

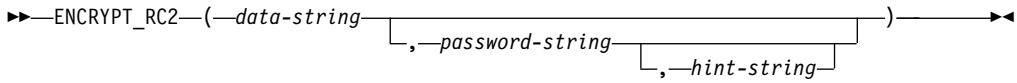
```
INSERT INTO EMP1 (SSN) VALUES ENCRYPT_AES( '289-46-8832', 'Ben123' )
```

- ヒント「Ocean」は、ユーザーが暗号化パスワード「Pacific」を思い出せるようにするために保管されます。

```
INSERT INTO EMP1 (SSN) VALUES ENCRYPT_AES( '289-46-8832', 'Pacific', 'Ocean' )
```

ENCRYPT_RC2

ENCRYPT_RC2 関数は、RC2 暗号化アルゴリズムを使用してデータ・ストリングを暗号化した結果の値を戻します。暗号化解除に使用されるパスワードは、パスワード・ストリング 値か、暗号化パスワードの値 (SET ENCRYPTION PASSWORD ステートメントで割り当てられる) です。



data-string
暗号化するストリング値を戻す式。ストリング式は、組み込みストリング・データ・タイプでなければなりません。

データ・ストリング のデータ・タイプの長さ属性は、 *m* - MOD(*m*,8) - *n* - 1 より小さくなければなりません。ここで、*m* は結果データ・タイプの最大長で、*n* は、値を暗号化するのに必要なオーバーヘッドの量です。

- ヒント・ストリング が指定されない場合、*n* は 8 バイトです。
- ヒント・ストリング が指定される場合、*n* は 40 バイトです。

password-string
6 バイト以上 127 バイト以下の文字ストリング値を戻す式。この式は CLOB であってはなりません。この値は、データ・ストリングを暗号化するために使用したパスワードを表します。パスワード引数の値が NULL であるか、値を指定しない場合は、データは ENCRYPTION PASSWORD 値を使用して暗号化されます。この値は、SET ENCRYPTION PASSWORD ステートメントを使用して設定しておく必要があります。

hint-string
データ所有者がパスワードを思い出すのに役立つ最大 32 バイトの文字ストリング値を戻す式 (例えば、「Ocean」は「Pacific」を思い出すためのヒント)。この式は CLOB であってはなりません。ヒント値を指定すると、ヒントは結果に埋め込まれ、GETHINT 関数を使用して検索できます。パスワード・ストリングが指定されており、この引数が NULL 値であるか提供されない場合は、ヒントは結果に埋め込まれません。パスワード・ストリング が指定されていない場合は、SET ENCRYPTION PASSWORD ステートメントを使用してヒントを指定できます。

結果のデータ・タイプは、以下の表に示すとおり、最初の引数によって決まります。

最初の引数のデータ・タイプ	結果のデータ・タイプ
BINARY または VARBINARY	VARBINARY
CHAR、VARCHAR、GRAPHIC、または VARGRAPHIC	VARCHAR FOR BIT DATA
BLOB、CLOB、または DBCLOB	BLOB

結果の長さ属性は、指定される引数によって次のように異なります。

- パスワード・ストリング が指定されているが、ヒント・ストリング は指定されていない場合は、データ・ストリング の長さ属性に 16 を足し、それに次の 8 バイト境界までのバイト数を足したものになります。[56]
- それ以外の場合は、データ・ストリング の長さ属性に 48 を足し、それに次の 8 バイト境界までのバイト数を足したものになります。[56]

結果の実際の長さは、以下の合計になります。

- *data-string* の実際の長さに、次の 8 バイト境界までのバイト数を足したもの。[56]
- ヒントの実際の長さ。

  ヒント・ストリング が関数の引数として、または SET ENCRYPTION PASSWORD ステートメントに指定されていない場合、ヒントの実際の長さはゼロです。
- *n* - ここで *n* (値を暗号化するのに必要なオーバーヘッドの量) は 8 バイト (*data-string* が LOB である場合、または *data-string*、パスワード、またはヒントに別の CCSID 値を使用する場合は、16 バイト) です。

引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

暗号化された結果は、データ・ストリング 値より長くなることに注意してください。したがって、暗号化された値を割り当てる場合は、暗号化された値の全体を入れるのに十分なサイズでターゲットを宣言するようにしてください。

注

パスワード保護**:** 暗号化パスワードへの不用意なアクセスを避けるため、プログラム、プロシージャー、または関数のソースにパスワード・ストリング をストリング定数として指定しないでください。代わりに、SET ENCRYPTION PASSWORD ステートメントまたはホスト変数を使用してください。

リモート・リレーショナル・データベースに接続しているとき、指定されたパスワード自体は「平文で」送信されます。つまり、パスワード自体は暗号化されません。このようなケースでパスワードを保護するには、 IPSEC (または IBM i 製品同士の接続の場合は SSL) などの通信暗号化メカニズムを使用することを考慮してください。

暗号化アルゴリズム**:** 使用される内部暗号化アルゴリズムは、埋め込み処理を行う RC2 ブロック暗号で、128 ビットの秘密鍵は、 MD5 メッセージ要約を使用してパスワードから引き出されます。

暗号化パスワードおよびデータ**:** パスワードは、ユーザーが責任を持って管理します。データを暗号化すると、データを暗号化解除するのに使用できるのは暗号化に使用したパスワードだけです。 CHAR 変数を使用してパスワード値を設定する場合は、パスワード値にブランクが埋め込まれることがあるので注意してください。暗号化された結果には、NULL 終止符や他の印刷できない文字が含まれる場合があります。

---

56. ENCRYPT_TDES や ENCRYPT_AES とは異なり、*data-string* の長さが既に 8 バイト境界に達しているとしても 8 バイトを追加します。

表列の定義**:** 列および特殊タイプに暗号化されたデータが入るように定義する場合:

- CHAR FOR BIT DATA、VARCHAR FOR BIT DATA、BINARY、VARBINARY、または BLOB のデータ・タイプを使用して列を定義する必要があります。
- 列の長さ属性に追加の *n* バイトが含まれている必要があります。 *n* は、前述のとおり、データを暗号化するのに必要なオーバーヘッドです。

これらのデータ・タイプの 1 つがない列、または提案されたデータ長より短い長さの列に割り当てまたはキャストを行うと、割り当てエラーになる場合があります。あるいは、割り当てが成功すると、その後データを暗号化解除するときに失敗してデータが失われます。ブランクは暗号化されたデータとして有効ですが、短い列に保管される場合は切り捨てられます。

列の長さの計算例を以下に示します。

```
暗号化されていないデータの最大長          6 バイト
次の 8 バイト境界までのバイト数            2 バイト
オーバーヘッド                           8 バイト (または 16 バイト)
                                        --------
暗号化されたデータの列の長さ              16 バイト (または 32 バイト)


暗号化されていないデータの最大長          32 バイト
次の 8 バイト境界までのバイト数            8 バイト
オーバーヘッド                           8 バイト (または 16 バイト)
                                        --------
暗号化されたデータの列の長さ              48 バイト (または 56 バイト)
```

暗号化されたデータの管理**:** 暗号化されたデータは、 ENCRYPT_RC2 関数に対応する暗号化解除関数をサポートするサーバーでのみ暗号化解除できます。したがって、暗号化されたデータを含む列のレプリケーションは、暗号化解除関数をサポートするサーバーに対してのみ行う必要があります。

代替構文**:** 旧バージョンの Db2 との互換性を確保するために、 ENCRYPT_RC2 の代わりに ENCRYPT を指定することもできます。

例

- 表 EMP1 に SSN という社会保障の列があると想定します。この例では、暗号化パスワードを保持するために ENCRYPTION PASSWORD 値を使用します。

```
SET ENCRYPTION PASSWORD = 'Ben123'

INSERT INTO EMP1 (SSN) VALUES ENCRYPT_RC2( '289-46-8832' )
```

- この例は、暗号化パスワードを明示的に受け渡しします。

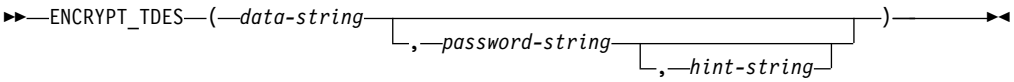
```
INSERT INTO EMP1 (SSN) VALUES ENCRYPT_RC2( '289-46-8832', 'Ben123' )
```

- ヒント「Ocean」は、ユーザーが暗号化パスワード「Pacific」を思い出せるようにするために保管されます。

```
INSERT INTO EMP1 (SSN) VALUES ENCRYPT_RC2( '289-46-8832', 'Pacific', 'Ocean' )
```

ENCRYPT_TDES

ENCRYPT_TDES 関数は、Triple DES 暗号化アルゴリズムを使用してデータ・ストリング を暗号化した結果の値を戻します。暗号化解除に使用されるパスワードは、パスワード・ストリング 値か、暗号化パスワードの値 (SET ENCRYPTION PASSWORD ステートメントで割り当てられる) です。



data-string
暗号化するストリング値を戻す式。ストリング式は、組み込みストリング・データ・タイプでなければなりません。

データ・ストリング のデータ・タイプの長さ属性は、 *m* - MOD(*m*,8) - *n* - 1 より小さくなければなりません。ここで、*m* は結果データ・タイプの最大長で、*n* は、値を暗号化するのに必要なオーバーヘッドの量です。

password-string
6 バイト以上 127 バイト以下の文字ストリング値を戻す式。この式は CLOB であってはならず、式の CCSID は 65535 であってはなりません。この値は、データ・ストリングを暗号化するために使用したパスワードを表します。パスワード引数の値が NULL であるか、値を指定しない場合は、データは ENCRYPTION PASSWORD 値を使用して暗号化されます。この値は、SET ENCRYPTION PASSWORD ステートメントを使用して設定しておく必要があります。

hint-string
データ所有者がパスワードを思い出すのに役立つ最大 32 バイトの文字ストリング値を戻す式 (例えば、「Ocean」は「Pacific」を思い出すためのヒント)。この式は CLOB であってはならず、式の CCSID は 65535 であってはなりません。ヒント値を指定すると、ヒントは結果に埋め込まれ、GETHINT 関数を使用して検索できます。パスワード・ストリング が指定されており、この引数が NULL 値であるか提供されない場合は、ヒントは結果に埋め込まれません。パスワード・ストリング が指定されていない場合は、SET ENCRYPTION PASSWORD ステートメントを使用してヒントを指定できます。

結果のデータ・タイプは、以下の表に示すとおり、最初の引数によって決まります。

最初の引数のデータ・タイプ	結果のデータ・タイプ
BINARY または VARBINARY	VARBINARY
CHAR、VARCHAR、GRAPHIC、または VARGRAPHIC	VARCHAR FOR BIT DATA
BLOB、CLOB、または DBCLOB	BLOB

結果の長さ属性は、指定される引数によって次のように異なります。

- パスワード・ストリング を指定し、ヒント・ストリング を指定しない場合、データ・ストリング の長さ属性に 24 を足し、それに 8 バイト境界までのバイト数を足したものになります。

- それ以外の場合は、データ・ストリング の長さ属性に 56 を足し、それに 8 バイト境界までのバイト数を足したものになります。

結果の実際の長さは、以下の合計になります。

- *data-string* の実際の長さに、8 バイト境界までのバイト数を足したもの。
- ヒントの実際の長さ。

  ヒント・ストリング が関数の引数として、または SET ENCRYPTION PASSWORD ステートメントに指定されていない場合、ヒントの実際の長さはゼロです。
- *n* - ここで *n* (値を暗号化するのに必要なオーバーヘッドの量) は 16 バイト (データ・ストリング が LOB である場合、またはデータ・ストリング、パスワード、またはヒントに別の CCSID 値を使用する場合は、24 バイト) です。

引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

暗号化された結果は、データ・ストリング 値より長くなることに注意してください。したがって、暗号化された値を割り当てる場合は、暗号化された値の全体を入れるのに十分なサイズでターゲットを宣言するようにしてください。

注

パスワード保護**:** 暗号化パスワードへの不用意なアクセスを避けるため、プログラム、プロシージャー、または関数のソースにパスワード・ストリング をストリング定数として指定しないでください。代わりに、SET ENCRYPTION PASSWORD ステートメントまたはホスト変数を使用してください。

リモート・リレーショナル・データベースに接続しているとき、指定されたパスワード自体は「平文で」送信されます。つまり、パスワード自体は暗号化されません。このようなケースでパスワードを保護するには、 IPSEC (または IBM i 製品同士の接続の場合は SSL) などの通信暗号化メカニズムを使用することを考慮してください。

暗号化アルゴリズム**:** 使用される内部暗号化アルゴリズムは、埋め込み処理を行う Triple DES ブロック暗号で、128 ビットの秘密鍵は、 SHA1 メッセージ要約を使用してパスワードから引き出されます。

暗号化パスワードおよびデータ**:** パスワードは、ユーザーが責任を持って管理します。データを暗号化すると、データを暗号化解除するのに使用できるのは暗号化に使用したパスワードだけです。 CHAR 変数を使用してパスワード値を設定する場合は、パスワード値にブランクが埋め込まれることがあるので注意してください。暗号化された結果には、NULL 終止符や他の印刷できない文字が含まれる場合があります。

表列の定義**:** 列および特殊タイプに暗号化されたデータが入るように定義する場合:

- CHAR FOR BIT DATA、VARCHAR FOR BIT DATA、BINARY、VARBINARY、または BLOB のデータ・タイプを使用して列を定義する必要があります。

- 列の長さ属性に追加の *n* バイトが含まれている必要があります。 *n* は、前述のとおり、データを暗号化するのに必要なオーバーヘッドです。

これらのデータ・タイプの 1 つがない列、または提案されたデータ長より短い長さの列に割り当てまたはキャストを行うと、割り当てエラーになる場合があります。あるいは、割り当てが成功すると、その後データを暗号化解除するときに失敗してデータが失われます。ブランクは暗号化されたデータとして有効ですが、短い列に保管される場合は切り捨てられます。

列の長さの計算例を以下に示します。

```
暗号化されていないデータの最大長          6 バイト
次の 8 バイト境界までのバイト数            2 バイト
オーバーヘッド                          16 バイト (または 24 バイト)
                                        --------
暗号化されたデータの列の長さ              24 バイト (または 32 バイト)


暗号化されていないデータの最大長          32 バイト
8 バイト境界までのバイト数                 0 バイト
オーバーヘッド                          16 バイト (または 24 バイト)
                                        --------
暗号化されたデータの列の長さ              48 バイト (または 56 バイト)
```

暗号化されたデータの管理**:** 暗号化されたデータは、 ENCRYPT_TDES 関数に対応する暗号化解除関数をサポートするサーバーでのみ暗号化解除できます。したがって、暗号化されたデータを含む列のレプリケーションは、暗号化解除関数をサポートするサーバーに対してのみ行う必要があります。

例

- 表 EMP1 に SSN という社会保障の列があると想定します。この例では、暗号化パスワードを保持するために ENCRYPTION PASSWORD 値を使用します。

```
SET ENCRYPTION PASSWORD = 'Ben123'

INSERT INTO EMP1 (SSN) VALUES ENCRYPT_TDES( '289-46-8832' )
```

- この例は、暗号化パスワードを明示的に受け渡しします。

```
INSERT INTO EMP1 (SSN) VALUES ENCRYPT_TDES( '289-46-8832', 'Ben123' )
```

- ヒント「Ocean」は、ユーザーが暗号化パスワード「Pacific」を思い出せるようにするために保管されます。

```
INSERT INTO EMP1 (SSN) VALUES ENCRYPT_TDES( '289-46-8832', 'Pacific', 'Ocean' )
```

EXP

EXP 関数は、自然対数の底 (e) を引数の指定だけ累乗した値を戻します。 EXP 関数と LN 関数は、逆の演算になります。

expression
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

引数のデータ・タイプが DECFLOAT(*n*) の場合、結果は DECFLOAT(*n*) です。それ以外の場合、結果のデータ・タイプは、倍精度の浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

注記

DECFLOAT 特殊値が関係する場合の結果: 10 進浮動小数点値の場合、特殊値は次のように扱われます。

- EXP(NaN) は NaN を返します。
- EXP(-NaN) は -NaN を返します。
- EXP(Infinity) は Infinity を返します。
- EXP(-Infinity) は 0 を返します。
- EXP(sNaN) および EXP(-sNaN) は警告またはエラーを返します。[57]

例

- ホスト変数 E は、値が 3.453789832 の DECIMAL(10、9) ホスト変数であると想定します。

```
SELECT EXP(:E)
  FROM SYSIBM.SYSDUMMY1
```

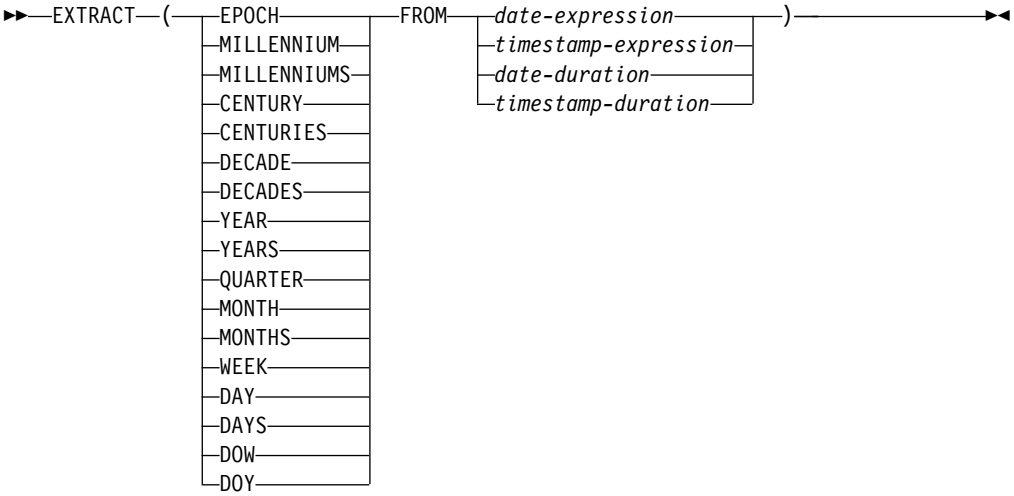
およそ 31.62 の値が戻されます。

57. SQL_DECFLOAT_WARNINGS 照会オプションに *YES を指定すると、NaN および -NaN がそれぞれ警告と一緒に返されます。

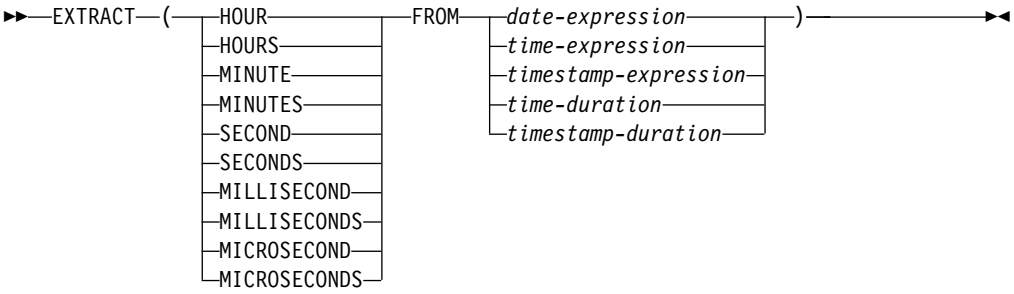
EXTRACT

EXTRACT 関数は、日時値の指定した部分を戻します。

日付値の抽出



時刻値の抽出



日付値の抽出

EPOCH

date-expression または *timestamp-expression* の、1970-01-01 00:00:00.00 からの秒数を戻すように指定します。値は正または負になります。これは、*date-duration* または *timestamp-duration* には指定できません。

MILLENNIUM または **MILLENNIUMS**

date-expression、*timestamp-expression*、*date-duration*、または *timestamp-duration* の年の、 1000 年を 1 期間とする期間が含まれる数を戻すことを指定します。例えば、2000-01-01 から 2999-12-31 までの日付の場合は 2 が戻されます。

CENTURY または **CENTURIES**

date-expression、*timestamp-expression*、*date-duration*、または *timestamp-duration* の年の、 100 年を 1 期間とする期間が含まれる数を戻すことを指定します。例えば、2000-01-01 から 2099-12-31 までの日付の場合は 20 が戻されます。

DECADE または **DECADES**

date-expression、*timestamp-expression*、*date-duration*、または *timestamp-duration*

の年の、 10 年を 1 期間とする期間が含まれる数を戻すことを指定します。例えば、2010-01-01 から 2019-12-31 までの日付の場合は 201 が戻されます。

YEAR または YEARS
date-expression、*timestamp-expression*、*date-duration*、または *timestamp-duration* の年の部分を戻すことを指定します。結果は、YEAR スカラー関数と同じです。詳しくは、 737 ページの『YEAR』を参照してください。

QUARTER
date-expression または *timestamp-expression* の四半期 (1 - 4) を戻すことを指定します。結果は、QUARTER スカラー関数と同じです。詳しくは、 583 ページの『QUARTER』を参照してください。これは、*date-duration* または *timestamp-duration* には指定できません。

MONTH または MONTHS
date-expression、*timestamp-expression*、*date-duration*、または *timestamp-duration* の月の部分を戻すことを指定します。結果は、MONTH スカラー関数と同じです。詳しくは、 549 ページの『MONTH』を参照してください。

WEEK
date-expression または *timestamp-expression* の年間通算週 (1 - 53) を戻すことを指定します。週は月曜日から始まります。結果は、WEEK_ISO スカラー関数と同じです。詳しくは、 699 ページの『WEEK_ISO』を参照してください。これは、*date-duration* または *timestamp-duration* には指定できません。

DAY または DAYS
date-expression、*timestamp-expression*、*date-duration* または *timestamp-duration* の日の部分を戻すことを指定します。結果は、DAY スカラー関数と同じです。詳しくは、 407 ページの『DAY』を参照してください。

DOW
date-expression または *timestamp-expression* の曜日 (1 が日曜日、7 が土曜日を表す) を戻すことを指定します。結果は、DAYOFWEEK スカラー関数と同じです。詳しくは、 410 ページの『DAYOFWEEK』を参照してください。これは、*date-duration* または *timestamp-duration* には指定できません。

DOY
date-expression または *timestamp-expression* の年間通算日 (1 - 366) を戻すことを指定します。結果は、DAYOFYEAR スカラー関数と同じです。詳しくは、 412 ページの『DAYOFYEAR』を参照してください。これは、*date-duration* または *timestamp-duration* には指定できません。

date-expression
組み込み日付データ・タイプ、組み込み文字ストリング・データ・タイプ、組み込みグラフィック・ストリング・データ・タイプのいずれかの値を戻す式。

date-expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付の有効な文字ストリング表現またはグラフィック・ストリング表現でなければなりません。日付の有効なストリング表現の形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

timestamp-expression
組み込みタイム・スタンプ・データ・タイプ、組み込み文字ストリング・データ・タイプ、組み込みグラフィック・ストリング・データ・タイプのいずれかの値を戻す式。

timestamp-expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、タイム・スタンプの有効な文字ストリング表現またはグラフィック・ストリング表現でなければなりません。タイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

date-duration
DECIMAL(8,0) の数値として表現される日付期間。日時期間の有効な形式については、 205 ページの『日付/時刻のオペランドと期間』を参照してください。

timestamp-duration
DECIMAL(14+*s*,*s*) の数値として表現されるタイム・スタンプ期間。ここで、*s* は、0 から 12 までの小数秒桁数です。日時期間の有効な形式については、205 ページの『日付/時刻のオペランドと期間』を参照してください。

時刻値の抽出

HOUR または **HOURS**
time-expression、*timestamp-expression*、*time-duration*、または *timestamp-duration* の時間の部分を戻すことを指定します。*date-expression* には 0 を戻します。 結果は、HOUR スカラー関数と同じです。詳しくは、 486 ページの『HOUR』を参照してください。

MINUTE または **MINUTES**
time-expression、*timestamp-expression*、*time-duration*、または *timestamp-duration* の分の部分を戻すことを指定します。*date-expression* には 0 を戻します。 結果は、MINUTE スカラー関数と同じです。詳しくは、 546 ページの『MINUTE』を参照してください。

SECOND または **SECONDS**
time-expression、*timestamp-expression*、*time-duration*、または *timestamp-duration* の秒の部分を戻すことを指定します。*date-expression* には 0 を戻します。 結果は以下と等しくなります。

- expression のデータ・タイプが TIME 値、TIME または TIMESTAMP のストリング表記、または時刻期間の場合、SECOND(expression, 6)。
- expression のデータ・タイプが TIMESTAMP(*s*) 値、またはタイム・スタンプ期間の場合、SECOND(expression, *s*)。

詳しくは、 623 ページの『SECOND』を参照してください。

MILLISECOND または **MILLISECONDS**
timestamp-expression または *timestamp-duration* の秒数 (1000 分の 1 秒までの小数部分を含む) に 1000 を掛けた値を戻すことを指定します (0 から 59999)。*date-expression*、*time-expression*、または *time-duration* には 0 を戻します。

MICROSECOND または **MICROSECONDS**
timestamp-expression または *timestamp-duration* の秒数 (100 万分の 1 秒までの

小数部分を含む) に 1000000 を掛けた値を戻すことを指定します (0 から 59999999)。 *date-expression*、*time-expression*、または *time-duration* には 0 を戻します。

date-expression
組み込み日付データ・タイプ、組み込み文字ストリング・データ・タイプ、組み込みグラフィック・ストリング・データ・タイプのいずれかの値を戻す式。

date-expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付の有効な文字ストリング表現またはグラフィック・ストリング表現でなければなりません。*expression* が日付の有効なストリング表現の場合、その形式は IBM SQL 標準形式のいずれかでなければなりません。日付の有効なストリング表現の形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

time-expression
組み込み時刻データ・タイプ、組み込み文字ストリング・データ・タイプ、組み込みグラフィック・ストリング・データ・タイプのいずれかの値を戻す式。

time-expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、時刻の有効な文字ストリング表現またはグラフィック・ストリング表現でなければなりません。時刻の有効なストリング表現の形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

timestamp-expression
組み込みタイム・スタンプ・データ・タイプ、組み込み文字ストリング・データ・タイプ、組み込みグラフィック・ストリング・データ・タイプのいずれかの値を戻す式。

timestamp-expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、タイム・スタンプの有効な文字ストリング表現またはグラフィック・ストリング表現でなければなりません。タイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

time-duration
DECIMAL(6,0) の数値として表現される時刻期間。日時期間の有効な形式については、 205 ページの『日付/時刻のオペランドと期間』を参照してください。

timestamp-duration
DECIMAL(14+*s*,*s*) の数値として表現されるタイム・スタンプ期間。ここで、*s* は、0 から 12 までの小数秒桁数です。日時期間の有効な形式については、205 ページの『日付/時刻のオペランドと期間』を参照してください。

関数の結果のデータ・タイプは、指定された日時値の部分によって、次のように決まります。

- EPOCH を指定した場合、結果のデータ・タイプは BIGINT になります。
- MILLENNIUM、CENTURY、DECADE、YEAR、QUARTER、MONTH、WEEK、DAY、DOW、DOY、HOUR、MINUTE、MILLISECOND、または MICROSECOND を指定した場合、結果のデータ・タイプは INTEGER になります。

- SECOND が TIMESTAMP(*p*) 値と一緒に指定された場合、結果のデータ・タイプは DECIMAL(2+*p*, *p*) になります。ここで、*p* は小数部分の秒数の精度です。
- SECOND が TIME 値か、TIME または TIMESTAMP のストリング表記と一緒に指定された場合、結果のデータ・タイプは DECIMAL(8,6) になります。

引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

- 列 PRSTDATE には、1988-12-25 に相当する内部値が入っていると想定します。

```
SELECT EXTRACT( MONTH FROM PRSTDATE )
  FROM PROJECT
```

  このステートメントは、整数値 12 を返します。
- タイム・スタンプ・グローバル変数 GV1 に値 '2007-02-14 12:15:06.123456' が入っていると想定します。

```
VALUES EXTRACT(MILLISECONDS FROM GV1);
```

  このステートメントは、整数値 6123 を返します。
- タイム・スタンプ・グローバル変数 GV1 に値 '2007-02-14 12:15:06.123456' が入っていると想定します。

```
VALUES EXTRACT(MICROSECONDS FROM GV1);
```

  このステートメントは、整数値 6123456 を返します。
- 日付グローバル変数 GV2 に値 '2013-02-14' が入っていると想定します。

```
VALUES EXTRACT(DECADE FROM GV2);
```

  このステートメントは、整数値 201 を返します。
- decimal(6,0) グローバル変数 GV3 に値 123020 が入っていると想定します。

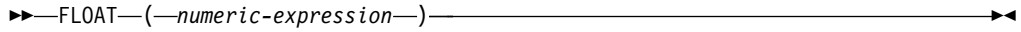
```
VALUES EXTRACT(SECONDS FROM GV3);
```

  このステートメントは、整数値 20 を返します。

FLOAT

FLOAT 関数は、数値またはストリングの浮動小数点数表現を戻します。

数値から浮動小数点数に

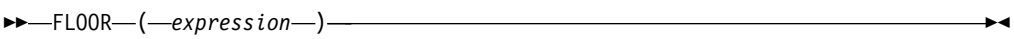


ストリングから浮動小数点に

FLOAT は、DOUBLE_PRECISION および DOUBLE 関数の同義語です。詳しくは、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

FLOOR

FLOOR 関数は、式 に等しいか数値式より小さい最大の整数を戻します。



expression
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

関数の結果のデータ・タイプと長さ属性は引数と同じになります。ただし、引数が 10 進数の場合の位取りは 0 になります。例えば、データ・タイプが DECIMAL(5,5) の引数の場合、結果は DECIMAL(5,0) となります。

引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

注記

DECFLOAT 特殊値が関係する場合の結果**:** 10 進浮動小数点値の場合、特殊値は次のように扱われます。

- FLOOR(NaN) は NaN を返します。
- FLOOR(-NaN) は -NaN を返します。
- FLOOR(Infinity) は Infinity を返します。
- FLOOR(-Infinity) は -Infinity を返します。
- FLOOR(sNaN) および FLOOR(-sNaN) は警告またはエラーを返します。[58]

例

- 小数点の右側の桁をすべて切り捨てるために、FLOOR 関数を使用します。

```
SELECT FLOOR(SALARY)
  FROM EMPLOYEE
```

- 正負両方の数値に関して FLOOR を使用します。

```
SELECT FLOOR( 3.5),
       FLOOR( 3.1),
       FLOOR(-3.1),
       FLOOR(-3.5)
  FROM SYSIBM.SYSDUMMY1
```

この例ではそれぞれ、

```
3.   3.  -4.  -4.
```

が戻されます。

58. SQL_DECFLOAT_WARNINGS 照会オプションに *YES を指定すると、NaN および -NaN がそれぞれ警告と一緒に返されます。

GENERATE_UNIQUE

GENERATE_UNIQUE 関数は、同じ関数の他の実行と比較して固有である、13 バイトの長さのビット・データ文字ストリング (CHAR(13) FOR BIT DATA) を戻します。この関数は、非決定論的であると定義されます。

関数の結果は、世界時の内部形式、協定世界時 (UTC)、およびシステム・シリアル番号を含む固有値です。結果が NULL になることはありません。

この関数の結果を使用して表中の固有値を提供することができます。各値は前の値より大きく、表内で使用できる順序を提供します。シーケンスは、関数が実行された時刻に基づきます。

この関数は、特殊レジスター CURRENT TIMESTAMP を使用する場合と異なり、同じ SQL ステートメント内でもこの関数のインスタンスごとに固有値が生成され、また、複数行 insert ステートメント、全選択のある insert ステートメント、または MERGE ステートメント内の insert ステートメントでは、行ごとに固有値が生成されます。

この関数の結果の一部であるタイム・スタンプ値は、GENERATE_UNIQUE の結果を引数として TIMESTAMP 関数を使用して決定することができます。

例

- 各行で固有の列を含む表を作成します。GENERATE_UNIQUE 関数を使用してこの列を移植します。 UNIQUE_ID 列をビット・データ文字ストリングとして識別するために FOR BIT DATA と定義していることに注意してください。

```
CREATE TABLE EMP_UPDATE
  (UNIQUE_ID CHAR(13) FOR BIT DATA,
   EMPNO CHAR(6),
   TEXT VARCHAR(1000))
INSERT INTO EMP_UPDATE VALUES (GENERATE_UNIQUE(),
                               '000020',
                               'Update entry 1...')
INSERT INTO EMP_UPDATE VALUES (GENERATE_UNIQUE(),
                               '000050',
                               'Update entry 2...')
```

  UNIQUE_ID 列が常に GENERATE_UNIQUE を使用して設定される限り、この表は各行に対して固有 ID を持つことになります。これは、表にトリガーを導入して行うことができます。

```
CREATE TRIGGER  EMP_UPDATE_UNIQUE
  NO CASCADE BEFORE INSERT ON EMP_UPDATE
  REFERENCING NEW AS NEW_UPD
  FOR EACH ROW MODE DB2SQL
  SET NEW_UPD.UNIQUE_ID = GENERATE_UNIQUE()
```

  このトリガーにより、表にデータを設定するために以前に使用された INSERT ステートメントを、以下のように UNIQUE_ID 列に値を指定しないで発行することができます。

```
INSERT INTO EMP_UPDATE(EMPNO, TEXT) VALUES ('000020', 'Update entry 1...')
INSERT INTO EMP_UPDATE(EMPNO, TEXT) VALUES ('000050', 'Update entry 2...')
```

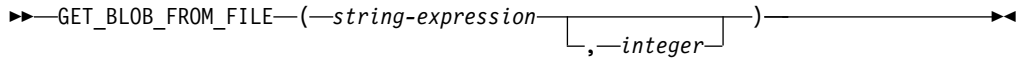
EMP_UPDATE にいつ行が加えられたかを示すタイム・スタンプ (UTC で) は、以下を使用すると戻されます。

```
SELECT TIMESTAMP(UNIQUE_ID), EMPNO, TEXT FROM EMP_UPDATE
```

したがって、行がいつ挿入されたかを記録するタイム・スタンプ列は表には必要ありません。

GET_BLOB_FROM_FILE

GET_BLOB_FROM_FILE 関数は、ソース・ストリーム・ファイルまたはソース物理ファイルからデータを戻します。



string-expression
この引数は、パスおよびファイル名を指定するストリング式でなければなりません。このファイル名は、ソース・ストリーム・ファイルまたはソース物理ファイルの名前にできます。ソース物理ファイルの場合、このストリングは「library/file(member)」という形式にしてください。

integer
指定したファイルがソース物理ファイルの場合に、末尾の空白の処理方法を示す整数定数。有効な値は以下のとおりです。

0
: ファイル内に空白がある場合に、空白を戻します。

1
: 末尾の空白 (ブランク) は、そのうちの最初のブランク以外が除去されます。

ソース物理ファイルの場合にこの引数を指定しないと、デフォルトは 0 になります。ソース・ストリーム・ファイルに指定しても、無視されます。

この関数の結果は、BLOB ロケーターになります。

この関数は、CCSID 変換せずにこの引数によって指定されたファイルを読み取り、それを BLOB ロケーターとして戻します。この関数は、コミットメント制御下で実行しなければなりません。ロケーターは、COMMIT または ROLLBACK が実行されると解放されます。

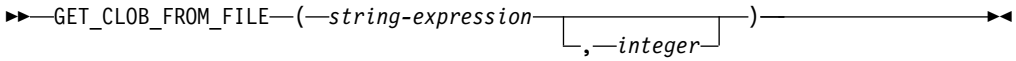
例

XML スキーマ文書を、その XML スキーマがソース・ストリーム・ファイル内にある XSR レジストリーに登録します。

```
CALL XSR_REGISTER ('myschemalib', 'myschema', NULL,
                   GET_BLOB_FROM_FILE('/home/XML/MySchema.XSD',0), NULL)
```

GET_CLOB_FROM_FILE

GET_CLOB_FROM_FILE 関数は、ソース・ストリーム・ファイルまたはソース物理ファイルからデータを戻します。



string-expression
この引数は、パスおよびファイル名を指定するストリング式でなければなりません。このファイル名は、ソース・ストリーム・ファイルまたはソース物理ファイルの名前にできます。ソース物理ファイルの場合、このストリングは「library/file(member)」という形式にしてください。

integer
指定したファイルがソース物理ファイルの場合に、末尾の空白の処理方法を示す整数定数。有効な値は以下のとおりです。

0 ファイル内に空白がある場合に、空白を戻します。
1 末尾の空白 (ブランク) は、そのうちの最初のブランク以外が除去されます。

ソース物理ファイルの場合にこの引数を指定しないと、デフォルトは 0 になります。ソース・ストリーム・ファイルに指定しても、無視されます。

この関数の結果は、CLOB ロケーターになります。

この関数は、引数によって指定されたファイルを読み取り、デフォルト・ジョブの CCSID にデータを変換し、それを CLOB ロケーターとして戻します。この関数は、コミットメント制御下で実行しなければなりません。ロケーターは、COMMIT または ROLLBACK が実行されると解放されます。

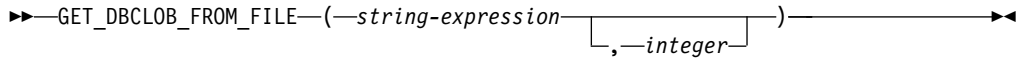
例

ソース・ファイルに含まれているデータを、ホスト変数 HV1 に割り当てます。データはデフォルト・ジョブ CCSID に変換され、ソース・ファイルの各行から末尾ブランクが除去されます。

```
SET :HV1 = GET_CLOB_FROM_FILE('MYLIB/MYFILE(MYMBR)',1)
```

GET_DBCLOB_FROM_FILE

GET_DBCLOB_FROM_FILE 関数は、ソース・ストリーム・ファイルまたはソース物理ファイルからデータを戻します。



string-expression
この引数は、パスおよびファイル名を指定するストリング式でなければなりません。このファイル名は、ソース・ストリーム・ファイルまたはソース物理ファイルの名前にできます。ソース物理ファイルの場合、このストリングは「library/file(member)」という形式にしてください。

integer
指定したファイルがソース物理ファイルの場合に、末尾の空白の処理方法を示す整数定数。有効な値は以下のとおりです。

0
: ファイル内に空白がある場合に、空白を戻します。

1
: 末尾の空白 (ブランク) は、そのうちの最初のブランク以外が除去されます。

ソース物理ファイルの場合にこの引数を指定しないと、デフォルトは 0 になります。ソース・ストリーム・ファイルに指定しても、無視されます。

この関数の結果は、DBCLOB ロケーターになります。

この関数は、引数によって指定されたファイルを読み取り、デフォルト CCSID に関連付けられている 2 バイト CCSID にデータを変換し、それを DBCLOB ロケーターとして戻します。この関数は、コミットメント制御下で実行しなければなりません。ロケーターは、COMMIT または ROLLBACK が実行されると解放されます。

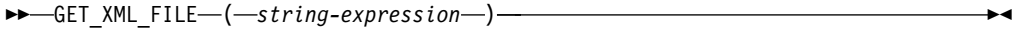
例

ソース・ストリーム・ファイルに含まれているデータを、ホスト変数 HV1 に割り当てます。データは、デフォルト CCSID に関連付けられている 2 バイト CCSID に変換されます。

```
SET :HV1 = GET_DBCLOB_FROM_FILE('/home/XML/MySchema.XSD')
```

GET_XML_FILE

GET_XML_FILE 関数は、ソース・ストリーム・ファイルまたはソース物理ファイルからデータを戻します。



string-expression
この引数は、パスおよびファイル名を指定するストリング式でなければなりません。このファイル名は、ソース・ストリーム・ファイルまたはソース物理ファイルの名前にできます。ソース物理ファイルの場合、このストリングは「library/file(member)」という形式にしてください。

この関数の結果は、BLOB ロケーターになります。

この関数は、引数によって指定されたファイルを読み取り、データを UTF-8 に変換します。ファイルに XML 宣言が含まれていない場合には、追加されます。BLOB ロケーターとして戻されます。この関数は、コミットメント制御下で実行しなければなりません。ロケーターは、COMMIT または ROLLBACK が実行されると解放されます。

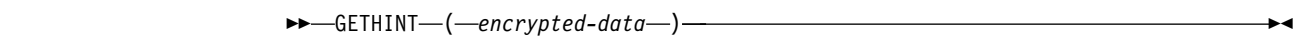
例

XML スキーマ文書を、その XML スキーマがソース・ストリーム・ファイル内にある XSR レジストリーに登録します。

```
CALL XSR_REGISTER ('myschemalib', 'myschema', NULL,
                   GET_XML_FILE('/home/XML/MySchema.XSD'), NULL)
```

GETHINT

GETHINT 関数は、暗号化されたデータ の中でパスワード・ヒントが検出されたら、それを戻します。パスワード・ヒントは、データ所有者がパスワードを思い出すのに役立つ句です (例えば、「Ocean」は「Pacific」を思い出すためのヒント)。



encrypted-data
CHAR FOR BIT DATA、VARCHAR FOR BIT DATA、BINARY、VARBINARY、または BLOB 組み込みデータ・タイプの完全な暗号化されたデータ値を戻すストリング式。データ・ストリングは、ENCRYPT_RC2 または ENCRYPT_TDES 関数を使用して暗号化しておく必要があります。

この結果のデータ・タイプは、VARCHAR(32) です。結果の実際の長さは、データが暗号化されたときに提供されたヒントの実際の長さです。

結果が、NULL になることもあります。引数が NULL の場合、または ENCRYPT_RC2 か ENCRYPT_TDES 関数によってヒントが 暗号化されたデータに追加されなかった場合は、結果は NULL 値になります。

結果の CCSID は、現行サーバーのデフォルトの CCSID になります。

例

- ヒント「Ocean」は、ユーザーが暗号化パスワード「Pacific」を思い出せるようにするために保管されます。

```
INSERT INTO EMP1 (SSN) VALUES ENCRYPT_RC2( '289-46-8832', 'Pacific', 'Ocean' )

SELECT GETHINT( SSN )
  FROM EMP1
```

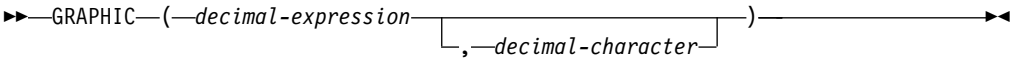
  GETHINT 関数は、元のヒント値「Ocean」を戻します。

GRAPHIC

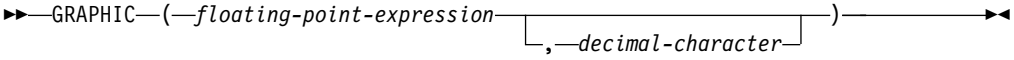
GRAPHIC 関数は、ストリング式の固定長グラフィック文字表現を戻します。

整数からグラフィックに

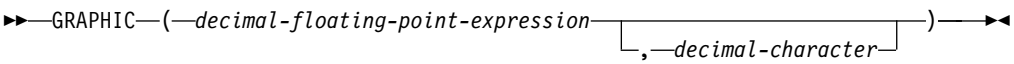
10 進数からグラフィックに



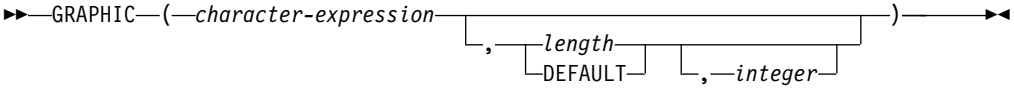
浮動小数点数からグラフィックに



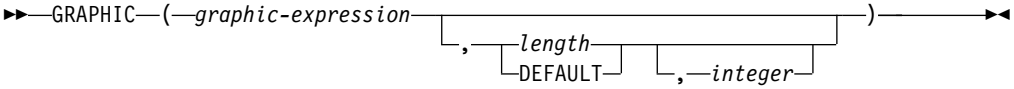
10 進浮動小数点数からグラフィックに



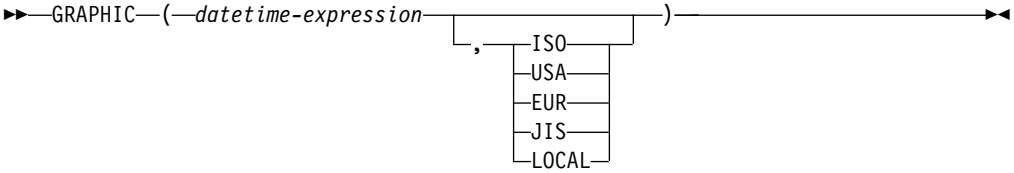
文字からグラフィックに



グラフィックからグラフィックへ



日時からグラフィックに



GRAPHIC 関数は、次のもののグラフィック・ストリング表現を返します。

- 整数 (最初の引数が SMALLINT、INTEGER、または BIGINT の場合)
- 10 進数 (最初の引数がパックまたはゾーン 10 進数の場合)
- 倍精度浮動小数点数 (最初の引数が DOUBLE または REAL の場合)

- 最初の引数が DECFLOAT である場合は 10 進浮動小数点数。
- 文字ストリング (最初の引数が任意のタイプの文字ストリングの場合)
- グラフィック・ストリング (最初の引数が任意のタイプのグラフィック・ストリングの場合)
- 日付値 (最初の引数が DATE の場合)
- 時刻値 (最初の引数が TIME の場合)
- タイム・スタンプ値 (最初の引数が TIMESTAMP)

この関数の結果は固定長グラフィック・ストリング (GRAPHIC) です。

最初の引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。最初の引数が NULL の場合、結果は NULL 値となります。

整数からグラフィックに

integer-expression
整数データ・タイプ (SMALLINT、INTEGER、または BIGINT) の値を戻す式。

結果は、SQL 整数定数の形式で引数を表現した固定長グラフィック・ストリングです。結果は、引数の値を表す n 文字の有効数字から成ります。引数が負数の場合は、負符号が前に付きます。結果は左寄せされます。

- 引数が短整数の場合は、結果の長さ属性は 6
- 引数が長整数の場合は、結果の長さ属性は 11
- 引数が 64 ビット整数の場合は、結果の長さ属性は 20

結果は、引数の値を表すために使用できる最小文字数です。先行ゼロは含まれません。引数が負数の場合は、結果の最初の文字は負符号になります。負符号でなければ、最初の文字は数字または *decimal-character* です。

結果の CCSID は 1200 (UTF-16) です。

10 進数からグラフィックに

decimal-expression
パックまたはゾーン 10 進数データ・タイプ (DECIMAL または NUMERIC) の値を戻す式。精度や位取りを変えたい場合は、DECIMAL スカラー関数を使用して変更することができます。

decimal-character
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

結果は、引数を固定長のグラフィック・ストリングで表現したものになります。この結果には、1 文字の小数点文字と *p* 桁までの数字が含まれます。 *p* は *10* 進数式 の精度で、引数が負数の場合は負符号が先頭に付きます。先行ゼロは戻されません。後続ゼロは戻されます。*decimal-expression* の位取りがゼロの場合、小数点文字は戻されません。

結果の長さ属性は 2+*p* です (*p* は *decimal-expression* の精度)。結果は、結果を表すために使用できる最小文字数です。先行ゼロは含まれません。引数が負数の場合は、結果の最初の文字は負符号になります。負符号でなければ、結果の最初の文字は数字または *decimal-character* になります。

結果の CCSID は 1200 (UTF-16) です。

浮動小数点数からグラフィックに

浮動小数点数式
浮動小数点データ・タイプ (DOUBLE または REAL) の値を戻す式。

decimal-character
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

結果は、浮動小数点定数の形式で引数を固定長グラフィック・ストリングで表現したものになります。

結果の長さ属性は、24 です。結果は、ゼロ以外の 1 桁の数字、その後ろに 1 つの小数点文字 と一連の数字が続く小数部の引数の値を表す最小文字数です。引数が負数の場合は、結果の最初の文字は負符号になります。負数でなければ、最初の文字は数字または *decimal-character* です。引数がゼロであると、結果は 0E0 になります。

結果の CCSID は 1200 (UTF-16) です。

10 進浮動小数点数からグラフィックに

decimal-floating-point expression
組み込みの 10 進浮動小数点データ・タイプの値を返す式。

decimal-character
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

結果は、浮動小数点定数の形式で引数を固定長グラフィック・ストリングで表現したものになります。

結果の長さ属性は、42 です。 結果の実際の長さは、記号、数字、および *decimal-character* を含む、引数の値を表す最小文字数です。後続ゼロは有効数字です。引数が負数の場合は、結果の最初の文字は負符号になります。負数でなければ、最初の文字は数字または *decimal-character* です。引数がゼロであると、結果は 0 になります。

DECFLOAT 値が Infinity、sNaN、または NaN の場合、それぞれストリング 'INFINITY'、'SNAN'、および 'NAN' が返されます。特殊値が負の場合、負符号 (-) がそのストリングの最初の文字となります。 DECFLOAT 特殊値 sNaN を使用しても、ストリングへの変換時に例外は発生しません。

結果の CCSID は 1200 (UTF-16) です。

文字からグラフィックに

character-expression
組み込み文字ストリング・データ・タイプである値を戻す式。 CHAR または VARCHAR ビット・データであってはなりません。式が空ストリング、または EBCDIC ストリング `X'0E0F'` である場合、結果は単一の 2 バイト・ブランクになります。

length
結果の長さ属性を指定する整数定数。これは、最初の引数がNULL 可能でない場合は、1 から 16383 の範囲の整数定数でなければならず、最初の引数が NULL 可能である場合は、1 から 16382 の範囲の整数定数でなければなりません。文字式 の長さが指定の長さより小さい場合、結果は、結果に指定されている長さまで 2 バイト・ブランクで埋め込まれます。

長さ の指定がない場合、または DEFAULT が指定されている場合は、結果の長さ属性は、最初の引数の長さ属性と同じになります。

引数の各文字ごとに、結果の 1 文字が決まります。結果の固定長ストリングの長さ属性が最初の引数の実際の長さより小さい場合は、切り捨てが行われ、警告が戻されることはありません。

integer
結果の CCSID を指定する整数定数。これは DBCS、UTF-16、または UCS-2 の CCSID でなければなりません。 CCSID が 65535 であることはできません。 CCSID が Unicode グラフィック・データを表す場合は、引数の各文字によって結果の 1 文字が決まります。結果の n 番目の文字は、引数の n 番目の文字と等価の UTF-16 または UCS-2 文字になります。

整数 が指定されていない場合、結果の CCSID は混合 CCSID によって決まります。 M でその混合 CCSID を示すことにします。

以下の規則では、S は次のいずれかを指します。

- ストリング式が外部コード化スキームのデータを含むホスト変数である場合は、データを固有コード化スキームの CCSID に変換した後の式の結果が S。 (詳しくは、 39 ページの『文字変換』を参照してください。)
- ストリング式が固有コード化スキームのデータである場合は、そのストリング式が S。

M は次のように決まります。

- S の CCSID が混合 CCSID である場合は、M はその CCSID になる。
- S の CCSID が SBCS CCSID である場合 :
  - S の CCSID が関連する混合 CCSID を持つ場合は、M はその CCSID になる。
  - それ以外の場合は、演算ができない。

次の表には、M をもとにした結果の CCSID を要約してあります。

M	結果の **CCSID**	説明	**DBCS** 置換文字
930	300	日本語 EBCDIC	X'FEFE'
933	834	韓国語 EBCDIC	X'FEFE'

M	結果の CCSID	説明	DBCS 置換文字
935	837	中国語 (簡体字) EBCDIC	X'FEFE'
937	835	中国語 (繁体字) EBCDIC	X'FEFE'
939	300	日本語 EBCDIC	X'FEFE'
5026	4396	日本語 EBCDIC	X'FEFE'
5035	4396	日本語 EBCDIC	X'FEFE'

SBCS と DBCS が等価になるかどうかは M によって決まります。CCSID に関係なく、引数の中の 2 バイトのコード・ポイントはすべて DBCS 文字と見なされ、引数の中の 1 バイトのコード・ポイントはすべて SBCS 文字と見なされます (ただし、EBCDIC 混合データのシフト・コード X'0E' および X'0F' は例外)。

- 引数の n 番目の文字が DBCS 文字である場合は、結果の n 番目の文字はその DBCS になる。
- 引数の n 番目の文字が、等価の DBCS 文字を持つ SBCS 文字である場合は、結果の n 番目の文字は、その等価の DBCS 文字になる。
- 引数の n 番目の文字が、等価の DBCS 文字を持たない SBCS 文字である場合は、結果の n 番目の文字は、DBCS 置換文字になる。

グラフィックからグラフィックに

graphic-expression
組み込みグラフィック・ストリング・データ・タイプの値を戻す式。

length
結果の長さ属性を指定する整数定数。これは、最初の引数がNULL 可能でない場合は、1 から 16383 の範囲の整数定数でなければならず、最初の引数が NULL 可能である場合は、1 から 16382 の範囲の整数定数でなければなりません。グラフィック式 の長さが指定の長さより小さい場合、結果は、結果に指定されている長さまで 2 バイト・ブランクで埋め込まれます。

2 番目の引数の指定がない場合、または DEFAULT が指定されている場合は、結果の長さ属性は、最初の引数の長さ属性と同じになります。

グラフィック式 の長さが結果の長さ属性よりも大きい場合は、切り捨てが行われます。切り捨てられた文字がすべてブランクであった場合以外は、警告 (SQLSTATE 01004) が戻されます。

integer
結果の CCSID を指定する整数定数。これは DBCS、UTF-16、または UCS-2 の CCSID でなければなりません。 CCSID が 65535 であることはできません。

整数 が指定されていない場合、結果の CCSID は、最初の引数の CCSID になります。

日時からグラフィックに

datetime-expression
次の 3 つの組み込みデータ・タイプのいずれかである式。

日付　結果は、2 番目の引数によって指定された形式の日付の GRAPHIC ストリング表記になります。 2 番目の引数を指定しなかった場合は、デフォルトの日付形式が使用されます。形式として ISO、USA、 EUR、または JIS を指定すると、結果の長さは 10 になります。その他の場合は、結果の長さはデフォルトの日付形式の長さになります。詳しくは、95 ページの『日付/時刻の値のストリング表記』を参照してください。

時刻　結果は、2 番目の引数によって指定された形式の時刻の GRAPHIC ストリング表記になります。 2 番目の引数を指定しなかった場合は、デフォルトの時刻形式が使用されます。結果の長さは 8 です。詳しくは、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

timestamp

2 番目の引数は適用されないので、指定してはなりません。

結果は、タイム・スタンプの GRAPHIC ストリング表記になります。*datetime-expression* が TIMESTAMP(0) である場合、結果の長さは 19 になります。*datetime-expression* のデータ・タイプが TIMESTAMP(*n*) の場合、結果の長さは 20+*n* です。これ以外の場合、結果の長さは 26 となります。

結果の CCSID は 1200 (UTF-16) です。

ISO、EUR、USA、または JIS

結果のグラフィック・ストリングの日付形式または時刻形式を指定します。詳しくは、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

LOCAL

結果のグラフィック・ストリングの日付または時刻の形式を、現行サーバーのジョブの DATFMT、DATSEP、 TIMFMT、および TIMSEP 属性から取る必要があることを指定します。

注記

代替構文**:** 最初の引数がストリングで、長さ属性を指定する場合、アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、 218 ページの『CAST の指定』を参照してください。

例

- 表 EMPLOYEE を使用して、ホスト変数 DESC (GRAPHIC(24)) を従業員番号 (EMPNO)「000050」に対応する氏名の名 (FIRSTNME) と等価の GRAPHIC にセットします。

```
SELECT GRAPHIC( VARGRAPHIC(FIRSTNME))
  INTO :DESC
  FROM EMPLOYEE
  WHERE EMPNO = '000050'
```

HASH_MD5、HASH_SHA1、 HASH_SHA256、および HASH_SHA512

ハッシュ関数は、選択したアルゴリズムに応じて、入力データの 128 ビット、160 ビット、256 ビット、または 512 ビットのハッシュを返します。

expression
ハッシュするストリング値を表す式。この式は、任意の組み込みデータ・タイプまたは特殊データ・タイプを戻すことができます。特殊タイプは、そのソース・データ・タイプとして扱われます。数値または日時の値は、関数の評価の前に VARCHAR に暗黙的にキャストされます。値が XML の場合、関数の評価の前に暗黙的に XMLSERIALIZE to CLOB(2G) CCSID 1208 が実行されます。

表 51 は、サポートされる各アルゴリズムに関する情報を示しています。

表 *51.* ハッシュ・アルゴリズム

関数名	アルゴリズム	結果のサイズ	戻り値の数	結果の長さ	**HASH** 関数で使用される対応アルゴリズム値
HASH_MD5	MD5	128 ビット	2^{128}	16	0
HASH_SHA1	SHA1	160 ビット	2^{160}	20	1
HASH_SHA256	SHA-256	256 ビット	2^{256}	32	2
HASH_SHA512	SHA-512	512 ビット	2^{512}	64	3

結果のデータ・タイプは BINARY で、結果の長さは 表 51 に示したように、関数名によって決まります。

引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL の場合、結果は NULL 値になります。

注

空白文字はハッシュに影響します。末尾にブランクを含む固定長文字ストリングでは、末尾にブランクを含まない可変長文字ストリングと異なる結果が生成されます。 *expression* の CCSID が原因で、同等と評価されるストリングが、異なる結果値を生成することがあります。

SHA1 と MD5 のどちらのアルゴリズムでもセキュリティーの欠陥が見つかっています。コンプライアンスに関する関連資料 (National Institute of Standards and Technology (NIST) Special Publication 800-131A など) で、使用できるハッシュ・アルゴリズムを確認してください。

代替の構文**:** 引数を 1 つ指定した HASH 関数は、HASH_MD5 に似ています。HASH に 2 番目の引数を指定して、使用するアルゴリズムを指示することができます。アルゴリズム値は、表 51 に示されています。 2 番目の引数は、任意の組み

込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を必ず戻す式にすることができます。ストリング引数は、関数を評価する前に整数に変換されます。 HASH の結果データ・タイプは VARBINARY です。引数が 1 つのみの場合、結果の長さ属性は 16 です。 2 番目の引数を整数定数で指定すると、結果の長さ属性は、 480 ページの表 51 で示された結果の長さになります。それ以外の場合、結果の長さ属性は 64 です。どちらかの引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数のどちらかが NULL である場合は、結果は NULL 値になります。

例

- MD5 アルゴリズムを使用して、ハッシュ・データを生成します。

```
VALUES HEX(HASH_MD5('ABCDEFGHIJKLMNOPQRSTUVWXYZ'))
```

  以下の値が返されます。

```
5156BECBC019E3F0F9520B143435427E
```

- SHA1 アルゴリズムを使用して、ハッシュ・データを生成します。

```
VALUES HEX(HASH_SHA1('ABCDEFGHIJKLMNOPQRSTUVWXYZ'))
```

  以下の値が返されます。

```
55324E3DD1FA95040F65709D193E82575237EF86
```

- SHA-256 アルゴリズムを使用して、ハッシュ・データを生成します。

```
VALUES HEX(HASH_SHA256('ABCDEFGHIJKLMNOPQRSTUVWXYZ'))
```

  以下の値が返されます。

```
011F7AD1ECD8E5A4CC8533D1ECD497DC5D95E848B14F8BCFD56A73D7F41843E2
```

- SHA-512 アルゴリズムを使用して、ハッシュ・データを生成します。

```
VALUES HEX(HASH_SHA512('ABCDEFGHIJKLMNOPQRSTUVWXYZ'))
```

  以下の値が返されます。

```
D8AC4B838921A83C4207B62B8B63628F8FBE836EB012167310331FFC070FC977D224F39148
  8806CB1FE2AA9C8C739E5104CAD1C4C6E97967DA6223D657CD9295
```

HASH_VALUES

HASH_VALUES 関数は、値の集合のパーティション番号を戻します。

483 ページの『HASHED_VALUE』 も参照してください。パーティション番号の詳細については、「Db2 マルチシステム」トピック集を参照してください。

expression
日付、時刻、タイム・スタンプ、浮動小数点数、XML、またはデータ・リンクの値を除く任意の組み込みデータ・タイプ値を戻す式。

この関数の結果は、0 から 1023 の値の長整数になります。

引数のうちどれかが NULL の場合、その結果はゼロになります。結果が NULL になることはありません。

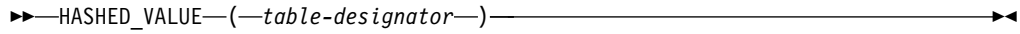
例

- パーティション・キーが EMPNO と LASTNAME から構成されている場合に、HASH_VALUES 関数を使用して、パーティションが何であるかを判別します。この照会は、EMPLOYEE の行すべてについてのパーティション番号を戻します。

```
SELECT HASH_VALUES(EMPNO, LASTNAME)
  FROM EMPLOYEE
```

HASHED_VALUE

HASHED_VALUE 関数は、行のパーティション・キー値にハッシュ関数を適用して取得した、行のパーティション・マップ索引番号を戻します。



482 ページの『HASH_VALUES』 関数も参照してください。引数が非分散表を示している場合は、値 0 が戻されます。パーティション・マップおよびパーティション・キーについての詳細は、「Db2 マルチシステム」トピック集を参照してください。

table-designator
副選択の表指定子。表指定子の詳細については、 166 ページの『表指定子』を参照してください。

SQL 命名規則では、表名は修飾できます。システム命名規則では、表名は修飾できません。

table-designator は、 *collection-derived-table*、VALUES 節、*table-function*、または *data-change-table-reference* を指定するものであってはなりません。

引数がビュー、共通表式、またはネストされた表式を示している場合、この関数はその基本表のパーティション・マップ索引番号を返します。引数が、複数の基本表から派生したビュー、共通表式、またはネストされた表式を示している場合、この関数は、そのビュー、共通表式、またはネストされた表式の外側の副選択内にある最初の表のパーティション・マップ索引番号を返します。

引数は、外側の全選択に、集約関数、GROUP BY 節、HAVING 節、UNION、INTERSECT、または EXCEPT 節、 DISTINCT 節、VALUES 節、または *table-function* が含まれている、ビュー、共通表式、またはネストされた表式を示していてはなりません。全選択が集約関数、GROUP BY 文節、または HAVING 文節を含む場合、SELECT 文節に HASHED_VALUE 関数を指定することはできません。

結果のデータ・タイプは、0 から 1023 の値を持つ長整数になります。結果は NULL になる場合があります。

注記

代替構文**:** PARTITION は HASHED_VALUE の同義語です。

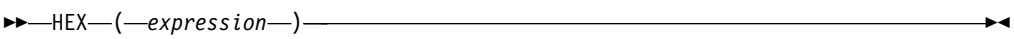
例

- パーティション・マップ索引番号が 100 である行すべてについて、表 EMPLOYEE から、従業員番号 (EMPNO) を選択します。

```
SELECT EMPNO
  FROM EMPLOYEE
  WHERE HASHED_VALUE(EMPLOYEE) = 100
```

HEX

HEX 関数は、値の 16 進数表現を戻します。



expression
長さ属性が 16,336 以下の文字ストリングまたはバイナリー・ストリング、または長さ属性が 8,168 以下のグラフィック・ストリングの任意の組み込みデータ・タイプ値を戻す式。

この関数の結果は、文字ストリングになります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

結果は 16 進数字のストリングです。最初の 2 桁が引数の 1 バイト目を表し、次の 2 桁が引数の 2 バイト目を表すというように、2 桁一組で引数の各バイトを順に表します。引数が日付時刻の値である場合、結果は引数の内部形式の 16 進数表現になります。[59]

引数がグラフィック・ストリングでない場合、結果の実際の長さは引数の長さの 2 倍になります。引数がグラフィック・ストリングの場合、結果の実際の長さは引数の長さの 4 倍になります。結果のデータ・タイプが可変長の場合、長さはそのデータ・タイプの最大長に制限されます。引数の長さは、引数が LENGTH スカラー関数へ渡された場合に戻される値です。詳しくは、 522 ページの『LENGTH』を参照してください。

結果のデータ・タイプおよび長さ属性は、引数の属性によって次のように異なります。

- 引数がストリングでない場合、結果は、長さ属性が引数の長さの 2 倍の CHAR です。
- 引数が、長さ属性が CHAR の長さ属性の最大長の半分より短い固定長文字ストリングの場合、結果は、長さ属性が引数の長さ属性の 2 倍である CHAR になります。引数が、長さ属性が CHAR の長さ属性の最大長の 4 分の 1 より短い固定長グラフィック・ストリングの場合、結果は、長さ属性が引数の長さ属性の 4 倍である CHAR になります。製品固有の最大長に関して詳しくは、 1849 ページの表 121を参照してください。
- その他の場合、結果は、長さ属性が以下によって異なる VARCHAR になります。
  - 引数が文字またはバイナリー・ストリングの場合は、結果の長さ属性は、この引数の長さ属性の 2 倍と、このデータ・タイプの最大長のいずれか小さい方です。
  - 引数がグラフィック・ストリングの場合は、結果の長さ属性は、この引数の長さ属性の 4 倍と、このデータ・タイプの最大長のいずれか小さい方です。

59. DATE、TIMESTAMP、および NUMERIC のデータ・タイプの内部形式は、他のデータベース・プロダクトの場合とは異なるため、これらのデータ・タイプの 16 進数表現も他のデータベース・プロダクトの場合とは異なります。

結果の長さ属性は、CHAR または VARCHAR の製品固有の長さ属性を超えることはできません。詳しくは、 1849 ページの表 121を参照してください。

ストリングの CCSID は、現行サーバーにおけるデフォルト SBCS CCSID です。

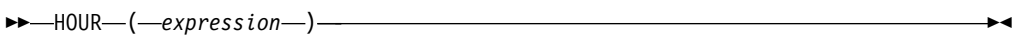
例

- HEX 関数を使用して、各従業員の教育レベルを 16 進数表現で戻します。

```
SELECT FIRSTNME, MIDINIT, LASTNAME, HEX(EDLEVEL)
  FROM EMPLOYEE
```

HOUR

HOUR 関数は、指定した値の時の部分を戻します。



expression
日付、時刻、タイム・スタンプ、文字ストリング、グラフィック・ストリング、または数値のいずれかの組み込みデータ・タイプの値を戻す式。

- *expression* が文字ストリングまたはグラフィック・ストリングである場合、その値は、日時値の有効なストリング表現でなければなりません。式が日付の有効なストリング表現の場合、その形式は IBM SQL 標準形式のいずれかでなければなりません。日時値のストリング表現の有効な形式については、95 ページの『日付/時刻の値のストリング表記』を参照してください。
- 引数が DATE である場合、時刻が午前 0 時ちょうど (00.00.00) であると想定して、最初に TIMESTAMP(0) 値に変換されます。
- 式 が数値である場合は、その数値は時刻期間またはタイム・スタンプ期間でなければなりません。日時期間の有効な形式については、205 ページの『日付/時刻のオペランドと期間』を参照してください。

この関数の結果は長精度整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

その他の規則は、引数のデータ・タイプに応じて以下のように異なります。

- 引数が日付、時刻、またはタイム・スタンプであるか、または、日付、時刻、またはタイム・スタンプの有効な文字ストリング表現である場合:

  結果は、指定した値の時の部分 (0 から 24 までの整数) になります。
- 引数が時刻期間またはタイム・スタンプ期間の場合 :

  結果は、指定した値の時の部分 (-99 から 99 までの整数) になります。ゼロ以外の結果の符号は、引数と同じになります。

例

- サンプル表 CL_SCHED を使用して、午後に始まるクラスをすべて選択します。

```
SELECT *
  FROM CL_SCHED
  WHERE HOUR(STARTING) BETWEEN 12 AND 17
```

IDENTITY_VAL_LOCAL

IDENTITY_VAL_LOCAL は、識別列に割り当てられた最も新しい値を戻す非決定性関数です。

この関数には入力パラメーターはありません。結果値に対する識別列の実際のデータ・タイプに関係なく、結果は DECIMAL(31,0) です。

戻される値は、最も新しい挿入操作 (INSERT ステートメントまたは MERGE ステートメントのいずれかで指定されたもの) で指定された表の識別列に割り当てられた値です。挿入操作は同じレベルで実行しなければなりません。つまり、値は、次の割り当て値によって置き換えられるまで、その値が割り当てられたレベル内でローカルに使用可能でなければなりません。新しいレベルが開始されるのは、トリガー、関数、またはストアード・プロシージャーが呼び出されたときです。トリガー状態は、それに関連してトリガーされるアクションと同じレベルにあります。

割り当てられる値には、ユーザーが指定する値 (識別列が GENERATED BY DEFAULT と定義されている場合) と、データベース・マネージャーが生成する識別値があります。

結果が、NULL になることもあります。結果がヌルになるのは、現行の処理レベルにある識別列を含まない表に対して、挿入操作を発行した場合です。これには、前挿入トリガーまたは後挿入トリガーの中でこの関数を呼び出した場合も含まれます。

以下のステートメントは、IDENTITY_VAL_LOCAL 関数の結果に影響を与えません。

- 識別列を含まない表に対する挿入操作
- UPDATE ステートメント
- COMMIT ステートメント
- ROLLBACK ステートメント

注

以下の注意事項では、幾つかの異なる状況下でこの関数を呼び出した場合の、この関数の動作を説明します。

挿入操作の **VALUES** 文節の中でこの関数を呼び出した場合
: 挿入操作のターゲット列に値が割り当てられる前に、挿入操作の中の式が評価されます。したがって、挿入操作の中で IDENTITY_VAL_LOCAL を呼び出した場合は、使用される値は、前回の挿入操作以降に識別列に割り当てられた最も新しい値です。以前に、この IDENTITY_VAL_LOCAL 関数の呼び出しと同じレベルで挿入操作が実行されていない場合は、この関数は NULL 値を戻します。

挿入操作が失敗した後でこの関数を呼び出した場合
: 識別列を含む表に対する挿入操作の実行が失敗した後でこの関数を呼び出した場合は、どのような結果が戻されるかは予測できません。戻される値は、

失敗した挿入操作の前にこの関数が呼び出されていたとすれば、そのときに戻されているものと予想される値になることもあり、また、挿入操作が成功していたとすれば戻されていたであろうと予想される値になることもあります。実際に戻される値は障害の発生時点によって決まるため、予測することはできません。

カーソルの **SELECT** ステートメントの中でこの関数を呼び出した場合
IDENTITY_VAL_LOCAL 関数の結果は非決定性のものなので、カーソルの SELECT ステートメントの中で IDENTITY_VAL_LOCAL 関数を呼び出した場合の結果は、各 FETCH ステートメントごとに異なる場合があります。

プロシージャーまたは関数のデフォルト式の中でこの関数を呼び出した場合
プロシージャーまたは関数のデフォルト式の中から IDENTITY_VAL_LOCAL 関数を呼び出した結果は定義されていません。この関数をデフォルト式の中で使用するべきではありません。

挿入トリガーのトリガー条件の中でこの関数を呼び出した場合
挿入トリガーの条件の中で IDENTITY_VAL_LOCAL 関数を呼び出した場合の結果は、NULL 値になります。

挿入トリガーによりトリガーされたアクションの中でこの関数を呼び出した場合
1 つの表について、複数の前挿入トリガーおよび後挿入トリガーが存在することができます。その場合は、各トリガーはそれぞれ個別に処理され、トリガーされたアクションの中で発行された SQL ステートメントが生成する識別値は、IDENTITY_VAL_LOCAL 関数を使用する他のトリガーされたアクションでは使用できません。これは、トリガーされる複数のアクションが概念的に同じレベルで定義されている場合も同じです。

前挿入トリガーのトリガーされたアクションの中では、IDENTITY_VAL_LOCAL 関数を使用しないでください。前挿入トリガーのトリガーされたアクションの中で IDENTITY_VAL_LOCAL 関数を呼び出した場合の結果は、NULL 値になります。トリガーが定義されている表の識別列の値を、前挿入トリガーのトリガーされたアクションの中で IDENTITY_VAL_LOCAL を呼び出すことによって取得することはできません。ただし、トリガーされたアクションで識別列に対するトリガー遷移変数を参照することにより、この識別列の値を取得することができます。

後挿入トリガーのトリガーされたアクションの中で IDENTITY_VAL_LOCAL 関数を呼び出した場合の結果は、識別列を含む表に対する同じトリガーされたアクションの中で呼び出された最新の挿入操作で指定されている表の識別列に割り当てられている値になります。IDENTITY_VAL_LOCAL 関数を呼び出す前に、同じトリガーされたアクションの中で、識別列を含む表に対する挿入操作が実行されていない場合は、この関数は NULL 値を戻します。

トリガーされたアクションを伴う挿入操作の後でこの関数を呼び出した場合
トリガーを活動化する挿入操作の後で関数を呼び出した場合の結果は、実際に識別列に割り当てられている値 (つまり、以後の SELECT ステートメントで戻されることになる値) になります。この値は、必ずしも、挿入操作で提供される値、またはデータベース・マネージャーが生成する値とは限りません。割り当てられる値は、識別列に関連したトリガー遷移変数に対する前

挿入トリガーのトリガーされたアクションの中で、SET 遷移変数ステートメントに指定されている値の場合もあります。

IDENTITY_VAL_LOCAL の有効範囲

IDENTITY_VAL_LOCAL 値は、現行セッションにおいて、定義されている ID 列が含まれる表に対して次の挿入操作が行われるか、アプリケーション・セッションが終了するまで存続します。この値は COMMIT または ROLLBACK ステートメントの影響を受けません。直接 IDENTITY_VAL_LOCAL 値を設定することはできず、それは行を表に挿入した結果として得られます。

特にパフォーマンスの目的で広く使用されている技法として、アプリケーションまたは製品に接続のセットを管理させ、トランザクションを任意の接続へ経路指定する、というものがあります。こうした状況では、IDENTITY_VAL_LOCAL 値はそのトランザクションが終了するまでの間のみ有効です。

IDENTITY_VAL_LOCAL の代替方法**:**

ID 列に割り当てられた値を取得するには、SELECT FROM INSERT を使用することをお勧めします。詳しくは、 794 ページの『table-reference』を参照してください。

例

- 変数 IVAR を、EMPLOYEE 表の識別列に割り当てられている値にセットします。 VALUES INTO ステートメントの中でこの関数から戻される値は 1 です。

```
CREATE TABLE EMPLOYEE
  (EMPNO INTEGER GENERATED ALWAYS AS IDENTITY,
   NAME CHAR(30),
   SALARY DECIMAL(5,2),
   DEPTNO SMALLINT)

INSERT INTO EMPLOYEE
  (NAME, SALARY, DEPTNO)
  VALUES('Rupert', 989.99, 50)

VALUES IDENTITY_VAL_LOCAL() INTO  :IVAR
```

- T1 および T2 という 2 つの表に、C1 という名前の識別列があるとします。データベース・マネージャーは、表 T1 の C1 列については 値 1、2、3 ... を生成し、表 T2 の C1 列については 値 10、11、12 ... を生成します。

```
CREATE TABLE T1
  (C1 SMALLINT GENERATED ALWAYS AS IDENTITY,
   C2 SMALLINT)

CREATE TABLE T2
  (C1 DECIMAL(15,0) GENERATED BY DEFAULT AS IDENTITY ( START WITH 10 ) ,
   C2 SMALLINT)

INSERT INTO T1 ( C2 ) VALUES(5)

INSERT INTO T1 ( C2 ) VALUES(5)

SELECT  * FROM T1
```

C1	C2
1	5

C1	C2
2	5

```
VALUES IDENTITY_VAL_LOCAL() INTO  :IVAR
```

この時点で、IDENTITY_VAL_LOCAL 関数は IVAR に値 2 を戻します。以下の INSERT ステートメントは、T2 に 1 つの行を挿入し、その行の列 C2 には、IDENTITY_VAL_LOCAL 関数が戻す 2 の値が入ります。

```
INSERT INTO T2 ( C2 ) VALUES( IDENTITY_VAL_LOCAL() )

SELECT  * FROM T2
  WHERE C1 = DECIMAL( IDENTITY_VAL_LOCAL(), 15, 0)
```

C1	C2
10	2

この INSERT の後で IDENTITY_VAL_LOCAL 関数を呼び出すと、値 10 が戻されます。これは、データベース・マネージャーが T2 の列 C1 用として生成した値です。ここで、T2 にもう 1 つ行を挿入するものとします。以下の INSERT ステートメントでは、データベース・マネージャーは、列 C1 を識別するために値 13 を割り当て、 C2 には IDENTITY_VAL_LOCAL から戻された 値 10 を割り当てます。したがって、C2 には、T2 に挿入された最後の識別値が与えられます。

```
INSERT INTO T2 ( C2, C1 ) VALUES( IDENTITY_VAL_LOCAL(), 13 )

SELECT  * FROM T2
  WHERE C1 = DECIMAL( IDENTITY_VAL_LOCAL(), 15, 0)
```

C1	C2
13	10

- IDENTITY_VAL_LOCAL 関数を呼び出すと同時に、識別列に新しい値を割り当てる INSERT ステートメントの中で、IDENTITY_VAL_LOCAL 関数を呼び出すこともできます。この場合、次に戻される値は、INSERT ステートメントの完了後に IDENTITY_VAL_LOCAL 関数が呼び出された時点で決定されます。例えば、以下の表定義について考えてみてください。

  ```
  CREATE TABLE T3
    (C1 SMALLINT GENERATED BY DEFAULT AS IDENTITY,
     C2 SMALLINT)
  ```

  以下の INSERT ステートメントでは、C2 列用の値として 25 を指定しており、データベース・マネージャーは、C1 (識別列) 用の値として 1 を生成します。その結果、次回の IDENTITY_VAL_LOCAL 関数の呼び出しで戻される値として、1 が設定されます。

  ```
  INSERT INTO T3 ( C2 ) VALUES( 25 )
  ```

  以下の INSERT ステートメントでは、IDENTITY_VAL_LOCAL 関数が呼び出されて、C2 列に入れる値を戻します。値 1 (最初の行の C1 列に割り当てられている識別値) が C2 列に割り当てられ、データベース・マネージャーは C1

(識別列) の値として 2 を生成します。その結果、次回の IDENTITY_VAL_LOCAL 関数の呼び出しで戻される値として、2 が設定されます。

```
INSERT INTO T3 ( C2 ) VALUES( IDENTITY_VAL_LOCAL() )
```

以下の INSERT ステートメントでは、再び IDENTITY_VAL_LOCAL 関数が呼び出されて C2 列に入れる値を戻し、ユーザーが C1 (識別列) 用の値として 11 を指定します。値 2 (2 番目の行の C1 列に割り当てられている識別値) が、C2 列に割り当てられます。 C1 には 11 が割り当てられ、次回の IDENTITY_VAL_LOCAL 関数の呼び出しでは 11 の値が戻されます。

```
INSERT INTO T3 ( C2, C1 ) VALUES( IDENTITY_VAL_LOCAL(), 11 )
```

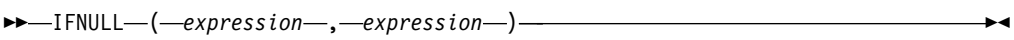
上記の 3 つの INSERT ステートメントの処理が終わると、表 T3 には以下の値が含まれています。

C1	C2
1	25
2	1
11	2

T3 の内容は、INSERT ステートメントの列の値が割り当てられる前に、VALUES 文節の中の式が評価されることを示しています。したがって、INSERT ステートメントの VALUES 文節から IDENTITY_VAL_LOCAL 関数を呼び出すと、前の INSERT ステートメントで識別列に割り当てられている最も新しい値が使用されます。

IFNULL

IFNULL 関数は、NULL でない最初の式の値を戻します。



IFNULL 関数は、2 つの引数を持つ COALESCE スカラー関数と同等です。詳しくは、 390 ページの『COALESCE』を参照してください。

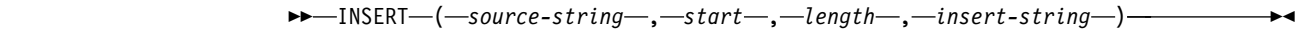
例

- 表 EMPLOYEE のすべての行から従業員番号 (EMPNO) および給与 (SALARY) を選択するとき、給与が欠落している (つまり、NULL である) と、値としてゼロを戻します。

```
SELECT EMPNO, IFNULL(SALARY,0)
  FROM EMPLOYEE
```

INSERT

ソース・ストリング の開始桁 から長さ 文字を削除し、ソース・ストリング の開始桁 の位置に挿入ストリング を挿入したストリングを戻します。



source-string
ソース・ストリングを指定する式。ソース・ストリング には、任意の組み込み数値またはストリング式を指定できます。これは、挿入ストリング と互換性のあるものでなければなりません。データ・タイプの互換性についての詳細は、113 ページの『割り当ておよび比較』を参照してください。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。ストリングの実際の長さはゼロより大きくなくてはなりません。

start
BIGINT、INTEGER、または SMALLINT の組み込みデータ・タイプを戻す式。整数は、文字の削除と別のストリングの挿入を開始するソース・ストリング内の開始文字を指定します。整数の値は、1 からソース・ストリング の長さに 1 を加えた数の範囲でなければなりません。

length
BIGINT、INTEGER、または SMALLINT の組み込みデータ・タイプを戻す式。この整数は、ソース・ストリング から削除される、開始桁 で示される文字位置から始まる文字数を指定します。整数の値は、0 からソース・ストリングの長さまでの範囲でなければなりません。

insert-string
ソース・ストリング に挿入する、開始桁 で示される位置から開始するストリングを指定する式。挿入ストリング には、任意の組み込み数値またはストリング式を指定できます。これは、ソース・ストリング と互換性のあるものでなければなりません。データ・タイプの互換性についての詳細は、 113 ページの『割り当ておよび比較』を参照してください。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。ストリングの実際の長さは 0 以上でなければなりません。

関数の結果のデータ・タイプは、1 番目と 4 番目の引数のデータ・タイプによって異なります。結果のデータ・タイプは、結果は常に可変長ストリングであることを除けば、 2 つの引数を連結した場合と同じです。詳しくは、 139 ページの『ストリングを結合する演算に適用される変換規則』を参照してください。

結果の長さ属性は、引数によって次のように異なります。

- 開始桁 と長さ が定数の場合、結果の長さ属性は次のとおりです。

  ```
  L1 - MIN((L1-V2 + 1), V3) + L4
  ```

  各値は、次のとおりです。

  ```
  L1 はソース・ストリングの長さ属性
  V2 は次のようにソース・ストリングのエンコード・スキーマに依存します。
  - ソース・ストリングが UTF-8 の場合は値 MIN(L1+1,start*3)
  ```

- ソース・ストリングが混合データの場合は値 MIN(L1+1,(start-1)*2.5+4)
上記以外の場合は値 start
V3 は長さの値
L4 は挿入ストリングの長さ属性
- それ以外の場合は、結果の長さ属性は、ソース・ストリング の長さ属性と挿入ストリング の長さ属性を足したものになります。

結果の長さ属性が結果のデータ・タイプの最大長を超える場合は、エラーが戻されます。

結果の実際の長さは、次のとおりです。

```
A1 - MIN((A1 -V2 + 1), V3) + A4
```

各値は、次のとおりです。

A1 はソース・ストリングの実際の長さ
V2 は開始桁の値
V3 は長さの値
A4 は挿入ストリングの実際の長さ

結果ストリングの実際の長さが結果のデータ・タイプの最大長を超える場合は、エラーが戻されます。

引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

結果の CCSID は、ソース・ストリング と挿入ストリング の CCSID によって決定されます。結果 CCSID は、2 つの引数を連結した場合と同じです。詳しくは、139 ページの『ストリングを結合する演算に適用される変換規則』を参照してください。

例

- 次の例は、ストリング「INSERTING」をどのように他のストリングに変更できるかを示しています。 CHAR 関数を使用すると、結果ストリングの長さが 10 文字に制限されます。

```
SELECT INSERT('INSERTING', 4, 2, 'IS'),
       INSERT('INSERTING', 4, 0, 'IS'),
       INSERT('INSERTING', 4, 2, '')
  FROM SYSIBM.SYSDUMMY1
```

この例は、「INSISTING 」、「INSISERTIN」、および「INSTING 」を戻します。.

- 前の例では、あるテキストの中にテキストを挿入する方法を示しました。この例は、開始点 (*start*) として 1 を使用して、テキストの前に他のテキストを挿入する方法を示しています。

```
SELECT INSERT('INSERTING', 1, 0, 'XX'),
       INSERT('INSERTING', 1, 1, 'XX'),
       INSERT('INSERTING', 1, 2, 'XX'),
       INSERT('INSERTING', 1, 3, 'XX')
  FROM SYSIBM.SYSDUMMY1
```

この例は、「XXINSERTIN」、「XXNSERTING」、「XXSERTING 」、および「XXERTING」を戻します。

- 次の例は、あるテキストの後ろにテキストを挿入する方法を示しています。ストリング「ABCABC」の末尾に「XX」を追加します。ソース・ストリングの長さが 6 文字なので、開始位置を 7 (ソース・ストリングに 1 を足した数) に設定します。

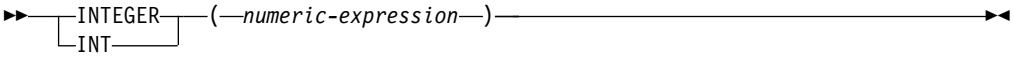
```
SELECT INSERT('ABCABC', 7, 0, 'XX')
  FROM SYSIBM.SYSDUMMY1
```

この例は、「ABCABCXX 」を戻します。

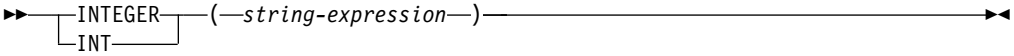
INTEGER または INT

INTEGER 関数は整数表現を返します。

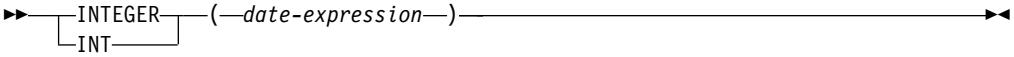
数値から整数



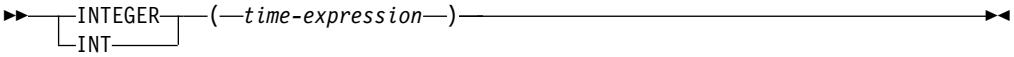
ストリングから整数



日付→整数



時刻→整数



INTEGER 関数は、次のものの整数表現を戻します。

- 数値
- 10 進数の文字ストリング表現またはグラフィック・ストリング表現
- 整数の文字ストリング表現またはグラフィック・ストリング表現
- 浮動小数点数の文字ストリング表現またはグラフィック・ストリング表現
- 10 進浮動小数点数の文字ストリング表現またはグラフィック・ストリング表現
- 日付
- 時刻

数値から整数へ

numeric-expression
任意の組み込み数値データ・タイプの数値を戻す式。

引数が数値式 の場合、結果は、引数を長精度整数 (large integer) の列または変数に割り当てた場合と同じ数値になります。 引数の整数部が、整数の範囲内にない場合は、エラーが戻されます。引数の小数部は切り捨てられます。

ストリングから整数へ

string-expression
数値の文字ストリング表現またはグラフィック・ストリング表現の値を戻す式。

引数がストリング式 の場合、結果は、 CAST(ストリング式 AS INTEGER) で得られる数値と同じです。先行ブランクと末尾ブランクは除去され、結果のストリングは、浮動小数点数、10 進浮動小数点数、整数、または 10 進数の定数を

形成する際の規則に合致している必要があります。 引数の整数部が、整数の範囲内にない場合は、エラーが戻されます。引数の小数部は切り捨てられます。

日付→整数

date-expression
DATE データ・タイプの値を戻す式。結果は、日付を *yyyymmdd* で表した INTEGER 値になります。

時刻→整数

time-expression
TIME データ・タイプの値を戻す式。結果は、時間を *hhmmss* で表した INTEGER 値になります。

この関数の結果は長精度整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

注記

代替構文**:** アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、 218 ページの『CAST の指定』を参照してください。

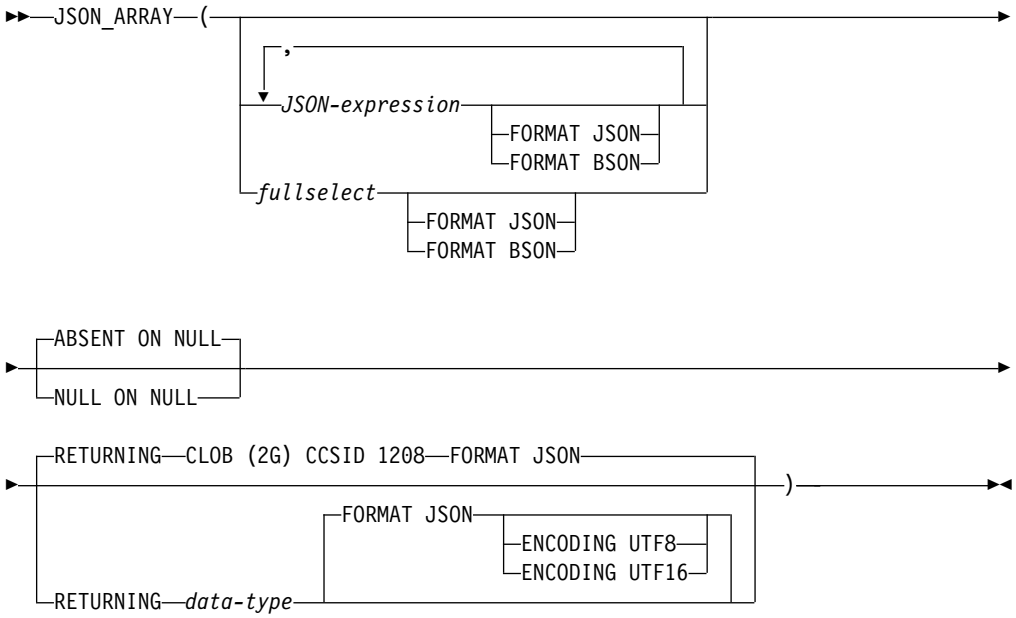
例

- 表 EMPLOYEE を使用して、給与 (SALARY) を教育レベル (EDLEVEL) で除算した値が入っているリストを選択します。計算で生じた小数部は、すべて切り捨てられます。このリストには、計算で使用した値と従業員番号 (EMPNO) も入れておきます。

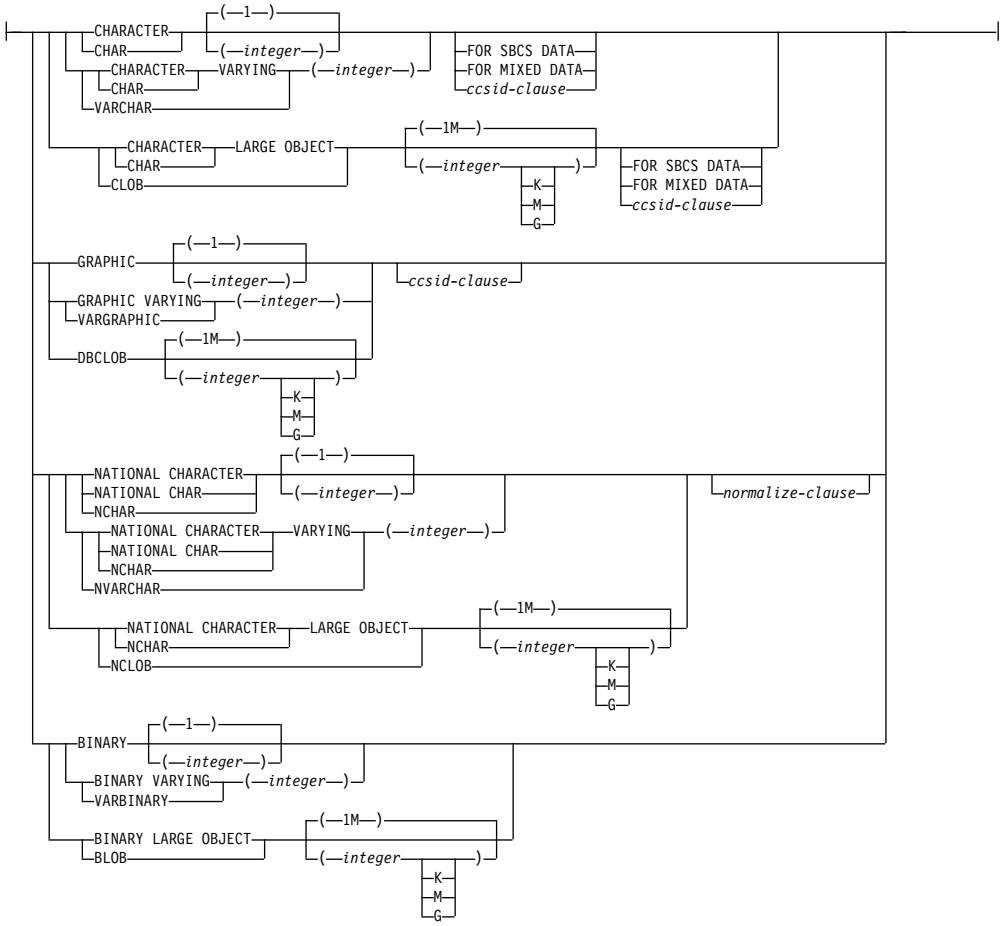
```
SELECT INTEGER(SALARY / EDLEVEL), SALARY, EDLEVEL, EMPNO
  FROM EMPLOYEE
```

JSON_ARRAY

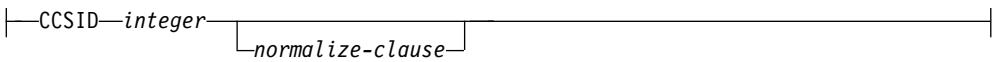
JSON_ARRAY 関数は、配列エレメントを明示的にリストするか、または照会を使用することで、JSON 配列を生成します。 *JSON-expression* が指定されない場合、*fullselect* が何も値を戻さない場合、またはすべての値が NULL で ABSENT ON NULL が指定された場合、空の配列が戻されます。



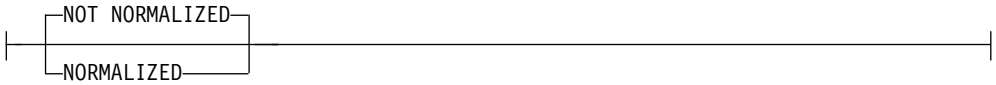
data-type:



ccsid-clause:



normalize-clause:



JSON-expression
JSON 配列の値を生成するために使用する式。この式の結果タイプには、XML、ROWID、および DATALINK を除く、任意の組み込みデータ・タイプが可能です。 CHAR または VARCHAR ビット・データであってはなりません。これらのいずれかのデータ・タイプをソースとするユーザー定義タイプにすることはできません。

fullselect
配列の値を生成するために使用する単一の列を戻す *fullselect* を指定します。各行の値から JSON 配列の値が 1 つ生成されます。この列の結果タイプには、XML、ROWID、DATALINK、およびこれらのいずれかのデータ・タイプをソ

ースとするユーザー定義タイプを除く、任意の組み込みデータ・タイプが可能です。 CHAR または VARCHAR ビット・データであってはなりません。

FORMAT JSON または **FORMAT BSON**
JSON-expression または *fullselect* が既に形式設定されたデータであるかどうかを指定します。

FORMAT JSON
JSON-expression または *fullselect* は JSON データとして形式設定されています。*JSON-expression* または *fullselect* が文字またはグラフィック・ストリング・データ・タイプであれば、JSON データとして扱われます。バイナリー・ストリング・データ・タイプの *JSON-expression* または *fullselect* は、UTF-8 または UTF-16 データとして解釈されます。

FORMAT BSON
JSON-expression または *fullselect* は JSON データの BSON 表現として形式設定されています。これは、バイナリー・ストリング・データ・タイプでなければなりません。

FORMAT JSON も FORMAT BSON も指定されない場合:

- *JSON-expression* が、組み込み関数 JSON_ARRAY、JSON_OBJECT、JSON_QUERY、 JSON_ARRAYAGG、または JSON_OBJECTAGG のいずれかである場合、関数の RETURNING 節の明示的または暗黙的な FORMAT 値が *JSON-expression* の形式を決定します。
- バイナリー・ストリング・タイプの *JSON-expression* は、FORMAT BSON として解釈されます。
- それ以外の場合、*JSON-expression* または *fullselect* は不定形式データと見なされます。生成値が数値以外の場合、結果ストリングは、引用符で囲んだストリングで構成され、特殊文字はエスケープされます。有効な JSON 数値ではない数値 (INFINITY や NAN など) は、エラーになります。

ABSENT ON NULL または **NULL ON NULL**
JSON-expression または *fullselect* によって生成された配列エレメントが NULL 値である場合に何を戻すかを指定します。

ABSENT ON NULL
NULL 配列エレメントは JSON 配列に含められません。これはデフォルトです。

NULL ON NULL
NULL 配列エレメントが JSON 配列に含められます。

RETURNING *data-type*
結果の形式を指定します。

data-type
結果のデータ・タイプです。CHAR 結果および VARCHAR 結果の場合、CCSID を 65535 にすることはできません。デフォルトは CLOB(2G) CCSID 1208 です。

CCSID が指定され、*data-type* が GRAPHIC、VARGRAPHIC、または DBCLOB の場合、CCSID は Unicode CCSID であることが必要です。

CCSID 属性が指定されないと、CCSID は 218 ページの『CAST の指定』に記されているように決定されます。

FORMAT JSON
JSON データは JSON ストリングとして戻されます。

ENCODING UTF8 または **ENCODING UTF16**
data-type がバイナリー・ストリング・タイプの場合に使用するエンコード方式。この節は、バイナリー・ストリング・タイプの場合にのみ使用できます。バイナリー・ストリングのデフォルトは UTF8 です。

例

- 値 Washington、Jefferson、および Hamilton を含む JSON 配列を生成します。

```
VALUES (JSON_ARRAY('Washington', 'Jefferson', 'Hamilton'));
```

結果は、以下の JSON 配列です。

```
["Washington","Jefferson","Hamilton"]
```

- すべての部門番号を含む JSON 配列を生成します。

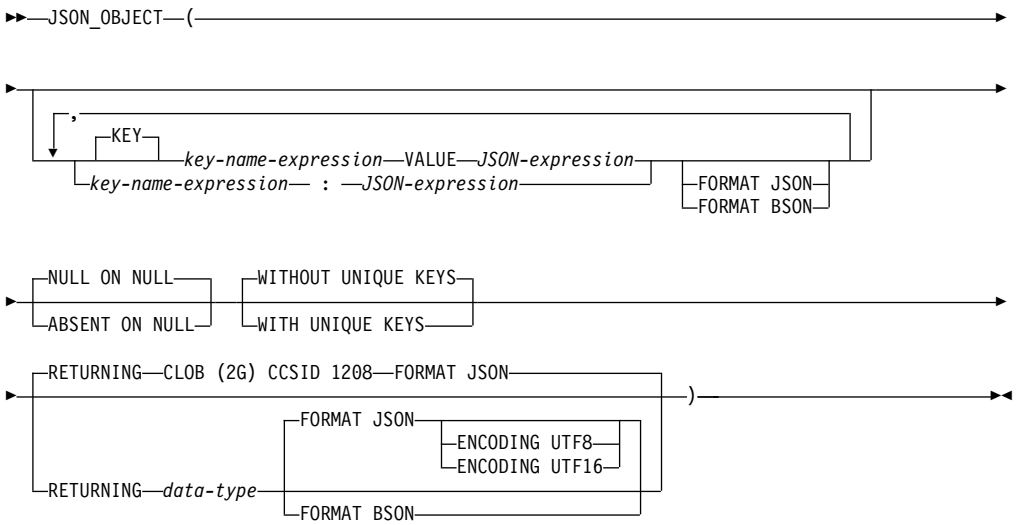
```
VALUES(JSON_ARRAY((SELECT DEPTNO FROM DEPT
                   WHERE DEPTNAME LIKE 'BRANCH OFFICE%' )));
```

結果は、以下の JSON 配列です。

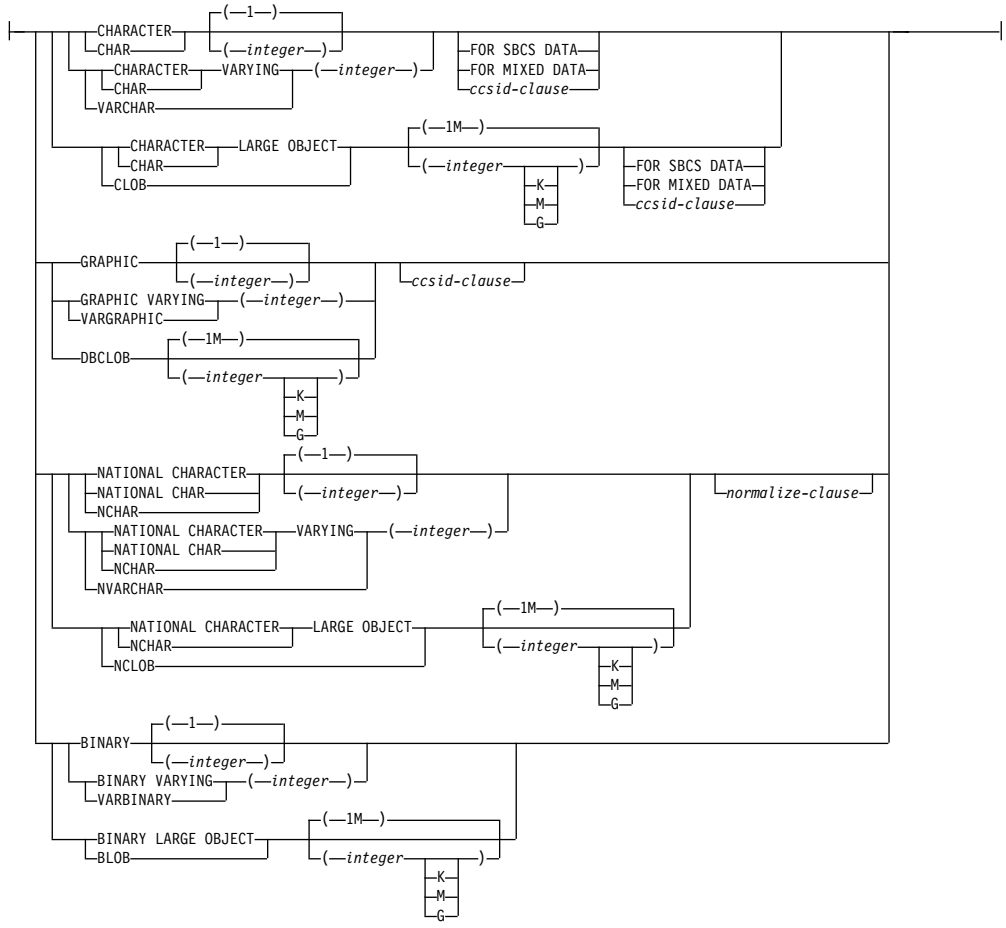
```
["F22","G22","H22","I22","J22"]
```

JSON_OBJECT

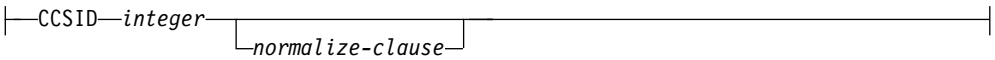
JSON_OBJECT 関数は、指定されたキー:値 のペアを使用して、JSON オブジェクトを生成します。キー:値 のペアを指定しないと、空のオブジェクトが戻されます。



data-type:



ccsid-clause:



normalize-clause:

key-name-expression
JSON キーの名前。この名前は、NULL であってはなりません。 *key:value* のペアの定義にコロン形式を使用する場合、*key-name-expression* は文字ストリング・リテラルでなければなりません。それ以外の場合、*key-name-expression* の結果は、組み込み文字またはグラフィック・ストリング・データ・タイプでなければなりません。 CHAR または VARCHAR ビット・データであってはなりません。

JSON-expression
key-name-expression に関連付けられた JSON 値を生成するために使用する式。この式の結果タイプには、XML、ROWID、および DATALINK を除く、任意の組み込みデータ・タイプが可能です。 CHAR または VARCHAR ビット・データであってはなりません。これらのいずれかのデータ・タイプをソースとするユーザー定義タイプにすることはできません。

FORMAT JSON または **FORMAT BSON**
JSON-expression が既に形式設定されたデータであるかどうかを指定します。

FORMAT JSON
JSON-expression は JSON データとして形式設定されています。 *JSON-expression* が文字またはグラフィック・ストリング・データ・タイプであれば、JSON データとして扱われます。バイナリー・ストリング・データ・タイプの *JSON-expression* は、UTF-8 または UTF-16 データとして解釈されます。

FORMAT BSON
JSON-expression は JSON データの BSON 表現として形式設定されています。これは、バイナリー・ストリング・データ・タイプでなければなりません。

FORMAT JSON も FORMAT BSON も指定されない場合:

- *JSON-expression* が、組み込み関数 JSON_ARRAY、JSON_OBJECT、JSON_QUERY、 JSON_ARRAYAGG、または JSON_OBJECTAGG のいずれかである場合、関数の RETURNING 節の明示的または暗黙的な FORMAT 値が *JSON-expression* の形式を決定します。
- バイナリー・ストリング・タイプの *JSON-expression* は、FORMAT BSON として解釈されます。

- それ以外の場合、*JSON-expression* は不定形式データと見なされます。生成値が数値以外の場合、結果ストリングは、引用符で囲んだストリングで構成され、特殊文字はエスケープされます。有効な JSON 数値ではない数値 (INFINITY や NAN など) は、エラーになります。

NULL ON NULL または **ABSENT ON NULL**
JSON-expression が NULL 値である場合に何を戻すかを指定します。

NULL ON NULL
NULL 値が戻されます。これはデフォルトです。

ABSENT ON NULL
キー:値 のペアを JSON オブジェクトから除外します。

WITHOUT UNIQUE KEYS または **WITH UNIQUE KEYS**
結果の JSON オブジェクトのキー値をユニークにする必要があるかどうかを指定します。

WITHOUT UNIQUE KEYS
結果の JSON オブジェクトに重複キーがあるかどうかは検査されません。これはデフォルトです。

WITH UNIQUE KEYS
結果の JSON オブジェクトは、ユニーク・キー値を持つ必要があります。重複キーが生成されると、エラーが発行されます。

ユニーク・キーを持つ JSON オブジェクトを生成することが、ベスト・プラクティスと見なされています。*key-name-expression* がユニークなキー名を生成する場合は、WITH UNIQUE KEYS を省いてパフォーマンスを向上させることができます。

RETURNING *data-type*
結果の形式を指定します。

data-type
結果のデータ・タイプです。CHAR 結果および VARCHAR 結果の場合、CCSID を 65535 にすることはできません。デフォルトは CLOB(2G) CCSID 1208 です。

CCSID が指定され、*data-type* が GRAPHIC、VARGRAPHIC、または DBCLOB の場合、CCSID は Unicode CCSID であることが必要です。

CCSID 属性が指定されないと、CCSID は 218 ページの『CAST の指定』に記されているように決定されます。

FORMAT JSON
JSON データは JSON ストリングとして戻されます。

ENCODING UTF8 または **ENCODING UTF16**
data-type がバイナリー・ストリング・タイプの場合に使用するエンコード方式。この節は、バイナリー・ストリング・タイプの場合にのみ使用できます。バイナリー・ストリングのデフォルトは UTF8 です。

FORMAT BSON
JSON データは、BSON 形式で戻されます。FORMAT BSON を指定した場合、*data-type* は VARBINARY または BLOB ストリング・タイプでなければなりません。

例

- 名前の JSON オブジェクトを生成します。

```
VALUES (JSON_OBJECT(KEY 'first' VALUE 'John', KEY 'last' VALUE 'Doe'));

VALUES (JSON_OBJECT('first' : 'John', 'last' : 'Doe'));
```

これらのステートメントはどちらも、結果は、以下の JSON ストリングになります。

```
{"first":"John","last":"Doe"}
```

- 従業員番号が '000020' の従業員の姓、雇用日、給与を含む JSON オブジェクトを生成します。

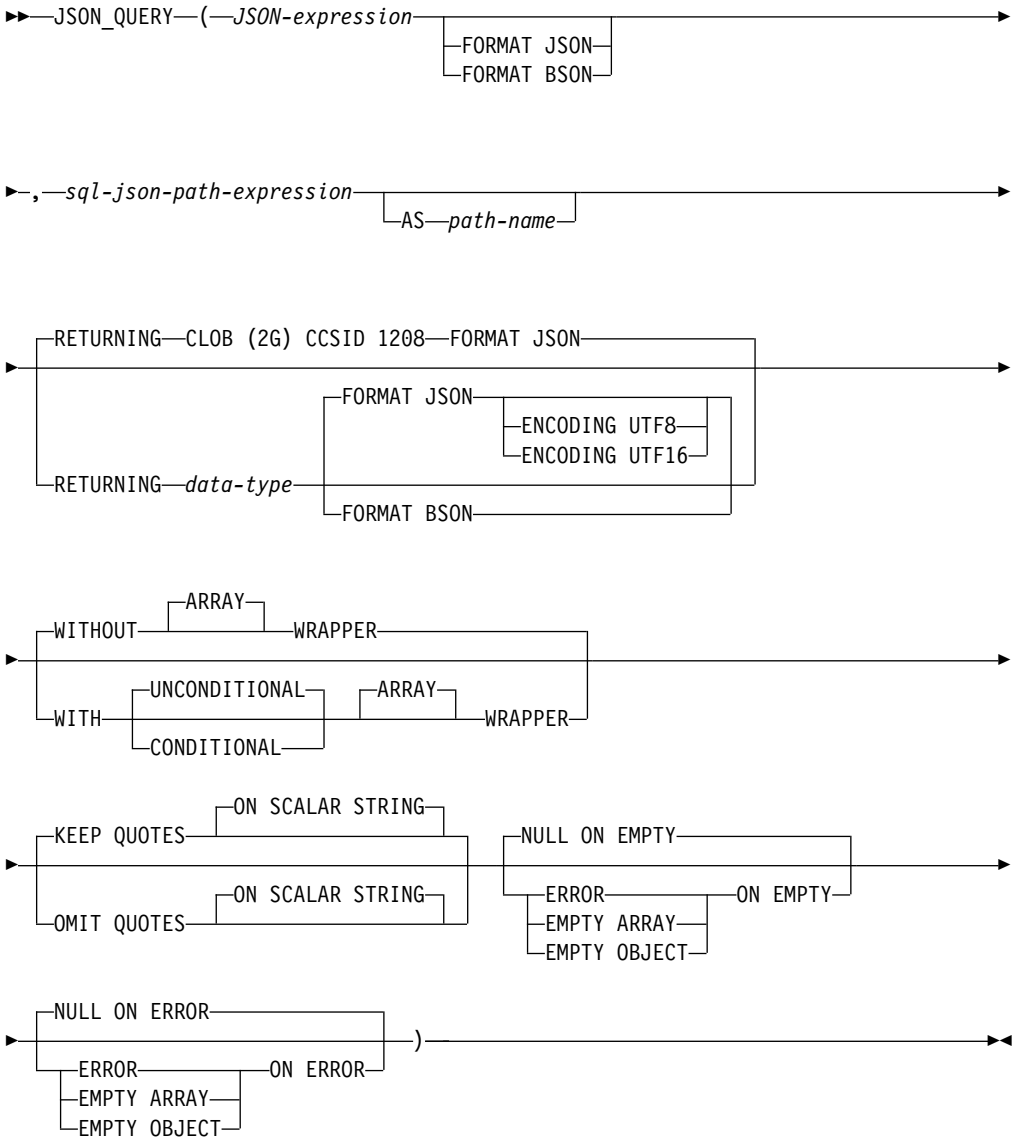
```
SELECT JSON_OBJECT(
               'Last name' : LASTNAME,
               'Hire date' : HIREDATE,
               'Salary'    : SALARY)
FROM EMPLOYEE
WHERE EMPNO  = '000020';
```

このステートメントの結果は、以下の JSON ストリングになります。

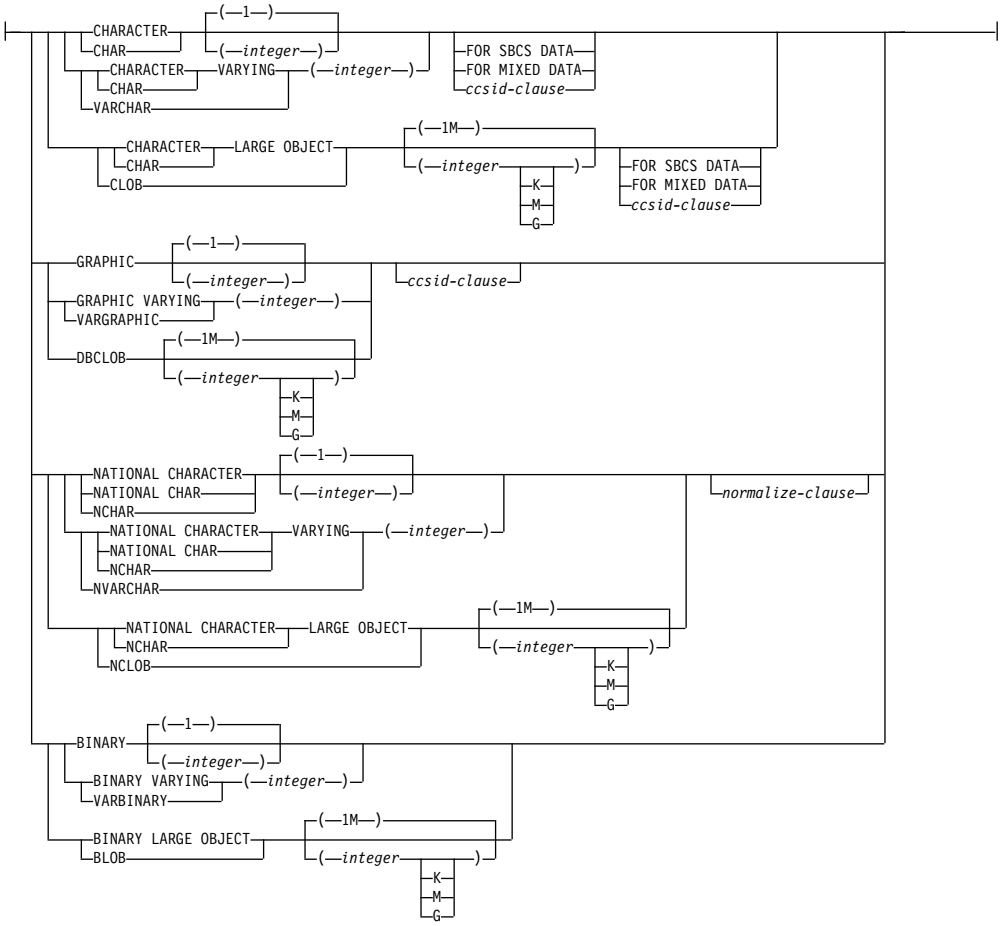
```
{"Last name":"THOMPSON","Hire date":"1973-10-10","Salary":41250.00}
```

JSON_QUERY

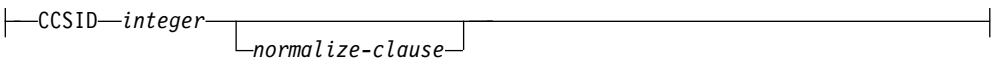
JSON_QUERY 関数は、SQL/JSON パス式を使用して、指定された JSON テキストの SQL/JSON 値を戻します。



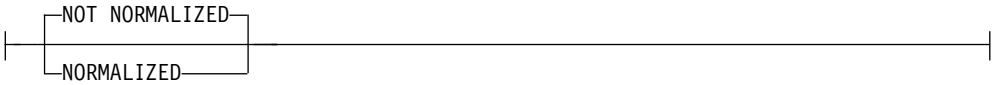
data-type:



ccsid-clause:



normalize-clause:



JSON-expression
組み込みストリング・データ・タイプの値を戻す式。文字またはグラフィックの値が戻される場合は、正しい形式の JSON データが含まれている必要があります。バイナリー・データ・タイプの場合は、明示的または暗黙的な FORMAT 節に従って解釈されます。

FORMAT JSON または **FORMAT BSON**
JSON-expression の解釈方法を指定します。

FORMAT JSON
JSON-expression には JSON データが含まれています。*JSON-expression* が

バイナリー・データの場合、データは UTF-8 または UTF-16 として解釈されます。EBCDIC CCSID を使用してバイナリー・データをエンコードすることはできません。

FORMAT BSON
JSON-expression には、JSON データの BSON 表現が含まれています。FORMAT BSON を指定した場合、*JSON-expression* はバイナリー・ストリング・データ・タイプでなければなりません。

FORMAT 節を指定しなかった場合、*JSON-expression* が文字ストリングまたはグラフィック・ストリングであれば、*JSON-expression* は JSON として扱われます。 *JSON-expression* がバイナリー・ストリングの場合、*JSON-expression* は BSON として扱われます。

sql-json-path-expression
組み込み文字またはグラフィック・ストリング・データ・タイプの値を戻す式。このストリングは SQL/JSON パス式として解釈され、*JSON-expression* で指定された JSON データ内で JSON 値を見つけるために使用されます。複数の値のキーが同じ場合、いずれかの JSON 値が選択されます。

SQL/JSON パス式の内容については、 260 ページの『sql-json-path-expression』を参照してください。

AS *path-name*
sql-json-path-expression の識別に使用する名前を指定します。

RETURNING *data-type*
結果の形式を指定します。

data-type
結果のデータ・タイプです。CHAR 結果および VARCHAR 結果の場合、CCSID を 65535 にすることはできません。デフォルトは CLOB(2G) CCSID 1208 です。

CCSID が指定され、*data-type* が GRAPHIC、VARGRAPHIC、または DBCLOB の場合、CCSID は Unicode CCSID であることが必要です。

CCSID 属性が指定されないと、CCSID は 218 ページの『CAST の指定』に記されているように決定されます。

FORMAT JSON
JSON データは JSON ストリングとして戻されます。

ENCODING UTF8 または **ENCODING UTF16**
data-type がバイナリー・ストリング・タイプの場合に使用するエンコード方式。この節は、バイナリー・ストリング・タイプの場合にのみ使用できます。バイナリー・ストリングのデフォルトは UTF8 です。

FORMAT BSON
JSON データは、BSON 形式で戻されます。FORMAT BSON を指定した場合、*data-type* は VARBINARY または BLOB ストリング・タイプでなければなりません。FORMAT BSON は、SQL/JSON オブジェクトが戻される場合にのみ使用できます。

WITHOUT ARRAY WRAPPER または **WITH ARRAY WRAPPER**
JSON 配列内で出力値をラップするかどうかを指定する必要があります。

WITHOUT ARRAY WRAPPER
結果をラップしません。これはデフォルトです。複数の SQL/JSON エレメントを含むシーケンスになる SQL/JSON パスを使用すると、エラーになります。

WITH UNCONDITIONAL ARRAY WRAPPER
結果を大括弧で囲んで、JSON 配列を作成します。

WITH CONDITIONAL ARRAY WRAPPER
複数の SQL/JSON エレメントが戻される場合、または JSON 配列でも JSON オブジェクトでもない単一の SQL/JSON エレメントが戻される場合、結果を大括弧で囲んで、JSON 配列を作成します。

以下の表は、JSON テキスト `{a:"10", b:[1, 2]}` にこれらの各オプションがどのように適用されるかを示しています。

表 *52.* 各 *WRAPPER* 節を使用した結果

WRAPPER 文節	**$.a** のパス値	**$.b** のパス値
WITHOUT ARRAY WRAPPER	"10"	[1,2]
WITH UNCONDITIONAL ARRAY WRAPPER	["10"]	[[1,2]]
WITH CONDITIONAL ARRAY WRAPPER	["10"]	[1,2]

KEEP QUOTES または **OMIT QUOTES**
スカラー・ストリングが戻された場合に、周囲の引用符を除去する必要があるかどうかを指定します。

KEEP QUOTES
引用符がスカラー・ストリングから除去されないことを示します。これはデフォルトです。

OMIT QUOTES
引用符がスカラー・ストリングから除去されることを示します。OMIT QUOTES を指定した場合、WITH ARRAY WRAPPER 節は指定できません。

ON EMPTY
sql-json-path-expression を使用して空シーケンスが戻された場合の動作を指定します。

NULL ON EMPTY
NULL 値が戻されます。これはデフォルトです。

ERROR ON EMPTY
エラーが戻されます。

EMPTY ARRAY ON EMPTY
空の配列が戻されます。

EMPTY OBJECT ON EMPTY
空のオブジェクトが戻されます。

ON ERROR
JSON_QUERY でエラーが発生した場合の動作を指定します。

NULL ON ERROR
NULL 値が戻されます。これはデフォルトです。

ERROR ON ERROR
エラーが戻されます。

EMPTY ARRAY ON ERROR
空の配列が戻されます。

EMPTY OBJECT ON ERROR
空のオブジェクトが戻されます。

結果が、NULL になることもあります。*JSON-expression* が NULL の場合、結果は NULL 値になります。

例

- JSON テキストから、name キーに関連付けられている JSON オブジェクトを返します。

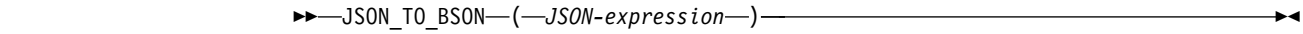
```
VALUES JSON_QUERY('{"id":"701", "name":{"first":"John", "last":"Doe"}}', '$.name');
```

この結果、JSON オブジェクトを表す以下のストリングが返されます。

```
{"first":"John", "last":"Doe"}
```

JSON_TO_BSON

JSON_TO_BSON 関数は、形式設定された JSON データを含むストリングを、BSON として形式設定されたデータを含むバイナリー・ストリングに変換します。



JSON-expression

文字またはグラフィック・ストリング値を戻す式を指定します。形式設定された JSON データを含まなければなりません。

JSON-expression 内の JSON オブジェクトに重複キーが含まれる場合、いずれか 1 つの *key:value* ペアだけが結果の BSON ストリングに組み込まれます。

結果のデータ・タイプは BLOB(2G) です。

引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

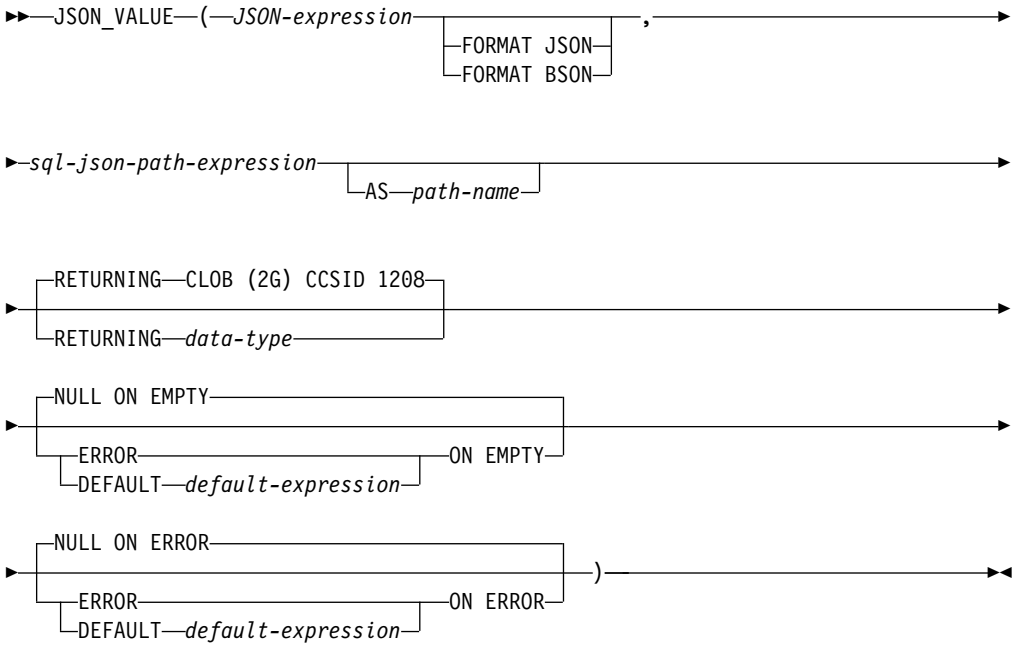
例

- 表から JSON 値を読み取り、BSON 形式に変換します。

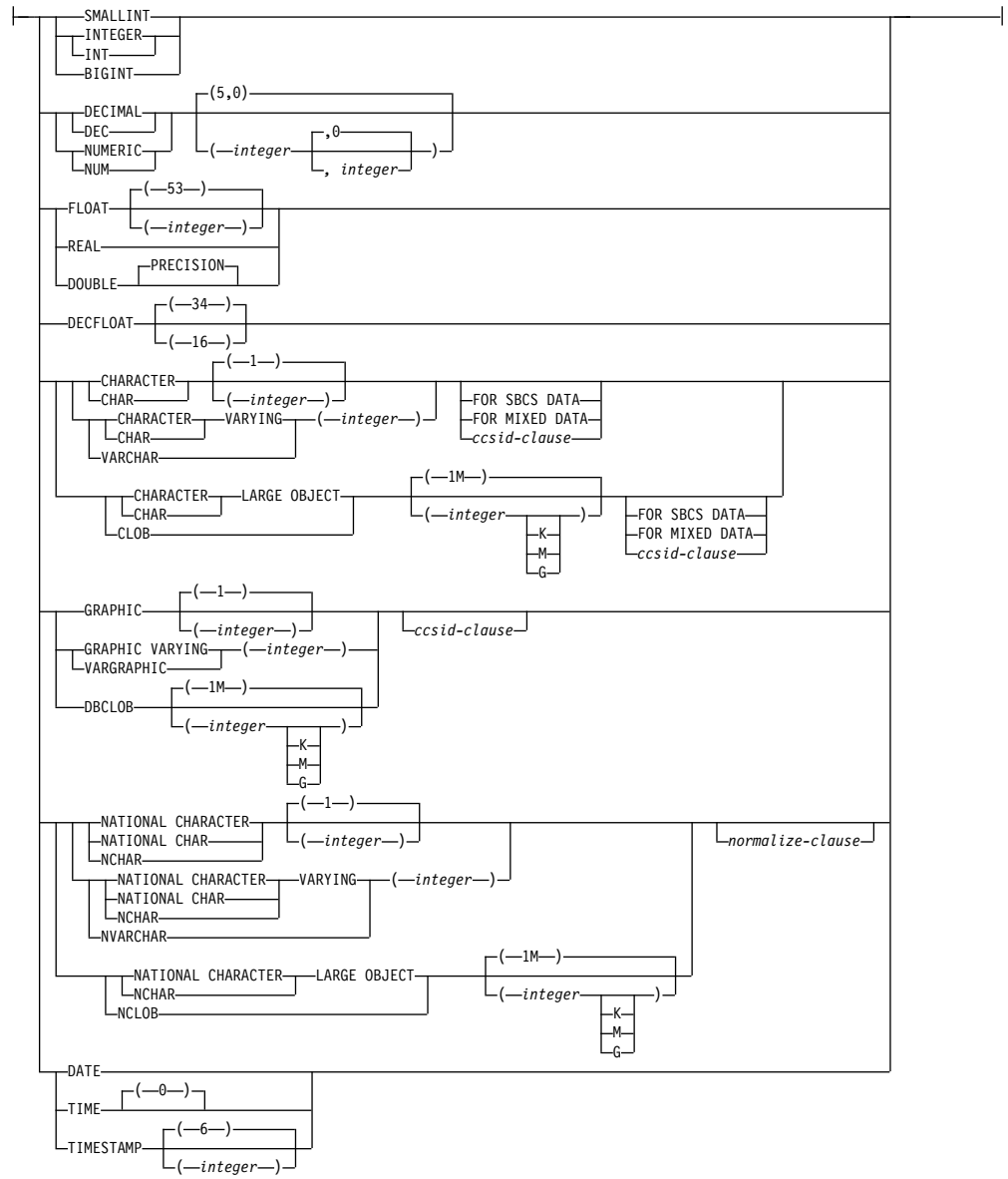
```
CREATE VARIABLE BSONVAR VARBINARY(2000);

SELECT JSON_TO_BSON(JSON_COL) INTO BSONVAR FROM TABLE2 WHERE KEY_COLUMN = 27;
```

JSON_VALUE

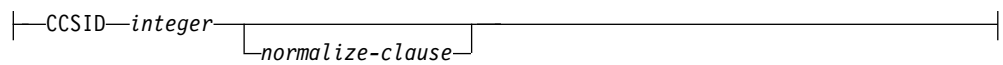
JSON_VALUE 関数は、SQL/JSON パス式を使用して、JSON テキストから SQL スカラー値を戻します。



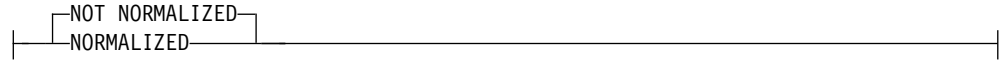
data-type:



ccsid-clause:



normalize-clause:



JSON-expression
組み込みストリング・データ・タイプの値を戻す式。文字またはグラフィックの値が戻される場合は、正しい形式の JSON データが含まれている必要があります。バイナリー・データ・タイプの場合は、明示的または暗黙的な FORMAT 節に従って解釈されます。

FORMAT JSON または **FORMAT BSON**
JSON-expression の解釈方法を指定します。

FORMAT JSON
JSON-expression には JSON データが含まれています。*JSON-expression* がバイナリー・データの場合、データは UTF-8 または UTF-16 として解釈されます。EBCDIC CCSID を使用してバイナリー・データをエンコードすることはできません。

FORMAT BSON
JSON-expression には、JSON データの BSON 表現が含まれています。FORMAT BSON を指定した場合、*JSON-expression* はバイナリー・ストリング・データ・タイプでなければなりません。

FORMAT 節を指定しなかった場合、*JSON-expression* が文字ストリングまたはグラフィック・ストリングであれば、*JSON-expression* は JSON として扱われます。 *JSON-expression* がバイナリー・ストリングの場合、*JSON-expression* は BSON として扱われます。

sql-json-path-expression
組み込み文字またはグラフィック・ストリング・データ・タイプの値を戻す式。このストリングは SQL/JSON パス式として解釈され、*JSON-expression* で指定された JSON データ内で JSON 値を見つけるために使用されます。複数の値のキーが同じ場合、いずれかの JSON 値が選択されます。

SQL/JSON パス式の内容については、 260 ページの『sql-json-path-expression』を参照してください。

AS *path-name*
sql-json-path-expression の識別に使用する名前を指定します。

RETURNING *data-type*
結果のデータ・タイプを指定します。CHAR 結果および VARCHAR 結果の場合、CCSID を 65535 にすることはできません。デフォルトは CLOB(2G) CCSID 1208 です。

ON EMPTY
sql-json-path-expression を使用して空シーケンスが検出された場合の動作を指定します。

NULL ON EMPTY
NULL 値が戻されます。これはデフォルトです。

ERROR ON EMPTY
エラーが戻されます。

DEFAULT *default-expression* **ON EMPTY**
default-expression で指定された値が戻されます。式は、結果のデータ・タイプに対して割り当ての互換性がなければなりません。

ON ERROR
JSON_VALUE でエラーが発生した場合の動作を指定します。

NULL ON ERROR
NULL 値が戻されます。これはデフォルトです。

ERROR ON ERROR
エラーが戻されます。

DEFAULT *default-expression* **ON ERROR**
default-expression で指定された値が戻されます。式は、結果のデータ・タイプに対して割り当ての互換性がなければなりません。

結果が、NULL になることもあります。*JSON-expression* が NULL の場合、結果は NULL 値になります。

例

- JSON テキストから値を整数として戻します。

```
VALUES (JSON_VALUE('{"id":"987"}', '$.id' RETURNING INTEGER));
```

結果は、987 になります。

- 配列値である JSON テキストから値を戻すことを試行します。デフォルト・ストリングを戻すことでエラー処理します。

```
VALUES (JSON_VALUE('{"friends":["John","Lisa"]}',
                   'strict $.friends' DEFAULT 'Not found' ON ERROR));
```

friends キーに対応する値が、スカラー値ではなく、配列であるため、結果は Not found になります。

JULIAN_DAY

JULIAN_DAY 関数は、紀元前 4713 年 1 月 1 日 (ユリウス暦の開始日) から引数で指定された日付までの日数を表す整数値を戻します。

expression
日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式。*expression* が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

この関数の結果は長精度整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

注記

ユリウス暦およびグレゴリオ暦**:** この関数では、1582 年 10 月 15 日のユリウス暦からグレゴリオ暦への移行は考慮されません。

例

- サンプル表 EMPLOYEE を使用して、整数ホスト変数 JDAY を、Christine Haas (EMPNO = '000010') が雇用された日付 (HIREDATE = '1965-01-01') のユリウス日にセットします。

```
SELECT JULIAN_DAY(HIREDATE)
  INTO :JDAY
  FROM EMPLOYEE
  WHERE EMPNO = '000010'
```

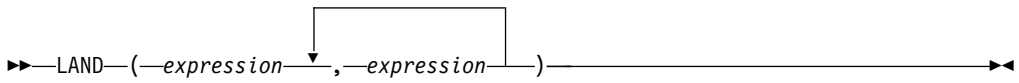
  この結果、JDAY は 2438762 にセットされます。
- 整数ホスト変数 JDAY を、1998 年 1 月 1 日のユリウス日にセットします。

```
SELECT JULIAN_DAY('1998-01-01')
  INTO :JDAY
  FROM SYSIBM.SYSDUMMY1
```

  この結果、JDAY は 2450815 にセットされます。

LAND

LAND 関数は、引数である 2 つのストリングの論理「AND」であるストリングを戻します。この関数は、最初の引数ストリングを取り、次のストリングとの AND 演算を行い、それ以後、次々に前の結果を使用して次の引数との AND 演算を繰り戻していきます。文字ストリング引数が前の結果より短い場合は、ブランクが埋め込まれます。 2 進ストリング引数が前の結果より短い場合は、16 進数のゼロが埋め込まれます。



各引数には、互換性がなければなりません。

expression
(LOB 以外の) 任意の組み込み数値データ・タイプ、またはストリング・データ・タイプの値を戻す式。引数は、混合データ文字ストリング、UTF-8 文字ストリング、またはグラフィック・ストリングであってはなりません。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。

必要ならば、引数は結果の属性に変換されます。結果の属性は、以下のように決められます。

- すべての引数が固定長ストリングである場合、結果は長さが *n* の固定長ストリングになります (ここで、*n* は最長の引数の長さ)。
- 引数の中に可変長ストリングがある場合、結果は長さ属性が *n* の可変長ストリングになります (ここで、*n* は最大の長さ属性を持つ引数の長さ属性)。結果の実際の長さは *m* です (ここで、*m* は、最長の引数の実際の長さ)。

引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

結果の CCSID は 65535 です。

例

- ホスト変数 L1 は値が X'A1B1' の CHARACTER(2) のホスト変数、ホスト変数 L2 は値が X'F0F040' の CHARACTER(3) のホスト変数、ホスト変数 L3 は値が X'A1B10040' の CHARACTER(4) のホスト変数であるとします。

```
SELECT LAND(:L1,:L2,:L3)
  FROM SYSIBM.SYSDUMMY1
```

値として X'A0B00040' が戻されます。

LAST_DAY

LAST_DAY 関数は、*expression* で指定される月の最後の日を表す日付またはタイム・スタンプを戻します。

expression
日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式。

expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、95 ページの『日付/時刻の値のストリング表記』を参照してください。

この関数の結果のデータ・タイプは *expression* と同じになります。ただし、*expression* がストリングの場合は結果は DATE になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

expression に含まれる情報は、時間、分、秒または 1 秒未満の値にいたるまで、関数によって変更されることはありません。

例

- ホスト変数 END_OF_MONTH を今月の最後の日に設定します。

```
SET :END_OF_MONTH = LAST_DAY(CURRENT_DATE)
```

ホスト変数 END_OF_MONTH は、今月の終わりを表す値に設定されます。現在の日付が 2000-02-10 である場合、END_OF_MONTH は 2000-02-29 に設定されます。

- ホスト変数 END_OF_MONTH を、EUR 形式で所定の日付けに対する月の最後の日に設定します。

```
SET :END_OF_MONTH = CHAR(LAST_DAY('1965-07-07'), EUR)
```

ホスト変数 END_OF_MONTH は値 '31.07.1965' に設定されます。

- デフォルトの日付形式が ISO であると想定します。

```
SELECT LAST_DAY('2000-04-24')
FROM SYSIBM.SYSDUMMY1
```

2000 年 4 月の最後の日である「2000–04–30」が戻されます。

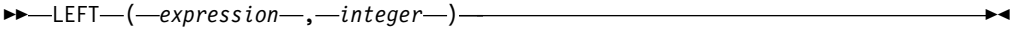
LCASE

LCASE 関数は、すべての文字を引数の CCSID に基づいて小文字に変換したストリングを戻します。

LCASE 関数は、LOWER 関数と同等です。 詳しくは、 533 ページの『LOWER』を参照してください。

LEFT

LEFT 関数は、式 の左端から整数 個の文字を戻します。



expression が文字ストリングの場合は、結果は文字ストリングになります。 *expression* がグラフィック・ストリングの場合は、結果はグラフィック・ストリングになります。 *expression* が 2 進ストリングの場合は、結果は 2 進ストリングになります。

expression
結果が導き出される元になるストリングを指定する式。引数は、任意の組み込み数値、文字ストリング、グラフィック・ストリング、または 2 進ストリング・データ・タイプの値を戻す式でなければなりません。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。

式 のサブストリングは、式 のゼロ個以上の連続したバイトです。 *expression* が文字ストリングまたはグラフィック・ストリングの場合、1 文字は SBCS、DBCS、またはマルチバイト文字です。*expression* が 2 進ストリングの場合は、結果は、引数のバイト数です。

integer
組み込み整数データ・タイプを戻す式。整数は結果の長さを指定します。整数の値は、0 以上で *n* 以下でなければなりません。 *n* は式 の長さ属性です。

この関数の結果は、式 と同じ長さ属性を持つ可変長ストリングで、データ・タイプは式 のデータ・タイプに応じて以下のようになります。

式 のデータ・タイプ	結果のデータ・タイプ
CHAR または VARCHAR	VARCHAR
CLOB	CLOB
GRAPHIC または VARGRAPHIC	VARGRAPHIC
DBCLOB	DBCLOB
BINARY または VARBINARY	VARBINARY
BLOB	BLOB

結果の実際の長さは整数 です。

引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

結果の CCSID は式 の CCSID と同じです。

例

- ホスト変数 NAME (VARCHAR(50)) は、'KATIE AUSTIN' という値を持ち、ホスト変数 FIRSTNAME_LEN (int) は、5 という値を持つと想定します。

```
SELECT LEFT(:NAME, :FIRSTNAME_LEN)
  FROM SYSIBM.SYSDUMMY1
```

値 'KATIE' が戻されます。

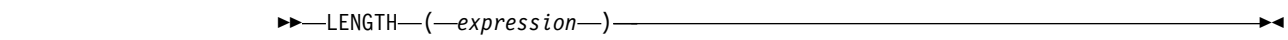
- NAME は VARCHAR(128) の列で Unicode UTF-8 でエンコードされ、値 'Jürgen' を含むものと仮定します。

```
SELECT LEFT(NAME, 2), SUBSTR(NAME, 1, 2)
  FROM T1
  WHERE NAME = 'Jürgen'
```

LEFT について値 'Jü' を、SUBSTR(NAME, 1, 2) について値 'JÊ' が戻されます。

LENGTH

LENGTH 関数は、値の長さを戻します。



類似の関数については、 382 ページの『CHARACTER_LENGTH』、 570 ページの『OCTET_LENGTH』、および 368 ページの『BIT_LENGTH』を参照してください。

expression
任意の組み込みデータ・タイプの値を戻す式。

この関数の結果は長精度整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

結果は、引数の長さです。ストリングの長さには、ブランクも含まれます。可変長ストリングの長さは、長さ属性ではなく実際の長さです。

漢字ストリングの長さは、 2 バイト文字の数 (バイト数を 2 で除算した値) になります。その他のすべての値の長さは、その値を表すのに使用するバイト数になります。

- 短精度整数の場合は 2
- 長精度整数の場合は 4
- 64 ビット整数の場合は 8
- 精度が *p* のパック 10 進数の場合は (*p*/2)+1 の整数部分
- 精度が *p* のゾーン 10 進数の場合は *p*
- 単精度浮動小数点の場合は 4
- 倍精度浮動小数点の場合は 8
- DECFLOAT(16) の場合は 8
- DECFLOAT(34) の場合は 16
- ストリングの場合はストリングの長さ
- 時刻の場合は 3
- 日付の場合は 4
- timestamp(*p*) の場合は 7+((*p*+1)/2)
- データ・リンクの場合はデータ・リンク値を保管するために実際に使用するバイト数 (データ・リンクが FILE LINK CONTROL で、しかも READ PERMISSION DB の場合は、これに 19 を加える)。
- 行 ID の場合は 26

例

- ホスト変数 ADDRESS は、値が'895 Don Mills Road' の可変長文字ストリングであると想定します。

```
SELECT LENGTH(:ADDRESS)
  FROM SYSIBM.SYSDUMMY1
```

値 18 が戻されます。

- PRSTDATE が、DATE タイプの列であるとします。

```
SELECT LENGTH(PRSTDATE)
  FROM PROJECT
```

値として 4 が戻されます。

- PRSTDATE が、DATE タイプの列であるとします。

```
SELECT LENGTH(CHAR(PRSTDATE, EUR))
  FROM PROJECT
```

値として 10 が戻されます。

LN

LN 関数は、数値の自然対数を戻します。LN 関数と EXP 関数は、互いに逆の演算になります。

expression
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。引数の値はゼロより大きくなくてはなりません。

引数のデータ・タイプが DECFLOAT(*n*) の場合、結果は DECFLOAT(*n*) です。それ以外の場合、結果のデータ・タイプは、倍精度の浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

注記

DECFLOAT 特殊値が関係する場合の結果**:** 10 進浮動小数点の場合、特殊値は次のように扱われます。

- LN(NaN) は NaN を返します。[60]
- LN(-NaN) は NaN を返します。[60]
- LN(Infinity) は Infinityを返します。
- LN(-Infinity) は NaN を返します。[60]
- LN(sNaN) および LN(-sNaN) は警告またはエラーを返します。[60]
- LN(0) は -Infinity を返します。
- -Infinity などの負の引数を持つ LN は NaN を返します。[60]

例

- ホスト変数 NATLOG は、値が 31.62 の DECIMAL(4,2) のホスト変数であると想定します。

```
SELECT LN(:NATLOG)
  FROM SYSIBM.SYSDUMMY1
```

およそ 3.45 の値が戻されます。

60. SQL_DECFLOAT_WARNINGS 照会オプションに *YES を指定すると、NaN が返され、警告が出されます。

LNOT

LNOT 関数は、引数ストリングの論理否定 (論理 NOT) であるストリングを戻します。

expression
(LOB 以外の) 任意の組み込み数値データ・タイプ、またはストリング・データ・タイプの値を戻す式。引数は、混合データ文字ストリング、UTF-8 文字ストリング、またはグラフィック・ストリングであってはなりません。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。

結果のデータ・タイプおよび長さ属性は、引数値のデータ・タイプおよび長さ属性と同じです。引数が可変長ストリングの場合、結果の実際の長さは引数値の実際の長さと同じです。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

結果の CCSID は 65535 です。

例

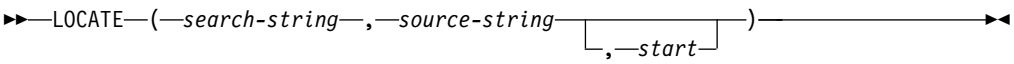
- ホスト変数 L1 は、値が X'F0F0' の CHARACTER(2) のホスト変数であると想定します。

```
SELECT LNOT(:L1)
  FROM SYSIBM.SYSDUMMY1
```

値として X'0F0F' が戻されます。

LOCATE

LOCATE 関数は、あるストリング (*search-string* と呼ばれる) の、別のストリング (*source-string* と呼ばれる) の中での、最初の出現箇所の開始位置を戻します。検索ストリング が見つからず、どちらの引数も NULL でない場合、結果はゼロになります。検索ストリング が見つかった場合、結果は 1 から ソース・ストリング の実際の長さまでの数値になります。オプションの *start* が指定されている場合、それは、*source-string* 中での検索が開始される文字位置を示します。



search-string
検索するオブジェクトのストリングを指定する式。検索ストリング には、任意の組み込み数値、日時、またはストリング式を指定できます。これは、ソース・ストリング と互換性のあるものでなければなりません。数値または日時引数は、関数を評価する前に文字ストリングにキャストされます。数値および日時から文字ストリングへの変換について詳しくは、 672 ページの『VARCHAR』を参照してください。

source-string
検索を行う相手先のソース・ストリングを指定する式。ソース・ストリング には、任意の組み込み数値、日時、またはストリング式を指定できます。数値または日時引数は、関数を評価する前に文字ストリングにキャストされます。数値および日時から文字ストリングへの変換について詳しくは、 672 ページの『VARCHAR』 を参照してください。

start
検索が開始される *source-string* 内の位置を指定する式。 *start* には、任意の組み込み数値、文字ストリング、またはグラフィック・ストリング式を指定できます。値がタイプ INTEGER でない場合、その値は関数を評価する前に暗黙的に INTEGER にキャストされます。値はゼロより大きくなければなりません。

開始桁 を指定した場合は、この関数は次と同じになります。

```
POSITION( search-string , SUBSTRING(source-string,start) ) + start - 1
```

開始桁 を指定しない場合、この関数は次と同じになります。

```
POSITION( search-string , source-string ) )
```

詳しくは、 575 ページの『POSITION』を参照してください。

この関数の結果は長精度整数になります。引数のいずれかが NULL になる可能性がある場合は、結果も NULL になる可能性があります。いずれかの引数が NULL の場合は、結果は NULL 値になります。

LOCATE 関数は文字単位で実行します。LOCATE は文字ストリング単位で実行されるため、シフトイン、シフトアウト文字がまったく同じ場所にある必要がなく、これらの文字は、どの文字が SBCS でどの文字が DBCS であるかを示すためにだけ意味があります。

検索ストリング の CCSID がソース・ストリング の CCSID と異なる場合は、ソース・ストリング の CCSID に変換されます。

LOCATE 関数を含むステートメントの実行時に *HEX 以外の照合順序が有効で、しかも引数が SBCS データ、混合データ、または Unicode データの場合、結果は、その集合の各値の重み付けされた値の比較によって求められます。値の重み付けは、該当の照合順序に基づいています。 ICU 照合順序表は、LOCATE 関数では指定できません。

例

- IN_TRAY 表の全項目から、RECEIVED 列と SUBJECT 列、それに NOTE_TEXT 列の語「GOOD」の開始位置を選択します。

```
SELECT RECEIVED, SUBJECT, LOCATE('GOOD', NOTE_TEXT)
  FROM IN_TRAY
  WHERE LOCATE('GOOD', NOTE_TEXT) <> 0
```

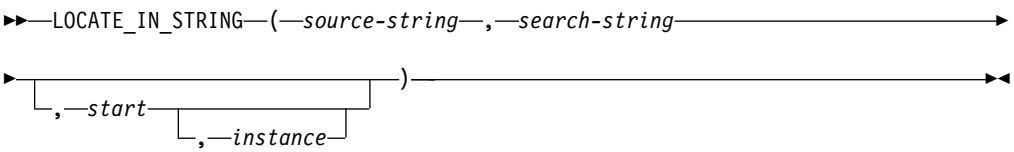
- NOTE は VARCHAR(128) の列で Unicode UTF-8 でエンコードされ、値 'Jürgen lives on Hegelstraße' を含むものと想定します。ストリング内で文字「ß」の文字位置を検索します。

```
SELECT LOCATE( 'ß', NOTE ), POSSTR( NOTE_TEXT, 'ß')
  FROM T1
```

  LOCATE の場合は値 26、POSSTR の場合には値 27 を戻します。

LOCATE_IN_STRING

LOCATE_IN_STRING 関数は、あるストリング (*source-string*、ソース・ストリングと呼ばれる) の中の、別のストリング (*search-string*、検索ストリングと呼ばれる) の開始位置を戻します。検索ストリング が見つからず、どちらの引数も NULL でない場合、結果はゼロになります。検索ストリング が見つかった場合、結果は 1 から ソース・ストリング の実際の長さまでの数値になります。オプションの *start* が指定されている場合、それは、*source-string* 中での検索が開始される文字位置を示します。



オプションの *start* が指定されている場合、それは、*source-string* 中での検索が開始される文字位置を示します。 *start* を指定した場合、オプションの *instance* 番号も指定できます。インスタンス 引数を使用して、*source-string* 内での *search-string* の特定のオカレンスが決定されます。それぞれの固有インスタンスには、前のインスタンスのいずれの文字も使用できますが、前のインスタンスのすべての文字を指定することはできません。

search-string の長さが 0 の場合、関数によって戻される結果は 1 です。*source-string* の長さが 0 の場合、関数によって戻される結果は 0 です。そのどちらでもない場合で、*search-string* 値が、*source-string* 値の中に隣接して位置する同じ長さのサブストリングと等しい場合、関数によって戻される結果は、*source-string* 値内のその最初のサブストリングの開始位置になります。その他の場合、関数によって戻される結果は 0 です。

source-string
検索を行う相手先のソース・ストリングを指定する式。ソース・ストリング には、任意の組み込み数値、日時、またはストリング式を指定できます。数値引数または日時引数は、関数を評価する前に文字ストリングにキャストされます。数値および日時から文字ストリングへの変換について詳しくは、 672 ページの『VARCHAR』 を参照してください。

search-string
検索するオブジェクトのストリングを指定する式。検索ストリング には、任意の組み込み数値、日時、またはストリング式を指定できます。これは、ソース・ストリング と互換性のあるものでなければなりません。数値引数または日時引数は、関数を評価する前に文字ストリングにキャストされます。数値および日時から文字ストリングへの変換について詳しくは、 672 ページの『VARCHAR』を参照してください。

start
検索が開始される *source-string* 内の位置を指定する式。 *start* には、任意の組み込み数値、文字ストリング、またはグラフィック・ストリング式を指定できます。値がタイプ INTEGER でない場合、その値は関数を評価する前に暗黙的に INTEGER にキャストされます。

この整数の値が 0 より大きい場合、検索は *start* の位置から開始され、ストリングの末尾にいたるまで各位置で実行されます。この整数の値が 0 より小さい場合、検索は CHARACTER_LENGTH(*source-string*) + *start* + 1 の位置から開始され、ストリングの先頭にいたるまで、各位置で実行されます。

開始桁 を指定しない場合、この関数は次と同じになります。

```
POSITION( search-string , source-string )
```

start が 0 の場合は、エラーが戻されます。

instance

source-string の中の検索する *search-string* のインスタンスを指定する式。式は、組み込み数値、文字ストリング、またはグラフィック・ストリングのいずれかのデータ・タイプの値を戻す必要があります。値がタイプ INTEGER でない場合、その値は関数を評価する前に暗黙的に INTEGER にキャストされます。*instance* を指定しない場合、デフォルトは 1 です。この整数の値は 1 以上である必要があります。

各検索位置で、その検索位置から CHARACTER_LENGTH (*search-string*) - 1 の値だけ右にある位置までの *source-string* のサブストリングが、*search-string* と等しい場合に、一致が検出されます。

この関数の結果は長精度整数になります。結果は、*source-string* 中の *search-string* のインスタンスの開始位置です。この値は、*start* 指定に関わらず、ストリングの開始位置を意味します。

引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

LOCATE_IN_STRING は文字を基本として機能します。LOCATE_IN_STRING は文字ストリング単位で実行されるため、シフトイン、シフトアウト文字がまったく同じ場所にある必要がなく、これらの文字は、どの文字が SBCS でどの文字が DBCS であるかを示すためにだけ意味があります。

検索ストリング の CCSID がソース・ストリング の CCSID と異なる場合は、ソース・ストリング の CCSID に変換されます。

LOCATE_IN_STRING 関数を含むステートメントの実行時に *HEX 以外の照合順序が有効で、しかも引数が SBCS データ、混合データ、または Unicode データの場合、結果は、その集合の各値の重み付けされた値の比較によって求められます。値の重み付けは、該当の照合順序に基づいています。 ICU 照合順序表は、LOCATE 関数では指定できません。

代替構文**:** INSTR は LOCATE_IN_STRING のシノニムとして使用できます。

例

- ストリング「Jürgen lives on Hegelstraße」内の文字「ß」の位置を、ストリング末尾から検索して特定し、そのストリング内での位置をホスト変数 POSITION に設定します。

```
SET :POSITION = LOCATE_IN_STRING('Jürgen lives on Hegelstraße','ß',-1);
```

ホスト変数 POSITION の値は 26 に設定されます。

- ストリングの開始位置から検索することにより、ストリング「WINNING」内で文字「N」が現れる位置を検索します。

```
SELECT LOCATE_IN_STRING('WINNING','N',1,3),
       LOCATE_IN_STRING('WINNING','N',3,2),
       LOCATE_IN_STRING('WINNING','N',3,3)
FROM SYSIBM.SYSDUMMY1;
```

次の値を戻します。

```
6     4     6
```

- ストリングの末尾から検索することにより、ストリング「WINNING」内で文字「N」が現れる位置を検索します。

```
SELECT LOCATE_IN_STRING('WINNING','N',-1,3),
       LOCATE_IN_STRING('WINNING','N',-3,2),
       LOCATE_IN_STRING('WINNING','N',-3,3)
FROM SYSIBM.SYSDUMMY1;
```

次の値を戻します。

```
3     3     0
```

LOG10

LOG10 関数は、数値の共通対数 (底 10) を戻します。 LOG10 関数と ANTILOG 関数は、逆の演算です。

expression
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

引数のデータ・タイプが DECFLOAT(*n*) の場合、結果は DECFLOAT(*n*) です。それ以外の場合、結果のデータ・タイプは、倍精度の浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

注記

DECFLOAT 特殊値が関係する場合の結果: 10 進浮動小数点の場合、特殊値は次のように扱われます。

- LOG10(NaN) は NaN を返します。[61]
- LOG10(-NaN) は NaN を返します。[61]
- LOG10(Infinity) は Infinity を返します。
- LOG10(-Infinity) は NaN を返します。[61]
- LOG10(sNaN) および LOG10(-sNaN) は警告またはエラーを返します。[61]
- LOG10(0) は -Infinity を返します。
- -Infinity などの負の引数を持つ LOG10 は NaN を返します。[61]

代替構文**:** LOG は LOG10 の同義語です。これは、Db2 の旧リリースとの互換性を維持するためにのみサポートされています。データベース・マネージャーおよびアプリケーションによっては、LOG を数値の共通対数ではなく数値の自然対数として設定しているものがあるため、LOG の代わりに LOG10 を使用してください。

例

- ホスト変数 L は、値が 31.62 の DECIMAL(4,2) のホスト変数であると想定します。

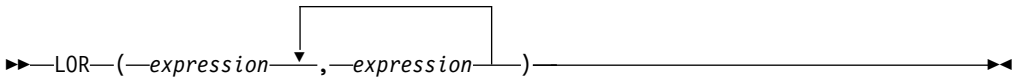
```
SELECT LOG10(:L)
  FROM SYSIBM.SYSDUMMY1
```

およそ 1.49 の値が戻されます。

61. SQL_DECFLOAT_WARNINGS 照会オプションに *YES を指定すると、NaN が返され、警告が出されます。

LOR

LOR 関数は、引数ストリングの論理和 (論理 OR) のストリングを戻します。この関数は、まず最初の引数ストリングと次のストリングとの OR 演算を行い、その後次々に得られた結果を使用して次の引数との OR 演算を行っていきます。文字ストリング引数が前の結果より短い場合は、ブランクが埋め込まれます。 2 進ストリング引数が前の結果より短い場合は、16 進数のゼロが埋め込まれます。



各引数には、互換性がなければなりません。

expression
(LOB 以外の) 任意の組み込み数値データ・タイプ、またはストリング・データ・タイプの値を戻す式。引数は、混合データ文字ストリング、UTF-8 文字ストリング、またはグラフィック・ストリングであってはなりません。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。

必要ならば、引数は結果の属性に変換されます。結果の属性は、以下のように決められます。

- すべての引数が固定長ストリングである場合、結果は長さが *n* の固定長ストリングになります (ここで、*n* は最長の引数の長さ)。
- 引数の中に可変長ストリングがある場合、結果は長さ属性が *n* の可変長ストリングになります (ここで、*n* は最大の長さ属性を持つ引数の長さ属性)。結果の実際の長さは *m* です (ここで、*m* は、最長の引数の実際の長さ)。

引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

結果の CCSID は 65535 です。

例

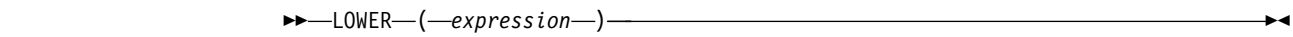
- ホスト変数 L1 は値が X'0101' の CHARACTER(2) のホスト変数、ホスト変数 L2 は値が X'F0F000' の CHARACTER(3) のホスト変数、ホスト変数 L3 は値が X'0000000F' の CHARACTER(4) のホスト変数であると想定します。

```
SELECT LOR(:L1,:L2,:L3)
  FROM SYSIBM.SYSDUMMY1
```

値として X'F1F1404F' が返されます。

LOWER

LOWER 関数は、すべての文字を引数の CCSID に基づいて小文字に変換したストリングを戻します。 SBCS、Unicode グラフィック文字だけが変換されます。 A から Z の文字は a から z に変換され、発音記号がある場合はそれぞれの下段シフトに変換されます。



この変換に使用する大文字変換表については、「グローバリゼーション」トピック集にあるトピック UCS-2 レベル 1 マッピング・テーブルを参照してください。

expression
変換するストリングを指定する式。*expression* は、任意の組み込み数値、文字、Unicode グラフィック・ストリングでなければなりません。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。

この関数の結果のデータ・タイプ、長さ属性、実際の長さ、および CCSID は、引数と同じになります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL である場合は、結果は NULL 値です。

注記

代替構文: LCASE は LOWER の同義語です。

例

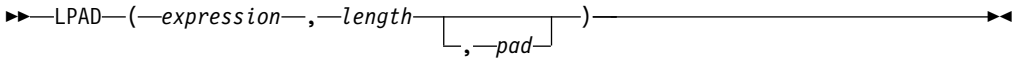
- ホスト変数 NAME の値に入っている文字を確実に小文字にしたいとします。NAME はデータ・タイプ VARCHAR(30)、値は「Christine Smith」です。

```
SELECT LOWER(:NAME)
    FROM SYSIBM.SYSDUMMY1
```

結果は、値「christine smith」です。

LPAD

LPAD 関数は、左側に埋め込みが行われた *expression* で構成されるストリングを戻します。



LPAD 関数は、*expression* 内の先行ブランクまたは末尾ブランクを有効として扱います。埋め込みは、*expression* の実際の長さが *length* より短く、*pad* が空ストリングでない場合のみ行われます。

expression
結果が導き出される元になるストリングを指定する式。

expression は、組み込みストリング、数値、または日時のデータ・タイプでなければなりません。数値引数または日時引数は、関数を評価する前に、現行サーバーでデフォルト SBCS CCSID である CCSID を使用して VARCHAR にキャストされます。数値または日時からさまざまな文字ストリングへの変換について詳しくは、672 ページの『VARCHAR』 を参照してください。

length
結果の長さを指定する式。式は、組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す必要があります。式のデータ・タイプが INTEGER でない場合、その値は関数を評価する前に暗黙的に INTEGER にキャストされます。値はゼロであるか、*n* 以下の正整数である必要があります。ここで、*n* は結果のデータ・タイプの最大長です。詳しくは、1845 ページの『付録 A. SQL の制約』を参照してください。

expression がグラフィック・ストリングの場合、*length* は DBCS または Unicode グラフィック文字の数を示します。 *expression* が文字ストリングの場合、*length* は文字数を示します (1 文字が 1 バイトまたは複数バイトで構成されることもあります)。 *expression* がバイナリー・ストリングの場合、*length* はバイト数を示します。

pad
埋め込むストリングを指定する式。この式は、組み込みデータ・タイプであるストリング、数値、日時のいずれかの値を戻す必要があります。値が数値または日時のデータ・タイプの場合、関数を評価する前に、現行サーバーでデフォルト SBCS CCSID である CCSID を使用して VARCHAR に暗黙的にキャストされます。

pad が指定されていない場合、埋め込み文字は次のように設定されます。

- 文字ストリングおよびグラフィック・ストリングの場合、*expression* のデータ・タイプと CCSID に基づいた 1 バイト、2 バイト、 UTF-16、または UTF-8 のブランク文字。[62]
- バイナリー・ストリングの場合、16 進のゼロ。

62. UTF-16 または UCS-2 では、コード・ポイント X'0020' および X'3000' でブランク文字を定義しています。データベース・マネージャーは、コード・ポイント X'0020' の位置にあるブランクを埋め込みに使用します。データベース・マネージャーは、UTF-8 でコード・ポイント X'20' のブランクを埋め込みます。

expression の値と *pad* の値は、互換性のあるデータ・タイプである必要があります。*pad* の CCSID が *expression* の CCSID と異なる場合、*pad* 値は *expression* の CCSID に変換されます。データ・タイプの互換性についての詳細は、 113 ページの『割り当ておよび比較』を参照してください。

結果のデータ・タイプは、式 のデータ・タイプによって異なります。

式 のデータ・タイプ	**LPAD** の場合の結果のデータ・タイプ
CHAR や VARCHAR または数値や日時	VARCHAR
CLOB	CLOB
GRAPHIC または VARGRAPHIC	VARGRAPHIC
DBCLOB	DBCLOB
BINARY または VARBINARY	VARBINARY
BLOB	BLOB

結果の長さ属性は *length* によって決まります。長さ をゼロより大きい整数定数で明示的に指定すると、結果の長さ属性は長さ になります。*length* にゼロの整数定数を明示的に指定すると、結果の長さ属性は 1 になります。 *length* を式で指定すると、結果の長さ属性は、*m*+100 と、結果データ・タイプの最大長のうち、小さい方になります。ここで、*m* は *expression* の長さ属性です。詳しくは、 1845 ページの『付録 A. SQL の制約』を参照してください。

結果の実際の長さは、*length* から決定されます。

- *length* が 0 の場合、実際の長さは 0 であり、結果は空の結果ストリングになります。
- *length* が *expression* の実際の長さに等しい場合、実際の長さは、*expression* の長さになります。
- *length* が *expression* の実際の長さよりも小さい場合、結果は切り捨てられます。実際の長さは、結果のデータ・タイプが可変長混合データまたは可変長 Unicode であるケースを除き、*length* になります。この場合、切り捨ては常に、完全な文字を単位として行われます。
  - Unicode データでは、2 バイト文字が分断されないように、実際の長さは *length*-1 になることがあります。
  - 混合データでは、2 バイト文字や、「シフトイン」文字 (X'0F') および「シフトアウト」文字 (X'0E') で発生しうる切り捨てのために、実際の長さは *length*-3 まで小さくなることがあります。
- *length* が *expression* の実際の長さより大きい場合、結果のデータ・タイプが可変長混合データまたは可変長 Unicode で *pad* に 2 バイト文字が含まれるケースを除き、実際の長さは *length* になります。この場合、埋め込みは常に、完全な文字を単位として行われます。
  - Unicode データでは、2 バイト文字が分断されないように、実際の長さは *length*-1 になることがあります。

– 混合データでは、2 バイト文字や、「シフトイン」文字 (X'0F') および「シフトアウト」文字 (X'0E') で発生しうる切り捨てのために、実際の長さは *length*-3 まで小さくなることがあります。また、その結果の「継ぎ目に」余分なシフト・コードが入ることはありません。したがって、pad が「シフトイン文字」文字 (X'0F') で終わるストリングで、expression が「シフトアウト」文字 (X'0E') で始まる場合、これらの 2 バイト (pad のシフトイン文字と expression のシフトアウト文字) は結果から除去されます。

引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

結果の CCSID は式 の CCSID と同じです。

例

- 例 *1:* NAME が VARCHAR(15) の列で、値「Chris」、「Meg」、および「Jeff」を含んでいるものとします。次の照会では、値の左側にピリオドが完全に埋め込まれます。

  ```
  SELECT LPAD(NAME,15,'.' ) AS NAME FROM T1;
  ```

  これは、以下のものを戻します。

  ```
  NAME
  ---------------
  ..........Chris
  ............Meg
  ...........Jeff
  ```

- 例 *2:* NAME が VARCHAR(15) の列で、値「Chris」、「Meg」、および「Jeff」を含んでいるものとします。次の照会では、長さ 5 までのみ各値にピリオドが埋め込まれます。

  ```
  SELECT LPAD(NAME,5,'.' ) AS NAME FROM T1;
  ```

  これは、以下のものを戻します。

  ```
  NAME
  -----
  Chris
  ..Meg
  .Jeff
  ```

- 例 *3:* NAME が CHAR(15) の列で、値「Chris」、「Meg」、および「Jeff」を含んでいるものとします。NAME は固定長の文字フィールドで、既にブランクが埋め込まれているため、LPAD 関数は埋め込みを行いません。ただし、結果の長さが 5 であるため、列の切り捨てが行われます。

  ```
  SELECT LPAD(NAME,5,'.' ) AS NAME FROM T1;
  ```

  これは、以下のものを戻します。

  ```
  NAME
  -----
  Chris
  Meg
  Jeff
  ```

- 例 *4:* NAME が VARCHAR(15) の列で、値「Chris」、「Meg」、および「Jeff」を含んでいるものとします。場合によって、指定された埋め込みストリングの部分的なインスタンスが戻されます。

```
SELECT LPAD(NAME,15,'123' ) AS NAME FROM T1;
```

これは、以下のものを戻します。

```
NAME
---------------
1231231231Chris
123123123123Meg
12312312312Jeff
```

- 例 *5:* NAME が VARCHAR(15) の列で、値「Chris」、「Meg」、および「Jeff」を含んでいるものとします。「Chris」には切り捨て、「Meg」には埋め込みが行われ、「Jeff」は変更されません。

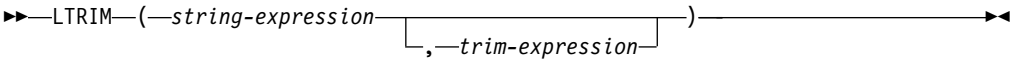
```
SELECT LPAD(NAME,4,'.' ) AS NAME FROM T1;
```

これは、以下のものを戻します。

```
NAME
----
Chri
.Meg
Jeff
```

LTRIM

LTRIM 関数は、指定した文字のいずれかを式の先頭から除去します。



LTRIM 関数は、*string-expression* の先頭から、*trim-expression* に含まれるすべての文字を除去します。照合順序は、検索に影響しません。*string-expression* が FOR BIT DATA として定義される場合、またはバイナリー・データ・タイプである場合、検索は、*trim-expression* に含まれる各バイトを *string-expression* の先頭にあるバイトと比較することによって行われます。

string-expression
任意の組み込み数値データ・タイプ、日時データ・タイプ、またはストリング・データ・タイプの値を戻す式。数値または日時の引数は、関数の評価前に文字ストリングにキャストされます。[63] 数値または日時から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。

trim-expression
string-expression の先頭から除去する文字を指定する式。式は任意の組み込み数値、日時、またはストリングのデータ・タイプの値を戻す必要があります。数値または日時引数は、関数を評価する前に文字ストリングにキャストされます。

trim-expression を指定しない場合、使用するデフォルト値は、*string-expression* のデータ・タイプによって以下のように決まります。

- 16 進数のゼロ (X'00') (引数がバイナリー・ストリングの場合)
- DBCS のブランク (引数が DBCS グラフィック・ストリングの場合)
- UTF-16 または UCS-2 のブランク (最初の引数が Unicode グラフィック・ストリング・ストリングの場合)
- UTF-8 のブランク (最初の引数が UTF-8 文字ストリングの場合)
- それ以外の場合は、SBCS のブランク。

string-expression の値と *trim-expression* の値は、互換性のあるデータ・タイプである必要があります。データ・タイプの互換性についての詳細は、 113 ページの『割り当ておよび比較』を参照してください。 *string-expression* と *trim-expression* の CCSID が異なる場合、*trim-expression* の CCSID は *string-expression* の CCSID に変換されます。

結果のデータ・タイプは、ストリング式 のデータ・タイプによって異なります。

string-expression のデータ・タイプ	結果のデータ・タイプ
CHAR または VARCHAR	VARCHAR
CLOB	CLOB
GRAPHIC または VARGRAPHIC	VARGRAPHIC
DBCLOB	DBCLOB
BINARY または VARBINARY	VARBINARY

63. 引数を 1 つ指定した LTRIM 関数は、STRIP(*string-expression*,LEADING) と同じ結果を返します。

string-expression のデータ・タイプ	結果のデータ・タイプ
BLOB	BLOB

結果の長さ属性は *string-expression* の長さ属性と同じになります。文字ストリングまたはバイナリー・ストリングの場合、結果の実際の長さは、除去されるバイト数を *string-expression* から引いた長さになります。結果がグラフィック・ストリングである場合の実際の長さは、除去されるグラフィック文字数を *string-expression* の長さから引いた値になります。すべての文字が除去された場合は、結果は空のストリングになります。

引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

結果の CCSID は*string-expression* の CCSID と同じです。

例

- 例 *1:* ホスト変数 HELLO が CHAR(9) として定義されており、値が以下であるときに、LTRIM 関数を使用します。

```
'    Hello'
```

```
VALUES LTRIM(:HELLO)
```

  結果は 'Hello' になります。*trim-expression* が指定されていないときは、ブランクのみが除去されます。

- 例 *2:* LTRIM 関数を使用して、*trim-expression* 内の個々の数値を *string-expression* の先頭 (左側) から除去します。

```
SELECT LTRIM ('123DEFG123', '321'),
    LTRIM ('12DEFG123', '321'),
    LTRIM ('123123222XYZ22', '123'),
    LTRIM ('12321', '213'),
    LTRIM ('XYX123 ', '321')
   FROM SYSIBM.SYSDUMMY1
```

  結果は、次のとおりです。

```
'DEFG123'
'DEFG123'
'XYZ22'
'' (an empty string - all characters removed)
'XYX123' (no characters removed)
```

  LTRIM 関数は、「1」、「2」、または「3」ではない文字の後に続く、ストリング右側の「1」、「2」、および「3」のインスタンスを除去しません。

- 例 *3:* LTRIM 関数を使用して、*trim-expression* に指定した文字を *string-expression* の先頭から除去します。

```
VALUES LTRIM(('...$V..$AR', '$.'))
```

  結果は、'V..$AR' になります。関数は、*trim-expression* に指定されていない文字を検出すると停止します。

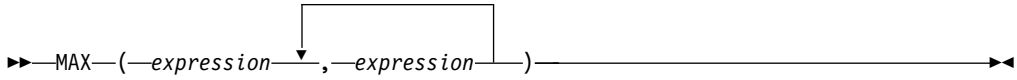
- 例 *4:* LTRIM 関数を使用して、*trim-expression* に指定した文字を *string-expression* の先頭から除去します。

```
VALUES LTRIM('[[ -78]]', '- []')
```

結果は、'78]]' になります。文字とブランクを除去するときは、*trim-expression* にブランクを含める必要があります。

MAX

MAX スカラー関数は、値の集合の中の最大値を戻します。



各引数には、互換性がなければなりません。文字ストリングの引数は、日付/時刻の値と互換性があります。引数をデータ・リンク値および XML 値とすることはできません。

expression
任意の組み込み数値データ・タイプ、またはストリング・データ・タイプの値を戻す式。引数のうちの 1 つが数値である場合は、文字およびグラフィック・ストリング引数は、関数を評価する前に数値にキャストされます。

この関数の結果は、最大の引数値になります。結果が NULL になる可能性があるのは、少なくとも 1 つの引数が NULL になる可能性がある場合です。結果が NULL 値になるのは、引数の 1 つが NULL の場合です。

選択された引数は、必要があれば、結果の属性に変換されます。結果の属性は、133 ページの『結果のデータ・タイプに関する規則』で説明しているすべてのオペランドを基にして決められます。

ステートメントの実行時点で *HEX 以外の照合順序が有効で、しかも引数が SBCS データ、混合データ、または Unicode データの場合には、ストリングの重み付けされた値が、実際の値の代わりに比較されます。値の重み付けは、該当の照合順序に基づいています。

注記

代替構文: GREATEST は MAX のシノニムとして指定できます。

例

- ホスト変数 M1 は値が 5.5 の DECIMAL(2,1) のホスト変数、ホスト変数 M2 は値が 4.5 の DECIMAL(3,1) のホスト変数、ホスト変数 M3 は値が 6.25 の DECIMAL(3,2) のホスト変数であると想定します。

```
SELECT MAX(:M1,:M2,:M3)
  FROM SYSIBM.SYSDUMMY1
```

  値として 6.25 が戻されます。

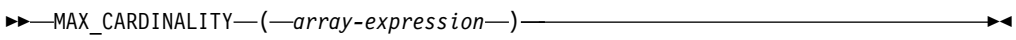
- ホスト変数 M1 は値「AA」の CHARACTER(2) のホスト変数、ホスト変数 M2 は値「AA 」の CHARACTER(3) のホスト変数、ホスト変数 M3 は値「AA A」の CHARACTER(4) のホスト変数であると想定します。

```
SELECT MAX(:M1,:M2,:M3)
  FROM SYSIBM.SYSDUMMY1
```

  結果として「AA A」の値が戻されます。

MAX_CARDINALITY

MAX_CARDINALITY 関数は、配列に含めることのできるエレメントの最大数を表わす値を戻します。この値は、ユーザー定義の配列タイプに対して CREATE TYPE (配列) ステートメントで指定したカーディナリティーです。



array-expression
この式は、配列データ・タイプの SQL 変数またはパラメーターでも、配列データ・タイプに対するパラメーター・マーカーのキャスト仕様であっても構いません。

関数の結果は BIGINT です。結果が NULL になることはありません。

例

配列タイプ PHONENUMBERS および配列変数 RECENT_CALLS が次のように定義されていると想定します。

```
CREATE TYPE PHONENUMBERS AS INTEGER ARRAY[50];
DECLARE RECENT_CALLS PHONENUMBERS;
```

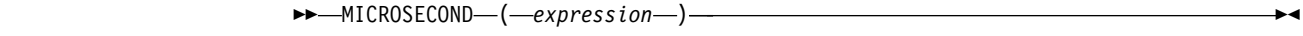
以下のステートメントは、RECENT_CALLS が定義されたときの最大カーディナリティーを LIST_SIZE に設定します。

```
SET LIST_SIZE = MAX_CARDINALITY(RECENT_CALLS)
```

このステートメントを実行した後、LIST_SIZE には 50 が入っています。

MICROSECOND

MICROSECOND 関数は、値のマイクロ秒の部分を戻します。



expression
日付、時刻、タイム・スタンプ、文字ストリング、グラフィック・ストリング、または数値のいずれかの組み込みデータ・タイプの値を戻す式。

- *expression* が文字ストリングまたはグラフィック・ストリングである場合、その値は、日時値の有効なストリング表現でなければなりません。式が日付の有効なストリング表現の場合、その形式は IBM SQL 標準形式のいずれかでなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。
- 引数が DATE である場合、時刻が午前 0 時ちょうど (00.00.00) であると想定して、最初に TIMESTAMP(0) 値に変換されます。
- *expression* が数値の場合、この数値はタイム・スタンプ期間であることが必要です。日時期間の有効な形式については、 205 ページの『日付/時刻のオペランドと期間』を参照してください。

この関数の結果は長精度整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

その他の規則は、引数のデータ・タイプに応じて以下のように異なります。

- 引数が日付、時刻、またはタイム・スタンプであるか、日付、時刻、またはタイム・スタンプの有効な文字ストリング表現である場合:

  結果は、0 から 999999 までの整数です。

  タイム・スタンプの精度が 6 を超える場合、値は切り捨てられます。
- 引数が期間の場合 :

  結果は、値のマイクロ秒の部分 (-999999 から 999999 までの整数) になります。ゼロ以外の結果の符号は、引数と同じになります。

例

- 表 TABLEA に、タイプが TIMESTAMP の TS1 および TS2 という 2 つの列が入っているものとします。 TS1 のマイクロ秒部分がゼロではなく、 TS1 と TS2 の秒部分が同じである行すべてを選択します。

```
SELECT *
  FROM TABLEA
  WHERE MICROSECOND(TS1) <> 0 AND SECOND(TS1) = SECOND(TS2)
```

MIDNIGHT_SECONDS

MIDNIGHT_SECONDS 関数は、真夜中から引数に指定されている時刻値までの秒数を表す 0 から 86 400 の整数値を戻します。

expression

日付、時刻、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式。値は、時刻またはタイム・スタンプの有効なストリング表現でなければなりません。時刻とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

引数が DATE である場合、時刻が午前 0 時ちょうど (00.00.00) であると想定して、最初に TIMESTAMP(0) 値に変換されます。

この関数の結果は、長整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

- 真夜中から 00:01:00 までの間、および真夜中から 13:10:10 までの間の秒数を調べます。ホスト変数 XTIME1 の値は「00:01:00」であり、XTIME2 の値は「13:10:10」であるものとします。

```
SELECT MIDNIGHT_SECONDS(:XTIME1), MIDNIGHT_SECONDS(:XTIME2)
  FROM SYSIBM.SYSDUMMY1
```

  この例は、60 と 47410 を戻します。 1 分は 60 秒、1 時間は 3600 秒なので、00:01:00 は真夜中から 60 秒後 ((60 * 1) + 0)、13:10:10 は 47410 秒後 ((3600 * 13) + (60 * 10) + 10) になります。

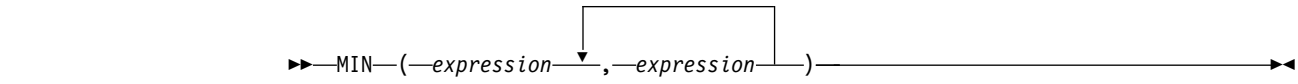
- 真夜中から 24:00:00 までの間、および真夜中から 00:00:00 までの間の秒数を調べます。

```
SELECT MIDNIGHT_SECONDS('24:00:00'), MIDNIGHT_SECONDS('00:00:00')
  FROM SYSIBM.SYSDUMMY1
```

  この例は、86400 と 0 を戻します。この 2 つの値は同じ時刻点を表していますが、異なる値が戻されます。

MIN

MIN スカラー関数は、値の集合の中の最小値を戻します。



各引数には、互換性がなければなりません。文字ストリングの引数は、日付/時刻の値と互換性があります。引数をデータ・リンク値および XML 値とすることはできません。

expression
任意の組み込み数値データ・タイプ、またはストリング・データ・タイプの値を戻す式。引数のうちの 1 つが数値である場合は、文字およびグラフィック・ストリング引数は、関数を評価する前に数値にキャストされます。

関数の結果は、最小の引数値です。結果が NULL になる可能性があるのは、少なくとも 1 つの引数が NULL になる可能性がある場合です。結果が NULL 値になるのは、引数の 1 つが NULL の場合です。

選択された引数は、必要があれば、結果の属性に変換されます。結果の属性は、133 ページの『結果のデータ・タイプに関する規則』で説明しているすべてのオペランドを基にして決められます。

ステートメントの実行時点で *HEX 以外の照合順序が有効で、しかも引数が SBCS データ、混合データ、または Unicode データの場合には、ストリングの重み付けされた値が、実際の値の代わりに比較されます。値の重み付けは、該当の照合順序に基づいています。

注記

代替構文: LEAST は MIN のシノニムとして指定できます。

例

- ホスト変数 M1 は値が 5.5 の DECIMAL(2,1) のホスト変数、ホスト変数 M2 は値が 4.5 の DECIMAL(3,1) のホスト変数、ホスト変数 M3 は値が 6.25 の DECIMAL(3,2) のホスト変数であると想定します。

```
SELECT MIN(:M1,:M2,:M3)
  FROM SYSIBM.SYSDUMMY1
```

  値として 4.50 が戻されます。

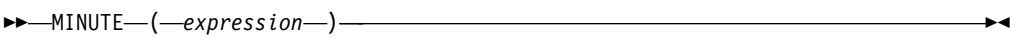
- ホスト変数 M1 は値「AA」の CHARACTER(2) のホスト変数、ホスト変数 M2 は値「AAA」の CHARACTER(3) のホスト変数、ホスト変数 M3 は値「AAAA」の CHARACTER(4) のホスト変数であると想定します。

```
SELECT MIN(:M1,:M2,:M3)
  FROM SYSIBM.SYSDUMMY1
```

  結果として「AA 」の値が戻されます。

MINUTE

MINUTE 関数は、指定した値の分の部分を戻します。



expression
日付、時刻、タイム・スタンプ、文字ストリング、グラフィック・ストリング、または数値のいずれかの組み込みデータ・タイプの値を戻す式。

- 式 が文字ストリングまたはグラフィック・ストリングである場合、その値は、日時値の有効なストリング表現でなければなりません。式が日付の有効なストリング表現の場合、その形式は IBM SQL 標準形式のいずれかでなければなりません。日時値のストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。
- 引数が DATE である場合、時刻が午前 0 時ちょうど (00.00.00) であると想定して、最初に TIMESTAMP(0) 値に変換されます。
- 式 が数値である場合は、その数値は時刻期間またはタイム・スタンプ期間でなければなりません。日時期間の有効な形式については、 205 ページの『日付/時刻のオペランドと期間』を参照してください。

この関数の結果は長精度整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

その他の規則は、引数のデータ・タイプに応じて以下のように異なります。

- 引数が日付、時刻、またはタイム・スタンプであるか、または、日付、時刻、またはタイム・スタンプの有効な文字ストリング表現である場合:

  結果は、指定した値の分の部分 (0 から 59 までの整数) になります。
- 引数が時刻期間またはタイム・スタンプ期間の場合 :

  結果は、指定した値の分の部分 (-99 から 99 までの整数) になります。ゼロ以外の結果の符号は、引数と同じになります。

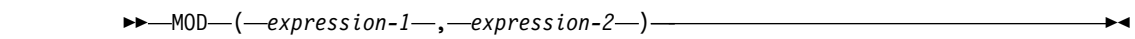
例

- CL_SCHED サンプル表を使用して、授業時間が 50 分未満のクラスを全選択します。

```
SELECT *
  FROM CL_SCHED
  WHERE HOUR(ENDING - STARTING) = 0 AND
      MINUTE(ENDING - STARTING) < 50
```

MOD

MOD 関数は、最初の引数を 2 番目の引数で割って、その剰余を戻します。



剰余の算出には、次の式が使用されます。

```
MOD(x,y) = x - (x/y) * y
```

x/y は、除算の結果を切り捨てた整数です。結果が負の値になるのは、最初の引数が負の場合だけです。

expression-1
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

expression-2
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。いずれかの引数が 10 進浮動小数点である場合を除いて、*expression-2* はゼロであってはなりません。

引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

結果の属性は、以下のように決められます。

- 両方の引数が位取りがゼロの長整数または短整数の場合、結果のデータ・タイプは長整数になります。
- 両方の引数が位取りがゼロの整数であり、少なくとも一方の引数が 64 ビット整数の場合、結果のデータ・タイプは 64 ビット整数になります。
- 一方の引数が位取りがゼロの整数で、他方が 10 進数である場合、結果は、10 進数の引数と同じ精度と位取りの 10 進数になります。
- 両方の引数が 10 進数または位取りを伴う整数の場合、結果は 10 進数になります。結果の精度は MIN(p-s,p'-s') + MAX(s,s') で、結果の位取りは MAX(s,s') になります。ここで、記号 p と s は第 1 オペランドの精度と位取りを表し、p' と s' は第 2 オペランドの精度と位取りを表します。
- いずれかの引数が浮動小数点数で、もう一方のオペランドが 10 進浮動小数点数ではない場合、結果のデータ・タイプは倍精度浮動小数点数になります。

  演算は浮動小数点数で実行されます。オペランドは、必要であれば、まずはじめに倍精度浮動小数点数に変換されます。例えば、浮動小数点数と整数または 10 進数のいずれかが関係する演算は、整数または 10 進数を一時的に倍精度浮動小数点数に変換したそのコピーを使って行われます。浮動小数点数演算の結果は、浮動小数点数の値の範囲内になければなりません。

- いずれかの引数が 10 進浮動小数点の場合、結果のデータ・タイプは DECFLOAT(34) となります。引数が特殊 10 進浮動小数点値の場合は、算術演算の一般的な規則が適用されます。詳しくは、 200 ページの『DECFLOAT の一般的な算術演算規則』を参照してください。

いずれかの引数が 10 進浮動小数点であり、2 番目のオペランドの評価が 0 になる場合、結果は NaN であり、無効演算を示す警告 (SQLSTATE 0168D) が発行されます。[64] MOD(1, -Infinity) は値 1 を戻します。

例

- ホスト変数 M1 は値が 5 の整数のホスト変数であり、ホスト変数 M2 は値が 2 の整数のホスト変数であると想定します。

```
SELECT MOD(:M1,:M2)
  FROM SYSIBM.SYSDUMMY1
```

  値として 1 が戻されます。

- ホスト変数 M1 は値が 5 の整数のホスト変数であり、ホスト変数 M2 は値が 2.20 の DECIMAL(3,2) ホスト変数であると想定します。

```
SELECT MOD(:M1,:M2)
  FROM SYSIBM.SYSDUMMY1
```

  値として 0.60 が戻されます。

- ホスト変数 M1 は値が 5.50 の DECIMAL(4.2) のホスト変数であり、ホスト変数 M2 は値が 2.0 の DECIMAL(4.1) のホスト変数であると想定します。

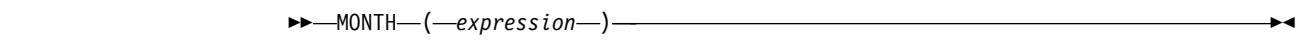
```
SELECT MOD(:M1,:M2)
  FROM SYSIBM.SYSDUMMY1
```

  値として 1.50 が戻されます。

64. SQL_DECFLOAT_WARNINGS 照会オプションに *YES が指定されている場合、NaN が戻され、警告が出されます。それ以外の場合、ゼロ除算警告またはエラーが戻されます。

MONTH

MONTH 関数は、指定した値の月の部分を戻します。



expression
日付、タイム・スタンプ、文字ストリング、グラフィック・ストリング、または数値のいずれかの組み込みデータ・タイプの値を戻す式。

- *expression* が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。
- 式 が数値である場合は、その数値は日付期間またはタイム・スタンプ期間でなければなりません。日時期間の有効な形式については、 205 ページの『日付/時刻のオペランドと期間』を参照してください。

この関数の結果は長精度整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

その他の規則は、引数のデータ・タイプに応じて以下のように異なります。

- 引数が日付、タイム・スタンプ、または、日付またはタイム・スタンプの有効な文字ストリング表現である場合 :

  結果は、指定した値の月の部分 (1 から 12 までの整数) になります。
- 引数が日付期間またはタイム・スタンプ期間の場合 :

  結果は、指定した値の月の部分 (-99 から 99 までの整数) になります。ゼロ以外の結果の符号は、引数と同じになります。

例

- 表 EMPLOYEE から、誕生日 (BIRTHDATE) が 12 月である社員に関する行をすべて選択します。

```
SELECT *
  FROM EMPLOYEE
  WHERE MONTH(BIRTHDATE) = 12
```

MONTHNAME

引数の月の部分の月の名前 (例えば、January) を含む大/小文字混合の文字ストリングを返します。

expression
日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式。

expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

この関数の結果は VARCHAR(100) になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

結果の CCSID は、現行サーバーのデフォルトの CCSID になります。

注記

各国語の考慮事項**:** 戻される月の名前は、ジョブのメッセージに使用される言語に基づいています。月の名前は、ライブラリー *LIBL の中のメッセージ・ファイル QCPFMSG のメッセージ CPX3BC0 から検索されます。

例

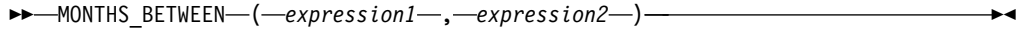
- 使用される言語が米国英語であると想定します。

```
SELECT MONTHNAME( '2003-01-02' )
  FROM SYSIBM.SYSDUMMY1
```

結果は「January」になります。

MONTHS_BETWEEN

MONTHS_BETWEEN 関数は、*expression1* および *expression2* の間の概算月数を返します。



expression1
日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を返す式。

expression1 が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

expression2
日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を返す式。

expression2 が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

expression1 が *expression2* より後の日付を表す場合、結果は正数になります。
expression2 が *expression1* 以降の日付を表す場合、結果は負数になります。

- *expression1* と *expression2* がその月の同じ日の日付または タイム・スタンプを表す場合、またはその月の最後の日を表す場合、あるいはどちらの引数もそれぞれの月の最後の日を表す場合、結果は、タイム・スタンプ引数の時刻部分は無視して、年と月の値に基づいて差を算出した整数となります。
- その他の場合、結果の整数部分は、年と月の値に基づいて算出された差となります。結果の小数部分は、どの月も 31 日まであるという想定に基づいた残りの日数から算出されます。いずれかの引数がタイム・スタンプを表す場合、どちらの引数も精度 12 のタイム・スタンプとして効率的に処理され、結果を決定するときにこれらの値の時刻部分も考慮に入れられます。

この関数の結果は DECIMAL(31,15) になります。引数のどちらかが NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数のどちらかが NULL である場合は、結果は NULL 値になります。

例

- 2 つの日付の間の月数を計算します。

```
SELECT MONTHS_BETWEEN('2005-01-17', '2005-02-17')
  FROM SYSIBM.SYSDUMMY1
```

値として -1.000000000000000 が返されます。

```
SELECT MONTHS_BETWEEN('2005-02-20', '2005-01-17')
  FROM SYSIBM.SYSDUMMY1
```

値として 1.096774193548387 が返されます。

- 次の表には別の例が示されています。

表 53. *MONTHS_BETWEEN* の他の使用例

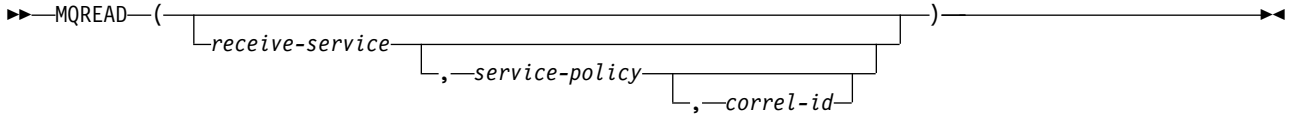
引数 *e1* の値	引数 *e2* の値	**MONTHS_BETWEEN (***e1,e2***)** によって戻される値	**ROUND(MONTHS_BETWEEN(** *e1,e2***)*31,2)** によって戻される値	コメント
2005-02-02	2005-01-01	1.032258064516129	32.00	
2007-11-01-09.00.00.00000	2007-12-07-14.30.12.12345	-1.200945386592741	-37.23	
2007-12-13-09.40.30.00000	2007-11-13-08.40.30.00000	1.000000000000000	31.00	注 1 を参照
2007–03–15	2007–02–20	0.838709677419354	26.00	注 2 を参照
2008-02-29	2008-02-28-12.00.00	0.016129032258064	0.50	
2008-03-29	2008-02-29	1.000000000000000	31.00	
2008-03-30	2008-02-29	1.032258064516129	32.00	
2008-03-31	2008-02-29	1.000000000000000	31.00	注 3 を参照

注**:**

1. 時間差は無視されます。時間差があってもどちらの値も月の日付は同じだからです。
2. 結果は 23 ではありません。2 月は 28 日しかない場合であっても、すべての月が 31 日まであると想定されているためです。
3. 結果は 33 ではありません。どちらの日付もそれぞれの月の最後の日ですので、結果は年と月の部分にのみ基づいて算出されるためです。

MQREAD

MQREAD 関数は、キューからメッセージを除去せずに、指定された MQSeries ロケーション (VARCHAR の戻り値) からメッセージを戻します。



MQREAD 関数は、*service-policy* で定義されたサービス品質ポリシーを使って、*receive-service* で指定された MQSeries ロケーションからメッセージを戻します。この処理を実行しても、*receive-service* で関連付けられたキューからメッセージは除去されず、その代わりにメッセージはキューの先頭に戻されます。

receive-service
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の値は、空ストリング、または末尾ブランクを含むストリングであってはなりません。式の実際の長さは、48 バイトを超えてはなりません。この式の値は、SYSIBM.MQSERVICE 表に定義されたサービス・ポイントを参照している必要があります。サービス・ポイントは、メッセージの送信元または受信相手側の論理エンドポイントです。サービス・ポイントの定義には、MQSeries キュー・マネージャーの名前およびキューの名前が含まれます。 MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

receive-service が指定されていないか NULL 値である場合、Db2.DEFAULT.SERVICE が使用されます。

service-policy
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の値は、空ストリング、または末尾ブランクを含むストリングであってはなりません。式の実際の長さは、48 バイトを超えてはなりません。この式の値は、SYSIBM.MQPOLICY 表に定義されたサービス・ポリシーを参照している必要があります。サービス・ポリシーは、このメッセージ処理に適用されるサービス品質オプションのセットを指定します。これらのオプションには、メッセージ優先順位やメッセージ持続性などがあります。MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

service-policy が指定されていないか NULL 値である場合、Db2.DEFAULT.POLICY が使用されます。

correl-id
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の実際の長さは、24 バイトを超えてはなりません。式の値は、このメッセージに関連付けられている相関 ID を指定します。相関 ID は、しばしば、要求を応答に関連付けるために「要求と応答」シナリオで指定されます。相関 ID が一致した最初のメッセージが戻されます。 MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

末尾ブランクを含む固定長ストリングは、有効な値として扱われます。ただし、*correl-id* を MQSEND など他の要求で指定する場合、一致していると認識されるためには、同じ *correl-id* を指定する必要があります。例えば、前の MQSEND 要求時に「test」(末尾ブランクがある) という *correl-id* 値を指定し、MQRECEIVE の *correl-id* に「test」という値を指定しても一致しません。

correl-id が、指定されていない、空ストリングである、または NULL 値である場合、相関 ID は使用されず、キューの先頭のメッセージが戻されます。

この関数の結果は、長さ属性 32000 の可変長ストリングです。結果が、NULL になることもあります。戻されるメッセージがない場合、結果は NULL 値です。

結果の CCSID は、現行サーバーにおけるデフォルト CCSID です。

注

前提条件**:** MQSeries の機能を使用するためには、IBM MQSeries for IBM i がインストールおよび構成され、作動可能である必要があります。

例

- この例は、デフォルト・サービス (Db2.DEFAULT.SERVICE) によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューの先頭にあるメッセージを読み取ります。

```
SELECT MQREAD ()
 FROM SYSIBM.SYSDUMMY1
```

- この例は、サービス「MYSERVICE」によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューの先頭にあるメッセージを読み取ります。

```
SELECT MQREAD ('MYSERVICE')
 FROM SYSIBM.SYSDUMMY1
```

- この例は、サービス「MYSERVICE」によって指定された、ポリシー「MYPOLICY」を使用するキューの先頭にあるメッセージを読み取ります。

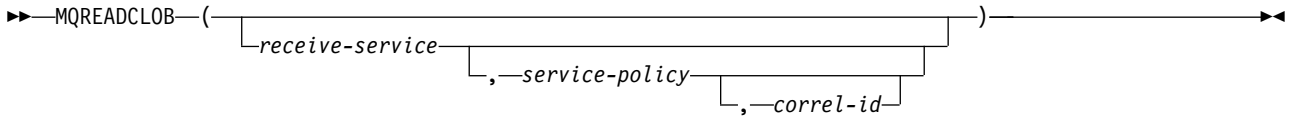
```
SELECT MQREAD ('MYSERVICE','MYPOLICY')
 FROM SYSIBM.SYSDUMMY1
```

- この例は、サービス「MYSERVICE」によって指定された、ポリシー「MYPOLICY」を使用するキューの先頭から、「1234」と一致する相関 ID をもつ最初のメッセージを読み取ります。

```
SELECT MQREAD ('MYSERVICE','MYPOLICY','1234')
 FROM SYSIBM.SYSDUMMY1
```

MQREADCLOB

MQREADCLOB 関数は、キューからメッセージを除去せずに、指定された MQSeries ロケーション (CLOB の戻り値) からメッセージを戻します。



MQREADCLOB 関数は、*service-policy* で定義されたサービス品質ポリシーを使って、*receive-service* で指定された MQSeries ロケーションからメッセージを戻します。この処理を実行しても、*receive-service* で関連付けられたキューからメッセージは除去されず、その代わりにメッセージはキューの先頭に戻されます。

receive-service
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の値は、空ストリング、または末尾ブランクを含むストリングであってはなりません。式の実際の長さは、48 バイトを超えてはなりません。この式の値は、SYSIBM.MQSERVICE 表に定義されたサービス・ポイントを参照している必要があります。サービス・ポイントは、メッセージの送信元または受信相手側の論理エンドポイントです。サービス・ポイントの定義には、MQSeries キュー・マネージャーの名前およびキューの名前が含まれます。 MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

receive-service が指定されていないか NULL 値である場合、Db2.DEFAULT.SERVICE が使用されます。

service-policy
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の値は、空ストリング、または末尾ブランクを含むストリングであってはなりません。式の実際の長さは、48 バイトを超えてはなりません。この式の値は、SYSIBM.MQPOLICY 表に定義されたサービス・ポリシーを参照している必要があります。サービス・ポリシーは、このメッセージ処理に適用されるサービス品質オプションのセットを指定します。これらのオプションには、メッセージ優先順位やメッセージ持続性などがあります。MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

service-policy が指定されていないか NULL 値である場合、Db2.DEFAULT.POLICY が使用されます。

correl-id
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の実際の長さは、24 バイトを超えてはなりません。式の値は、このメッセージに関連付けられている相関 ID を指定します。相関 ID は、しばしば、要求を応答に関連付けるために「要求と応答」シナリオで指定されます。相関 ID が一致した最初のメッセージが戻されます。 MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

末尾ブランクを含む固定長ストリングは、有効な値として扱われます。ただし、*correl-id* を MQSEND など他の要求で指定する場合、一致していると認識されるためには、同じ *correl-id* を指定する必要があります。例えば、前の MQSEND 要求時に「test」(末尾ブランクがある) という *correl-id* 値を指定し、MQRECEIVE の *correl-id* に「test」という値を指定しても一致しません。

correl-id が、指定されていない、空ストリングである、または NULL 値である場合、相関 ID は使用されず、キューの先頭のメッセージが戻されます。

この関数の結果は、長さ属性 2 MB の CLOB です。結果が、NULL になることもあります。戻されるメッセージがない場合、結果は NULL 値です。

結果の CCSID は、現行サーバーにおけるデフォルト CCSID です。

注

前提条件: MQSeries の機能を使用するためには、IBM MQSeries for IBM i がインストールおよび構成され、作動可能である必要があります。

例

- この例は、デフォルト・サービス (Db2.DEFAULT.SERVICE) によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューの先頭にあるメッセージを読み取ります。

```
SELECT MQREADCLOB ()
 FROM SYSIBM.SYSDUMMY1
```

- この例は、サービス「MYSERVICE」によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューの先頭にあるメッセージを読み取ります。

```
SELECT MQREADCLOB ('MYSERVICE')
 FROM SYSIBM.SYSDUMMY1
```

- この例は、サービス「MYSERVICE」によって指定された、ポリシー「MYPOLICY」を使用するキューの先頭にあるメッセージを読み取ります。

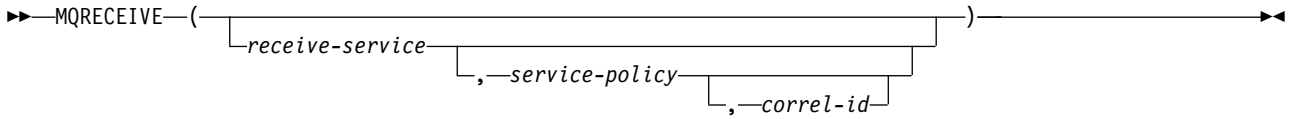
```
SELECT MQREADCLOB ('MYSERVICE','MYPOLICY')
 FROM SYSIBM.SYSDUMMY1
```

- この例は、サービス「MYSERVICE」によって指定された、ポリシー「MYPOLICY」を使用するキューの先頭から、「1234」と一致する相関 ID をもつ最初のメッセージを読み取ります。

```
SELECT MQREADCLOB ('MYSERVICE','MYPOLICY','1234')
 FROM SYSIBM.SYSDUMMY1
```

MQRECEIVE

MQRECEIVE 関数は、指定された MQSeries ロケーション (VARCHAR の戻り値) からメッセージを戻し、キューからメッセージを除去します。



MQRECEIVE 関数は、*service-policy* で定義されたサービス品質ポリシーを使って、*receive-service* で指定された MQSeries ロケーションからメッセージを戻します。この処理を行うと、*receive-service* に関連付けられたキューの先頭からメッセージは除去されます。

receive-service
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の値は、空ストリング、または末尾ブランクを含むストリングであってはなりません。式の実際の長さは、48 バイトを超えてはなりません。この式の値は、SYSIBM.MQSERVICE 表に定義されたサービス・ポイントを参照している必要があります。サービス・ポイントは、メッセージの送信元または受信相手側の論理エンドポイントです。サービス・ポイントの定義には、MQSeries キュー・マネージャーの名前およびキューの名前が含まれます。 MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

receive-service が指定されていないか NULL 値である場合、Db2.DEFAULT.SERVICE が使用されます。

service-policy
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の値は、空ストリング、または末尾ブランクを含むストリングであってはなりません。式の実際の長さは、48 バイトを超えてはなりません。この式の値は、SYSIBM.MQPOLICY 表に定義されたサービス・ポリシーを参照している必要があります。サービス・ポリシーは、このメッセージ処理に適用されるサービス品質オプションのセットを指定します。これらのオプションには、メッセージ優先順位やメッセージ持続性などがあります。MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

service-policy が指定されていないか NULL 値である場合、Db2.DEFAULT.POLICY が使用されます。

correl-id
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の実際の長さは、24 バイトを超えてはなりません。式の値は、このメッセージに関連付けられている相関 ID を指定します。相関 ID は、しばしば、要求を応答に関連付けるために「要求と応答」シナリオで指定されます。相関 ID が一致した最初のメッセージが戻されます。 MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

末尾ブランクを含む固定長ストリングは、有効な値として扱われます。ただし、*correl-id* を MQSEND など他の要求で指定する場合、一致していると認識されるためには、同じ *correl-id* を指定する必要があります。例えば、前の MQSEND 要求時に「test」(末尾ブランクがある) という *correl-id* 値を指定し、MQRECEIVE の *correl-id* に「test」という値を指定しても一致しません。

correl-id が、指定されていない、空ストリングである、または NULL 値である場合、相関 ID は使用されず、キューの先頭のメッセージが戻されます。

この関数の結果は、長さ属性 32000 の可変長ストリングです。結果が、NULL になることもあります。戻されるメッセージがない場合、結果は NULL 値です。

結果の CCSID は、現行サーバーにおけるデフォルト CCSID です。

注

前提条件: MQSeries の機能を使用するためには、IBM MQSeries for IBM i がインストールおよび構成され、作動可能である必要があります。

例

- この例は、デフォルト・サービス (Db2.DEFAULT.SERVICE) によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューの先頭にあるメッセージを取り出します。

```
SELECT MQRECEIVE ()
 FROM SYSIBM.SYSDUMMY1
```

- この例は、サービス「MYSERVICE」によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューの先頭にあるメッセージを取り出します。

```
SELECT MQRECEIVE ('MYSERVICE')
 FROM SYSIBM.SYSDUMMY1
```

- この例は、サービス「MYSERVICE」によって指定された、ポリシー「MYPOLICY」を使用するキューの先頭にあるメッセージを取り出します。

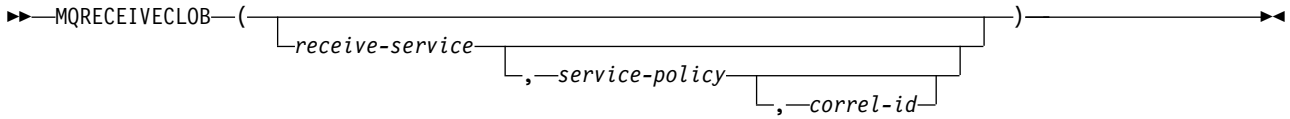
```
SELECT MQRECEIVE ('MYSERVICE','MYPOLICY')
 FROM SYSIBM.SYSDUMMY1
```

- この例は、サービス「MYSERVICE」によって指定された、ポリシー「MYPOLICY」を使用するキューの先頭から、「1234」と一致する相関 ID をもつ最初のメッセージを取り出します。

```
SELECT MQRECEIVE ('MYSERVICE','MYPOLICY','1234')
 FROM SYSIBM.SYSDUMMY1
```

MQRECEIVECLOB

MQRECEIVECLOB 関数は、指定された MQSeries ロケーション (CLOB の戻り値) からメッセージを戻し、キューからメッセージを除去します。



MQRECEIVE 関数は、*service-policy* で定義されたサービス品質ポリシーを使って、*receive-service* で指定された MQSeries ロケーションからメッセージを戻します。この処理を行うと、*receive-service* に関連付けられたキューの先頭からメッセージは除去されます。

receive-service
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の値は、空ストリング、または末尾ブランクを含むストリングであってはなりません。式の実際の長さは、48 バイトを超えてはなりません。この式の値は、SYSIBM.MQSERVICE 表に定義されたサービス・ポイントを参照している必要があります。サービス・ポイントは、メッセージの送信元または受信相手側の論理エンドポイントです。サービス・ポイントの定義には、MQSeries キュー・マネージャーの名前およびキューの名前が含まれます。 MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

receive-service が指定されていないか NULL 値である場合、Db2.DEFAULT.SERVICE が使用されます。

service-policy
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の値は、空ストリング、または末尾ブランクを含むストリングであってはなりません。式の実際の長さは、48 バイトを超えてはなりません。この式の値は、SYSIBM.MQPOLICY 表に定義されたサービス・ポリシーを参照している必要があります。サービス・ポリシーは、このメッセージ処理に適用されるサービス品質オプションのセットを指定します。これらのオプションには、メッセージ優先順位やメッセージ持続性などがあります。MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

service-policy が指定されていないか NULL 値である場合、Db2.DEFAULT.POLICY が使用されます。

correl-id
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の実際の長さは、24 バイトを超えてはなりません。式の値は、このメッセージに関連付けられている相関 ID を指定します。相関 ID は、しばしば、要求を応答に関連付けるために「要求と応答」シナリオで指定されます。相関 ID が一致した最初のメッセージが戻されます。 MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

末尾ブランクを含む固定長ストリングは、有効な値として扱われます。ただし、*correl-id* を MQSEND など他の要求で指定する場合、一致していると認識されるためには、同じ *correl-id* を指定する必要があります。例えば、前の MQSEND 要求時に「test」(末尾ブランクがある) という *correl-id* 値を指定し、MQRECEIVE の *correl-id* に「test」という値を指定しても一致しません。

correl-id が、指定されていない、空ストリングである、または NULL 値である場合、相関 ID は使用されず、キューの先頭のメッセージが戻されます。

この関数の結果は、長さ属性 2 MB の CLOB です。結果が、NULL になることもあります。戻されるメッセージがない場合、結果は NULL 値です。

結果の CCSID は、現行サーバーにおけるデフォルト CCSID です。

注

前提条件: MQSeries の機能を使用するためには、IBM MQSeries for IBM i がインストールおよび構成され、作動可能である必要があります。

例

- この例は、デフォルト・サービス (Db2.DEFAULT.SERVICE) によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューの先頭にあるメッセージを取り出します。

```
SELECT MQRECEIVECLOB ()
 FROM SYSIBM.SYSDUMMY1
```

- この例は、サービス「MYSERVICE」によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューの先頭にあるメッセージを取り出します。

```
SELECT MQRECEIVECLOB ('MYSERVICE')
 FROM SYSIBM.SYSDUMMY1
```

- この例は、サービス「MYSERVICE」によって指定された、ポリシー「MYPOLICY」を使用するキューの先頭にあるメッセージを取り出します。

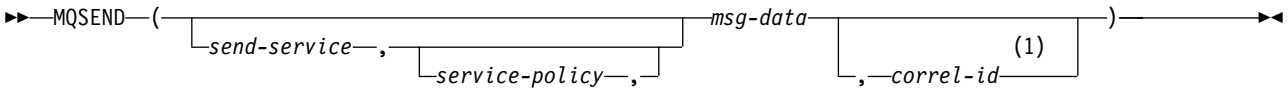
```
SELECT MQRECEIVECLOB ('MYSERVICE','MYPOLICY')
 FROM SYSIBM.SYSDUMMY1
```

- この例は、サービス「MYSERVICE」によって指定された、ポリシー「MYPOLICY」を使用するキューの先頭から、「1234」と一致する相関 ID をもつ最初のメッセージを取り出します。

```
SELECT MQRECEIVECLOB ('MYSERVICE','MYPOLICY','1234')
 FROM SYSIBM.SYSDUMMY1
```

MQSEND

MQSEND 関数は、指定の MQSeries ロケーションにメッセージを送信します。



注:

1 *correl-id* は、*send-service* および *service-policy* も指定されていない場合、指定できません。

MQSEND 関数は *msg-data* に入っているメッセージ・データを、*service-policy* で定義されたサービス品質ポリシーを使って、*send-service* で指定された MQSeries ロケーションに送信します。メッセージは、MQSeries 組み込みフォーマット MQFMT_STRING を使用して送信されます。

send-service
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の値は、空ストリング、または末尾ブランクを含むストリングであってはなりません。式の実際の長さは、48 バイトを超えてはなりません。この式の値は、SYSIBM.MQSERVICE 表に定義されたサービス・ポイントを参照している必要があります。サービス・ポイントは、メッセージの送信元または受信相手側の論理エンドポイントです。サービス・ポイントの定義には、MQSeries キュー・マネージャーの名前およびキューの名前が含まれます。 MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

send-service が指定されていないか NULL 値である場合、Db2.DEFAULT.SERVICE が使用されます。

service-policy
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の値は、空ストリング、または末尾ブランクを含むストリングであってはなりません。式の実際の長さは、48 バイトを超えてはなりません。この式の値は、SYSIBM.MQPOLICY 表に定義されたサービス・ポリシーを参照している必要があります。サービス・ポリシーは、このメッセージ処理に適用されるサービス品質オプションのセットを指定します。これらのオプションには、メッセージ優先順位やメッセージ持続性などがあります。MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

service-policy が指定されていないか NULL 値である場合、Db2.DEFAULT.POLICY が使用されます。

msg-data
組み込み文字ストリング・データ・タイプの値を戻す式。式が CLOB の場合、値の長さは 2 MB を超えてはなりません。式が CLOB でない場合、値の長さは 32000 バイトを超えてはなりません。式の値は、MQSeries を介して送られるメッセージ・データです。NULL 値、空ストリング、および末尾ブランクを含む固定長ストリングはすべて有効な値と見なされます。

correl-id

LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の実際の長さは、24 バイトを超えてはなりません。式の値は、このメッセージに関連付けられている相関 ID を指定します。相関 ID は、しばしば、要求を応答に関連付けるために「要求と応答」シナリオで指定されます。 MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

末尾ブランクを含む固定長ストリングは、有効な値として扱われます。ただし、*correl-id* を MQRECEIVE など他の要求で指定する場合、一致していると認識されるためには、同じ *correl-id* を指定する必要があります。例えば、MQSEND の *correl-id* に「test」という値を指定し、後続の MQRECEIVE 要求時に *correl-id* 値に「test」(末尾ブランクがある) を指定しても一致しません。

correl-id が、指定されていない、空ストリングである、または NULL 値である場合、相関 ID は送信されません。

この関数の結果は、長さ属性 1 の可変長ストリングです。NULL 値が戻されることはありませんが、結果はNULL 可能です。結果は、関数が正常に実行された場合は「1」、関数が正常に実行されなかった場合は「0」です。

結果の CCSID は、現行サーバーのデフォルトの SBCS CCSID になります。

注

前提条件**:** MQSeries の機能を使用するためには、IBM MQSeries for IBM i がインストールおよび構成され、作動可能である必要があります。

例

- この例は、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使って、相関 ID は使わずにストリング「Testing 123」をデフォルト・サービス (Db2.DEFAULT.SERVICE) に送信します。

```
SELECT MQSEND ('Testing 123')
 FROM SYSIBM.SYSDUMMY1
```

- この例は、ポリシー「MYPOLICY」を使って、相関 ID は使わずにストリング「Testing 345」をサービス「MYSERVICE」に送信します。

```
SELECT MQSEND ('MYSERVICE','MYPOLICY','Testing 345')
 FROM SYSIBM.SYSDUMMY1
```

- この例は、ポリシー「MYPOLICY」および相関 ID「TEST3」を使って、ストリング「Testing 678」をサービス「MYSERVICE」に送信します。

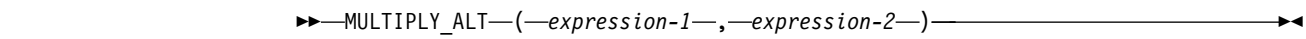
```
SELECT MQSEND ('MYSERVICE','MYPOLICY','Testing 678','TEST3')
 FROM SYSIBM.SYSDUMMY1
```

- この例は、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使って、相関 ID は使わずにストリング「Testing 901」をサービス「MYSERVICE」に送信します。

```
SELECT MQSEND ('MYSERVICE','Testing 901')
 FROM SYSIBM.SYSDUMMY1
```

MULTIPLY_ALT

MULTIPLY_ALT スカラー関数は、2 つの引数の積を 10 進数として戻します。これは、引数の精度の合計が 63 を超える場合は特に、乗算演算子の代わりとして提供されます。



expression-1
任意の組み込み数値データ・タイプ (浮動小数点または 10 進浮動小数点を除く)、文字ストリング、またはグラフィック・ストリング・データ・タイプの値を返す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

expression-2
任意の組み込み数値データ・タイプ (浮動小数点または 10 進浮動小数点を除く)、文字ストリング、またはグラフィック・ストリング・データ・タイプの値を返す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。式 2 はゼロであってはなりません。

この関数の結果は、DECIMAL になります。結果の精度と位取りは、以下のように決定されます。記号 *p* および *s* を使用して最初の引数の精度と位取りを、記号 *p* ' および *s* ' を使用して 2 番目の引数の精度と位取りを指定します。

- 精度は MIN(*mp*, p+p')
- 位取りは以下のとおりです。
  - 引数が両方とも 0 の場合は 0
  - p+p' が *mp* より小か等しい場合は、MIN(*ms*, s+s')
  - p+p' が *mp* より大きい場合は、 MIN(*ms*, MAX(MIN(3, s+s'), *mp*-(p-s+p'-s')))

p、*s*、*ms*、および *mp* の値の説明については、 198 ページの『SQL での 10 進数演算』を参照してください。

引数のどちらかが NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数のどちらかが NULL である場合は、結果は NULL 値になります。

MULTIPLY_ALT 関数は、少なくとも 3 の位取りが必要で、精度の合計が 63 を超えるような 10 進数の演算を実行するときは、乗算演算子より適切な選択といえます。このような場合、内部計算が実行されるため、オーバーフローが回避されます。最終結果は、位取りを合わせるために必要な切り捨てを使用して、結果のタイプ値に割り当てられます。最終結果のオーバーフローは、位取りが 3 のときは依然として起こり得ることに注意してください。

次の表は、最大精度が 31 で最大位取りが 31 の場合の、 MULTIPLY_ALT と乗算演算子を使用した結果タイプを比較しています。

引数 **1** のタイプ	引数 **2** のタイプ	**MULTIPLY_ALT** を使用した結果	乗算演算子を使用した結果
DECIMAL(31,3)	DECIMAL(15,8)	DECIMAL(31,3)	DECIMAL(31,11)
DECIMAL(26,23)	DECIMAL(10,1)	DECIMAL(31,19)	DECIMAL(31,24)
DECIMAL(18,17)	DECIMAL(20,19)	DECIMAL(31,29)	DECIMAL(31,31)
DECIMAL(16,3)	DECIMAL(17,8)	DECIMAL(31,9)	DECIMAL(31,11)
DECIMAL(26,5)	DECIMAL(11,0)	DECIMAL(31,3)	DECIMAL(31,5)
DECIMAL(21,1)	DECIMAL(15,1)	DECIMAL(31,2)	DECIMAL(31,2)

例

- 最初の引数のデータ・タイプが DECIMAL(26,3) で 2 番目の引数のデータ・タイプが DECIMAL(9,8) の場合に、2 つの値を乗算します。この結果のデータ・タイプは 10 進数 (31,7) です。

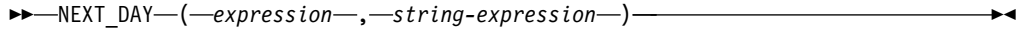
```
SELECT MULTIPLY_ALT(9876543210987654321098 7.654,5.43210987)
  FROM SYSIBM.SYSDUMMY1
```

値として 536504678578875294857887.5277415 が戻されます。

これら 2 つの値の完全な積は 536504678578875294857887.52774154498 ですが、結果データ・タイプの位取りを合わせるために 4 桁切り捨てられていることに注意してください。同じ値で乗算演算子を使用すると算術オーバーフローが生じます。結果データ・タイプが DECIMAL(31,11) で、結果値の小数点の左側は 24 桁になりますが、結果データ・タイプは 20 桁しかサポートしないからです。

NEXT_DAY

NEXT_DAY 関数は、*expression* の日付より後の、*string-expression* で指定された曜日の最初の日付またはタイム・スタンプ値を戻します。



expression
日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式。

expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

string-expression
組み込み文字ストリング・データ・タイプか、グラフィック・ストリング・データ・タイプを戻す式。値は曜日のフルネームと等しいか、または曜日の省略形と等しくなければなりません。例えば、英語の場合、以下のとおりです。

曜日	省略形
MONDAY	MON
TUESDAY	TUE
WEDNESDAY	WED
THURSDAY	THU
FRIDAY	FRI
SATURDAY	SAT
SUNDAY	SUN

入力値の最短の長さは、省略形の長さです。先行ブランクと末尾ブランクはストリング式 から削除されます。その結果の値は大文字変換されるため、値の中の文字は大文字、小文字のどちらでも構いません。

関数の結果は、*expression* がストリングである場合を除き、*expression* と同じデータ・タイプになります。ストリングである場合は、TIMESTAMP(6) になります。引数のどちらかが NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数のどちらかが NULL である場合は、結果は NULL 値になります。

expression に含まれる情報は、時間、分、秒または 1 秒未満の値にいたるまで、関数によって変更されることはありません。 *expression* が日付を表すストリングである場合、結果として戻る TIMESTAMP 値の時刻情報には、すべてゼロが設定されています。

注記

各国語の考慮事項: ストリング式 内の曜日 (または省略形) の値は、上記の表にリストした米国英語の値でも、ジョブのメッセージに使用される言語に基づいた値のいずれでもかまいません。省略されていない曜日の名前は、ライブラリー *LIBL の中のメッセージ・ファイル QCPFMSG のメッセージ CPX9034 から検索されま

す。省略された曜日の名前は、ライブラリー *LIBL の中のメッセージ・ファイル QCPFMSG のメッセージ CPX9039 から検索されます。

米国英語の値は常に NEXT_DAY 関数に受け入れられるため、多くの異なる言語処理環境で稼働する必要のあるアプリケーションの場合は、米国英語の値の使用を考慮することをお勧めします。

例

- ジョブのデフォルト言語が米国英語であると想定し、ホスト変数 NEXTDAY を 2000 年 4 月 24 日の次の火曜日の日付に設定します。

```
SET :NEXTDAY = NEXT_DAY(CURRENT_DATE, 'TUESDAY')
```

  ホスト変数 NEXTDAY には、値「2000–04–25–00.00.00.000000」が設定されます。ここで、CURRENT_DATE 特殊レジスターの値は「2000–04–24」とします。

- ジョブのデフォルト言語が米国英語であると想定し、ホスト変数 NEXTDAY を 2000 年 5 月 の最初の月曜日の日付に設定します。ホスト変数 DAYHV = 'MON' であると想定します。

```
SET :NEXTDAY = NEXT_DAY(LAST_DAY(CURRENT_TIMESTAMP), :DAYHV)
```

  ホスト変数 NEXTDAY は、CURRENT_TIMESTAMP 特殊レジスターの値が '2000-04-24-12.01.01.123456' であるという想定では '2000-05-01-12.01.01.123456' の値に設定されます。

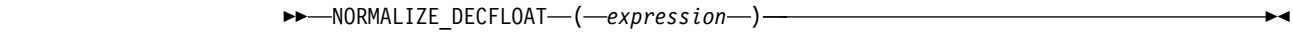
- ジョブのデフォルト言語が米国英語であると想定し、以下を設定します。

```
SELECT NEXT_DAY('2000-04-24', 'TUESDAY')
FROM SYSIBM.SYSDUMMY1
```

  '2000-04-24' の次の火曜日である '2000-04-25-00.00.00.000000' が戻されます。

NORMALIZE_DECFLOAT

NORMALIZE_DECFLOAT 関数は、最も単純な形式に設定された入力引数と等しい DECFLOAT 値を返します。



NORMALIZE_DECFLOAT 関数は、最も単純な形式に設定された入力引数と等しい 10 進浮動小数点値を返します。つまり、係数内で後続ゼロを持つゼロ以外の数値では、それらのゼロは除去されます。この場合、係数を適切な 10 のべき乗で除算し、指数を適宜調整することにより、数値を正規形で表す必要があります。ゼロの値ではその指数は 0 に設定されます。

expression
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数のデータ・タイプが SMALLINT、INTEGER、REAL、DOUBLE、DECIMAL(p,s) (p <=16)、または NUMERIC(p,s) (p <=16) の場合、引数は処理を行うために DECFLOAT(16) に変換されます。これ以外の場合、引数は処理を行うために DECFLOAT(34) に変換されます。

引数が特殊値の場合は、算術演算の一般的な規則が適用されます。詳しくは、 200 ページの『DECFLOAT の一般的な算術演算規則』を参照してください。

この関数の結果は、10 進浮動小数点への変換後の *expression* のデータ・タイプが DECFLOAT(16) の値である場合は DECFLOAT(16) になります。そうでない場合、この関数の結果は DECFLOAT(34) の値になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

以下の例に、さまざまな DECFLOAT 値で NORMALIZE_DECFLOAT 関数を使用した結果を示します。

```
NORMALIZE_DECFLOAT(DECFLOAT(2.1))          =   2.1
NORMALIZE_DECFLOAT(DECFLOAT(-2.0))         =  -2
NORMALIZE_DECFLOAT(DECFLOAT(1.200))        =   1.2
NORMALIZE_DECFLOAT(DECFLOAT(-120))         =  -1.2E+2
NORMALIZE_DECFLOAT(DECFLOAT(120.00))       =   1.2E+2
NORMALIZE_DECFLOAT(DECFLOAT(0.00))         =   0
NORMALIZE_DECFLOAT(-NAN)                   =   -NAN
NORMALIZE_DECFLOAT(-INFINITY)              =   -INFINITY
```

NOW

NOW 関数は、SQL ステートメントが現行サーバーで実行される時点の刻時機構の読み取りに基づくタイム・スタンプを戻します。

integer

タイム・スタンプの精度を指定します。有効な値は 0 から 12 までです。値が指定されていない場合は、デフォルトの精度として 6 が使用されます。

NOW 関数によって戻される値は、CURRENT_TIMESTAMP 特殊レジスターによって戻される値と同じです。この関数が 1 つの SQL ステートメント内で複数回使用される場合、または 1 つのステートメント内で CURDATE または CURTIME スカラー関数、あるいは CURRENT_DATE、CURRENT_TIME、または CURRENT_TIMESTAMP 特殊レジスターとともに使用される場合は、値はすべて 1 回の刻時機構読み取りに基づきます。

結果のデータ・タイプは、*integer* で指定された精度のタイム・スタンプです。結果が NULL になることはありません。

注記

代替構文: CURRENT_TIMESTAMP 特殊レジスターを使用して移植性を最大限に引き出す必要があります。詳しくは、 149 ページの『特殊レジスター』を参照してください。

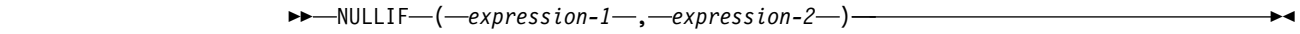
例

- 刻時機構に基づく現在のタイム・スタンプが戻されます。

```
SELECT NOW()
  FROM SYSIBM.SYSDUMMY1
```

NULLIF

NULLIF 関数は、引数が等しい場合に NULL を戻します。等しくない場合は、最初の引数の値を戻します。



それぞれの引数のデータ・タイプには、互換性がなければなりません。

expression-1
DATALINK または XML 以外の任意の組み込みデータ・タイプ、あるいは DATALINK または XML に基づくもの以外の任意の特殊データ・タイプの値を戻す式。

expression-2
DATALINK または XML 以外の任意の組み込みデータ・タイプ、あるいは DATALINK または XML に基づくもの以外の任意の特殊データ・タイプの値を戻す式。

結果の属性は最初の引数の属性です。結果が、NULL になることもあります。最初の引数が NULL か、または両方の引数が等しい場合は、結果は NULL になります。

NULLIF(e1,e2) を使用した結果は、次の式を使用した結果と同じです。

```
CASE WHEN e1=e2 THEN NULL ELSE e1 END
```

e1=e2 が未知であると評価された (引数の片方または両方が NULL であったため) 場合は、CASE 式はこれを真ではないと考えます。したがって、この場合は、NULLIF は第 1 オペランドの e1 を戻します。

例

- ホスト変数 PROFIT、CASH および LOSSES は DECIMAL データ・タイプで、値がそれぞれ 4500.00、500.00 および 5000.00 であると想定します。

  ```
  SELECT NULLIF (:PROFIT + :CASH, :LOSSES )
    FROM SYSIBM.SYSDUMMY1
  ```

  結果として NULL 値が戻ります。

OCTET_LENGTH

OCTET_LENGTH 関数は、ストリング式の長さをオクテット数 (バイト数) で戻します。

類似の関数については、 522 ページの『LENGTH』および 382 ページの『CHARACTER_LENGTH』を参照してください。

expression
任意の組み込み数値データ・タイプ、またはストリング・データ・タイプの値を戻す式。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。

この関数の結果は DECIMAL(31) になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

結果はその引数のオクテット数 (バイト数) です。ストリングの長さには、末尾ブランクも含まれます。可変長ストリングを指定した場合に戻る長さは、オクテット数 (バイト数) で表した実際の長さであり、最大長ではありません。

例

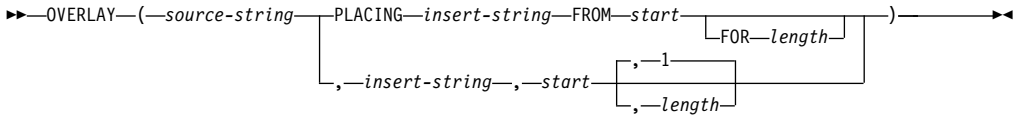
- 表 T1 に C1 という名前の GRAPHIC(10) 列があると想定します。

```
SELECT OCTET_LENGTH( C1 )
  FROM T1
```

値として 20 が戻されます。

OVERLAY

ソース・ストリング の開始桁 から長さ 文字を削除し、ソース・ストリング の開始桁 の位置に挿入ストリング を挿入したストリングを戻します。



source-string

ソース・ストリングを指定する式。 *source-string* には、任意の組み込み数値、ストリング、または日時式を指定できます。これは、挿入ストリング と互換性のあるものでなければなりません。データ・タイプの互換性についての詳細は、 113 ページの『割り当ておよび比較』を参照してください。数値引数または日時引数は、関数を評価する前に、現行サーバーでデフォルト SBCS CCSID である CCSID を使用して VARCHAR にキャストされます。数値または日時からさまざまな文字ストリングへの変換について詳しくは、 672 ページの『VARCHAR』 を参照してください。ストリングの実際の長さはゼロより大きくなくてはなりません。

OVERLAY 関数は、引数の順序が異なり、*length* がオプションである点を除けば、INSERT 関数と同一です。

insert-string

ソース・ストリング に挿入する、開始桁 で示される位置から開始するストリングを指定する式。 *insert-string* には、任意の組み込み数値、ストリング、または日時式を指定できます。これは、ソース・ストリング と互換性のあるものでなければなりません。データ・タイプの互換性についての詳細は、 113 ページの『割り当ておよび比較』を参照してください。数値引数または日時引数は、関数を評価する前に、現行サーバーでデフォルト SBCS CCSID である CCSID を使用して VARCHAR にキャストされます。数値または日時からさまざまな文字ストリングへの変換について詳しくは、 672 ページの『VARCHAR』 を参照してください。

start

任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。値がタイプ INTEGER でない場合、その値は関数を評価する前に暗黙的に INTEGER にキャストされます。整数は、文字の削除と別のストリングの挿入を開始するソース・ストリング 内の開始文字を指定します。整数の値は、1 からソース・ストリング の長さに 1 を加えた数の範囲でなければなりません。

length

任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。値がタイプ INTEGER でない場合、その値は関数を評価する前に暗黙的に INTEGER にキャストされます。この整数は、ソース・ストリング から削除される、開始桁 で示される文字位置から始まる文字数を指定します。整数の値は、0 からソース・ストリング の長さまでの範囲でなければなりません。

関数の結果のデータ・タイプは、1 番目と 2 番目の引数のデータ・タイプによって決まります。結果のデータ・タイプは、結果は常に可変長ストリングであることを除けば、 2 つの引数を連結した場合と同じです。詳しくは、 139 ページの『ストリングを結合する演算に適用される変換規則』を参照してください。

結果の長さ属性は、引数によって次のように異なります。

- 開始桁 と長さ が定数の場合、結果の長さ属性は次のとおりです。

  ```
  L1 - MIN((L1-V2 + 1), V3) + L4
  ```

  ここで、

  ```
  L1 はソース・ストリングの長さ属性
  V2 は次のようにソース・ストリングのエンコード・スキーマに依存します。
  - ソース・ストリングが UTF-8 の場合は値 MIN(L1+1,start*3)
  - ソース・ストリングが混合データの場合は値 MIN(L1+1,(start-1)*2.5+4)
  上記以外の場合は値 start
  V3 は長さの値
  L4 は挿入ストリングの長さ属性
  ```
- それ以外の場合は、結果の長さ属性は、ソース・ストリング の長さ属性と挿入ストリング の長さ属性を足したものになります。

結果の長さ属性が結果のデータ・タイプの最大長を超える場合は、エラーが戻されます。

結果の実際の長さは、次のとおりです。

```
A1 - MIN((A1 -V2 + 1), V3) + A4
```

各値は、次のとおりです。

```
A1 はソース・ストリングの実際の長さ
V2 は開始桁の値
V3 は長さの値
A4 は挿入ストリングの実際の長さ
```

結果ストリングの実際の長さが結果のデータ・タイプの最大長を超える場合は、エラーが戻されます。

引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

結果の CCSID は、ソース・ストリング と挿入ストリング の CCSID によって決定されます。結果 CCSID は、2 つの引数を連結した場合と同じです。詳しくは、139 ページの『ストリングを結合する演算に適用される変換規則』を参照してください。

例

- 次の例は、ストリング「INSERTING」をどのように他のストリングに変更できるかを示しています。 CHAR 関数を使用すると、結果ストリングの長さが 10 文字に制限されます。

```
SELECT OVERLAY('INSERTING', 'IS', 4, 2),
       OVERLAY('INSERTING', 'IS', 4, 0),
       OVERLAY('INSERTING', '', 4, 2)
  FROM SYSIBM.SYSDUMMY1
```

この例は、「INSISTING」、「INSISERTING」、および「INSTING」を戻します。

- 前の例では、あるテキストの中にテキストを挿入する方法を示しました。この例は、開始点 (*start*) として 1 を使用して、テキストの前に他のテキストを挿入する方法を示しています。

```
SELECT OVERLAY('INSERTING', 'XX', 1, 0),
       OVERLAY('INSERTING', 'XX', 1, 1),
       OVERLAY('INSERTING', 'XX', 1, 2),
       OVERLAY('INSERTING', 'XX', 1, 3)
  FROM SYSIBM.SYSDUMMY1
```

この例は、「XXINSERTING」、「XXNSERTING」、「XXSERTING」、および「XXERTING」を戻します。

- 次の例は、あるテキストの後ろにテキストを挿入する方法を示しています。ストリング「ABCABC」の末尾に「XX」を追加します。ソース・ストリングの長さは 6 文字なので、開始位置を 7 (1 + ソース・ストリングの長さ) に設定します。

```
SELECT OVERLAY('ABCABC', 'XX', 7, 0)
  FROM SYSIBM.SYSDUMMY1
```

この例は、「ABCABCXX 」を戻します。

- 以下の例では、ストリング 'Hegelstraße' を 'Hegelstrasse' に変更します。

```
SELECT OVERLAY('Hegelstraße', 'ss', 10, 1)
  FROM SYSIBM.SYSDUMMY1
```

この例は「Hegelstrasse」を戻します。

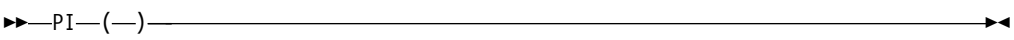
- 変数 UTF8_VAR は UTF8 として定義され、UTF16_VAR は UTF16 として定義されているとします。この両方に Unicode ストリング '&N~AB' が含まれているとします。ここで、'&' は音符のト音記号、'~' は結合チルド文字です。

```
SELECT OVERLAY(UTF8_VAR, 'C', 1),
       OVERLAY(UTF8_VAR, 'C', 5),
       OVERLAY(UTF16_VAR, 'C', 1),
       OVERLAY(UTF16_VAR, 'C', 5)
  FROM SYSIBM.SYSDUMMY1
```

この例は、'CN~AB'、'&N~AC'、'CN~AB'、および '&N~AC' を戻します。

PI

π の値として 3.141592653589793 が戻されます。引数はありません。



この関数の結果は、倍精度の浮動小数点になります。結果が NULL になることはありません。

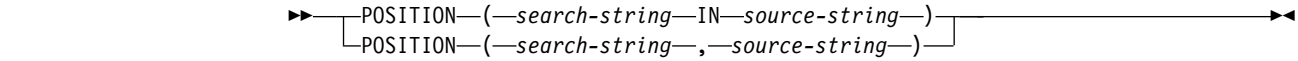
例

- 次の関数は、直径 10 の円の円周を戻します。

```
SELECT PI()*10
  FROM SYSIBM.SYSDUMMY1
```

POSITION

POSITION 関数は、あるストリング (検索ストリング と呼ばれる) の、別のストリング (ソース・ストリング と呼ばれる) の中での、最初の出現箇所の開始位置を戻します。検索ストリング が見つからず、どちらの引数も NULL でない場合、結果はゼロになります。検索ストリング が見つかった場合、結果は 1 から ソース・ストリング の実際の長さまでの数値になります。



関連関数については、 526 ページの『LOCATE』および 577 ページの『POSSTR』を参照してください。

search-string
検索するオブジェクトのストリングを指定する式。検索ストリング には、任意の組み込み数値、日時、またはストリング式を指定できます。これは、ソース・ストリング と互換性のあるものでなければなりません。数値または日時引数は、関数を評価する前に文字ストリングにキャストされます。数値および日時から文字ストリングへの変換について詳しくは、 672 ページの『VARCHAR』を参照してください。

source-string
検索を行う相手先のソース・ストリングを指定する式。ソース・ストリング には、任意の組み込み数値、日時、またはストリング式を指定できます。数値または日時引数は、関数を評価する前に文字ストリングにキャストされます。数値および日時から文字ストリングへの変換について詳しくは、 672 ページの『VARCHAR』 を参照してください。

この関数の結果は長精度整数になります。どちらかの引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。どちらかの引数が NULL の場合は、結果は NULL 値になります。

POSITION 関数は文字単位で実行します。POSITION は文字ストリング単位で実行されるため、シフトイン、シフトアウト文字がまったく同じ場所にある必要がなく、これらの文字は、どの文字が SBCS でどの文字が DBCS であるかを示すためにだけ意味があります。

検索ストリング の CCSID がソース・ストリング の CCSID と異なる場合は、ソース・ストリング の CCSID に変換されます。ソース・ストリング の CCSID が混合データまたは UTF-8 の場合、CCSID は UTF-16 に変換されます。

POSITION 関数を含むステートメントの実行時に *HEX 以外の照合順序が有効で、しかも引数が SBCS データ、混合データ、または Unicode データの場合、結果は、集合の各値の重み付けされた値の比較によって求められます。値の重み付けは、該当の照合順序に基づいています。 ICU 照合順序表は、POSITION 関数では指定できません。

検索ストリング の長さがゼロの場合は、この関数が戻す結果は 1 です。その他の場合は、次のようになります。

- ソース・ストリング の長さがゼロの場合は、この関数が戻す結果は 0 です。

- その他の場合は、
  - 検索ストリング の値がソース・ストリング の値の範囲内の連続位置のサブストリングの長さに等しければ、この関数の戻す結果は、ソース・ストリング 値内のそのような最初のサブストリングの開始位置です。
  - それ以外の場合は、この関数の戻す結果は 0 です。[65]

例

- IN_TRAY 表の全項目から、RECEIVED 列と SUBJECT 列、それに NOTE_TEXT 列の語「GOOD」の開始位置を選択します。

```
SELECT RECEIVED, SUBJECT, POSITION('GOOD', NOTE_TEXT)
  FROM IN_TRAY
  WHERE POSITION('GOOD', NOTE_TEXT) <> 0
```

- NOTE は VARCHAR(128) の列で Unicode UTF-8 でエンコードされ、値 'Jürgen lives on Hegelstraße' を含むものと想定します。ストリング内で文字「ß」の文字位置を検索します。

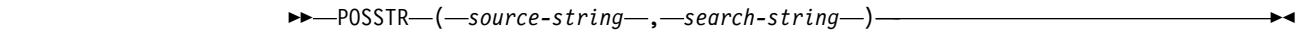
```
SELECT POSITION('ß', NOTE), POSSTR( NOTE, 'ß')
  FROM T1
```

  POSITION の場合は値 26、POSSTR の場合には値 27 を戻します。

65. この中には、検索ストリング の方がソース・ストリング よりも長い場合が含まれます。

POSSTR

POSSTR 関数は、あるストリング (*search-string* と呼ばれる) の、別のストリング (*source-string* と呼ばれる) の中での、最初の出現箇所の開始位置を戻します。検索ストリング が見つからず、どちらの引数も NULL でない場合、結果はゼロになります。検索ストリング が見つかった場合、結果は 1 から ソース・ストリング の実際の長さまでの数値になります。



関連関数については、 526 ページの『LOCATE』および 575 ページの『POSITION』を参照してください。

source-string
検索を行う相手先のソース・ストリングを指定する式。ソース・ストリング には、任意の組み込み数値、日時、またはストリング式を指定できます。数値または日時引数は、関数を評価する前に文字ストリングにキャストされます。数値および日時から文字ストリングへの変換について詳しくは、 672 ページの『VARCHAR』 を参照してください。

search-string
検索するオブジェクトのストリングを指定する式。検索ストリング には、任意の組み込み数値、日時、またはストリング式を指定できます。これは、ソース・ストリング と互換性のあるものでなければなりません。数値または日時引数は、関数を評価する前に文字ストリングにキャストされます。数値および日時から文字ストリングへの変換について詳しくは、 672 ページの『VARCHAR』を参照してください。

この関数の結果は長精度整数になります。どちらかの引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。どちらかの引数が NULL の場合は、結果は NULL 値になります。

POSSTR 関数は、MIXED データ・ストリングを受け入れます。ただし、 POSSTR は、1 バイト文字であるか 2 バイト文字に関係なく厳密にバイト・カウント単位で実行されます。[66] 検索ストリング かソース・ストリング の一方が混合データを含んでいる場合は、POSSTR ではなく POSITION を使用してください。 POSITION 関数は文字単位で実行します。EBCDIC エンコード・スキームでは、シフトイン、シフトアウト文字がまったく同じ場所にある必要がなく、これらの文字は、どの文字が SBCS でどの文字が DBCS であるかを示すためにだけ意味があります。

検索ストリング の CCSID がソース・ストリング の CCSID と異なる場合は、ソース・ストリング の CCSID に変換されます。ソース・ストリング の CCSID が混合データまたは UTF-8 の場合、CCSID は UTF-16 に変換されます。

POSSTR 関数を含むステートメントの実行時に *HEX 以外の照合順序が有効で、しかも引数が SBCS データ、混合データ、または Unicode データの場合、結果は、

66. 例えば、EBCDIC エンコード・スキームで *source-string* に混合データが含まれていると、*search-string* が検出されるのは、シフトイン文字とシフトアウト文字も *source-string* 内の正確に同じ位置で検出される場合だけです。

その集合の各値の重み付けされた値の比較によって求められます。値の重み付けは、該当の照合順序に基づいています。 ICU 照合順序表は、POSSTR 関数では指定できません。

検索ストリング の長さがゼロの場合は、この関数が戻す結果は 1 です。その他の場合は、次のようになります。

- ソース・ストリング の長さがゼロの場合は、この関数が戻す結果は 0 です。
- その他の場合は、
  - 検索ストリング の値がソース・ストリング の値の範囲内の連続位置のサブストリングの長さに等しければ、この関数の戻す結果は、ソース・ストリング 値内のそのような最初のサブストリングの開始位置です。
  - それ以外の場合は、この関数の戻す結果は 0 です。[67]

例

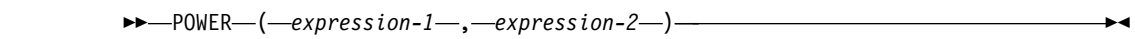
- IN_TRAY 表の全項目から、RECEIVED 列と SUBJECT 列、それに NOTE_TEXT 列の語「GOOD」の開始位置を選択します。

```
SELECT RECEIVED, SUBJECT, POSSTR(NOTE_TEXT, 'GOOD')
  FROM IN_TRAY
  WHERE POSSTR(NOTE_TEXT, 'GOOD') <> 0
```

67. この中には、検索ストリング の方がソース・ストリング よりも長い場合が含まれます。

POWER

POWER 関数は、最初の引数を 2 番目の引数で累乗した値を戻します。



expression-1
任意の組み込み数値、文字ストリング、またはグラフィック・ストリング・データ・タイプの値を返す式。[68] ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

expression-2
任意の組み込み数値データ・タイプの値を戻す式。式 *1* の値がゼロである場合は、式 *2* はゼロまたはそれより大きい値でなければなりません。式 *1* の値がゼロより小さい場合は、式 *2* は整数値でなければなりません。

引数のデータ・タイプが 10 進浮動小数点の場合、結果のデータ・タイプは DECFLOAT(34) です。そうでない場合、この関数の結果は、倍精度浮動小数点数になります。引数が両方とも 0 の場合は、結果は 1 です。引数が NULL の可能性があれば、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

注記

DECFLOAT 特殊値の関係する場合の結果: いずれかの引数が 10 進浮動小数点の場合、両方の引数が DECFLOAT(34) に変換されます。10 進浮動小数点値の場合、特殊値は次のように扱われます。

- いずれかの引数が NaN または -NaN の場合、NaN が返されます。[69]
- POWER®(Infinity, 有効な任意の第 2 引数) は、Infinity を返します。
- POWER(-Infinity, 有効な任意の奇数整数値) は、-Infinity を返します。
- POWER(-Infinity, 有効な任意の偶数整数値) は、Infinity を返します。
- POWER(0,Infinity) は 0 を返します。
- POWER(1,Infinity) は 1 を返します。
- POWER(1,Infinityより大きい任意の数) は、Infinity を返します。
- POWER(0 より大きく、1,Infinity より小さい任意の数) は、0 を返します。
- POWER(0,Infinityより小さい任意の数) は、NaN を返します。[69]
- いずれかの引数が sNaN または -sNaN の場合、警告またはエラーが返されます。[69]

例

- ホスト変数 HPOWER は、値が 3 の整数であると想定します。

```
SELECT POWER(2,:HPOWER)
  FROM SYSIBM.SYSDUMMY1
```

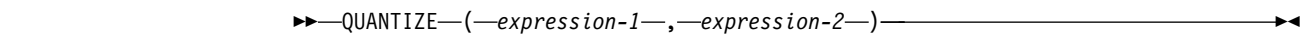
68. POWER 関数の結果は、指数 *expression-1* ** *expression-2* の結果とまったく同じです。

69. SQL_DECFLOAT_WARNINGS 照会オプションに *YES を指定すると、NaN が返され、警告が出されます。

値として 8 が戻されます。

QUANTIZE

QUANTIZE 関数は、値 (丸めが行われる場合は除く) および符号が *expression-1* と等しく、また、指数セットが *expression-2* 内の指数と等しい 10 進浮動小数点値を返します。



expression-1
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数のデータ・タイプが DECFLOAT 値でない場合、そのデータ・タイプは処理を行うために DECFLOAT(34) に変換されます。

expression-2
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数のデータ・タイプが DECFLOAT 値でない場合、そのデータ・タイプは処理を行うために DECFLOAT(34) に変換されます。

一方の引数 (変換後) が DECFLOAT(16) で、もう一方の引数が DECFLOAT(34) の場合、この関数の処理が行われる前に DECFLOAT(16) の引数が DECFLOAT(34) に変換されます。

結果の係数は、*expression-1* の係数から派生したものです。この係数は、必要に応じて、丸められたり (指数が増えている場合)、10 の累乗で乗算されたり (指数が減っている場合)、あるいは未変更のままにされます (既に指数が *expression-2* の指数に等しい場合)。

必要であれば、QUANTIZE 関数で丸めモードが使用されます。詳しくは、 154 ページの『CURRENT DECFLOAT ROUNDING MODE』を参照してください。

DECFLOAT データ・タイプでの他の算術演算とは異なり、量子化演算後の係数の長さが結果の DECFLOAT 数値の精度より長い場合は、エラーが発生します。これにより、エラーが発生しない限り、QUANTIZE 関数の結果の指数は必ず *expression-2* の指数と等しくなります。

この関数の結果は、両方の引数が DECFLOAT(16) の場合は DECFLOAT(16) の値になります。そうでない場合、この関数の結果は DECFLOAT(34) の値になります。引数のどちらかが NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数のどちらかが NULL である場合は、結果は NULL 値になります。

注記

DECFLOAT 特殊値が関係する場合の結果: 10 進浮動小数点の特殊値は次のように扱われます。

- いずれかの引数が NaN であり、第 1 引数が -NaN でない場合、 NaN が返されます。

70. SQL_DECFLOAT_WARNINGS 照会オプションに *YES を指定すると、NaN が返され、警告が出されます。

- いずれかの引数が sNaN の場合、警告またはエラーが発生します。[70]
- いずれかの引数が -NaN であり、第 1 引数が NaN でない場合、 -NaN が返されます。
- いずれかの引数が -sNaN の場合、警告またはエラーが発生します。[70]
- 両方の引数が Infinity (正または負) の場合、Infinity (正または負) が返されます。
- 一方の引数が Infinity (正または負) で、もう一方の引数が Infinity (正または負) ではない場合、NaN が返されます。[70]

例

以下の例では、以下の入力 DECFLOAT 値の場合に、QUANTIZE 関数で返される値を示しています。

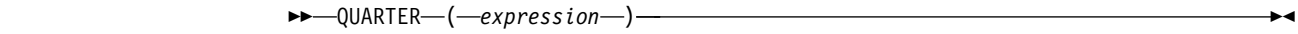
```
QUANTIZE(2.17, 0.001)           ==> 2.170
QUANTIZE(2.17, 0.01)            ==> 2.17
QUANTIZE(2.17, 0.1)             ==> 2.2
QUANTIZE(2.17, 1e+0)            ==> 2
QUANTIZE(2.17, 1e+1)            ==> 0E+1
QUANTIZE(2, Infinity)           ==> NaN (exception)
QUANTIZE(0, 1e+5)               ==> 0E+5
QUANTIZE(217, 1e-1)             ==> 217.0
QUANTIZE(217, 1e+0)             ==> 217
QUANTIZE(217, 1e+1)             ==> 2.2E+2
QUANTIZE(217, 1e+2)             ==> 2E+2
```

次の例では、QUANTIZE 関数に対して値 -0 が戻ります。CHAR 関数を使用すると、クライアント・プログラムによって負符号 (-) が除去されないようにできます。

```
CHAR(QUANTIZE(-0.1, 1))         ==> -0
```

QUARTER

QUARTER 関数は、日付が存在する四半期を表す 1 から 4 までの整数を戻します。例えば、1 月、2 月、3 月の日付は、いずれも整数 1 を戻します。



expression
日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式。

expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

この関数の結果は長精度整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

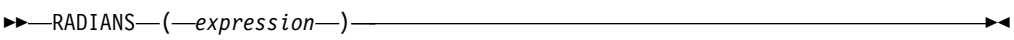
- 表 PROJECT を使用して、ホスト変数 QUART (INTEGER) をプロジェクト 'PL2100' が終了した四半期 (PRENDATE) にセットします。

```
SELECT QUARTER(PRENDATE)
  INTO :QUART
  FROM PROJECT
  WHERE PROJNO = 'PL2100'
```

結果として、QUART は 3 に設定されます。

RADIANS

RADIANS 関数は、度で表された引数に対してラジアン数を戻します。



expression
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点にキャストされます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

引数のデータ・タイプが DECFLOAT(*n*) の場合、結果は DECFLOAT(*n*) です。それ以外の場合、結果のデータ・タイプは、倍精度の浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

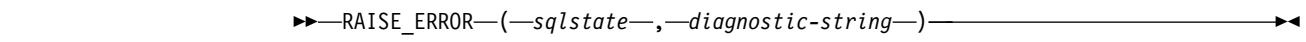
- ホスト変数 HDEG は、値が 180 の整数であると想定します。次のステートメントは、

```
SELECT RADIANS(:HDEG)
  FROM SYSIBM.SYSDUMMY1
```

  概略値 3.1415926536 の倍精度の浮動小数点数を戻します。

RAISE_ERROR

RAISE_ERROR 関数は、この関数を呼び出すステートメントが、指定された SQLSTATE (SQLCODE -438 と共に) および診断ストリングを付けてエラーを返すようにします。



sqlstate
組み込み CHAR データ・タイプまたは VARCHAR データ・タイプの値を、正確に 5 文字で戻す式。*sqlstate* 値は、アプリケーションが定義した SQLSTATE に関する以下の規則に従う必要があります。

- 各文字は、数字 ('0' から '9') またはアクセント記号なしの英大文字 ('A' から 'Z') でなければなりません。
- SQLSTATE クラス (最初の 2 文字) は、'00'、'01'、または '02' (これらはエラー・クラスではないため) であってはなりません。

SQLSTATE がこの規則に従っていないと、エラーが戻されます。

diagnostic-string
エラー状態について説明した、組み込み CHAR データ・タイプまたは VARCHAR データ・タイプで、長さが最大で 1000 バイトまでの値を戻す式。このストリングが 1000 バイトよりも長い場合は、切り捨てられます。

SQLCA を使用する場合、以下のアクションが実行されます。

- このストリングは、SQLCA の SQLERRMC フィールドに戻されます。
- ストリングの実際の長さが 70 バイトを超える場合、警告せずに切り捨てられます。

RAISE_ERROR の結果のデータ・タイプが未定義であるため、これはパラメーター・マーカーが許可される場合にのみ使用できます。パラメーター・マーカーが許可されないコンテキスト (選択リストで単独で使用など) でこの関数を使用するには、キャスト指定を使用して、戻される NULL 値にデータ・タイプを与えなければなりません。

RAISE_ERROR 関数は、未定義のデータ・タイプについては常に NULL を戻します。

例

- 従業員番号と学歴のリストを、学歴を Post Graduate、Graduate、および Diploma として示します。学歴が 20 を超える場合は、エラーになります。

```
SELECT EMPNO,
   CASE WHEN EDLEVEL < 16 THEN 'Diploma'
        WHEN EDLEVEL < 18 THEN 'Graduate'
        WHEN EDLEVEL < 21 THEN 'Post Graduate'
        ELSE RAISE_ERROR( '07001',
                  'EDLEVEL has a value greater than 20' )
   END
 FROM EMPLOYEE
```

RAND

RAND 関数は、0 以上で 1 以下の浮動小数点値を戻します。

expression
式が指定されている場合、その式がシード値として使用されます。引数は、組み込み短整数、長精度整数、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式でなければなりません。ストリング引数は、関数を評価する前に整数にキャストされます。ストリングを整数に変換する方法については、 496 ページの『INTEGER または INT』を参照してください。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

特定のシード値は、照会が実行されるごとに、その照会の RAND 関数の特定のインスタンスに関して、乱数の同じシーケンスを生成します。シード値が指定されない場合、照会が実行されるごとに乱数の異なるシーケンスが生成されます。

シード値が使用されるのは、ステートメントで RAND 関数のインスタンスが最初に呼び出される場合だけです。

RAND は非決定性関数です。

例

- ホスト変数 HRAND は、値が 100 の整数であると想定します。次のステートメントは、

```
SELECT RAND(:HRAND)
  FROM SYSIBM.SYSDUMMY1
```

  0 と 1 の間の乱数の浮動小数点数 (概略値 .0121398 など) を戻します。

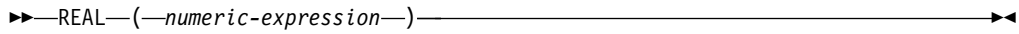
- 0 から 1 以外の数値間隔の値を生成するには、RAND 関数に必要な間隔の大きさを乗算します。例えば、0 と 10 の間の乱数 (概略値 5.8731398 など) を入手するときは、関数に 10 を乗算します。

```
SELECT RAND(:HRAND) * 10
  FROM SYSIBM.SYSDUMMY1
```

REAL

REAL 関数は、単精度浮動小数点表現を返します。

数値から実数に



ストリングから実数に

REAL 関数は、次のものの単精度浮動小数点表現を戻します。

- 数値
- 10 進数の文字ストリング表現またはグラフィック・ストリング表現
- 整数の文字ストリング表現またはグラフィック・ストリング表現
- 浮動小数点数の文字ストリング表現またはグラフィック・ストリング表現
- 10 進浮動小数点数の文字ストリング表現またはグラフィック・ストリング表現

数値から実数に

numeric-expression
引数は、任意の組み込み数値データ・タイプの値を戻す式です。

結果は、引数が単精度浮動小数点の列または変数に割り当てられたときに得られる数値と同じです。引数の数値が単精度浮動小数点数の範囲内にない場合は、エラーが戻されます。

ストリングから実数に

string-expression
数値の文字ストリング表現またはグラフィック・ストリング表現の値を戻す式。

引数がストリング式 の場合、結果は、 CAST(ストリング式 AS REAL で得られる数値と同じです。先行ブランクと末尾ブランクは除去され、結果のストリングは、浮動小数点数、10 進浮動小数点数、整数、または 10 進数の定数を形成する際の規則に合致している必要があります。引数の数値が単精度浮動小数点数の範囲内にない場合は、エラーが戻されます。

数値の整数部分からストリング式 の小数桁数を区切るために使用する必要のある 1 バイトの文字定数は、デフォルトの小数点文字です。詳しくは、 147 ページの『小数点』を参照してください。

この関数の結果は、単精度浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

注記

代替構文: アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、 218 ページの『CAST の指定』を参照してください。

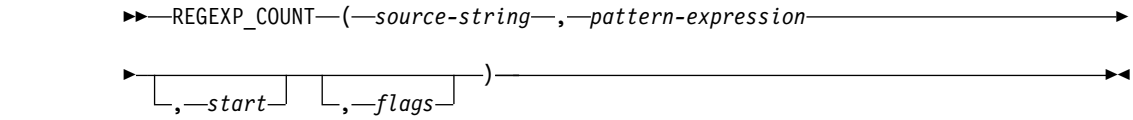
例

- 表 EMPLOYEE を使用して、何らかの手数料を得ている社員について、給与に占める手数料の割合を求めます。給与 (列 SALARY) および手数料 (列 COMM) のデータ・タイプは、DECIMAL (10 進数) です。範囲外の結果が生じる可能性を避けるために、除算が浮動小数点数で行われるように、SALARY に対して REAL が使用されます。

```
SELECT EMPNO, REAL(SALARY)/COMM
  FROM EMPLOYEE
  WHERE COMM > 0
```

REGEXP_COUNT

REGEXP_COUNT 関数は、ストリングで正規表現パターンが一致した回数を戻します。



source-string
その中で検索が行われるストリングを指定する式。式は、組み込み文字ストリング、グラフィック・ストリング、数値、日時のいずれかのデータ・タイプの値を戻す必要があります。値が UTF-16 DBCLOB ではない場合、正規表現パターンを検索する前に、暗黙的に UTF-16 DBCLOB にキャストされます。 FOR BIT DATA 属性の文字ストリングと、バイナリー・ストリングはサポートされません。ストリングの長さが 1 GB を超えてはなりません。

pattern-expression
検索パターンの正規表現ストリングを指定する式。式は、組み込み文字ストリング、グラフィック・ストリング、数値、日時のいずれかのデータ・タイプの値を戻す必要があります。値が UTF-16 DBCLOB ではない場合、正規表現パターンを検索する前に、暗黙的に UTF-16 DBCLOB にキャストされます。 FOR BIT DATA 属性の文字ストリングと、バイナリー・ストリングはサポートされません。ストリングの長さが 32K を超えてはなりません。

有効な *pattern-expression* は、検索のパターンを記述する文字および制御文字のセットで構成されます。有効な制御文字の説明については、 269 ページの『正規表現の制御文字』 を参照してください。

start
検索が開始される *source-string* 内の位置を指定する式。式は任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す必要があります。引数は、関数を評価する前に INTEGER にキャストされます。INTEGER への変換について詳しくは、 496 ページの『INTEGER または INT』を参照してください。整数の値は、1 以上でなければなりません。整数の値が *source-string* の実際の長さより大きい場合、結果は 0 になります。

flags
パターン・マッチングの特性を制御するフラグを指定する式。式は、組み込み文字ストリングまたはグラフィック・ストリングのいずれかのデータ・タイプの値を戻す必要があります。FOR BIT DATA 属性の文字ストリングと、バイナリー・ストリングはサポートされません。このストリングには、1 つ以上の有効なフラグ値を含めることができます。フラグ値の組み合わせは有効でなければなりません。空ストリングは、値「c」と同じです。

有効なフラグ文字の説明については、 269 ページの『正規表現のフラグ値』 を参照してください。

この関数の結果は、*source-string* での *pattern-expression* の出現回数を表す INTEGER になります。 *pattern-expression* が見つからず、どの引数も NULL でない場合、結果は 0 になります。

REGEXP_COUNT 関数の引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

注

前提条件: REGEXP_COUNT 関数を使用するには、 International Components for Unicode (ICU) オプションがインストールされていなければなりません。

処理: 正規表現の処理は、 International Components for Unicode (ICU) 正規表現インターフェースを使用して行われます。詳しくは、http://userguide.icu-project.org/strings/regexp を参照してください。

3 つの引数のみが指定された場合、 3 番目の引数は、*start* 引数または *flags* 引数の可能性があります。 3 番目の引数がストリングの場合は、*flags* 引数として解釈されます。それ以外の場合は、*start* 引数として解釈されます。

代替構文: REGEXP_MATCH_COUNT は REGEXP_COUNT の同義語です。

例

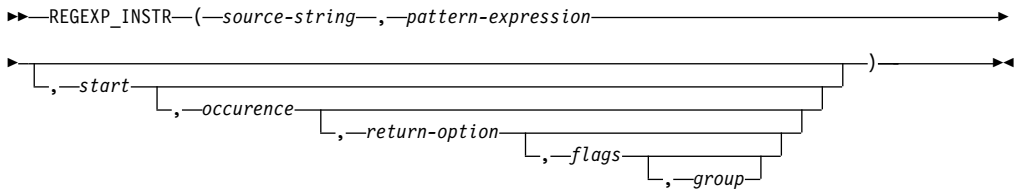
- ストリング「Steven Jones and Stephen Smith are the best players」での「Steven」または「Stephen」の出現回数を数えます。

```
SELECT REGEXP_COUNT(
  'Steven Jones and Stephen Smith are the best players',
  'Ste(v|ph)en')
FROM sysibm.sysdummy1
```

結果は 2 です。

REGEXP_INSTR

REGEXP_INSTR は、*return_option* 引数の値に応じて、一致したサブストリングの開始位置、または終了後の位置を戻します。



source-string
その中で検索が行われるストリングを指定する式。式は、組み込み文字ストリング、グラフィック・ストリング、数値、日時のいずれかのデータ・タイプの値を戻す必要があります。値が UTF-16 DBCLOB ではない場合、正規表現パターンを検索する前に、暗黙的に UTF-16 DBCLOB にキャストされます。 FOR BIT DATA 属性の文字ストリングと、バイナリー・ストリングはサポートされません。ストリングの長さが 1 GB を超えてはなりません。

pattern-expression
検索パターンの正規表現ストリングを指定する式。式は、組み込み文字ストリング、グラフィック・ストリング、数値、日時のいずれかのデータ・タイプの値を戻す必要があります。値が UTF-16 DBCLOB ではない場合、正規表現パターンを検索する前に、暗黙的に UTF-16 DBCLOB にキャストされます。 FOR BIT DATA 属性の文字ストリングと、バイナリー・ストリングはサポートされません。ストリングの長さが 32K を超えてはなりません。

有効な *pattern-expression* は、検索のパターンを記述する文字および制御文字のセットで構成されます。有効な制御文字の説明については、 269 ページの『正規表現の制御文字』 を参照してください。

start
検索が開始される *source-string* 内の位置を指定する式。式は任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す必要があります。引数は、関数を評価する前に INTEGER にキャストされます。INTEGER への変換について詳しくは、 496 ページの『INTEGER または INT』を参照してください。整数の値は、1 以上でなければなりません。整数の値が *source-string* の実際の長さより大きい場合、結果は 0 になります。

occurrence
source-string での *pattern-expression* のどのオカレンスを検索するかを指定する式。式は任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す必要があります。引数は、関数を評価する前に INTEGER にキャストされます。INTEGER への変換について詳しくは、 496 ページの『INTEGER または INT』を参照してください。整数の値は、1 以上でなければなりません。 *occurrence* が指定されない場合、デフォルト値は 1 で、*pattern-expression* の最初のオカレンスのみが考慮されることを示します。

return-option
パターンに一致したストリングの開始位置を戻すか、終了後の位置を戻すかを指定する式。式は任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す必要があります。引数は、関数を評価する

前に INTEGER にキャストされます。INTEGER への変換について詳しくは、496 ページの『INTEGER または INT』を参照してください。整数の値は、0 または 1 でなければなりません。値 0 は、そのオカレンスの開始位置を戻します。値 1 は、そのオカレンスの終了位置を戻します。 *return-option* が指定されない場合、デフォルト値は 0 になります。

flags
パターン・マッチングの特性を制御するフラグを指定する式。式は、組み込み文字ストリングまたはグラフィック・ストリングのいずれかのデータ・タイプの値を戻す必要があります。FOR BIT DATA 属性の文字ストリングと、バイナリー・ストリングはサポートされません。このストリングには、1 つ以上の有効なフラグ値を含めることができます。フラグ値の組み合わせは有効でなければなりません。空ストリングは、値「c」と同じです。

有効なフラグ文字の説明については、 269 ページの『正規表現のフラグ値』 を参照してください。

group
source-string 内の戻す位置を決定するために *pattern-expression* のどのキャプチャー・グループを使用するかを指定する式。式は任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す必要があります。引数は、関数を評価する前に INTEGER にキャストされます。INTEGER への変換について詳しくは、 496 ページの『INTEGER または INT』を参照してください。整数の値は、0 以上でなければならず、*pattern-expression* 内のキャプチャー・グループの数より大きくてはなりません。*group* を指定しない場合、デフォルトは 0 で、パターン全体と一致するストリング全体が戻されることを示します。

この関数の結果は長精度整数になります。 *pattern-expression* が見つかった場合、結果は 1 から *n* の数値です。 *n* は、*source-string* の実際の長さ + 1 です。結果値は、関数の処理に使用される位置を表します。 *pattern-expression* が見つからず、どの引数も NULL でない場合、結果は 0 になります。

REGEXP_INSTR 関数の引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

注

前提条件**:** REGEXP_INSTR 関数を使用するには、 International Components for Unicode (ICU) オプションがインストールされていなければなりません。

処理**:** 正規表現の処理は、 International Components for Unicode (ICU) 正規表現インターフェースを使用して行われます。詳しくは、http://userguide.icu-project.org/strings/regexp を参照してください。

例

- 例 *1*: 文字の後に 'o' がある最初のオカレンスを検索します。

```
SELECT REGEXP_INSTR('hello to you', '.o',1,1)
FROM sysibm.sysdummy1
```

結果は 4 です。これは、2 番目の 'l' の文字の位置です。

- 例 2: 文字の後に 'o' がある 2 番目のオカレンスを検索します。

```
SELECT REGEXP_INSTR('hello to you', '.o',1,2)
FROM sysibm.sysdummy1
```

  結果は 7 です。これは、文字 't' の位置です。

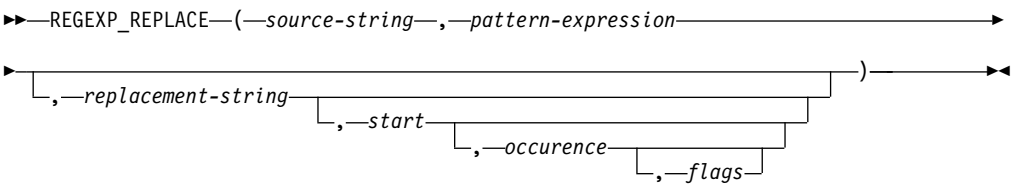
- 例 3: 大/小文字を区別しない一致を使用して、正規表現 '(.o).' の最初のキャプチャー・グループで 3 番目のオカレンスの後の位置を検索します。

```
SELECT REGEXP_INSTR('hello to you', '(.o).', 1,3,1,'i',1)
FROM sysibm.sysdummy1
```

  結果は 12 です。これは、ストリングの末尾にある文字 'u' の位置です。

REGEXP_REPLACE

REGEXP_REPLACE 関数は、ソース・ストリング内で見つかった正規表現パターンのオカレンスを、指定した置換ストリングに置き換えて、ソース・ストリングを変更したものを戻します。



source-string
その中で検索が行われるストリングを指定する式。式は、組み込み文字ストリング、グラフィック・ストリング、数値、日時のいずれかのデータ・タイプの値を戻す必要があります。値が UTF-16 DBCLOB ではない場合、正規表現パターンを検索する前に、暗黙的に UTF-16 DBCLOB にキャストされます。 FOR BIT DATA 属性の文字ストリングと、バイナリー・ストリングはサポートされません。ストリングの長さが 1 GB を超えてはなりません。

pattern-expression
検索パターンの正規表現ストリングを指定する式。式は、組み込み文字ストリング、グラフィック・ストリング、数値、日時のいずれかのデータ・タイプの値を戻す必要があります。値が UTF-16 DBCLOB ではない場合、正規表現パターンを検索する前に、暗黙的に UTF-16 DBCLOB にキャストされます。 FOR BIT DATA 属性の文字ストリングと、バイナリー・ストリングはサポートされません。ストリングの長さが 32K を超えてはなりません。

有効な *pattern-expression* は、検索のパターンを記述する文字および制御文字のセットで構成されます。有効な制御文字の説明については、 269 ページの『正規表現の制御文字』 を参照してください。

replacement-string
一致したサブストリングに対する置換ストリングを指定する式。式は、組み込み文字ストリング、グラフィック・ストリング、数値、日時のいずれかのデータ・タイプの値を戻す必要があります。値が UTF-16 DBCLOB ではない場合、正規表現パターンを検索する前に、暗黙的に UTF-16 DBCLOB にキャストされます。 FOR BIT DATA 属性の文字ストリングと、バイナリー・ストリングはサポートされません。ストリングの長さが 32K を超えてはなりません。 *replacement-string* が指定されない場合、デフォルトは空ストリングです。

replacement-string の中には、置換テキストで使用する、検索でのキャプチャー・グループ・テキストの参照を含めることができます。これらの参照の形式は '$n' または '¥n'33 です。ここで、n はキャプチャー・グループの番号で、0 はパターンに一致するストリング全体を表します。 n の値は、0 から 9 の範囲でなければならず、パターン内のキャプチャー・グループの数以下である必要があります。例えば、'$2' または '¥2' を使用して、*source-string* の中で見つかった、*pattern-expression* で指定された 2 番目のキャプチャー・グループの内容を参照できます。 *replacement-string* に文字 '$' または '¥' のリテラル参照を含める必要がある場合は、必ず、文字 '¥' をリテラル参照の前に付けて、*replacement-string* が '¥$' または '¥¥' となるようにします。

start
検索が開始される *source-string* 内の位置を指定する式。式は任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す必要があります。引数は、関数を評価する前に INTEGER にキャストされます。INTEGER への変換について詳しくは、 496 ページの『INTEGER または INT』を参照してください。整数の値は、1 以上でなければなりません。整数の値が *source-string* の実際の長さより大きい場合、元のストリングが戻されます。

occurrence
source-string での *pattern-expression* のどのオカレンスを検索して置換するかを指定する式。式は任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す必要があります。引数は、関数を評価する前に INTEGER にキャストされます。INTEGER への変換について詳しくは、 496 ページの『INTEGER または INT』を参照してください。整数の値は、0 以上でなければなりません。 *occurrence* が指定されない場合、デフォルト値は 0 で、*source-string* 内の *pattern-expression* のすべてのオカレンスが置換されることを示します。

flags
パターン・マッチングの特性を制御するフラグを指定する式。式は、組み込み文字ストリングまたはグラフィック・ストリングのいずれかのデータ・タイプの値を戻す必要があります。FOR BIT DATA 属性の文字ストリングと、バイナリー・ストリングはサポートされません。このストリングには、1 つ以上の有効なフラグ値を含めることができます。フラグ値の組み合わせは有効でなければなりません。空ストリングは、値「c」と同じです。

有効なフラグ文字の説明については、 269 ページの『正規表現のフラグ値』 を参照してください。

この関数の結果はストリングです。置換対象のパターンのオカレンスがなく、どの引数も NULL でない場合、元のストリングが戻されます。ストリングのデータ・タイプは、結果が常に可変長ストリングであることを除けば、最初の引数と 3 番目の引数を連結した場合と同じです。詳しくは、 202 ページの『連結演算子』を参照してください。

結果のデータ・タイプの長さ属性は、*source-string* と *replacement-string* の長さ属性を基に、以下の計算で求められます。すなわち、MIN(MaxTypeLen, LAS+(LAS+1)*LAR) です。ここで、MaxTypeLen は結果のデータ・タイプの最大長属性、 LAS は *source-string* のデータ・タイプの長さ属性、LAR は *replacement-string* のデータ・タイプの長さ属性です。 *replacement-string* が指定されない場合、LAR の値は 0 になります。結果ストリングの実際の長さが戻りデータ・タイプの最大長を超える場合は、エラーが戻されます。

結果の CCSID は、*source-string* と *replacement-string* の CCSID によって決まります。結果の CCSID は、最初の引数と 3 番目の引数を連結した場合と同じです。詳しくは、 133 ページの『結果のデータ・タイプに関する規則』を参照してください。

REGEXP_REPLACE 関数の引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

注

前提条件: REGEXP_REPLACE 関数を使用するには、 International Components for Unicode (ICU) オプションがインストールされていなければなりません。

処理: 正規表現の処理は、 International Components for Unicode (ICU) 正規表現インターフェースを使用して行われます。詳しくは、http://userguide.icu-project.org/strings/regexp を参照してください。

例

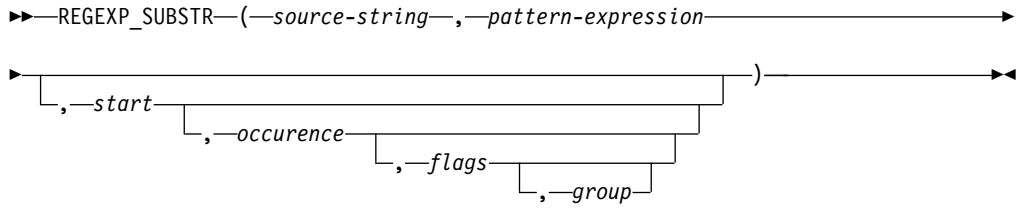
- 大文字小文字を区別して検索し、パターン 'R.d' の 2 番目のオカレンスを 'Orange' に置き換えます。

```
SELECT REGEXP_REPLACE(
  'Red Yellow RED Blue Red Green Blue',
  'R.d','Orange',1,2,'c')
FROM sysibm.sysdummy1
```

結果は、'Red Yellow RED Blue Orange Green Blue' になります。

REGEXP_SUBSTR

REGEXP_SUBSTR 関数は、ストリングで正規表現パターンに一致するサブストリングの 1 つのオカレンスを戻します。



source-string
その中で検索が行われるストリングを指定する式。式は、組み込み文字ストリング、グラフィック・ストリング、数値、日時のいずれかのデータ・タイプの値を戻す必要があります。値が UTF-16 DBCLOB ではない場合、正規表現パターンを検索する前に、暗黙的に UTF-16 DBCLOB にキャストされます。 FOR BIT DATA 属性の文字ストリングと、バイナリー・ストリングはサポートされません。ストリングの長さが 1 GB を超えてはなりません。

pattern-expression
検索パターンの正規表現ストリングを指定する式。式は、組み込み文字ストリング、グラフィック・ストリング、数値、日時のいずれかのデータ・タイプの値を戻す必要があります。値が UTF-16 DBCLOB ではない場合、正規表現パターンを検索する前に、暗黙的に UTF-16 DBCLOB にキャストされます。 FOR BIT DATA 属性の文字ストリングと、バイナリー・ストリングはサポートされません。ストリングの長さが 32K を超えてはなりません。

有効な *pattern-expression* は、検索のパターンを記述する文字および制御文字のセットで構成されます。有効な制御文字の説明については、 269 ページの『正規表現の制御文字』 を参照してください。

start
検索が開始される *source-string* 内の位置を指定する式。式は任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す必要があります。引数は、関数を評価する前に INTEGER にキャストされます。INTEGER への変換について詳しくは、 496 ページの『INTEGER または INT』を参照してください。整数の値は、1 以上でなければなりません。整数の値が *source-string* の実際の長さより大きい場合、結果は NULL 値になります。

occurrence
source-string での *pattern-expression* のどのオカレンスを検索するかを指定する式。式は任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す必要があります。引数は、関数を評価する前に INTEGER にキャストされます。INTEGER への変換について詳しくは、 496 ページの『INTEGER または INT』を参照してください。整数の値は、1 以上でなければなりません。 *occurrence* が指定されない場合、デフォルト値は 1 で、*pattern-expression* の最初のオカレンスのみが考慮されることを示します。

flags
パターン・マッチングの特性を制御するフラグを指定する式。式は、組み込み文

字ストリングまたはグラフィック・ストリングのいずれかのデータ・タイプの値を戻す必要があります。FOR BIT DATA 属性の文字ストリングと、バイナリー・ストリングはサポートされません。このストリングには、1 つ以上の有効なフラグ値を含めることができます。フラグ値の組み合わせは有効でなければなりません。空ストリングは、値「c」と同じです。

有効なフラグ文字の説明については、 269 ページの『正規表現のフラグ値』 を参照してください。

group

source-string での *pattern-expression* のどのキャプチャー・グループを戻すかを指定する式。式は任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す必要があります。引数は、関数を評価する前に INTEGER にキャストされます。INTEGER への変換について詳しくは、 496 ページの『INTEGER または INT』を参照してください。整数の値は、0 以上でなければならず、*pattern-expression* 内のキャプチャー・グループの数より大きくてはなりません。 *group* を指定しない場合、デフォルトは 0 で、パターン全体と一致するストリング全体が戻されることを示します。

この関数の結果はストリングです。結果のデータ・タイプは、*source-string* のデータ・タイプによって異なります。

source-string のデータ・タイプ	**REGEXP_SUBSTR** の場合の結果のデータ・タイプ
CHAR や VARCHAR または数値や日時	VARCHAR
CLOB	CLOB
GRAPHIC または VARGRAPHIC	VARGRAPHIC
DBCLOB	DBCLOB

結果のデータ・タイプの長さ属性は、*source-string* の長さ属性と同じです。結果の実際の長さは、ストリング内で *pattern-expression* に一致するオカレンスの長さです。 *pattern-expression* が見つからない場合、結果は NULL 値になります。

結果の CCSID は、*source-string* と同じです。

REGEXP_SUBSTR 関数の引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

注

前提条件**:** REGEXP_SUBSTR 関数を使用するには、 International Components for Unicode (ICU) オプションがインストールされていなければなりません。

処理**:** 正規表現の処理は、 International Components for Unicode (ICU) 正規表現インターフェースを使用して行われます。詳しくは、http://userguide.icu-project.org/strings/regexp を参照してください。

代替構文**:** REGEXP_EXTRACT は REGEXP_SUBSTR の同義語です。

例

- 例 *1*: 任意の文字の後に 'o' があるパターンに一致するストリングを戻します。

```
SELECT REGEXP_SUBSTR('hello to you', '.o',1,1)
FROM sysibm.sysdummy1
```

  結果は 'lo' です。

- 例 *2*: 任意の文字の後に 'o' があるパターンに一致するストリングの 2 番目のオカレンスを戻します。

```
SELECT REGEXP_SUBSTR('hello to you', '.o',1,2)
FROM sysibm.sysdummy1
```

  結果は 'to' です。

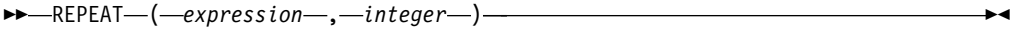
- 例 *3*: 任意の文字の後に 'o' があるパターンに一致するストリングの 3 番目のオカレンスを戻します。

```
SELECT REGEXP_SUBSTR('hello to you', '.o',1,3)
FROM sysibm.sysdummy1
```

  結果は 'yo' です。

REPEAT

REPEAT 関数は、整数 回繰り返される式 で構成されるストリングを戻します。



expression
繰り返すストリングを指定する式。このストリングは組み込み数値またはストリング式でなければなりません。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、672 ページの『VARCHAR』を参照してください。

integer
その値が正整数かゼロである BIGINT、INTEGER、または SMALLINT の組み込みデータ・タイプを戻す式。整数は、ストリングを繰り返す回数を指定します。

関数の結果のデータ・タイプは、最初の引数のデータ・タイプによって異なります。

式 のデータ・タイプ	結果のデータ・タイプ
CHAR、VARCHAR、または任意の数値タイプ	VARCHAR
CLOB	CLOB
GRAPHIC または VARGRAPHIC	VARGRAPHIC
DBCLOB	DBCLOB
BINARY または VARBINARY	VARBINARY
BLOB	BLOB

integer が定数の場合、結果の長さ属性は、*expression* の長さ属性を *integer* 倍したものと結果データ・タイプの最大長の小さい方です。定数ではない場合は、長さ属性は結果のデータ・タイプによって次のように異なります。

- BLOB、CLOB、または DBCLOB の場合は 1,048,576
- VARCHAR または VARBINARY の場合は 4000
- VARGRAPHIC の場合は 2000

結果の実際の長さは、*expression* の実際の長さに *integer* を乗算した値になります。結果ストリングの実際の長さが戻りタイプの最大長を超える場合は、エラーが戻されます。

引数のどちらかが NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数のどちらかが NULL である場合は、結果は NULL 値になります。

結果の CCSID は *expression* の CCSID です。[71]

71. *expression* の値が、適切な形式の混合データ・ストリングではない混合データの場合、結果は適切な形式の混合データ・ストリングにはなりません。

例

- 「abc」を 2 回繰り返して「abcabc」を作成します。

```
SELECT REPEAT('abc', 2)
  FROM SYSIBM.SYSDUMMY1
```

- 「REPEAT THIS」という句を 5 回リストします。 CHAR 関数を使用して、出力を 60 バイトに制限します。

```
SELECT CHAR( REPEAT('REPEAT THIS', 5), 60)
  FROM SYSIBM.SYSDUMMY1
```

  この例の結果は「REPEAT THISREPEAT THISREPEAT THISREPEAT THISREPEAT THIS　　」です。

- 次の照会の場合、ストリングをゼロ回繰り返した結果は長さがゼロのストリングである空ストリングであるため、LENGTH 関数は 0 の値を戻します。

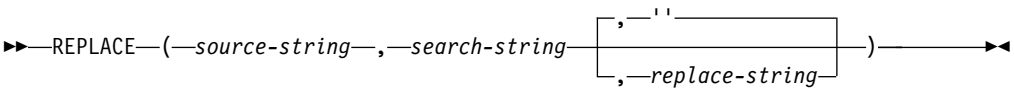
```
SELECT LENGTH( REPEAT('REPEAT THIS', 0) )
  FROM SYSIBM.SYSDUMMY1
```

- 次の照会の場合、空ストリングを任意の回数繰り返した結果は長さがゼロのストリングである空ストリングであるため、LENGTH 関数は 0 の値を戻します。

```
SELECT LENGTH( REPEAT('', 5) )
  FROM SYSIBM.SYSDUMMY1
```

REPLACE

REPLACE 関数は、ソース・ストリング の中の検索ストリング のすべての出現箇所を置換ストリング で置き換えます。ソース・ストリング の中で検索ストリングが検出されない場合は、ソース・ストリング が未変更のまま戻されます。



source-string
ソース・ストリングを指定する式。ソース・ストリング は、組み込み数値またはストリング式でなければなりません。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。

search-string
ソース・ストリングから除去するストリングを指定する式。検索ストリングは、組み込み数値またはストリング式でなければなりません。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。

replace-string
置換ストリングを指定する式。置換ストリング は、組み込み数値またはストリング式でなければなりません。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、672 ページの『VARCHAR』を参照してください。

replace-string が空ストリングであるか指定されない場合、ソース・ストリングから削除されるストリングは置き換えられません。

source-string、*search-string*、および *replace-string* は、互換性がなければなりません。データ・タイプの互換性についての詳細は、 113 ページの『割り当ておよび比較』を参照してください。

関数の結果のデータ・タイプは、引数のデータ・タイプによって異なります。結果のデータ・タイプは、結果は常に可変長ストリングであることを除けば、 3 つの引数を連結した場合と同じです。詳しくは、 139 ページの『ストリングを結合する演算に適用される変換規則』を参照してください。

結果の長さ属性は、引数によって次のように異なります。

- 検索ストリング が可変長の場合、結果の長さ属性は次のとおりです。

  ```
  (L3 * L1)
  ```

- 置換ストリング の長さ属性が検索ストリング の長さ属性より小さいか等しい場合は、結果の長さ属性はソース・ストリング の長さ属性です。
- それ以外の場合、結果の長さ属性は次のとおりです。

  ```
  (L3 * (L1/L2)) + MOD(L1,L2)
  ```

各値は、次のとおりです。

L1 はソース・ストリングの長さ属性
L2 は検索ストリングの長さ属性
L3 は置換ストリングの長さ属性

結果の長さ属性が結果のデータ・タイプの最大長を超える場合は、エラーが戻されます。

結果の実際の長さは、ソース・ストリング 内の検索ストリング の出現回数に置換ストリング の実際の長さを乗算したものをソース・ストリング の長さに足して、検索ストリング の実際の長さを引いた長さです。結果ストリングの実際の長さが結果のデータ・タイプの最大長を超える場合は、エラーが戻されます。

引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

結果の CCSID は、ソース・ストリング、検索ストリング、および置換ストリングの CCSID によって決定されます。結果の CCSID は、3 つの引数を連結した場合と同じです。詳しくは、 139 ページの『ストリングを結合する演算に適用される変換規則』を参照してください。

例

- ストリング「DINING」の文字「N」のすべての出現箇所を「VID」に置換します。 CHAR 関数を使用して、出力を 10 バイトに制限します。

```
SELECT CHAR(REPLACE( 'DINING', 'N', 'VID' ), 10)
  FROM SYSIBM.SYSDUMMY1
```

  結果は、ストリング「DIVIDIVIDG」です。

- ストリング「ABCXYZ」のストリング「ABC」を空ストリングで置換します。これは、ストリングから「ABC」を除去するのと同じです。

```
SELECT REPLACE( 'ABCXYZ', 'ABC', '' )
  FROM SYSIBM.SYSDUMMY1
```

  結果は、ストリング「XYZ」になります。

  3 番目の引数を省略しても、同じ結果が戻ります。

```
SELECT REPLACE( 'ABCXYZ', 'ABC')
  FROM SYSIBM.SYSDUMMY1
```

- ストリング「ABCCABCC」のストリング「ABC」を「AB」に置換します。この例は、置換するストリングのすべての出現箇所は置換が行われるよりも前に識別されるため、置換するストリング (この場合は「ABC」) が結果に残されている場合があることを示しています。

```
SELECT REPLACE( 'ABCCABCC', 'ABC', 'AB' )
  FROM SYSIBM.SYSDUMMY1
```

  結果は、ストリング「ABCABC」です。

RID

RID 関数は、行の相対レコード番号を BIGINT として返します。

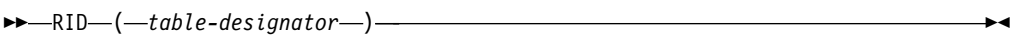


table-designator
SQL ステートメント内で RID 関数と同じ相対位置にある列を修飾するために使用できる表指定子。表指定子の詳細については、 166 ページの『表指定子』を参照してください。

SQL 命名規則では、表名は修飾できます。システム命名規則では、表名は修飾できません。

table-designator は、*table-function*、*collection-derived-table*、VALUES 節、または *data-change-table-reference* を示すものであってはなりません。引数がビュー、共通表式、またはネストされた表式を示している場合、その外部副選択は直接的または間接的に表を参照しなければなりません。

引数がビュー、共通表式、またはネストされた表式を示している場合、この関数はその基本表の相対レコード番号を返します。引数が、複数の基本表から派生したビュー、共通表式、またはネストされた表式を示している場合、この関数は、そのビュー、共通表式、またはネストされた表式の外側の副選択内にある最初の表の相対レコード番号を返します。

引数が分散表を示している場合、この関数は、その行が位置指定されているノードの行の相対レコード番号を戻します。引数がパーティション化された表を示している場合、この関数は、その行が位置指定されているパーティションの行の相対レコード番号を戻します。これは、RID がパーティション化された表または分散表の各行に固有のものではないことを意味します。

引数は、外側の全選択に、集約関数、 GROUP BY 節、HAVING 節、UNION、INTERSECT、または EXCEPT 節、DISTINCT 節、VALUES 節、または *table-function* が含まれている、ビュー、共通表式、またはネストされた表式を指定してはなりません。全選択が集約関数、GROUP BY 文節、または HAVING 文節を含む場合、SELECT 文節に RID 関数を指定することはできません。

結果のデータ・タイプは、64 ビット整数になります。結果は NULL になる場合があります。

例

- この例では、表 EMPLOYEE から、部門 20 の社員について、その相対レコード番号と社員名を戻します。

```
SELECT RID(EMPLOYEE), LASTNAME
  FROM EMPLOYEE
  WHERE DEPTNO = 20
```

RIGHT

RIGHT 関数は、式 の右端から整数 個の文字を戻します。

expression が文字ストリングの場合は、結果は文字ストリングになります。 *expression* がグラフィック・ストリングの場合は、結果はグラフィック・ストリングになります。 *expression* が 2 進ストリングの場合は、結果は 2 進ストリングになります。

expression
結果が導き出される元になるストリングを指定する式。このストリングは組み込み数値またはストリング式でなければなりません。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。

式 のサブストリングは、式 のゼロ個以上の連続したバイトです。 *expression* が文字ストリングまたはグラフィック・ストリングの場合、1 文字は SBCS、DBCS、またはマルチバイト文字です。*expression* が 2 進ストリングの場合は、結果は、引数のバイト数です。

integer
組み込み整数データ・タイプを戻す式。整数は結果の長さを指定します。整数は、0 以上で *n* 以下でなければなりません。 *n* は式 の長さ属性です。

この関数の結果は、式 と同じ長さ属性を持つ可変長ストリングで、データ・タイプは式 のデータ・タイプに応じて以下のようになります。

式 のデータ・タイプ	結果のデータ・タイプ
CHAR または VARCHAR	VARCHAR
CLOB	CLOB
GRAPHIC または VARGRAPHIC	VARGRAPHIC
DBCLOB	DBCLOB
BINARY または VARBINARY	VARBINARY
BLOB	BLOB

結果の実際の長さは整数 です。

引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

結果の CCSID は式 の CCSID と同じです。

例

- ホスト変数 ALPHA の値が「ABCDEF」であると想定します。次のステートメントは、

```
SELECT RIGHT( :ALPHA, 3)
  FROM SYSIBM.SYSDUMMY1
```

ALPHA の右端の 3 文字である値「DEF」を戻します。

- 次のステートメントは、長さゼロのストリングを戻します。

```
SELECT RIGHT( 'ABCABC', 0)
  FROM SYSIBM.SYSDUMMY1
```

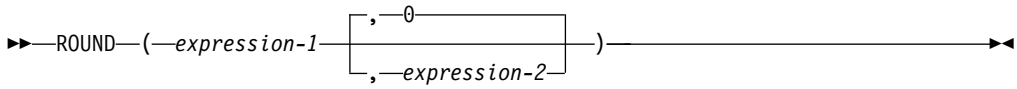
- NAME は VARCHAR(128) の列で Unicode UTF-8 でエンコードされ、値 'Jürgen' を含むものと仮定します。

```
SELECT RIGHT(NAME, 5), SUBSTR(NAME, 3, 5)
  FROM T1
  WHERE NAME = 'Jürgen'
```

  RIGHT については値 'ürgen'、SUBSTR(NAME, 3, 5) については印刷不能なストリング (X'BC7267656E') が戻ります。

ROUND

ROUND 関数は、*expression–1* を、小数点の右側または左側の特定の桁で丸めた値を返します。



expression–1
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点に変換されます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

expression-1 が 10 進浮動小数点データ・タイプの場合、DECFLOAT ROUNDING MODE は使用されません。ROUND の丸め動作は、ROUND_HALF_UP の値に対応します。別の丸め動作が必要な場合は、QUANTIZE 関数を使用します。

expression–2
組み込みデータ・タイプ BIGINT、INTEGER、または SMALLINT の値を戻す式。

expression–2 が正の場合、*expression–1* は小数点の右側の *expression–2* 桁目で丸められます。

expression–2 が負の場合、*expression–1* は小数点の左側の 1 + (*expression–2* の絶対値) 桁に相当する桁で丸められます。 *expression–2* の絶対値が小数点の左側の桁数より大きい場合、結果は 0 になります。(例えば、ROUND(748.58,-4) は 0 を返します。)

expression–2 が指定されていない場合、 *expression–1* は小数点の左側の 0 桁で丸められます。

expression–1 が正で、その桁の値が 5 である場合は、次に大きい正の数に切り上げられます。 *expression–1* が負で、その桁の値が 5 である場合は、次に低い負の数に切り下げられます。

結果のデータ・タイプおよび長さ属性は、最初の引数のデータ・タイプおよび長さ属性と同じです。ただし、*expression–1* が DECIMAL または NUMERIC であり、精度が最大精度 (*mp*) より小さい場合は、精度は 1 だけ増加します。例えば、データ・タイプが DECIMAL(5,2) の引数の場合、結果は DECIMAL(6,2) になります。データ・タイプが DECIMAL(63,2) の引数の場合、結果は DECIMAL(63,2) になります。

どちらかの引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数のどちらかが NULL である場合は、結果は NULL 値になります。

例

- 数値 873.726 を、小数点から 2、1、0、-1、-2、-3、-4 の位置で丸めます。

```
SELECT ROUND(873.726,  2),
       ROUND(873.726,  1),
       ROUND(873.726,  0),
       ROUND(873.726, -1),
       ROUND(873.726, -2),
       ROUND(873.726, -3),
       ROUND(873.726, -4)
  FROM SYSIBM.SYSDUMMY1
```

それぞれ以下の値が戻されます。

```
0873.730   0873.700  0874.000  0870.000  0900.000  1000.000  0000.000
```

- 正負両方の数値を計算します。

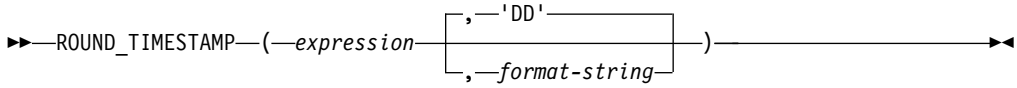
```
SELECT ROUND( 3.5, 0),
       ROUND( 3.1, 0),
       ROUND(-3.1, 0),
       ROUND(-3.5, 0)
  FROM SYSIBM.SYSDUMMY1
```

それぞれ以下の値が戻されます。

```
04.0   03.0  -03.0  -04.0
```

ROUND_TIMESTAMP

ROUND_TIMESTAMP 関数は、*expression* を、*format-string* で指定された単位に丸めたタイム・スタンプを戻します。*format-string* を指定しないと、*expression* は、*format-string* に「DD」が指定されたものとして、最も近い日に丸められます。



expression
日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を返す式。

expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。最初に TIMESTAMP(12) 値に変換されます。 日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

format-string
組み込み文字ストリング・データ・タイプか、グラフィック・ストリング・データ・タイプを戻す式。 *format-string* には、*expression* を丸める方法を示すテンプレートが含まれています。例えば、*format-string* が「DD」ならば、*expression* によって表されるタイム・スタンプは、最も近い日に丸められます。前後のブランクはストリングから除去され、結果として得られるサブストリングは、タイム・スタンプとして有効なテンプレートでなければなりません。その結果の値は大文字変換されるため、値の中の文字は大文字、小文字のどちらでも構いません。*format-string* に指定できる値を、表 54 にリストします。

関数の結果は以下のタイム・スタンプ精度の TIMESTAMP になります。

- *expression* のデータ・タイプが TIMESTAMP(*p*) の場合は *p*。
- *expression* のデータ・タイプが DATE の場合は 0。
- それ以外の場合は 6。

引数のどちらかが NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数のどちらかが NULL である場合は、結果は NULL 値になります。

表 *54. ROUND_TIMESTAMP* および *TRUNC_TIMESTAMP* の形式モデル

形式モデル	丸めまたは切り捨ての単位	**ROUND_TIMESTAMP** の例	**TRUNC_TIMESTAMP** の例
CC SCC	4 桁の年号のうち最初の 2 桁より 1 大きい桁。(世紀の 50 番目の年で切り上げる)	入力値: 1897-12-04-12.22.22.000000 結果: 1901-01-01-00.00.00.000000	入力値: 1897-12-04-12.22.22.000000 結果: 1801-01-01-00.00.00.000000

表 54. *ROUND_TIMESTAMP* および *TRUNC_TIMESTAMP* の形式モデル (続き)

形式モデル	丸めまたは切り捨ての単位	**ROUND_TIMESTAMP** の例	**TRUNC_TIMESTAMP** の例
YYYY SYYYY YEAR SYEAR YYY YY Y	年 (7 月 1 日を次の年の 1 月 1 日に切り上げる)	入力値: 1897-12-04-12.22.22.000000 結果: 1898-01-01- 00.00.00.000000	入力値: 1897-12-04-12.22.22.000000 結果: 1897-01-01- 00.00.00.000000
IYYY IYY IY I	ISO 年 (7 月 1 日を次の ISO 年の最初の日に切り上げる。ISO 年の最初の日は、最初の ISO 週の月曜日と定義される。)	入力値: 1897-12-04-12.22.22.000000 結果: 1898-01-03- 00.00.00.000000	入力値: 1897-12-04-12.22.22.000000 結果: 1897-01-04- 00.00.00.000000
Q	四半期 (四半期の 2 番目の月の 16 日目で切り上げる)	入力値: 1999-06-04-12.12.30.000000 結果: 1999-07-01- 00.00.00.000000	入力値: 1999-06-04-12.12.30.000000 結果: 1999-04-01- 00.00.00.000000
MONTH MON MM RM	月 (月の 16 日目で切り上げる)	入力値: 1999-06-18-12.12.30.000000 結果: 1999-07-01- 00.00.00.000000	入力値: 1999-06-18-12.12.30.000000 結果: 1999-06-01- 00.00.00.000000
WW	その年の元日と同じ曜日 (元日を基準として、週の第 4 日目の 12 時間目で切り上げる)	入力値: 2000-05-05-12.12.30.000000 結果: 2000-05-06- 00.00.00.000000	入力値: 2000-05-05-12.12.30.000000 結果: 2000-04-29- 00.00.00.000000
IW	その ISO 年の最初の日と同じ曜日 (ISO 年の最初の日を基準として、週の第 4 日目の 12 時間目で切り上げる)	入力値: 2000-05-05-12.12.30.000000 結果: 2000-05-08- 00.00.00.000000	入力値: 2000-05-05-12.12.30.000000 結果: 2000-05-01- 00.00.00.000000
W	その月の最初の日と同じ曜日 (その月の初日を基準として、週の第 4 日目の 12 時間目で切り上げる)	入力値: 2000-06-21-12.12.30.000000 結果: 2000-06-22- 00.00.00.000000	入力値: 2000-06-22-12.12.30.000000 結果: 2000-06-15- 00.00.00.000000
DDD DD J	日 (その日の 12 時間目で切り上げる)	入力値: 2000-05-17-12.59.59.000000 結果: 2000-05-18- 00.00.00.000000	入力値: 2000-05-17-12.59.59.000000 結果: 2000-05-17- 00.00.00.000000

表 54. *ROUND_TIMESTAMP* および *TRUNC_TIMESTAMP* の形式モデル (続き)

形式モデル	丸めまたは切り捨ての単位	**ROUND_TIMESTAMP** の例	**TRUNC_TIMESTAMP** の例
DAY DY D	週の開始日 (週の第 4 日目の 12 時間目を基準として切り上げる。週の最初の日は、常に日曜日です)	入力値: 2000-05-17-12.59.59.000000 結果: 2000-05-21-00.00.00.000000	入力値: 2000-05-17-12.59.59.000000 結果: 2000-05-14-00.00.00.000000
HH HH12 HH24	時 (30 分で切り上げる)	入力値: 2000-05-17-23.59.59.000000 結果: 2000-05-18-00.00.00.000000	入力値: 2000-05-17-23.59.59.000000 結果: 2000-05-17-23.00.00.000000
MI	分 (30 秒で切り上げる)	入力値: 2000-05-17-23.58.45.000000 結果: 2000-05-17-23.59.00.000000	入力値: 2000-05-17-23.58.45.000000 結果: 2000-05-17-23.58.00.000000
SS	秒 (500000 マイクロ秒で切り上げる)	入力値: 2000-05-17-23.58.45.500000 結果: 2000-05-17-23.58.46.000000	入力値: 2000-05-17-23.58.45.500000 結果: 2000-05-17-23.58.45.000000
注: ISO 年は、その年の最初の ISO 週の最初の日に始まります。つまり、1 月 1 日の最大で 3 日前であるか、3 日後である可能性があります。詳しくは、 699 ページの『WEEK_ISO』を参照してください。			

例

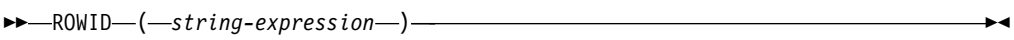
- 現在の年を最も近い月の値に丸めてホスト変数 RND_TMSTMP を設定します。

```
SET :RND_TMSTMP = ROUND_TIMESTAMP('2000-03-18-17.30.00', 'MONTH');
```

ホスト変数 RND_TMSTMP には、値 2000-04-01-00.00.00.000000 が設定されます。

ROWID

ROWID 関数は、文字ストリングを行 ID にキャストします。



string-expression
文字ストリング値を戻す式。ストリングにはどのような値が含まれていても構いませんが、有効な ROWID 値が返されるようにするには、Db2 for z/OS または Db2 for i により既に生成されている ROWID 値を指定することをお勧めします。例えば、この関数を使用して、CHAR 値にキャストされた ROWID 値を、再び ROWID 値に戻すことができます。

ストリング式 の実際の長さが 40 より小さい場合、結果の埋め込みは行われません。ストリング式 の実際の長さが 40 より大きい場合は、結果は切り捨てられます。非ブランク文字が切り捨てられた場合は、警告が戻されます。

結果の長さ属性は、40 です。 結果の実際の長さは、ストリング式 の長さです。

この関数の結果は行 ID です。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

注記

代替構文: アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、 218 ページの『CAST の指定』を参照してください。

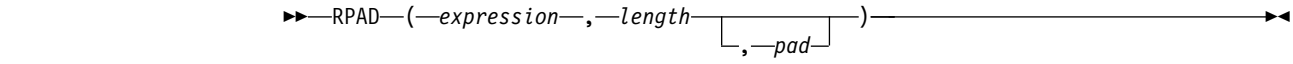
例

- 表 EMPLOYEE に ROWID 列 EMP_ROWID が含まれているとします。また、この表には、X'F0DFD230E3C0D80D81C201AA0A280100000000000203' という行 ID 値で識別される行が含まれているものとします。直接行アクセスを使用して、その行に該当する社員番号を選択します。

```
SELECT EMPNO
  FROM EMPLOYEE
  WHERE EMP_ROWID = ROWID(X'F0DFD230E3C0D80D81C201AA0A280100000000000203')
```

RPAD

RPAD 関数は、右側に埋め込みが行われた *expression* で構成されるストリングを戻します。



RPAD 関数は、*expression* 内の先行ブランクまたは末尾ブランクを有効として扱います。埋め込みは、*expression* の実際の長さが *length* より短く、*pad* が空ストリングでない場合のみ行われます。

expression
結果が導き出される元になるストリングを指定する式。

expression は、組み込みストリング、数値、または日時のデータ・タイプでなければなりません。数値引数または日時引数は、関数を評価する前に、現行サーバーでデフォルト SBCS CCSID である CCSID を使用して VARCHAR にキャストされます。数値または日時から文字ストリングへの変換について詳しくは、672 ページの『VARCHAR』 を参照してください。

length
結果の長さを指定する式。式は、組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す必要があります。式のデータ・タイプが INTEGER でない場合、その値は関数を評価する前に暗黙的に INTEGER にキャストされます。値はゼロであるか、*n* 以下の正整数である必要があります。ここで、*n* は結果のデータ・タイプの最大長です。詳しくは、1845 ページの『付録 A. SQL の制約』を参照してください。

expression がグラフィック・ストリングの場合、*length* は DBCS または Unicode グラフィック文字の数を示します。 *expression* が文字ストリングの場合、*length* は文字数を示します (1 文字が 1 バイトまたは複数バイトで構成されることもあります)。 *expression* がバイナリー・ストリングの場合、*length* はバイト数を示します。

pad
埋め込むストリングを指定する式。この式は、組み込みデータ・タイプであるストリング、数値、日時のいずれかの値を戻す必要があります。値が数値または日時のデータ・タイプの場合、関数を評価する前に、現行サーバーでデフォルト SBCS CCSID である CCSID を使用して VARCHAR に暗黙的にキャストされます。

pad が指定されていない場合、埋め込み文字は次のように設定されます。

- 文字ストリングおよびグラフィック・ストリングの場合、*expression* のデータ・タイプと CCSID に基づいた 1 バイト、2 バイト、 UTF-16、または UTF-8 のブランク文字。[72]
- バイナリー・ストリングの場合、16 進のゼロ。

72. UTF-16 または UCS-2 では、コード・ポイント X'0020' および X'3000' でブランク文字を定義しています。データベース・マネージャーは、コード・ポイント X'0020' の位置にあるブランクを埋め込みに使用します。データベース・マネージャーは、UTF-8 でコード・ポイント X'20' のブランクを埋め込みます。

expression の値と *pad* の値は、互換性のあるデータ・タイプである必要があります。*pad* の CCSID が *expression* の CCSID と異なる場合、*pad* 値は *expression* の CCSID に変換されます。データ・タイプの互換性についての詳細は、 113 ページの『割り当ておよび比較』を参照してください。

結果のデータ・タイプは、式 のデータ・タイプによって異なります。

式 のデータ・タイプ	**RPAD** の場合の結果のデータ・タイプ
CHAR や VARCHAR または数値や日時	VARCHAR
CLOB	CLOB
GRAPHIC または VARGRAPHIC	VARGRAPHIC
DBCLOB	DBCLOB
BINARY または VARBINARY	VARBINARY
BLOB	BLOB

結果の長さ属性は *length* によって決まります。長さ をゼロより大きい整数定数で明示的に指定すると、結果の長さ属性は長さ になります。*length* にゼロの整数定数を明示的に指定すると、結果の長さ属性は 1 になります。 *length* を式で指定すると、結果の長さ属性は、*m*+100 と、結果データ・タイプの最大長のうち、小さい方になります。ここで、*m* は *expression* の長さ属性です。詳しくは、 1845 ページの『付録 A. SQL の制約』を参照してください。

結果の実際の長さは、*length* から決定されます。

- *length* が 0 の場合、実際の長さは 0 であり、結果は空の結果ストリングになります。
- *length* が *expression* の実際の長さに等しい場合、実際の長さは、*expression* の長さになります。
- *length* が *expression* の実際の長さよりも小さい場合、結果は切り捨てられます。実際の長さは、結果のデータ・タイプが可変長混合データまたは可変長 Unicode であるケースを除き、*length* になります。この場合、切り捨ては常に、完全な文字を単位として行われます。
  - Unicode データでは、2 バイト文字が分断されないように、実際の長さは *length*-1 になることがあります。
  - 混合データでは、2 バイト文字や、「シフトイン」文字 (X'0F') および「シフトアウト」文字 (X'0E') で発生しうる切り捨てのために、実際の長さは *length*-3 まで小さくなることがあります。
- *length* が *expression* の実際の長さより大きい場合、結果のデータ・タイプが可変長混合データまたは可変長 Unicode で *pad* に 2 バイト文字が含まれるケースを除き、実際の長さは *length* になります。この場合、埋め込みは常に、完全な文字を単位として行われます。
  - Unicode データでは、2 バイト文字が分断されないように、実際の長さは *length*-1 になることがあります。

– 混合データでは、2 バイト文字や、「シフトイン」文字 (X'0F') および「シフトアウト」文字 (X'0E') で発生しうる切り捨てのために、実際の長さは *length*-3 まで小さくなることがあります。また、その結果の「継ぎ目に」余分なシフト・コードが入ることはありません。したがって、pad が「シフトイン文字」文字 (X'0F') で終わるストリングで、expression が「シフトアウト」文字 (X'0E') で始まる場合、これらの 2 バイト (pad のシフトイン文字と expression のシフトアウト文字) は結果から除去されます。

引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

結果の CCSID は式 の CCSID と同じです。

例

- 例 *1:* NAME が VARCHAR(15) の列で、値「Chris」、「Meg」、および「Jeff」を含んでいるものとします。次の照会では、値の右側にピリオドが完全に埋め込まれます。

  ```
  SELECT RPAD(NAME,15,'.' ) AS NAME FROM T1;
  ```

  これは、以下のものを戻します。

  ```
  NAME
  ---------------
  Chris..........
  Meg............
  Jeff...........
  ```

- 例 *2:* NAME が VARCHAR(15) の列で、値「Chris」、「Meg」、および「Jeff」を含んでいるものとします。次の照会では、長さ 5 までのみ各値にピリオドが埋め込まれます。

  ```
  SELECT RPAD(NAME,5,'.' ) AS NAME FROM T1;
  ```

  これは、以下のものを戻します。

  ```
  NAME
  -----
  Chris
  Meg..
  Jeff.
  ```

- 例 *3:* NAME が CHAR(15) の列で、値「Chris」、「Meg」、および「Jeff」を含んでいるものとします。RTRIM の結果は、ブランクが取り除かれた可変長ストリングです。

  ```
  SELECT RPAD(RTRIM(NAME),15,'.' ) AS NAME FROM T1;
  ```

  これは、以下のものを戻します。

  ```
  NAME
  ---------------
  Chris..........
  Meg............
  Jeff...........
  ```

- 例 *4:* NAME が VARCHAR(15) の列で、値「Chris」、「Meg」、および「Jeff」を含んでいるものとします。次の照会では、値の右側に *pad* が完全に埋め込まれます (場合によっては、埋め込み指定の部分的なインスタンスが戻されることがあります)。

```
SELECT RPAD(NAME,15,'123' ) AS NAME FROM T1;
```

これは、以下のものを戻します。

```
NAME
---------------
Chris1231231231
Meg123123123123
Jeff12312312312
```

- 例 *5:* NAME が VARCHAR(15) の列で、値「Chris」、「Meg」、および「Jeff」を含んでいるものとします。「Chris」には切り捨て、「Meg」には埋め込みが行われ、「Jeff」は変更されません。

```
SELECT RPAD(NAME,4,'.' ) AS NAME FROM T1;
```

これは、以下のものを戻します。

```
NAME
----
Chri
Meg.
Jeff
```

RRN

RRN 関数は、行の相対レコード番号を戻します。

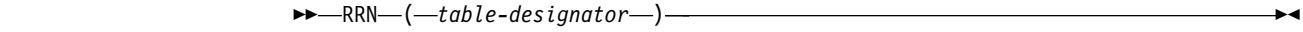


table-designator
SQL ステートメント内で RRN 関数と同じ相対位置にある列を修飾するために使用できる表指定子。表指定子の詳細については、 166 ページの『表指定子』を参照してください。

SQL 命名規則では、表名は修飾できます。システム命名規則では、表名は修飾できません。

table-designator は、*collection-derived-table*、VALUES 節、*table-function*、または *data-change-table-reference* を指定するものであってはなりません。引数がビュー、共通表式、またはネストされた表式を示している場合、その外部副選択は直接的または間接的に表を参照しなければなりません。

引数がビュー、共通表式、またはネストされた表式を示している場合、この関数はその基本表の相対レコード番号を返します。引数が、複数の基本表から派生したビュー、共通表式、またはネストされた表式を示している場合、この関数は、そのビュー、共通表式、またはネストされた表式の外側の副選択内にある最初の表の相対レコード番号を返します。

引数が分散表を示している場合、この関数は、その行が位置指定されているノードの行の相対レコード番号を戻します。引数がパーティション化された表を示している場合、この関数は、その行が位置指定されているパーティションの行の相対レコード番号を戻します。これは、RRN がパーティション化された表または分散表の各行に固有のものではないことを意味します。

引数は、外側の全選択に、集約関数、GROUP BY 節、HAVING 節、UNION、INTERSECT、または EXCEPT 節、 DISTINCT 節、VALUES 節、または *table-function* が含まれている、ビュー、共通表式、またはネストされた表式を指定してはなりません。全選択が集約関数、GROUP BY 文節、または HAVING 文節を含む場合、SELECT 文節に RRN 関数を指定することはできません。

結果のデータ・タイプは、精度が 15 で位取りが 0 の 10 進数です。結果は NULL の場合があります。

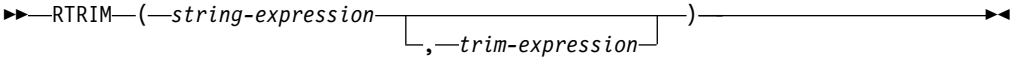
例

- この例では、表 EMPLOYEE から、部門 20 の社員について、その相対レコード番号と社員名を戻します。

```
SELECT RRN(EMPLOYEE), LASTNAME
  FROM EMPLOYEE
  WHERE DEPTNO = 20
```

RTRIM

RTRIM 関数は、指定した文字のいずれかを式の末尾から除去します。



RTRIM 関数は、*string-expression* の末尾から、*trim-expression* に含まれるすべての文字を除去します。照合順序は、検索に影響しません。*string-expression* が FOR BIT DATA として定義される場合、またはバイナリー・データ・タイプである場合、検索は、*trim-expression* に含まれる各バイトを *string-expression* の末尾にあるバイトと比較することによって行われます。

string-expression
任意の組み込み数値データ・タイプ、日時データ・タイプ、またはストリング・データ・タイプの値を戻す式。[73]数値または日時の引数は、関数の評価前に文字ストリングにキャストされます。数値または日時から文字ストリングへの変換について詳しくは、 672 ページの『VARCHAR』 を参照してください。

trim-expression
string-expression の末尾から除去する文字を指定する式。式は任意の組み込み数値、日時、またはストリングのデータ・タイプの値を戻す必要があります。数値または日時引数は、関数を評価する前に文字ストリングにキャストされます。

trim-expression を指定しない場合、使用するデフォルト値は、*string-expression* のデータ・タイプによって以下のように決まります。

- 16 進数のゼロ (X'00') (引数がバイナリー・ストリングの場合)
- DBCS のブランク (引数が DBCS グラフィック・ストリングの場合)
- UTF-16 または UCS-2 のブランク (最初の引数が Unicode グラフィック・ストリング・ストリングの場合)
- UTF-8 のブランク (最初の引数が UTF-8 文字ストリングの場合)
- それ以外の場合は、SBCS のブランク。

string-expression の値と *trim-expression* の値は、互換性のあるデータ・タイプである必要があります。データ・タイプの互換性についての詳細は、 113 ページの『割り当ておよび比較』を参照してください。 *string-expression* と *trim-expression* の CCSID が異なる場合、*trim-expression* の CCSID は *string-expression* の CCSID に変換されます。

結果のデータ・タイプは、ストリング式 のデータ・タイプによって異なります。

string-expression のデータ・タイプ	結果のデータ・タイプ
CHAR または VARCHAR	VARCHAR
CLOB	CLOB
GRAPHIC または VARGRAPHIC	VARGRAPHIC
DBCLOB	DBCLOB
BINARY または VARBINARY	VARBINARY

73. RTRIM 関数は STRIP(*expression*,TRAILING) と同じ結果を返します。

string-expression のデータ・タイプ	結果のデータ・タイプ
BLOB	BLOB

結果の長さ属性は *string-expression* の長さ属性と同じになります。文字ストリングまたはバイナリー・ストリングの場合、結果の実際の長さは、除去されるバイト数を *string-expression* から引いた長さになります。結果がグラフィック・ストリングである場合の実際の長さは、除去されるグラフィック文字数を *string-expression* の長さから引いた値になります。すべての文字が除去された場合は、結果は空のストリングになります。

引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

結果の CCSID は*string-expression* の CCSID と同じです。

例

- 例 *1:* ホスト変数 HELLO が CHAR(9) として定義されており、値が以下であるときに、RTRIM 関数を使用します。

  ```
  'Hello    '
  ```

  ```
  VALUES RTRIM(:HELLO)
  ```

  結果は 'Hello' になります。*trim-expression* が指定されていないときは、ブランクのみが除去されます。

- 例 *2:* RTRIM 関数を使用して、*trim-expression* 内の個々の数値を *string-expression* の末尾 (右側) から除去します。

  ```
  SELECT RTRIM ('123DEFG123', '321'),
      RTRIM ('12322XYZ12322222', '123'),
      RTRIM ('12321', '213'),
      RTRIM ('123XYX', '321')
    FROM SYSIBM.SYSDUMMY1
  ```

  結果は、次のとおりです。

  ```
  '123DEFG'
  '12322XYZ'
  '' (empty string - all characters removed)
  '123XYX' (no characters removed)
  ```

  RTRIM 関数は、「1」、「2」、または「3」ではない文字の前にある、ストリング左側の「1」、「2」、および「3」のインスタンスを除去しません。

- 例 *3:* RTRIM 関数を使用して、*trim-expression* に指定した文字を *string-expression* の末尾から除去します。

  ```
  VALUES RTRIM('...$VAR$...', '$.')
  ```

  結果は、'...$VAR' になります。

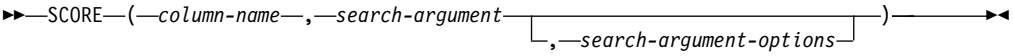
- 例 *4:* RTRIM 関数を使用して、*trim-expression* に指定した文字を *string-expression* の末尾から除去します。

  ```
  VALUES RTRIM('((-78.0) )', '-0. ()')
  ```

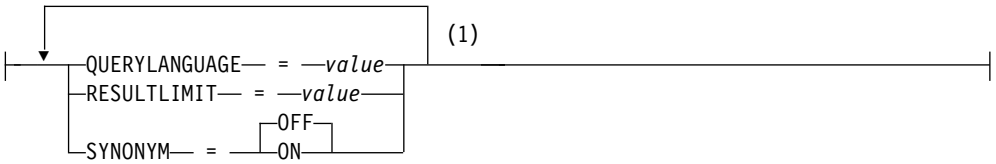
  結果は、'((-78' になります。文字とブランクを除去するときは、*trim-expression* にブランクを含める必要があります。

SCORE

SCORE 関数は、検索引数で指定した基準を使用してテキスト検索索引を検索し、文書が照会内容に一致する割合を測定した関連度スコアを返します。



search-argument-options:



注:

1 同じ文節を複数回指定することはできません。

column-name
検索するテキスト検索索引がある列の修飾名または非修飾名を指定します。この列は、ステートメントの FROM 文節で指定されている表またはビューになければならず、表の列、またはビューの基礎となる基本表の列には関連付けられたテキスト検索索引がなければなりません。ビューの列の基礎となる式は、基礎表の列に対して、直接または他のネストされたビューを介して参照する単純な列参照であることが必要です。

search-argument
検索する用語が含まれる、文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプを戻す式です。空ストリングであったり、すべてを空白にしたりすることはできません。このストリングの実際の長さが Unicode への変換後に 32,740 バイトを超えてはならず、検索引数構文で指定されたテキスト検索限界や用語数を超えてもなりません。*search-argument* 構文については、 2187 ページの『付録 G. テキスト検索引数の構文』を参照してください。

search-argument-options
検索に使用する検索引数オプションを指定する文字ストリング値またはグラフィック・ストリング値。定数または変数でなければなりません。

search-argument-options の一部として指定できるオプションは、以下のとおりです。

QUERYLANGUAGE = *value*
言語値を指定します。この値は、サポートされている任意の言語コードにすることできます。QUERYLANGUAGE を指定しないと、この関数の呼び出し時に使用されるテキスト検索索引の言語値がデフォルトとなります。テキスト検索索引の言語値が AUTO の場合、QUERYLANGUAGE のデフォルト値は en_US です。照会言語オプションについては、 2197 ページの『テキスト検索の言語オプション』を参照してください。

RESULTLIMIT = *value*
基礎となる検索エンジンから戻される結果の最大数を指定します。*value* は、1 から 2,147,483,647 の整数でなければなりません。RESULTLIMIT を指定しないと、照会での結果に制限がなくなります。

オプティマイザーが選択するプランによって、結果表の各行に対して SCORE が呼び出されることもあれば、呼び出されない場合もあります。基礎となる検索エンジンに対する照会で SCORE がいったん呼び出されると、検索エンジンから、一致する ROWID または主キーすべての結果セットが戻されます。その後、この結果セットはそうした結果行を識別する列が含まれる表に結合されます。この場合、RESULTLIMIT 値は基礎となるテキスト検索エンジンからの FETCH FIRST *n* ROWS ONLY のような動作になり、最適化のために使用できます。オプティマイザーが最適なプランであると判断し、SCORE が結果の各行に対して呼び出される場合には、RESULTLIMIT オプションは無効です。

SYNONYM = OFF または **SYNONYM = ON**
テキスト検索索引に関連付けられている同義語ディクショナリーを使用するかどうかを指定します。デフォルトは OFF です。

オフ
同義語ディクショナリーを使用しません。

ON テキスト検索索引に関連付けられている同義語ディクショナリーを使用します。

search-argument-options が空ストリングまたは NULL 値の場合、この関数は *search-argument-options* が指定されていない場合のように評価されます。

この関数の結果は、倍精度浮動小数点数になります。 *search-argument* を NULL にできる場合には、結果が NULL になる可能性があります。*search-argument* が NULL の場合、結果も NULL 値になります。

SCORE の結果は、0 から 1 までの値です。*search-argument* で指定した検索基準に一致する項目が対象列に数多く含まれているほど、結果値も大きくなります。一致項目が見つからないと、結果は 0 になります。この列値が NULL の場合、または *search-argument* に空白しか含まれていないか空ストリングである場合には、結果は 0 です。

SCORE は非決定性関数です。

注

前提条件**:** CONTAINS および SCORE 関数を使用するためには、OmniFind Text Search Server for Db2 for i がインストールされ、開始されている必要があります。

規則**:** ビュー、ネストされた表の式、または共通表式で、スカラー関数 CONTAINS または SCORE の対象になるテキスト検索列を用意する場合に、その該当するビュー、ネストされた表の式、共通表式の最外部の SELECT ステートメントで DISTINCT 文節を使用するのであれば、SELECT リストに、テキスト検索索引のすべての対応するキー・フィールドを組み込む必要があります。

ビュー、ネストされた表の式、または共通表式で、スカラー関数 CONTAINS または SCORE の対象になるテキスト検索列を用意する場合は、その該当するビュー、ネストされた表の式、または共通表式の最外部の SELECT で、UNION、EXCEPT、INTERSECT を使用できません。

共通表式で、スカラー関数 CONTAINS または SCORE の対象になるテキスト検索列を用意する場合に、その後、その共通表式を照会全体の中で再度参照できるのは、その参照によって、スカラー関数 CONTAINS または SCORE の対象になるテキスト検索列を用意しない場合に限られます。

CONTAINS および SCORE スカラー関数は、照会が以下を指定する場合には使用できません。

- 分散表
- 読み取りトリガーを指定する表
- 複数の物理ファイル・メンバー上に構築された論理ファイル

例

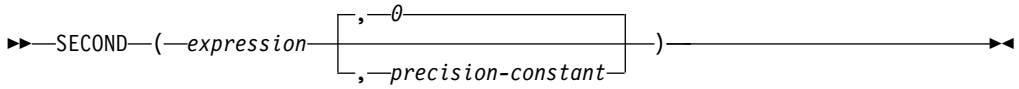
- 照会「programmer AND (java OR cobol)」に一致する経歴を持つ従業員を、一致する割合の順に配列し、0 から 100 の間の正規化された関連性値とともに示すリストを生成するステートメントを以下に示します。

```
SELECT EMPNO, INTEGER(SCORE(RESUME, 'programmer AND
  (java OR cobol)') * 100) AS RELEVANCE
  FROM EMP_RESUME
  WHERE RESUME_FORMAT = 'ascii'
  AND CONTAINS(RESUME, 'programmer AND (java OR cobol)') = 1
  ORDER BY RELEVANCE DESC
```

  データベース・マネージャーは、最初に WHERE 文節の CONTAINS 述部を評価するため、表の各行に対して SELECT リストの SCORE 関数が評価されるわけではありません。この場合、SCORE と CONTAINS の引数は同じでなければなりません。

SECOND

SECOND 関数は、値の秒の部分と、オプションで小数秒を戻します。



expression
日付、時刻、タイム・スタンプ、文字ストリング、グラフィック・ストリング、または数値のいずれかの組み込みデータ・タイプの値を戻す式。

- 式 が文字ストリングまたはグラフィック・ストリングである場合、その値は、日時値の有効なストリング表現でなければなりません。式 がタイム・スタンプの有効なストリング表現である場合、まず最初に TIMESTAMP(12) 値に変換されます。式が日付の有効なストリング表現の場合、その形式は IBM SQL 標準形式のいずれかでなければなりません。 日時値のストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。
- 引数が DATE である場合、時刻が午前 0 時ちょうど (00.00.00) であると想定して、最初に TIMESTAMP(0) 値に変換されます。
- 式 が数値である場合は、その数値は時刻期間またはタイム・スタンプ期間でなければなりません。日時期間の有効な形式については、 205 ページの『日付/時刻のオペランドと期間』を参照してください。

precision-constant
小数秒の桁数を表す整数定数。値の範囲は 0 から 12 です。

引数が 1 つの場合のこの関数の結果は、長精度整数です。引数が 2 つの場合のこの関数の結果は、DECIMAL(2+*s*,*s*) です。ここで、*s* は *precision-constant* の値です。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

その他の規則は、引数のデータ・タイプに応じて以下のように異なります。

- 引数が日付、時刻、またはタイム・スタンプであるか、または、日付、時刻、またはタイム・スタンプの有効な文字ストリングである場合:

  引数が 1 つだけ指定されている場合、結果は値の秒の部分 (0 から 59) になります。

  両方の引数が指定されている場合、結果は、値の秒の部分 (0 から 59) と、桁数が *precision-constant* の小数秒の部分とになります。値に小数秒がない場合、ゼロが戻されます。
- 引数が時刻期間またはタイム・スタンプ期間の場合 :

  引数が 1 つだけ指定されている場合、結果は値の秒の部分 (-99 から 99) になります。ゼロ以外の結果の符号は、引数と同じになります。

  両方の引数が指定されている場合、結果は、値の秒の部分 (-99 から 99) と、桁数が *precision-constant* の小数秒の部分とになります。値に小数秒がない場合、ゼロが戻されます。ゼロ以外の結果の符号は、引数と同じになります。

例

- ホスト変数 TIME_DUR (DECIMAL(6,0)) は、値が 153045 であると想定します。

```
SELECT SECOND(:TIME_DUR)
  FROM SYSIBM.SYSDUMMY1
```

  値として 45 が戻されます。

- 列 RECEIVED (TIMESTAMP) には、1988-12-25-17.12.30.000000 に相当する内部値が入っているものと想定します。

```
SELECT SECOND(RECEIVED)
  FROM IN_TRAY
```

  値として 30 が戻されます。

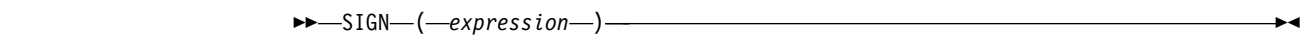
- ミリ秒単位の現在のタイム・スタンプから、小数秒を持つ秒を取得します。

```
SELECT SECOND(CURRENT TIMESTAMP(3),3)
  FROM SYSIBM.SYSDUMMY1
```

  現在のタイム・スタンプ (54.321 など) に基づいて DECIMAL(5,3) 値を戻します。

SIGN

SIGN 関数は式の符号の標識を戻します。



戻り値は以下のとおりです。

–1 引数がゼロ未満の場合

-0 引数が DECFLOAT の負のゼロである場合

0 引数がゼロの場合

1 引数がゼロより大きい場合

expression
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点に変換されます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

結果のデータ・タイプと長さ属性は引数と同じになります。ただし、引数が DECIMAL または NUMERIC で、引数の位取りが精度と同じである場合は、結果の精度は 1 だけ増やされます。例えば、データ・タイプが DECIMAL(5,5) の引数の場合、結果は DECIMAL(6,5) になります。精度が既に最大精度 (*mp*) の場合は、位取りが 1 下がります。例えば、DECIMAL(63,63) の場合、結果は DECIMAL(63,62) になります。

引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

- ホスト変数 PROFIT は、値が 50000 の長整数であると想定します。

```
SELECT SIGN(:PROFIT)
  FROM EMPLOYEE
```

値として 1 が戻されます。

SIN

SIN 関数は引数のサイン (正弦) を戻すもので、引数はラジアンで表された角度です。 SIN 関数と ASIN 関数は、逆の演算になります。

expression
任意の組み込み数値データ・タイプ (DECFLOAT を除く)、文字ストリングまたはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点に変換されます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

- ホスト変数 SINE は、値が 1.5 の 10 進数 (2,1) のホスト変数であると想定します。

```
SELECT SIN(:SINE)
  FROM SYSIBM.SYSDUMMY1
```

およそ 0.99 の値が戻されます。

SINH

SINH 関数は引数の双曲線サイン (双曲線正弦) を戻すもので、引数はラジアンで表された角度です。

expression
任意の組み込み数値データ・タイプ (DECFLOAT を除く)、文字ストリングまたはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点に変換されます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

- ホスト変数 HSINE は、値が 1.5 の 10 進数 (2,1) のホスト変数であると想定します。

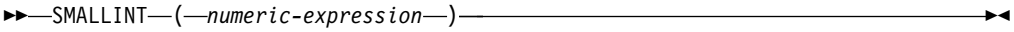
```
SELECT SINH(:HSINE)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 2.12 の値が戻されます。

SMALLINT

SMALLINT 関数は、短整数表現を返します。

数値から短整数に



ストリングから短整数に

SMALLINT 関数は、次のものの短整数表現を戻します。

- 数値
- 10 進数の文字ストリング表現またはグラフィック・ストリング表現
- 整数の文字ストリング表現またはグラフィック・ストリング表現
- 浮動小数点数の文字ストリング表現またはグラフィック・ストリング表現
- 10 進浮動小数点数の文字ストリング表現またはグラフィック・ストリング表現

数値から短整数に

numeric-expression
任意の組み込み数値データ・タイプの数値を戻す式。

結果は、引数が短整数の列または変数に割り当てられたときに得られる数値と同じです。引数の整数部が、短整数の範囲内にない場合は、エラーが戻されます。引数の小数部は切り捨てられます。

ストリングから短整数に

string-expression
数値の文字ストリング表現またはグラフィック・ストリング表現の値を戻す式。

引数がストリング式 の場合、結果は、 CAST(ストリング式 AS SMALLINT で得られる数値と同じです。先行ブランクと末尾ブランクは除去され、結果のストリングは、浮動小数点数、10 進浮動小数点数、整数、または 10 進数の定数を形成する際の規則に合致している必要があります。引数の整数部が、短整数の範囲内にない場合は、エラーが戻されます。引数の小数部は切り捨てられます。

この関数の結果は、短整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL である場合は、結果は NULL 値です。

注記

代替構文: アプリケーションの移植性を拡張するには、CAST 指定を使用します。
詳しくは、 218 ページの『CAST の指定』を参照してください。

例

- 表 EMPLOYEE を使用して、給与 (SALARY) を教育レベル (EDLEVEL) で除算した値が入っているリストを選択します。計算で生じた小数部は、すべて切り捨てられます。このリストには、計算で使用した値と従業員番号 (EMPNO) も入れておきます。

```
SELECT SMALLINT(SALARY / EDLEVEL), SALARY, EDLEVEL, EMPNO
  FROM EMPLOYEE
```

SOUNDEX

SOUNDEX 関数は、引数内のワードの音を表す 4 文字コードを戻します。この結果は、他のストリングの音と比較するのに使用できます。

expression
CLOB または DBCLOB 以外の任意の組み込み数値またはストリング・データ・タイプの値を戻す式。引数は 2 進ストリングであってはなりません。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。

結果のデータ・タイプは、CHAR(4) です。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

結果の CCSID は、現行サーバーのデフォルトの CCSID になります。

SOUNDEX 関数は、音は分かっているが正確なスペルが分からないストリングを見つけるのに便利です。これは、文字や文字の組み合わせの音に関する想定をして、類似の音を持つワードを検索するのに役立てます。比較は、直接行うことも、ストリングを DIFFERENCE 関数への引数として渡して行うこともできます。詳しくは、 437 ページの『DIFFERENCE』を参照してください。

例

- EMPLOYEE 表を使用して、姓 が「Loucesy」のような音をもつ従業員の EMPNO と LASTNAME を検索します。

```
SELECT EMPNO, LASTNAME
       FROM EMPLOYEE
       WHERE SOUNDEX(LASTNAME) = SOUNDEX('Loucesy')
```

以下の行が戻されます。

```
000110 LUCCHESSI
```

SPACE

SPACE 関数は、引数で指定された SBCS ブランク数からなる文字ストリングを戻します。

expression

組み込み SMALLINT、INTEGER、BIGINT、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に整数に変換されます。ストリングを整数に変換する方法については、496 ページの『INTEGER または INT』を参照してください。

式 は、結果の SBCS ブランクの数を示し、0 から 32740 でなければなりません。式 が定数の場合、定数 0 であってはなりません。

この関数の結果は、SBCS データを含む可変長文字ストリング (VARCHAR) になります。

式 が定数の場合、結果の長さ属性は定数です。それ以外の場合、結果の長さ属性は 4000 です。結果の実際の長さは、式 の値です。結果の実際の長さは、結果の長さ属性を超えてはなりません。

引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

CCSID は、そのジョブの SBCS データのデフォルト CCSID です。

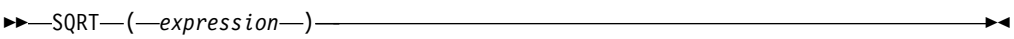
例

- 次のステートメントは、5 つのブランクからなる文字ストリングを戻します。

```
SELECT SPACE(5)
  FROM SYSIBM.SYSDUMMY1
```

SQRT

SQRT 関数は、数値の平方根を戻します。



expression
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点に変換されます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。式 の値は、ゼロまたはそれより大きい値でなければなりません。

引数のデータ・タイプが DECFLOAT(*n*) の場合、結果は DECFLOAT(*n*) です。それ以外の場合、結果のデータ・タイプは、倍精度の浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

注記

DECFLOAT 特殊値が関係する場合の結果: 10 進浮動小数点の場合、特殊値は次のように扱われます。

- SQRT(NaN) は NaN を返します。[74]
- SQRT(-NaN) は NaN を返します。[74]
- SQRT(Infinity) は Infinity を返します。
- SQRT(-Infinity) は NaN を返します。[74]
- SQRT(sNaN) および SQRT(-sNaN) は警告またはエラーを返します。[61]

例

- ホスト変数 SQUARE は、値が 9.0 の DECIMAL(2,1) のホスト変数であると想定します。

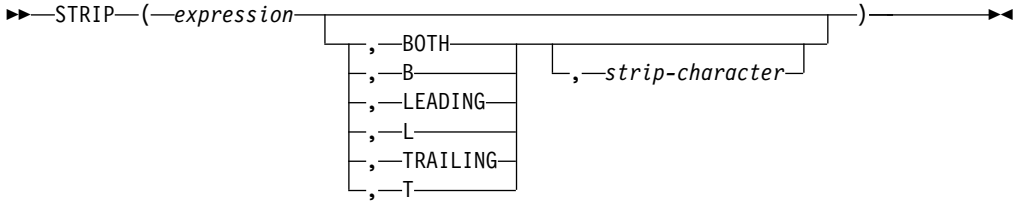
```
SELECT SQRT(:SQUARE)
  FROM SYSIBM.SYSDUMMY1
```

およそ 3.00 の値が戻されます。

74. SQL_DECFLOAT_WARNINGS 照会オプションに *YES を指定すると、NaN が返され、警告が出されます。

STRIP

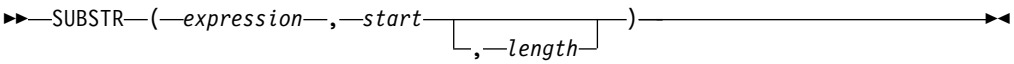
STRIP 関数はブランクまたは指定されたその他の文字を、ストリング式の末尾、先頭、またはその両側から除去します。



STRIP 関数は、TRIM スカラー関数と同等です。詳しくは、 660 ページの『TRIM』を参照してください。

SUBSTR

SUBSTR 関数は、ストリングのサブストリングを戻します。



expression
結果が導き出される元になるストリングを指定する式。

式 には、任意の組み込み数値またはストリング・データ・タイプを指定できます。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。式 が文字ストリングの場合は、この関数の結果は文字ストリングになります。ストリング式がグラフィック・ストリングの場合は、関数の結果はグラフィック・ストリングになります。ストリング式が 2 進ストリングの場合は、関数の結果は 2 進ストリングになります。

式 のサブストリングは、式 のゼロ個以上の連続したバイトです。式 がグラフィック・ストリングの場合、文字は DBCS または Unicode グラフィック文字です。式 が文字ストリングである場合、1 文字は 1 バイトです。[75] 式 が 2 進ストリングである場合、1 文字は 1 バイトです。

start
式 の中の、結果の最初の文字 (またはバイト) の位置を指定する式。式は、BIGINT、INTEGER、または SMALLINT の組み込みデータ・タイプである値を戻す必要があります。値 1 は、結果の最初の文字が *expression* の最初の文字になることを示します。負の値またはゼロは、そのストリングが始まるより前の位置を示します。また、式 の長さ属性より大きくても構いません。 (可変長ストリングの長さ属性は、そのストリングの最大長です。)

length
結果の長さを指定する式。指定する場合、長さ は、BIGINT、INTEGER、または SMALLINT の組み込みデータ・タイプの値を戻す式でなければなりません。値は、0 以上にする必要があります。

長さ を明示的に指定した場合、式 の指定されたサブストリングが必ず存在するように、実際上は 式 の右側に必要な数のブランク文字が埋め込まれます。式 が 2 進ストリングの場合は、16 進数のゼロが埋め込み文字として使用されます。

式 が固定長ストリングの場合は、長さ を省略すると、 LENGTH(式) - 開始桁 + 1 (式 の 開始 文字 (またはバイト) から最終文字 (またはバイト) までの文字数) が暗黙指定されます。式 が可変長ストリングの場合に、長さ の指定を省略すると、 0 と LENGTH(式) - 開始桁 + 1 のいずれか大きい方が、暗黙の長さの指定として使用されます。結果の長さがゼロの場合は、結果は空ストリングになります。

結果のデータ・タイプは、式 のデータ・タイプによって異なります。

---

75. SUBSTR 関数は混合データ・ストリングを受け入れます。ただし、SUBSTR は厳密なバイト・カウントに基づいて演算を行うため、結果は必ずしも適切な形式の混合データ・ストリングにはなりません。

式 のデータ・タイプ	**SUBSTR** の場合の結果のデータ・タイプ
CHAR または VARCHAR	次の場合は CHAR • 長さ がゼロより大きい整数定数で明示的に指定されている。 • 長さ は明示的に指定されていないが、式 が固定長ストリングであり、開始桁 が整数定数である。 VARCHAR (それ以外のすべての場合)
CLOB	CLOB
GRAPHIC または VARGRAPHIC	次の場合は GRAPHIC • 長さ がゼロより大きい整数定数で明示的に指定されている。 • 長さ は明示的に指定されていないが、式 が固定長ストリングであり、開始桁 が整数定数である。 VARGRAPHIC (それ以外のすべての場合)。
DBCLOB	DBCLOB
BINARY または VARBINARY	次の場合は BINARY • 長さ がゼロより大きい整数定数で明示的に指定されている。 • 長さ は明示的に指定されていないが、式 が固定長ストリングであり、開始桁 が整数定数である。 VARBINARY (それ以外のすべての場合)。
BLOB	BLOB

式 が LOB でない場合、結果の長さ属性は、長さ、開始桁、および式 の属性によって決まります。

- 長さ をゼロより大きい整数定数で明示的に指定すると、結果の長さ属性は長さになります。
- 長さ は明示的に指定されていないが、式 が固定長ストリングであり、開始桁 が整数定数である場合は、結果の長さ属性が LENGTH(式) - 開始桁 + 1 になります。

それ以外のすべての場合、結果の長さ属性は式 の長さ属性と同じになります。 (式の実際の長さが開始桁 の値より小さい場合は、サブストリングの実際の長さはゼロになります。)

引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

結果の CCSID は式 の CCSID と同じです。

例

- ホスト変数 NAME (VARCHAR(50)) の値は 'KATIE AUSTIN' で、ホスト変数 SURNAME_POS (INTEGER) の値は 7 であると想定します。

```
SELECT SUBSTR(:NAME, :SURNAME_POS)
  FROM SYSIBM.SYSDUMMY1
```

値 'AUSTIN' が戻されます。

- 同様に、

```
SELECT SUBSTR(:NAME, :SURNAME_POS, 1)
  FROM SYSIBM.SYSDUMMY1
```

値 'A' が戻されます。

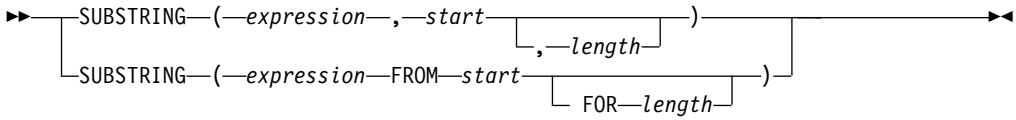
- PROJECT 表から、語 'OPERATION' で始まるプロジェクト名 (PROJNAME) の行を全選択します。

```
SELECT *
  FROM PROJECT
  WHERE SUBSTR(PROJNAME,1,10) = 'OPERATION '
```

定数の最後にあるスペースは、 'OPERATIONS' などの語で始まるものを除外するために必要です。

SUBSTRING

SUBSTRING 関数は、ストリングのサブストリングを戻します。



expression
結果が導き出される元になるストリングを指定する式。

式 には、任意の組み込み数値またはストリング・データ・タイプを指定できます。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。式 が文字ストリングの場合は、この関数の結果は文字ストリングになります。ストリング式がグラフィック・ストリングの場合は、関数の結果はグラフィック・ストリングになります。ストリング式が 2 進ストリングの場合は、関数の結果は 2 進ストリングになります。

式 のサブストリングは、式 のゼロ個以上の連続したバイトです。式 がグラフィック・ストリングの場合、文字は DBCS または Unicode グラフィック文字です。式 が文字ストリングである場合、1 文字は 1 バイト以上の文字です。式が 2 進ストリングである場合、1 文字は 1 バイトです。

start
式 の中の、結果の最初の文字 (またはバイト) の位置を指定する式。式は、BIGINT、INTEGER、または SMALLINT の組み込みデータ・タイプである値を戻す必要があります。値 1 は、結果の最初の文字が *expression* の最初の文字になることを示します。負の値またはゼロは、そのストリングが始まるより前の位置を示します。また、式 の長さ属性より大きくても構いません。 (可変長ストリングの長さ属性は、そのストリングの最大長です。)

length
結果のサブストリングの実際の最大長を指定する式。指定する場合、長さ は、BIGINT、INTEGER、または SMALLINT の組み込みデータ・タイプの値を戻す式でなければなりません。値は、0 以上にする必要があります。

長さ を明示的に指定した場合、埋め込みは行われません。

式 が固定長ストリングの場合は、長さ を省略すると、 LENGTH(式) - 開始桁 + 1 (式 の 開始 文字 (またはバイト) から最終文字 (またはバイト) までの文字数) が暗黙指定されます。式 が可変長ストリングの場合に、長さ の指定を省略すると、 0 と LENGTH(式) - 開始桁 + 1 のいずれか大きい方が、暗黙の長さの指定として使用されます。結果の長さがゼロの場合は、結果は空ストリングになります。

結果のデータ・タイプは、式 のデータ・タイプによって異なります。

式 のデータ・タイプ	**SUBSTRING** の場合の結果のデータ・タイプ
CHAR または VARCHAR	VARCHAR
CLOB	CLOB

式 のデータ・タイプ	**SUBSTRING** の場合の結果のデータ・タイプ
GRAPHIC または VARGRAPHIC	VARGRAPHIC
DBCLOB	DBCLOB
BINARY または VARBINARY	VARBINARY
BLOB	BLOB

結果の長さ属性は、式 の長さ属性と同じになります。 (式 の実際の長さが開始桁の値より小さい場合は、サブストリングの実際の長さはゼロになります。)

引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

結果の CCSID は式 の CCSID と同じです。

例

- PROJECT 表から、語 'OPERATION' で始まるプロジェクト名 (PROJNAME) の行を全選択します。

```
SELECT *
  FROM PROJECT
  WHERE SUBSTRING(PROJNAME,1,10) = 'OPERATION '
```

  定数の最後にあるスペースは、 'OPERATIONS' などの語で始まるものを除外するために必要です。

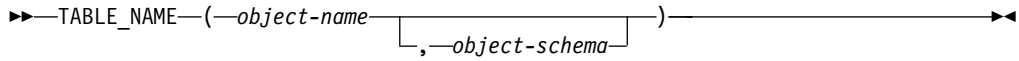
- FIRSTNAME は T1 の VARCHAR(12) の列であり、Unicode UTF-8 でエンコードされるものと仮定します。その値の 1 つが 6 文字ストリング 'Jürgen' です。これが FIRSTNAME の値の場合、

```
SELECT SUBSTRING(FIRSTNAME, 1,2), SUBSTR(FIRSTNAME, 1,2)
  FROM T1
```

  値 'Jü' (x'4AC3BC') と 'Jô' (x'4AC3') が戻ります。

TABLE_NAME

TABLE_NAME 関数は、別名に対して検出されたオブジェクトの非修飾名を戻します。



指定された *object-name* (および *object-schema*) が、その名前で別名を検出するために使用されます。

object-name
解決するオブジェクトの SQL 名またはシステム名を示す文字ストリング式またはグラフィック・ストリング式。*object-name* の実際の長さは 129 文字未満でなければならず、ブランク名を指定することはできません。この名前には大/小文字の区別があり、引用符で区切られていてはなりません。

object-schema
object-name を修飾するために使用されるスキーマの SQL 名またはシステム名を示す文字ストリング式またはグラフィック・ストリング式。*object-schema* の実際の長さは 129 文字未満でなければならず、ブランク名を指定することはできません。この名前には大/小文字の区別があり、引用符で区切られていてはなりません。

object-schema を指定しない場合は、修飾子にデフォルトのスキーマが使用されます。

関数の結果は VARCHAR(128) です。 *object-name* が NULL になる可能性がある場合は、結果も NULL になる可能性があります。 *object-name* が NULL であれば、結果も NULL 値になります。 *object-schema* が NULL 値の場合は、デフォルトのスキーマ名が使用されます。結果は、非修飾名を表す文字ストリングになります。

結果の名前は、別名が参照する表名またはビュー名です。*object-name* が別名ではない場合、*object-name* が戻されます。

例

以下のように作成された別名で参照される表の名前を取得します。

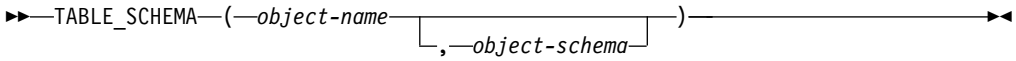
```
CREATE ALIAS MYLIB2.ALIAS1 FOR MYLIB.EMPLOYEE


VALUES TABLE_NAME('ALIAS1', 'MYLIB2')
```

結果は、次のとおりです。

```
EMPLOYEE
```

TABLE_SCHEMA

TABLE_SCHEMA 関数は、別名に対して検出されたオブジェクトのスキーマ名を戻します。



指定された *object-name* (および *object-schema*) が、その名前で別名を検出するために使用されます。

object-name
解決するオブジェクトの SQL 名またはシステム名を示す文字ストリング式またはグラフィック・ストリング式。*object-name* の実際の長さは 129 文字未満でなければならず、ブランク名を指定することはできません。この名前には大/小文字の区別があり、引用符で区切られていてはなりません。

object-schema
object-name を修飾するために使用されるスキーマの SQL 名またはシステム名を示す文字ストリング式またはグラフィック・ストリング式。*object-schema* の実際の長さは 129 文字未満でなければならず、ブランク名を指定することはできません。この名前には大/小文字の区別があり、引用符で区切られていてはなりません。

object-schema を指定しない場合は、修飾子にデフォルトのスキーマが使用されます。

関数の結果は VARCHAR(128) です。 *object-name* が NULL になる可能性がある場合は、結果も NULL になる可能性があります。 *object-name* が NULL であれば、結果も NULL 値になります。 *object-schema* が NULL 値の場合は、デフォルトのスキーマ名が使用されます。結果は、スキーマ名を表す文字ストリングになります。

結果の名前は、別名が参照する表またはビューのスキーマ名です。*object-name* が別名ではない場合、*object-schema* が戻されます。

例

以下のように作成された別名で参照されるスキーマの名前を取得します。

```
CREATE ALIAS MYLIB2.ALIAS1 FOR MYLIB.EMPLOYEE


VALUES TABLE_SCHEMA('ALIAS1', 'MYLIB2')
```

結果は、次のとおりです。

```
MYLIB
```

TAN

TAN 関数は引数のタンジェント (正接) を戻すもので、引数はラジアンで表された角度です。 TAN 関数と ATAN 関数は、逆の演算になります。

expression
任意の組み込み数値データ・タイプ (DECFLOAT を除く)、文字ストリングまたはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点に変換されます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

- ホスト変数 TANGENT は、値が 1.5 の DECIMAL(2,1) のホスト変数であると想定します。

```
SELECT TAN(:TANGENT)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 14.10 の値が戻されます。

TANH

TANH 関数は引数の双曲線タンジェント (双曲線正接) を戻すもので、引数はラジアンで表された角度です。 TANH 関数と ATANH 関数は、逆の演算になります。

expression
任意の組み込み数値データ・タイプ (DECFLOAT を除く)、文字ストリングまたはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点に変換されます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

結果のデータ・タイプは、倍精度の浮動小数点数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

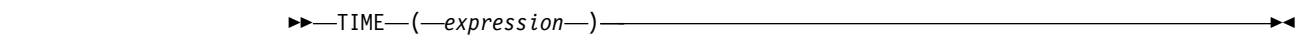
- ホスト変数 HTANGENT は、値が 1.5 の DECIMAL(2,1) のホスト変数であると想定します。

```
SELECT TANH(:HTANGENT)
  FROM SYSIBM.SYSDUMMY1
```

  およそ 0.90 の値が戻されます。

TIME

TIME 関数は、指定された値から時刻を戻します。



expression
日付、時刻、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式。

expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付、時刻、またはタイム・スタンプの有効なストリング表現でなければなりません。式が日付の有効なストリング表現の場合、その形式は IBM SQL 標準形式の 1 つでなければなりません。日付、時刻、およびタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

この関数の結果は、時刻になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

その他の規則は、引数のデータ・タイプに応じて以下のように異なります。

- 引数が日付の場合 :

  結果は、夜の 12 時になります。
- 引数が時刻の場合 :

  結果は指定した時刻になります。
- 引数がタイム・スタンプの場合 :

  結果は、タイム・スタンプの時刻の部分です。
- 引数が文字ストリングまたはグラフィック・ストリングの場合 :

  結果は、ストリングで表された時刻、またはストリングで表されたタイム・スタンプ値の時刻部分です。

注記

代替構文: アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、 218 ページの『CAST の指定』を参照してください。

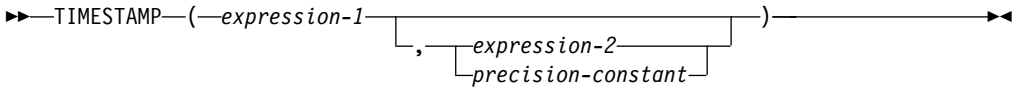
例

- サンプル表 IN_TRAY から、現在の時刻より 1 時間以上あと (日付は問わない) に受け取ったコメントをすべて選択します。

```
SELECT *
  FROM IN_TRAY
  WHERE TIME(RECEIVED) >= CURRENT TIME + 1 HOUR
```

TIMESTAMP

TIMESTAMP 関数は、1 つ以上の引数から導き出されるタイム・スタンプを戻します。



expression-1 および *expression-2*

これらの引数に関する規則は、 2 番目の引数が指定されるかどうかと、2 番目の引数のデータ・タイプによって異なります。

- *expression-1* のみを指定した場合:

  引数は、日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式であることが必要です。 *expression-1* が文字ストリングまたはグラフィック・ストリングの場合、値は以下のいずれかでなければなりません。

  – 日付またはタイム・スタンプの有効なストリング表現。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。
  – GENERATE_UNIQUE 関数からの結果と見なされる実際の長さが 13 の文字ストリング。GENERATE_UNIQUE については、 467 ページの『GENERATE_UNIQUE』 を参照してください。

- 引数を両方とも指定した場合:
  – 2 番目の引数が *expression-2* の場合:

    最初の引数は、日付、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式であることが必要です。*expression-1* が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付の有効なストリング表現でなければなりません。

    expression-2 は、時刻、文字ストリング、またはグラフィック・ストリングのうちのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。*expression-2* が文字ストリングまたはグラフィック・ストリングの場合、その値は、時刻の有効なストリング表現でなければなりません。日付と時刻のストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

  – 2 番目の引数が *precision-constant* の場合:

    最初の引数は、日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式でなければなりません。*expression-1* が文字ストリングまたはグラフィック・ストリングの場合、値は以下のいずれかでなければなりません。

    - 日付またはタイム・スタンプの有効なストリング表現。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

- GENERATE_UNIQUE 関数からの結果と見なされる実際の長さが 13 の文字ストリング。GENERATE_UNIQUE については、 467 ページの『GENERATE_UNIQUE』 を参照してください。

precision-constant
小数秒の桁数を表す整数定数。値の範囲は 0 から 12 です。

この関数の結果は、タイム・スタンプになります。引数のどちらかが NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数のどちらかが NULL である場合は、結果は NULL 値になります。

その他の規則も、2 番目の引数を指定するかどうかに応じて以下のように異なります。

- 引数を両方とも指定し、2 番目の引数が *expression-2* の場合:

  結果は、最初の引数によって日付が指定され、 2 番目の引数によって時刻が指定された TIMESTAMP(6) です。タイム・スタンプの秒未満の部分はゼロです。
- 引数を両方とも指定し、2 番目の引数が *precision-constant* の場合:

  結果は、 2 番目の引数で指定された精度のタイム・スタンプです。
- 引数が 1 つだけ指定され、それが TIMESTAMP(*p*) の場合

  結果は、指定した TIMESTAMP(*p*) になります。
- 引数が 1 つだけ指定され、それが DATE の場合

  結果は、その日付で、TIMESTAMP(0) にキャストされた真夜中の想定時刻となります。
- 引数が 1 つだけ指定され、それがストリングの場合

  結果は、そのストリングで表される TIMESTAMP(6) です。引数が長さ 14 のストリングの場合、タイム・スタンプの端数秒の部分はゼロになります。

注記

代替構文**:** 引数を 1 つだけ指定する場合、アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、 218 ページの『CAST の指定』を参照してください。

例

- 日付と時刻の値が以下のとおりであるとします。

```
SELECT TIMESTAMP( DATE('1988-12-25'), TIME('17.12.30') )
  FROM SYSIBM.SYSDUMMY1
```

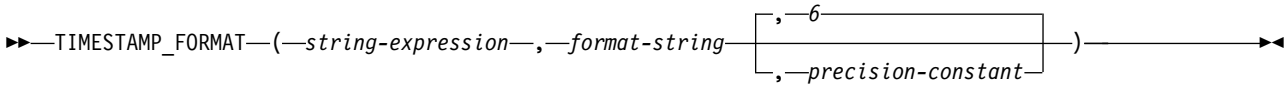
  値として、「1988-12-25-17.12.30.000000」が戻されます。
- 7 桁の秒未満タイム・スタンプ・ストリングを、TIMESTAMP(9) の値に変換します。

```
TIMESTAMP('2007-09-24-15.53.37.2162474', 9)
```

  この例では 「2007-09-24-15.53.37.216247400」の値を戻します。

TIMESTAMP_FORMAT

TIMESTAMP_FORMAT 関数は、指定された形式を使用した入力ストリングの解釈に基づくタイム・スタンプを戻します。



string-expression
組み込み文字ストリング・データ・タイプの値か、グラフィック・ストリング・データ・タイプを返す式。

ストリングは、*format-string* に指定された形式を使用する日付またはタイム・スタンプと解釈されます。*string-expression* には、*format-string* で指定したフォーマット・エレメントに対応する日付またはタイム・スタンプの構成要素のみが含まれている必要があります。

format-string
組み込み文字ストリング・データ・タイプか、グラフィック・ストリング・データ・タイプを戻す式。 *format-string* は、*string-expression* を日付またはタイム・スタンプ値としてどのように解釈するかを示すテンプレートを含みます。

有効な *format-string* には少なくとも 1 つのフォーマット・エレメントを含める必要があり、日付またはタイム・スタンプのすべての構成要素に複数の指定を含めてはならず、また 647 ページの表 55 に特に注記のない限り、フォーマット・エレメントの任意の組み合わせを含めることができます。例えば、*format-string* には YY と YYYY の両方を含めることはできません。これは、それらが *string-expression* の年の構成要素を解釈するためにともに使用されてしまうためです。下の表を参照して、どのフォーマット・エレメントを同時に指定できないかを確認してください。

2 つのフォーマット・エレメントは、オプションで以下の 1 つ以上の区切り文字で分離することができます。

- 負符号 (-)
- ピリオド (.)
- スラッシュ (/)
- コンマ (,)
- アポストロフィ (')
- セミコロン (;)
- コロン (:)
- ブランク ( )

区切り文字は *format-string* の先頭または末尾に指定することもできます。これらの区切り文字は、フォーマット・ストリングで任意に組み合わせて使用できます (例えば、「YYYY/MM-DD HH24:MM.SS」)。 *string-expression* で指定する区切り文字は構成要素を区切るために使用するものであり、*format-string* で指定する区切り文字と一致する必要はありません。

表 55. *TIMESTAMP_FORMAT* 関数のフォーマット・エレメント

フォーマット・エレメント	タイム・スタンプの関連する構成要素	説明
AM または PM [1、2]	時	ピリオドが付かない午前/午後の指定子。この午前/午後の指定子は、ライブラリー *LIBL の中のメッセージ・ファイル QCPFMSG 内のメッセージ CPX9035 から取り出されます。
A.M. または P.M. [1、2]	時	ピリオドが付いた午前/午後の指定子。このフォーマット・エレメントは、「A.M.」または「P.M.」と正確に一致するストリングを使用し、ジョブのメッセージに使用される言語には関係ありません。
DAY、Day、または day [1、3]	なし	大文字、タイトル文字、または小文字のフォーマットの曜日の名前。曜日の名前は、ライブラリー *LIBL の中のメッセージ・ファイル QCPFMSG のメッセージ CPX9034 から検索されます。
DY、Dy、または dy [1、3]	なし	大文字、タイトル文字、または小文字のフォーマットの曜日の省略名。省略された曜日の名前は、ライブラリー *LIBL の中のメッセージ・ファイル QCPFMSG のメッセージ CPX9039 から検索されます。
D [1、3]	なし	曜日 (1-7)。1 は日曜日です。
DD	日	日 (01 から 31)。
DDD	月、日	年間通算日 (001 から 366)。
FF または FF*n*	端数秒	小数秒 (0 から 999999999999)。数字 *n* は、*string-expression* 内に予想される桁数の指定に使用します。*n* の有効な値は 1 から 12 です。FF を指定することは、FF6 を指定するのと同等です。FF フォーマット・エレメントに対応する *string-expression* 内の構成要素の後に区切り文字が続いている場合、またはそれが最後の構成要素である場合、小数秒の桁数はフォーマット・エレメントで指定した桁数よりも少ないことがあります。この場合は、指定した桁数の右側に数字のゼロが埋め込まれます。
HH	時	HH の動作は HH12 と同様です。
HH12	時	12 時間形式の時 (01-12)。AM がデフォルトの午前/午後の指定子です。
HH24	時	24 時間形式の時刻 (00 から 24)。
J	年、月、および日	ユリウス日付 (紀元前 4713 年 1 月 1 日からの日数)。
MI	分	分 (00 から 59)。
MM	月	月 (01 から 12)。
MONTH、Month、または month [1]	月	大文字、タイトル文字、または小文字のフォーマットの月の名前。月の名前は、ライブラリー *LIBL の中のメッセージ・ファイル QCPFMSG のメッセージ CPX3BC0 から検索されます。

表 55. *TIMESTAMP_FORMAT* 関数のフォーマット・エレメント (続き)

フォーマット・エレメント	タイム・スタンプの関連する構成要素	説明
MON、Mon、または mon [1]	月	大文字、タイトル文字、または小文字のフォーマットの月の省略名。月の名前は、ライブラリー *LIBL の中のメッセージ・ファイル QCPFMSG のメッセージ CPX8601 から検索されます。
NNNNNN	マイクロ秒	マイクロ秒 (FF6 と同じ)。
RR [4]	年	調整される年の最後の 2 桁 (00 から 99)。
RRRR [4]	年	4 桁の調整済みの年 (0000 から 9999)。
SS	秒	秒 (00 から 59)。
SSSSS	時、分、および秒	前の深夜 12 時以降の秒数 (00000 から 86400)。
Y	年	年の最後の 1 桁 (0-9)。現在の年の最初の 3 桁が、完全な 4 桁の年の判別に使用されます。
YY	年	その年の最後の 2 桁 (00 から 99)。現在の年の最初の 2 桁が、完全な 4 桁の年の判別に使用されます。
YYY	年	年の最後の 3 桁 (000-999)。現在の年の最初の桁が、完全な 4 桁の年の判別に使用されます。
YYYY	年	4 桁の年 (0000-9999)。

注:

1. 正確なスペルと大文字小文字の組み合わせだけが使用できます。このフォーマット・エレメントが無効な大文字小文字の組み合わせで指定された場合、エラーが返されます。
2. 午前/午後の指定子として、A.M. と P.M. と同様に AM と PM のセットを *format-string* で使用できます。 *format-string* で午前/午後の指定子とともに HH24 が使用された場合、結果のタイム・スタンプの時間部分を決定するために、*string-expression* 内の午前/午後の指定子の値は使用されません。
3. DAY、Day、day、DY、Dy、dy、および D のフォーマット・エレメントは結果のタイム・スタンプのいずれかの構成要素に提供されません。ただし、これらのフォーマット・エレメントのいずれかに指定した値は、結果のタイム・スタンプの年、月、および日の構成要素の組み合わせに適切でなければなりません。例えば、*string-expression* の値 'Monday 2008-10-06' は、値 'Day YYYY-MM-DD' に対して有効です。ただし、*string-expression* への 'Tuesday 2008-10-06' という値は同じ *format-string* ではエラーになります。
4. RR と RRRR の各フォーマット・エレメントを使用すれば、次の表に従って現在の年の左端の 2 桁に基づいて 2 桁の値または 4 桁の値を生成するように値を調整することによって、年の指定の解釈を変更することができます。

現在の年の最後の **2** 桁	ストリング式 内の年の **2** 桁	日付またはタイム・スタンプの年コンポーネントの最初の **2** 桁
0-50	0-49	現在の年の最初の 2 桁
51-99	0-49	現在の年の最初の 2 桁 + 1
0-50	50-99	現在の年の最初の 2 桁 - 1
51-99	50-99	現在の年の最初の 2 桁

例えば、現在の年が 2007 の場合、フォーマット 'RR' の '86' は 1986 を意味しますが、現在の年が 2052 の場合は 2086 を意味します。

format-string にタイム・スタンプの以下の構成要素の 1 つに対してフォーマット・エレメントが含まれていない場合は、デフォルトが使用されます。

タイム・スタンプの構成要素	デフォルト
年	現在の年、4 桁
月	現在の月、2 桁
日	01 (現在の月の最初の日)
時	00
分	00
第 2	00
端数秒	結果のタイム・スタンプの精度に一致するゼロの数

string-expression に、*format-string* に指定された時、分、秒、または端数秒のフォーマット・エレメントに対応する値が含まれていない場合は、これらの同じデフォルトが使用されます。

日付またはタイム・スタンプ値のコンポーネント (月、日、時間、分、秒など) が *format-string* 内の対応するフォーマット・エレメントの最大有効桁数に達していない場合、そのコンポーネントに先行ゼロを指定することができます。

日付またはタイム・スタンプのコンポーネント (年、月、日、時間、分、秒など) を表す *string-expression* のサブストリングの桁数は、日付またはタイム・スタンプのそのコンポーネントの最大桁数より少なくてもかまいません。指定されていない桁は、デフォルトでゼロと解釈されます。例えば、フォーマット・ストリング 'YYYY-MM-DD HH24:MI:SS' では、入力した値が '999-3-9 5:7:2' の場合、'0999-03-09 05:07:02' と同じ結果が生成されます。

precision-constant
結果のタイム・スタンプ精度を指定する整数定数。値の範囲は 0 から 12 です。*precision-constant* が指定されていない場合、タイム・スタンプ精度はデフォルトで 6 になります。

結果は、*precision-constant* に基づいた精度を持つタイム・スタンプです。引数の最初の 2 つのいずれかが NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引数の最初の 2 つのいずれかが NULL 値の場合、その結果は NULL 値です。

注

ユリウス暦およびグレゴリオ暦: この関数では、1582 年 10 月 15 日のユリウス暦からグレゴリオ暦への移行が考慮されます。

代替構文: TO_DATE は TIMESTAMP_FORMAT の同義語です。*precision-constant* が指定されていない場合に結果のタイム・スタンプ精度がデフォルトで 12 になることを除き、TO_TIMESTAMP は TIMESTAMP_FORMAT と同じです。

例

- 2000 年が始まる 1 秒前 (1999 年 12 月 31 日 23 時 59 分 59 秒) にあたる受信タイム・スタンプで、IN_TRAY 表に行を挿入します。

```
INSERT INTO IN_TRAY (RECEIVED)
     VALUES (TIMESTAMP_FORMAT('1999-12-31 23:59:59',
       'YYYY-MM-DD HH24:MI:SS'))
```

- アプリケーションで、日付情報のストリングが INDATEVAR という変数に受け取られます。この値は厳密にはフォーマット設定されていなく、また年に 2 桁または 4 桁の数字、月と日に 1 桁または 2 桁の数字が含まれています。日付の各構成要素は負符号 (-) またはスラッシュ (/) 文字で分離され、日、月、および年の順序であることが予期されています。時間情報は、時 (24 時間形式) および分で構成され、通常コロンで分離されています。サンプル値としては、'15/12/98 13:48' や '9-3-2004 8:02' などがあります。このような値を IN_TRAY 表に挿入します。

```
INSERT INTO IN_TRAY (RECEIVED)
     VALUES (TIMESTAMP_FORMAT(:INDATEVAR,
       'DD/MM/RRRR HH24:MI'))
```

  フォーマットに RRRR を使用すると、2 桁と 4 桁の年の値が考慮され、現在の年に基づいて入力されていない最初の 2 桁が割り当てられます。 YYYY を使用すると、2 桁の年の入力値には先行ゼロが設定されます。また、スラッシュ区切り文字では負符号 (-) 文字も許可されます。現在の年が 2007 年とすると、サンプル値の結果として生成されるタイム・スタンプは以下のとおりです。

```
'15/12/98 13:48' --> 1998-12-15-13.48.00.000000
'9-3-2004 8:02'  --> 2004-03-09-08.02.00.000000
```

- 例: QSYS2.SYSPROCS の ROUTINE_CREATED の値が、2000 年の開始の 1 秒前 (「1999-12-31 23:59:59」) に等しい場合、文字変数 TVAR をその値に設定します。文字ストリングは、示されたフォーマット・ストリングに従って解釈してください。

```
SELECT VARCHAR_FORMAT(ROUTINE_CREATED, 'YYYY-MM-DD HH24:MI:SS')
     INTO :TVAR
     FROM QSYS2.SYSPROCS
     WHERE ROUTINE_CREATED =
               TIMESTAMP_FORMAT('1999-12-31 23:59:59', 'YYYY-MM-DD HH24:MI:SS')
```

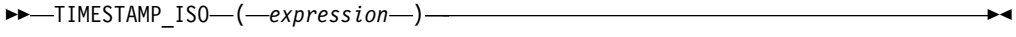
- 午前/午後の指定子を含むストリングのタイム・スタンプ値を戻します。

string-expression	*format-expression*	結果のタイム・スタンプ値
'2015-10-28 10:29AM'	'YYYY-MM-DD HH12:MIAM'	2015-10-28-10.29.00.000000
'2015-10-28 10:29PM'	'YYYY-MM-DD HH12:MIAM'	2015-10-28-22.29.00.000000

string-expression	*format-expression*	結果のタイム・スタンプ値
'2015-10-28 10:29AM'	'YYYY-MM-DD HH24:MIAM'	2015-10-28-10.29.00.000000
'2015-10-28 10:29PM'	'YYYY-MM-DD HH24:MIAM'	2015-10-28-10.29.00.000000
'2015-10-28 22:29AM'	'YYYY-MM-DD HH24:MIAM'	2015-10-28-22.29.00.000000
'2015-10-28 22:29PM'	'YYYY-MM-DD HH24:MIAM'	2015-10-28-22.29.00.000000

TIMESTAMP_ISO

日付、時刻、またはタイム・スタンプ引数に基づくタイム・スタンプ値を戻します。引数が日付の場合、タイム・スタンプの時刻の部分および小数秒の部分にはゼロが挿入されます。引数が時刻の場合、タイム・スタンプの日付の部分に CURRENT DATE の値が挿入され、タイム・スタンプの小数秒の部分にゼロが挿入されます。



expression
タイム・スタンプ、日付、時刻、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式。

expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

expression がタイム・スタンプの場合、この関数の結果は、*expression* と同じ精度のタイム・スタンプです。それ以外の場合、この関数の結果は TIMESTAMP(6) です。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

expression が時刻の場合、この関数は非決定論的です。

注記

代替構文: アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、 218 ページの『CAST の指定』を参照してください。

例

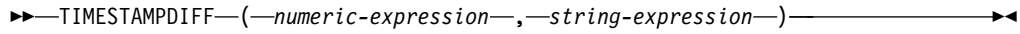
- 日付の値が以下のとおりであるとします。

```
SELECT TIMESTAMP_ISO( DATE( '1988-12-25' ) )
  FROM SYSIBM.SYSDUMMY1
```

「1988-12-25-00.00.00.000000」の値が戻されます。

TIMESTAMPDIFF

TIMESTAMPDIFF 関数は、2 つのタイム・スタンプの差に基づいて、最初の引数によって定義されたタイプの間隔の見積数を戻します。



numeric-expression

最初の引数は、INTEGER または SMALLINT のいずれかの組み込みデータ・タイプでなければなりません。この値は、2 つのタイム・スタンプの間の差を判別するのに使用する間隔を指定します。間隔の有効な値は、以下のとおりです。

表 *56. 2* つのタイム・スタンプの間の差を判別するのに使用する *numeric-expression* およびそれに相当する間隔に対する有効値

numeric-expression に対する有効値	相当する間隔
1	マイクロ秒
2	秒
4	分
8	時間
16	日
32	週
64	月
128	四半期
256	年

string-expression

ストリング式 は、2 つのタイム・スタンプを減算し、その結果を長さ 22 のストリングに変換したものです。 *string-expression* の小数点の右側が 6 桁を超える場合、ストリングは 6 桁になるよう切り捨てられます。 引数は、組み込み文字ストリングまたはグラフィック・ストリングの値を戻す式でなければなりません。

正または負の符号が付いている場合、それがそのストリングの最初の文字になります。下表は、文字ストリングによる期間を表すエレメントを説明しています。

表 *57. TIMESTAMPDIFF* ストリング・エレメント

ストリング・エレメント	有効値	小数点からの文字位置 **(**負は左方向**)**
年	1 から 9998 またはブランク	-14 から -11
月	0 から 11 またはブランク	-10 から -9
日	0 から 30 またはブランク	-8 から -7
時間	0 から 24 またはブランク	-6 から -5
分	0 から 59 またはブランク	-4 から -3
秒	0 から 59	-2 から -1
小数点	ピリオド	0
マイクロ秒	000000 から 999999	1 から 6

この関数の結果は、*string-expression* と同じ符号の整数になります。引数のどちらかが NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数のどちらかが NULL である場合は、結果は NULL 値になります。

戻り値は、それぞれの間隔に関して以下の表に示されているように決定されます。

表 *58. TIMESTAMPDIFF* の計算方法

結果の間隔	期間エレメントを使用した計算
年	年
四半期	(月 + (年 *12))/3 の整数値部分
月	月 + (年 *12)
週	((日 + (月 *30))/7) + (年 *52) の整数値部分
日	日 + (月 *30) + (年 *365)
時間	時間 + ((日 + (月 *30) + (年 *365))*24)
分 (期間の絶対値が 40850913020759.999999 を超えることはできません)	分 + (時間 + ((日 + (月 *30) + (年 *365))*24))*60
秒 (期間の絶対値が 680105031408.000000 より小さくなければなりません)	秒 + (分 + (時間 + ((日 + (月 *30) + (年 *365))*24))*60)*60
マイクロ秒 (期間の絶対値が 3547.483648 より小さくなければなりません)	マイクロ秒 + (秒 + (分 *60))*1000000

エレメント値を、要求された間隔タイプに変換する際、以下の前提事項が使用されます。

- 1 年は 365 日。
- 1 年は 52 週。
- 1 年は 12 カ月。
- 四半期は 3 カ月。
- 1 カ月は 30 日。
- 1 週は 7 日。
- 1 日は 24 時間。
- 1 時間は 60 分。
- 1 分は 60 秒。
- 1 秒は 1000000 マイクロ秒。

こうした前提事項を使用することにより、結果値の中には間隔が概数になる場合があります。以下の例について考えてみます。

- 1 カ月が 30 日に満たない場合の差。

```
TIMESTAMPDIFF(16, CHAR(TIMESTAMP('1997-03-01-00.00.00') - TIMESTAMP('1997-02-01-00.00.00') ) )
```

このタイム・スタンプ算術計算の結果は、期間 00000100000000.000000、つまり 1 カ月になります。TIMESTAMPDIFF 関数が間隔引数 16 (日) で呼び出されると、1 カ月は 30 日であるという前提が適用されて、結果は 30 になります。

- 30 日に満たない月の場合における、1 カ月より 1 日短いときの差。

```
TIMESTAMPDIFF(16, CHAR(TIMESTAMP('1997-03-01-00.00.00') - TIMESTAMP('1997-02-02-00.00.00') ) )
```

このタイム・スタンプ算術計算の結果は、期間 00000027000000.000000、つまり 27 日になります。TIMESTAMPDIFF 関数が間隔引数 16 (日) で呼び出されると、結果は 27 になります。

- 31 日ある月の場合における、1 カ月より 1 日短いときの差。

```
TIMESTAMPDIFF(64, CHAR(TIMESTAMP('1997-09-01-00.00.00') - TIMESTAMP('1997-08-02-00.00.00') ) )
```

このタイム・スタンプ算術計算の結果は、期間 00000030000000.000000、つまり 30 日になります。TIMESTAMPDIFF 関数が間隔引数 64 (月) で呼び出されると、結果は 0 になります。この期間の日部分は 30 ですが、指定された間隔が月なのでこれは無視されます。

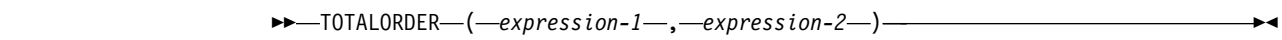
例

- 以下のステートメントでは月単位で従業員の年齢を見積もり、その値を AGE_IN_MONTHS として戻します。

```
SELECT
    TIMESTAMPDIFF(64,
    CAST(CURRENT_TIMESTAMP-CAST(BIRTHDATE AS TIMESTAMP) AS CHAR(22)))
      AS AGE_IN_MONTHS
  FROM EMPLOYEE
```

TOTALORDER

TOTALORDER 関数は DECFLOAT 値の順序付けを返します。



TOTALORDER 関数は、*expression-1* と *expression-2* の比較方法を示す短精度整数値を返します。

expression-1
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数が DECFLOAT(34) でない場合、その引数は処理のために DECFLOAT(34) に論理的に変換されます。

expression-2
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。引数が DECFLOAT(34) でない場合、その引数は処理のために DECFLOAT(34) に論理的に変換されます。

数値の比較は正確であり、有限オペランドの結果は、範囲と精度が無制限の場合と同様に求められます。オーバーフローまたはアンダーフローが起こってはなりません。

TOTALORDER は、IEEE 754R の全順序述部規則に基づいて順序付けを決定し、結果は以下のようになります。

-1	第 2 オペランドと比較して第 1 オペランドの順序が低い場合。
0	両方のオペランドが同じ順序の場合。
1	第 2 オペランドと比較して第 1 オペランドの順序が高い場合。

特殊値および有限数の順序付けは、以下のようになります。

-NAN<-SNAN<-INFINITY<-0.10<-0.100<-0<0<0.100<0.10<INFINITY<SNAN<NAN

関数の結果は SMALLINT になります。引数のどちらかが NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数のどちらかが NULL である場合は、結果は NULL 値になります。

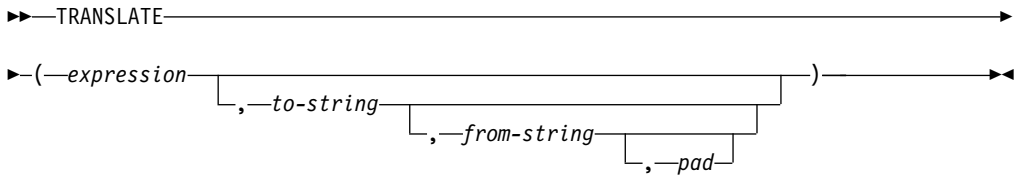
例

以下の例に、10 進浮動小数点値を比較する場合の TOTALORDER 関数の使用方法を示します。

```
TOTALORDER(-INFINITY, -INFINITY)                =    0
TOTALORDER(DECFLOAT(-1.0), DECFLOAT(-1.0))      =    0
TOTALORDER(DECFLOAT(-1.0), DECFLOAT(-1.00))     =   -1
TOTALORDER(DECFLOAT(-1.0), DECFLOAT(-0.5))      =   -1
TOTALORDER(DECFLOAT(-1.0), DECFLOAT(0.5))       =   -1
TOTALORDER(DECFLOAT(-1.0), INFINITY)            =   -1
TOTALORDER(DECFLOAT(-1.0), SNAN)                =   -1
TOTALORDER(DECFLOAT(-1.0), NAN)                 =   -1
TOTALORDER(NAN, DECFLOAT(-1.0))                 =    1
TOTALORDER(-NAN, -NAN)                          =    0
TOTALORDER(-SNAN, -SNAN)                        =    0
TOTALORDER(NAN, NAN)                            =    0
TOTALORDER(SNAN, SNAN)                          =    0
```

TRANSLATE

TRANSLATE 関数は、式 の中の 1 つ以上の文字を他の文字に変換した値を戻します。



expression
変換されるストリングを指定する式。式 は、任意の組み込み数値またはストリング・データ・タイプでなければなりません。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。

to-string
式 の中の特定の文字をどのような文字に変換するかを指定するストリング。このストリングは出力変換表 とも呼ばれます。このストリングは任意の組み込み数値またはストリング定数でなければなりません。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。文字ストリング引数の実際の長さは、256 以下でなければなりません。

変換先ストリング の実際の長さが変換元ストリング の実際の長さ より小さい場合、変換先ストリング は、埋め込み 文字が指定されている場合はその文字で、埋め込み 文字が指定されていない場合はブランクで、長い方の長さに合わせて埋められます。変換先ストリング の実際の長さが変換元ストリング の実際の長さよりも大きい場合、変換先ストリング 中の余分な文字は無視され、警告は出されません。

from-string
式 の中のどの文字を変換するのかを指定するストリング。このストリングは入力変換表 とも呼ばれます。変換元ストリング に指定されている文字のいずれかが式 の中に見つかると、その文字は、変換先ストリング の中の文字のうち、変換元ストリング でのその文字と同じ位置にある文字に変換されます。

このストリングは任意の組み込み数値またはストリング定数でなければなりません。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。文字ストリング引数の実際の長さは、256 以下でなければなりません。

変換元ストリング に重複する文字がある場合は、左からスキャンした最初の文字が使用され、警告は出されません。変換元ストリング のデフォルト値は、文字 X'00' で始まり、文字 X'FF' (10 進数 255) で終わるストリングです。

pad
to-string の長さが *from-string* より短い場合に埋め込む文字を指定するストリング。このストリングは、長さが 1 の文字ストリング定数でなければなりません。デフォルト値は SBCS のブランクです。

最初の引数が Unicode グラフィックまたは UTF-8 ストリングの場合は、他の引数を指定することができません。

最初の引数だけが指定されている場合は、引数の SBCS 文字は、その引数の CCSID に基づいて、大文字に変換されます。 SBCS 文字だけが変換されます。 a から z の文字は A から Z に変換され、発音記号付きの文字はそれぞれの大文字に変換されます。最初の引数が UTF-16、UCS-2、または UTF-8 の場合は、英字の UTF-16、UCS-2、または UTF-8 の文字は大文字に変換されます。この変換に使用する大文字変換表については、「グローバリゼーション」トピック集にあるトピック UCS-2 レベル 1 マッピング・テーブルを参照してください。

複数の引数を指定した場合は、結果のストリングは、変換元ストリング の文字を変換先ストリング の対応する文字に変換して、1 文字ずつ式 から構築されます。式の中の各文字ごとに、同一文字が変換元ストリング で検索されます。同じ文字が *from-string* の *n* 番目の文字として検出された場合、結果として得られるストリングは *to-string* からの *n* 番目の文字を含むことになります。変換先ストリング が *n* 文字より短い場合は、結果のストリングには埋め込み文字が入ります。その文字が変換元ストリング に見つからない場合は、未変換のまま結果のストリングに移されます。

変換はバイト基準で行われ、使用を誤った場合は、結果的に無効の混合ストリングになります。SRTSEQ 属性は、TRANSLATE 関数には適用されません。

この関数の結果のデータ・タイプ、長さ属性、実際の長さ、および CCSID は、引数と同じになります。 最初の引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。 引数が NULL である場合は、結果は NULL 値です。

例

- ストリング「abcdef」を大文字変換します。

```
SELECT TRANSLATE('abcdef')
  FROM SYSIBM.SYSDUMMY1
```

「ABCDEF」の値が戻されます。

- 混合文字ストリングを大文字変換します。

```
SELECT TRANSLATE('ab<SO>c<SI>def')

  FROMSYSIBM.SYSDUMMY1
```

次の値が戻されます。'AB<SO>C<SI>DEF'

- ホスト変数 SITE が、「Pivabiska Lake Place」という値の可変長文字ストリングである場合

```
SELECT TRANSLATE(:SITE, '$', 'L')
  FROM SYSIBM.SYSDUMMY1
```

値「Pivabiska $ake Place」が戻されます。

```
SELECT TRANSLATE(:SITE, '$$', 'Ll')
  FROM SYSIBM.SYSDUMMY1
```

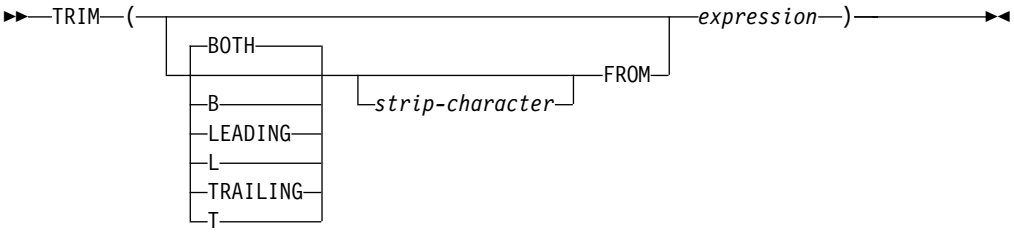
値「Pivabiska $ake P$ace」が戻されます。

```
SELECT TRANSLATE(:SITE, 'pLA', 'Place', '.')
  FROM SYSIBM.SYSDUMMY1
```

値 'pivAbiskA LAk. pLA..' が戻されます。 .

TRIM

TRIM 関数はブランクまたは指定されたその他の文字を、ストリング式の末尾、先頭、またはその両側から除去します。



最初の引数を指定する場合は、ストリングの後部と前部のどちらから文字を除去するのかを指示します。最初の引数を指定しない場合は、ストリングの前部と後部の両方から文字が除去されます。

strip-character
2 番目の引数が指定された場合、除去する 2 進数、SBCS または DBCS 文字を示す 1 文字定数となります。式 が 2 進ストリングの場合、 2 番目の引数は 2 進ストリング定数でなければなりません。式 が DBCS グラフィック・ストリングまたは DBCS 専用ストリングである場合は、 2 番目の引数は、1 つの DBCS 文字からなるグラフィック定数にする必要があります。 2 番目の引数を指定しない場合は、次のようになります。

- 式 が 2 進ストリングの場合、デフォルトの除去文字は 16 進ゼロ (X'00') になる。
- 式 が DBCS グラフィック・ストリングである場合は、デフォルトの除去文字は DBCS ブランクになる。
- 式 が Unicode グラフィック・ストリングである場合は、デフォルトの除去文字は UTF-16 または UCS-2 ブランクになる。
- 式 が UTF-8 文字ストリングである場合は、デフォルトの除去文字は UTF-8 ブランクになる。
- それ以外の場合は、デフォルトの除去文字は、SBCS ブランクになる。

expression
任意の組み込み数値データ・タイプ、またはストリング・データ・タイプの値を戻す式。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。

結果のデータ・タイプは、式 のデータ・タイプによって異なります。

式 のデータ・タイプ	結果のデータ・タイプ
CHAR または VARCHAR	VARCHAR
CLOB	CLOB
GRAPHIC または VARGRAPHIC	VARGRAPHIC
DBCLOB	DBCLOB
BINARY または VARBINARY	VARBINARY

式 のデータ・タイプ	結果のデータ・タイプ
BLOB	BLOB

結果の長さ属性は、式 の長さ属性と同じになります。結果の実際の長さは、式の長さから、除去したバイトの数を引いたものになります。すべての文字が除去された場合は、結果は空のストリングになります。

最初の引数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引数がヌルの場合は、結果は NULL 値になります。

結果の CCSID は、指定したストリングの CCSID と同じになります。

SRTSEQ 属性は、TRIM 関数には適用されません。

例

- ホスト変数 HELLO (タイプ CHAR(9)) には、値として「Hello　」が入っていると想定します。

```
SELECT TRIM(:HELLO), TRIM( TRAILING FROM :HELLO)
  FROM SYSIBM.SYSDUMMY1
```

  結果は、それぞれ、「Hello」および「 Hello」になります。

- ホスト変数 BALANCE (CHAR(9)) には、値として「000345.50」が入っていると想定します。

```
SELECT TRIM( L '0' FROM :BALANCE )
  FROM SYSIBM.SYSDUMMY1
```

  結果は「345.50」になります。

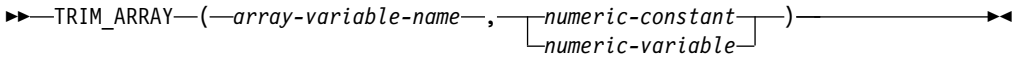
- 除去するストリングの中に、混合データが入っていると想定します。

```
SELECT TRIM( BOTH 'SO SI' FROM 'SO  AB  C  SI' )
    FROM SYSIBM.SYSDUMMY1
```

  結果は次のようになります。 'SOAB CSI'

TRIM_ARRAY

TRIM_ARRAY 関数は、配列引数の末尾から指定の数だけエレメントを削除して、配列のコピーを戻します。



array-variable-name
SQL 変数またはパラメーターを識別します。変数またはパラメーターは、配列タイプでなければなりません。

numeric-constant または *numeric-variable*
array-variable-name のコピーから切り取るエレメントの数を指定します。これは、整数データ・タイプにキャストできる定数、または SQL 変数またはパラメーターである必要があります。値は、0 から *array-variable-name* のカーディナリティーまでの範囲内でなければなりません。

結果の配列タイプは、最初の引数の配列タイプと同じですが、カーディナリティーは、切り取られたエレメント数だけ少なくなります。

結果は NULL になる可能性があります。どちらかの引数が NULL である場合、その結果は NULL 値になります。

TRIM_ARRAY は、SQL プロシージャーまたは SQL 関数内で *assignment-statement* の右側の唯一の式としてのみ使用できます。

例

配列タイプ PHONENUMBERS および配列変数 RECENT_CALLS が次のように定義されていると想定します。

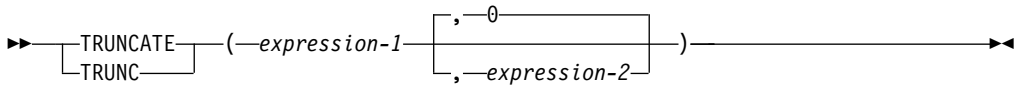
```
CREATE TYPE PHONENUMBERS AS INTEGER ARRAY[50];
DECLARE RECENT_CALLS PHONENUMBERS;
```

以下のステートメントは、配列変数 RECENT_CALLS から最後のエレメントを削除します。

```
SET RECENT_CALLS = TRIM_ARRAY(RECENT_CALLS, 1)
```

TRUNCATE または TRUNC

TRUNCATE 関数は、*expression–1* を、小数点の右側または左側の特定の桁で切り捨てた値を返します。



expression–1
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。ストリング引数は、関数を評価する前に倍精度の浮動小数点に変換されます。ストリングを倍精度の浮動小数点に変換する方法については、 448 ページの『DOUBLE_PRECISION または DOUBLE』を参照してください。

expression-1 が 10 進浮動小数点データ・タイプの場合、DECFLOAT ROUNDING MODE は使用されません。TRUNCATE の切り下げ動作は、ROUND_DOWN の値に対応します。別の丸め動作が必要な場合は、QUANTIZE 関数を使用します。

expression–2
組み込み短精度整数、長精度整数、または 64 ビット整数データ・タイプの値を戻す式。整数の絶対値は、*expression–2* が負でなければ、結果の小数点より右の桁数を指定し、 *expression–2* が負であれば、小数点より左の桁数を指定します。

expression–2 が負でない場合、*expression–1* は小数点の右側の *expression–2* 桁で切り捨てられます。

expression–2 が負の場合、 *expression–1* は、小数点の左側の *expression–2* の絶対値 +1 桁で切り捨てられます。

expression–2 が指定されていない場合、 *expression–1* は小数点の左側の 0 桁で切り捨てられます。

expression–2 の絶対値が小数点より左の桁数より大きい場合、結果は 0 になります。例えば、TRUNCATE (748.58,-4) = 0 です。

結果のデータ・タイプおよび長さ属性は、最初の引数のデータ・タイプおよび長さ属性と同じです。

どちらかの引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数のどちらかが NULL である場合は、結果は NULL 値になります。

例

- 最も給与の高い従業員の平均月収を計算します。その結果を小数点の右、2 桁目までで切り捨てます。

```
SELECT TRUNCATE(MAX(SALARY/12) , 2)
  FROM EMPLOYEE
```

サンプルの従業員表内で給与の最も高い従業員の年収は $52750.00 なので、この例では 4395.83 の値が戻されます。

- 数値 873.726 を、小数点から 2、1、0、-1、-2、-3 までで切り捨てます。

```
SELECT TRUNCATE(873.726,  2),
       TRUNCATE(873.726,  1),
       TRUNCATE(873.726,  0),
       TRUNCATE(873.726, -1),
       TRUNCATE(873.726, -2),
       TRUNCATE(873.726, -3)
  FROM SYSIBM.SYSDUMMY1
```

それぞれ以下の値が戻されます。

```
0873.720   0873.700  0873.000  0870.000  0800.000  0000.000
```

- 正負両方の数値を計算します。

```
SELECT TRUNCATE( 3.5, 0),
       TRUNCATE( 3.1, 0),
       TRUNCATE(-3.1, 0),
       TRUNCATE(-3.5, 0)
  FROM SYSIBM.SYSDUMMY1
```

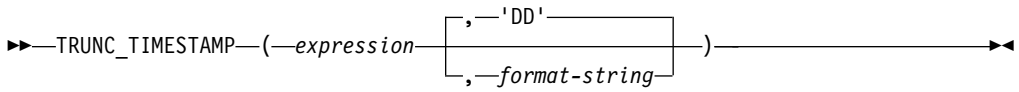
この例ではそれぞれ、

```
3.0   3.0  -3.0  -3.0
```

が戻されます。

TRUNC_TIMESTAMP

TRUNC_TIMESTAMP 関数は、*timestamp-expression* を *format-string* によって指定された単位に切り捨てたタイム・スタンプを返します。*format-string* を指定しないと、*expression* は、*format-string* に「DD」が指定されたものとして、最も近い日に切り捨てられます。



expression
日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を返す式。

expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。最初に TIMESTAMP(12) 値に変換されます。 日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

format-string
組み込み文字ストリング・データ・タイプか、グラフィック・ストリング・データ・タイプを戻す式。 *format-string* には、*expression* をどのように切り捨てるのかを示すテンプレートが含まれます。例えば、*format-string* が 'DD' である場合、*expression* で表されるタイム・スタンプは、直近の日に切り捨てられます。前後のブランクはストリングから除去され、結果として得られるサブストリングは、タイム・スタンプとして有効なテンプレートでなければなりません。その結果の値は大文字変換されるため、値の中の文字は大文字、小文字のどちらでも構いません。*format-string* に指定できる値を、 609 ページの表 54 にリストします。

関数の結果は以下のタイム・スタンプ精度の TIMESTAMP になります。

- *expression* のデータ・タイプが TIMESTAMP(*p*) の場合は *p*。
- *expression* のデータ・タイプが DATE の場合は 0。
- それ以外の場合は 6。

引数のどちらかが NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数のどちらかが NULL である場合は、結果は NULL 値になります。

例

- 現在の年を最も近い年の値に切り捨てた値を、ホスト変数 TRN_TMSTMP に設定します。

```
SET :TRN_TMSTMP = TRUNC_TIMESTAMP('2000-03-14-17.30.00', 'YEAR');
```

ホスト変数 TRN_TMSTMP には、値 2000-01-01-00.00.00.000000 が設定されます。

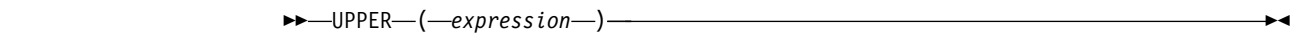
UCASE

UCASE 関数は、すべての文字を引数の CCSID に基づいて大文字に変換したストリングを戻します。

UCASE 関数は、UPPER 関数と同等です。 詳しくは、 667 ページの『UPPER』を参照してください。

UPPER

UPPER 関数は、すべての文字を引数の CCSID に基づいて大文字に変換したストリングを戻します。 SBCS および Unicode グラフィック文字だけが変換されます。a から z の文字は A から Z に変換され、発音記号付きの文字はそれぞれの大文字に変換されます。



この変換に使用する大文字変換表については、「グローバリゼーション」トピック集にあるトピック UCS-2 レベル 1 マッピング・テーブルを参照してください。

expression
変換するストリングを指定する式。*expression* は、任意の組み込み数値、文字、Unicode グラフィック・ストリングでなければなりません。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。

この関数の結果のデータ・タイプ、長さ属性、実際の長さ、および CCSID は、引数と同じになります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

注記

代替構文**:** UCASE は UPPER の同義語です。

例

- UPPER スカラー関数を使用して、ストリング「abcdef」を大文字変換します。

```
SELECT UPPER('abcdef')
    FROM SYSIBM.SYSDUMMY1
```

「ABCDEF」の値が戻されます。

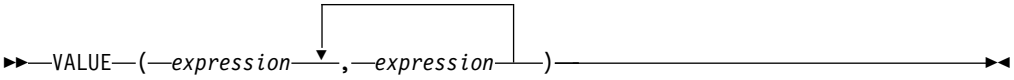
- UPPER スカラー関数を使用して、混合文字ストリングを大文字変換します。

```
SELECT UPPER( 'ab<SO>c<SI>def' )
    FROM SYSIBM.SYSDUMMY1
```

次の値が戻されます。 'AB<SO>C<SI>DEF'

VALUE

VALUE 関数は、NULL でない最初の式の値を戻します。



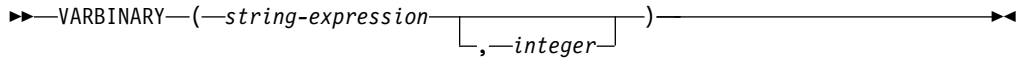
VALUE 関数は、COALESCE スカラー関数と同等です。詳しくは、 390 ページの『COALESCE』を参照してください。

注記

代替構文: SQL 2003 規格に準拠して COALESCE を使用する必要があります。

VARBINARY

VARBINARY 関数は、任意のタイプのストリングの VARBINARY 表現を戻します。



この関数の結果は、VARBINARY になります。最初の引数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引数がヌルの場合は、結果は NULL 値になります。

string-expression
値が文字ストリング、グラフィック・ストリング、2 進ストリング、または行 ID のいずれかであるストリング式。

integer
結果の 2 進ストリングの長さ属性を指定する整数定数。値は 1 から 32740 (NULL 可能な場合は 32739) までででなければなりません。

整数 を指定しなかった場合は、以下のようになります。

- ストリング式 が空ストリング定数の場合は、結果の長さ属性は 1。
- 空ストリング定数以外の場合は、最初の引数が漢字ストリングでなければ、結果の長さ属性は最初の引数の長さ属性と同じになる。グラフィック・ストリングの場合は、結果の長さ属性は、引数の長さ属性の 2 倍です。

結果の実際の長さは、結果の長さ属性と式の実際の長さ (または入力がグラフィック・データの場合は式の長さの 2 倍) のいずれか小さい方となります。ストリング式 の長さが結果の長さ属性よりも大きい場合は、切り捨てが行われます。最初の入力引数が文字ストリングで、切り捨てられた文字がすべてブランクである場合、最初の入力引数がグラフィック・ストリングで、切り捨てられた文字がすべて 2 バイト・ブランクである場合、または最初の入力引数が 2 バイト・ストリングで、切り捨てられたバイトがすべて 16 進数のゼロである場合以外は、警告 (SQLSTATE 01004) が戻されます。

注記

代替構文: 長さを指定する場合、アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、 218 ページの『CAST の指定』を参照してください。

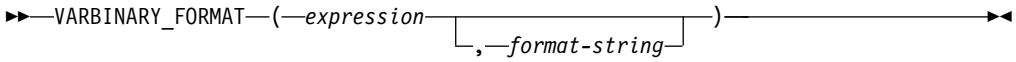
例

- 次の関数は、ストリング「This is a VARBINARY」の VARBINARY を戻します。

```
SELECT VARBINARY('This is a VARBINARY')
  FROM SYSIBM.SYSDUMMY1
```

VARBINARY_FORMAT

VARBINARY_FORMAT 関数は、*format-string* を使用してフォーマットされた文字ストリングのバイナリー・ストリング表現を戻します。



expression
任意の組み込み数値、文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。数値引数またはグラフィック引数は、関数を評価する前に文字ストリングにキャストされます。数値またはグラフィックから文字ストリングへの変換について詳しくは、 672 ページの『VARCHAR』 を参照してください。

すべての先行ブランクと末尾ブランクは、関数を評価する前に *expression* から除去されます。

format-string が指定されている場合、*expression* の長さは、*format-string* の長さに等しく、かつ、*expression* の値は、*format-string* で指定されたテンプレートに準拠しなければなりません。 *format-string* が指定されていない場合、*expression* の (先行ブランクと末尾ブランクを除去した後の) 値は、範囲 '0' から '9'、'a' から 'f'、および 'A' から 'F' に属する偶数個の文字でなければなりません。長さの文字数が奇数である場合は、ストリングの右側に文字 '0' が 1 つ埋め込まれます。

format-string
組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプを返す式。*format-string* には、*expression* の値を解釈する方法に関するテンプレートが含まれています。

有効なフォーマット・ストリングは 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx' と 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX' です。ここで、それぞれの 'x' または 'X' は、結果内の 1 つの 16 進数字に対応します。 'X' が指定されている場合、対応する 16 進数字は小文字であってはなりません。 'x' が指定されている場合、対応する 16 進数字は大文字であってはなりません。

この関数の結果は、可変長バイナリー・ストリングです。結果の長さ属性は、*expression* の長さ属性の半分になります。 *format-string* が指定されていない場合、実際の長さは (先行ブランクと末尾ブランクが除去されて、埋め込みにより文字数が偶数になった後の) *expression* の実際の長さの半分になります。 *format-string* が指定されている場合、実際の長さは (数字以外の区切り文字を除去した後の) *format-string* の実際の長さの半分になります。引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

注記

代替構文: HEXTORAW は、*expression* の長さが奇数 (文字数) である場合、ストリングの左側に文字 '0' が 1 つ埋め込まれることを除けば、VARBINARY_FORMAT の同義語です。 VARCHAR_BIT_FORMAT は、関数の結果が可変長文字ストリン

グ FOR BIT DATA であることを除けば、VARBINARY_FORMAT の同義語です。

例

- UUID (Universal Unique Identifier) をそのバイナリー形式で表す

```
VALUES VARBINARY_FORMAT('d83d6360-1818-11db-9804-b622a1ef5492',
                        'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx')
```

  戻される結果:

```
BX'D83D6360181811DB9804B622A1EF5492'
```

- UUID (Universal Unique Identifier) をそのバイナリー形式で表す

```
VALUES VARBINARY_FORMAT('D83D6360-1818-11DB-9804-B622A1EF5492',
                        'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX')
```

  戻される結果:

```
BX'D83D6360181811DB9804B622A1EF5492'
```

- 16 進文字のストリングをバイナリー形式で表します。

```
VALUES VARBINARY_FORMAT('ef01abC9')
```

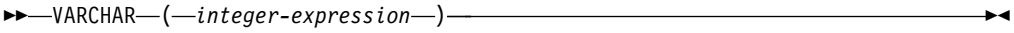
  戻される結果:

```
BX'EF01ABC9'
```

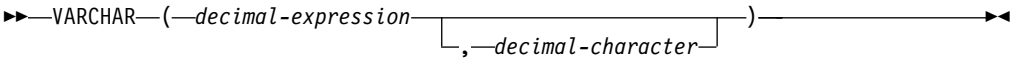
VARCHAR

VARCHAR 関数は、文字ストリング表現を返します。

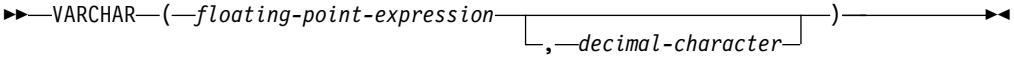
整数から可変長文字



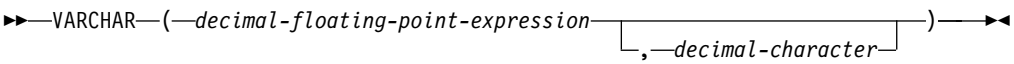
10 進数から可変長文字



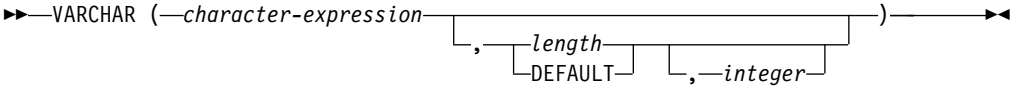
浮動小数点数から可変長文字に



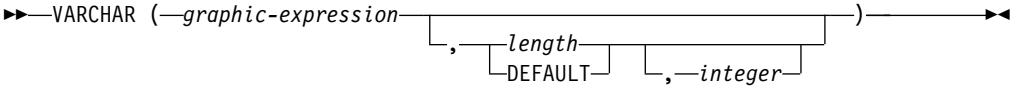
10 進浮動小数点数から可変長文字に



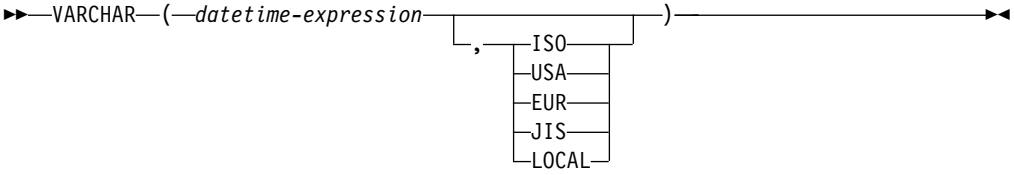
文字から可変長文字に



グラフィックから可変長文字に



日付/時刻から可変長文字に



VARCHAR 関数は、次のものの文字ストリング表現を戻します。

- 整数 (最初の引数が SMALLINT、INTEGER、または BIGINT の場合)
- 10 進数 (最初の引数がパックまたはゾーン 10 進数の場合)
- 倍精度浮動小数点数 (最初の引数が DOUBLE または REAL の場合)

- 最初の引数が DECFLOAT である場合は 10 進浮動小数点数。
- 文字ストリング (最初の引数が任意のタイプの文字ストリングの場合)
- グラフィック・ストリング (最初の引数が任意のグラフィック・ストリングの場合)
- 日付値 (最初の引数が DATE の場合)
- 時刻値 (最初の引数が TIME の場合)
- タイム・スタンプ値 (最初の引数が TIMESTAMP)

この関数の結果は可変長ストリングです。最初の引数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引数がヌルの場合は、結果は NULL 値になります。

整数から可変長文字に

integer-expression
整数データ・タイプ (SMALLINT、INTEGER、または BIGINT) の値を戻す式。

結果は、SQL 整数定数の形式で引数を表現した可変長文字ストリングです。結果は、引数の値を表す n 文字の有効数字から成ります。引数が負数の場合は、負符号が前に付きます。結果は左寄せされます。

- 引数が短整数の場合は、結果の長さ属性は 6
- 引数が長整数の場合は、結果の長さ属性は 11
- 引数が 64 ビット整数の場合は、結果の長さ属性は 20

結果の実際の長さは、引数の値を表すために使用できる最小文字数です。先行ゼロは含まれません。引数が負数の場合は、結果の最初の文字は負符号になります。そうでない場合、先頭文字は数字です。

結果の CCSID は、現行サーバーのデフォルトの SBCS CCSID になります。

10 進数から可変長文字に

decimal-expression
パックまたはゾーン 10 進数データ・タイプ (DECIMAL または NUMERIC) の値を戻す式。精度や位取りを変えたい場合は、DECIMAL スカラー関数を使用して変更することができます。

decimal-character
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

結果は、引数を可変長文字ストリングで表現したものになります。この結果には、1 文字の小数点文字と *p* 桁までの数字が含まれます。 *p* は *10* 進数式 の精度で、引数が負数の場合は負符号が先頭に付きます。先行ゼロは戻されません。後続ゼロは戻されます。*decimal-expression* の位取りがゼロの場合、小数点文字は戻されません。

結果の長さ属性は 2+*p* です (*p* は *decimal-expression* の精度)。結果の実際の長さは、引数の値を表すために使用できる最小文字数ですが、ただし、後書き文字も含まれます。先行ゼロは含まれません。引数が負数の場合は、結果の最初の文字は負符号になります。負符号でなければ、結果の最初の文字は数字または *decimal-character* になります。

結果の CCSID は、現行サーバーのデフォルトの SBCS CCSID になります。

浮動小数点数から可変長文字に

浮動小数点数式
浮動小数点データ・タイプ (DOUBLE または REAL) の値を戻す式。

decimal-character
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

結果は、浮動小数点定数の形式で引数を可変長文字ストリングで表現したものになります。

結果の長さ属性は、24 です。結果の実際の長さは、ゼロ以外の 1 桁の数字、その後ろに 1 つの小数点文字 と一連の数字が続く小数部の引数の値を表す最小文字数です。引数が負数の場合は、結果の最初の文字は負符号になります。負数でなければ、最初の文字は数字または *decimal-character* です。引数がゼロであると、結果は 0E0 になります。

結果の CCSID は、現行サーバーのデフォルトの SBCS CCSID になります。

10 進浮動小数点数から可変長文字に

decimal-floating-point expression
10 進浮動小数点データ・タイプの値を返す式。

decimal-character
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

結果は、10 進浮動小数点定数の形式で引数を可変長文字ストリングで表現したものになります。

結果の長さ属性は、42 です。 結果の実際の長さは、記号、数字、および *decimal-character* を含む、引数の値を表す最小文字数です。後続ゼロは有効数字です。引数が負数の場合は、結果の最初の文字は負符号になります。負数でなければ、最初の文字は数字または *decimal-character* です。引数がゼロであると、結果は 0 になります。

DECFLOAT 値が Infinity、sNaN、または NaN の場合、それぞれストリング 'INFINITY'、'SNAN'、および 'NAN' が返されます。特殊値が負の場合、負符号 (-)

がそのストリングの最初の文字となります。 DECFLOAT 特殊値 sNaN を使用しても、ストリングへの変換時に例外は発生しません。

結果の CCSID は、現行サーバーのデフォルトの SBCS CCSID になります。

文字から可変長文字に

character-expression
組み込みの CHAR、VARCHAR、または CLOB データ・タイプの値を返す式。[76]

length
結果の可変長文字ストリングの長さ属性を指定する整数定数。値は 1 から 32740 (NULL 可能な場合は 32739) まででなければなりません。最初の引数が DBCS 専用混合データである場合は、2 番目の引数は 4 より小さくてはなりません。

2 番目の引数が指定されないか DEFAULT が指定された場合は、次のようになります。

- 文字式 が空ストリング定数の場合は、結果の長さ属性は 1。
- 空ストリング定数でない場合は、結果の長さ属性は、最初の引数の長さ属性と同じ。

結果の実際の長さは、結果の長さ属性と *character-expression* の実際の長さのうち、小さい方になります。*character-expression* の長さが結果の長さ属性よりも大きい場合は、切り捨てが行われます。切り捨てられた文字がすべてブランクであった場合以外は、警告 (SQLSTATE 01004) が戻されます。

integer
結果の CCSID を指定する整数定数。これは有効な SBCS CCSID、混合データ CCSID、または 65535 (ビット・データ) とする必要があります。 3 番目の引数が SBCS CCSID の場合は、結果は SBCS データになります。 3 番目の引数が混合 CCSID の場合、結果は混合データです。結果の長さ属性が 4 より小さくてはなりません。 3 番目の引数が 65535 の場合、結果はビット・データです。 3 番目の引数が SBCS CCSID の場合は、最初の引数が DBCS 択一または DBCS 専用のストリングであることはありません。

3 番目の引数の指定がない場合は、次のようになります。

- 最初の引数が SBCS データであれば、結果は SBCS データになる。結果の CCSID は、最初の引数の CCSID と同一です。
- 最初の引数が混合データで、結果の長さ属性が 4 以上の場合、結果は混合データです。結果の CCSID は、最初の引数の CCSID と同一です。
- 最初の引数が DBCS 混用または DBCS 択一の混合データで、結果の長さ属性が 4 未満である場合、結果の CCSID は、混合データ CCSID に関連した SBCS CCSID です。

76. CCSID が指定されていない場合、または CCSID として 65535 が明示的に指定されている場合、バイナリー・ストリングも使用できます。

グラフィックから可変長文字に

graphic-expression
GRAPHIC、VARGRAPHIC、または DBCLOB データ・タイプの値を戻す式。最初の引数は、DBCS グラフィック・データであってはなりません。

length
結果の可変長文字ストリングの長さ属性を指定する整数定数。値は 1 から 32740 (NULL 可能な場合は 32739) までででなければなりません。 最初の引数が DBCS データを含む場合は、2 番目の引数は 4 より小さくてはなりません。

2 番目の引数が指定されていないか、または DEFAULT が指定されている場合は、結果の長さ属性は、次のように決まります。(ただし、*n* は最初の引数の長さ属性です。)

- グラフィック式 が空グラフィック・ストリング定数の場合は、結果の長さ属性は 1 になる。
- 結果が SBCS データの場合、結果の長さは *n* です。
- 結果が混合データであれば、結果の長さは (2.5*(*n* -1)) + 4 になる。

結果の実際の長さは、結果の長さ属性と *graphic-expression* の実際の長さのうち、小さい方になります。*character-expression* の長さが結果の長さ属性よりも大きい場合は、切り捨てが行われます。切り捨てられた文字がすべてブランクであった場合以外は、警告 (SQLSTATE 01004) が戻されます。

integer
結果の CCSID を指定する整数定数。これは有効な SBCS CCSID または混合データ CCSID とする必要があります。 3 番目の引数が SBCS CCSID の場合は、結果は SBCS データになります。 3 番目の引数が混合 CCSID の場合、結果は混合データです。結果の長さ属性が 4 より小さくてはなりません。 3 番目の引数は 65535 であってはなりません。

3 番目の引数が指定されていない場合は、結果の CCSID は現行サーバーのデフォルト CCSID になります。デフォルト CCSID が混合データで、結果の長さ属性が 4 以上の場合、結果は混合データです。それ以外の場合、その結果は SBCS データとなります。

日付/時刻から可変長文字に

datetime-expression
次の 3 つの組み込みデータ・タイプのいずれかである式。

日付　結果は、2 番目の引数によって指定された形式の日付の可変長文字文字ストリング表記になります。 2 番目の引数を指定しなかった場合は、デフォルトの日付形式が使用されます。形式として ISO、USA、EUR、または JIS を指定すると、長さ属性と結果の実際の長さは 10 になります。その他の場合は、長さ属性と結果の実際の長さはデフォルトの日付形式の長さになります。詳しくは、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

時刻　結果は、2 番目の引数によって指定された形式の時刻の可変長文字文字ストリング表記になります。 2 番目の引数を指定しなかった場合は、

デフォルトの時刻形式が使用されます。長さ属性と結果の実際の長さは 8 になります。詳しくは、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

timestamp
2 番目の引数は適用されないので、指定してはなりません。

結果は、そのタイム・スタンプを可変長文字ストリング表現したものとなります。*datetime-expression* が TIMESTAMP(0) の場合、結果の長さ属性および実際の長さは 19 です。*datetime-expression* のデータ・タイプが TIMESTAMP(*n*) の場合、結果の長さ属性および実際の長さは 20+*n* です。それ以外の場合、結果の長さ属性および実際の長さは 26 です。

ストリングの CCSID は、現行サーバーにおけるデフォルト SBCS CCSID です。

ISO、EUR、USA、または JIS
結果の文字ストリングの日付形式または時刻形式を指定します。詳しくは、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

LOCAL
結果の文字ストリングの日付または時刻の形式を、現行サーバーのジョブの DATFMT、DATSEP、 TIMFMT、および TIMSEP 属性から取る必要があることを指定します。

注記

代替構文**:** 最初の引数がストリングで、長さ引数を指定する場合、アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、 218 ページの『CAST の指定』を参照してください。

例

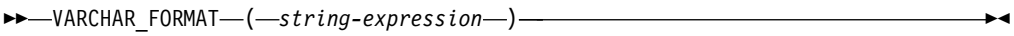
- EMPNO を長さ 10 の可変長にします。

```
SELECT VARCHAR(EMPNO,10)
  INTO :VARHV
  FROM EMPLOYEE
```

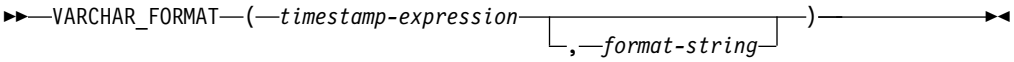
VARCHAR_FORMAT

VARCHAR_FORMAT 関数は、最初の引数の文字ストリング表現を、オプションの *format-string* で示されたフォーマットで戻します。

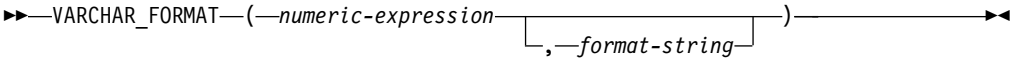
文字から可変長文字に



タイム・スタンプから可変長文字に



数値から可変長文字に



VARCHAR_FORMAT 関数のいずれかの引数が NULL 値の可能性がある場合、結果も NULL 値になる可能性があります。いずれかの引数が NULL 値の場合、結果は NULL 値になります。

文字から可変長文字に

string-expression

組み込み文字ストリングまたはグラフィック・ストリングのデータ・タイプの値を戻す式。

引数が文字ストリングの場合 :

- 結果の長さ属性と実際の長さは、以下のようにして決まります。
  - *string-expression* が空ストリング定数の場合は、結果の長さ属性は 1。
  - 空ストリング定数でない場合は、結果の長さ属性と実際の長さは、*string-expression* の長さ属性と同じになります。
  - 結果の実際の長さは、結果の長さ属性と *string-expression* の実際の長さのうち、小さい方になります。
- 結果の CCSID は以下のように決定されます。
  - *string-expression* が SBCS データであれば、結果は SBCS データになる。結果の CCSID は *string-expression* の CCSID と同じです。
  - *string-expression* が混合データ (DBCS 混用、DBCS 専用、または DBCS 択一) であれば、結果は混合データになる。結果の CCSID は *string-expression* の CCSID と同じです。

引数がグラフィック・ストリングの場合、DBCS グラフィック・データであってはなりません。

- 結果の長さ属性と実際の長さは、以下のようにして決まります (ここで、*n* は *string-expression* の長さ属性です)。
  - *string-expression* が空のグラフィック・ストリング定数の場合、結果の長さ属性は 1 です。

  - 結果が SBCS データの場合、結果の長さ属性は *n* です。
  - 結果が混合データの場合、結果の長さ属性は (2.5*(*n*-1)) + 4 です。
  - 結果の実際の長さは、結果の長さ属性と *string-expression* の実際の長さのうち、小さい方になります。
- 結果の CCSID は、現行サーバーにおけるデフォルト CCSID です。デフォルト CCSID が混合データの場合は、結果は混合データになります。デフォルト CCSID が SBCS データの場合は、結果は SBCS データになります。

タイム・スタンプから可変長文字に

timestamp-expression
日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式。引数が DATE である場合、時刻が午前 0 時ちょうど (00.00.00) であると想定して、最初に TIMESTAMP(0) 値に変換されます。

timestamp-expression が文字ストリングまたはグラフィック・ストリングである場合、*timestamp-expression* の値は日付またはタイム・スタンプの有効なストリング表現でなければなりません。最初に TIMESTAMP(12) 値に変換されます。タイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

引数がストリングの場合は、*format-string* 引数も指定しなければなりません。

format-string
組み込み文字ストリング・データ・タイプか、グラフィック・ストリング・データ・タイプを戻す式。値が CHAR データ・タイプでも VARCHAR データ・タイプでもない場合、その値は関数を評価する前に暗黙的に VARCHAR にキャストされます。*format-string* には、*timestamp-expression* のフォーマット設定方法を示すテンプレートが含まれています。結果値は大/小文字の任意の文字を含むことができます。 有効な形式は、以下にリストされた形式を任意に組み合わせたものであり、オプションで、有効な区切り文字で区切ることができます。有効な区切り文字は次のとおりです。

- 負符号 (-)
- ピリオド (.)
- スラッシュ (/)
- コンマ (,)
- アポストロフィ (')
- セミコロン (;)
- コロン (:)
- ブランク ( )

表 *59. VARCHAR_FORMAT (*タイム・スタンプから可変長文字*)* 関数のフォーマット・エレメント

形式	単位
AM または PM [1, 2]	ピリオドが付かない午前/午後の指定子。戻される午前/午後の指定子は、ジョブのメッセージに使用される言語に基づいています。この午前/午後の指定子は、ライブラリー *LIBL の中のメッセージ・ファイル QCPFMSG 内のメッセージ CPX9035 から取り出されます。

表 59. *VARCHAR_FORMAT (タイム・スタンプから可変長文字) 関数のフォーマット・エレメント (続き)*

形式	単位
A.M. または P.M. [1、2]	ピリオドが付いた午前/午後の指定子。このフォーマット・エレメントは、「A.M.」または「P.M.」と正確に一致するストリングを使用し、ジョブのメッセージに使用される言語には関係ありません。
CC	世紀 (00 から 99)。4 桁の年の最後の 2 桁がゼロの場合、結果は年の最初の 2 桁です。それ以外の場合、結果は年の最初の 2 桁に 1 を加えた数です。
DAY、Day、または day [1、2]	大文字、タイトル文字、または小文字のフォーマットの曜日の名前。戻される曜日の名前は、ジョブのメッセージに使用される言語に基づいています。曜日の名前は、ライブラリー *LIBL の中のメッセージ・ファイル QCPFMSG のメッセージ CPX9034 から検索されます。
DY、Dy、または dy [1、3]	大文字、タイトル文字、または小文字のフォーマットの曜日の省略名。省略された曜日の名前は、ライブラリー *LIBL の中のメッセージ・ファイル QCPFMSG のメッセージ CPX9039 から検索されます。
D [1]	曜日 (1-7)。1 は日曜日です。
DD	日 (01 から 31)。
DDD	年間通算日 (001 から 366)。
FF または FF*n*	小数秒 (0 から 999999999999)。数値 *n* は、返される値に含む桁数を指定します。 *n* の有効な値は 1 から 12 です。デフォルトは 6 です。
HH	HH の動作は HH12 と同様です。
HH12	12 時間形式の時 (01-12)。
HH24	24 時間形式の時刻 (00 から 24)。
ID	ISO の曜日 (1 から 7)。1 は月曜日、7 は日曜日です。
IW	ISO の年の週 (01 から 53)。この週は月曜日に始まり、7 日間を含みます。週 1 は、木曜日を含む年の第 1 週目です。これは、1 月 4 日を含むその年の第 1 週と同じです。
I	ISO の年 (0 から 9)。返される ISO 週番号に基づく年の最後の桁。
IY	ISO の年 (00 から 99)。返される ISO 週番号に基づく年の最後の 2 桁。
IYY	ISO の年 (000 から 999)。返される ISO 週番号に基づく年の最後の 3 桁。
IYYY	ISO の年 (0000 から 9999)。返される ISO の週に基づく年。
J	ユリウス日付 (0000000 から 9999999)。
MI	分 (00 から 59)。
MM	月 (01 から 12)。
MONTH、Month、または month [1、3]	大文字、タイトル文字、または小文字のフォーマットの月の名前。戻される月の名前は、ジョブのメッセージに使用される言語に基づいています。月の名前は、ライブラリー *LIBL の中のメッセージ・ファイル QCPFMSG のメッセージ CPX3BC0 から検索されます。

表 59. VARCHAR_FORMAT (タイム・スタンプから可変長文字) 関数のフォーマット・エレメント (続き)

形式	単位
MON、Mon、または mon [1, 3]	大文字、タイトル文字、または小文字のフォーマットの月の省略名。戻される月の名前は、ジョブのメッセージに使用される言語に基づいています。月の名前は、ライブラリー *LIBL の中のメッセージ・ファイル QCPFMSG のメッセージ CPX8601 から検索されます。
MS	ミリ秒 (000-999)。FF3 と同様。
NNNNNN	マイクロ秒 (000000-999999)。FF6 と同様。
Q	四半期 (1 から 4)。
RR	RR の動作は YY と同様です。
RRRR	RRRR の動作は YYYY と同様です。
SS	秒 (00 から 59)。
SSSSS	前の深夜 12 時以降の秒数 (00000 から 86400)。
US	マイクロ秒 (000000-999999)。FF6 と同様。
W	月刊通算週 (1 から 5)。週 1 は、月の最初の日から始まり、7 日目に終了します。
WW	年の週 (01 から 53)。第 1 週は、1 月 1 日に始まり、1 月 7 日に終わります。
Y	年の最後の 1 桁の数字 (0 から 9)。
YY	年の最後の 2 桁の数字 (00 から 99)。
YYY	年の最後の 3 桁の数字 (000 から 999)。
YYYY	年 (0000 から 9999)。

注:

1. このフォーマット・エレメントでは、大文字小文字が区別されます。フォーマット・エレメントがあいまいな場合は、大/小文字を区別しないフォーマット・エレメントが最初に考慮されます。
2. 午前/午後の指定子として、A.M. と P.M. と同様に AM と PM のセットを *format-string* で使用できます。結果ストリングには、実際の時刻値に適切な午前/午後の指定子が入ります。
3. 正確なスペルと大文字小文字の組み合わせだけが使用できます。このフォーマット・エレメントが無効な大文字小文字の組み合わせで指定された場合、エラーが返されます。

有効なフォーマット・ストリングの例を次に示します。

```
'HH24-MI-SS'
'HH24-MI-SS-NNNNNN'
'YYYY-MM-DD'
'YYYY-MM-DD-HH24-MI-SS'
'YYYY-MM-DD-HH24-MI-SS-NNNNNN'
'FF3.J/Q-YYYY'
```

結果は、*timestamp-expression* を *format-string* で指定された形式で表現したものです。*format-string* は、1 つ以上の区切り文字で区切ることができる一連のフォーマット・エレメントとして解釈されます。*format-string* 内の文字ストリングは、前述の表のエレメントに一致する最長のフォーマット・エレメントとして解釈されま

す。2 つのフォーマット・エレメントが同じ文字で構成され、区切り文字で分離されていない場合、その指定は、左から始まり、前述の表のエレメントに一致する最長エレメントとして解釈され、フォーマット・ストリングの残りの部分について一致が見つかるまで続きます。例えば、DDYYYY は D の後に DY その後に YYY が続くと解釈されるのではなく、DD の後に YYYY が続くと解釈されます。

format-string が指定されていない場合、*timestamp-expression* は、タイム・スタンプの可変長文字ストリング表現として戻されます。

結果のデータ・タイプは、*format-string* のデータ・タイプに基づき、可変長文字または可変長グラフィックです。結果の長さ属性は最大 255 で、*format-string* の長さ属性です。*format-string* は、結果の実際の長さも決定します。実際の長さは、結果の長さ属性を超えてはなりません。

結果の CCSID は *format-string* の CCSID と同じです。*format-string* が指定されていない場合、結果の CCSID は現行サーバーのデフォルトの SBCS CCSID になります。

数値から可変長文字に

numeric-expression
: 任意の組み込み数値データ・タイプの値を戻す式。引数が 10 進浮動小数点値ではない場合、処理のために DECFLOAT(34) に変換されます。

format-string
: 組み込み文字ストリング、グラフィック・ストリング、または数値のデータ・タイプを戻す式。値が CHAR データ・タイプでも VARCHAR データ・タイプでもない場合、その値は関数を評価する前に暗黙的に VARCHAR にキャストされます。*format-string* には、*numeric-expression* のフォーマット設定方法を示すテンプレートが含まれています。*format-string* には、以下にリストするフォーマット・エレメントの有効な組み合わせが含まれていなければなりません。有効な組み合わせは、次の規則に従います。

  - 符号のフォーマット・エレメント (「S」、「MI」、「PR」) を指定できるのは 1 回だけです。
  - 小数点のフォーマット・エレメントを指定できるのは 1 回だけです。
  - 英字のフォーマット・エレメントは、大文字で指定する必要があります。
  - 接頭部のフォーマット・エレメントを指定できるのは、フォーマット・ストリングの先頭のみです。また、接頭部のフォーマット・エレメントの前に接頭部以外のフォーマット・エレメントがあってはなりません。複数の接頭部のフォーマット・エレメントを指定するときには、どのような順番で指定してもかまいません。
  - 接尾部のフォーマット・エレメントを指定できるのは、フォーマット・ストリングの末尾のみです。また、接尾部のフォーマット・エレメントの後に接尾部以外のフォーマット・エレメントがあってはなりません。複数の接尾部のフォーマット・エレメントを指定するときには、どのような順番で指定してもかまいません。

- コンマおよび G のフォーマット・エレメントは、接頭部のフォーマット・エレメント以外の最初のフォーマット・エレメントにしてはなりません。コンマまたは G のフォーマット・エレメントは、いくつでも使用することができます。
- ブランクをフォーマット・エレメントの間に指定することはできません。前後のブランクを指定することはできますが、それらは結果のフォーマット設定で無視されます。

表 *60. VARCHAR_FORMAT (*数値から可変長文字*)* 関数のフォーマット・エレメント

フォーマット・エレメント	説明
0	各 0 は、有効数字桁を表します。数字の先行ゼロは、ゼロで表示されます。
9	各 9 は、有効数字桁を表します。数字の先行ゼロは、ブランクで表示されます。区切り文字の左に少なくとも 1 桁があるグループ区切り文字のみが生成されます。
S	接頭部: *numeric-expression* が負の数値の場合は、結果に先行の負符号 (-) が含まれます。*numeric-expression* が正数の場合は、結果に先行の正符号 (+) が含まれます。
$	接頭部: 結果に先行のドル記号 ($) が含まれます。
MI	接尾部: *numeric-expression* が負の数値の場合は、結果に末尾の負符号 (-) が含まれます。*numeric-expression* が正数の場合は、結果に末尾ブランクが含まれます。
PR	接尾部: *numeric-expression* が負の数値の場合は、結果に先行の「より小さい」の文字 (<) と末尾の「より大きい」の文字 (>) が含まれます。*numeric-expression* が正数の場合は、結果に先行スペースと末尾スペースが含まれます。
,	コンマが結果のその位置に含まれることを指定します。このコンマは、グループ分離文字として使用されます。
.	ピリオドが結果のその位置に含まれることを指定します。このピリオドは小数点として使用されます。
L	接頭部または接尾部: ローカル通貨記号が結果のその位置に含まれることを指定します。通貨記号は、ライブラリー *LIBL の中のメッセージ・ファイル QCPFMSG のメッセージ CPX8416 から検索されます。
D	ローカルの小数点文字が結果のその位置に含まれることを指定します。小数点文字は、ライブラリー *LIBL の中のメッセージ・ファイル QCPFMSG のメッセージ CPX8416 から検索されます。
G	ローカルのグループ区切り文字が結果のその位置に含まれることを指定します。ライブラリー *LIBL の中のメッセージ・ファイル QCPFMSG のメッセージ CPX8416 から検索されたローカルの小数点文字がピリオドである場合、グループ区切り文字はコンマになります。ローカルの小数点文字がコンマの場合、グループ区切り文字はピリオドになります。

format-string が指定されていなければ、この関数は VARCHAR(*numeric-expression*) と同等です。

結果は、*numeric-expression* 値 (丸められる場合もあります) を *format-string* で指定されたフォーマットで表現したものです。小数点の右側の桁数が *format-string* 内の小数点の右側の桁フォーマット・エレメント (「0」または「9」) の数より大きい場合、フォーマット設定の前に、*numeric-expression* の値が ROUND 関数によって丸められます。 *format-string* は、以下の規則に従って、この *rounded-input-value* に適用されます。

- 以下の条件のすべてを満たす場合、結果に含まれる小数点の左側には数字がありません。
  - -1 < *rounded-input-value* < 1
  - *format-string* に、小数点の左側の「0」フォーマット・エレメントが含まれていない。
  - *format-string* に、小数点の右側の桁フォーマット・エレメント (「0」または「9」) が含まれている。
- 以下の条件をすべて満たす場合、結果に含まれる暗黙的または明示的小数点の直前に、単一の 0 文字が組み込まれます。
  - *rounded-input-value* の値が 0 または -0 である。
  - *format-string* に、暗黙的または明示的小数点の左側の「9」桁フォーマット・エレメントだけが含まれる。
  - *format-string* に、小数点の右側の桁フォーマット・エレメントが含まれていない。
- *format-string* に、小数点の左側の「0」と「9」の両方のフォーマット・エレメントが含まれる場合、フォーマット・ストリングの左側からの最初の桁フォーマット・エレメントの位置により、先行ブランクまたはゼロの有無が決まります。暗黙的または明示的小数点の左側で、一番左の「0」フォーマット・エレメントの後に指定されたすべての「9」フォーマット・エレメントは、「0」フォーマット・エレメントが指定された場合と同じように扱われます。例えば、*format-string* 値「99099」は、値「99000」と同じです。
- *rounded-input-value* に含まれる小数点の右側の桁数が、*format-string* に含まれる小数点の右側の桁フォーマット・エレメントの数より少ない場合、結果には、*format-string* に含まれる小数点右側の桁フォーマット・エレメントの数に対応する数字の桁数になるまで右側にゼロが埋め込まれます。
- *rounded-input-value* に含まれる小数点の左側の桁数が、*format-string* に含まれる小数点の左側の桁フォーマット・エレメントの数より多い場合、結果は、*format-string* が結果で有効な値として生成する長さと一致する番号記号 (#) 文字のストリングになります。
- *rounded-input-value* の値が正または負の特殊値 Infinity、sNaN、または NaN を表す場合、*format-string* で指定されたフォーマットを使用せずに、ストリング「INFINITY」、「SNAN」、「NAN」、「-INFINITY」、「-SNAN」、または「-NAN」が返されます。 10 進浮動小数点の特殊値 sNaN は、ストリングに変換される場合、例外を生じません。
- *format-string* に符号フォーマット・エレメント「S」、「MI」、または「PR」がいずれも含まれていない場合、*rounded-input-value* の値が負であると、結果に負符号 (-) が組み込まれます。そうでなければ、結果のストリングにブランクが組

み込まれます。負符号 (-) またはブランクは、結果の小数点の左側の最初の数字の直前、または小数点の左側に数字がない場合には、小数点の直前に置かれます。

結果は、*rounded-input-value* のストリング表現です。結果のデータ・タイプは、*format-string* のデータ・タイプに基づき、可変長文字または可変長グラフィックです。単一の引数が指定されている場合、長さ属性は 42 です。それ以外の場合、長さ属性は 254 です。結果の実際の長さは、*format-string* (指定されている場合) によって決まります。指定されていない場合、結果の実際の長さは、*rounded-input-value* の値を表現可能な最小文字数になります。結果のストリングが結果の長さ属性より長い場合、結果は切り捨てられます。

結果の CCSID は、*format-string* の CCSID と同一です。*format-string* が指定されていない場合、結果の CCSID は現行サーバーのデフォルトの SBCS CCSID になります。

注記

代替構文: TO_CHAR は VARCHAR_FORMAT の同義語です。

例

例: タイム・スタンプから可変長文字

- 文字変数 *TVAR* を、CORPDATA.IN_TRAY からの RECEIVED のタイム・スタンプ値のストリング表現 (YYYY-MM-DD HH24:MI:SS としてフォーマット) に設定します。

```
SELECT VARCHAR_FORMAT(RECEIVED,'YYYY-MM-DD HH24:MI:SS')
  INTO :TVAR
  FROM CORPDATA.IN_TRAY
  WHERE SOURCE = 'CHAAS'
```

以下のストリングを戻します。

```
1988-12-22 14:07:21
```

RECEIVED 列内の値が 2000 年の開始の 1 秒前 (December 31, 1999 at 23:59:59pm) とすると、次のストリングが戻されます。

```
1999-12-31 23:59:59
```

フォーマット・ストリングで HH24 の代わりに HH12 が指定されていた場合は、結果が異なります。

```
1999-12-31 11:59:59
```

例: タイム・スタンプから可変長文字

- 変数 *TMSTAMP* が TIMESTAMP として定義されており、2007-03-09-14.07.38.123456 という値があるとします。以下の各例では、いくつかの関数の呼び出しと結果のストリング値が示されています。各ケースの結果のデータ・タイプは VARCHAR(255) です。

```
Function invocation                          Result
-------------------                          ------
VARCHAR_FORMAT(TMSTAMP,'YYYYMMDDHHMISSFF3')  20070309020738123
VARCHAR_FORMAT(TMSTAMP,'YYYYMMDDHH24MISS')   20070309140738
VARCHAR_FORMAT(TMSTAMP,'YYYYMMDDHHMI')       200703090207
```

```
VARCHAR_FORMAT(TMSTAMP,'DD/MM/YY')          09/03/07
VARCHAR_FORMAT(TMSTAMP,'MM-DD-YYYY')        03-09-2007
VARCHAR_FORMAT(TMSTAMP,'J')                 2454169
VARCHAR_FORMAT(TMSTAMP,'Q')                 1
VARCHAR_FORMAT(TMSTAMP,'W')                 2
VARCHAR_FORMAT(TMSTAMP,'IW')                10
VARCHAR_FORMAT(TMSTAMP,'WW')                10
VARCHAR_FORMAT(TMSTAMP,'Month')             March
VARCHAR_FORMAT(TMSTAMP,'MONTH')             MARCH
VARCHAR_FORMAT(TMSTAMP,'MON')               MAR
```

例: タイム・スタンプから可変長文字

- 変数 *DTE* が DATE として定義されており、その値が「2007-03-09」であるとします。以下の各例では、いくつかの関数の呼び出しと結果のストリング値が示されています。各ケースの結果のデータ・タイプは VARCHAR(255) です。

```
Function invocation                         Result
-------------------                         ------
VARCHAR_FORMAT(DTE,'YYYYMMDD')              20070309
VARCHAR_FORMAT(DTE,'YYYYMMDDHH24MISS')      20070309000000
```

例: タイム・スタンプから可変長文字

- タイム・スタンプの指定したストリング表現の時間をフォーマットします。

```
SELECT
  VARCHAR_FORMAT(TIMESTAMP('1979-04-07-14.00.00.000000'), 'HH'),
  VARCHAR_FORMAT(TIMESTAMP('1979-04-07-14.00.00.000000'), 'HH12')
  VARCHAR_FORMAT(TIMESTAMP('1979-04-07-14.00.00.000000'), 'HH24'),
  VARCHAR_FORMAT(TIMESTAMP('2000-01-01-00.00.00.000000'), 'HH'),
  VARCHAR_FORMAT(TIMESTAMP('2000-01-01-12.00.00.000000'), 'HH'),
  VARCHAR_FORMAT(TIMESTAMP('2000-01-01-24.00.00.000000'), 'HH'),
  VARCHAR_FORMAT(TIMESTAMP('2000-01-01-00.00.00.000000'), 'HH12'),
  VARCHAR_FORMAT(TIMESTAMP('2000-01-01-12.00.00.000000'), 'HH12'),
  VARCHAR_FORMAT(TIMESTAMP('2000-01-01-24.00.00.000000'), 'HH12'),
  VARCHAR_FORMAT(TIMESTAMP('2000-01-01-00.00.00.000000'), 'HH24'),
  VARCHAR_FORMAT(TIMESTAMP('2000-01-01-12.00.00.000000'), 'HH24'),
  VARCHAR_FORMAT(TIMESTAMP('2000-01-01-24.00.00.000000'), 'HH24')
FROM SYSIBM.SYSDUMMY1;
```

この SELECT ステートメントは以下の値を戻します。

```
'02'  '02'  '14'  '12'  '12'  '12'  '12'  '12'  '12'  '00'  '12'  '24'
```

例: 数値から可変長文字

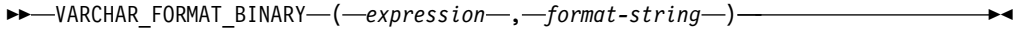
- 変数 POSNUM および NEGNUM が DECFLOAT(34) として定義され、それぞれ値 '1234.56' および '-1234.56' を持っているとします。以下の各例では、いくつかの関数の呼び出しと結果のストリング値が示されています。

```
Function invocation                         Result
------------------------                    -----------
VARCHAR_FORMAT(POSNUM)                      '1234.56'
VARCHAR_FORMAT(NEGNUM)                      '-1234.56'

VARCHAR_FORMAT(POSNUM,'9999.99')            ' 1234.56'
VARCHAR_FORMAT(NEGNUM,'9999.99')            '-1234.56'

VARCHAR_FORMAT(POSNUM,'99999.99')           '  1234.56'
VARCHAR_FORMAT(NEGNUM,'99999.99')           ' -1234.56'

VARCHAR_FORMAT(POSNUM,'00000.00')           ' 01234.56'
VARCHAR_FORMAT(NEGNUM,'00000.00')           '-01234.56'

VARCHAR_FORMAT(POSNUM,'9999.99MI')          '1234.56 '
```

```
VARCHAR_FORMAT(NEGNUM,'9999.99MI')         '1234.56-'

VARCHAR_FORMAT(POSNUM,'S9999.99')          '+1234.56'
VARCHAR_FORMAT(NEGNUM,'S9999.99')          '-1234.56'

VARCHAR_FORMAT(POSNUM,'9999.99PR')         ' 1234.56 '
VARCHAR_FORMAT(NEGNUM,'9999.99PR')         '<1234.56>'

VARCHAR_FORMAT(POSNUM,'S$9,999.99')        '+$1,234.56'
VARCHAR_FORMAT(NEGNUM,'S$9,999.99')        '-$1,234.56'
```

VARCHAR_FORMAT_BINARY

VARCHAR_FORMAT_BINARY 関数は、*format-string* を使用してフォーマットされたビット・ストリングの文字ストリング表現を戻します。



expression
組み込みバイナリー・ストリングまたは文字 FOR BIT DATA ストリングを戻す式。 *expression* の長さは、*format-string* 内の 'x' または 'X' の文字数を 2 で割った数に等しくなければなりません。

format-string
組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプを返す式。*format-string* には、*expression* の値をフォーマットする方法に関するテンプレートが含まれています。

有効なフォーマット・ストリングは 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx' と 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX' です。ここで、それぞれの 'x' または 'X' は、*expression* の 1 つの 16 進数字に対応します。 'x' が指定されている場合、対応する 16 進数字に対して戻される文字は小文字になります。それ以外の場合、戻される文字は大文字になります。

この関数の結果は、フォーマット・ストリングに基づいた長さ属性および実際の長さを持つ可変長文字ストリングです。 2 つの有効なフォーマット・ストリングの場合、結果の長さ属性と実際の長さは 36 になります。引数のどちらかが NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数のどちらかが NULL である場合は、結果は NULL 値になります。

結果の CCSID は *format-string* の CCSID です。

注記

代替構文: VARCHAR_FORMAT_BIT は VARCHAR_FORMAT_BINARY の同義語です。

例

- UUID (Universal Unique Identifier) をその定様式で表す

```
VALUES VARCHAR_FORMAT_BINARY(BX'd83d6360181811db9804b622a1ef5492',
                             'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx')
```

戻される結果:

```
'd83d6360-1818-11db-9804-b622a1ef5492'
```

- UUID (Universal Unique Identifier) をその定様式で表す

```
VALUES VARCHAR_FORMAT_BINARY(BX'd83d6360181811db9804b622a1ef5492',
                             'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX')
```

戻される結果:

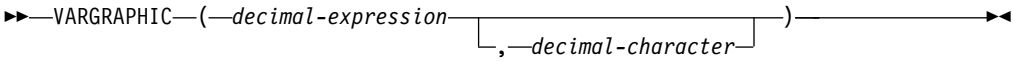
```
'D83D6360-1818-11DB-9804-B622A1EF5492'
```

VARGRAPHIC

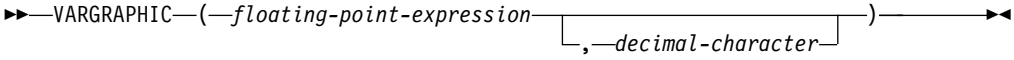
VARGRAPHIC 関数は、グラフィック・ストリング表現を返します。

整数から可変長グラフィックに

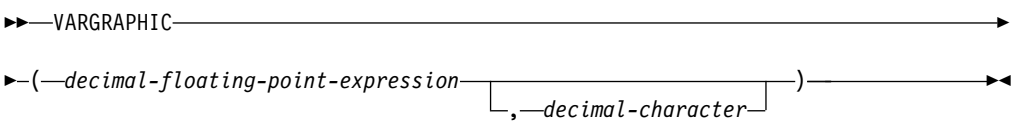
10 進数から可変長グラフィックに



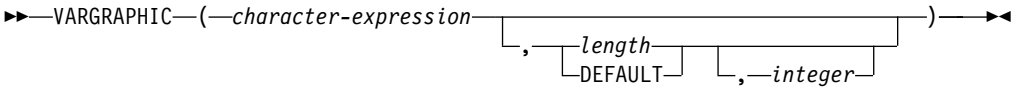
浮動小数点数から可変長グラフィックに



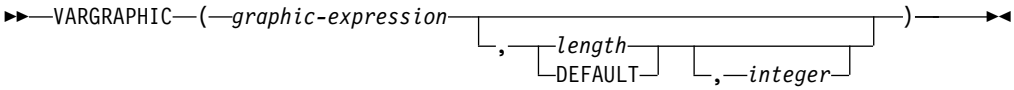
10 進浮動小数点数から可変長グラフィックに



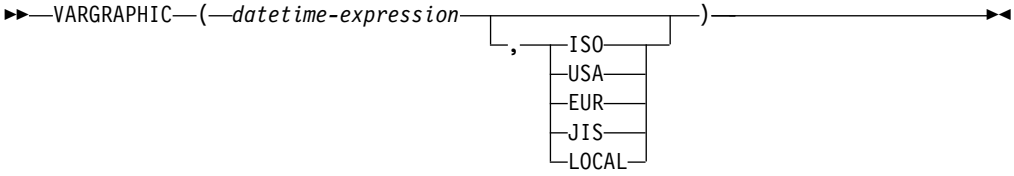
文字から可変長グラフィックに



グラフィックから可変長グラフィックに



日時から可変長グラフィックに



VARGRAPHIC 関数は、次のもののグラフィック・ストリング表現を戻します。

- 整数 (最初の引数が SMALLINT、INTEGER、または BIGINT の場合)
- 10 進数 (最初の引数がパックまたはゾーン 10 進数の場合)

- 倍精度浮動小数点数 (最初の引数が DOUBLE または REAL の場合)
- 最初の引数が DECFLOAT である場合は 10 進浮動小数点数。
- 文字ストリング (最初の引数が任意のタイプの文字ストリングの場合)
- グラフィック・ストリング (最初の引数が Unicode グラフィック・ストリング・ストリングの場合)
- 日付値 (最初の引数が DATE の場合)
- 時刻値 (最初の引数が TIME の場合)
- タイム・スタンプ値 (最初の引数が TIMESTAMP)

この関数の結果は可変長グラフィック・ストリング (VARGRAPHIC) です。

最初の引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。最初の引数が NULL の場合、結果は NULL 値となります。最初の引数が空ストリング、または EBCDIC ストリング `X'0E0F'` である場合は、結果は空ストリングになります。

整数から可変長グラフィックに

integer-expression
整数データ・タイプ (SMALLINT、INTEGER、または BIGINT) の値を戻す式。

結果は、SQL 整数定数の形式で引数を表現した可変長グラフィック・ストリングです。結果は、引数の値を表す n 文字の有効数字から成ります。引数が負数の場合は、負符号が前に付きます。結果は左寄せされます。

- 引数が短整数の場合は、結果の長さ属性は 6
- 引数が長整数の場合は、結果の長さ属性は 11
- 引数が 64 ビット整数の場合は、結果の長さ属性は 20

結果の実際の長さは、引数の値を表すために使用できる最小文字数です。先行ゼロは含まれません。引数が負数の場合は、結果の最初の文字は負符号になります。負符号でなければ、最初の文字は数字または *decimal-character* です。

結果の CCSID は 1200 (UTF-16) です。

10 進数から可変長グラフィックに

decimal-expression
パックまたはゾーン 10 進数データ・タイプ (DECIMAL または NUMERIC) の値を戻す式。精度や位取りを変えたい場合は、DECIMAL スカラー関数を使用して変更することができます。

decimal-character
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

結果は、引数を可変長グラフィック・ストリングで表現したものになります。この結果には、1 文字の小数点文字と *p* 桁までの数字が含まれます。 *p* は *10* 進数式

の精度で、引数が負数の場合は負符号が先頭に付きます。先行ゼロは戻されません。後続ゼロは戻されます。*decimal-expression* の位取りがゼロの場合、小数点文字は戻されません。

結果の長さ属性は 2+*p* です (*p* は *decimal-expression* の精度)。結果の実際の長さは、引数の値を表すために使用できる最小文字数ですが、ただし、後書き文字も含まれます。先行ゼロは含まれません。引数が負数の場合は、結果の最初の文字は負符号になります。負符号でなければ、結果の最初の文字は数字または *decimal-character* になります。

結果の CCSID は 1200 (UTF-16) です。

浮動小数点数から可変長グラフィックに

浮動小数点数式
浮動小数点データ・タイプ (DOUBLE または REAL) の値を戻す式。

decimal-character
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

結果は、浮動小数点定数の形式で引数を可変長グラフィック・ストリングで表現したものになります。

結果の長さ属性は、24 です。結果の実際の長さは、ゼロ以外の 1 桁の数字、その後ろに 1 つの小数点文字 と一連の数字が続く小数部の引数の値を表す最小文字数です。引数が負数の場合は、結果の最初の文字は負符号になります。負数でなければ、最初の文字は数字または *decimal-character* です。引数がゼロであると、結果は 0E0 になります。

結果の CCSID は 1200 (UTF-16) です。

10 進浮動小数点数から可変長グラフィックに

decimal-floating-point expression
10 進浮動小数点データ・タイプの値を返す式。

decimal-character
結果の文字ストリングにおいて、小数点以下を区切るために使用する 1 バイト文字の定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

結果は、10 進浮動小数点定数の形式で引数を可変長グラフィック・ストリングで表現したものになります。

結果の長さ属性は、42 です。 結果の実際の長さは、記号、数字、および *decimal-character* を含む、引数の値を表す最小文字数です。後続ゼロは有効数字です。引数が負数の場合は、結果の最初の文字は負符号になります。負数でなければ、最初の文字は数字または *decimal-character* です。引数がゼロであると、結果は 0 になります。

DECFLOAT 値が Infinity、sNaN、または NaN の場合、それぞれストリング 'INFINITY'、'SNAN'、および 'NAN' が返されます。特殊値が負の場合、負符号 (-) がそのストリングの最初の文字となります。 DECFLOAT 特殊値 sNaN を使用しても、ストリングへの変換時に例外は発生しません。

結果の CCSID は 1200 (UTF-16) です。

文字から可変長グラフィックに

character-expression
文字ストリング式を指定します。CHAR または VARCHAR ビット・データであってはなりません。

length
結果の長さ属性を指定する整数定数。これは、最初の引数がNULL 可能でない場合は、1 から 16370 の範囲の整数定数でなければならず、最初の引数が NULL 可能である場合は、1 から 16369 の範囲の整数定数でなければなりません。

2 番目の引数を指定しなかった場合、または DEFAULT を指定した場合は、結果の長さ属性は最初の引数の長さ属性と同じになります。ただし、式が空ストリングまたは EBCDIC ストリング X'0E0F' である場合は結果の長さ属性は 1 です。

結果の実際の長さは、結果の長さ属性と *character-expression* の実際の長さのうち、小さい方になります。引数の各文字ごとに、結果の 1 文字が決まります。*character-expression* の長さ (文字数) が結果の長さ属性よりも大きい場合は、切り捨てが行われます。切り捨てられた文字がすべてブランクであった場合以外は、警告 (SQLSTATE 01004) が戻されます。

integer
結果の CCSID を指定する整数定数。これは DBCS、UTF-16、または UCS-2 の CCSID でなければなりません。 CCSID が 65535 であることはできません。 CCSID が Unicode グラフィック・データを表す場合は、引数の各文字によって結果の 1 文字が決まります。結果の n 番目の文字は、引数の n 番目の文字と等価の UTF-16 または UCS-2 文字になります。

整数 が指定されていない場合、結果の CCSID は混合 CCSID によって決まります。 M でその混合 CCSID を示すことにします。

以下の規則では、S は次のいずれかを指します。

- ストリング式が外部コード化スキームのデータを含むホスト変数である場合は、データを固有コード化スキームの CCSID に変換した後の式の結果が S。 (詳しくは、 39 ページの『文字変換』を参照してください。)
- ストリング式が固有コード化スキームのデータである場合は、そのストリング式が S。

M は次のように決まります。

- S の CCSID が 1208 (UTF-8) である場合、M は 1200 (UTF-16) になる。
- S の CCSID が混合 CCSID である場合は、M はその CCSID になる。
- S の CCSID が SBCS CCSID である場合 :

- S の CCSID が関連する混合 CCSID を持つ場合は、M はその CCSID になる。
- それ以外の場合は、演算ができない。

次の表には、M をもとにした結果の CCSID を要約してあります。

M	結果の **CCSID**	説明	**DBCS** 置換文字
930	300	日本語 EBCDIC	X'FEFE'
933	834	韓国語 EBCDIC	X'FEFE'
935	837	中国語 (簡体字) EBCDIC	X'FEFE'
937	835	中国語 (繁体字) EBCDIC	X'FEFE'
939	300	日本語 EBCDIC	X'FEFE'
5026	4396	日本語 EBCDIC	X'FEFE'
5035	4396	日本語 EBCDIC	X'FEFE'

SBCS と DBCS が等価になるかどうかは M によって決まります。CCSID に関係なく、引数の中の 2 バイトのコード・ポイントはすべて DBCS 文字と見なされ、引数の中の 1 バイトのコード・ポイントはすべて SBCS 文字と見なされます (ただし、EBCDIC 混合データのシフト・コード `X'0E'` および `X'0F'` は例外)。

- 引数の n 番目の文字が DBCS 文字である場合は、結果の n 番目の文字はその DBCS になる。
- 引数の n 番目の文字が、等価の DBCS 文字を持つ SBCS 文字である場合は、結果の n 番目の文字は、その等価の DBCS 文字になる。
- 引数の n 番目の文字が、等価の DBCS 文字を持たない SBCS 文字である場合は、結果の n 番目の文字は、DBCS 置換文字になる。

グラフィックから可変長グラフィックに

graphic-expression
グラフィック・ストリングの値を戻す式。

length
結果の長さ属性を指定する整数定数。これは、最初の引数がNULL 可能でない場合は、1 から 16370 の範囲の整数定数でなければならず、最初の引数が NULL 可能である場合は、1 から 16369 の範囲の整数定数でなければなりません。

2 番目の引数を指定しなかった場合、または DEFAULT を指定した場合は、結果の長さ属性は最初の引数の長さ属性と同じになります。ただし、式が空ストリングである場合は結果の長さ属性は 1 です。

結果の実際の長さは、結果の長さ属性と *graphic-expression* の実際の長さのうち、小さい方になります。引数の各文字ごとに、結果の 1 文字が決まります。*graphic-expression* の長さ (文字数) が結果の長さ属性よりも大きい場合は、切り捨てが行われます。切り捨てられた文字がすべてブランクであった場合以外は、警告 (SQLSTATE 01004) が戻されます。

integer
結果の CCSID を指定する整数定数。これは DBCS、UTF-16、または UCS-2 の CCSID でなければなりません。 CCSID が 65535 であることはできません。

整数 が指定されていない場合、結果の CCSID は、最初の引数の CCSID になります。

日時から可変長グラフィックに

datetime-expression
次の 3 つの組み込みデータ・タイプのいずれかである式。

日付　結果は、2 番目の引数によって指定された形式の日付の可変長 GRAPHIC ストリング表記になります。 2 番目の引数を指定しなかった場合は、デフォルトの日付形式が使用されます。形式として ISO、USA、 EUR、または JIS を指定すると、長さ属性と結果の実際の長さは 10 になります。その他の場合は、長さ属性と結果の実際の長さはデフォルトの日付形式の長さになります。詳しくは、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

時刻　結果は、2 番目の引数によって指定された形式の時刻の可変長 GRAPHIC ストリング表記になります。 2 番目の引数を指定しなかった場合は、デフォルトの時刻形式が使用されます。長さ属性と結果の実際の長さは 8 になります。詳しくは、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

timestamp
2 番目の引数は適用されないので、指定してはなりません。

結果は、タイム・スタンプの可変長 GRAPHIC ストリング表記になります。 *datetime-expression* が TIMESTAMP(0) の場合、結果の長さ属性および実際の長さは 19 です。*datetime-expression* のデータ・タイプが TIMESTAMP(*n*) の場合、結果の長さ属性および実際の長さは 20+*n* です。それ以外の場合、結果の長さ属性および実際の長さは 26 です。

結果の CCSID は 1200 (UTF-16) です。

ISO、EUR、USA、または JIS
結果のグラフィック・ストリングの日付形式または時刻形式を指定します。詳しくは、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

LOCAL
結果のグラフィック・ストリングの日付または時刻の形式を、現行サーバーのジョブの DATFMT、DATSEP、 TIMFMT、および TIMSEP 属性から取る必要があることを指定します。

注記

代替構文**:** 最初の引数がストリングで、長さ属性を指定する場合、アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、 218 ページの『CAST の指定』を参照してください。

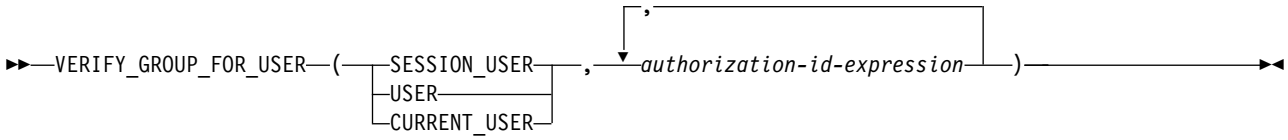
例

- 表 EMPLOYEE を使用して、ホスト変数 VAR_DESC (VARGRAPHIC(24)) を従業員番号 (EMPNO)「000050」に対応する氏名の名 (FIRSTNME) と等価の VARGRAPHIC にセットします。

```
SELECT VARGRAPHIC(FIRSTNME)
  INTO :VAR_DESC
  FROM EMPLOYEE
  WHERE EMPNO = '000050'
```

VERIFY_GROUP_FOR_USER

VERIFY_GROUP_FOR_USER 関数は、指定されたユーザーが、ユーザー・プロファイル・リスト中にあるか、または、 *authorization-id-expression* 引数のリストで指定されたグループ・ユーザー・プロファイルのいずれかのメンバーであるかを示す値を戻します。



SESSION_USER または **USER** または **CURRENT_USER**
許可 ID を指定します。

authorization-id-expression
許可名を指定する式。許可名が現行サーバーにあるかどうかは確認されません。*authorization-id-expression* は、文字ストリングまたはグラフィック・ストリング組み込みデータ・タイプの値を戻す式でなければなりません。各 *authorization-id-expression* の値は、1 以上、10 以下の長さでなければなりません。

この関数の結果は長精度整数になります。結果が NULL になることはありません。

結果の値は、最初の引数で表される許可 ID が *authorization-id-expression* のリストのどこかにある場合、1 です。それ以外の場合、結果は 0 です。

VERIFY_GROUP_FOR_USER 関数は、接続内では deterministic 関数となり、接続間では非 deterministic 関数となります。また、CREATE MASK ステートメントまたは CREATE PERMISSION ステートメント内で、データへのアクセス権を検証するためにこれを参照できます。

例

表 EMPLOYEE が存在し、この表に対して列レベルのアクセス制御がアクティブになっていると想定します。Alex (QIBM_DB_SECADM 権限を持っている) は、社会保障番号についての情報の要求に対してどのような情報を戻すのかを、誰が情報を要求しているのかに基づいて制御するために、以下の列マスクを作成しました。この列マスクは、セッション・ユーザーが MGR グループ・プロファイルのメンバーである場合のみ、実際の社会保障番号を戻します。そうでない場合、社会保障番号は一部を隠した表現で戻されます。

```
CREATE MASK SSN_MASK
       ON EMPLOYEE
       FOR COLUMN SSN
   RETURN
     CASE
       WHEN VERIFY_GROUP_FOR_USER(SESSION_USER,'MGR') = 1
            THEN SSN
       ELSE 'XXX-XX' CONCAT SUBSTR(SSN, 8,4)
     END
   ENABLE;
```

ALTER TABLE ステートメントが発行され、EMPLOYEE 表で列マスクがアクティブ化されます。

```
ALTER TABLE EMPLOYEE
     ACTIVATE COLUMN ACCESS CONTROL;
COMMIT;
```

Mary はマネージャーであり、MGR グループ・プロファイルのメンバーであると想定します。Mary は次のステートメントを発行します。

```
SELECT SSN FROM EMPLOYEE WHERE NAME = 'Tom';
```

結果表を生成するために、SSN_MASK 列マスクが SSN 列に適用されます。Mary は MGR グループ・プロファイルのメンバーであるため、結果表には Tom の実際の社会保障番号が含まれます。

この後、 Mary はマネージャーではなくなり、MGR グループ・プロファイルから除去されます。彼女が同じ照会をもう一度発行します。

```
SELECT SSN FROM EMPLOYEE WHERE NAME = 'Tom';
```

前のように、SSN_MASK 列マスクが SSN 列に適用されて結果表が生成されます。今回は、結果表に含まれるのは、一部が隠されたバージョンの Tom の社会保障番号です。実際の番号の末尾 4 桁のみが戻され、他の桁には「X」が戻されます。

WEEK

WEEK 関数は、年間通算週を表す 1 から 54 までの範囲の整数を戻します。週は日曜日から始まります。1 月 1 日は常に第 1 週に入ります。

expression
日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式。

expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

この関数の結果は長精度整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

- 表 PROJECT を使用して、ホスト変数 WEEK (INTEGER) をプロジェクト ('PL2100') が終了した週にセットします。

```
SELECT WEEK(PRENDATE)
  INTO :WEEK
  FROM PROJECT
  WHERE PROJNO = 'PL2100'
```

  結果として、WEEK は 38 にセットされます。

- 表 X に DATE_1 という名前の DATE 列があり、以下のリストに示すような日付が入っているとします。

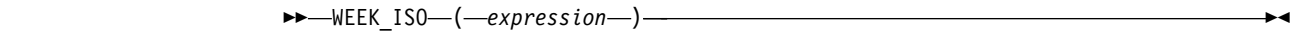
```
SELECT DATE_1, WEEK(DATE_1)
  FROM X
```

  結果として、各日付について WEEK 関数が戻した値を示す以下のようなリストが表示されます。

```
1997-12-28    53
1997-12-31    53
1998-01-01    1
1999-01-01    1
1999-01-04    2
1999-12-31    53
2000-01-01    1
2000-01-03    2
```

WEEK_ISO

WEEK_ISO 関数は、年間通算週を表す 1 から 53 までの範囲の整数を戻します。週は月曜日から始まります。週 1 は、木曜日が含まれるその年の最初の週を表します。つまり、1 月 4 日が含まれる最初の週と同じことです。したがって、年初の最高 3 日間が前年の最後の週と見なされたり、年末の最高 3 日間が来年の最初の週と見なされたりする可能性があります。



expression
日付、タイム・スタンプ、文字ストリング、またはグラフィック・ストリングのいずれかの組み込みデータ・タイプの値を戻す式。

expression が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。

この関数の結果は長精度整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

- 表 PROJECT を使用して、ホスト変数 WEEK (INTEGER) をプロジェクト ('AD2100') が終了した週にセットします。

```
SELECT WEEK_ISO(PRENDATE)
  INTO :WEEK
  FROM PROJECT
  WHERE PROJNO = 'AD3100'
```

  結果として、WEEK は 5 にセットされます。

- 表 X に DATE_1 という名前の DATE 列があり、以下のリストに示すような日付が入っているとします。

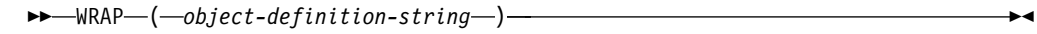
```
SELECT DATE_1, WEEK_ISO(DATE_1)
  FROM X
```

  結果は以下のようになります。

```
1997-12-28    52
1997-12-31    1
1998-01-01    1
1999-01-01    53
1999-01-04    1
1999-12-31    52
2000-01-01    52
2000-01-03    1
```

WRAP

WRAP 関数は、判読可能な DDL ステートメントを、難読化された DDL ステートメントに変換します。



難読化された DDL ステートメント内の手順ロジックおよび組み込み SQL ステートメントは、ロジックに含まれる知的財産を簡単には抽出できないように、暗号化されます。

スキーマは SYSIBMADM です。

object-definition-string
DDL ステートメントを含んでいる、タイプ CLOB または DBCLOB のストリング。以下のいずれかの SQL ステートメントを指定できます。

- CREATE FUNCTION (SQL スカラー)
- CREATE FUNCTION (SQL 表)
- CREATE PROCEDURE (SQL)
- CREATE TRIGGER

結果は、エンコードされたバージョンの入力ステートメントを含んでいる、タイプ CLOB(2M) のストリングです。結果が NULL になることはありません。エンコードは、元のステートメントのままにしたルーチン・シグニチャーまたはトリガー名までの接頭部と、その後に続くキーワード WRAPPED から構成されます。このキーワードの後には、関数を起動したアプリケーション・サーバーについての情報が続きます。この情報の形式は *pppvvrrm* で、各部の意味は次のとおりです。

- *ppp* は、以下の 3 文字を使用して製品を示します。
  - DSN (Db2 for z/OS の場合)
  - QSQ (Db2 for i の場合)
  - SQL (Db2 for LUW の場合)
- *vv* は、2 桁のバージョン ID です (例えば、'07' など)。
- *rr* は、2 桁のリリース ID です (例えば、'02' など)。
- *m* は、1 文字の修正レベル ID です (例えば、'0' など)。

例えば、Db2 for i バージョン 7.3 は、「QSQ07030」で示されます。

このアプリケーション・サーバー情報の後に、文字 (a から z および A から Z)、数字 (0 から 9)、下線、およびコロンからなるストリングが続きます。

エンコードされた DDL ステートメントは、平文形式のステートメントよりも最大 3 分の 1 だけ長くなる可能性があります。結果が SQL ステートメントの最大長を超える場合、エラーが発行されます。

注記

ステートメントのエンコードは、内容を難読化することを意図したものであり、強い暗号化の一種と考えるべきではありません。

例

例 1: 週 40 時間労働として時間給から年間給与を計算する関数の難読化版を生成します。

```
VALUES WRAP('CREATE FUNCTION salary(wage DECFLOAT) RETURNS DECFLOAT
                      RETURN wage * 40 * 52')
```

このステートメントの結果は、例えば次のようになります。

```
CREATE FUNCTION salary(wage DECFLOAT) WRAPPED QSQ07020 <encoded-suffix>
```

例 2: 複雑なデフォルトを設定するトリガーの難読化版を生成します。

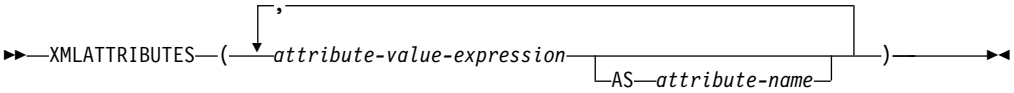
```
VALUES WRAP('CREATE OR REPLACE TRIGGER trig1 BEFORE INSERT ON emp
                      REFERENCING NEW AS n FOR EACH ROW
                      WHEN (n.bonus IS NULL) SET n.bonus = n.salary * .04')
```

このステートメントの結果は、例えば次のようになります。

```
CREATE TRIGGER trig1 WRAPPED QSQ07020 <encoded-suffix>
```

XMLATTRIBUTES

XMLATTRIBUTES 関数は、引数から XML 属性を構成します。



この関数は XMLELEMENT 関数の引数としてのみ使用できます。結果は、NULL 以外の *attribute-value-expression* 引数ごとに XML 属性が含まれる XML シーケンスです。

attribute-value-expression
結果が属性値になる式。*attribute-value-expression* のデータ・タイプは、ROWID、DATALINK、XML、および ROWID、DATALINK、XML に基づく特殊タイプであってはなりません。式として、SQL 式を指定することもできます。式が単純な列参照でない場合は、属性名を指定する必要があります。

attribute-name
属性名を指定します。名前は XML 修飾名の形式 (つまり、QName) の SQL ID でなければなりません。有効な名前についての詳細は、W3C XML ネーム・スペースの指定の項目を参照してください。属性名を「xmlns」にしたり、その前に「xmlns:」を付けたりすることはできません。ネーム・スペースは、関数 XMLNAMESPACES を使用して宣言します。(暗黙的であれ明示的であれ) 重複した属性名を使用することはできません。

attribute-name を指定しない場合、*attribute-value-expression* は列名にする必要があります。属性名は、列名から XML 属性名への完全エスケープ・マッピングを使用して、列名から作成されます。

この関数の結果は XML です。いずれかの *attribute-value-expression* の結果が NULL になる可能性がある場合は、結果も NULL になる可能性があります。すべての *attribute-value-expression* の結果が NULL の場合は、結果は NULL 値になります。

例

注: XMLATTRIBUTES は、出力の中にブランク・スペースまたは改行文字を挿入しません。読みやすくするために、すべての出力例はフォーマット設定されています。

- エレメントを、属性付きで生成します。

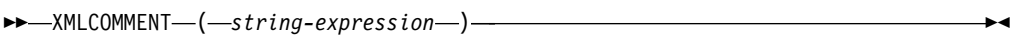
```
SELECT E.EMPNO, XMLELEMENT(
  NAME "Emp",
  XMLATTRIBUTES(
   E.EMPNO, E.FIRSTNME ||' '|| E.LASTNAME AS "name"
    )
   )
 AS "Result"
  FROM EMPLOYEE E
  WHERE E.EDLEVEL = 12
```

この照会は下記の結果を生成します。

```
EMPNO   Result
000290  <Emp EMPNO="000290" name="JOHN PARKER"/>
000310  <Emp EMPNO="000310" name="MAUDE SETRIGHT"/>
200310  <Emp EMPNO="200310" name="MICHELLE SPRINGER"/>
```

XMLCOMMENT

XMLCOMMENT 関数は、内容が入力引数の XML 値を戻します。



string-expression
任意の組み込み文字ストリング、またはグラフィック・ストリングのデータ・タイプの値を戻す式。CHAR または VARCHAR ビット・データであってはなりません。*string-expression* の結果は構文解析されて、以下の規則に定められた XML コメントの内容に準拠しているかどうかが検査されます。

- 隣接する 2 つのハイフン (「--」) がストリング式の中にあってはならない。
- ストリング式はハイフン (「-」) で終わってはならない。
- ストリングの各文字は、サロゲート・ブロック X'FFFE' および X'FFFF' を除く、任意の Unicode 文字とすることができる。[77]

string-expression が上記の規則に準拠していない場合は、エラーが戻されます。

この関数の結果は XML です。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

例

- XML コメントを生成します。

```
SELECT XMLCOMMENT('This is an XML comment')
  FROM SYSIBM.SYSDUMMY1
```

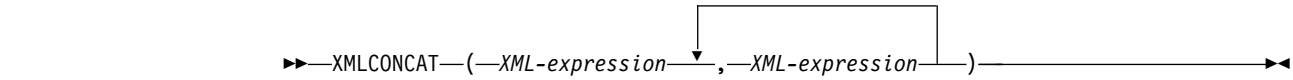
この照会は下記の結果を生成します。

```
<!--This is an XML comment-->
```

77. 有効な Unicode 文字の Unicode コード・ポイントは、#x9、 #xA、#xD、#x20-#xD7FF、#xE000-#xFFFD、#x10000-#x10FFFF です。

XMLCONCAT

XMLCONCAT 関数は、可変数の XML 入力引数を連結したものを含むシーケンスを戻します。



XML-expression
XML 値を戻す式。

この関数の結果は、NULL 以外の入力 XML 値の連結を含む XML シーケンスです。入力の中の NULL 値は無視されます。

この関数の結果は XML です。結果は NULL になる可能性があります。 すべての入力値の結果が NULL の場合、結果は NULL 値となります。

例

注: XMLCONCAT は、出力の中にブランク・スペースまたは改行文字を挿入しません。読みやすくするために、すべての出力例はフォーマット設定されています。

- 従業員ごとに「first」と「last」のエレメント名を使用して、ファーストネームとラストネームのエレメントを連結します。

```
SELECT XMLSERIALIZE(
      XMLCONCAT(
        XMLELEMENT(NAME "first", e.firstnme),
        XMLELEMENT(NAME "last", e.lastname)
       ) AS VARCHAR(100) ) AS "result"
FROM EMPLOYEE E
WHERE e.lastname = 'SMITH'
```

照会の結果は、次の結果のようになります。

```
result
----------------------------------------
<first>DANIEL</first><last>SMITH</last>
<first>PHILIP</first><last>SMITH</last>
```

- ファーストネームによってソートされた従業員のリストが含まれる、部門 A00 と B01 の部門エレメントを構成します。この部門エレメントの直前に、紹介コメントを含めます。

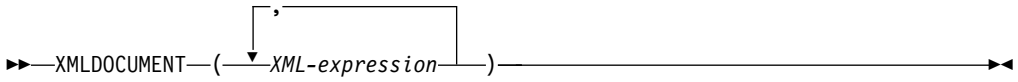
```
SELECT XMLCONCAT(
  XMLCOMMENT (
   'Confirm these employees are on track for their product schedule'),
  XMLELEMENT(
     NAME "Department",
     XMLATTRIBUTES( E.WORKDEPT AS "name"),
     XMLAGG(
       XMLELEMENT(NAME "emp", E.FIRSTNME)
       ORDER BY E.FIRSTNME)
           ))
          FROM EMPLOYEE E
  WHERE E.WORKDEPT IN ('A00', 'B01')
GROUP BY E.WORKDEPT
```

この照会は下記の結果を生成します。

```
<!--Confirm these employees are on track for their product schedule-->
<Department name="A00">
<emp>CHRISTINE</emp>
<emp>SEAN</emp>
<emp>VINCENZO</emp>
</Department>
<!--Confirm these employees are on track for their product schedule-->
<Department name="B01">
<emp>MICHAEL</emp>
</Department>
```

XMLDOCUMENT

XMLDOCUMENT 関数は、XML 値を戻します。



XML-expression
XML 値を戻す式。

この関数の結果は XML です。*XML-expression* の結果が NULL になる可能性がある場合は、結果も NULL になる可能性があります。すべての *XML-expression* が NULL の場合は、結果は NULL 値になります。

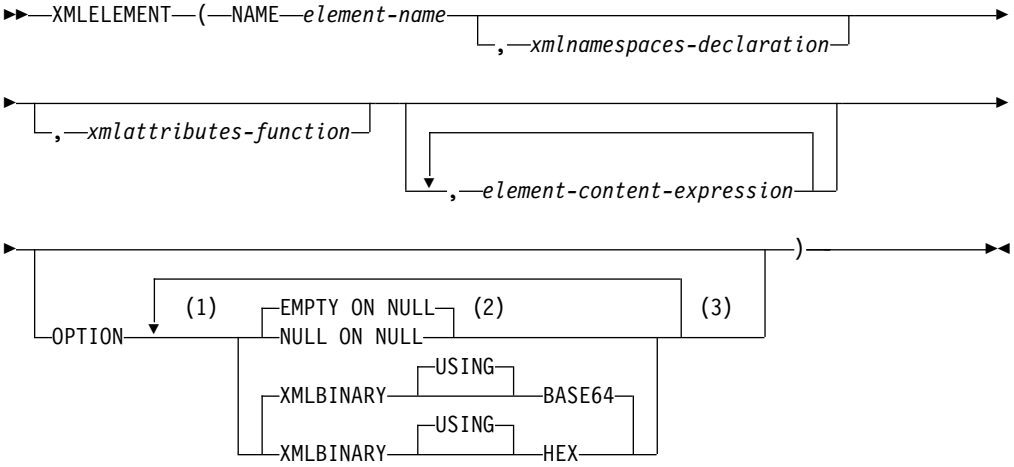
例

- 構成された文書を XML 列に挿入します。

```
INSERT INTO T1 VALUES(123,
            (SELECT XMLDOCUMENT(
                 XMLELEMENT(NAME "Emp",
                                  E.FIRSTNME || ' ' || E.LASTNAME,
                            XMLCOMMENT('This is just a simple example')
                           ))
                FROM EMPLOYEE E
                WHERE E.EMPNO = '000120'))
```

XMLELEMENT

XMLELEMENT 関数は、XML エレメントである XML 値を戻します。



注:

1 OPTION 文節を指定できるのは、少なくとも 1 つの*xmlattributes-function* または *element-content-expression* を指定した場合に限られます。

2 *element-content-expression* を指定しない場合、EMPTY ON NULL または NULL ON NULL は指定できません。

3 同じ文節を複数回指定することはできません。

NAME *element-name*

XML エレメントの名前を指定します。名前は XML 修飾名の形式 (つまり、QName) の SQL ID でなければなりません。有効な名前についての詳細は、W3C XML ネーム・スペースの指定の項目を参照してください。名前を修飾する場合は、有効範囲内にネーム・スペース接頭部が宣言されている必要があります。

xmlnamespaces-declaration

XMLNAMESPACES 宣言の結果である XML ネームス・ペース宣言を指定します。宣言されるネーム・スペースは、XMLELEMENT 関数の有効範囲内にあるものです。ネーム・スペースは、XMLELEMENT 関数内でネストされるすべての XML 関数に適用され、これらの関数が別の副選択の中で使用されるかどうかは関係ありません。 XML ネーム・スペースの宣言について詳しくは、 715 ページの『XMLNAMESPACES』を参照してください。

xmlnamespaces-declaration を指定しない場合、ネーム・スペース宣言は構成されたエレメントに関連付けられません。

xmlattributes-function

エレメントの XML 属性を指定します。属性は XMLATTRIBUTES 関数の結果です。属性の構成について詳しくは、 702 ページの『XMLATTRIBUTES』を参照してください。

element-content-expression

生成される XML エレメント・ノードの内容は、1 つの式または式のリストで指定します。この式には、ROWID または DATALINK 以外の任意の SQL デ

ータ・タイプの SQL 式を指定できます。この式は、構成されるエレメントのネーム・スペース宣言、属性、および内容を構成するために使用されます。

element-content-expression を指定しない場合、空ストリングがエレメントの内容として使われます。NULL ON NULL または EMPTY ON NULL は指定してはなりません。

OPTION

XML エレメントを構成するための追加オプションを指定します。この文節は、*element-content-expression* で指定された、ネストされた XMLELEMENT 呼び出しには影響を与えません。

EMPTY ON NULL または NULL ON NULL

すべての *element-content-expression* 値が NULL 値の場合に、NULL 値と空エレメントのどちらを戻すかを指定します。このオプションは、エレメントの内容の NULL の処理にのみ影響を与え、属性値には影響を与えません。デフォルトは EMPTY ON NULL です。

EMPTY ON NULL

各 *element-content-expression* の値が NULL の場合、空のエレメントが戻されます。

NULL ON NULL

各 *element-content-expression* の値が NULL の場合、NULL 値が戻されます。

XMLBINARY USING BASE64 または XMLBINARY USING HEX

バイナリー入力データ、FOR BIT DATA 属性を指定した文字ストリング、またはこれらのタイプのいずれかに基づく特殊タイプに対して想定されるエンコードを指定します。エンコードは要素コンテントまたは属性値に適用されます。デフォルトは XMLBINARY USING BASE64 です。

XMLBINARY USING BASE64

想定されるエンコードは、XML スキーマ・タイプ xs:base64Binary エンコードに対して定義された base64 文字であることを指定します。Base64 エンコードは US-ASCII (10 個の数字、26 個の小文字、26 個の大文字、「+」および「/」) の 65 文字サブセットを使用して、6 ビットのすべてのバイナリー・データまたはビット・データをこのサブセットの印刷可能文字で表します。これらの文字は、例外なく表示できるように選択されます。この方式を使用すると、エンコードされたデータのサイズは元のバイナリー・データまたはビット・データのサイズよりも 33 パーセント大きくなります。

XMLBINARY USING HEX

想定されるエンコードは、XML スキーマ・タイプ xs:hexBinary エンコードに対して定義された 16 進文字であることを指定します。16 進エンコードは、それぞれのバイト (8 ビット) を 2 桁の16 進文字で表します。この方式を使用すると、エンコードされたデータのサイズは元のバイナリー・データまたはビット・データのサイズの 2 倍になります。

この関数は、エレメント名、オプションのネーム・スペース宣言のコレクション、オプションの属性のコレクション、および XML エレメントの内容を構成するゼロ

個以上の引数を取ります。結果は、1 つの XML エレメント・ノードまたは NULL 値が入っている、XML シーケンスです。すべての *element-content-expression* 引数の結果が空ストリングの場合、結果の XML シーケンスには空のエレメントが入ります。

この関数の結果は XML です。結果は NULL になる可能性があります。すべての *element-content-expression* 引数値が NULL であり、NULL ON NULL オプションが有効な場合、結果は NULL 値となります。

XMLELEMENT 内でネーム・スペースを使用するための規則**:** 以下の規則は、ネーム・スペースの有効範囲について説明しています。

- XMLNAMESPACES 宣言で宣言されるネーム・スペースは、XMLELEMENT 関数によって構成されたエレメントの有効範囲内ネーム・スペースです。エレメントが直列化されている場合、それぞれの有効範囲内ネーム・スペースはネーム・スペース属性として直列化されます。ただし、そのエレメントが含まれる XML 値の有効範囲内ネーム・スペースである場合を除きます。
- これらのネーム・スペースの有効範囲は、XMLELEMENT 関数の字句範囲を表し、これには、エレメント名、XMLATTRIBUTES 関数で指定された属性名、およびすべての *element-content-expression* が含まれます。これらは、有効範囲内の QNames を解決するために使用されます。
- 構成されたエレメントの属性が *element-content-expression* から得られた場合、そのネーム・スペースはまだ、構成されたエレメントの有効範囲内ネーム・スペースとして宣言されていない場合があります。この場合は、その属性のために新しいネーム・スペースが作成されます。これによって競合が生じる場合、つまり属性名の接頭部が既に有効範囲内ネーム・スペースによって別の URI にバインドされている場合、Db2 はその属性名に使用する別の接頭部を生成します。この生成された接頭部のためにネーム・スペースが作成されます。生成される接頭部の名前は、パターン db2ns-*xx* に従います。ここで、*xx* は [A から Z、a から z、0 から 9] のセットから選択された 1 対の文字です。

例

注**:** XMLELEMENT は、出力の中にブランク・スペースまたは改行文字を挿入しません。読みやすくするために、すべての出力例はフォーマット設定されています。

以下の例では、一時 CANDIDATES 従業員表を使用します。

```
DECLARE GLOBAL TEMPORARY TABLE CANDIDATES
  (EMPNO CHAR(6),
   FIRSTNME VARCHAR(12),
   MIDINIT CHAR(1),
   LASTNAME VARCHAR(15),
   WORKDEPT CHAR(4),
   EDLEVEL INT)
INSERT INTO SESSION.CANDIDATES
   VALUES('A0001', 'John', 'A', 'Parker', 'X001', 12)
INSERT INTO SESSION.CANDIDATES
   VALUES('B0001', NULL, NULL, 'Smith', 'X001', 12)
INSERT INTO SESSION.CANDIDATES
   VALUES('B0002', NULL, NULL, NULL, 'X001', NULL)
INSERT INTO SESSION.CANDIDATES
   VALUES(NULL, NULL, NULL, NULL, 'X001', NULL)
```

- 次のステートメントは、XMLELEMENT 関数を使用して、従業員名を含む XML エレメントを作成します。このステートメントは、従業員番号も serial という属性名として保管します。参照される列に NULL 値がある場合、関数は NULL 値を戻します。

```
SELECT E.EMPNO, E.FIRSTNME, E.LASTNAME,
       XMLELEMENT(NAME "foo:Emp",
                  XMLNAMESPACES('http://www.foo.com' AS "foo"),
                  XMLATTRIBUTES(E.EMPNO AS "serial"),
                  E.FIRSTNME, E.LASTNAME
                  OPTION NULL ON NULL) AS "Result"
FROM SESSION.CANDIDATES E
```

この照会は下記の結果を生成します。

```
EMPNO  FIRSTNME   LASTNAME  Result
------ ---------  --------  ------------------
A0001  John       Parker    <foo:Emp xmlns:foo="http://www.foo.com"
                                serial="A0001">JohnParker</foo:Emp>
B0001  (null)     Smith     <foo:Emp xmlns:foo="http://www.foo.com"
                                serial="B0001">Smith</foo:Emp>
B0002  (null)     (null)    (null)
(null) (null)     (null)    (null)
```

- 以下の例は、前の例に似ています。ただし、参照される列に NULL 値がある場合、空エレメントが戻ります。

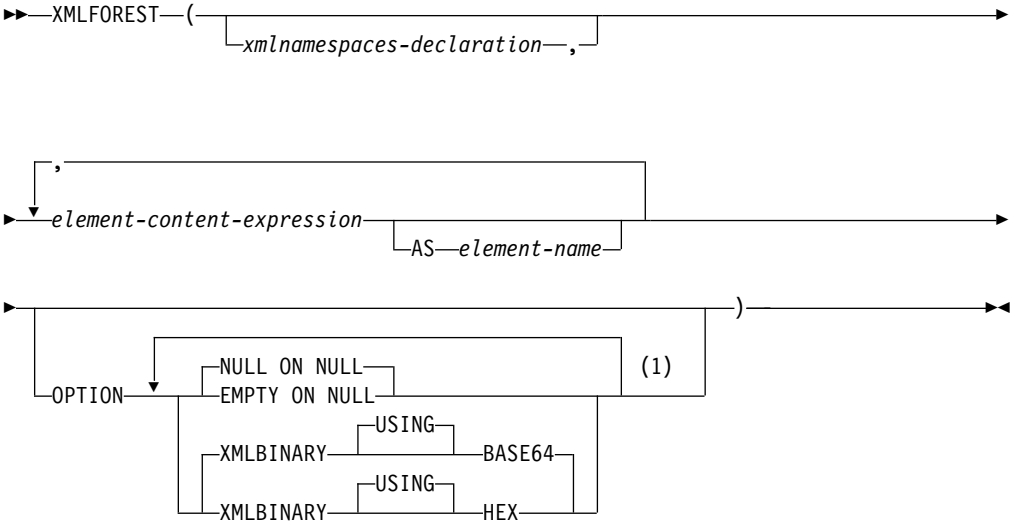
```
SELECT E.EMPNO, E.FIRSTNME, E.LASTNAME,
       XMLELEMENT(NAME "foo:Emp",
                  XMLNAMESPACES('http://www.foo.com' AS "foo"),
                  XMLATTRIBUTES(E.EMPNO AS "serial"),
                  E.FIRSTNME, E.LASTNAME
                  OPTION EMPTY ON NULL) AS "Result"
FROM SESSION.CANDIDATES E
```

この照会は下記の結果を生成します。

```
EMPNO  FIRSTNME   LASTNAME  Result
------ ---------  --------  ------------------
A0001  John       Parker    <foo:Emp xmlns:foo="http://www.foo.com"
                                serial="A0001">JohnParker</foo:Emp>
B0001  (null)     Smith     <foo:Emp xmlns:foo="http://www.foo.com"
                                serial="B0001">Smith</foo:Emp>
B0002  (null)     (null)    <foo:Emp xmlns:foo="http://www.foo.com"
                                serial="B0002"></foo:Emp>
(null) (null)     (null)    <foo:Emp xmlns:foo="http://www.foo.com"></foo:Emp>
```

XMLFOREST

XMLFOREST 関数は、XML エレメントのシーケンスである XML 値を戻します。



注:

1 同じ文節を複数回指定することはできません。

xmlnamespace-declaration
XMLNAMESPACES 宣言の結果である XML ネームス・ペース宣言を指定します。宣言されるネーム・スペースは、XMLFOREST 関数の有効範囲内にあるものです。名前空間は、別の副選択の中で現れるかどうかにかかわらず、XMLFOREST 関数内でネストされているすべての XML 関数に対して適用されます。 XML ネーム・スペースの宣言について詳しくは、 715 ページの『XMLNAMESPACES』を参照してください。

xmlnamespace-declaration を指定しない場合、ネーム・スペース宣言は構成された XML エレメントに関連付けられません。

element-content-expression
生成される XML エレメントの内容に使われる値を戻す式を指定します。式のデータ・タイプは、ROWID または DATALINK であってはなりません。式が単純な列参照でない場合は、*element-name* を指定する必要があります。

AS *element-name*
XML エレメント名として使用される ID を指定します。

XML エレメント名は XML 修飾名または QName であることが必要です。有効な名前についての詳細は、W3C XML ネーム・スペースの指定の項目を参照してください。名前を修飾する場合は、有効範囲内にネーム・スペース接頭部が宣言されている必要があります。

element-name を指定しない場合、*element-content-expression* は列名にする必要があります。エレメント名は、列名から QName への完全エスケープ・マッピングを使用して、列名から作成されます。

OPTION

NULL 値、バイナリー・データ、およびビット・データの結果に関するオプションを指定します。このオプションは、*element-content-expression* に現れる XMLELEMENT 関数または XMLFOREST 関数によって継承されません。

NULL ON NULL または **EMPTY ON NULL**

すべての *element-content-expression* 値が NULL 値の場合に、NULL 値と空エレメントのどちらを戻すかを指定します。このオプションが影響を与えるのは、*element-content-expression* 引数の NULL の処理に関してのみです。デフォルトは NULL ON NULL です。

NULL ON NULL

各 *element-content-expression* の値が NULL の場合、NULL 値が戻されます。

EMPTY ON NULL

各 *element-content-expression* の値が NULL の場合、空のエレメントが戻されます。

XMLBINARY USING BASE64 または **XMLBINARY USING HEX**

バイナリー入力データ、FOR BIT DATA 属性を指定した文字ストリング、ROWID、またはこれらのタイプのいずれかに基づく特殊タイプに対して想定されるエンコードを指定します。エンコードは要素コンテントまたは属性値に適用されます。デフォルトは XMLBINARY USING BASE64 です。

XMLBINARY USING BASE64

想定されるエンコードは、XML スキーマ・タイプ xs:base64Binary エンコードに対して定義された base64 文字であることを指定します。Base64 エンコードは US-ASCII (10 個の数字、26 個の小文字、26 個の大文字、「+」および「/」) の 65 文字サブセットを使用して、6 ビットのすべてのバイナリー・データまたはビット・データをこのサブセットの印刷可能文字で表します。これらの文字は、例外なく表示できるように選択されます。この方式を使用すると、エンコードされたデータのサイズは元のバイナリー・データまたはビット・データのサイズよりも 33 パーセント大きくなります。

XMLBINARY USING HEX

想定されるエンコードは、XML スキーマ・タイプ xs:hexBinary エンコードに対して定義された 16 進文字であることを指定します。16 進エンコードは、それぞれのバイト (8 ビット) を 2 桁の16 進文字で表します。この方式を使用すると、エンコードされたデータのサイズは元のバイナリー・データまたはビット・データのサイズの 2 倍になります。

この関数の結果は XML 値です。いずれかの *element-content-expression* の結果が NULL の場合、結果も NULL になる可能性があります。すべての *element-content-expression* の結果が NULL の場合、NULL ON NULL オプションが有効になっていると、結果は NULL 値になります。

XMLFOREST 関数は、XMLCONCAT 関数と XMLELEMENT 関数を使用して表すことができます。例えば、次の 2 つの式は同じ意味合いです。

```
XMLFOREST(xmlnamespaces-declaration, arg1 AS name1, arg2 AS name2, ...)

XMLCONCAT(XMLELEMENT(NAME name1, xmlnamespaces-declaration, arg1),
          XMLELEMENT(NAME name2, xmlnamespaces-declaration, arg2),
 ... )
```

デフォルトのネーム・スペースを定義する別のエレメントの内容としてコピーされるエレメントを構成する場合、デフォルトのネーム・スペースはコピーされたエレメント内で明示的に宣言解除する必要があります。これは、新規の親エレメントからデフォルトのネーム・スペースを継承した結果として生じる可能性のあるエラーを避けるためです。事前定義ネーム・スペース接頭部 (「xs」、「xsi」、「xml」、および「sqlxml」) をその使用時に明示的に宣言する必要もあります。

例

注: XMLFOREST は、出力の中にブランク・スペースまたは改行文字を挿入しません。読みやすくするために、すべての出力例はフォーマット設定されています。

- デフォルトのネーム・スペースを持つエレメントのフォレストを構成します。

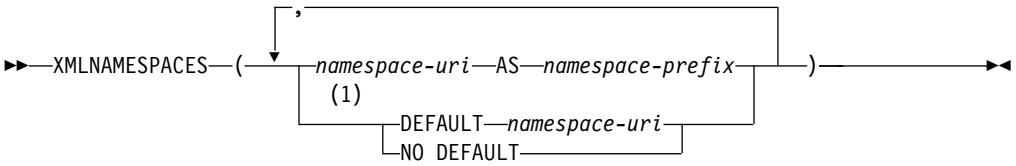
```
SELECT EMPNO,
       XMLFOREST(XMLNAMESPACES(DEFAULT 'http://hr.org',
                               'http://fed.gov' AS "d"),
                 LASTNAME, JOB AS "d:job") AS "Result"
FROM EMPLOYEE WHERE EDLEVEL = 12
```

この照会は下記の結果を生成します。

```
EMPNO  Result
000290 <LASTNAME xmlns:"http://hr.org" xmlns:d="http://fed.gov">PARKER
       </LASTNAME>
       <d:job xmlns:"http://hr.org" xmlns:d="http://fed.gov">OPERATOR</d:job>
000310 <LASTNAME xmlns:"http://hr.org" xmlns:d="http://fed.gov">SETRIGHT
       </LASTNAME>
       <d:job xmlns:"http://hr.org" xmlns:d="http://fed.gov">OPERATOR</d:job>
200310 <LASTNAME xmlns:"http://hr.org" xmlns:d="http://fed.gov">SPRINGER
       </LASTNAME>
       <d:job xmlns:"http://hr.org" xmlns:d="http://fed.gov">OPERATOR</d:job>
```

XMLNAMESPACES

XMLNAMESPACES 宣言は、引数からネーム・スペース宣言を構成します。この宣言は XMLELEMENT 関数と XMLFOREST 関数の引数としてのみ使用できます。結果は、NULL 以外の入力値のそれぞれの有効範囲内ネーム・スペースが入っている、1 つ以上の XML ネーム・スペース宣言です。



注:

1 DEFAULT 文節または NO DEFAULT 文節は 1 度だけ指定できます。

namespace-uri
ネーム・スペース名または URI を含む、SQL 文字ストリング定数を指定します。*namespace-prefix* と一緒に使用する場合、文字ストリング定数は空ストリングであってはなりません。

AS *namespace-prefix*
ネーム・スペース接頭部を指定します。接頭部は XML NCName 形式の SQL ID でなければなりません。有効な名前についての詳細は、W3C XML ネーム・スペースの指定の項目を参照してください。接頭部には「xml」または「xmlns」は使用できません。接頭部はネーム・スペース宣言のリスト内で固有でなければなりません。

ネーム・スペース接頭部「xml」、「xs」、「xsd」、「xsi」、および「sqlxml」は、SQL/XML 内で事前定義されています。これらのバインディングは次のとおりです。

- xmlns:xml = "http://www.w3.org/XML/1998/namespace"
- xmlns:xs = "http://www.w3.org/2001/XMLSchema"
- xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
- xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
- xmlns:sqlxml = "http:/standards.iso.org/iso/9075/2003/sqlxml"

DEFAULT *namespace-uri* または **NO DEFAULT**
このネーム・スペース宣言の有効範囲内でデフォルトのネーム・スペースを使用するかどうかを指定します。

このネーム・スペース宣言の有効範囲は、指定された XML エレメントと、指定された XML エレメントに含まれるすべての XML 式です。

DEFAULT *namespace-uri*
このネーム・スペース宣言の有効範囲内で使用するデフォルトのネーム・スペースを指定します。*namespace-uri* は、有効範囲内の非修飾名に適用されます。ただし、別の DEFAULT 宣言または NO DEFAULT 宣言によって、ネストされた有効範囲の中でオーバーライドされる場合は、その限りではありません。

namespace-uri は、ネーム・スペース名または URI を含む、SQL 文字ストリング定数を指定します。文字ストリング定数は、DEFAULT 文節のコンテキスト内では空ストリングであってもかまいません。

NO DEFAULT
このネーム・スペース宣言の有効範囲内ではデフォルトのネーム・スペースを使用しないことを指定します。NO DEFAULT 文節がネストされた有効範囲内の DEFAULT 宣言によってオーバーライドされない限り、この有効範囲にはデフォルトのネーム・スペースはありません。

この関数の結果の XML 値は、指定された各ネーム・スペースの XML ネーム・スペース宣言が入っている、XML シーケンスです。結果が NULL になることはありません。

例

注: XML 処理は、出力の中にブランク・スペースまたは改行文字を挿入しません。読みやすくするために、すべての出力例はフォーマット設定されています。

- それぞれの従業員ごとに「employee」エレメントを生成します。employee エレメントは、XML ネーム・スペース「urn:bo」に関連付けられ、このネーム・スペースは接頭部「bo」にバインドされます。このエレメントには、名前と hiredate サブエレメントの属性が含まれています。

```
SELECT E.EMPNO,
       XMLSERIALIZE(XMLELEMENT(NAME "bo:employee",
                               XMLNAMESPACES('urn:bo' AS "bo"),
                               XMLATTRIBUTES(E.LASTNAME, E.FIRSTNME),
                               XMLELEMENT(NAME "bo:hiredate", E.HIREDATE))
                    AS CLOB(200))
  FROM EMPLOYEE E WHERE E.EDLEVEL = 12
```

この照会は下記の結果を生成します。

```
00029 <bo:employee xmlns:bo="urn:bo" LASTNAME="PARKER" FIRSTNME="JOHN">
      <bo:hiredate>1988-05-30</bo:hiredate>
      </bo:employee>
00031 <bo:employee xmlns:bo="urn:bo" LASTNAME="SETRIGHT" FIRSTNME="MAUDE">
      <bo:hiredate>1964-09-12</bo:hiredate>
      </bo:employee>
```

- それぞれの従業員ごとに、XMLFOREST を使用して 2 つのエレメントを生成します。最初の「lastname」エレメントはデフォルト・ネーム・スペース「http://hr.org」に関連付けられ、2 番目の「job」エレメントは、接頭部「d」にバインドされた XML ネーム・スペース「http://fed.gov」に関連付けられます。

```
SELECT EMPNO,
       XMLSERIALIZE(XMLFOREST(XMLNAMESPACES(DEFAULT 'http://hr.org',
                                            'http://fed.gov' AS "d"),
                              LASTNAME, JOB AS "d:job")
                    AS CLOB(200))
  FROM EMPLOYEE WHERE EDLEVEL = 12
```

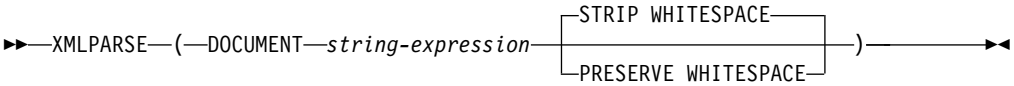
この照会は下記の結果を生成します。

```
00029 <LASTNAME xmlns="http://hr.org" xmlns:d="http://fed.gov">PARKER
      </LASTNAME>
      <d:job xmlns="http://hr.org" xmlns:d="http://fed.gov">
      OPERATOR</d:job>
```

```
00031 <LASTNAME xmlns="http://hr.org" xmlns:d="http://fed.gov">
      SETRIGHT</LASTNAME>
      <d:job xmlns="http://hr.org" xmlns:d="http://fed.gov">
      OPERATOR</d:job>
```

XMLPARSE

XMLPARSE 関数は、引数を XML 文書として解析し、XML 値を戻します。



DOCUMENT
解析する文字ストリング式は、XML 1.0 に準拠した整形式 XML 文書として評価する必要があることを指定します。

string-expression
組み込み文字ストリング、Unicode グラフィック・ストリング、またはバイナリー・ストリングの値を戻す式。パラメーター・マーカーを使用する場合、サポートされているいずれかのデータ・タイプに明示的にキャストする必要があります。

STRIP WHITESPACE または **PRESERVE WHITESPACE**
入力引数内の空白を保持するかどうかを指定します。どちらも指定しない場合、デフォルトは STRIP WHITESPACE です。

STRIP WHITESPACE
空白文字を除去することを指定します (最も近い収容エレメントが属性 *xml:space='preserve'* を持つ場合以外)。このオプションによって、CDATA セクション内の空白文字も影響を受けます。

PRESERVE WHITESPACE
すべての空白を保持することを指定します (最も近い収容エレメントが属性 *xml:space='default'* を持つ場合でも同様)。

この関数の結果は XML です。ストリング式 がヌルになる可能性がある場合は、結果もヌルになる可能性があります。ストリング式 がヌルの場合は、結果は NULL 値になります。結果の CCSID は、*string-expression* によって決まります。*string-expression* の CCSID が 65535 の場合、SQL_XML_DATA_CCSID QAQQINI オプションの値が使用されます。

入力ストリングには、XML 文書内の文字のエンコードを識別する XML 宣言を含めることができます。XML 宣言内のエンコードは、*string-expression* のエンコードと一致しなければなりません。

例

例 *1:* XML 文書を EMP 表に挿入し、オリジナル XML 文書の空白を保存します。

```
INSERT INTO EMP (ID, XVALUE) VALUES(1001,
   XMLPARSE(DOCUMENT '<a xml:space=''preserve''> <b> <c>c</c>b </b> </a>'
            PRESERVE WHITESPACE))
```

XMLPARSE は、INSERT ステートメントの値を以下の値と同等なものとして処理します。

```
<a xml:space='preserve'> <b> <c>c</c>b </b> </a>
```

例 *2:* XML 文書を EMP 表に挿入し、オリジナル XML 文書の空白を除去します。

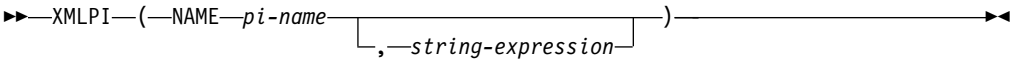
```
INSERT INTO EMP (ID, XVALUE) VALUES(1001,
   XMLPARSE(DOCUMENT
            '<a xml:space=''preserve''> <b xml:space=''default''> <c>c</c>b </b> </a>'
          STRIP WHITESPACE))
```

XMLPARSE は、INSERT ステートメントの値を以下の値と同等なものとして処理します。

```
<a xml:space='preserve'>
<b xml:space='default'><c>c</c>b </b>
</a>
```

XMLPI

XMLPI 関数は、単一の処理命令を含む XML 値を戻します。



NAME *pi-name*
処理命令の名前を指定します。名前は XML NCName 形式の SQL ID でなければなりません。有効な名前についての詳細は、W3C XML ネーム・スペースの指定の項目を参照してください。名前には「xml」は使用できません (どのような大/小文字の組み合わせでも)。

string-expression
組み込み文字ストリングまたは組み込みグラフィック・ストリングの値を戻す式。CHAR または VARCHAR ビット・データであってはなりません。結果ストリングは、以下の規則に定められた XML 処理命令の内容に準拠していなければなりません。

- ストリングにはサブストリング「?>」を含めてはならない。このサブストリングは処理命令を終了させます。
- ストリングの各文字は、サロゲート・ブロック X'FFFE' および X'FFFF' を除く、任意の Unicode 文字とすることができる。[78]

string-expression が上記の規則に準拠していない場合は、エラーが戻されます。

結果ストリングは、処理命令の内容になります。

この関数の結果は XML です。ストリング式 がヌルになる可能性がある場合は、結果もヌルになる可能性があります。ストリング式 がヌルの場合は、結果は NULL 値になります。*string-expression* が空ストリングの場合、または指定されない場合、結果は空の処理命令となります。

例

- XML 処理命令を生成します。

```
SELECT XMLPI(
     NAME "Instruction", 'Push the red button')
  FROM SYSIBM.SYSDUMMY1
```

この照会は下記の結果を生成します。

```
<?Instruction Push the red button?>
```

- 空の XML 処理命令を生成します。

```
SELECT XMLPI(NAME "Warning")
  FROM SYSIBM.SYSDUMMY1
```

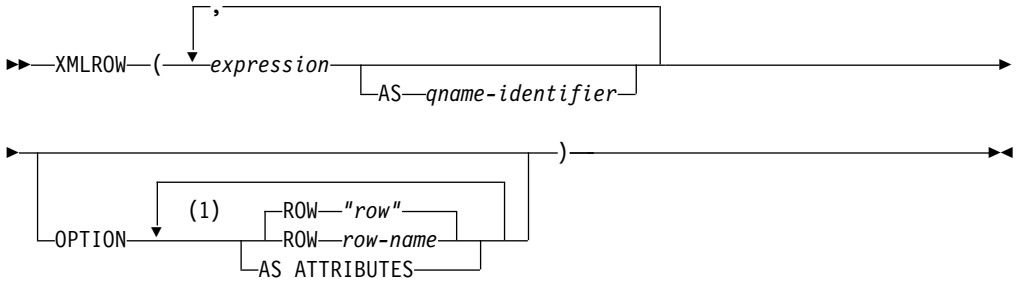
この照会は下記の結果を生成します。

```
<?Warning ?>
```

78. 有効な Unicode 文字の Unicode コード・ポイントは、#x9、 #xA、#xD、#x20-#xD7FF、#xE000-#xFFFD、#x10000-#x10FFFF です。

XMLROW

XMLROW 関数は、整形式 XML 文書である XML 値を戻します。



注:

1 同じ文節を複数回指定することはできません。

expression

各 XML エレメントの内容を式で指定します。 *expression* のデータ・タイプは、ROWID または DATALINK、あるいは ROWID または DATALINK に基づく特殊タイプ以外であってはなりません。AS ATTRIBUTES を指定する場合、*expression* のデータ・タイプは、XML、または XML に基づく特殊タイプ以外でなければなりません。式として、SQL 式を指定することもできます。式が単純な列参照でない場合は、エレメント名を指定する必要があります。

AS *qname-identifier*

XML の要素名または属性名を SQL ID として指定します。 *qname-identifier* は、XML 修飾名 (つまり QName) の形式でなければなりません。有効な名前についての詳細は、W3C XML ネーム・スペースの指定の項目を参照してください。名前を修飾する場合は、有効範囲内にネーム・スペース接頭部が宣言されている必要があります。*qname-identifier* を指定しない場合は、*expression* を列名にする必要があります。エレメント名または属性名は、列名から QName への完全エスケープ・マッピングを使用して、列名から作成されます。

OPTION

XML 値を構成するための追加オプションを指定します。OPTION 文節を指定しない場合は、デフォルトの動作が適用されます。

ROW *row-name*

各行に対応付ける要素の名前を指定します。このオプションを指定しない場合のデフォルトの要素名は、「row」です。

AS ATTRIBUTES

それぞれの式を属性値に対応付けることを指定します。属性名としての役割を果たすのは、列名または *qname-identifier* です。AS ATTRIBUTES は、式の結果のデータ・タイプが XML の場合には指定できません。

結果は、NULL 以外の入力 XML 値を連結したものを含む XML シーケンスです。

この関数の結果は XML 値です。入力の中の NULL 値は無視されます。いずれかの *expression* の結果が NULL になる可能性がある場合は、結果も NULL になる可能性があります。すべての *expression* の結果が NULL の場合は、結果は NULL 値になります。

注

デフォルトで、結果セットの各行は次のように XML 値にマップされます。

- 各行は「row」という名前を持つ XML エレメントに変換され、各 *expression* はネストされたエレメントに変換されます。その際、エレメント名として列名または *qname-identifier* が使用されます。
- NULL 処理の動作は NULL ON NULL です。サブエレメントが存在しないという状態に、式の NULL 値が対応付けられます。すべての式の値が NULL の場合、この関数によって NULL 値が戻されます。
- バイナリー・データ・タイプと FOR BIT DATA データ・タイプのバイナリー・コード化スキームは、base64Binary エンコード方式です。

例

表 T1 に列 C1 と C2 があるとします。

```
 C1    C2
----  ----
  1     2
  -     2
  1     -
  -     -
```

- 以下の例は、XMLROW 照会と、デフォルト動作による出力断片を示しています。表を表すために一連の行エレメントがその中で使用されています。

```
  SELECT XMLROW(C1, C2) FROM T1

<row><C1>1</C1><C2>2</C2></row>
<row><C2>2</C2></row>
<row><C1>1</C1></row>
-
```

- 以下の例は、XMLROW 照会と、属性を中心としたマッピングによる出力断片を示しています。データはネストされたエレメントとして表示されず、エレメント属性にマップされています。

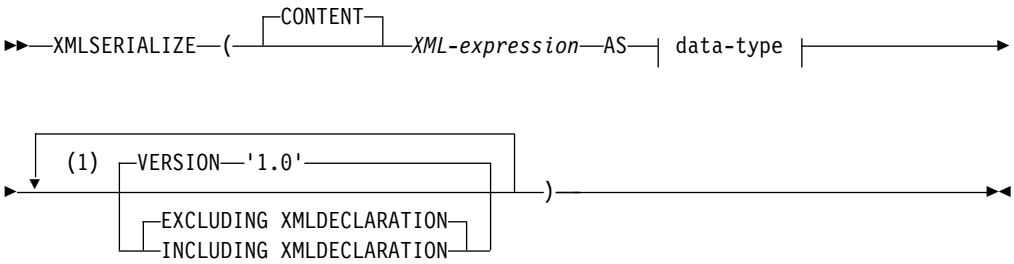
```
  SELECT XMLROW(C1, C2 OPTION AS ATTRIBUTES) FROM T1

<row C1="1" C2="2"/>
<row C2="2"/>
<row C1="1"/>
-
```

- 以下の例は、XMLROW 照会と、デフォルト <row> エレメントが <entry> によって置き換えられた出力断片を示しています。列 C1 と C2 が <column1> および <column2> エレメントとしても戻され、C1 と C2 の合計が <total> エレメント内に戻されます。

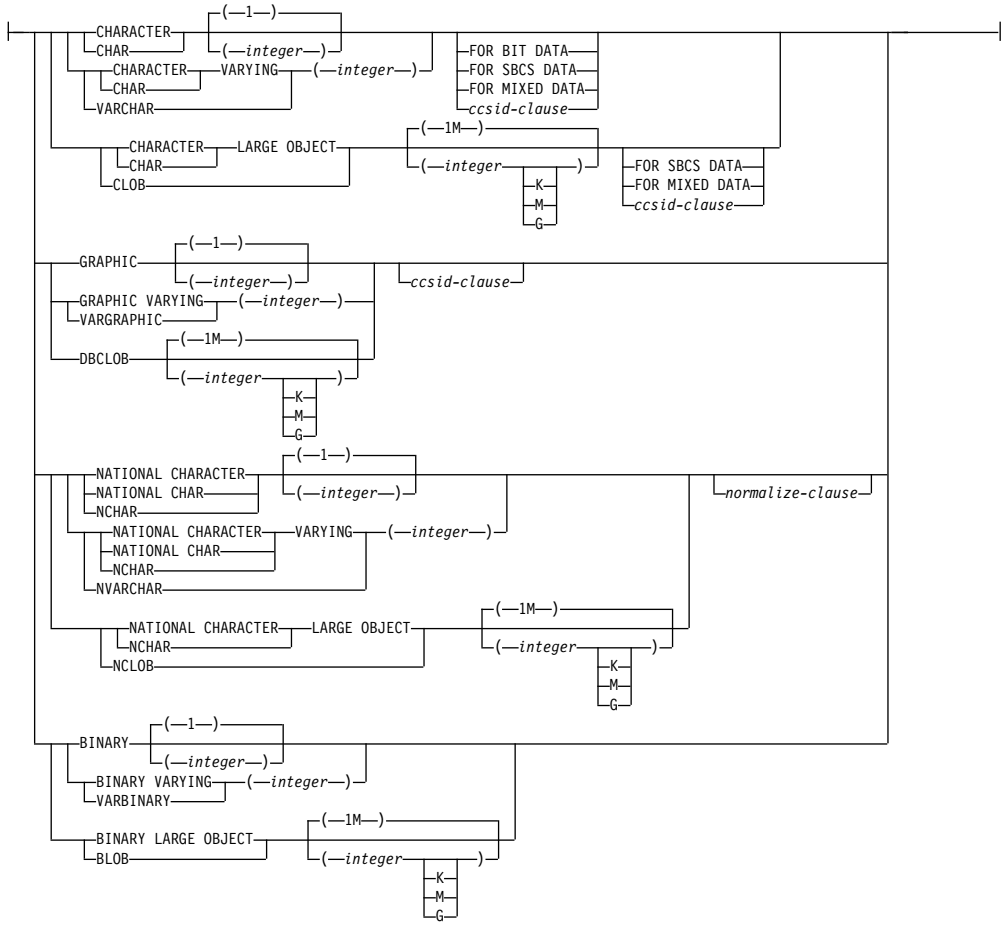
```
  SELECT XMLROW(C1 AS "column1", C2 AS "column2",
                C1+C2 AS "total" OPTION ROW "entry") FROM T1

<entry><column1>1</column1><column2>2</column2><total>3</total></entry>
<entry><column2>2</column2></entry>
<entry><column1>1</column1></entry>
-
```

XMLSERIALIZE

XMLSERIALIZE 関数は、*XML-expression* 引数から生成される、指定のデータ・タイプの直列化された XML 値を戻します。



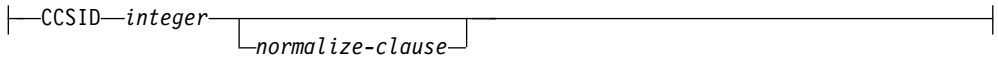
data-type:



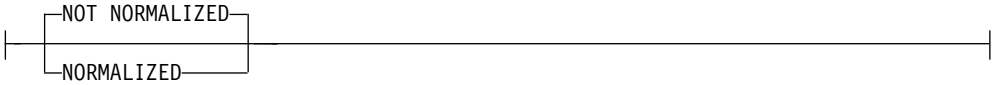
注:

1 同じ文節を複数回指定することはできません。

ccsid-clause:



normalize-clause:



CONTENT
任意の XML 値を指定できることを示し、直列化の結果はこの入力値に基づきます。

XML-expression
組み込み XML ストリングの値を戻す式。これは、直列化処理への入力です。

AS *data-type*
結果タイプを指定します。指定された結果のデータ・タイプの暗黙的または明示的な長さ属性は、直列化された出力を収容するのに十分な大きさでなければなりません。

CCSID が指定され、*data-type* が GRAPHIC、VARGRAPHIC、または DBCLOB の場合、CCSID は Unicode CCSID であることが必要です。

CCSID 属性が指定されないと、CCSID は 218 ページの『CAST の指定』に記されているように決定されます。

VERSION '1.0'
直列化された値の XML バージョンを指定します。サポートされる唯一のバージョンは「1.0」で、これをストリング定数として指定する必要があります。

EXCLUDING XMLDECLARATION または **INCLUDING XMLDECLARATION**
XML 宣言を結果に含めるかどうかを指定します。デフォルトは EXCLUDING XMLDECLARATION です。

EXCLUDING XMLDECLARATION
XML 宣言を結果に含めないことを指定します。

INCLUDING XMLDECLARATION
XML 宣言を結果に含めることを指定します。 XML 宣言は、ストリング '<?xml version="1.0" encoding="*encoding-name*"?> で、 *encoding-name* は結果のデータ・タイプの CCSID と一致します。

XML 式 がヌルになる可能性がある場合は、結果もヌルになる可能性があります。*XML* 式 がヌルの場合は、結果は NULL 値になります。

例

例 *1:* XMLELEMENT 関数によって戻される XML 値 (エレメント名が「Emp」で、従業員名をエレメントの内容として持つ単純 XML エレメント) を直列化して UTF-8 の CLOB にします。

```
SELECT e.empno, XMLSERIALIZE(XMLELEMENT(NAME "Emp",
                                        e.firstnme || ' ' ||e.lastname)
                             AS CLOB(100) CCSID 1208) AS "result"
    FROM employee e WHERE e.lastname = 'SMITH'
```

結果は以下のようになります。

```
EMPNO   result
-----   --------------------
000250  <Emp>DANIEL SMITH</Emp>
000300  <Emp>PHILIP SMITH</Emp>
```

例 *2:* XMLELEMENT 関数によって戻される XML 値を直列化して BLOB タイプのストリングにします。

```
SELECT XMLSERIALIZE(XMLELEMENT(NAME "Emp",
                                    e.firstnme || ' ' ||e.lastname)
                    AS BLOB(1K)
                    VERSION '1.0') AS "result"
    FROM employee e WHERE e.empno = '000300'
```

結果は以下のようになります。

```
result
--------------------
<Emp>PHILIP SMITH</Emp>
```

例 *3:* XMLELEMENT 関数によって戻される XML 値を直列化して CLOB タイプのストリングにします。XMLDECLARATION を以下のように組み込みます。

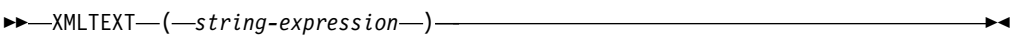
```
SELECT e.empno, e.firstnme, e.lastname,
     XMLSERIALIZE(XMLELEMENT(NAME "xmp:Emp",
                             XMLNAMESPACES('http://www.xmp.com' as "xmp"),
                             XMLATTRIBUTES(e.empno as "serial"),
                             e.firstnme, e.lastname
                             OPTION NULL ON NULL)
                  AS CLOB(1000) CCSID 1208
                  INCLUDING XMLDECLARATION) AS "Result"
    FROM employee e WHERE e.empno = '000300'
```

結果は以下のようになります。

```
EMPNO    FIRSTNME    LASTNAME    結果
------  ---------  ---------  ----------------------
000300  PHILIP     SMITH      <?xml version="1.0" encoding="UTF-8" ?>
                              <xmp:Emp xmlns:xmp="http://www.xmp.com"
                                  serial="000300">PHILIPSMITH</xmp:Emp>
```

XMLTEXT

XMLTEXT 関数は、*string-expression* の値を含む XML 値を戻します。



string-expression
組み込み文字ストリングまたは組み込みグラフィック・ストリングの値を戻す式。 CHAR または VARCHAR ビット・データであってはなりません。

この関数の結果は XML 値です。ストリング式 がヌルになる可能性がある場合は、結果もヌルになる可能性があります。ストリング式 がヌルの場合は、結果は NULL 値になります。*string-expression* の結果が空ストリングの場合、結果値は空テキストです。

例

- 単純な XMLTEXT 照会を作成します。

```
VALUES (XMLTEXT (
          'The stock symbol for Johnson&Johnson is JNJ.'))
```

この照会は、以下の直列化された結果を生成します。

```
The stock symbol for Johnson&amp;Johnson is JNJ.
```

「&」記号は、テキストが直列化されるときには「&」にマップされることに注意してください。

- XMLTEXT を XMLAGG と共に使用して、混合の内容を構成します。表 T の内容が以下のようであるとします。
-

```
SEQNO  PLAINTEXT                                                  EMPHTEXT
-----  ---------------------------------------------------------  ----------
1      This query shows how to construct                          mixed content
2       using XMLAGG and XMLTEXT. Without                         XMLTEXT
3      , XMLAGG will not have text nodes to group with other nodes,  mixed content
       therefore, cannot generate
```

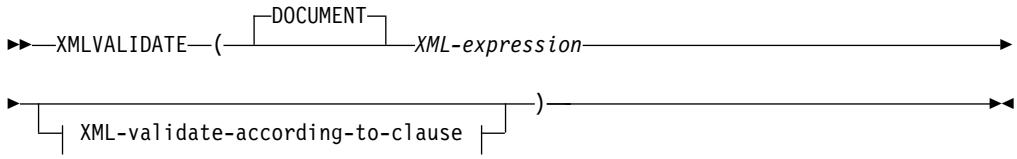
```
SELECT XMLELEMENT(NAME "para",
              XMLAGG(XMLCONCAT(
                                XMLTEXT(PLAINTEXT),
                                XMLELEMENT(
                                     NAME "emphasis", EMPHTEXT))
                         ORDER BY SEQNO), '.') AS "result"
  FROM T
```

この照会は下記の結果を生成します。

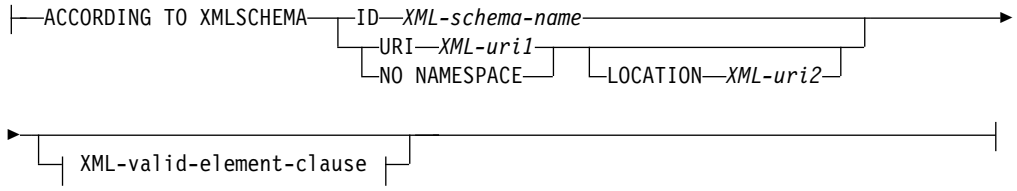
```
result
------------------------------------------------------------------------------
<para>This query shows how to construct <emphasis>mixed content</emphasis>
using XMLAGG and XMLTEXT. Without <emphasis>XMLTEXT</emphasis>, XMLAGG
will not have text nodes to group with other nodes, therefore, cannot generate
<emphasis>mixed content</emphasis>.</para>
```

XMLVALIDATE

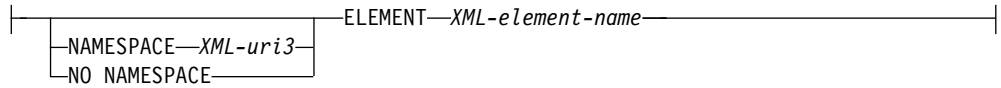
XMLVALIDATE 関数は、デフォルト値およびタイプ・アノテーションを含む、XML スキーマ妥当性検査から取得した情報が加えられた入力 XML 値のコピーを戻します。



XML-validate-according-to-clause:



XML-valid-element-clause:



DOCUMENT
XML-expression の結果の XML 値が、XML バージョン 1.0 に準拠する整形式 XML 文書でなければならないことを指定します。

XML-expression
データ・タイプ XML の値を戻す式。*XML-expression* が、XML ホスト変数、または暗黙的あるいは明示的なタイプ付きパラメーター・マーカーである場合、この関数は、無視できる空白文字を除去する妥当性検査のための構文解析を実行し、CURRENT IMPLICIT XMLPARSE OPTION 設定は考慮されません。

XML-validate-according-to-clause
入力 XML 値の妥当性検査時に使用する情報を指定します。

ACCORDING TO XMLSCHEMA
妥当性検査用の XML スキーマ情報を明示的に指定することを示します。この節が組み込まれない場合、XML スキーマ情報は *XML-expression* 値の中で提供される必要があります。

ID *XML-schema-name*
妥当性検査に使用される XML スキーマの SQL ID を指定します。この名前 (暗黙的または明示的 SQL スキーマ修飾子を含む) は、現行のサーバーの XML スキーマ・リポジトリー内で既存の XML スキーマを固有に識別しなければなりません。暗黙的または明示的に指定した SQL スキーマにこの名前の XML スキーマが存在しない場合は、エラーが戻されます。

URI *XML-uri1*
妥当性検査に使用される XML スキーマのターゲット・ネーム・スペース URI を指定します。*XML-uri1* の値は、URI を空でない文字ストリング定数として指定します。URI は、登録済み XML スキーマのターゲット・ネーム・スペースでなければならず、LOCATION 節を指定しない場合は、登録済み XML スキーマを固有に識別する必要があります。

NO NAMESPACE
妥当性検査用の XML スキーマはターゲット・ネーム・スペースを持たないことを指定します。ターゲット・ネーム・スペース URI は、明示的なターゲット・ネーム・スペース URI として指定できない空の文字ストリングと同等です。

LOCATION *XML-uri2*
妥当性検査に使用される XML スキーマの XML スキーマ・ロケーション URI を指定します。*XML-uri2* の値は、URI を空でない文字ストリング定数として指定します。XML スキーマ・ロケーション URI は、ターゲット・ネーム・スペース URI と結合されて登録済み XML スキーマを識別する必要があり、登録済みのそのような XML スキーマは 1 つだけでなければなりません。

XML-valid-element-clause
XML-expression 内の XML 値が、指定されたエレメント名を、XML 文書のルート・エレメントとして持つ必要があることを指定します。

NAMESPACE *XML-uri3* **または NO NAMESPACE**
妥当性検査されるエレメントのターゲット・ネーム・スペースを指定します。どちらの節も指定されない場合、指定されたエレメントは、妥当性検査に使用される登録済み XML スキーマのターゲット・ネーム・スペースと同じネーム・スペース内にあると想定されます。

NAMESPACE *XML-uri3*
妥当性検査されるエレメントのネーム・スペース URI を指定します。*XML-uri3* の値は、URI を空でない文字ストリング定数として指定します。これは、妥当性検査に使用される登録済み XML スキーマが複数のネーム・スペースを持つ場合に使用することができます。

NO NAMESPACE
妥当性検査用のエレメントはターゲット・ネーム・スペースを持たないことを指定します。ターゲット・ネーム・スペース URI は、明示的なターゲット・ネーム・スペース URI として指定できない空の文字ストリングと同等です。

ELEMENT *xml-element-name*
妥当性検査に使用される XML スキーマ内のグローバル・エレメントの名前を指定します。指定されるエレメントは、暗黙的または明示的ネーム・スペースを持ち、*XML-expression* の値のルート・エレメントと一致しなければなりません。

この関数の結果は XML です。*XML-expression* の値が NULL になる可能性がある場合、結果も NULL になる可能性があります。*XML-expression* の値が NULL の場合、結果も NULL 値です。結果の CCSID は、*XML-expression* によって決まります。

XML 妥当性検査プロセスは、直列化された XML 値に対して実行されます。XMLVALIDATE は XML タイプの引数で呼び出されるので、この値は、妥当性検査プロセスに先だって自動的に直列化されます。ただし、以下の 2 つの例外があります。

- XMLVALIDATE への引数が XML ホスト変数であるか、または暗黙的あるいは明示的なタイプ付きパラメーター・マーカーである場合、妥当性検査のための構文解析操作が入力値に対して実行されます (暗黙の妥当性検査以外の構文解析は実行されず、CURRENT IMPLICIT XMLPARSE OPTION 設定は考慮されません)。
- XMLVALIDATE への引数が、オプション PRESERVE WHITESPACE を使用した XMLPARSE 呼び出しである場合、文書の XML 構文解析および XML 妥当性検査を組み合わせて、単一の妥当性検査のための構文解析操作にすることができます。

ルート・エレメントにネーム・スペースがない文書を妥当性検査するには、`xsi:noNamespaceSchemaLocation` 属性がルート・エレメント上に存在していなければなりません。

注

XML スキーマの決定**:** XML スキーマは、XMLVALIDATE 呼び出しの一部として明示的に指定したり、入力 XML 値内の XML スキーマ情報から判別したりすることができます。XML スキーマ情報が呼び出し中に指定されない場合、入力 XML 値内のターゲット・ネーム・スペースおよびスキーマ・ロケーションが、妥当性検査のための登録済みスキーマを特定するために使用されます。明示的な XML スキーマが指定されない場合、入力 XML 値には、XML スキーマ情報のヒントが含まれなければなりません。明示的または暗黙的な XML スキーマ情報は、登録済み XML スキーマを特定する必要があり、登録済みのそのような XML スキーマは 1 つだけでなければなりません。

XML スキーマ文書を指定しない場合、データベース・サーバーは、入力文書内で、XML スキーマのネーム・スペースと場所のヒントを指定する *xsi:schemaLocation* 属性を探します。XML スキーマのターゲット・ネーム・スペースがない場合、XML スキーマの場所についてのヒントを指定するために *xsi:noNamespaceSchemaLocation* 属性が使用されます。

xsi:schemaLocation 属性または *xsi:noNamespaceSchemaLocation* 属性は W3C XML スキーマ指定によって定義され、XML スキーマ・ヒントと呼ばれます。*xsi:schemaLocation* 属性には、XML スキーマ文書を見つけるのに役立つ値のペアが 1 つ以上含まれています。各ペアの最初の値はネーム・スペースで、2 番目の値はそのネーム・スペースの XML スキーマがある場所を示すヒントです。Db2 for i は、 1 次 XML スキーマ文書のターゲット・ネーム・スペースおよび

XSR_REGISTER プロシージャー呼び出しに指定された *schemalocation* パラメーターを使用して、ネーム・スペースおよび場所のヒントと XML スキーマとのマッチングを試みます。

XMLVALIDATE 関数で XML スキーマ文書を指定すると、*xsi:schemaLocation* 属性または *xsi:noNamespaceSchemaLocation* 属性がオーバーライドされます。

xsi:schemaLocation 属性および *xsi:noNamespaceSchemaLocation* 属性が XML 文書によって定義されていない場合、Db2 for i は、1 次 XSD のターゲット・ネーム・スペースが XML 文書中のネーム・スペースと一致する XML スキーマを見つけようとします。

XMLVALIDATE 関数で XML スキーマを明示的に指定すると、XML 値で XML スキーマ情報のヒントを見つけるのに必要な構文解析をしないで済みます。

XML スキーマの権限: 妥当性検査に使用される XML スキーマは、使用前に XML スキーマ・リポジトリーに登録されていなければなりません。このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 妥当性検査時に使用される XML スキーマに対する USAGE 特権
- データベース管理者権限

例

- XML インスタンス文書内で XML スキーマのヒントによって識別される XML スキーマを使用して妥当性検査します。

```
INSERT INTO T1(XMLCOL)
     VALUES (XMLVALIDATE(?))
```

入力パラメーター・マーカーは、XML スキーマ情報を含む XML 値にバインドされると想定します。

```
<po:order
    xmlns:po='http://my.world.com'
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://my.world.com/world.xsd" >
  ...
</po:order>
```

さらに、ターゲット・ネーム・スペース「http://my.world.com」と関連し、schemaLocation ヒント「http://my.world.com/world.xsd」による XML スキーマが、XML スキーマ・リポジトリー内にあると想定します。

これらの前提事項に基づき、その XML スキーマに従って入力 XML 値は妥当性検査されます。

- SQL 名 PODOCS.WORLDPO によって特定される XML スキーマを使用して妥当性検査を行います。

```
INSERT INTO T1(XMLCOL)
   VALUES(
      XMLVALIDATE(? ACCORDING TO XMLSCHEMA ID PODOCS.WORLDPO))
```

SQL 名 PODOC.WORLDPO に関連した XML スキーマが XML スキーマ・リポジトリー内にあると想定して、その XML スキーマに従って入力 XML 値は妥当性検査され、タイプにはアノテーションが付けられます。

- XML 値の指定されたエレメントを妥当性検査します。

```
INSERT INTO T1(XMLCOL)
   VALUES(
      XMLVALIDATE(?
        ACCORDING TO XMLSCHEMA ID FOO.WORLDPO
               NAMESPACE 'http://my.world.com/Mary'
               ELEMENT "po"))
```

  SQL 名 FOO.WORLDPO に関連した XML スキーマが XML スキーマ・リポジトリー内にあると想定して、XML スキーマは、ネーム・スペースが「http://my.world.com/Mary」であるエレメント「po」に対して妥当性検査されます。

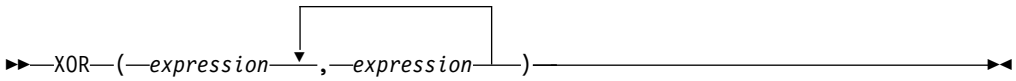
- XML スキーマは、ターゲット・ネーム・スペースおよびスキーマ・ロケーションにより特定されます。

```
INSERT INTO T1(XMLCOL)
   VALUES(
      XMLVALIDATE(?
        ACCORDING TO XMLSCHEMA URI 'http://my.world.com'
               LOCATION 'http://my.world.com/world.xsd'))
```

  ターゲット・ネーム・スペース「http://my.world.com」と関連し、schemaLocation ヒント「http://my.world.com/world.xsd」による XML スキーマが XML スキーマ・リポジトリー内にあると想定して、その XML スキーマに従って入力 XML 値は妥当性検査されます。

XOR

XOR 関数は、引数ストリングの論理 XOR であるストリングを戻します。この関数は、最初の引数ストリングを次のストリングと XOR 演算し、それから前の結果を使用して、連続する各引数との XOR 演算を繰り返します。文字ストリング引数が前の結果より短い場合は、ブランクが埋め込まれます。 2 進ストリング引数が前の結果より短い場合は、16 進数のゼロが埋め込まれます。



各引数には、互換性がなければなりません。

expression
(LOB 以外の) 任意の組み込み数値データ・タイプ、またはストリング・データ・タイプの値を戻す式。引数は、混合データ文字ストリング、UTF-8 文字ストリング、またはグラフィック・ストリングであってはなりません。数値引数は、関数を評価する前に文字ストリングにキャストされます。数値から文字ストリングへの変換の詳細については、 672 ページの『VARCHAR』を参照してください。

必要ならば、引数は結果の属性に変換されます。結果の属性は、以下のように決められます。

- すべての引数が固定長ストリングである場合、結果は長さが *n* の固定長ストリングになります (ここで、*n* は最長の引数の長さ)。
- 引数の中に可変長ストリングがある場合、結果は長さ属性が *n* の可変長ストリングになります (ここで、*n* は最大の長さ属性を持つ引数の長さ属性)。結果の実際の長さは *m* です (ここで、*m* は、最長の引数の実際の長さ)。

引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

結果の CCSID は 65535 です。

例

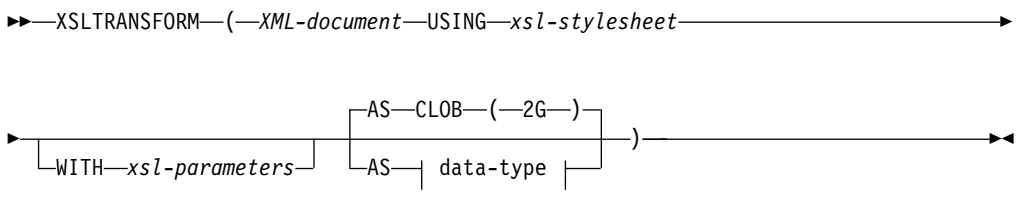
- ホスト変数 L1 は値が X'E1E1' の CHARACTER(2) のホスト変数、ホスト変数 L2 は値が X'F0F000' の CHARACTER(3) のホスト変数、ホスト変数 L3 は値が X'0000000F' の CHARACTER(4) のホスト変数であると想定します。

```
SELECT XOR(:L1,:L2,:L3)
  FROM SYSIBM.SYSDUMMY1
```

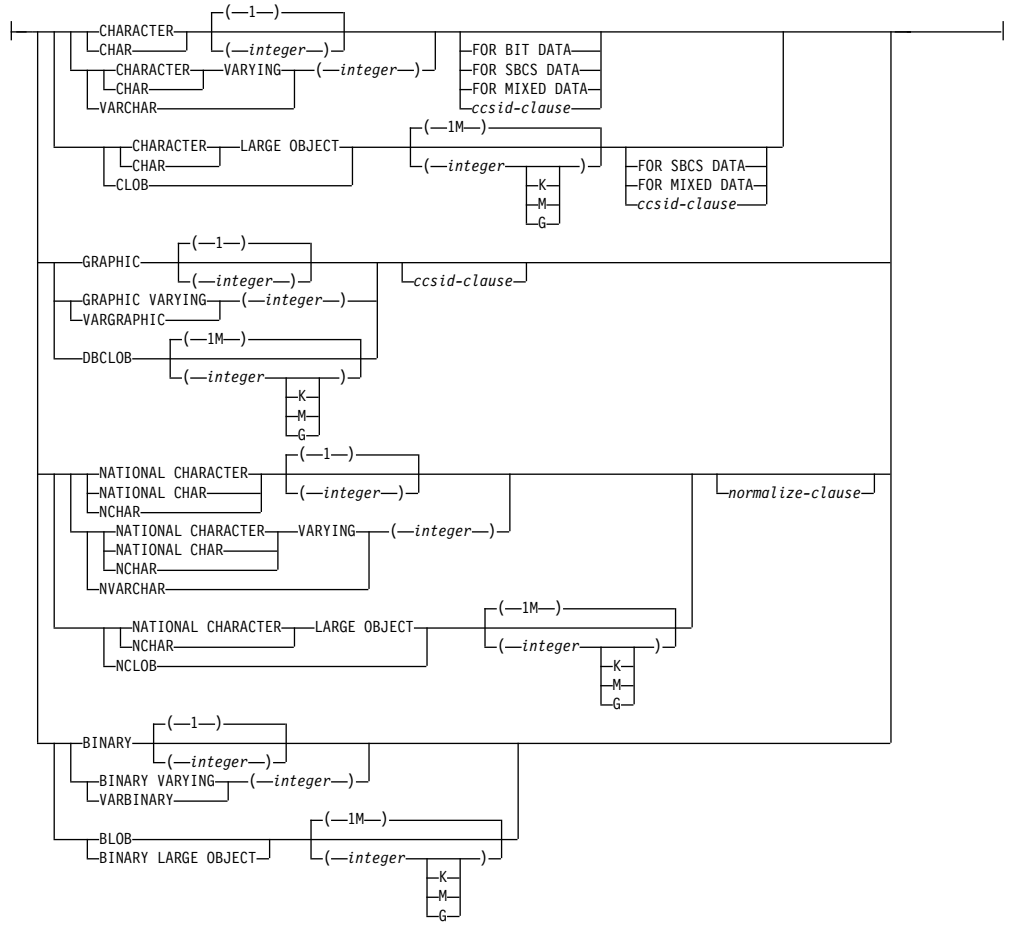
値として X'1111404F' が戻されます。

XSLTRANSFORM

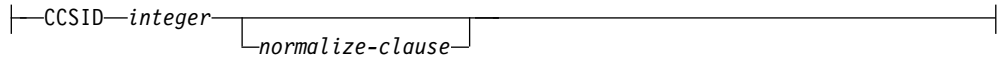
XSLTRANSFORM は、XML 文書を他のデータ形式に変換します。データは XSLT プロセッサーで可能なあらゆる形式、例えば、XML、HTML、またはプレーン・テキスト (ただし必ずしもこれらに限定されない) などに変換できます。



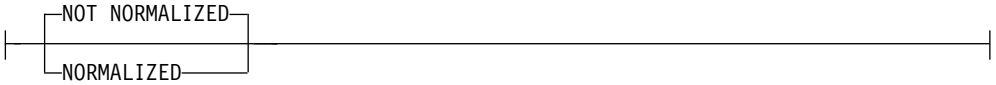
data-type:



ccsid-clause:



normalize-clause:



XSLTRANSFORM を使用して XML データを他の形式に変換します。これには、1 つの XML スキーマに準拠する XML 文書を別の XML スキーマに準拠する文書に変換することも含まれます。

XML-document
整形式 XML 文書を戻す文字ストリング、Unicode グラフィック・ストリング、2 進ストリング、または XML 式。これは、*xsl-stylesheet* で指定された XSL スタイルシートを使用して変換される文書です。

xsl-stylesheet
整形式 XML 文書を戻す文字ストリング、Unicode グラフィック・ストリング、2 進ストリング、または XML 式。文書は XSLT バージョン 1.10 勧告に準拠した XSL スタイルシートです。xsl:include 宣言を取り込むスタイルシートはサポートされていません。このスタイルシートは、*xml-document* で指定された値を変換するために適用されます。

xsl-parameters
整形式 XML 文書を戻す文字ストリング、Unicode グラフィック・ストリング、2 進ストリング、または XML 式。これは、*xsl-stylesheet* で指定された XSL スタイルシートにパラメーター値を提供する文書です。パラメーターの値は、属性またはテキストとして指定できます。

パラメーター文書の構文は次のとおりです。

```
<params xmlns="http://www.ibm.com/XSLTransformParameters">
<param name="..." value="..."/>
<param name="...">enter value here</param> ... </params>
```

注: スタイルシート文書には xsl:param エレメントが含まれている必要があり、パラメーター文書で指定されたものと一致する名前属性値がなければなりません。

AS *data-type*
結果のデータ・タイプを指定します。指定された結果のデータ・タイプの暗黙的または明示的な長さ属性は、変換された出力を収容するのに十分な大きさでなければなりません。デフォルトの結果のデータ・タイプは CLOB(2G) です。

CCSID が指定され、*data-type* が GRAPHIC、VARGRAPHIC、または DBCLOB の場合、CCSID は Unicode CCSID であることが必要です。

CCSID 属性が指定されないと、CCSID は 218 ページの『CAST の指定』に記されているように、*XML-document* が *data-type* にキャストされた場合のように決定されます。

関数の結果は、指定のデータ・タイプになります。前述の文書を文字データ・タイプで保管すると、CCSID 変換時にデータが失われる恐れがあります。

XML-document と *xsl-stylesheet* のどちらかが NULL の場合、結果は NULL 値になります。

注記

前提条件: XSLTRANSFORM 関数を使用するには、XML Toolkit for IBM i および International Components for Unicode (ICU オプション) がインストールされていなければなりません。

例

この例では、XSLT をフォーマット・エンジンとして使用する方法を示します。セットアップのために、まず以下の 2 つの例の文書をデータベースに挿入します。

```
CREATE TABLE XML_TAB (c1 INT, xml_doc CLOB(2M), xsl_doc CLOB(256K));
INSERT INTO XML_TAB VALUES
  (1, '<?xml version="1.0"?>
<students xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation = "/home/steffen/xsd/xslt.xsd">
<student studentID="1" firstName="Steffen" lastName="Siegmund"
    age="23" university="Rostock"/>
</students>',

'<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
            xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:param name="headline"/>
<xsl:param name="showUniversity"/>

  <xsl:template match="students">
    <html>
      <head/>
        <body>
          <h1><xsl:value-of select="$headline"/></h1>
          <table border="1">
            <th>
              <tr>
                <td width="80">StudentID</td>
                <td width="200">First Name</td>
                <td width="200">Last Name</td>
                <td width="50">Age</td>
                <xsl:choose>
                  <xsl:when test="$showUniversity =''true''">
                    <td width="200">University</td>
                  </xsl:when>
                </xsl:choose>
              </tr>
            </th>
          <xsl:apply-templates/>
          </table>
        </body>
    </html>
  </xsl:template>
  <xsl:template match="student">
    <tr>
      <td><xsl:value-of select="@studentID"/></td>
      <td><xsl:value-of select="@firstName"/></td>
      <td><xsl:value-of select="@lastName"/></td>
      <td><xsl:value-of select="@age"/></td>
      <xsl:choose>
        <xsl:when test="$showUniversity = ''true''">
          <td><xsl:value-of select="@university"/></td>
        </xsl:when>
      </xsl:choose>
    </tr>
  </xsl:template>
</xsl:stylesheet>');
```

次に、XSLTRANSFORM 関数を呼び出して XML データを HTML に変換し、表示します。

```
SELECT XSLTRANSFORM (XML_DOC USING XSL_DOC
  WITH '<params xmlns="http://www.ibm.com/XSLTransformParameters"></params>')
FROM XML_TAB;
```

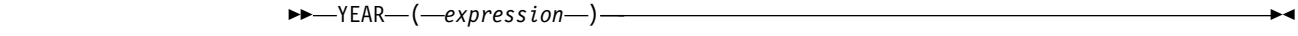
結果は、次の文書になります。

```
<html>
<head>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<h1></h1>
<table border="1">
<th>
<tr>
<td width="80">StudentID</td>
<td width="200">First Name</td>
<td width="200">Last Name</td>
<td width="50">Age</td>
</tr>
</th>
<tr>
<td>1</td>
<td>Steffen</td>
<td>Siegmund</td>
<td>23</td>
</tr>
</table>
</body>
</html>
```

この例では、出力は HTML であり、各パラメーターは生成される HTML と、それに含まれるデータのみに影響を与えます。このため、これはエンド・ユーザーの出力用のフォーマット・エンジンとしての XSLT の使用例を示しています。

YEAR

YEAR 関数は、指定された値の年の部分を戻します。



expression
日付、タイム・スタンプ、文字ストリング、グラフィック・ストリング、または数値のいずれかの組み込みデータ・タイプの値を戻す式。

- *expression* が文字ストリングまたはグラフィック・ストリングの場合、その値は、日付またはタイム・スタンプの有効なストリング表現でなければなりません。日付とタイム・スタンプのストリング表現の有効な形式については、 95 ページの『日付/時刻の値のストリング表記』を参照してください。
- 式 が数値である場合は、その数値は日付期間またはタイム・スタンプ期間でなければなりません。日時期間の有効な形式については、 205 ページの『日付/時刻のオペランドと期間』を参照してください。

この関数の結果は長精度整数になります。引数が NULL になる可能性がある場合は、結果も NULL になる可能性があります。引数が NULL の場合は、結果は NULL 値になります。

その他の規則は、引数のデータ・タイプに応じて以下のように異なります。

- 引数が日付またはタイム・スタンプ、または、日付またはタイム・スタンプの有効な文字ストリング表現である場合 :

  結果は、指定した値の年の部分 (1 から 9999 までの整数) になります。
- 引数が日付期間またはタイム・スタンプ期間の場合 :

  結果は、指定した値の年の部分 (-9999 から 9999 までの整数) になります。ゼロ以外の結果の符号は、引数と同じになります。

例

- 表 PROJECT から、開始 (PRSTDATE) と終了 (PRENDATE) が同じ年に予定されているプロジェクトをすべて選択します。

```
SELECT *
  FROM PROJECT
  WHERE YEAR(PRSTDATE) = YEAR(PRENDATE)
```

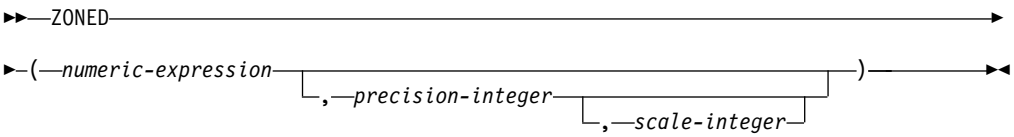
- 表 PROJECT から、1 年未満で完了するように予定されているプロジェクトをすべて選択します。

```
SELECT *
  FROM PROJECT
  WHERE YEAR(PRENDATE - PRSTDATE) < 1
```

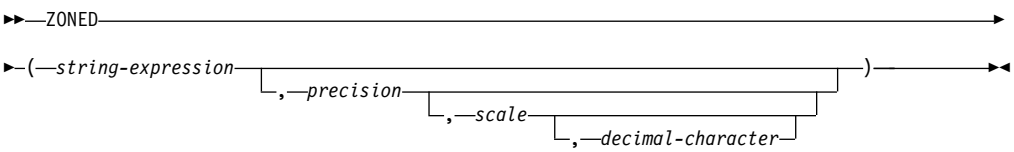
ZONED

ZONED 関数は、ゾーン 10 進数表現を返します。

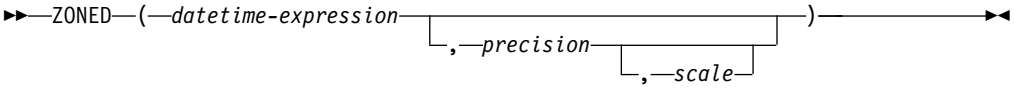
数値からゾーン **10** 進数に



ストリングからゾーン **10** 進数に



日付/時刻→ **10** 進数



ZONED 関数は、次のもののゾーン 10 進数表現を戻します。

- 数値
- 10 進数の文字ストリング表現またはグラフィック・ストリング表現
- 整数の文字ストリング表現またはグラフィック・ストリング表現
- 浮動小数点数の文字ストリング表現またはグラフィック・ストリング表現
- 10 進浮動小数点数の文字ストリング表現またはグラフィック・ストリング表現
- 日付
- 時刻
- タイム・スタンプ

数値からゾーン 10 進数に

numeric-expression
任意の組み込み数値データ・タイプの値を戻す式。

precision
1 以上で 63 以下の値を持つ整数定数。

デフォルトの精度 は、数値式 のデータ・タイプによって決まります。

- 5 (最初の引数が短整数の場合)
- 11 (最初の引数が長整数の場合)
- 19 (最初の引数が 64 ビット整数の場合)
- 15 (最初の引数が浮動小数点数、10 進数、数字、または位取りがゼロ以外の 2 進数の場合)

- 31 (10 進浮動小数点の場合)

scale
0 以上で精度 以下である整数定数。これを指定しないと、デフォルト値の 0 になります。

結果は、最初の引数が、精度 *precision*、位取り *scale* の 10 進数の列または変数に割り当てられたときに得られる数値と同じです。数値全体を表すのに必要な有効 10 進数字の桁数が *precision-scale* よりも大きい場合は、エラーが戻されます。最初の引数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引数がヌルの場合は、結果は NULL 値になります。

ストリングからゾーン 10 進数に

string-expression
数値の文字ストリング表現またはグラフィック・ストリング表現の値を戻す式。先行ブランクと末尾ブランクは除去され、結果のストリングは、浮動小数点数、10 進浮動小数点数、整数、または 10 進数の定数を形成する際の規則に合致している必要があります。

precision
1 以上で 63 以下である整数定数。これを指定しないと、デフォルト値の 15 になります。

scale
0 以上で精度 以下である整数定数。これを指定しないと、デフォルト値の 0 になります。

decimal-character
数値の整数部分からストリング式 の小数桁数を区切るために使用された 1 バイトの文字定数を指定します。この文字は、ピリオドかコンマとする必要があります。 2 番目の引数を指定しなかった場合は、デフォルトの小数点が使用されます。詳しくは、 147 ページの『小数点』を参照してください。

decimal-character の右側の桁数が位取り *s* より大きい場合は、末尾の桁が切り捨てられます。 *string-expression* の中の *decimal-character* より左側にある有効桁数 (数値の整数部分) が *precision-scale* より大きい場合は、エラーが戻されます。小数点文字 引数の指定がある場合は、サブストリング内のデフォルト小数点文字は無効です。

日時からゾーン 10 進数に

datetime-expression
タイプ DATE、TIME、または TIMESTAMP の値を戻す式。

precision
結果の精度を指定する、1 以上で 63 以下の整数定数。この指定がない場合、精度および位取りのデフォルトは、以下のように *datetime-expression* のデータ・タイプによって決まります。

- DATE の場合は、精度が 8 で、位取りが 0 です。結果は、日付を *yyyymmdd* で表した NUMERIC(8,0) 値になります。
- TIME の場合は、精度が 6 で、位取りが 0 です。結果は、時間を *hhmmss* で表した NUMERIC(6,0) 値になります。

- TIMESTAMP(*tp*) の場合は、精度が 14+*tp* で、位取りが *tp* です。結果は、タイム・スタンプを *yyyymmddhhmmss.nnnnnnnnnnnn* で表した NUMERIC(14+*tp*、*tp*) 値になります。

scale
0 以上で精度 以下である整数定数。この指定がなく、*precision* が指定されている場合、デフォルトは 0 です。

結果は、CAST (*datetime-expression* AS NUMERIC*precision,scale*)) によって得られる数と同じです。小数点の右側にある桁数が位取り *s* よりも多い場合、末尾から桁が切り捨てられます。 *datetime-expression* の小数点の左側にある有効数字 (数値の整数部分) の桁数が *precision* - *scale* より多い場合、エラーが戻されます。

この関数の結果は、精度が *precision* で位取りが *scale* であるゾーン 10 進数です。最初の引数がヌルである可能性がある場合は、結果もヌルになる可能性があります。最初の引数がヌルの場合は、結果は NULL 値になります。

注記

代替構文: 精度を指定する場合、アプリケーションの移植性を拡張するには、CAST 指定を使用します。詳しくは、 218 ページの『CAST の指定』を参照してください。

例

- ホスト変数 Z1 は、値が 1.123 の 10 進数ホスト変数であると想定します。

```
SELECT ZONED(:Z1,15,14)
  FROM SYSIBM.SYSDUMMY1
```

値として 1.12300000000000 が戻されます。

- ホスト変数 Z1 は、値が 1123 の 10 進数ホスト変数であると想定します。

```
SELECT ZONED(:Z1,11,2)
  FROM SYSIBM.SYSDUMMY1
```

値として 1123.00 が戻されます。

- 同様に、

```
SELECT ZONED(:Z1,4)
  FROM SYSIBM.SYSDUMMY1
```

値として 1123 が戻されます。

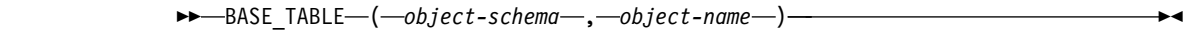
表関数

表関数は表の列を戻し、それは、CREATE TABLE ステートメントで作成された表と似ています。

表関数は、SQL ステートメントの FROM 文節でのみ使用できます。表関数は、スキーマ名で修飾することができます。

BASE_TABLE

BASE_TABLE 関数は、別名に対して検出されたオブジェクトのオブジェクト名とスキーマ名を戻します。



スキーマは SYSPROC です。

object-schema
指定された *object-name* を修飾するために使用される SQL スキーマ名またはシステム・スキーマ名を示す文字ストリング式またはグラフィック・ストリング式。 *object-schema* の実際の長さは 129 文字未満でなければなりません。 特殊値 *LIBL を指定することができ、その場合は、ライブラリー・リスト内で見つかった、*object-name* という名前のファイルの最初のインスタンスが使用されます。この名前には大/小文字の区別があり、引用符で区切られていてはなりません。

object-name
解決するオブジェクトの SQL 名またはシステム名を示す文字ストリング式またはグラフィック・ストリング式。*object-name* の実際の長さは 129 文字未満でなければなりません。この名前には大/小文字の区別があり、引用符で区切られていてはなりません。

指定されたオブジェクトが別名を参照しないか、そのオブジェクトが見つからない場合、この関数の結果は入力のオブジェクト名とスキーマになります。

この関数の結果は、次の表に示された形式の、単一の行を含んでいる表です。列はすべてNULL 可能です。

表 *61. BASE_TABLE* の結果表の形式

列名	データ・タイプ	内容
BASESCHEMA	VARCHAR(128)	別名が参照する表またはビューが含まれている SQL スキーマの名前。別名が見つからなかった場合、これは *object-schema* です。この名前は引用符で区切られず、大/小文字の区別があります。
BASENAME	VARCHAR(128)	別名が参照する表またはビューの名前。別名が見つからなかった場合、これは *object-name* です。この名前は引用符で区切られず、大/小文字の区別があります。
SYSTEM_TABLE_SCHEMA	CHAR(10)	システムのスキーマ名。別名が見つからなかったか、別名がリモート RDB を参照する場合、この列には NULL 値が入ります。この名前は引用符で区切られることがあり、大/小文字の区別があります。
SYSTEM_TABLE_NAME	CHAR(10)	システム表名。別名が見つからなかったか、別名リモート RDB を参照する場合、この列には NULL 値が入ります。この名前は引用符で区切られることがあり、大/小文字の区別があります。

表 *61. BASE_TABLE* の結果表の形式 *(*続き*)*

列名	データ・タイプ	内容
MEMBER_NAME	CHAR(10)	メンバー別名に対して識別されたメンバー名。別名が見つからなかったか、別名が特定のメンバーを参照しない場合、この列には NULL 値が入ります。この名前は引用符で区切られることがあり、大/小文字の区別があります。
RDBNAME	VARCHAR(128)	オブジェクトがリモート・オブジェクトの 3 部構成の別名の場合、RDB。別名が見つからなかったか、別名に関連付けられた RDB がない場合、この列には NULL 値が入ります。

結果列の CCSID は、現行サーバーでのデフォルト CCSID です。

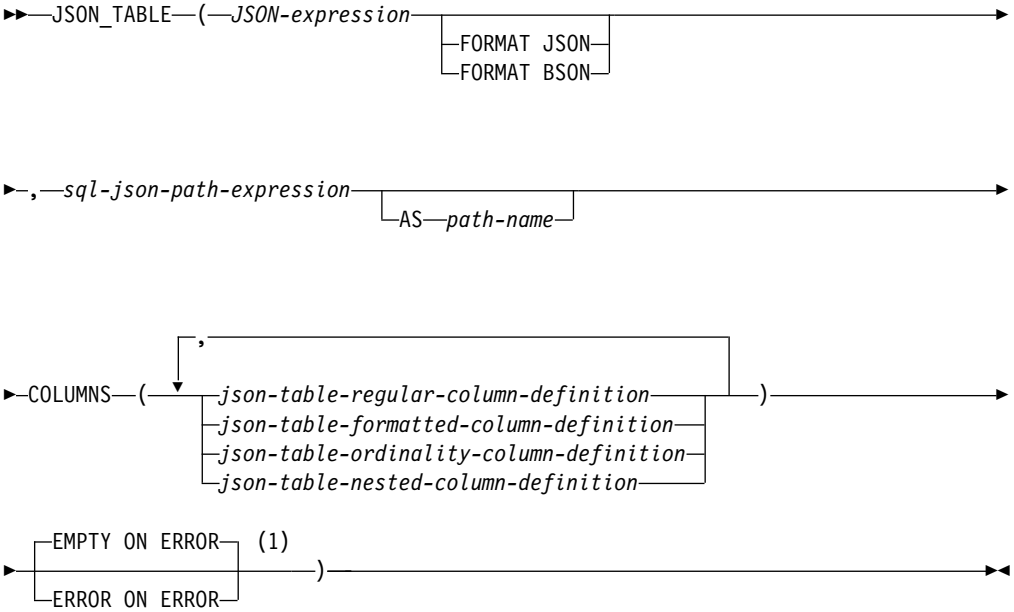
例

- 以下の照会は、SYSTABLES 内で識別された各別名の基本表情報を戻します。

```
SELECT C.BASESCHEMA, C.BASENAME
FROM QSYS2.SYSTABLES A,
LATERAL (
    SELECT * FROM TABLE(SYSPROC.BASE_TABLE(A.TABLE_SCHEMA,A.TABLE_NAME)) AS X)
         AS C
WHERE A.TABLE_TYPE='A'
```

JSON_TABLE

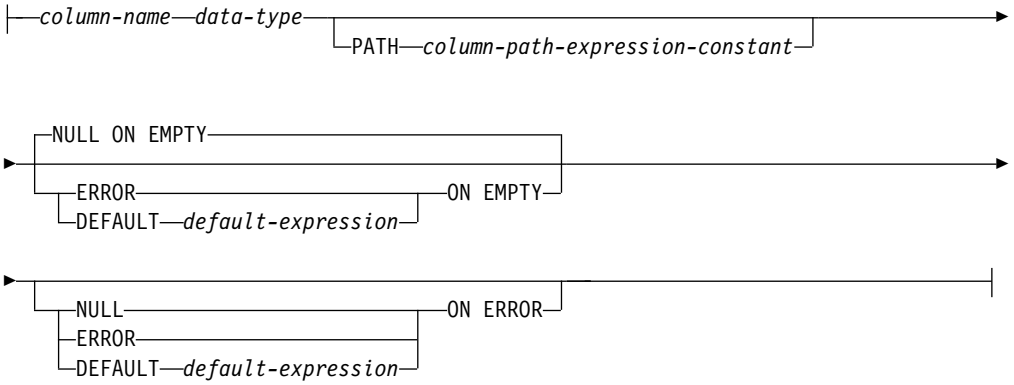
JSON_TABLE 表関数は結果表を SQL/JSON パス式の評価から戻します。行 SQL/JSON パス式の結果シーケンス内の各項目は、結果表の 1 つ以上の行を表しています。



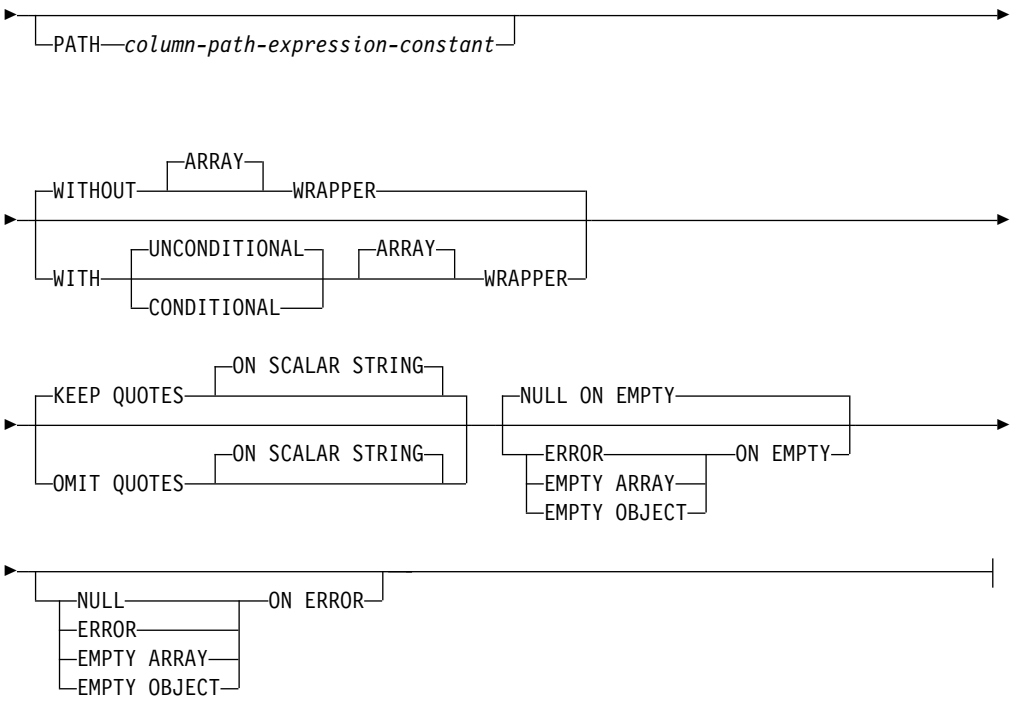
注:

1 ON ERROR 節は、オプションで COLUMNS 節の前に指定することもできます。

json-table-regular-column-definition:

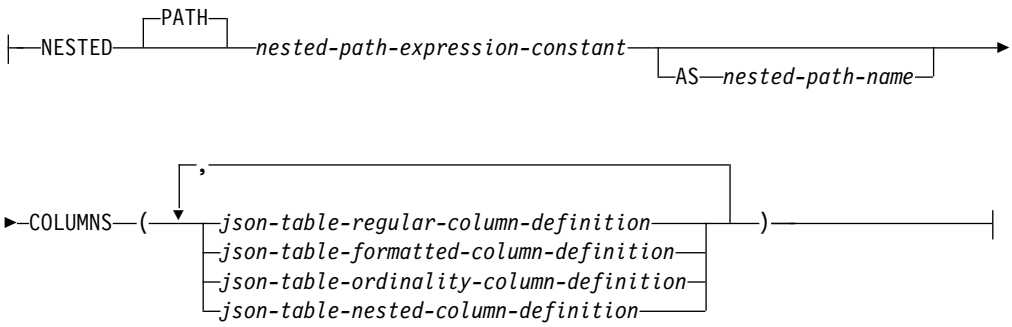


json-table-formatted-column-definition:

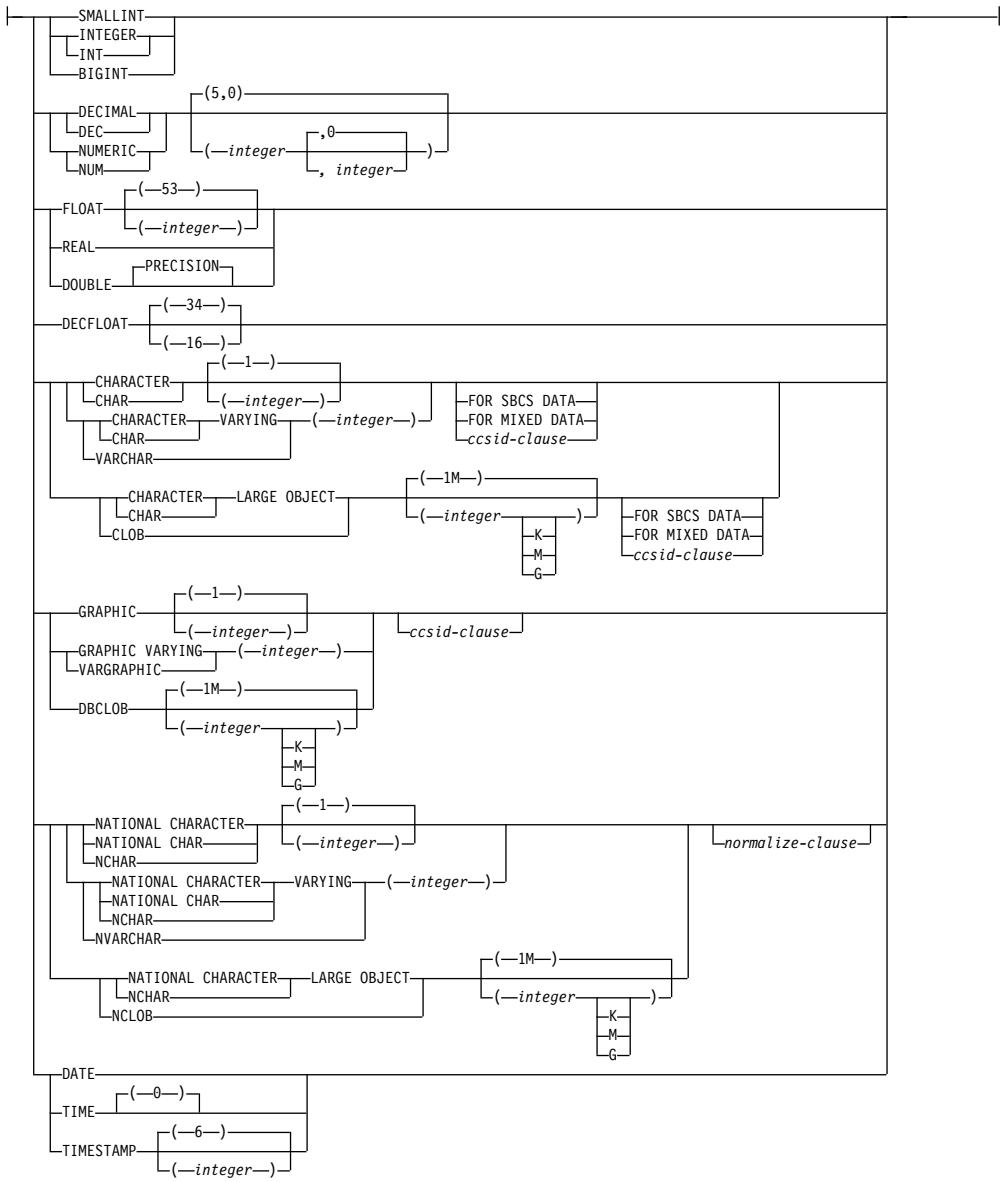


json-table-ordinality-column-definition:

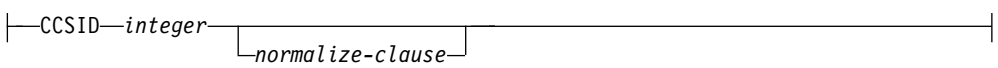
json-table-nested-column-definition:



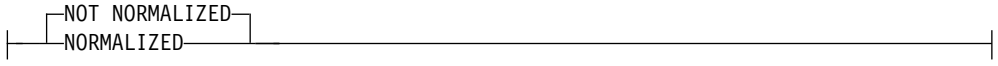
data-type:



ccsid-clause:



normalize-clause:



JSON-expression
文字ストリング、グラフィック・ストリング、またはバイナリー・ストリングの値を戻す式を指定します。文字またはグラフィックの値が戻される場合は、正しい形式の JSON データが含まれている必要があります。バイナリーの値が戻される場合は、 JSON データの BSON 表現が含まれている必要があります。

JSON-expression は、$ として *sql-json-path-expression* 内で識別される、*sql-json-path-expression* の初期コンテキスト項目を指定します。

JSON-expression に正しい形式のデータが含まれない場合、ON ERROR 節で別のエラー動作が指定されていない限り、JSON_TABLE は空の表を戻します。

FORMAT JSON または **FORMAT BSON**
JSON-expression の解釈方法を指定します。

FORMAT JSON
JSON-expression には JSON データが含まれていることを示します。*JSON-expression* がバイナリー・データの場合、データは UTF-8 または UTF-16 として解釈されます。EBCDIC CCSID を使用してバイナリー・データをエンコードすることはできません。

FORMAT BSON
JSON-expression には、JSON データの BSON 表現が含まれていることを示します。FORMAT BSON を指定した場合、*JSON-expression* はバイナリー・ストリング・データ・タイプでなければなりません。

FORMAT 節を指定しなかった場合、*JSON-expression* が文字ストリングまたはグラフィック・ストリングであれば、*JSON-expression* は JSON として扱われます。 *JSON-expression* がバイナリー・ストリングの場合、*JSON-expression* は BSON として扱われます。

sql-json-path-expression
SQL/JSON パス式として解釈される文字またはグラフィック・ストリング式を指定します。この式は、出力シーケンスを戻します。このシーケンス内の各項目が列定義によって使用され、出力表の 1 つ以上の行が生成されます。出力シーケンスが空の場合、JSON_TABLE の結果は空の表です。 *sql-json-path-expression* に、空ストリングまたはすべてがブランクのストリングを指定してはいけません。

SQL/JSON パス式の内容については、 260 ページの『sql-json-path-expression』を参照してください。

AS *path-name*
sql-json-path-expression の識別に使用する名前を指定します。

EMPTY ON ERROR または **ERROR ON ERROR**
表レベルのエラーが発生したときの、JSON_TABLE の必要な動作を指定します。

EMPTY ON ERROR
表レベルのエラーが発生すると、空の表が戻されます。これはデフォルトです。

ERROR ON ERROR
表レベルのエラーが発生すると、エラーが戻されます。

COLUMNS
列名、データ・タイプ、および各行の列値の計算方法を組み込んで、結果表の出力列を指定します。

すべての結果列の長さの合計は 64K バイトを超えてはなりません。データ・タイプごとの列のバイト・カウントについては、 1295 ページの『最大行サイズ』を参照してください。列名の長さと列パスの長さに応じて、この関数は、最大約 200 列を戻すことができます。

json-table-regular-column-definition
結果表の出力列を 1 つ指定します。列名、データ・タイプ、行のシーケンス項目から値を抽出するための SQL/JSON パス式などを指定します。

column-name
結果表の列の名前を指定します。名前を修飾することはできず、結果表の複数の列に対して同じ名前を使用することはできません。

data-type
列のデータ・タイプを指定します。CHAR 列および VARCHAR 列の場合、CCSID を 65535 にすることはできません。

PATH *column-path-expression-constant*
SQL/JSON パスとして解釈される文字またはグラフィック・ストリング定数を指定します。

column-path-expression-constant は SQL/JSON パス式を指定しますが、これは *sql-json-path-expression* 内の SQL/JSON パス式の評価の結果である項目に関連して列値を決定します。外部で提供されたコンテキスト項目として *sql-json-path-expression* の処理の結果による項目がある場合、*column-path-expression-constant* が評価され、出力シーケンスが戻されます。同じキーの複数の値が JSON オブジェクトに含まれる場合、キーに対するいずれか 1 つの値だけが出力シーケンスに戻されます。

この出力シーケンスに基づいて、列値は以下のように決定されます。

- 空シーケンスが戻された場合、列の値は ON EMPTY 節から取得されます。ERROR ON EMPTY が指定されると、エラーが出されます。
- 空シーケンスが戻され、ON EMPTY 節が指定されていない場合、列には NULL 値が割り当てられます。
- 単一エレメントのシーケンスが戻され、エレメントのタイプが JSON 配列でも JSON オブジェクトでもない場合、その値は列に指定された *data-type* に変換されます。
- 単一エレメントのシーケンスが戻され、エレメントのタイプが JSON 配列または JSON オブジェクトである場合、エラーが戻されます。
- 複数のエレメントを含むシーケンスが戻された場合は、エラーが戻されます。
- エラーが発生した場合、列の値は ON ERROR 節によって指定されます。

column-path-expression-constant の値は、空ストリングまたはすべてがブランクのストリングにすることはできません。PATH 節を指定しなかった場合、*column-path-expression-constant* は、*column-name* の先頭に '$.' を付けたものとして定義されます。

ON EMPTY
列で空シーケンスが戻された場合の動作を指定します。

NULL ON EMPTY
SQL NULL 値が戻されます。これはデフォルトです。

ERROR ON EMPTY
エラーが戻されます。

DEFAULT *default-expression* **ON EMPTY**
default-expression で指定された値が戻されます。

ON ERROR
列でエラーが戻された場合の動作を指定します。

NULL ON ERROR
SQL NULL 値が戻されます。これはデフォルトです。

ERROR ON ERROR
エラーが戻されます。

DEFAULT *default-expression* **ON ERROR**
default-expression で指定された値が戻されます。

この文節を指定しない場合:

- 表レベルの ERROR ON ERROR 節が指定されると、エラーが戻されます。
- 指定されていない場合は、SQL NULL 値が戻されます。

json-table-formatted-column-definition
結果表の出力列を 1 つ指定します。列名、データ・タイプ、行のシーケンス項目から値を抽出するための SQL/JSON パス式などを指定します。抽出された値は、JSON ストリングとして形式設定されます。

column-name
結果表の列の名前を指定します。名前を修飾することはできず、結果表の複数の列に対して同じ名前を使用することはできません。

data-type
列のデータ・タイプを指定します。データ・タイプは、文字またはグラフィック・タイプでなければなりません。CHAR 列および VARCHAR 列の場合、CCSID を 65535 にすることはできません。

FORMAT JSON
取得されるデータが JSON ストリングとして形式設定されることを示します。

PATH *column-path-expression-constant*
SQL/JSON パスとして解釈される文字またはグラフィック・ストリング定数を指定します。

column-path-expression-constant は SQL/JSON パス式を指定しますが、これは *sql-json-path-expression* 内の SQL/JSON パス式の評価の結果である項目と、前の NESTED PATH で指定されたすべてのパスに関連して列値を決定します。外部で提供されたコンテキスト項目として *sql-json-path-expression* の処理の結果による項目がある場合、*column-path-expression-constant* が評価され、出力シーケンスが戻されます。同じキーの複数の値が JSON オブジェクトに含まれる場合、キーに対するいずれか 1 つの値だけが出力シーケンスに戻されます。

この出力シーケンスに基づいて、列値は以下のように決定されます。

- 空シーケンスが戻された場合、列の値は ON EMPTY 節から取得されます。ERROR ON EMPTY が指定されると、エラーが戻されます。
- 空シーケンスが戻され、ON EMPTY 節が指定されていない場合、列には NULL 値が割り当てられます。
- エラーが発生した場合、列の値は ON ERROR 節によって指定されます。

column-path-expression-constant の値は、空ストリングまたはすべてがブランクのストリングにすることはできません。PATH 節を指定しなかった場合、*column-path-expression-constant* は、*column-name* の先頭に '$.' を付けたものとして定義されます。

WITHOUT ARRAY WRAPPER または WITH ARRAY WRAPPER
JSON 配列内で出力値をラップするかどうかを指定する必要があります。

WITHOUT ARRAY WRAPPER
結果をラップしないことを示します。これはデフォルトです。複数の SQL/JSON エレメントを含むシーケンスになる SQL/JSON パスを使用すると、エラーになります。

WITH UNCONDITIONAL ARRAY WRAPPER
結果を大括弧で囲んで JSON 配列を作成することを示します。

WITH CONDITIONAL ARRAY WRAPPER
複数の SQL/JSON エレメントが戻される場合、結果を大括弧で囲んで JSON 配列を作成することを示します。

KEEP QUOTES または OMIT QUOTES
スカラー・ストリングが戻された場合に、周囲の引用符を除去する必要があるかどうかを指定します。

KEEP QUOTES
引用符がスカラー・ストリングから除去されないことを示します。これはデフォルトです。

OMIT QUOTES
引用符がスカラー・ストリングから除去されることを示します。OMIT QUOTES を指定した場合、WITH ARRAY WRAPPER 節は指定できません。

ON EMPTY
列で空シーケンスが戻された場合の動作を指定します。

NULL ON EMPTY
SQL NULL 値が戻されます。これはデフォルトです。

ERROR ON EMPTY
エラーが戻されます。

EMPTY ARRAY ON EMPTY
空の JSON 配列が戻されます。

EMPTY OBJECT ON EMPTY
空の JSON オブジェクトが戻されます。

ON ERROR
列でエラーが戻された場合の動作を指定します。

NULL ON ERROR
SQL NULL 値が戻されます。

ERROR ON ERROR
エラーが戻されます。

EMPTY ARRAY ON ERROR
空の JSON 配列が戻されます。

EMPTY OBJECT ON ERROR
空の JSON オブジェクトが戻されます。

この文節を指定しない場合:

- 表レベルの ERROR ON ERROR 節が指定されると、エラーが戻されます。
- 指定されていない場合は、SQL NULL 値が戻されます。

json-table-ordinality-column-definition
結果表の順序を示す列を指定します。

column-name
結果表の列の名前を指定します。名前を修飾することはできず、結果表の複数の列に対して同じ名前を使用することはできません。

FOR ORDINALITY
column-name は、包含しているネスティング・レベルの結果表の順序を示す列であることを指定します。この列のデータ・タイプは BIGINT です。

- この順序列が、ネストされた列の定義内にない場合、結果表の行が 1 から順に番号付けされ、順序列は現在行のシーケンス番号を含みます。
- この順序列が、ネストされた列の定義内にある場合、包含しているネストされた列の定義で生成された行が 1 から順に番号付けされ、順序列は現在行のシーケンス番号を含みます。包含しているネストされた列の定義が外部のネストされた列の定義内にある場合、包含しているネストされた列の定義のパス式が外部のネストされた列の定義の結果に適用されるたびに、番号付けは再び 1 から開始します。

json-table-nested-column-definition
現行レベルでネストされた 1 つ以上の列を指定します。

NESTED PATH *nested-path-expression-constant*
SQL/JSON パスとして解釈される文字またはグラフィック・ストリング定数を指定します。*nested-path-expression-constant* は SQL/JSON パス式を指定しますが、これは *sql-json-path-expression* 内の SQL/JSON パス式の評価の結果である項目と、前の NESTED PATH の *nested-path-expression-constant* に関連して列値を決定します。外部で提供されたコンテキスト項目として *sql-json-path-expression* の処理の結果

による項目があり、前の NESTED PATH の *nested-path-expression-constant* がある場合、*nested-path-expression-constant* が評価され、ネストされた列のコンテキストとして使用されます。

AS *nested-path-name*
現行パスの名前を指定します。

COLUMNS
このネスティング・レベルに含まれる列を指定します。

表 *62. JSON* から *SQL* 結果列へのサポートされる変換

JSON タイプ	**SQL** タイプ	注
番号	SMALLINT INTEGER BIGINT	ソース値がターゲット・データ・タイプの範囲を超える場合、オーバーフロー・エラーが戻されます。
番号	DECIMAL NUMERIC	結果の数値は、必要に応じて、ターゲット・データ・タイプの精度と位取りに変換されます。必要な数の先行ゼロが追加または除去されます。数字の小数部分では、必要な数の後続ゼロが追加されるか、必要な桁数が除去されます。この切り捨て動作は、DECIMAL から DECIMAL へのキャストの動作と同様です。
番号	FLOAT DOUBLE REAL DECFLOAT	ソース値がターゲット・データ・タイプの範囲を超える場合、オーバーフロー・エラーが戻されます。ソース値にターゲット・データ・タイプの精度を超える有効数字が含まれている場合、ソース値はターゲット・データ・タイプの精度に丸められます。
ストリングまたは数値	CHAR VARCHAR CLOB GRAPHIC VARGRAPHIC DBCLOB	結果の値は、必要に応じて、限定された長さのターゲット・タイプに変換される前に、 120 ページの『割り当ての際の変換規則』に説明されている規則を使用してターゲット・データ・タイプの CCSID に変換されます。指定された長さの制限が CCSID 変換後のストリングの長さより短い場合、切り捨てが行われます。非ブランク文字が切り捨てられる場合は、警告が出されます。ターゲット・タイプが固定長ストリング (CHAR または GRAPHIC) であり、ターゲット・タイプの指定の長さが CCSID 変換後のストリングの長さより長い場合、末尾にブランクが埋め込まれます。この切り捨ておよび埋め込みの動作は、文字ストリングまたはグラフィック・ストリングの取り出し割り当てと同様です。
ストリング	DATE	JSON ストリングは、日付値に変換されます。JSON ストリングは、以下のいずれかの形式でなければなりません。 **ISO** yyyy-mm-dd **USA** mm/dd/yyyy **EUR** dd.mm.yyyy **JIS** yyyy-mm-dd

表 62. JSON から SQL 結果列へのサポートされる変換 (続き)

JSON タイプ	**SQL** タイプ	注
ストリング	TIME	JSON ストリングは、時刻値に変換されます。JSON ストリングは、以下のいずれかの形式でなければなりません。 **ISO** hh.mm.ss **EUR** hh.mm.ss **JIS** hh:mm:ss **HMS** hh:mm:ss
ストリング	TIMESTAMP	JSON ストリングは、タイム・スタンプ値に変換されます。JSON ストリングは、以下のいずれかの形式でなければなりません。 **ISO** yyyy-mm-dd hh:mm:ss.nnnnnn **IBMSQL** yyyy-mm-dd-hh.mm.ss.nnnnnn **ISO-8601** yyyy-mm-ddThh:mm:ss.nnnnnn<+/-Offset>
配列またはオブジェクト	CHAR VARCHAR CLOB GRAPHIC VARGRAPHIC DBCLOB	FORMAT JSON 列を使用して戻されなければなりません。 JSON 配列またはオブジェクトは、JSON 形式文字ストリングとして戻されます。行われる変換は、ストリング・タイプの場合と同じです。
NULL	任意	JSON NULL 値は、SQL NULL 値に変換されます。
Boolean	CHAR VARCHAR CLOB GRAPHIC VARGRAPHIC DBCLOB	JSON Boolean 値は、 `true` または `false` ストリングに変換され、 JSON ストリングの変換規則を使用してストリングとして戻されます。

例

以下の例では、次の JSON 文書を操作します。

```
{
  "id" : 901,
  "name" : { "first":"John", "last":"Doe" },
  "phones": [ { "type":"home", "number":"555-3762"},
              { "type":"work", "number":"555-8792"}]
}
```

- 従業員 ID、名、姓、1 つ目の電話のタイプと番号をリストします。

```
SELECT U."id", U."first name",U."last name",U."phone type",U."phone number"
  FROM EMPLOYEE_TABLE E
    JSON_TABLE(E.jsondoc,
               'lax $'
               COLUMNS( "id" INTEGER,
                        "first name"  VARCHAR(20) PATH 'lax $.name.first',
```

```
                    "last name"   VARCHAR(20) PATH 'lax $.name.last',
                    "phone type"  VARCHAR(20) PATH 'lax $.phones[0].type',
                    "phone number"  VARCHAR(20) PATH 'lax $.phones[0].number')
                  ) AS U
```

これは、以下のものを戻します。

```
id    first name    last name    phone type    phone number
---   ----------    ---------    ----------    ------------
901   John          Doe          home          555-3762
```

- 従業員 ID、名、姓、入手可能なすべての電話のタイプと番号をリストします。

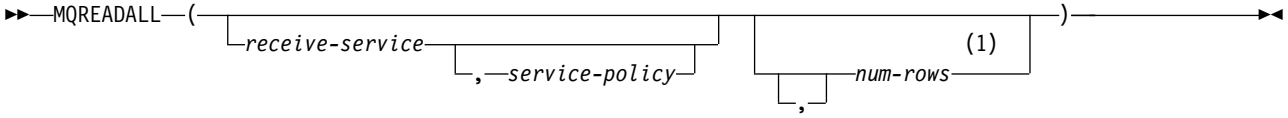
```
SELECT U."id", U."first name",U."last name",U."phone type",U."number" AS "phone number"
  FROM EMPLOYEE_TABLE E
    JSON_TABLE(E.jsondoc,
               'lax $'
                COLUMNS( "id" INTEGER,
                         "first name"  VARCHAR(20) PATH 'lax $.name.first',
                         "last name"   VARCHAR(20) PATH 'lax $.name.last',
                         NESTED PATH 'lax $.phones[*]'
                           COLUMNS (
                                    "phone type"  VARCHAR(20) PATH 'lax $.type',
                                    "number"  VARCHAR(20) )
                                        )
                       ) AS U
```

これは、以下のものを戻します。

```
id    first name    last name    phone type    phone number
---   ----------    ---------    ----------    ------------
901   John          Doe          home          555-3762
901   John          Doe          work          555-8792
```

MQREADALL

MQREADALL 関数は、キューからメッセージを除去せずに、VARCHAR 列で指定された MQSeries ロケーションからのメッセージおよびメッセージ・メタデータが入った表を戻します。



注:

1 先行する引数が関数に指定されている場合は、*num-rows* の前にコンマが必要です。

MQREADALL 関数は、*service-policy* で定義されたサービス品質ポリシーを使って、*receive-service* で指定された MQSeries ロケーションからのメッセージおよびメッセージ・メタデータが入った表を戻します。この処理を行っても、*receive-service* に関連付けられたキューからメッセージは除去されません。

receive-service
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の値は、空ストリング、または末尾ブランクを含むストリングであってはなりません。式の実際の長さは、48 バイトを超えてはなりません。この式の値は、SYSIBM.MQSERVICE 表に定義されたサービス・ポイントを参照している必要があります。サービス・ポイントは、メッセージの送信元または受信相手側の論理エンドポイントです。サービス・ポイントの定義には、MQSeries キュー・マネージャーの名前およびキューの名前が含まれます。 MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

receive-service が指定されていないか NULL 値である場合、Db2.DEFAULT.SERVICE が使用されます。

service-policy
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の値は、空ストリング、または末尾ブランクを含むストリングであってはなりません。式の実際の長さは、48 バイトを超えてはなりません。この式の値は、SYSIBM.MQPOLICY 表に定義されたサービス・ポリシーを参照している必要があります。サービス・ポリシーは、このメッセージ処理に適用されるサービス品質オプションのセットを指定します。これらのオプションには、メッセージ優先順位やメッセージ持続性などがあります。MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

service-policy が指定されていないか NULL 値である場合、Db2.DEFAULT.POLICY が使用されます。

num-rows
その値が正整数かゼロである SMALLINT または INTEGER データ・タイプの値を戻す式。式の値は、戻すメッセージの最大数を指定します。

num-rows が指定されないか、または式の値がゼロの場合、使用可能なすべてのメッセージが戻されます。

この関数の結果は、以下の表に示された形式の表です。列はすべてNULL 可能です。

表 *63. MQREADALL* の結果表の形式

列名	データ・タイプ	内容
MSG	VARCHAR(32000)	MQSeries メッセージの内容
CORRELID	VARCHAR(24)	メッセージを関連付けるために使用される相関 ID
TOPIC	VARCHAR(40)	予約済み
QNAME	VARCHAR(48)	メッセージ受信相手先のキューの名前
MSGID	VARCHAR(24)	このメッセージの、MQSeries 割り当ての固有な ID
MSGFORMAT	VARCHAR(8)	MQSeries で定義されたメッセージの形式

結果列の CCSID は、現行サーバーにおけるデフォルト CCSID です。

注

前提条件**:** MQSeries の機能を使用するためには、IBM MQSeries for IBM i がインストールおよび構成され、作動可能である必要があります。

例

- この例は、デフォルト・サービス (Db2.DEFAULT.SERVICE) によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューからすべてのメッセージを取り出します。これらのメッセージとすべてのメタデータが表として戻されます。

```
SELECT *
 FROM TABLE (MQREADALL ()) AS T
```

- この例は、サービス MYSERVICE によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューの先頭からすべてのメッセージを取り出します。MSG 列と CORRELID 列のみが戻されます。

```
SELECT T.MSG, T.CORRELID
 FROM TABLE (MQREADALL ('MYSERVICE')) AS T
```

- この例は、デフォルト・サービス (Db2.DEFAULT.SERVICE) によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューの先頭を読み取ります。 CORRELID が '1234' のメッセージだけが戻されます。列はすべて戻されます。

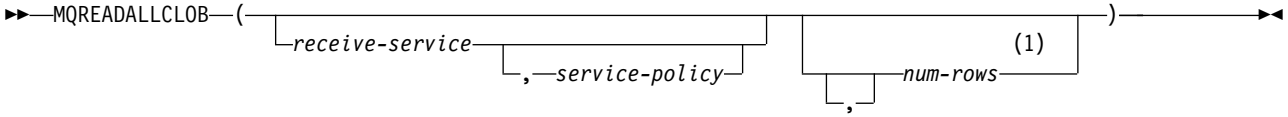
```
SELECT *
 FROM TABLE (MQREADALL ()) AS T
 WHERE T.CORRELID = '1234'
```

- この例は、デフォルト・サービス (Db2.DEFAULT.SERVICE) によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューの先頭から最初の 10 個のメッセージを取り出します。列はすべて戻されます。

```
SELECT *
 FROM TABLE (MQREADALL (10)) AS T
```

MQREADALLCLOB

MQREADALLCLOB 関数は、キューからメッセージを除去せずに、CLOB 列で指定された MQSeries ロケーションからのメッセージおよびメッセージ・メタデータが入った表を戻します。



注:

1 先行する引数が関数に指定されている場合は、*num-rows* の前にコンマが必要です。

MQREADALLCLOB 関数は、*service-policy* で定義されたサービス品質ポリシーを使って、*receive-service* で指定された MQSeries ロケーションからのメッセージおよびメッセージ・メタデータが入った表を戻します。この処理を行っても、*receive-service* に関連付けられたキューからメッセージは除去されません。

receive-service
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の値は、空ストリング、または末尾ブランクを含むストリングであってはなりません。式の実際の長さは、48 バイトを超えてはなりません。この式の値は、SYSIBM.MQSERVICE 表に定義されたサービス・ポイントを参照している必要があります。サービス・ポイントは、メッセージの送信元または受信相手側の論理エンドポイントです。サービス・ポイントの定義には、MQSeries キュー・マネージャーの名前およびキューの名前が含まれます。 MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

receive-service が指定されていないか NULL 値である場合、Db2.DEFAULT.SERVICE が使用されます。

service-policy
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の値は、空ストリング、または末尾ブランクを含むストリングであってはなりません。式の実際の長さは、48 バイトを超えてはなりません。この式の値は、SYSIBM.MQPOLICY 表に定義されたサービス・ポリシーを参照している必要があります。サービス・ポリシーは、このメッセージ処理に適用されるサービス品質オプションのセットを指定します。これらのオプションには、メッセージ優先順位やメッセージ持続性などがあります。MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

service-policy が指定されていないか NULL 値である場合、Db2.DEFAULT.POLICY が使用されます。

num-rows
その値が正整数かゼロである SMALLINT または INTEGER データ・タイプの値を戻す式。式の値は、戻すメッセージの最大数を指定します。

num-rows が指定されないか、または式の値がゼロの場合、使用可能なすべてのメッセージが戻されます。

この関数の結果は、以下の表に示された形式の表です。列はすべてNULL 可能です。

表 *64. MQREADALLCLOB* の結果表の形式

列名	データ・タイプ	内容
MSG	CLOB(2M)	MQSeries メッセージの内容
CORRELID	VARCHAR(24)	メッセージを関連付けるために使用される相関 ID
TOPIC	VARCHAR(40)	予約済み
QNAME	VARCHAR(48)	メッセージ受信相手先のキューの名前
MSGID	VARCHAR(24)	このメッセージの、MQSeries 割り当ての固有な ID
MSGFORMAT	VARCHAR(8)	MQSeries で定義されたメッセージの形式

結果列の CCSID は、現行サーバーにおけるデフォルト CCSID です。

注

前提条件**:** MQSeries の機能を使用するためには、IBM MQSeries for IBM i がインストールおよび構成され、作動可能である必要があります。

例

- この例は、デフォルト・サービス (Db2.DEFAULT.SERVICE) によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューからすべてのメッセージを取り出します。これらのメッセージとすべてのメタデータが表として戻されます。

```
SELECT *
 FROM TABLE (MQREADALLCLOB ()) AS T
```

- この例は、サービス MYSERVICE によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューの先頭からすべてのメッセージを取り出します。MSG 列と CORRELID 列のみが戻されます。

```
SELECT T.MSG, T.CORRELID
 FROM TABLE (MQREADALLCLOB ('MYSERVICE')) AS T
```

- この例は、デフォルト・サービス (Db2.DEFAULT.SERVICE) によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューの先頭を読み取ります。CORRELID が '1234' のメッセージだけが戻されます。列はすべて戻されます。

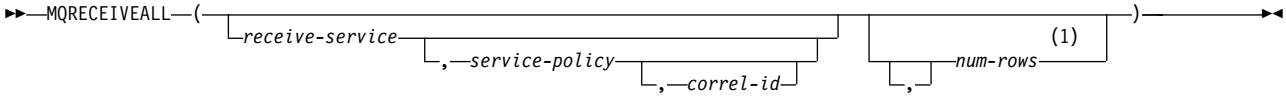
```
SELECT *
 FROM TABLE (MQREADALLCLOB ()) AS T
 WHERE T.CORRELID = '1234'
```

- この例は、デフォルト・サービス (Db2.DEFAULT.SERVICE) によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューの先頭から最初の 10 個のメッセージを取り出します。列はすべて戻されます。

```
SELECT *
 FROM TABLE (MQREADALLCLOB (10)) AS T
```

MQRECEIVEALL

MQRECEIVEALL 関数は、VARCHAR 列で指定された MQSeries ロケーションからのメッセージおよびメッセージ・メタデータが入った表を戻し、キューからメッセージを除去します。



注:

1 先行する引数が関数に指定されている場合は、*num-rows* の前にコンマが必要です。

MQRECEIVEALL 関数は、*service-policy* で定義されたサービス品質ポリシーを使って、*receive-service* で指定された MQSeries ロケーションからのメッセージおよびメッセージ・メタデータが入った表を戻します。この処理を行うと、*receive-service* に関連付けられたキューからメッセージは除去されます。

receive-service
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の値は、空ストリング、または末尾ブランクを含むストリングであってはなりません。式の実際の長さは、48 バイトを超えてはなりません。この式の値は、SYSIBM.MQSERVICE 表に定義されたサービス・ポイントを参照している必要があります。サービス・ポイントは、メッセージの送信元または受信相手側の論理エンドポイントです。サービス・ポイントの定義には、MQSeries キュー・マネージャーの名前およびキューの名前が含まれます。 MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

receive-service が指定されていないか NULL 値である場合、Db2.DEFAULT.SERVICE が使用されます。

service-policy
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の値は、空ストリング、または末尾ブランクを含むストリングであってはなりません。式の実際の長さは、48 バイトを超えてはなりません。この式の値は、SYSIBM.MQPOLICY 表に定義されたサービス・ポリシーを参照している必要があります。サービス・ポリシーは、このメッセージ処理に適用されるサービス品質オプションのセットを指定します。これらのオプションには、メッセージ優先順位やメッセージ持続性などがあります。MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

service-policy が指定されていないか NULL 値である場合、Db2.DEFAULT.POLICY が使用されます。

correl-id
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の実際の長さは、24 バイトを超えてはなりません。式の値は、このメッセージに関連付けられている相関 ID を指定します。相関 ID は、しばしば、要求を応答に関連付けるために「要求

と応答」シナリオで指定されます。相関 ID が一致した最初のメッセージが戻されます。 MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

末尾ブランクを含む固定長ストリングは、有効な値として扱われます。ただし、*correl-id* を MQSEND など他の要求で指定する場合、一致していると認識されるためには、同じ *correl-id* を指定する必要があります。例えば、前の MQSEND 要求時に「test」(末尾ブランクがある) という *correl-id* 値を指定し、MQRECEIVEALL の *correl-id* に「test」という値を指定しても一致しません。

correl-id が、指定されていない、空ストリングである、または NULL 値である場合、相関 ID は使用されず、キューの先頭のメッセージが戻されます。

num-rows

その値が正整数かゼロである SMALLINT または INTEGER データ・タイプの値を戻す式。式の値は、戻すメッセージの最大数を指定します。

num-rows が指定されないか、または式の値がゼロの場合、使用可能なすべてのメッセージが戻されます。

この関数の結果は、以下の表に示された形式の表です。列はすべてNULL 可能です。

表 *65. MQRECEIVEALL* の結果表の形式

列名	データ・タイプ	内容
MSG	VARCHAR(32000)	MQSeries メッセージの内容
CORRELID	VARCHAR(24)	メッセージを関連付けるために使用される相関 ID
TOPIC	VARCHAR(40)	予約済み
QNAME	VARCHAR(48)	メッセージ受信相手先のキューの名前
MSGID	VARCHAR(24)	このメッセージの、MQSeries 割り当ての固有な ID
MSGFORMAT	VARCHAR(8)	MQSeries で定義されたメッセージの形式

結果列の CCSID は、現行サーバーにおけるデフォルト CCSID です。

注

前提条件**:** MQSeries の機能を使用するためには、IBM MQSeries for IBM i がインストールおよび構成され、作動可能である必要があります。

例

- この例は、デフォルト・サービス (Db2.DEFAULT.SERVICE) によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューからすべてのメッセージを取り出します。これらのメッセージとすべてのメタデータが表として戻されます。

```
SELECT *
 FROM TABLE (MQRECEIVEALL ()) AS T
```

- この例は、サービス MYSERVICE によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューの先頭からすべてのメッセージを取り出します。MSG 列と CORRELID 列のみが戻されます。

```
SELECT T.MSG, T.CORRELID
 FROM TABLE (MQRECEIVEALL ('MYSERVICE')) AS T
```

- この例は、サービス「MYSERVICE」によって指定された、ポリシー「MYPOLICY」を使用するキューの先頭からすべてのメッセージを取り出します。CORRELID が '1234' のメッセージだけが戻されます。MSG 列と CORRELID 列のみが戻されます。

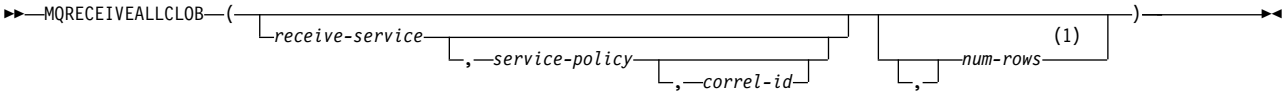
```
SELECT *
 FROM TABLE (MQRECEIVEALL ('MYSERVICE','MYPOLICY','1234')) AS T
```

- この例は、デフォルト・サービス (Db2.DEFAULT.SERVICE) によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューの先頭から最初の 10 個のメッセージを取り出します。列はすべて戻されます。

```
SELECT *
 FROM TABLE (MQRECEIVEALL (10)) AS T
```

MQRECEIVEALLCLOB

MQRECEIVEALLCLOB 関数は、CLOB 列で指定された MQSeries ロケーションからのメッセージおよびメッセージ・メタデータが入った表を戻し、キューからメッセージを除去します。



注:

1 先行する引数が関数に指定されている場合は、*num-rows* の前にコンマが必要です。

MQRECEIVEALLCLOB 関数は、*service-policy* で定義されたサービス品質ポリシーを使って、*receive-service* で指定された MQSeries ロケーションからのメッセージおよびメッセージ・メタデータが入った表を戻します。この処理を行うと、*receive-service* に関連付けられたキューからメッセージは除去されます。

receive-service
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の値は、空ストリング、または末尾ブランクを含むストリングであってはなりません。式の実際の長さは、48 バイトを超えてはなりません。この式の値は、SYSIBM.MQSERVICE 表に定義されたサービス・ポイントを参照している必要があります。サービス・ポイントは、メッセージの送信元または受信相手側の論理エンドポイントです。サービス・ポイントの定義には、MQSeries キュー・マネージャーの名前およびキューの名前が含まれます。 MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

receive-service が指定されていないか NULL 値である場合、Db2.DEFAULT.SERVICE が使用されます。

service-policy
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の値は、空ストリング、または末尾ブランクを含むストリングであってはなりません。式の実際の長さは、48 バイトを超えてはなりません。この式の値は、SYSIBM.MQPOLICY 表に定義されたサービス・ポリシーを参照している必要があります。サービス・ポリシーは、このメッセージ処理に適用されるサービス品質オプションのセットを指定します。これらのオプションには、メッセージ優先順位やメッセージ持続性などがあります。MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

service-policy が指定されていないか NULL 値である場合、Db2.DEFAULT.POLICY が使用されます。

correl-id
LOB 以外の組み込みの文字ストリング・データ・タイプまたはグラフィック・ストリング・データ・タイプの値を戻す式。式の実際の長さは、24 バイトを超えてはなりません。式の値は、このメッセージに関連付けられている相関 ID を指定します。相関 ID は、しばしば、要求を応答に関連付けるために「要求

と応答」シナリオで指定されます。相関 ID が一致した最初のメッセージが戻されます。 MQSeries Application Messaging について詳しくは、「SQL プログラミング」を参照してください。

末尾ブランクを含む固定長ストリングは、有効な値として扱われます。ただし、*correl-id* を MQSEND など他の要求で指定する場合、一致していると認識されるためには、同じ *correl-id* を指定する必要があります。例えば、前の MQSEND 要求時に「test」(末尾ブランクがある) という *correl-id* 値を指定し、MQRECEIVEALLCLOB の *correl-id* に「test」という値を指定しても一致しません。

correl-id が、指定されていない、空ストリングである、または NULL 値である場合、相関 ID は使用されず、キューの先頭のメッセージが戻されます。

num-rows
その値が正整数かゼロである SMALLINT または INTEGER データ・タイプの値を戻す式。式の値は、戻すメッセージの最大数を指定します。

num-rows が指定されないか、または式の値がゼロの場合、使用可能なすべてのメッセージが戻されます。

この関数の結果は、以下の表に示された形式の表です。列はすべてNULL 可能です。

表 *66. MQRECEIVEALLCLOB* の結果表の形式

列名	データ・タイプ	内容
MSG	CLOB(2M)	MQSeries メッセージの内容
CORRELID	VARCHAR(24)	メッセージを関連付けるために使用される相関 ID
TOPIC	VARCHAR(40)	予約済み
QNAME	VARCHAR(48)	メッセージ受信相手先のキューの名前
MSGID	VARCHAR(24)	このメッセージの、MQSeries 割り当ての固有な ID
MSGFORMAT	VARCHAR(8)	MQSeries で定義されたメッセージの形式

結果列の CCSID は、現行サーバーにおけるデフォルト CCSID です。

注

前提条件: MQSeries の機能を使用するためには、IBM MQSeries for IBM i がインストールおよび構成され、作動可能である必要があります。

例

- この例は、デフォルト・サービス (Db2.DEFAULT.SERVICE) によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューからすべてのメッセージを取り出します。これらのメッセージとすべてのメタデータが表として戻されます。

```
SELECT *
 FROM TABLE (MQRECEIVEALLCLOB ()) AS T
```

- この例は、サービス MYSERVICE によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューの先頭からすべてのメッセージを取り出します。MSG 列と CORRELID 列のみが戻されます。

```
SELECT T.MSG, T.CORRELID
 FROM TABLE (MQRECEIVEALLCLOB ('MYSERVICE')) AS T
```

- この例は、サービス「MYSERVICE」によって指定された、ポリシー「MYPOLICY」を使用するキューの先頭からすべてのメッセージを取り出します。CORRELID が '1234' のメッセージだけが戻されます。MSG 列と CORRELID 列のみが戻されます。

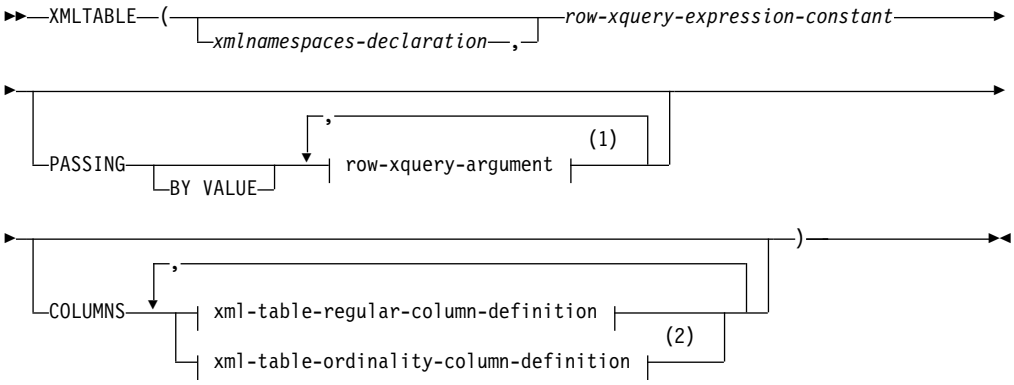
```
SELECT *
 FROM TABLE (MQRECEIVEALLCLOB ('MYSERVICE','MYPOLICY','1234')) AS T
```

- この例は、デフォルト・サービス (Db2.DEFAULT.SERVICE) によって指定された、デフォルト・ポリシー (Db2.DEFAULT.POLICY) を使用するキューの先頭から最初の 10 個のメッセージを取り出します。列はすべて戻されます。

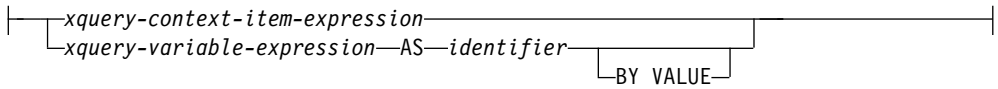
```
SELECT *
 FROM TABLE (MQRECEIVEALLCLOB (10)) AS T
```

XMLTABLE

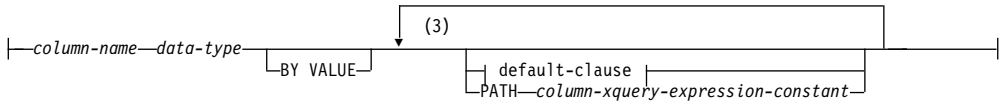
XMLTABLE 関数は、 XPath 式の評価から結果表を戻します。その際、場合によっては、指定された入力引数を XPath 変数として使用します。行 XPath 式の結果シーケンスに含まれる各項目が、結果表の 1 つの行を表します。



row-xquery-argument:



xml-table-regular-column-definition:



xml-table-ordinality-column-definition:

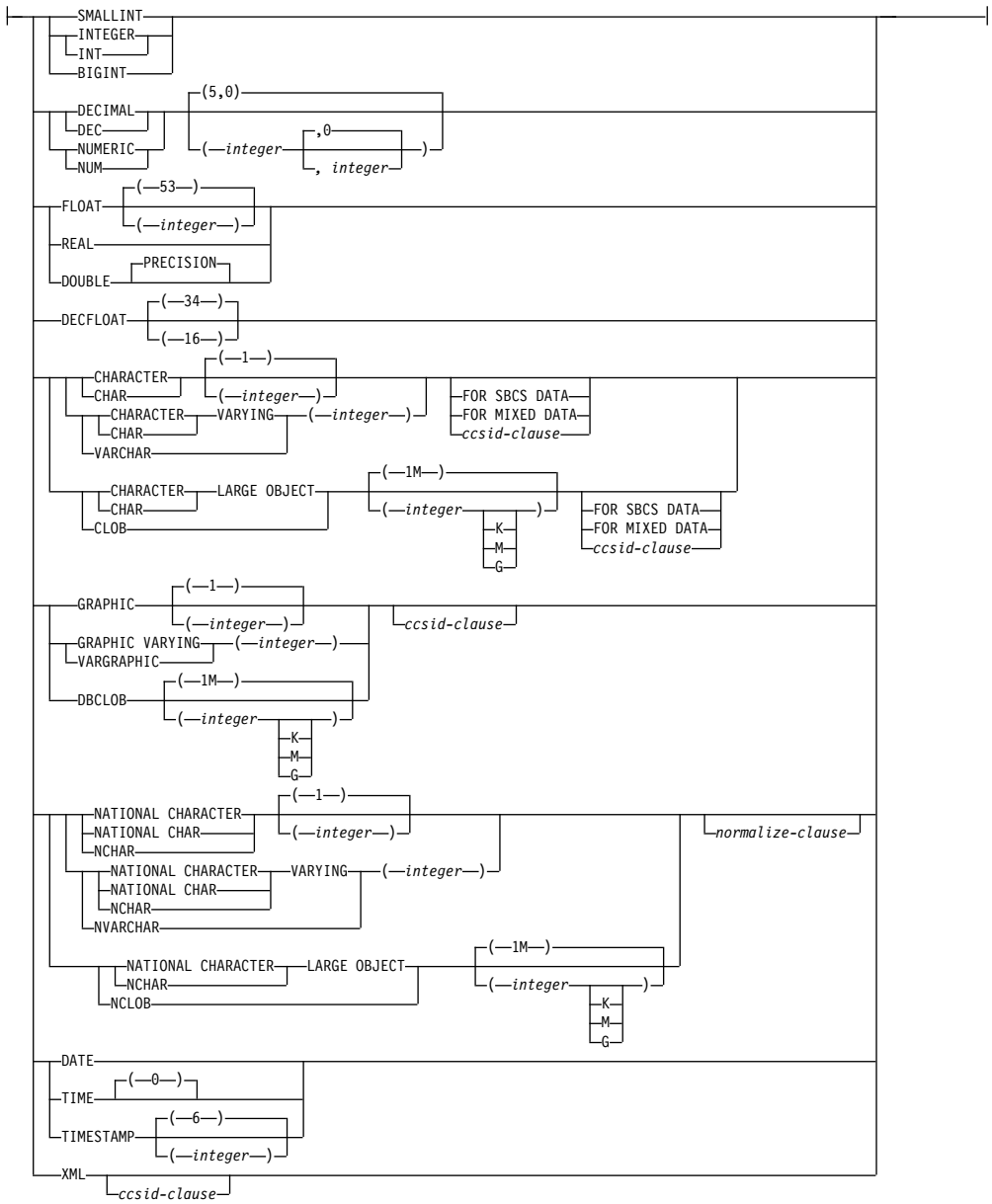
注:

1 *xquery-context-item-expression* は、複数回指定してはなりません。

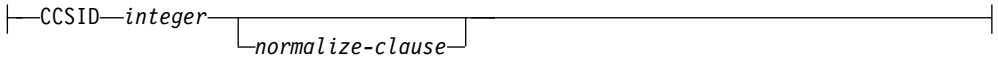
2 *xml-table-ordinality-column-definition* 文節は、複数回指定してはなりません。

3 *default-clause* と PATH 節はどちらも複数回指定することはできません。

data-type:

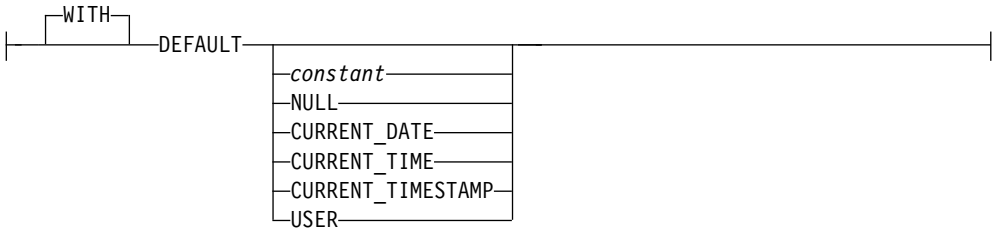


ccsid-clause:



normalize-clause:

default-clause:



関数名は修飾名として指定することはできません。

xmlnamespaces-declaration
XMLNAMESPACES 関数を使用して、1 つ以上の XML ネーム・スペース宣言を指定します。これは *row-xquery-expression-constant* および *column-xquery-expression-constant* の静的コンテキストの一部になります。XMLTABLE の引数である XPath 式の静的な既知のネーム・スペースの集合は、事前設定された静的な既知のネーム・スペースの集合と、この文節で指定されたネーム・スペース宣言を組み合わせたものです。XPath 式内の XPath プロローグはこれらのネーム・スペースをオーバーライドすることができます。

xmlnamespaces-declaration を指定しない場合は、事前設定された静的な既知のネーム・スペースの集合のみが XPath 式に適用されます。

row-xquery-expression-constant
サポートされる XPath 言語構文を使用して、XPath 式として解釈される SQL ストリング定数を指定します。この式は、結果表の行数を決定します。*row-xquery-argument* で指定されたオプションの 1 組の入力 XML 値を使用して式を評価し、出力 XPath シーケンス (シーケンス内の各項目ごとに 1 行を生成) を戻します。シーケンスが空の場合、XMLTABLE の結果は空の表です。*row-xquery-expression-constant* に、空ストリングまたはすべてがブランクのストリングを指定してはいけません。

PASSING
入力値、およびそれらの値を *row-xquery-expression-constant* で指定された XPath 式に渡す方法を指定します。

BY VALUE
すべての XML 引数が値によって受け渡されることを指定します。XML 値が値によって受け渡される場合、XPath 評価は XML データのコピーを使用します。これがデフォルトの動作です。 Db2 for i は、XPath 変数式を、XML 入力値を表す文書ノードにバインドします。

この節は、非 XML 値の受け渡し方法には影響しません。非 XML 値は、常に、XML へのキャスト中に値のコピーを作成します。

row-xquery-argument
row-xquery-expression-constant により指定された XPath 式に渡される引数を指定します。 *row-xquery-argument* は、XPath 式に渡される前に評価される SQL 式を指定します。

row-xquery-argument のデータ・タイプが XML でない場合、式の結果は XML に変換されます。*xquery-variable-expression* の場合、 NULL 値は XML 空シーケンスに変換されます。

row-xquery-argument を XPath 式で使用する方法は、引数が *xquery-context-item-expression* または *xquery-variable-expression* のどちらとして指定されているかによって異なります。

row-xquery-argument には、 NEXT VALUE 式または PREVIOUS VALUE 式、あるいは OLAP 仕様を含めてはなりません。

xquery-context-item-expression
XML であるか、または XML への変換がサポートされているタイプの値を戻す、SQL 式を指定します。

xquery-context-item-expression は *row-xquery-expression* の初期コンテキスト項目を指定します。初期コンテキスト項目の値は、XML への変換後の *xquery-context-item-expression* の結果です。 *xquery-context-item-expression* は、複数回指定してはなりません。

xquery-variable-expression
実行中に *row-xquery-expression-constant* により指定された XPath 式が使用できる値を持つ SQL 式を指定します。この式は、XML であるか、または XML への変換がサポートされているタイプの値を戻す必要があります。

xquery-variable-expression は、*row-xquery-expression-constant* に XPath 変数として渡される引数を指定します。*xquery-variable-expression* が NULL 値である場合、XPath 変数は XML 空シーケンスに設定されます。PASSING 文節から作成された XPath 変数の有効範囲は、*row-xquery-expression-constant* で指定される XPath 式です。

AS *identifier*
xquery-variable-expression により生成された値が、*row-xquery-expression-constant* に XPath 変数として渡されることを指定します。 *identifier* が示す名前は、XML NCName の形式でなければなりません。有効な名前についての詳細は、W3C XML ネーム・スペースの指定の項目を参照してください。XPath 言語で変数名の前に置かれる先行ドル記号 ($) を *identifier* の一部として含めてはなりません。この ID の長さは 128 バイトを超えてはなりません。同じ PASSING 文節内の 2 つの引数が同じ ID を使用することはできません。

BY VALUE
xquery-variable-expression が値によって受け渡されることを指定します。XML 値が値によって受け渡される場合、XPath 評価は XML データのコピーを使用します。Db2 for i は、XPath 変数式を、XML 入力値を表す文書ノードにバインドします。BY VALUE が *xquery-variable-expression* に続いて指定されない場合、XML 引数は、PASSING キーワードに続く構文により提供されるデフォルトの受け渡しメカニズムを使用して渡されます。

この節は、入力値が XML データ・タイプの場合のみ有効です。非 XML 値は、常に、XML へのキャスト中に値のコピーを作成します。

表 *67. SQL* から *XML* へのサポートされる変換

SQL タイプ	**XML** タイプ	注
SMALLINT	xs:integer	

表 67. SQL から XML へのサポートされる変換 (続き)

SQL タイプ	XML タイプ	注
INTEGER	xs:integer	
BIGINT	xs:integer	
DECIMAL NUMERIC	xs:decimal	精度が 34 より大きい 10 進数の場合、処理中に精度が失われる可能性があります。
FLOAT DOUBLE DECFLOAT	xs:double	
CHAR VARCHAR CLOB GRAPHIC VARGRAPHIC DBCLOB	xs:string	文字ストリング値を XML 値にキャストする場合、結果の xs:string アトミック値は正しくない XML 文字を含むことができません。入力文字ストリングが Unicode でない場合、入力文字は Unicode に変換されます。 CHAR ストリングおよび VARCHAR ストリングでは、CCSID を 65535 にすることはできず、ビット・データ用に指定することもできません。
DATE	xs:date	xs:date 値にはタイム・ゾーン・コンポーネントは含まれません。比較のため、タイム・ゾーンは暗黙的に UTC であると想定されます。 必要な場合、fn:adjust-timezone() 関数を使用してタイム・ゾーンを明示的に設定できます。
TIME	xs:time	xs:time 値にはタイム・ゾーン・コンポーネントは含まれません。比較のため、タイム・ゾーンは暗黙的に UTC であると想定されます。 必要な場合、fn:adjust-timezone() 関数を使用してタイム・ゾーンを明示的に設定できます。
TIMESTAMP	xs:dateTime	xs:dateTime 値にはタイム・ゾーン・コンポーネントは含まれません。比較のため、タイム・ゾーンは暗黙的に UTC であると想定されます。 必要な場合、fn:adjust-timezone() 関数を使用してタイム・ゾーンを明示的に設定できます。

COLUMNS

列名、データ・タイプ、および各行の列値の計算方法を組み込んで、結果表の出力列を指定します。この節を指定しない場合は、タイプが XML の単一の無名列が戻され、その値は *row-xquery-expression* 内の XPath 式の評価から得られるシーケンス項目に基づきます (`PATH '.'` を指定することと同等です)。この結果列を参照するには、表関数の後に続く *correlation-clause* で *column-name* を指定する必要があります。

すべての結果列の長さの合計は 64K バイトを超えてはなりません。データ・タイプごとの列のバイト・カウントについては、 1295 ページの『最大行サイズ』を参照してください。*row-xquery-arguments* の数が *N* であるとすると、列の数は 8000-*N* 以下でなければなりません。

xml-table-regular-column-definition
列名、データ・タイプ、およびその行のシーケンス項目から値を抽出するための XPath 式を組み込んで、結果表の 1 つの出力列を指定します。

column-name
結果表の列の名前を指定します。名前を修飾することはできず、結果表の複数の列に対して同じ名前を使用することはできません。

data-type
列のデータ・タイプを指定します。CHAR 列および VARCHAR 列の場合、CCSID を 65535 にすることはできません。

BY VALUE
結果列が値によって戻されることを指定します。XML 値が値によって戻される場合、XML データのコピーが戻されます。これがデフォルトの動作です。 Db2 for i は、値が表関数から戻されると、XML 結果の文書ノードを作成します。この節は、データ・タイプが XML ではない列に対して指定してはなりません。

default-clause
列のデフォルト値を指定します。XMLTABLE 結果列の場合、*column-xquery-expression-constant* に含まれる XPath 式の処理が空のシーケンスを戻す場合は、デフォルトが適用されます。

PATH *column-xquery-expression-constant*
サポートされる XPath 言語構文を使用して、XPath 式として解釈されるストリング定数を指定します。 *column-xquery-expression-constant* は XPath 式を指定しますが、これは *row-xquery-expression-constant* 内の XPath 式の評価の結果である項目に関連して列値を決定します。*row-query-expression-constant* の処理の結果得られた項目が外部提供のコンテキスト項目として入力されると、*column-xquery-expression-constant* が評価されて、出力シーケンスが戻されます。この出力シーケンスに基づいて、列値は以下のように決定されます。

- 空シーケンスが戻される場合、*default-clause* が列の値を提供します。
- 空のシーケンスが戻され、*default-clause* が指定されていない場合、列には NULL 値が割り当てられます。
- 空でないシーケンスが戻される場合、値は列に対して指定されたデータ・タイプに変換されます。この暗黙的な変換処理からエラーが戻される可能性があります。

column-xquery-expression-constant の値は、空ストリングまたはすべてがブランクのストリングであってはなりません。この節が指定されない場合、デフォルトの XPath 式は *column-name* になります。

xml-table-ordinality-column-definition
結果表の順序を示す列を指定します。

colum-name
結果表の列の名前を指定します。名前を修飾することはできず、結果表の複数の列に対して同じ名前を使用することはできません。

FOR ORDINALITY

column-name は結果表の順序を示す列であることを指定します。この列のデータ・タイプは BIGINT です。結果表のこの列の値は、*row-xquery-expression-constant* 内の XPath 式を評価した結果シーケンスにおける行の項目の順序番号です。

表 *68. XML* から *SQL* 結果列へのサポートされる変換

XML タイプ	**SQL** タイプ	注
xs:integer	SMALLINT INTEGER BIGINT	
xs:decimal	DECIMAL NUMERIC	結果の xs:decimal 値は、必要に応じて、ターゲット・データ・タイプの精度と位取りに変換されます。必要な数の先行ゼロが追加または除去されます。数字の小数部分では、必要な数の後続ゼロが追加されるか、必要な桁数が除去されます。この切り捨て動作は、DECIMAL から DECIMAL へのキャストの動作と同様です。精度が 34 より大きい 10 進数の場合、処理中に精度が失われる可能性があります。
xs:double	FLOAT DOUBLE REAL DECFLOAT	ターゲット・タイプが FLOAT、DOUBLE、または REAL で、 XPath キャストの後のソース XML 値が、INF、-INF、または Nan の xs:double 値である場合、エラーが戻されます。ソース値が xs:double の負のゼロの場合、値は正のゼロに変換されます。ソース値がターゲット・データ・タイプの範囲を超える場合、オーバーフロー・エラーが戻されます。ソース値にターゲット・データ・タイプの精度を超える有効数字が含まれている場合、ソース値はターゲット・データ・タイプの精度に丸められます。 ターゲット・タイプが DECFLOAT で、ソース XML 値が INF、-INF、または NaN の xs:double 値である場合、結果は対応する特殊 DECFLOAT 値 INF、-INF、または NaN になります。ソース値が xs:double の負のゼロの場合、結果は負のゼロです。ターゲット・タイプが DECFLOAT(16) で、ソース値が DECFLOAT(16) の範囲を超える場合、オーバーフロー・エラーが戻されます。ソース値の有効数字が 16 桁を超える場合、値は有効な ROUNDING モードに従って丸められます。丸めの動作は、DECFLOAT(34) から DECFLOAT(16) へのキャスト時に使用されるものと同じです。

表 68. XML から SQL 結果列へのサポートされる変換 (続き)

XML タイプ	SQL タイプ	注
xs:string	CHAR VARCHAR CLOB GRAPHIC VARGRAPHIC DBCLOB	結果の XML 値は、必要に応じて、限定された長さのターゲット・タイプに変換される前に、120 ページの『割り当ての際の変換規則』に説明されている規則を使用してターゲット・データ・タイプの CCSID に変換されます。指定された長さの制限が CCSID 変換後のストリングの長さより短い場合、切り捨てが行われます。非ブランク文字が切り捨てられる場合は、警告が出されます。ターゲット・タイプが固定長ストリング・タイプ (CHAR または GRAPHIC) であり、ターゲット・タイプの指定の長さが CCSID 変換後のストリングの長さより長い場合、末尾にブランクが埋め込まれます。この切り捨ておよび埋め込みの動作は、文字ストリングまたはグラフィック・ストリングの取り出し割り当てと同様です。
xs:date	DATE	結果の XML 値は UTC 時間に調整され、タイム・ゾーン・コンポーネントは除去されます。結果の xs:date 値の年の部分は 0001 から 9999 の範囲内でなければなりません。
xs:time	TIME	結果の XML 値は UTC 時間に調整され、タイム・ゾーン・コンポーネントは除去されます。小数秒は結果から切り捨てられます。
xs:dateTime	TIMESTAMP	結果の XML 値は UTC 時間に調整され、タイム・ゾーン・コンポーネントは除去されます。結果の xs:dateTime 値の年の部分は 0001 から 9999 までの範囲内でなければなりません。ターゲットのタイム・スタンプ・タイプの精度が 12 より小さい場合、xs:dateTime 値の小数秒の部分は、ターゲットのタイム・スタンプの精度までに切り捨てられます。

この関数の結果は表です。いずれかの XPath 式の評価結果がエラーになる場合、XMLTABLE 関数は XPath エラーを戻します。

例

- 以下は、注文の購入注文項目で状況が「Unshipped」の結果である表のリストです。

```
SELECT U."PO ID", U."Part #", U."Product Name",
       U."Quantity", U."Price", U."Order Date"
  FROM PURCHASEORDER P,
       XMLTABLE('$po/PurchaseOrder/itemlist/item' PASSING P.PORDER AS "po"
                COLUMNS "PO ID"         INTEGER       PATH '../@PoNum',
                        "Part #"        CHAR(10)      PATH 'partid',
                        "Product Name"  VARCHAR(50)   PATH 'name',
                        "Quantity"      INTEGER       PATH 'quantity',
                        "Price"         DECIMAL(9,2)  PATH 'price',
                        "Order Date"    DATE          PATH '../@OrderDate'
                ) AS U
  WHERE P.STATUS = 'Unshipped'
```

第 5 章 プロシージャー

この章では、システム提供プロシージャーについて、構文図、意味の説明、規則、および使用例を示します。

CREATE_WRAPPED

CREATE_WRAPPED プロシージャーは、判読可能な DDL ステートメントを難読化された DDL ステートメントに変換し、そのオブジェクトをデータベースに配置します。

難読化された DDL ステートメント内の手順ロジックおよび組み込み SQL ステートメントは、ロジックに含まれる知的財産を簡単には抽出できないように、暗号化されます。

スキーマは SYSIBMADM です。

object-definition-string
DDL ステートメントを含んでいる、タイプ CLOB のストリング。以下のいずれかの SQL ステートメントを指定できます。

- CREATE FUNCTION (SQL スカラー)
- CREATE FUNCTION (SQL 表)
- CREATE PROCEDURE (SQL)
- CREATE TRIGGER

このプロシージャーは、入力を難読化された DDL ステートメント・ストリングに変換した後、その DDL ステートメントを動的に実行します。エンコードは、元のステートメントのままにしたルーチン・シグニチャーまたはトリガー名までの接頭部と、その後に続くキーワード WRAPPED から構成されます。このキーワードの後には、関数を起動したアプリケーション・サーバーについての情報が続きます。この情報の形式は *pppvvrrm* で、各部の意味は次のとおりです。

- *ppp* は、以下の 3 文字を使用して製品を示します。
  - DSN (Db2 for z/OS の場合)
  - QSQ (Db2 for i の場合)
  - SQL (Db2 for LUW の場合)
- *vv* は、2 桁のバージョン ID です (例えば、'07' など)。
- *rr* は、2 桁のリリース ID です (例えば、'02' など)。
- *m* は、1 文字の修正レベル ID です (例えば、'0' など)。

例えば、Db2 for i バージョン 7.3 は、「QSQ07030」で示されます。

このアプリケーション・サーバー情報の後に、文字 (a から z および A から Z)、数字 (0 から 9)、下線、およびコロンからなるストリングが続きます。

エンコードされた DDL ステートメントは、平文形式のステートメントよりも最大 3 分の 1 だけ長くなる可能性があります。結果が SQL ステートメントの最大長を超える場合、エラーが発行されます。

注記

ステートメントのエンコードは、内容を難読化することを意図したものであり、強い暗号化の一種と考えるべきではありません。

例

例 1: 週 40 時間労働として時間給から年間給与を計算する関数の難読化版を生成します。

```
CALL CREATE_WRAPPED('CREATE FUNCTION salary(wage DECFLOAT)
                           RETURNS DECFLOAT RETURN wage * 40 * 52');

SELECT ROUTINE_DEFINITION FROM QSYS2.SYSROUTINES
    WHERE routine_name ='SALARY' AND routine_schema = CURRENT SCHEMA;
```

この CALL ステートメントが正常に完了すると、ルーチン「SALARY」に対応する行の、QSYS2.SYSROUTINES 内の ROUTINE_DEFINITION 列は、次のようなものになります。

```
WRAPPED QSQ07020 <encoded-suffix>
```

例 2: 複雑なデフォルトを設定するトリガーの難読化版を生成します。

```
CALL CREATE_WRAPPED('CREATE OR REPLACE TRIGGER trig1 BEFORE INSERT ON emp
                      REFERENCING NEW AS n FOR EACH ROW
                      WHEN (n.bonus IS NULL) SET n.bonus = n.salary * .04');

SELECT ACTION_STATEMENT FROM QSYS2.SYSTRIGGERS
    WHERE trigname ='TRIG1' AND trigschema = CURRENT SCHEMA;
```

この CALL ステートメントが正常に完了すると、トリガー「TRIG1」に対応する行の、QSYS2.SYSTRIGGERS の ACTION_STATEMENT 列は、次のようになります。

```
WRAPPED QSQ07020 <encoded-suffix>
```

XDBDECOMPXML

XDBDECOMPXML プロシージャーは直列化された XML データから値を抽出し、それらの値をリレーショナル表に挿入します。

権限

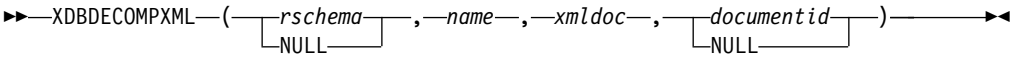
このステートメントの権限 ID が保持する特権には、以下が含まれていなければなりません。

- 次のシステム権限
  - そのプロシージャーに関連する XDBDECOMPXML サービス・プログラムに対する *EXECUTE システム権限、および
  - SYSPROC スキーマに対する USAGE 特権

このステートメントの権限 ID が保持する特権には、以下が含まれていなければなりません。

- アノテーションで指定された任意の表に対する INSERT 特権。または
- データベース管理者権限

構文



説明

スキーマは SYSPROC です。

XDBDECOMPXML ストアード・プロシージャーは、分解された XML 値を格納するために使用する列と表を示す、アノテーションが含まれる XML スキーマを使用します。参照される XML スキーマは XSR になければならず、分解で使用可能でなければなりません。XSR_COMPLETE ストアード・プロシージャーを使用すると、XML スキーマを分解で使用できます。XSR_COMPLETE ストアード・プロシージャーの呼び出し時にその時点で存在しない表を XML スキーマが参照すると、Db2 はエラーを戻します。

rschema
XML スキーマの SQL スキーマを指定する、VARCHAR(128) タイプの入力パラメーター。有効な SQL ID を指定する必要があります。SQL スキーマは、XSR 内のこの XML スキーマを識別するのに使用される修飾名の一部です (名前の残りの部分は *name* パラメーターで指定されます)。このパラメーターには NULL 値を指定することができます。NULL 値は、*name* が、 72 ページの『非修飾オブジェクト名の修飾』に指定された規則に基づいて暗黙に修飾されることを示します。

rschema を指定する場合、QSYS、QSYS2、SYSIBM、 SYSPROC、または QTEMP にすることはできません。

name
XML スキーマの名前を指定する、VARCHAR(128) タイプの入力パラメーター。有効な SQL ID を指定する必要があります。分解を実行予定の XML スキ

ーマの完全な名前は、*rschema.name* です。この XML スキーマ名は、XSR_COMPLETE ストアード・プロシージャーを呼び出した結果として既に存在し、分解で使用可能でなければなりません。このパラメーターを NULL 値にすることはできません。

xmldoc
タイプ BLOB(2G) の入力パラメーターで、分解される XML 値を指します。このパラメーターを NULL にすることはできません。

documentid
タイプ VARCHAR(1024) の入力パラメーターで、呼び出し元がこの入力 XML 文書を識別するために使用可能なストリングが含まれます。このパラメーターは NULL にすることができます。

XSR_ADDSCHEMADOC

XSR_ADDSCHEMADOC ストアード・プロシージャーは、1 次 XML スキーマ文書以外のすべての XML スキーマを XSR に追加します。

権限

XSR 内のそれぞれの XML スキーマは、1 つ以上の XML スキーマ文書で構成可能です。XML スキーマが複数の文書で構成されている場合、追加の文書に対して XSR_ADDSCHEMADOC を呼び出す必要があります。

このステートメントの権限 ID が保持する特権には、以下が含まれていなければなりません。

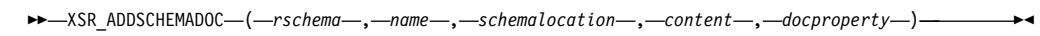
- 次のシステム権限
  - そのプロシージャーに関連するサービス・プログラムに対する *EXECUTE システム権限
  - SYSPROC スキーマに対する USAGE 特権
  - *SQLXSR オブジェクトが含まれるスキーマに対する USAGE 特権、および
  - その *SQLXSR オブジェクトに対する ALTER 特権
- データベース管理者権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- XSROBJECTCOMPONENTS および XSROBJECTHIERARCHIES カタログ表の場合:
  - 該当の表に対する INSERT 特権、および
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

プロシージャーの呼び出し元のユーザー ID には XSR_ADDSCHEMADOC ストアード・プロシージャーに対する EXECUTE 特権がなければならず、そのユーザー ID は XSROBJECTS カタログ表に記録されている XSR オブジェクトの定義者でなければなりません。

構文



説明

スキーマは SYSPROC です。

rschema
XML スキーマの SQL スキーマを指定する、VARCHAR(128) タイプの入力パラメーター。有効な SQL ID を指定する必要があります。SQL スキーマは、XSR 内のこの XML スキーマを識別するのに使用される修飾名の一部です (名前の残りの部分は *name* パラメーターで指定されます)。このパラメーターには

NULL 値を指定することができます。NULL 値は、*name* が、 72 ページの『非修飾オブジェクト名の修飾』に指定された規則に基づいて暗黙に修飾されることを示します。

rschema を指定する場合、QSYS、QSYS2、SYSIBM、 SYSPROC、または QTEMP にすることはできません。

name
XML スキーマの名前を指定する、VARCHAR(128) タイプの入力パラメーター。有効な SQL ID を指定する必要があります。この XML スキーマの完全な名前は *rschema.name* です。この XML スキーマは XSR_REGISTER ストアード・プロシージャーの呼び出し結果として既に存在していなければならず、XML スキーマ登録を完了させることはまだできません。このパラメーターを NULL 値にすることはできません。

schemalocation
タイプ VARCHAR(1000) の入力パラメーターで、NULL 値にすることができます。このパラメーターは、XML スキーマ文書の追加先である 1 次 XML スキーマ文書のスキーマ・ロケーションを示します。この引数は XML スキーマの外部名で、つまりこの 1 次文書は xsi:schemaLocation 属性を使用して XML インスタンス文書で識別できます。*schemalocation* を参照する文書では、有効な URI 形式を使用している必要があります。

content
タイプ BLOB(30M) の入力パラメーターで、追加している XML スキーマ文書のコンテンツが含まれます。この引数を NULL 値にすることはできません。XML スキーマ文書を提供する必要があります。XML スキーマ文書のコンテンツは UTF-8 でエンコードされていなければなりません。

docproperty
タイプ BLOB(5M) の入力パラメーターで、追加している XML スキーマ文書のプロパティーを示します。この引数は NULL 値にできます。NULL 値以外の場合には、値は XML 文書になります。

例

以下は、XSR_ADDSCHEMADOC ストアード・プロシージャーの呼び出し例です。

```
CALL SYSPROC.XSR_ADDSCHEMADOC(
  'MyLib',
  'MySchema',
  'http://myschema/addschema1.xsd',
  :schema_content,
  :schema_docproperties)
```

XSR_COMPLETE

XSR_COMPLETE プロシージャーは、XML スキーマ登録プロセスの一部として呼び出す最後のストアード・プロシージャーです。この登録プロセスでは、XML スキーマが XSR に登録されます。スキーマ登録がこのストアード・プロシージャーに対する呼び出しを完了するまでは、XML スキーマを妥当性検査に使用することはできません。

権限

このステートメントの権限 ID が保持する特権には、以下が含まれていなければなりません。

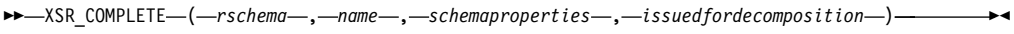
- 次のシステム権限
  - そのプロシージャーに関連するサービス・プログラムに対する *EXECUTE システム権限
  - SYSPROC スキーマに対する USAGE 特権
  - *SQLXSR オブジェクトが含まれるスキーマに対する USAGE 特権、および
  - その *SQLXSR オブジェクトに対する ALTER 特権
- データベース管理者権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- XSROBJECTS、XSROBJECTCOMPONENTS、XSROBJECTHIERARCHIES、および XSRANNOTATIONINFO カタログ表の場合:
  - XSROBJECTS、XSROBJECTCOMPONENTS、および XSROBJECTHIERARCHIES に対する UPDATE 特権
  - その XSRANNOTATIONINFO に対する INSERT 特権、および
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

このプロシージャーの呼び出し元のユーザー ID には、XSR_COMPLETE ストアード・プロシージャーに対する EXECUTE 特権がなければなりません。

構文



説明

スキーマは SYSPROC です。

rschema
XML スキーマの SQL スキーマを指定する、VARCHAR(128) タイプの入力パラメーター。有効な SQL ID を指定する必要があります。SQL スキーマは、XSR 内のこの XML スキーマを識別するのに使用される修飾名の一部です (名前の残りの部分は *name* パラメーターで指定されます)。このパラメーターには

NULL 値を指定することができます。NULL 値は、*name* が、 72 ページの『非修飾オブジェクト名の修飾』に指定された規則に基づいて暗黙に修飾されることを示します。

rschema を指定する場合、QSYS、QSYS2、SYSIBM、 SYSPROC、または QTEMP にすることはできません。

name

XML スキーマの名前を指定する、VARCHAR(128) タイプの入力パラメーター。有効な SQL ID を指定する必要があります。完了検査を実行予定の XML スキーマの完全な名前は、*rschema.name* です。この XML スキーマ名は XSR_REGISTER ストアード・プロシージャーの呼び出し結果として既に存在していなければならず、XMLスキーマ登録を完了させることはまだできません。このパラメーターを NULL 値にすることはできません。有効な文字と区切り文字に関して SQL ID に適用される規則が、このパラメーターに対しても適用されます。

schemaproperties

タイプ BLOB(5M) の入力パラメーターで、存在する場合には、この XML スキーマと関連するプロパティーを指定します。このパラメーターの値は、関連するプロパティーがない場合には NULL にでき、その他の場合にはこの XML スキーマのプロパティーを表わす XML 文書となります。

issuedfordecomposition

タイプ INTEGER の入力パラメーターで、XML スキーマを分解に使用するかどうかを示します。XML スキーマを分解に使用する場合、この値を 1 に設定してください。使用しない場合には 0 に設定します。

例

以下は、XSR_COMPLETE ストアード・プロシージャーの呼び出し例です。

```
CALL SYSPROC.XSR_COMPLETE(
  'MyLib',
  'MySchema',
  :schemaproperty_host_var,
  0)
```

XSR_REGISTER

XSR_REGISTER プロシージャーは、XML スキーマ登録プロセスの一部として呼び出す最初のストアード・プロシージャーです。この登録プロセスでは、XML スキーマが XML スキーマ・リポジトリー (XSR) に登録されます。

権限

このストアード・プロシージャーを呼び出すユーザーが、この XML スキーマの作成者と見なされます。Db2 は、XSR_COMPLETE の呼び出し時にスキーマ文書からネーム・スペース属性を取得します。

このステートメントの権限 ID が保持する特権には、以下が含まれていなければなりません。

- 次のシステム権限
  - そのプロシージャーに関連するサービス・プログラムに対する *EXECUTE システム権限。および
  - SYSPROC スキーマに対する USAGE 特権

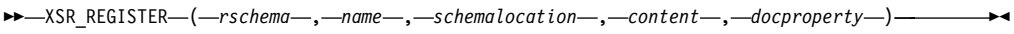
このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- スキーマ内に作成する特権
- データベース管理者権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- XSROBJECTS、XSROBJECTCOMPONENTS、および XSROBJECTHIERARCHIES カタログ表の場合:
  - 該当の表に対する INSERT 特権、および
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

構文



説明

スキーマは SYSPROC です。

rschema
XML スキーマの SQL スキーマを指定する、VARCHAR(128) タイプの入力パラメーター。有効な SQL ID を指定する必要があります。SQL スキーマは、XSR 内のこの XML スキーマを識別するのに使用される修飾名の一部です (名前の残りの部分は *name* パラメーターで指定されます)。このパラメーターには NULL 値を指定することができます。NULL 値は、*name* が、 72 ページの『非修飾オブジェクト名の修飾』に指定された規則に基づいて暗黙に修飾されることを示します。

rschema を指定する場合、QSYS、QSYS2、SYSIBM、 SYSPROC、または QTEMP にすることはできません。

name
XML スキーマの名前を指定する、VARCHAR(128) タイプの入力パラメーター。有効な SQL ID を指定する必要があります。この XML スキーマの完全な名前は *rschema.name* で、XSR 内のすべてのオブジェクトで固有でなければなりません。

schemalocation
タイプ VARCHAR(1000) の入力パラメーターで、NULL 値にすることができます。このパラメーターは、1 次 XML スキーマ文書のスキーマ・ロケーションを示します。このパラメーターは XML スキーマの外部名で、つまりこの 1 次文書は xsi:schemaLocation 属性を使用して XML インスタンス文書で識別できます。

content
タイプ BLOB(30M) の入力パラメーターで、1 次 XML スキーマ文書のコンテンツが含まれます。このパラメーターを NULL 値にすることはできません。XML スキーマ文書を提供する必要があります。XML スキーマ文書のコンテンツは UTF-8 でエンコードされていなければなりません。

docproperty
タイプ BLOB(5M) の入力パラメーターで、1 次 XML スキーマ文書のプロパティーを示します。このパラメーターは NULL 値にできます。NULL 値以外の場合には、値は XML 文書になります。

例

以下は、XSR_REGISTER ストアード・プロシージャーの呼び出し例です。

```
CALL SYSPROC.XSR_REGISTER(
  'MyLib',
  'MySchema',
  'http://myschema/primary.xsd',
  :content_host_var,
  :docproperty_host_var)
```

XSR_REMOVE

XSR_REMOVE プロシージャーは、XML スキーマのすべてのコンポーネントを除去するのに使用します。その XML スキーマを除去した後には、新しい XML スキーマを登録する際に、除去した XML スキーマの名前を再使用できます。

権限

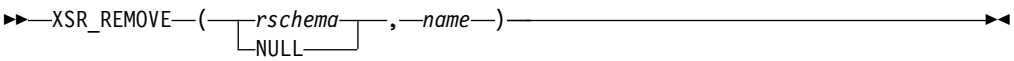
このステートメントの権限 ID が保持する特権には、以下が含まれていなければなりません。

- 次のシステム権限
  - そのプロシージャーに関連するサービス・プログラムに対する *EXECUTE システム権限。および
  - SYSPROC スキーマに対する USAGE 特権

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 次のシステム権限
  - XSR オブジェクトに関連するオブジェクトに対する *OBJOPR および *OBJEXIST システム権限
  - 除去したい XSR オブジェクトが含まれるスキーマに対する USAGE 特権、および
  - XSROBJECTS、XSROBJECTCOMPONENTS、XSROBJECTHIERARCHIES、および XSRANNOTATIONINFO カタログ表に対する DELETE 特権
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

構文



説明

スキーマは SYSPROC です。

rschema
XML スキーマの SQL スキーマを指定する、VARCHAR(128) タイプの入力パラメーター。有効な SQL ID を指定する必要があります。SQL スキーマは、XSR 内のこの XML スキーマを識別するのに使用される修飾名の一部です (名前の残りの部分は *name* パラメーターで指定されます)。このパラメーターには NULL 値を指定することができます。NULL 値は、*name* が、 72 ページの『非修飾オブジェクト名の修飾』に指定された規則に基づいて暗黙に修飾されることを示します。

rschema を指定する場合、QSYS、QSYS2、SYSIBM、 SYSPROC、または QTEMP にすることはできません。

name
XML スキーマの名前を指定する、VARCHAR(128) タイプの入力パラメータ

ー。有効な SQL ID を指定する必要があります。除去予定の XML スキーマの完全な名前は、*rschema.name* です。この XML スキーマ名は、XSR_REGISTER ストアード・プロシージャーの呼び出し結果として既に存在していなければなりません。このパラメーターを NULL 値にすることはできません。

例

以下は、XSR_REMOVE ストアード・プロシージャーの呼び出し例です。

```
CALL SYSPROC.XSR_REMOVE(
  'MyLib',
  'MySchema')
```

第 6 章 照会

照会 は、結果表または中間結果表を指定するものです。照会は、特定の SQL ステートメントのコンポーネントの 1 つになります。

照会には、副選択、全選択、および選択ステートメント という 3 つの形式があります。単一行のみを検索できる別の SQL ステートメントがあります。これについては、 1670 ページの『SELECT INTO』を参照してください。

権限

どのような照会形式の場合でも、ステートメントの権限 ID によって保持される特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれの表またはビューごとに、
  - 表やビューに対する SELECT 特権、および
  - 表またはビューが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

照会にユーザー定義関数が含まれている場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

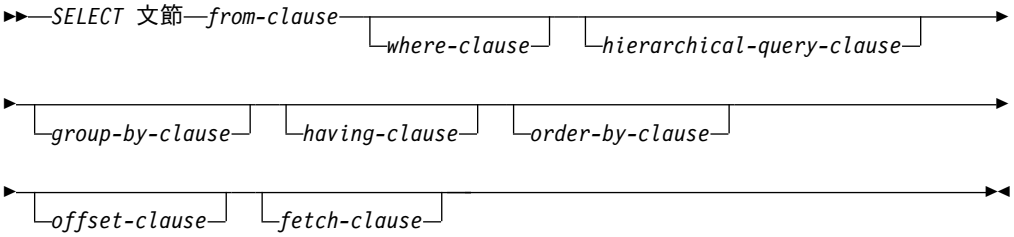
- ステートメント内で識別されるそれぞれのユーザー定義関数に対して:
  - その関数に対する EXECUTE 特権
- データベース管理者権限

SQL 特権に対応するシステム権限については、『表またはビューへの権限を検査する際の対応するシステム権限』および『関数またはプロシージャーへの権限を検査する際の対応するシステム権限』を参照してください。

照会が、アクティブな行アクセス制御または列アクセス制御が含まれている表を参照していて、その表に対して行の許可または列マスクが定義されている場合、ステートメントの権限 ID は、それらの行の許可または列マスクの定義に指定されているオブジェクトを参照するための権限を必要としません。

副選択

副選択 は、全選択のコンポーネントの 1 つです。



副選択では、FROM 文節で識別されている表またはビューから得られる結果表を指定します。結果表が得られる過程は、各操作の結果が次の操作への入力となるような一連の操作として説明できます。 (これは、副選択の説明でだけ使用する考え方です。結果表を得るために、この説明とはまったく異なる方法が使用されることもあります。正しい結果を得るために副選択の部分を実際に実行する必要がない場合は、実行してもしなくても構いません。)

行または列のアクセス制御が施行されている表を副選択で直接または間接的に参照する場合、行の許可または列マスクに定義されている規則は、結果表の行の派生方法に影響します。通常、これらの規則は、プロセスの許可 ID に基づきます。

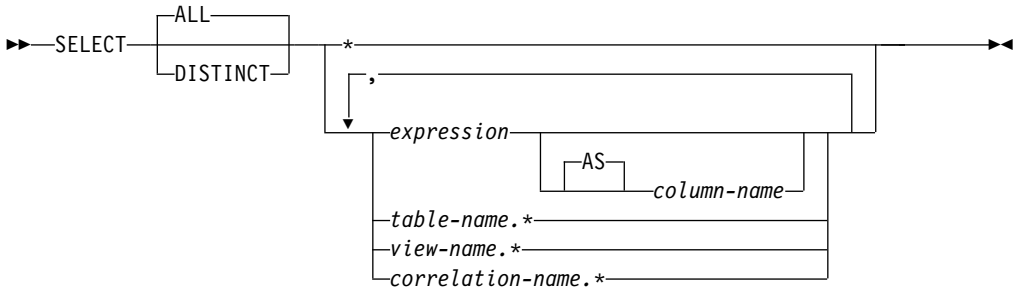
スカラー副選択 は括弧で囲んだ副選択で、単一の結果行および単一の結果列を戻します。副選択の結果に行が含まれない場合、NULL 値が戻されます。結果の中に複数の行がある場合には、エラーが戻されます。

操作の (仮の) 順序は、次のようになります。

1. FROM 文節
2. 階層照会文節
3. WHERE 文節
4. GROUP-BY 文節
5. HAVING 文節
6. SELECT 文節
7. ORDER BY 文節
8. OFFSET 文節
9. FETCH 節

SELECT 文節

SELECT 文節は、最終的な結果表の列を指定します。



列の値は、R に選択リスト を適用することによって生成されます。選択リストは SELECT 文節に指定する名前または式です。R は、副選択でのその前の演算結果です。例えば、 SELECT、 FROM、および WHERE の文節だけを指定した場合、R は WHERE 文節の結果です。

ALL

最終的な結果表のすべての行を選択します。重複行の除去は行いません。これはデフォルトです。

DISTINCT

最終的な結果表にある重複行の組から、1 行だけを残して他の行をすべて除去します。 2 つの行が互いに重複したものとして扱われるのは、一方の行にあるそれぞれの値が、もう一方の行の対応する値にすべて等しい場合だけです。 (重複行を判別する場合、NULL 値どうしは等しいものとされます。) 除去する値を判別するために、照合順序も使用されます。

選択リスト に DATALINK または XML 列、あるいは XML データ・タイプの値を戻す式が入っている場合、DISTINCT は使用できません。

列アクセス制御は SELECT DISTINCT の操作に影響しません。重複行の除去は、マスク値ではなく、元の列値に基づきます。ただし、列マスクの適用後、最終結果表内のマスク値が SELECT DISTINCT による一意性を反映しない可能性があります。

SELECT DISTINCT の結果を直接的または間接的に派生させる列に列マスクが適用される場合、SELECT DISTINCT から非 deterministic の結果が戻されることがあります。以下の条件は、非 deterministic の結果が戻されるいくつかの例を示しています。

- 列マスクの定義が列マスクの適用先の表にある他の列を参照している。
- 列が組み込みスカラー関数 (COALESCE、IFNULL、NULLIF、MAX、MIN、LOCATE など) の引数で参照されている。
- 列が集約関数の引数で参照されている。
- 列が式に組み込まれ、その式に非 deterministic 関数かまたは外部アクションのある関数が含まれている。

選択リストの表記法

***** 表 R の列のリストを表します。列は、ここに指定する順序で FROM 文節によ

って生成されます。隠し属性を指定して定義された列は含まれません。名前のリストは、SELECT 文節が入っているステートメントが準備されるときに設定されます。このため、ステートメントが準備された後で表に追加された列があっても、* ではその列を識別しません。

* は、行の許可または列マスクの定義では使用できません。

expression
結果列の値を指定します。式 の中の各列名 は、R の列を明瞭に識別するものでなければなりません。

列名 または **AS** 列名
結果の列の名前、または名前の付け直しを指定します。名前は、修飾してはなりません。また固有である必要はありません。

*name.**
name の列のリストを表します。隠し属性を指定して定義された列は含まれません。名前 には、表名、ビュー名、または相関名を指定できます。ここには、SELECT 文節の直後の FROM 文節で指定した直接的な表名、ビュー名、または相関名を指定しなければなりません。リストの最初の名前は、表またはビューの最初の列を識別し、2 番目の名前は表またはビューの 2 番目の列を識別するというように、対応する列を順に識別します。

名前のリストは、SELECT 文節が入っているステートメントが準備されるときに設定されます。このため、ステートメントが準備された後で表に追加された列があっても、* ではその列を識別しません。

*name.** は、行の許可または列マスクの定義では使用できません。

通常、SQL ステートメントが暗黙に再バインド (再準備) される時点では、名前のリストは再確立されません。したがって、ステートメントによって戻される列の数は変わりません。ただし、名前のリストが再度確立され、列の数が変わる場合が 4 つあります。

- SQL プログラムまたは SQL パッケージが保管され、その保管元システムとは異なるリリースの IBM i 製品上で復元される場合。
- SQL プログラムまたはパッケージに SQL 命名規則の指定があり、その SQL プログラムまたはパッケージの作成後に、その SQL プログラムの所有者が変更されている場合。
- より新しいリリースの IBM i オペレーティング・システムのインストール後、初めて SQL ステートメントが実行される場合。
- INSERT ステートメントの全選択、または述部の全選択で SELECT * が行われ、しかもその全選択で参照される表またはビューが削除され、他の列によって再作成されている場合。

SELECT の結果の列の数は、実行形式の選択リスト (つまり、準備時に確立されたリスト) にある式の数と同じであり、8000 を超えることはできません。副照会を EXISTS 述部で使用している場合を除いて、副照会の結果は単一の式でなければなりません。

選択リストの適用

選択リストを R に適用した結果は、GROUP BY または HAVING が使用されているかによって異なる場合があります。

GROUP BY または HAVING が使用されている場合

- 選択リスト内の各 *column-name* は、グループ化式にするか、集約関数の中で指定するか、相関参照にする必要があります。
  - グループ化式が列名の場合、選択リストには列名への加算演算子を適用できます。例えば、グループ化式が列 C1 であれば、選択リストに C1+1 を含めることができます。
  - グループ化式が列名でない場合、選択リストには式への加算演算子を適用できません。例えば、グループ化式が C1+1 の場合、選択リストに C1+1 を含めることはできますが、(C1+1)/8 は含めることができません。
- 選択リストは R の各グループに適用され、結果には R にあるグループと同じ数の行が含まれます。選択リストが R のグループに適用されるとき、選択リスト内の集約関数の引数はそのグループの中から与えられます。
- RRN、RID、DATAPARTITIONNAME、DATAPARTITIONNUM、DBPARTITIONNAME、DBPARTITIONNUM、および HASHED_VALUE 関数を指定することはできません。

GROUP BY または HAVING のどちらも使用されていない場合

- 各列名 を集約関数で指定するか相関参照にする場合以外は、選択リストの中で集約関数を使用してはなりません。
- 集約関数が入っていない選択リストは、R の各行に適用され、結果には R にある行と同じ数の行が入ります。
- 選択リストを集約関数のリストにした場合は、関数の引数が R から与えられ、選択リストを適用した結果は 1 つの行になります。

どちらの場合も、結果の *n* 番目の列には、命令形式の選択リストにある *n* 番目の式を適用することによって指定された値が入ります。

結果列に対する列マスクの効果**:** 列マスクが有効であると、それらが最外部の選択リストの最終結果表の値を決定します。列マスクが列に対して有効であると、その列が最外部の選択リストに現れる場合 (暗黙的または明示的のいずれか)、列マスクがその列に適用され、最終結果表の値が生成されます。列自体は最外部の選択リストには現れないが、出力に含まれる場合 (例えば、マテリアライズ表式またはビューに現れる場合)、マスクされた値は最終結果表で使用できるように表式またはビューの結果表に含まれます。

有効な列マスクが、WHERE 節、GROUP BY 節、HAVING 節、SELECT DISTINCT 節、および ORDER BY 節などの、ステートメント内の他の節の操作と競合することはありません。

最終結果表に戻される行は同じですが、結果行の値がマスクされている可能性があります。したがって、マスクされた値を持つ列も sort-key 式が指定された ORDER BY 節に現れる場合、順序は元の列値に基づきます (最終結果表のマスクされた値がその順序を反映していない可能性があります)。同様に、マスクされた値には、SELECT DISTINCT により適用される固有性が反映されない可能性があります。マスクされた列が式に組み込まれている場合、式が評価される前に列マスクが列に適用されるため、式の結果が異なることがあります。例えば、列 SSN の列マスクにより、マスク値に対して DISTINCT 操作が行われるため、関数 COUNT(DISTINCT SSN) の結果が変わることがあります。

列マスクの定義が最終結果表の列値をマスクする SQL ステートメントに適用される場合、列マスクのセマンティクスがそのステートメントの特定の SQL セマンティクスと競合する可能性があります。これらの場合、ステートメントと列マスクを組み合わせると、エラーが戻されます。

有効な列マスクの適用の詳細については、 942 ページの『ALTER TABLE』を参照してください。

結果列の NULL 属性

結果列が以下のものから得られた場合は、結果列に NULL 値が入ることがあります。

- COUNT および COUNT_BIG を除く集約関数
- NULL 値が許される任意の列
- NULL 値のオペランドを使用できるスカラー関数または式
- 標識変数、SQL パラメーター、または変数を持つホスト変数か、 Java NULL 値を表すことが可能なタイプの変数または式 (Java の場合)
- UNION または INTERSECT の結果 (選択リスト内の対応する項目の中に、NULL 可能な項目が少なくとも 1 つある場合)
- 外側の選択リスト内の算術式
- スカラー全選択
- ユーザー定義のスカラー関数または表関数
- GROUPING SETS グループ化式

結果列の名前

- AS 文節の指定がある場合、結果列の名前は、AS 文節で指定された名前です。
- AS 文節を指定しないで列リストを相関文節で指定した場合は、結果列の名前は、その相関列リストの対応する名前になります。
- AS 文節も相関文節の列リストも指定せず、結果列が単一の列からだけ得られる (関数も演算子もない) 場合は、結果列の名前はその列の修飾の付かない名前になります。
- AS 文節も相関文節の列リストも指定せず、結果の列を 1 つの変数からだけ派生させる (関数も演算子もない) 場合は、結果の列の名前がその変数の非修飾名になります。
- AS 文節も相関文節の列リストも指定せず、結果列が単一の疑似列からだけ得られる (関数も演算子もない) 場合は、結果列の名前はその疑似列の名前になります。
- 上記以外の結果列はすべて、名前を持ちません。

結果列のデータ・タイプ

SELECT の結果の各列のデータ・タイプは、その列を得るときの元になった式から取得されます。

元になった式	結果列のデータ・タイプ
数値の列の名前	その列のデータ・タイプと同じ (10 進数の列については、同じ精度および位取りを持つ)。

元になった式	結果列のデータ・タイプ
整数定数	INTEGER または BIGINT (定数の値が INTEGER の範囲外であっても、BIGINT の範囲内である場合)。
10 進数または浮動小数点定数	その定数のデータ・タイプと同じ (10 進定数については同じ精度および位取りを持つ)。
DECFLOAT(7) 変数の名前	DECFLOAT(16)
数値変数の名前	その変数のデータ・タイプと同じ (10 進数の変数については同じ精度および位取りを持つ)。変数のデータ・タイプが、SQL データ・タイプに一致しない (例えば、COBOL の DISPLAY SIGN LEADING SEPARATE など) 場合、結果列は 10 進数になります。
式	式の結果のデータ・タイプと同じ (10 進数の結果については、同じ精度および位取りを持つ)。式の結果については、196 ページの『式』で説明しています。
任意の関数	関数の結果のデータ・タイプ。組み込み関数については、291 ページの『第 4 章 組み込み関数』を参照してデータ・タイプを判別してください。ユーザー定義の関数の場合は、結果のデータ・タイプは、その関数の CREATE FUNCTION ステートメントで定義されているデータ・タイプになります。
ストリング列の名前	その列のデータ・タイプと同じ (長さ属性も同じ)。
ストリング変数の名前	その変数のデータ・タイプと同じ (長さ属性もその変数の長さ属性と同じ)。変数のデータ・タイプが、SQL データ・タイプと一致しない (例えば、C の NUL 終了ストリングなど) 場合、結果列は可変長ストリングになります。
長さ *n* の文字ストリング定数	VARCHAR(*n*)
長さ *n* のグラフィック・ストリング定数	VARGRAPHIC(*n*)
XML 列または変数の名前	XML
日付/時刻の列の名前、または ILE RPG コンパイラーまたは ILE COBOL コンパイラー日付/時刻ホスト変数の名前	その列または変数のデータ・タイプと同じ。
特殊タイプ列の名前	その列の特殊タイプと同じ (存在する場合は、長さ、精度、および位取り属性が同じ)。
データ・リンク列の名前	同じ長さ属性のデータ・リンク。
行 ID 列または行 ID 変数の名前	ROWID

FROM 文節

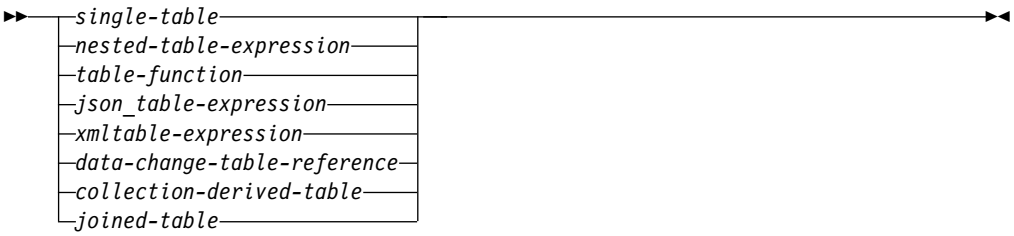
FROM 文節は、中間結果の表を指定します。

table-reference が 1 つだけ指定されている場合、中間結果表は単に、その *table-reference* の結果です。 FROM 文節で複数の表参照 を指定した場合は、中間結果の表は、指定された表参照の行のあらゆる組み合わせ (カルテシアン積) から構成されます。結果の各行は、最初の *table-reference* の行を 2 番目の *table-reference* の行に連結し、それを 3 番目の *table-reference* の行に連結し、以下同様の手順で連結した行です。結果の行数は、個々の表参照 すべての行数の積です。

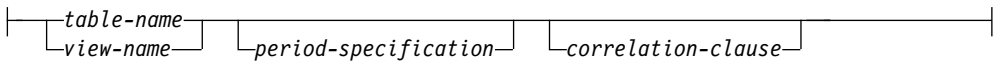
table-reference で行のアクセス制御が実施されている場合、*table-reference* は少なくとも 1 つの行の許可 (デフォルトの行の許可) を持ちます。単一の *table-reference* に対して複数の行の許可が定義されている場合は、有効にされている各許可にある検索条件に対して論理 OR 演算子を適用することにより、行アクセス制御検索条件が派生されます。この派生された検索条件は、*table-reference* に対するフィルターとして機能し、それにより、*table-reference* の結果表が決定されます。この結果表には、副選択の許可 ID によってアクセスできます。

table-reference

table-reference は、中間結果の表を指定します。



single-table:



nested-table-expression:

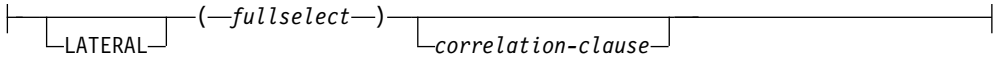
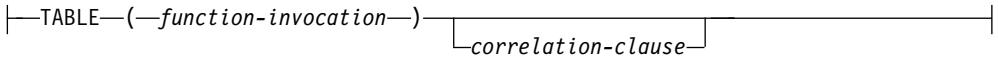
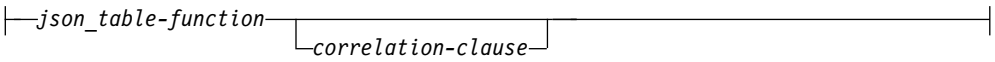


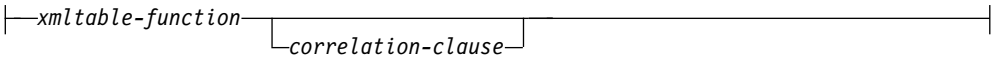
table-function:



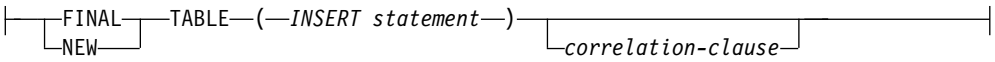
json_table-expression:



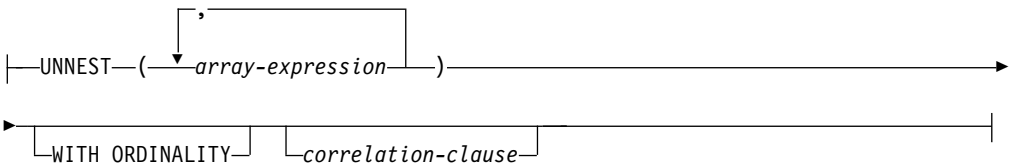
xmltable-expression:



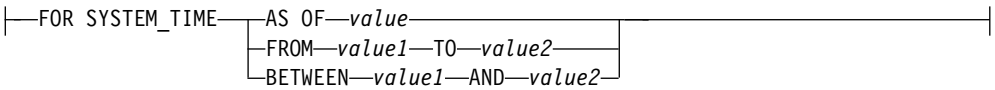
data-change-table-reference:



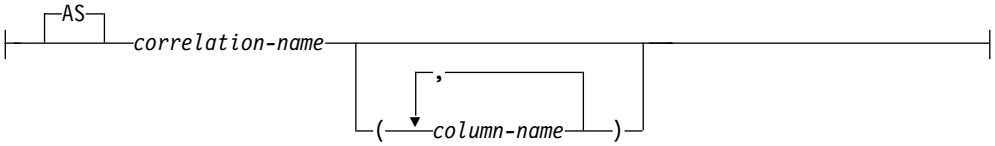
collection-derived-table:



period-specification:



correlation-clause:



- *period-specification* なしで単一の表またはビューが識別される場合、中間結果表は単にその表またはビューになります。 *period-specification* が *table-name* または *view-name* に指定された場合、中間結果表は、テンポラル表の行のうち期間が指定と一致するもので構成されます。
- 括弧内の全選択は、ネストされた表式 と呼ばれる。[79] ネストされた表式を指定すると、結果表は、そのネストされた表式の結果となります。結果の列には固有の名前は必要ありませんが、固有の名前を持たない列は明示的に参照することができません。

79. ネストされた表式 は、派生表 とも呼ばれます。

- *function-name*、*json_table-expression*、または *xmltable-expression* を指定した場合、中間結果表はその表関数によって戻された行のセットです。
- *data-change-table-reference* を指定した場合、中間結果の表は、INSERT ステートメントによって挿入される行の集合になる。
- *collection-derived-table* を指定する場合、中間結果表は、1 つ以上の配列からの行の集合になる。
- 結合表 を指定した場合は、中間結果の表は、1 つ以上の結合演算の結果になる。詳しくは、 804 ページの『結合表』を参照してください。

表参照 が分散表または読み取りトリガーを持つ表を識別する場合、照会には以下を含めることはできません。

- EXCEPT または INTERSECT 操作
- 全選択の中の VALUES
- OLAP の指定
- 反復共通表式および CONNECT BY
- ORDER OF
- スカラー全選択 (スカラー副選択はサポートされています)
- 全外部結合
- GROUP BY の中の LOB、
- グループ化集合またはスーパー・グループ
- 副選択の中の ORDER BY または FETCH 節
- OFFSET 節、または N 行の変数を指定した FETCH 節
- CORRELATION、COVARIANCE、COVARIANCE_SAMP、LISTAGG、MEDIAN、PERCENTILE_CONT、 PERCENTILE_DISC、または回帰集約関数
- VERIFY_GROUP_FOR_USER、LOCATE_IN_STRING、2 つの引数を指定した LTRIM または RTRIM、EPOCH を指定した EXTRACT 関数、 HASH 関数
- BSON_TO_JSON、JSON_ARRAY、JSON_ARRAYAGG、JSON_OBJECT、JSON_OBJECTAGG、JSON_QUERY、JSON_TABLE、JSON_TO_BSON、および JSON_VALUE 関数と、IS JSON および JSON_EXISTS 述部
- XMLAGG、XMLATTRIBUTES、XMLCOMMENT、XMLCONCAT、XMLDOCUMENT、XMLELEMENT、XMLFOREST、XMLGROUP、XMLNAMESPACES、 XMLPI、XMLROW、XMLTABLE、または XMLTEXT 関数
- CONTAINS または SCORE 関数
- ユーザー定義関数のデフォルト値
- グローバル変数、または
- 配列の参照

CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターが NULL 以外の値の *CTST* に設定されていて、SYSTIME オプションの値が YES で、*table-name* または

view-name がシステム期間テンポラル表を示している場合、表参照は、あたかも特殊レジスターが NULL 値に設定された状態で次の指定が含まれるかのように実行されます。

```
table-name FOR SYSTEM_TIME AS OF CTST
```

または

```
view-name FOR SYSTEM_TIME AS OF CTST
```

FROM 文節中の名前のリストは、以下の規則に従わなければなりません。

- それぞれの表名 およびビュー名 は、現行サーバーの既存の表またはビューの名前とするか、その表参照 を含む副選択に先立って定義された共通表式の表 *ID* とする必要がある。
- 直接名 (exposed name) は固有でなければならない。直接名には次のものがあります。
  - 相関名
  - 後ろに相関名 が続かない表名 またはビュー名
  - データ変更表参照 の後に相関名 が続いていない場合、データ変更表参照 のターゲットである表名 またはビュー名

  correlation-clause が *nested-table-expression*、*table-function*、*json_table-expression*、*xmltable-expression*、*data-change-table-reference*、および *collection-derived-table* に指定されていない場合、その表参照には直接名がありません。
- 各関数名 とそれぞれの引数のタイプを解決した結果が、現行サーバー上に存在する表関数のどれかにならなければならない。関数解決と呼ばれるアルゴリズム (185 ページの『関数解決』に記載) は、関数名と引数を使用して、使用すべき正しい関数を決定します。
- 各 *array-variable-name* は、SQL ルーチンの配列変数を示していなければなりません。

各相関名 は、直前の先行表参照 によって指定される中間結果表の指定子として定義されます。

列に対する修飾参照では、直接的な名前を使用しなければなりません。同じ表名またはビュー名を 2 回指定する場合は、少なくとも 1 つの指定の後に *correlation-name* がなければなりません。表またはビューの列に対する参照を修飾するためには、*correlation-name* を使用します。*correlation-name* を指定する場合は、column-name も同時に指定することによって、*table-reference* の列に名前を指定することができます。列リストを指定する場合は、その列リストの中で、表またはビューの各列について、および、*nested-table-expression*、*table-function*、*json_table-expression*、*xmltable-expression*、*data-change-table-reference*、*collection-derived-table* の個々の結果列について、それぞれ名前を指定する必要があります。詳しくは、 163 ページの『相関名』を参照してください。
correlation-clause に *column-name* が含まれない場合、直接的な列名は以下のようになります。

- *table-reference* が *table-name* または *view-name* である場合は、参照される表またはビューの列名。

- *table-reference* が *function-name* 参照である場合は、CREATE FUNCTION ステートメントの RETURNS 節で指定されている列名。
- *table-reference* が *json_table-expression* または *xmltable-expression* である場合は、*json_table-expression* または *xmltable-expression* の COLUMNS 節で指定されている列名。
- *table-reference* が *nested-table-expression* である場合は、全選択により返される列名。
- *table-reference* が *data-change-table-reference* である場合は、任意の定義済み INCLUDE 列に加えて、データ変更ステートメントのターゲット表にある列名。

一般に、*nested-table-expressions*、*table-functions*、および *collection-derived-tables* は、どの FROM 文節でも指定することができます。ネストされた表式、表関数、およびコレクション派生表からの列は、選択リストの中および副選択の後続部分で、相関名を使用して参照することができます。この相関名の有効範囲は、FROM 文節の他の表名またはビュー名のための相関名と同じです。ネストされた表式は、次の場合に使用することができます。

- ビューの代わりとして。これはそのビューを作成するのを避けるためです (そのビューの一般的な使用が要求されない場合)。
- 必要な結果表が変数に基づいているとき。

period-specification

期間指定が *table-reference* に適用されることを指定します。 *period-specification* を指定した照会は、テンポラル照会です。 CONTAINS または SCORE 組み込み関数で参照される列を含む表またはビューに *period-specification* を指定することはできません。

表参照の行は、期間指定を適用することによって得られます。テンポラル照会の中間結果表には、システム期間テンポラル表定義の ON DELETE ADD EXTRA ROW 属性のために関連する履歴表に追加された行は含まれません。

ビュー参照の行は、そのビューの結果表を計算する際にアクセスするすべてのテンポラル表に対して期間指定を適用することにより導出されます。ビューがテンポラル表にアクセスしない場合、期間指定はビューの結果表に影響を及ぼしません。ビュー定義内のいずれかの表参照に *period-specification* が指定された場合、そのビューの表参照は *period-specification* を含むことはできません。そのビューの定義は、NO SQL 以外のデータ・アクセス標識を持つ外部関数を参照してはなりません。また、インライン関数でない限り、SQL 関数を参照してはなりません。

CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターが NULL 値以外の値に設定された場合は、SYSTIME オプションの値が NO でない限り、*period-specification* を使用できません。

FOR SYSTEM_TIME
period-specification に SYSTEM_TIME 期間を使用することを指定します。表参照は、ビューまたはシステム期間テンポラル表でなければなりません。

AS OF *value*
table-reference に、指定された期間の開始値が *value* 以下であり、期間の終了値

が *value* よりも大きいという条件に一致する行が含まれることを指定します。*value* が NULL 値の場合、 *table-reference* に行は含まれません。

value
組み込みデータ・タイプの値を戻す式を指定します。式の結果は、 130 ページの『日付/時刻の比較』で規定された比較規則に従って、TIMESTAMP(12) と比較可能でなければなりません。

式に、列参照およびスカラー全選択を含めてはなりません。また、非決定的な関数、外部アクションである関数、および SQL データを変更する関数を含めることもできません。

FROM *value1* **TO** *value2*
table-reference には、*value1* から *value2* までに指定された期間に存在する行が含まれることを指定します。行の期間の開始値が *value2* より小さく、かつ行の期間の終了値が *value1* よりも大きい場合に、その行が *table-reference* に含まれます。*value1* が *value2* 以上である場合、あるいは *value1* または *value2* が NULL 値である場合、*table-reference* に行は含まれません。

value1 または *value2*
組み込みデータ・タイプの値を戻す式を指定します。式の結果は、 130 ページの『日付/時刻の比較』で規定された比較規則に従って、TIMESTAMP(12) と比較可能でなければなりません。

式に、列参照およびスカラー全選択を含めてはなりません。また、非決定的な関数、外部アクションである関数、および SQL データを変更する関数を含めることもできません。

BETWEEN *value1* **AND** *value2*
table-reference に、*value1* から *value2* での範囲内の任意の時点で指定された期間がオーバーラップする行を含むことを指定します。行の期間の開始値が *value2* 以下であり、かつ行の期間の終了値が *value1* よりも大きい場合に、その行が *table-reference* に含まれます。*value1* が *value2* より大きい場合、あるいは *value1* または *value2* が NULL 値である場合、*table-reference* に行は含まれません。*value1* = *value2* の場合、式は AS OF *value1* と等価です。

value1 または *value2*
組み込みデータ・タイプの値を戻す式を指定します。式の結果は、 130 ページの『日付/時刻の比較』で規定された比較規則に従って、TIMESTAMP(12) と比較可能でなければなりません。

式に、列参照およびスカラー全選択を含めてはなりません。また、非決定的な関数、外部アクションである関数、および SQL データを変更する関数を含めることもできません。

代替構文:

- FOR SYSTEM_TIME AS OF の代わりに AS OF TIMESTAMP を指定できます。
- FOR SYSTEM_TIME BETWEEN の代わりに VERSIONS BETWEEN TIMESTAMP を指定できます。

json_table-function

組み込み JSON_TABLE 表関数の呼び出しを指定します。詳しくは、 744 ページの『JSON_TABLE』を参照してください。

xmltable-function

組み込み XMLTABLE 表関数の呼び出しを指定します。詳しくは、 765 ページの『XMLTABLE』を参照してください。

データ変更表参照

データ変更表参照 は、その文節に入っている INSERT ステートメントによって直接変更される行に基づく中間結果表を指定します。データ変更表参照 は、SELECT ステートメント、SELECT INTO ステートメント、SET *variable* ステートメント内で使用されるか、または割り当てステートメント内の唯一の全選択として使用される、外側の全選択の FROM 文節内において唯一の表参照 でなければなりません。

データ変更表参照 の中間結果表は、挿入されたすべての行を含みます。挿入された表の列はすべて、INSERT ステートメントで定義された INCLUDE 列と一緒に副選択内で参照することができます。データ変更表参照 には、以下の制限があります。

- 外側レベルの全選択内にしか指定できません。
- INSERT ステートメントのターゲット表またはビューは、照会内で参照されている表またはビューと見なされます。そのため、照会の権限 ID には、INSERT に必要な特権だけでなく、その表またはビューに対する権限も必要です。
- INSERT ステートメント内の全選択には、INSERT ステートメントの全選択の外部にある列への相関参照が含まれていてはなりません。
- SELECT ステートメントのデータ変更表参照 により、カーソルは読み取り専用となります。これは、UPDATE WHERE CURRENT OF および DELETE WHERE CURRENT OF が使用できないことを意味します。
- INSERT がビューを参照している場合、そのビューは WITH CASCADED CHECK OPTION を使用して定義する必要があります。あるいは、そのビューは WITH CHECK OPTION を使用して定義されている可能性があります。さらに、そのビューには、以下を含む WHERE 文節を入れることはできません。
  - SQL データを変更する関数
  - deterministic でない関数または外部アクションを持つ関数
- データ変更表参照 文節は、ビュー定義またはマテリアライズ照会表定義に指定することはできません。
- SQL データ変更ステートメントのターゲットが INSTEAD OF INSERT トリガーによって定義されたビューである場合は、エラーが返されます。

データ変更ステートメントのターゲット表に対する行アクセス制御が施行されている場合、中間結果表の行は、有効にされている行の許可で指定されている規則を既に満たしています。データ変更ステートメントのターゲット表に対する列アクセス制御が施行されている場合、有効にされている列マスクは、最外部の選択リストに適用されます。詳しくは、 789 ページの『SELECT 文節』を参照してください。

FINAL TABLE
中間結果表の行が SQL データ変更ステートメントの完了時に表示される際に、

これらの行がその SQL データ変更ステートメントによって挿入された一連の行を表すことを示します。 AFTER INSERT トリガーまたは参照制約があり、その結果として、データ変更ステートメントのターゲットとなる表に挿入された行に対して追加変更が行われた場合は、エラーが返されます。

NEW TABLE
中間結果表の行が、参照制約や AFTER トリガーのアプリケーションより前に SQL データ変更ステートメントによって変更された一連の行を表していることを示します。参照制約や AFTER トリガーに対する追加の処理のため、ステートメントの完了時にターゲット表にあるデータは、中間結果表のデータと一致しないことがあります。

コレクション派生表

コレクション派生表を使用すると、配列のエレメントを行にネスト解除できます。

array-expression
配列データ・タイプを戻す式。式は、以下の式のいずれかでなければなりません。

- SQL 変数
- SQL パラメーター
- パラメーター・マーカーの CAST 指定

UNNEST 関数によって生成される結果列の名前は、collection-derived-table 文節の correlation-clause の一部として提供できます。

結果表は入力引数に応じて異なります。

- 単一の配列引数が指定されている場合、結果は列のデータ・タイプが配列エレメントのデータ・タイプと一致する単一列の表になります。
- 複数の配列を指定すると、最初の配列が結果表の最初の列に入れられ、2 番目の配列が 2 番目の列に入るといった形で、以下同様に続きます。各列のデータ・タイプは、対応する配列引数の配列エレメントのデータ・タイプと一致します。WITH ORDINALITY を指定すると、タイプ BIGINT の追加列 (配列のエレメントの位置が含まれる) が付加されます。

  配列のカーディナリティーが等しくない場合には、結果表のカーディナリティーは最も大きいカーディナリティーを持つ配列と同じになります。行の配列指標の値が対応する配列のカーディナリティーより大きい場合は常に、表の列値は NULL 値に設定されます。つまり、各配列が 2 つの列 (1 つは副指標用、もう 1 つはデータ用) が入っている表と見なされる場合、UNNEST は、同じ副指標を結合述部として使用して配列間で OUTER JOIN を実行します。

UNNEST を指定できるのは、SQL プロシージャーまたは SQL 関数内のみです。

表参照における相関参照

相関参照は、ネストされた表式 で使用することができます。基本的な規則として、相関参照は、副照会階層内の上位レベルにおける表参照 からの参照である必要があります。この階層には、FROM 文節の左から右の処理で既に解決済みの表参照 が含まれます。ネストされた表式の場合、TABLE または LATERAL キーワードは全選択の前に来なければなりません。詳しくは、 1773 ページの『SQL パラメーター

および変数の参照』を参照してください。

表関数には、同じ FROM 文節の中にある他の表に対する 1 つ以上の相関参照を含めることができます。ただし、これは、FROM 文節内での左から右への表の順序において、参照される表がその参照より前にある場合に限られます。オプションのキーワード TABLE または LATERAL を指定する場合は、ネストされた表式でも同じ機能を使用できます。そうでない場合は、副照会の階層内の上位レベルに対する参照のみが許されます。

同じ FROM 文節内の他の表への相関参照が含まれている場合、ネストされた表式または表関数の扱いは次のようになります。

- RIGHT OUTER JOIN、FULL OUTER JOIN、または RIGHT EXCEPTION JOIN で使用することはできない
- FROM 文節内での左から右への表の順序において参照される表がその参照より前にある場合は、 LEFT OUTER JOIN または INNER JOIN で使用できる

表参照 が分散表または読み取りトリガーを持つ表を識別する場合、以下のときには、ネストされた表式に同じ FROM 文節内の他の表に対する相関参照を含めることはできません。

- ネストされた表の式に UNION、EXCEPT、または INTERSECT が含まれている。
- ネストされた表の式の選択リストで DISTINCT キーワードを使用している。
- ネストされた表の式に ORDER BY および FETCH 文節が含まれている。
- ネストされた表の式が、上記の制約事項の 1 つを含む別のネストされた表の式の FROM 文節の中にある。

代替構文: LATERAL の代わりに TABLE を指定できます。

例 1

次は正しい例です。

```
SELECT D.DEPTNO, D.DEPTNAME, EMPINFO.AVGSAL, EMPINFO.EMPCOUNT
FROM DEPARTMENT D,
  (SELECT AVG(E.SALARY) AS AVGSAL,COUNT (*) AS EMPCOUNT
     FROM EMPLOYEE E
     WHERE E.WORKDEPT =
       (SELECT X.DEPTNO
          FROM DEPARTMENT X
          WHERE X.DEPTNO = E.WORKDEPT ) ) AS EMPINFO
```

次の例は、ネストされた表式 の WHERE 文節において、D.DEPTNO に対する参照が副照会階層の外側にある表を参照しようとしているため、正しくありません。

```
SELECT D.DEPTNO, D.DEPTNAME,
        EMPINFO.AVGSAL, EMPINFO.EMPCOUNT                    ***INCORRECT***
  FROM DEPARTMENT D,
  (SELECT AVG(E.SALARY) AS AVGSAL,COUNT (*) AS EMPCOUNT
     FROM EMPLOYEE E
     WHERE E.WORKDEPT = D.DEPTNO ) AS EMPINFO
```

次の例は、ネストされた表式 の WHERE 文節において、 D.DEPTNO に対する参照がネストされた表式 の前にある DEPT を参照し、 LATERAL キーワードが指定されているため、有効です。

```
SELECT D.DEPTNO, D.DEPTNAME,
        EMPINFO.AVGSAL, EMPINFO.EMPCOUNT
  FROM DEPARTMENT D,
  LATERAL (SELECT AVG(E.SALARY) AS AVGSAL,COUNT (*) AS EMPCOUNT
     FROM EMPLOYEE E
     WHERE E.WORKDEPT = D.DEPTNO ) AS EMPINFO
```

例 2

次の表関数の例は有効です。

```
SELECT t.c1, z.c5
  FROM t, TABLE(tf3 (t.c2 ) ) AS z
  WHERE t.c3 = z.c4
```

次の例は、t.c2 に対する参照が、FROM 文節内で表関数より右にある表を参照しているため、無効です。

```
SELECT t.c1, z.c5
  FROM TABLE(tf6 (t.c2 ) ) AS z, t                  ***INCORRECT***
  WHERE t.c3 = z.c4
```

例 3

次の表関数の例は有効です。

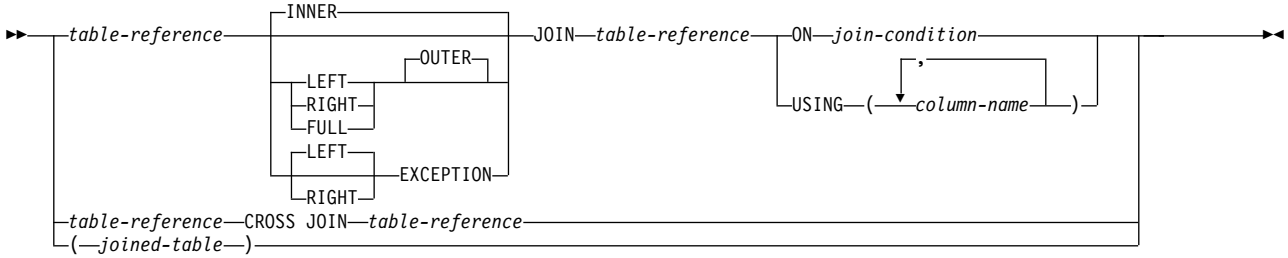
```
SELECT t.c1, z.c5
  FROM t, TABLE(tf4 (2 * t.c2 ) ) AS z
  WHERE t.c3 = z.c4
```

次の例は、b.c2 に対する参照が、FROM 文節内で b.c2 に対する参照が含まれる表関数より右にある表関数を参照しているため、無効です。

```
SELECT a.c1, b.c5
  FROM TABLE(tf7a (b.c2 ) ) AS z,                  ***INCORRECT***
       TABLE(tf7b (a.c6 ) ) AS b
  WHERE a.c3 = b.c4
```

結合表

結合表 は、内部結合、外部結合、クロス結合、または例外結合のいずれかの結果である中間結果表を指定します。この表は、結合演算子 INNER、LEFT OUTER、RIGHT OUTER、FULL OUTER、LEFT EXCEPTION、RIGHT EXCEPTION、または CROSS の 1 つをそのオペランドに適用することによって得られます。



結合演算子を指定しないと、暗黙に INNER になります。複数の結合が行われる順序は、結果に影響する可能性があります。結合の中で、他の結合をネストすることができます。結合の処理順序は、通常は左から右ですが、必須結合条件 または USING 文節の位置によって決まります。ネストされた結合の順序を読みやすくするために、括弧を使用することをお勧めします。例えば、次のようになります。

```
TB1 LEFT JOIN TB2 ON TB1.C1=TB2.C1
    LEFT JOIN TB3 LEFT JOIN TB4 ON TB3.C1=TB4.C1
           ON TB1.C1=TB3.C1
```

これは、次と同じです。

```
(TB1 LEFT JOIN TB2 ON TB1.C1=TB2.C1)
     LEFT JOIN (TB3 LEFT JOIN TB4 ON TB3.C1=TB4.C1)
          ON TB1.C1=TB3.C1
```

内部結合は、結合条件 (または USING 文節) が真の行だけを保持して、左表の各行を右表の各行と結合します。したがって、結果表からは、結合された表の片方または両方の行が欠落している場合があります。外部結合は、内部結合によって生成された行に加えて、外部結合のタイプに応じた欠落行が含まれています。例外結合には、例外結合のタイプに基づいて、欠落行のみが含まれます。

- 左外部結合の場合は、内部結合から欠落した左の表の行が組み込まれます。
- 右外部結合の場合は、内部結合から欠落した右の表の行が組み込まれます。
- 全外部結合の場合は、内部結合から欠落した両方の表の行が組み込まれます。
- 左除外結合の場合は、内部結合から欠落した左の表の行だけが組み込まれます。
- 右除外結合の場合は、内部結合から欠落した右の表の行だけが組み込まれます。

結合表は、任意の形式の SELECT ステートメントが使用されている、どのような文脈でも使用することができます。ビューまたはカーソルは、その SELECT ステートメントが結合表を含んでいる場合は、読み取り専用です。

結合条件

結合条件 は、次の規則に適合した検索条件 です。

- 結合条件には、限定副照会、副選択を伴う IN 述部、または EXISTS 副照会を含めることはできません。基本述部副照会およびスカラー全選択は含めることができます。
- *join-condition* の式で参照する列は、関連する結合 (同じ joined-table 文節の有効範囲にある結合) のいずれかのオペランド表の列でなければなりません。
- 各列名は、*FROM* 文節 の表の 1 つの列を明確に示すものでなければなりません。
- 集約関数は、式 で使用することはできません。

どのタイプの結合の場合も、まず 163 ページの『列名』で指定された列名修飾子を解決するための規則を適用して、結合条件 の式の中の列参照を解決した後、列がどの表に属するかに関する規則を適用します。

結合 USING

USING 文節は、結合条件を定義する簡便な方法を指定します。この形式は、名前付き列結合 と呼ばれます。

column-name
結合表の両方の表参照 に存在する列を明確に識別する必要があります。列は、DATALINK 列であってはなりません。

結合の結果表には、最初に USING 節からの列が含まれ、次に、USING 節になかった、結合の最初の表からの列、その次に、USING 節になかった、結合の 2 番目の表からの残りの列が含まれます。照会では、USING 文節で指定する列を修飾できません。

USING 文節は、左側の表参照 の中の各列が右側の表参照 の中の同じ名前の列と比較されるという点で、結合条件 と同等です。例えば、TB1 および TB2 に、次の形式の列 C1、C2、... Cn、D1、D2 *named-columns-join* があるとします。

```
TB1 INNER JOIN TB2
    USING (C1, C2, ... Cn)
```

これは、以下と同等の結果表を定義します。

```
SELECT TB1.*, TB2.D1, TB2.D2
FROM TB1 INNER JOIN TB2
    ON TB1.C1 = TB2.C1 AND
       TB1.C2 = TB2.C2 AND
       ...
       TB1.Cn = TB2.Cn
```

結合操作

結合条件 (または USING 文節) は、T1 と T2 の組み合わせを指定します。ここで、T1 と T2 は、結合条件 (または USING 文節) の JOIN 演算子の左右のオペランド表です。 T1 と T2 の行の可能な組み合わせに対して、結合条件 (または USING 文節) が真の場合、T1 の行と T2 の行が結合されます。 T1 の行と T2 の行が結合された場合、結果の行は、T1 のその行の値と T2 のその行の値が連結されたものになります。 OUTER 結合の場合は、実行によって NULL 行が生成されることもあります。列が NULL 値を許すかどうかに関係なく、表の NULL 行は、表の各列の NULL 値から成ります。

INNER JOIN または JOIN

T1 INNER JOIN T2 の結果は、それぞれの対にされた行から構成されます。

join-condition (または USING 文節) で INNER JOIN 構文を使用した場合の結果と、FROM 文節で 2 つの表をコンマで区切って記述し、*where-clause* で結合条件を記述して結合を指定した場合の結果は同じです。

LEFT JOIN または LEFT OUTER JOIN

T1 LEFT OUTER JOIN T2 の結果は、それぞれの対にされた行と、T1 の対にされない各行について、その行と T2 の NULL 行を連結したものから構成されます。T2 から派生した行はすべて NULL 値を許します。

RIGHT JOIN または RIGHT OUTER JOIN

T1 RIGHT OUTER JOIN T2 の結果は、それぞれの対にされた行と、T2 の対にされない各行について、その行と T1 の NULL 行を連結したものから構成されます。T1 から派生した行はすべて NULL 値を許します。

FULL JOIN または FULL OUTER JOIN

T1 FULL OUTER JOIN T2 の結果は、ペアの行、およびペアになっていない T2 の行ごとにその行を T1 の NULL 行に連結したもの、およびペアになっていない T1 の行ごとにその行を T2 の NULL 行に連結したもので構成されます。 T1 および T2 から得られるすべての列には NULL 値を使用することができます。

照会に以下の指定がある場合、FULL OUTER JOIN は使用できません。

- 分散表
- 読み取りトリガーを指定する表
- 複数の物理ファイル・メンバー上に構築された論理ファイル

LEFT EXCEPTION JOIN および EXCEPTION JOIN

T1 LEFT EXCEPTION JOIN T2 の結果は、T1 の対にされない各行について、その行と T2 の NULL 行を連結したものだけから構成されます。T2 から派生した行はすべて NULL 値を許します。

RIGHT EXCEPTION JOIN

T1 RIGHT EXCEPTION JOIN T2 の結果は、T2 の対にされない各行について、その行と T1 の NULL 行を連結したものだけから構成されます。T1 から派生した行はすべて NULL 値を許します。

CROSS JOIN

T1 CROSS JOIN T2 の結果は、T1 の各行が T2 の各行と対にされたものから構成されます。 CROSS JOIN はデカルト積とも呼ばれます。

階層照会

階層照会は再帰的照会の 1 つの形式であり、CONNECT BY 節を使用して、リレーショナル・データから階層 (例えば、部品表など) を取り出すことをサポートします。

CONNECT BY 再帰は、シード (開始) と再帰ステップ (接続) に同じ副照会を使用します。この組み合わせによって、部品表、報告先チェーン、E メール・スレッドなど、再帰を簡潔に表す方法が提供されます。

CONNECT BY 再帰は、循環が発生するとエラーを戻します。 循環は、直接的または間接的に自身を作成する行があると発生します。オプションの CONNECT BY NOCYCLE 節を使用すると、重複行を無視するよう再帰に指示できるため、循環とエラーの両方を回避できます。

副選択

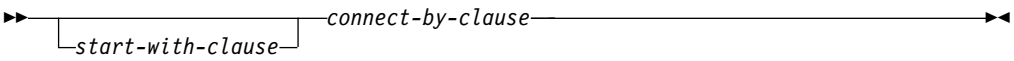
階層照会のサポートには、副選択に関する以下の拡張機能が含まれています。

- 副選択は階層照会文節 を含みます。
- 副選択の文節は、次の順序で処理されます。
  1. FROM 文節
  2. 階層照会文節
  3. WHERE 文節
  4. GROUP-BY 文節
  5. HAVING 文節
  6. SELECT 文節
  7. ORDER BY 文節
  8. OFFSET 文節
  9. FETCH 節
- 副選択に階層照会文節 が含まれている場合、WHERE 文節の述部を処理する順序に関して特別な規則が適用されます。検索条件 は、AND 条件 (結合) を伴う述部に分解されます。述部が暗黙的結合述部である (つまり、FROM 文節の複数の表を参照する) 場合、その述部は階層照会文節 の前に適用されます。 FROM 文節の最大 1 つの表を参照する述部は、階層照会文節 の中間結果表に適用されます。

  結合が関係する階層照会は、WHERE 文節の述部の適用についての混乱を回避するため、ON 文節で明示的な結合表を使用して記述するようにしてください。
- ORDER SIBLINGS BY 文節は、副選択に階層照会文節 が含まれている場合に指定できます。この節は、階層内の兄弟にのみ順序が適用されることを指定します。

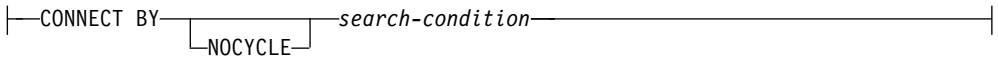
階層照会文節

階層照会文節 を含んでいる副選択は、階層照会と呼ばれます。



start-with-clause:

connect-by-clause:



最初の中間結果表 H_1 の生成の後、後続の中間結果表 H_2、 H_3 (以下同様) が、$H_{n+}1$ を生成するための結合条件として *CONNECT BY* 節 を使用して H_n を R と結合することによって生成されます。R は、副選択の FROM 節、および WHERE 節の結合述部の結果です。このプロセスは、H_{n+1} が空の結果表を生成したとき、または、サポートされている最大の深さレベル 250 に達したときに停止します。階層照会文節 の結果表 H は、すべての H_i の UNION ALL です。

start-with-clause
検索条件 を満たす R の行からなる、階層照会の中間結果表 H_1 を指定します。 *start-with-clause* が指定されない場合、H_1 は中間結果表 R 全体です。

connect-by-clause
検索条件 を使用して H_n を R と結合することによって、中間結果表 H_n から H_{n+1} を生成します。

中間結果表 H_{n+1} が、ある階層パスに関して R からの行を戻そうとして、それが既にその階層パスにある R からの行と同じである場合、エラーが戻されます。

NOCYCLE
エラーは戻されないが、繰り返される行は中間結果表 H_{n+1} に組み込まれないことを指定します。

start-with-clause と *connect-by-clause* の *search-condition* に関する規則は、*where-clause* の場合と同じです。

直前の再帰ステップ H_n への列参照を R への列参照と区別するために、単項演算子 PRIOR が使用されます。以下に例を示します。

```
CONNECT BY MGRID = PRIOR EMPID
```

MGRID は R で解決され、EMPID は前の中間結果表 H_n 内で解決されます。

階層照会である副選択は、中間結果セットを部分順序で戻します。ただし、その順序が、明示的な ORDER BY 節、GROUP BY または HAVING 節、または選択リスト中の DISTINCT キーワードの使用によって壊されない場合に限ります。部分順序は、ある特定の階層について H_{n+1} 内に生成される行が、それらを生成した H_n

の行の直後になるように行を戻します。ORDER SIBLINGS BY 節を使用すると、同じ親から生成される行セット内で順序を強制的に決めることができます。

NEXT VALUE 式を以下の場所に指定することはできません。

- CONNECT_BY_ROOT 演算子または SYS_CONNECT_BY_PATH 関数のパラメーター・リスト
- START WITH 節または CONNECT BY 節

例

- 以下の報告先チェーンの例は、CONNECT BY 再帰を示します。この例は、MY_EMP という名前の表に基づいています。この表は、次のように作成され、データ設定されています。

```
CREATE TABLE MY_EMP(
  EMPID  INTEGER NOT NULL PRIMARY KEY,
  NAME   VARCHAR(10),
  SALARY DECIMAL(9, 2),
  MGRID  INTEGER);

INSERT INTO MY_EMP VALUES ( 1, 'Jones',    30000, 10);
INSERT INTO MY_EMP VALUES ( 2, 'Hall',     35000, 10);
INSERT INTO MY_EMP VALUES ( 3, 'Kim',      40000, 10);
INSERT INTO MY_EMP VALUES ( 4, 'Lindsay',  38000, 10);
INSERT INTO MY_EMP VALUES ( 5, 'McKeough', 42000, 11);
INSERT INTO MY_EMP VALUES ( 6, 'Barnes',   41000, 11);
INSERT INTO MY_EMP VALUES ( 7, 'O''Neil',  36000, 12);
INSERT INTO MY_EMP VALUES ( 8, 'Smith',    34000, 12);
INSERT INTO MY_EMP VALUES ( 9, 'Shoeman',  33000, 12);
INSERT INTO MY_EMP VALUES (10, 'Monroe',   50000, 15);
INSERT INTO MY_EMP VALUES (11, 'Zander',   52000, 16);
INSERT INTO MY_EMP VALUES (12, 'Henry',    51000, 16);
INSERT INTO MY_EMP VALUES (13, 'Aaron',    54000, 15);
INSERT INTO MY_EMP VALUES (14, 'Scott',    53000, 16);
INSERT INTO MY_EMP VALUES (15, 'Mills',    70000, 17);
INSERT INTO MY_EMP VALUES (16, 'Goyal',    80000, 17);
INSERT INTO MY_EMP VALUES (17, 'Urbassek', 95000, NULL);
```

以下の照会は、Goyal の下で働いているすべての従業員と、報告先チェーンなどの追加情報を戻します。

```
 1 SELECT NAME,
 2        LEVEL,
 3        SALARY,
 4        CONNECT_BY_ROOT NAME AS ROOT,
 5        SUBSTR(SYS_CONNECT_BY_PATH(NAME, ':'), 1, 25) AS CHAIN
 6   FROM MY_EMP
 7   START WITH NAME = 'Goyal'
 8   CONNECT BY PRIOR EMPID = MGRID
 9   ORDER SIBLINGS BY SALARY;
NAME       LEVEL       SALARY      ROOT  CHAIN
---------- ----------- ----------- ----- --------------
Goyal                1    80000.00 Goyal :Goyal
Henry                2    51000.00 Goyal :Goyal:Henry
Shoeman              3    33000.00 Goyal :Goyal:Henry:Shoeman
Smith                3    34000.00 Goyal :Goyal:Henry:Smith
O'Neil               3    36000.00 Goyal :Goyal:Henry:O'Neil
Zander               2    52000.00 Goyal :Goyal:Zander
Barnes               3    41000.00 Goyal :Goyal:Zander:Barnes
McKeough             3    42000.00 Goyal :Goyal:Zander:McKeough
Scott                2    53000.00 Goyal :Goyal:Scott
```

7 行目と 8 行目が、再帰の中核です。オプションの START WITH 節は、再帰のシードとするためにソース表で使用される WHERE 節を記述します。この例では、従業員 Goyal の行のみが選択されます。START WITH 節が省略されると、ソース表全体が再帰のシードに使用されます。CONNECT BY 節は、既存の行から、次の行セットをどのように見つけるのかを記述します。単項演算子 PRIOR は、前のステップでの値と現在のステップでの値を区別するために使用されます。PRIOR は、前の再帰ステップの従業員 ID として EMPID を指定し、現在の再帰ステップから発生するものとして MGRID を指定しています。

2 行目の LEVEL は、再帰の現在のレベルを記述する疑似列です。

CONNECT_BY_ROOT は、最初の再帰ステップ時の引数の値 (つまり、明示的または暗黙的な START WITH 節で戻される値) を常に戻す単項演算子です。

SYS_CONNECT_BY_PATH() は、2 項関数であり、 2 番目の引数を最初の引数の前に付加し、その結果を、前の再帰ステップで生成した値に付加します。

明示的にオーバーライドされない限り、 CONNECT BY 再帰は、部分順序で結果セットを戻します。つまり、 1 つの再帰ステップで生成された行は、常にそれらの行を生成した行の後になります。再帰の同じレベルにある兄弟には特定の順序はありません。9 行目の ORDER SIBLINGS BY 節は、こういった兄弟の順序を定義していて、部分順序をさらに詳細化して、可能であれば全体順序にします。

- DEPARTMENT 表の組織構造を戻します。部門のレベルを使用して階層を視覚化します。

```
SELECT LEVEL, CAST(SPACE((LEVEL - 1) * 4) || '/' || DEPTNAME
    AS VARCHAR(40)) AS DEPTNAME
  FROM DEPARTMENT
  START WITH DEPTNO = 'A00'
  CONNECT BY NOCYCLE PRIOR DEPTNO = ADMRDEPT
```

照会が、以下を戻します。

```
LEVEL       DEPTNAME
----------- ----------------------------------------
          1 /SPIFFY COMPUTER SERVICE DIV.
          2     /PLANNING
          2     /INFORMATION CENTER
          2     /DEVELOPMENT CENTER
          3         /MANUFACTURING SYSTEMS
          3         /ADMINISTRATION SYSTEMS
          2     /SUPPORT SERVICES
          3         /OPERATIONS
          3         /SOFTWARE SUPPORT
          3         /BRANCH OFFICE F2
          3         /BRANCH OFFICE G2
          3         /BRANCH OFFICE H2
          3         /BRANCH OFFICE I2
          3         /BRANCH OFFICE J2
```

疑似列

疑似列とは、列および変数と同じ名前空間を共有する識別子です。列についての通常の名前解決が失敗した後、 Db2 は非修飾識別子に一致する疑似列名を探します。

CONNECT_BY_ISCYCLE

CONNECT_BY_ISCYCLE は、階層照会で使用するための疑似列です。この列は、行が階層内の循環の一部である場合、つまり、CONNECT BY 節の検索条件 で、行がそれ自体の直接的または間接的な祖先となる場合、値 1 を戻します。行が循環の一部ではない場合、この列は値 0 を戻します。

この列のデータ・タイプは SMALLINT であり、NULL にはできません。

CONNECT_BY_ISCYCLE は階層照会のコンテキストで指定されなければなりませんが、 START WITH 節または CONNECT BY 節に指定することはできず、また、CONNECT_BY_ROOT 演算子の引数として、あるいは SYS_CONNECT_BY_PATH 関数の引数として指定することもできません。

CONNECT_BY_ISLEAF

CONNECT_BY_ISLEAF は、階層照会で使用するための疑似列です。この列は、CONNECT BY 節での定義で、行が階層内のリーフである場合、値 1 を戻します。行がリーフでない場合、この列は値 0 を戻します。

この列のデータ・タイプは SMALLINT であり、NULL にはできません。

CONNECT_BY_ISLEAF は階層照会のコンテキストで指定されなければなりませんが、 START WITH 節または CONNECT BY 節に指定することはできず、また、CONNECT_BY_ROOT 演算子の引数として、あるいは SYS_CONNECT_BY_PATH 関数の引数として指定することもできません。

LEVEL

LEVEL は、階層照会で使用するための疑似列です。この列は、行が生成された、階層内の再帰的ステップを戻します。START WITH 節によって生成されるすべての行は値 1 を戻します。 CONNECT BY 節の最初の反復を適用することによって生成される行は値 2 を戻し、以下同様になります。

この列のデータ・タイプは INTEGER であり、NULL にはできません。

LEVEL は階層照会のコンテキストで指定されなければなりませんが、 START WITH 節に指定することはできず、また、CONNECT_BY_ROOT 演算子の引数として、あるいは SYS_CONNECT_BY_PATH 関数の引数として指定することもできません。

CONNECT_BY_ROOT

CONNECT_BY_ROOT 単項演算子は、階層照会でのみ使用できます。この演算子は、階層内の各行について、行のルート祖先の式を戻します。

expression
NEXT VALUE 式、階層照会構造 (LEVEL 疑似列など)、SYS_CONNECT_BY_PATH 関数、および OLAP 仕様を含まない式。

この演算子の結果のデータ・タイプおよび長さは、式の結果のデータ・タイプおよび長さと同じです。

CONNECT_BY_ROOT 演算子を、階層照会の START WITH 節または CONNECT BY 節に指定することはできません。これを SYS_CONNECT_BY_PATH 関数の引数として指定することはできません。

CONNECT_BY_ROOT 演算子は、どの 2 項演算子よりも優先されます。したがって、2 項演算子 (+ や || など) を使用する式を引数として渡すには、括弧を使用する必要があります。例を以下に示します。

```
CONNECT_BY_ROOT FIRSTNME || LASTNAME
```

は、以下と等価であるため、ルート祖先行の FIRSTNME 値に、階層内の実際の行の LASTNAME 値が連結されたものを戻します。

```
(CONNECT_BY_ROOT FIRSTNME) || LASTNAME
```

次のものとは等価ではありません。

```
CONNECT_BY_ROOT (FIRSTNME || LASTNAME)
```

例

- DEPARTMENT 表内の、部門の階層およびそれらのルート部門を戻します。

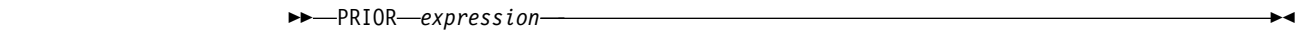
```
SELECT CONNECT_BY_ROOT DEPTNAME AS ROOT, DEPTNAME
     FROM DEPARTMENT
     START WITH DEPTNO IN ('B01','C01','D01','E01')
     CONNECT BY PRIOR DEPTNO = ADMRDEPT
```

この照会は次の値を返します。

```
ROOT                DEPTNAME
------------------- ----------------------
PLANNING            PLANNING
INFORMATION CENTER  INFORMATION CENTER
DEVELOPMENT CENTER  DEVELOPMENT CENTER
DEVELOPMENT CENTER  MANUFACTURING SYSTEMS
DEVELOPMENT CENTER  ADMINISTRATION SYSTEMS
SUPPORT SERVICES    SUPPORT SERVICES
SUPPORT SERVICES    OPERATIONS
SUPPORT SERVICES    SOFTWARE SUPPORT
SUPPORT SERVICES    BRANCH OFFICE F2
SUPPORT SERVICES    BRANCH OFFICE G2
SUPPORT SERVICES    BRANCH OFFICE H2
SUPPORT SERVICES    BRANCH OFFICE I2
SUPPORT SERVICES    BRANCH OFFICE J2
```

PRIOR

PRIOR 単項演算子は、階層照会の CONNECT BY 節でのみ使用できます。



CONNECT BY 節は、階層照会の中間結果表 H_n と FROM 節に指定されたソース結果表との間で内部結合を実行します。FROM 節で参照されている表に対する列参照で、 PRIOR 演算子の引数になっている列参照はすべて、H_n が対象になると見なされます。

中間結果表 H_n の主キーをソース結果表の外部キーに結合することによって階層を再帰的に全探索するというのが典型的な使用法です。

```
CONNECT BY PRIOR T.PK = T.FK
```

主キーが複合キーである場合、各列の前に PRIOR を付けるよう注意が必要です。

```
CONNECT BY PRIOR T.PK1 = T.FK1 AND PRIOR T.PK2 = T.FK2
```

expression
NEXT VALUE 式、階層照会構造 (LEVEL 疑似列など)、SYS_CONNECT_BY_PATH 関数、OLAP 仕様を含まない式。

この演算子の結果のデータ・タイプおよび長さは、式の結果のデータ・タイプおよび長さと同じです。

PRIOR 演算子は、どの 2 項演算子よりも優先されます。したがって、2 項演算子 (+ や || など) を使用する式を引数として渡すには、括弧を使用する必要があります。例えば、

```
PRIOR FIRSTNME || LASTNAME
```

この式は、以下のものと等価であるため、前の行の FIRSTNME 値を階層内の実際の行の LASTNAME 値と連結したものを戻します。

```
(PRIOR FIRSTNME) || LASTNAME
```

次のものとは等価ではありません。

```
PRIOR (FIRSTNME || LASTNAME)
```

例

- DEPARTMENT 表の部門の階層を戻します。

```
SELECT LEVEL, DEPTNAME
     FROM DEPARTMENT
     START WITH DEPTNO = 'A00'
     CONNECT BY NOCYCLE PRIOR DEPTNO = ADMRDEPT
```

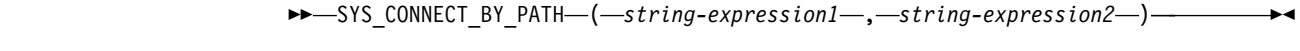
この照会は次の値を返します。

```
LEVEL       DEPTNAME
----------- ----------------------------------------
          1 SPIFFY COMPUTER SERVICE DIV.
          2 PLANNING
          2 INFORMATION CENTER
          2 DEVELOPMENT CENTER
          3 MANUFACTURING SYSTEMS
          3 ADMINISTRATION SYSTEMS
```

```
2 SUPPORT SERVICES
3 OPERATIONS
3 SOFTWARE SUPPORT
3 BRANCH OFFICE F2
3 BRANCH OFFICE G2
3 BRANCH OFFICE H2
3 BRANCH OFFICE I2
3 BRANCH OFFICE J2
```

SYS_CONNECT_BY_PATH

SYS_CONNECT_BY_PATH 関数は、階層照会で使用され、ルート行からこの行までのパスを表すストリングを構築します。



LEVEL *n* にある行のストリングは次のように構築されます。

- ステップ 1 (最初の中間結果表 H_1 からルート行の値を使用):

  `path1 := string-expression2 || string-expression1`

- ステップ n (中間結果表 H_n からの行に基づく):

  $path_n$:= $path_{n-1}$ || string-expression2 || string-expression1

string-expression1
行を示す文字ストリング式。組み込み文字ストリング・データ・タイプでなければなりません。この式には、シーケンスの NEXT VALUE 式、階層照会構造体 (LEVEL 疑似列や CONNECT_BY_ROOT 演算子など)、 OLAP 仕様、または集約関数 が含まれていてはなりません。

string-expression2
区切り文字として機能する文字ストリング式。組み込み文字ストリング・データ・タイプでなければなりません。この式には、シーケンスの NEXT VALUE 式、階層照会構造体 (LEVEL 疑似列や CONNECT_BY_ROOT 演算子など)、OLAP 仕様、または集約関数 が含まれていてはなりません。

結果は、CLOB(1M) です。

SYS_CONNECT_BY_PATH 関数を階層照会のコンテキスト以外で使用してはなりません。これを START WITH 節または CONNECT BY 節で使用することはできません。

例

- DEPARTMENT 表の部門の階層を戻します。

```
SELECT CAST(SYS_CONNECT_BY_PATH(DEPTNAME, '/')
                AS VARCHAR(76)) AS ORG
       FROM DEPARTMENT
       START WITH DEPTNO = 'A00'
       CONNECT BY NOCYCLE PRIOR DEPTNO = ADMRDEPT
```

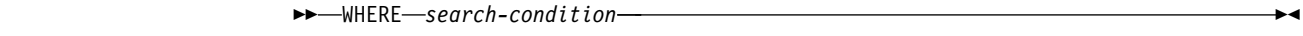
この照会は次の値を返します。

```
ORG
------------------------------------------------------------------------
/SPIFFY COMPUTER SERVICE DIV.
/SPIFFY COMPUTER SERVICE DIV./PLANNING
/SPIFFY COMPUTER SERVICE DIV./INFORMATION CENTER
/SPIFFY COMPUTER SERVICE DIV./DEVELOPMENT CENTER
/SPIFFY COMPUTER SERVICE DIV./DEVELOPMENT CENTER/MANUFACTURING SYSTEMS
/SPIFFY COMPUTER SERVICE DIV./DEVELOPMENT CENTER/ADMINISTRATION SYSTEMS
/SPIFFY COMPUTER SERVICE DIV./SUPPORT SERVICES
/SPIFFY COMPUTER SERVICE DIV./SUPPORT SERVICES/OPERATIONS
/SPIFFY COMPUTER SERVICE DIV./SUPPORT SERVICES/SOFTWARE SUPPORT
/SPIFFY COMPUTER SERVICE DIV./SUPPORT SERVICES/BRANCH OFFICE F2
/SPIFFY COMPUTER SERVICE DIV./SUPPORT SERVICES/BRANCH OFFICE G2
```

```
/SPIFFY COMPUTER SERVICE DIV./SUPPORT SERVICES/BRANCH OFFICE H2
/SPIFFY COMPUTER SERVICE DIV./SUPPORT SERVICES/BRANCH OFFICE I2
/SPIFFY COMPUTER SERVICE DIV./SUPPORT SERVICES/BRANCH OFFICE J2
```

WHERE 文節

WHERE 節は、 *search-condition* (検索条件) が真である R の行で構成される中間結果表を指定します。 R は、ステートメントの FROM 文節の結果です。



search-condition は、以下の規則に適合していなければなりません。

- 各 列名 は、R の列をあいまいなところなく指定するか、あるいは相関参照でなければなりません。 *column-name* が外側の全選択で示される表、ビュー、*common-table-expression*、または *nested-table-expression* の列を識別する場合、その列名は相関参照になります。
- HAVING 文節の副照会に WHERE 文節が指定され、その関数の引数がグループに対する相関参照である場合を除いて、集約関数を指定することはできません。

検索条件 における副照会は、R の各行について有効に実行され、その結果は、R の所定の行に対する検索条件 の適用で使用されます。副照会が R の各行ごとに実行されるのは、副照会が R の列に対する相関参照を含んでいる場合です。通常、相関参照のない副照会は 1 回しか実行されません。

表に対して行アクセス制御が有効で、その他の行の許可は定義されていない場合、行アクセス制御検索条件はデフォルトの行の許可 (1 = 0) です。行の許可が 1 つのみ定義されている場合、行アクセス制御検索条件はその許可で指定されている検索条件です。または、表に対して複数の行の許可が定義されている場合、行アクセス制御検索条件は、各行許可に対して指定されている検索条件に対して論理 OR 演算子を適用することにより派生します。この行アクセス制御検索条件は、全体として、WHERE 節で指定された検索条件に論理 AND 演算子を適用することにより接続され、WHERE 節のその他の検索条件と同じ優先順位を持ちます。この処理は、行アクセス制御が有効である副選択の FROM 節の各 *table-reference* に対して繰り返されます。

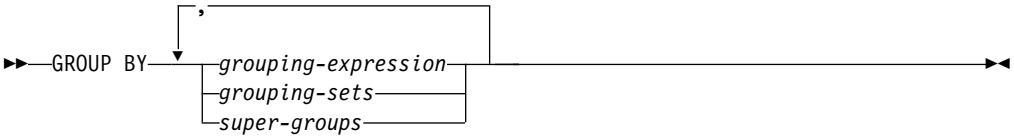
行アクセス制御検索条件は、副選択の権限 ID がアクセスできる *table-reference* の結果を決定するための、*table-reference* へのフィルターとして機能します。同じ優先順位の演算子の場合、演算子が評価される順序は定義されていないため、WHERE 節のその他の検索条件が行アクセス制御検索条件より前に評価される可能性があります。したがって、その他の検索条件が、行アクセス規則で制限されている行にアクセスできます。機密データが保護されるようにするために、NOT SECURED オプションで定義されているユーザー定義関数を参照する述部は常に行アクセス制御検索条件の後で評価されます。

列アクセス制御は WHERE 節の操作に影響しません。

WHERE 文節を含むステートメントの実行時に *HEX 以外の照合順序が有効で、しかもその検索条件 に SBCS データ、混合データまたは、 Unicode データのオペランドが含まれている場合、それらの述部の比較は、重み付けされた値を使用して行われます。重み付けされた値は、述部のオペランドに対して照合順序を適用することにより得られます。

GROUP-BY 文節

GROUP BY 文節は、R のグループ化された行で構成される中間結果表を指定します。R は、副選択の直前の文節の結果です。



最も単純な形式では、 GROUP BY 文節に *grouping-expression* (グループ化式) が入っています。グループ化式 は、R のグループ化を定義する式です。以下の制約事項は、グループ化式 に当てはまります。

- グループ化式 に含まれる各列名は、R の列を明瞭に識別するものでなければなりません。
- *grouping-expression* の結果が、DataLink データ・タイプまたは XML データ・タイプ、あるいは DataLink または XML に基づく特殊タイプになることはあり得ません。
- グループ化式に以下の項目を含めることはできません。
  - 相関列
  - 変数
  - 集約関数
  - 非決定的な関数

複雑な形式の GROUP BY 節には、 *grouping-sets* (グループ化集合) および *super-groups* (スーパー・グループ) が組み入れられます。これらの形式の説明については、それぞれ 819 ページの『grouping-sets』 と 820 ページの『スーパー・グループ』 を参照してください。

GROUP BY 文節の結果は、行のグループの集まりになります。複数行を持つグループそれぞれにおいて、各グループ化式 の値はすべて等しく、グループ化式 の値の集合が同じ行は、すべて同じグループに入ります。グループ化では、グループ化式 の NULL 値はすべて等しいものと見なされます。

グループのすべての行にはグループ化式 の同じ値が含まれるので、グループ化式は、HAVING 文節、SELECT 文節、または ORDER BY 文節のソート・キー式 (詳細は 833 ページの『ORDER BY 文節』を参照) の検索条件で使用することができます。どの場合も、参照は各グループの 1 つの値だけを指定します。これらの文節に指定されるグループ化式 は、GROUP BY 文節の中のグループ化式 と厳密に一致する必要があります。ただし、ブランクは意味を持ちません。例えば、次のグループ化式 は、

```
SALARY*.10
```

次の *HAVING* 文節 の式に一致します。

```
HAVING SALARY*.10
```

しかし、次とは一致しません。

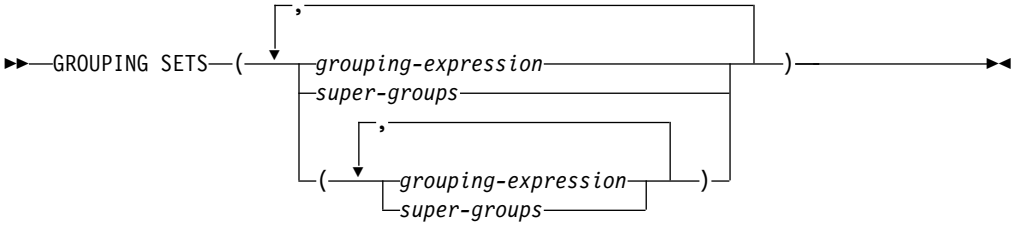
```
HAVING .10 *SALARY
 あるいは
HAVING (SALARY*.10)+100
```

グループ化式 に末尾のブランクの付いた可変長ストリングが入っている場合、そのグループの値は末尾のブランクの数が異なる場合があり、必ずしも同じ長さになるとはかぎりません。そのような場合でも、グループ化式 への参照は各グループに 1 つの値だけを指定しますが、グループの値は使用可能な値の集合の中から任意に選択されます。したがって、選択された値の実際の長さは、予期できないものになります。

列アクセス制御は GROUP BY 節の操作に影響しません。グループ化は元の列値を使用して実行されます。

GROUP BY 文節を含むステートメントの実行時に *HEX 以外の照合順序が有効な場合で、しかもグループ化式 が SBCS データ、混合データ、または Unicode データの場合は、行は重み付けされた値を使用してグループ化されます。この重み付けされた値は、グループ化式 に該当の照合順序を適用することにより得られます。そのような場合でも、グループ化式 への参照は各グループに 1 つの値だけを指定しますが、グループの値は使用可能な値の集合の中から任意に選択されます。したがって、結果の実際の長さは、予期できないものになります。

grouping-sets



grouping-sets (グループ化集合) の指定により、複数のグループ化節を単一ステートメントで指定することができます。これは、行の複数のグループを単一の結果セットにまとめたものと見なすことができます。これは、1 つのグループ化集合に対応する各副選択ごとに GROUP BY 節を持つ複数の副選択を合併したものと論理的に等しくなります。グループ化集合は、単一のエレメント、もしくは括弧によって区切られる複数のエレメントのリストになります。この場合、エレメントはグループ化式 またはスーパー・グループ のいずれかです。グループ化集合を使用すると、基本表の単一パスを使用してグループを計算することができます。

グループ化集合 の指定により、単純なグループ化式 を使用するか、またはより複雑な形式のスーパー・グループ を使用することができます。スーパー・グループについては、 820 ページの『スーパー・グループ』 を参照してください。

グループ化集合は、GROUP BY 演算の基礎的な構築ブロックであることに注意してください。単純な列を使用する単純な GROUP BY は、 1 つのエレメントを持つグループ化集合と見なすことができます。例えば、次の例があります。

```
GROUP BY a
```

これは、次と同じです。

```
GROUP BY GROUPING SETS( (a) )
```

さらに

```
GROUP BY a, b, c
```

これは、次と同じです。

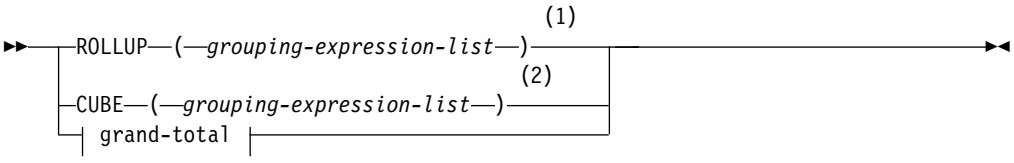
```
GROUP BY GROUPING SETS( (a,b,c) )
```

グループ化集合から除外される副選択の選択リストの非集約の列は、そのグループ化集合用に生成される各行の列に NULL 値を戻します。これは、集約が列の値を考慮せずに行われたことを表しています。

照会内の直前の文節にある表参照 で、分散表、または読み取りトリガーが設定されている表が指定されている場合は、*grouping-sets* を指定できません。

例 C2 から例 C7 は、グループ化集合の使用法を示しています。

スーパー・グループ

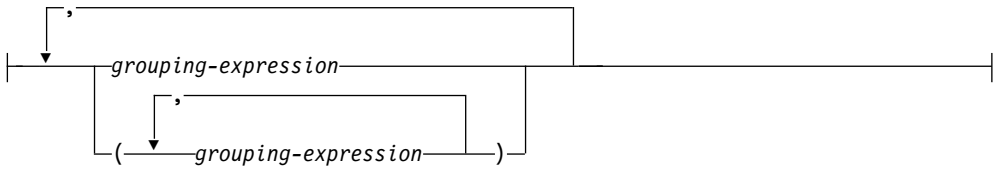


注:

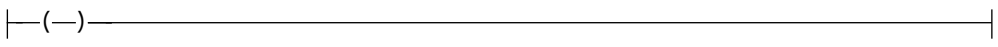
1 *group-by-clause* で単独で使用される場合の代替仕様は、*grouping-expression-list* WITH ROLLUP です。

2 *group-by-clause* で単独で使用される場合の代替仕様は、*grouping-expression-list* WITH CUBE です。

grouping-expression-list:



grand-total:



ROLLUP (grouping-expression-list)

ROLLUP grouping (ROLLUP グループ化) は GROUP BY 節の拡張であり、「通常の」グループ化行に加えて小計 行の入った結果セットを生成します。小計 行は、グループ化行を入手するのに使用されたのと同じ集約関数を適用して値が得られる集合体が入っている「スーパー集約」行です。これらの行は小計が

最も一般的な用途なので小計行と呼ばれますが、集約には任意の集約関数を使用することができます。例えば、例 C8 では MAX および AVG が使用されます。

ROLLUP グループ化は、一連のグループ化集合 です。 *n* 個のエレメントを持つ ROLLUP の一般的な仕様は、次のとおりです。

```
GROUP BY ROLLUP( C1,C2,...,Cn-1,Cn)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS( (C1,C2,...,Cn-1,Cn),
                        (C1,C2,...,Cn-1),
                        ...
                        (C1,C2),
                        (C1),
                        () )
```

ROLLUP の *n* のエレメントは、 *n* +1 のグループ化集合に変換される点に注意してください。グループ化式 が指定されている順序が、 ROLLUP にとって重要である点も注意してください。例えば、次の例があります。

```
GROUP BY ROLLUP (a,b)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS ( (a,b),
                         (a),
                         () )
```

一方、

```
GROUP BY ROLLUP (b,a)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS ( (b,a),
                         (b),
                         () )
```

ORDER BY 節は、結果セットの行の ORDER BY を確保する唯一の方法です。
例 C3 は ROLLUP の使用法を示しています。

CUBE(*grouping-expression-list***)**

CUBE grouping (CUBE グループ化) は GROUP BY 節の拡張であり、すべての ROLLUP 集約行に加えて、「クロス集計」行の入った結果セットを生成します。クロス集計 行は、小計による集約の一部ではない、追加の「スーパー集約」行です。 グループ化式リスト 中に指定できる式は 10 個のみです。

ROLLUP と同じように、CUBE グループ化も、一連のグループ化集合 と見なすことができます。 CUBE の場合、グループ化式リスト のすべての順列が総計とともに計算されます。したがって、CUBE の *n* 個のエレメントは、 2**n (2 の *n* 乗) のグループ化集合に変換されます。例えば、次のように指定するとします。

```
GROUP BY CUBE (a,b,c)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS ( (a,b,c),
                         (a,b),
                         (a,c),
                         (b,c),
                         (a),
                         (b),
                         (c),
                         () )
```

CUBE の 3 個のエレメントは、8 個のグループ化集合に変換されることに注意してください。

エレメントの指定の順序は、CUBE の場合、重要ではありません。 'CUBE (DayOfYear, Sales_Person)' と 'CUBE (Sales_Person, DayOfYear) は同じ結果セットを生成します。「同じ」とは、結果セットの順序ではなく、結果セットの内容を指します。 ORDER BY 節は、結果セットの行の ORDER BY を確保する唯一の方法です。
例 C4 に、CUBE の使用例が示されています。

grouping-expression-list

grouping-expression-list (グループ化式リスト) は、 CUBE または ROLLUP 演算のエレメント数を定義するために、 CUBE または ROLLUP グループ化で使用されます。これは、複数のグループ化式 を持つエレメントを区切るために括弧を使用して制御されます。

グループ化式 の規則については、 818 ページの『GROUP-BY 文節』 で説明しています。例えば、照会が、County ではなく Province 内の City の ROLLUP について合計費用を戻すものと想定します。しかし、以下の文節の場合、

```
GROUP BY ROLLUP (Province, County, City)
```

County についての不要な小計行が生じます。 以下の節では、

```
GROUP BY ROLLUP (Province, (County, City))
```

複合 (County, City) が ROLLUP の 1 つのエレメントを形成するので、この文節を使用する照会は、必要な結果を生じます。つまり、次のように 2 つのエレメントからなる ROLLUP の場合、

```
GROUP BY ROLLUP (Province, (County, City))
```

以下を生成します。

```
GROUP BY GROUPING SETS ( (Province, County, City)
                         (Province)
                         () )
```

一方、3 つのエレメントからなる ROLLUP の場合は、次のようになります。

```
GROUP BY GROUPING SETS ( (Province, County, City),
                         (Province, County),
                         (Province),
                         () )
```

例 2 でも、複合列値が使用されています。

grand-total

CUBE および ROLLUP は、全体の集約 (総計) である行を戻します。これは、GROUPING SETS 文節に空の括弧を使用して別々に指定することができます。

また、GROUP BY 節に直接指定することもできます。これは照会の結果には影響しません。例 C4 では、総計構文を使用しています。

グループ化集合の結合

これは、任意のタイプの GROUP BY 節を組み合わせるのに使用することができます。単純なグループ化式 を他のグループと組み合わせる場合、結果のグループ化集合 の始めに「追加」されます。 ROLLUP 式および CUBE 式を組み合わせる場合、それらの式は、残りの式では「乗数」のように動作し、 ROLLUP または CUBE の定義に応じて追加のグループ化集合項目を生成します。

例えば、グループ化式 のエレメントを組み合わせると、次のようになります。

```
GROUP BY a, ROLLUP (b,c)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS( (a,b,c),
                        (a,b),
                        (a) )
```

または、同様に

```
GROUP BY a,b, ROLLUP (c,d)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS( (a,b,c,d),
                        (a,b,c),
                        (a,b) )
```

ROLLUP エレメントを組み合わせると、次のようになります。

```
GROUP BY ROLLUP(a), ROLLUP (b,c)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS( (a,b,c),
                        (a,b),
                        (a),
                        (b,c),
                        (b),
                        () )
```

同様に、

```
GROUP BY ROLLUP(a), CUBE (b,c)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS( (a,b,c),
                        (a,b),
                        (a,c),
                        (a),
                        (b,c),
                        (b),
                        (c),
                        () )
```

CUBE と ROLLUP のエレメントの組み合わせは次のようになります。

```
GROUP BY CUBE(a,b), ROLLUP (c,d)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS( (a,b,c,d),
                        (a,b,c),
                        (a,b),
                        (a,c,d),
                        (a,c),
                        (a),
                        (b,c,d),
                        (b,c),
                        (b),
                        (c,d),
                        (c),
                        () )
```

単純なグループ化式 の場合のように、グループ化集合を組み合わせると、各グループ化集合内で重複したものが除去されます。例えば、次のようになります。

```
GROUP BY a, ROLLUP (a,b)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS( (a,b),
                        (a) )
```

グループ化集合を組み合わせる場合の詳しい例は、完全な CUBE 集約について戻される特定行を除去する結果セットを構成する場合です。

例えば、以下の GROUP BY 節について考えてみます。

```
GROUP BY Region,
         ROLLUP (Sales_Person, WEEK(Sales_Date)),
         CUBE (YEAR(Sales_Date), MONTH(Sales_Date))
```

GROUP BY のすぐ右側にリストされている列はグループ化され、ROLLUP の後の括弧内の列がロールアップされ、 CUBE の後の括弧内の列は 3 乗されます。したがって、上記の節の結果は、YEAR 内の MONTH のキューブが生成されてから、REGION 内の Sales_Person 内の WEEK の集約内でロールアップが行われます。この結果は、Region、Sales_Person、または WEEK(Sales_Date) の総計行にもクロス集計行にもならないため、生成される行は、以下の文節より少なくなります。

```
GROUP BY  ROLLUP (Region, Sales_Person, WEEK(Sales_Date),
                  YEAR(Sales_Date), MONTH(Sales_Date))
```

グループ化集合、CUBE、および ROLLUP の例

例 C1 から C4 の照会では、述部 'WEEK(SALES_DATE) = 13' に基づいて SALES 表の行のサブセットを使用しています。

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SALES_PERSON,
       SALES AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
```

これは次の結果になります。

```
WEEK    DAY_WEEK   SALES_PERSON  UNITS_SOLD
------- ---------- ------------- ------------
     13          6 LUCCHESSI               3
     13          6 LUCCHESSI               1
     13          6 LEE                     2
     13          6 LEE                     2
     13          6 LEE                     3
```

```
     13          6 LEE                    5
     13          6 GOUNOT                 3
     13          6 GOUNOT                 1
     13          6 GOUNOT                 7
     13          7 LUCCHESSI              1
     13          7 LUCCHESSI              2
     13          7 LUCCHESSI              1
     13          7 LEE                    7
     13          7 LEE                    3
     13          7 LEE                    7
     13          7 LEE                    4
     13          7 GOUNOT                 2
     13          7 GOUNOT                18
     13          7 GOUNOT                 1
```

例 C1:

これは、3 つの列に対して基本の GROUP BY を使用している照会です。

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SALES_PERSON,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

これは次のような結果になります。

```
WEEK    DAY_WEEK   SALES_PERSON  UNITS_SOLD
------- ---------- ------------- ------------
     13          6 GOUNOT                11
     13          6 LEE                   12
     13          6 LUCCHESSI              4
     13          7 GOUNOT                21
     13          7 LEE                   21
     13          7 LUCCHESSI              4
```

例 C2:

SALES 表の行の 2 つのグループ化集合に基づいて結果を生成します。

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SALES_PERSON,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY GROUPING SETS( (WEEK(SALES_DATE), SALES_PERSON),
                        (DAYOFWEEK(SALES_DATE), SALES_PERSON) )
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

これは次のような結果になります。

```
WEEK    DAY_WEEK   SALES_PERSON  UNITS_SOLD
------- ---------- ------------- ------------
     13          - GOUNOT                32
     13          - LEE                   33
     13          - LUCCHESSI              8
      -          6 GOUNOT                11
      -          6 LEE                   12
      -          6 LUCCHESSI              4
      -          7 GOUNOT                21
      -          7 LEE                   21
      -          7 LUCCHESSI              4
```

WEEK 13 の行は、最初のグループ化集合のものであり、それ以外の行は 2 番目のグループ化集合のものです。

例 C3:

例 C2 のグループ化集合に含まれる 3 つの特殊な列を使用して、 ROLLUP を実行する場合、(WEEK、DAY_WEEK、SALES_PERSON)、(WEEK、DAY_WEEK)、(WEEK) および総計のグループ化集合を見ることができます。

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SALES_PERSON,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY ROLLUP( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON )
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

これは次のような結果になります。

```
WEEK    DAY_WEEK   SALES_PERSON  UNITS_SOLD
------- ---------- ------------- ------------
     13          6 GOUNOT                11
     13          6 LEE                   12
     13          6 LUCCHESSI              4
     13          6 -                     27
     13          7 GOUNOT                21
     13          7 LEE                   21
     13          7 LUCCHESSI              4
     13          7 -                     46
     13          - -                     73
      -          - -                     73
```

例 C4:

例 C3 と同じ照会を実行して ROLLUP を CUBE に置き換える場合、結果に (WEEK、SALES_PERSON)、(DAY_WEEK、SALES_PERSON)、(DAY_WEEK)、(SALES_PERSON) の追加のグループ化集合を見ることができます。

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SALES_PERSON,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY CUBE( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON )
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

これは次のような結果になります。

```
WEEK    DAY_WEEK   SALES_PERSON  UNITS_SOLD
------- ---------- ------------- ------------
     13          6 GOUNOT                11
     13          6 LEE                   12
     13          6 LUCCHESSI              4
     13          6 -                     27
     13          7 GOUNOT                21
     13          7 LEE                   21
     13          7 LUCCHESSI              4
     13          7 -                     46
     13          - GOUNOT                32
     13          - LEE                   33
     13          - LUCCHESSI              8
     13          - -                     73
```

```
-          6 GOUNOT                 11
-          6 LEE                    12
-          6 LUCCHESSI               4
-          6 -                      27
-          7 GOUNOT                 21
-          7 LEE                    21
-          7 LUCCHESSI               4
-          7 -                      46
-          - GOUNOT                 32
-          - LEE                    33
-          - LUCCHESSI               8
-          - -                      73
```

例 C5:

SALES 表から選択された行の総計と、SALES_PERSON および MONTH によって集計された行のグループを含む結果セットを入手します。

```
SELECT SALES_PERSON,
       MONTH(SALES_DATE) AS MONTH,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY GROUPING SETS( (SALES_PERSON, MONTH(SALES_DATE)),
                        () )
ORDER BY SALES_PERSON, MONTH
```

これは次のような結果になります。

```
SALES_PERSON  MONTH   UNITS_SOLD
------------- ------- ------------
GOUNOT              3           35
GOUNOT              4           14
GOUNOT             12            1
LEE                 3           60
LEE                 4           25
LEE                12            6
LUCCHESSI           3            9
LUCCHESSI           4            4
LUCCHESSI          12            1
-                   -          155
```

例 C6:

この例では、2 つの単純な ROLLUP 照会を示し、その後に、2 つの ROLLUP を単一結果セットのグループ化集合として扱い、グループ化集合に含まれている列ごとに行の順序を指定する照会を示しています。

例 C6-1:

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY ROLLUP( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE) )
ORDER BY WEEK, DAY_WEEK
```

これは次のような結果になります。

```
WEEK     DAY_WEEK   UNITS_SOLD
-------- ---------- ------------
      13          6           27
      13          7           46
      13          -           73
      14          1           31
      14          2           43
```

```
      14          -           74
      53          1            8
      53          -            8
       -          -          155
```

例 *C6-2:*

```
SELECT MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY ROLLUP( MONTH(SALES_DATE), REGION )
ORDER BY MONTH, REGION
```

これは次のような結果になります。

```
MONTH    REGION          UNITS_SOLD
-------- --------------  ------------
       3 Manitoba                 22
       3 Ontario-North             8
       3 Ontario-South            34
       3 Quebec                   40
       3 -                       104
       4 Manitoba                 17
       4 Ontario-North             1
       4 Ontario-South            14
       4 Quebec                   11
       4 -                        43
      12 Manitoba                  2
      12 Ontario-South             4
      12 Quebec                    2
      12 -                         8
       - -                       155
```

例 *C6-3:*

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY GROUPING SETS(ROLLUP(WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE) ),
                       ROLLUP( MONTH(SALES_DATE), REGION ) )
ORDER BY WEEK, DAY_WEEK, MONTH, REGION
```

これは次のような結果になります。

```
WEEK     DAY_WEEK   MONTH   REGION        UNITS_SOLD
-------- ---------- ------- ------------- -----------
      13          6       - -                      27
      13          7       - -                      46
      13          -       - -                      73
      14          1       - -                      31
      14          2       - -                      43
      14          -       - -                      74
      53          1       - -                       8
      53          -       - -                       8
       -          -       3 Manitoba               22
       -          -       3 Ontario-North           8
       -          -       3 Ontario-South          34
       -          -       3 Quebec                 40
       -          -       3 -                     104
       -          -       4 Manitoba               17
       -          -       4 Ontario-North           1
       -          -       4 Ontario-South          14
       -          -       4 Quebec                 11
```

```
-          -       4 -                      43
-          -      12 Manitoba                2
-          -      12 Ontario-South           4
-          -      12 Quebec                  2
-          -      12 -                       8
-          -       - -                     155
-          -       - -                     155
```

2 つの ROLLUP をグループ化集合として使用すると、結果に重複した行が組み入れられます。総計行も 2 つになります。

ORDER BY を使用すると結果にどのような影響があるかを調べてみます。

- 最初のグループ化集合では、week 53 が最後に位置変更されています。
- 2 番目のグループ化集合では、month 12 が最後に位置付けられ、地域がアルファベット順になっています。
- NULL 値は上位にソートされます。

例 C7:

単一のパスで複数の ROLLUP を実行する照会 (例 C6-3 など) では、各行を生成したのがどのグループ化集合であるかを示すことができます。以下のステップは、結果セット内の各行の発生点を示す列 (GROUP と呼ばれます) を提供する方法を示しています。発生点とは、結果セットの行を生成したのが、2 つのグループ化集合のいずれであるかということです。

ステップ *1:* VALUES 文節から選択する照会 (代替形式の全選択) を使用して、新しいデータ値を生成する方法を導入します。この照会は、2 つの列 R1 と R2、および 1 行のデータがある、 X と呼ばれる表を得る方法を示しています。

```
SELECT R1, R2
FROM (VALUES ('GROUP 1', 'GROUP 2')) AS X(R1, R2)
```

結果は、以下のようになります。

```
R1       R2
-------- -------
GROUP 1  GROUP 2
```

ステップ *2:* この表 X と SALES 表とのクロス乗積を生成します。これにより、すべての行に列 R1 と R2 が追加されます。

```
SELECT R1, R2,
       WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION,
       SALES AS UNITS_SOLD
FROM SALES,
     (VALUES ('GROUP 1', 'GROUP 2')) AS X(R1, R2)
```

ステップ *3:* これで、これらの列をグループ化集合と組み合わせて、 ROLLUP 分析にこれらの列を含めることができるようになりました。

```
SELECT R1, R2,
       WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD
```

```
FROM SALES,
     (VALUES ('GROUP 1', 'GROUP 2')) AS X(R1, R2)
GROUP BY GROUPING SETS((R1, ROLLUP(WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE)) ),
                       (R2, ROLLUP(MONTH(SALES_DATE), REGION)) )
ORDER BY WEEK, DAY_WEEK, MONTH, REGION
```

これは次のような結果になります。

```
R1       R2      WEEK     DAY_WEEK   MONTH   REGION        UNITS_SOLD
-------- ------- -------- ---------- ------- ------------- -----------
GROUP 1  -             13          6       - -                      27
GROUP 1  -             13          7       - -                      46
GROUP 1  -             13          -       - -                      73
GROUP 1  -             14          1       - -                      31
GROUP 1  -             14          2       - -                      43
GROUP 1  -             14          -       - -                      74
GROUP 1  -             53          1       - -                       8
GROUP 1  -             53          -       - -                       8
-        GROUP 2        -          -       3 Manitoba               22
-        GROUP 2        -          -       3 Ontario-North           8
-        GROUP 2        -          -       3 Ontario-South          34
-        GROUP 2        -          -       3 Quebec                 40
-        GROUP 2        -          -       3 -                     104
-        GROUP 2        -          -       4 Manitoba               17
-        GROUP 2        -          -       4 Ontario-North           1
-        GROUP 2        -          -       4 Ontario-South          14
-        GROUP 2        -          -       4 Quebec                 11
-        GROUP 2        -          -       4 -                      43
-        GROUP 2        -          -      12 Manitoba                2
-        GROUP 2        -          -      12 Ontario-South           4
-        GROUP 2        -          -      12 Quebec                  2
-        GROUP 2        -          -      12 -                       8
-        GROUP 2        -          -       - -                     155
GROUP 1  -             -          -       - -                     155
```

ステップ *4:* R1 および R2 が異なるグループ化集合で使用されるため、 R1 の結果が非 NULL である場合は常に R2 は NULL であり、 R2 の結果が非 NULL である場合は常に R1 は NULL になることに注意してください。つまり、 COALESCE 関数を使用すれば、これらの列を単一列に統合できるということです。また、ORDER BY 節でこの列を使用すれば、 2 つのグループ化集合の結果をまとめることもできます。

```
SELECT COALESCE(R1, R2) AS GROUP,
       WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD
FROM SALES,
         (VALUES ('GROUP 1', 'GROUP 2')) AS X(R1, R2)
GROUP BY GROUPING SETS((R1, ROLLUP(WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE)) ),
                       (R2, ROLLUP(MONTH(SALES_DATE), REGION)) )
ORDER BY GROUP, WEEK, DAY_WEEK, MONTH, REGION
```

これは次のような結果になります。

```
GROUP    WEEK     DAY_WEEK   MONTH   REGION        UNITS_SOLD
-------- -------- ---------- ------- ------------- -----------
GROUP 1        13          6       - -                      27
GROUP 1        13          7       - -                      46
GROUP 1        13          -       - -                      73
GROUP 1        14          1       - -                      31
GROUP 1        14          2       - -                      43
GROUP 1        14          -       - -                      74
GROUP 1        53          1       - -                       8
```

```
GROUP 1        53          -       - -                     8
GROUP 1         -          -       - -                   155
GROUP 2         -          -       3 Manitoba             22
GROUP 2         -          -       3 Ontario-North         8
GROUP 2         -          -       3 Ontario-South        34
GROUP 2         -          -       3 Quebec               40
GROUP 2         -          -       3 -                   104
GROUP 2         -          -       4 Manitoba             17
GROUP 2         -          -       4 Ontario-North         1
GROUP 2         -          -       4 Ontario-South        14
GROUP 2         -          -       4 Quebec               11
GROUP 2         -          -       4 -                    43
GROUP 2         -          -      12 Manitoba              2
GROUP 2         -          -      12 Ontario-South         4
GROUP 2         -          -      12 Quebec                2
GROUP 2         -          -      12 -                     8
GROUP 2         -          -       - -                   155
```

例 C8:

以下の例は、CUBE を実行する場合の種々の集約関数の使用例を示しています。また、この例は、cast 関数および丸めを利用して、妥当な精度と位取りで 10 進数の結果を生成します。

```
SELECT MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS SALES,
       MAX(SALES) AS BEST_SALE,
       CAST(ROUND(AVG(DECIMAL(SALES)),2) AS DECIMAL(5,2)) AS AVG_UNITS_SOLD
FROM SALES
GROUP BY CUBE (MONTH(SALES_DATE), REGION)
ORDER BY MONTH, REGION
```

これは次のような結果になります。

```
MONTH   REGION        UNITS_SOLD   BEST_SALE  AVG_UNITS_SOLD
------- ------------- ------------ ---------- ---------------
      3 Manitoba               22          7            3.14
      3 Ontario-North           8          3            2.67
      3 Ontario-South          34         14            4.25
      3 Quebec                 40         18            5.00
      3 -                     104         18            4.00
      4 Manitoba               17          9            5.67
      4 Ontario-North           1          1            1.00
      4 Ontario-South          14          8            4.67
      4 Quebec                 11          8            5.50
      4 -                      43          9            4.78
     12 Manitoba                2          2            2.00
     12 Ontario-South           4          3            2.00
     12 Quebec                  2          1            1.00
     12 -                       8          3            1.60
      - Manitoba               41          9            3.73
      - Ontario-North           9          3            2.25
      - Ontario-South          52         14            4.00
      - Quebec                 53         18            4.42
      - -                     155         18            3.87
```

HAVING 文節

HAVING 節は、 *search-condition* (検索条件) が真である R のグループで構成される中間結果表を指定します。 R は、副選択の直前の文節の結果です。この直前の文節が GROUP BY 文節でない場合、R はグループ化式のない単一グループとして扱われます。

検索条件に列名 を含むそれぞれの式は、次のいずれかを満たす必要があります。

- R のグループ化式を明確に識別すること。
- 集約関数の中に指定されていること。
- 相関参照であること。列名 が外側の副選択で識別される表、ビュー、共通表式、またはネストされた表式の列を識別する場合、その列名は相関参照になります。

RRN、RID、DATAPARTITIONNAME、DATAPARTITIONNUM、DBPARTITIONNAME、DBPARTITIONNUM、および HASHED_VALUE 関数は、集約関数の内部にある場合を除いて、 HAVING 文節に指定することはできません。集約関数を使用する場合の制約事項については、 303 ページの『集約関数』を参照してください。

検索条件の中のそれぞれの集約関数 (引数が相関参照であるものを除く) には、検索条件が適用される R のグループから引数が与えられます。

検索条件の中に副照会がある場合は、R のグループに検索条件が適用されるたびにその副照会が実行され、その副照会の結果が、検索条件を適用する際に使用されると考えても構いません。実際には、副照会が各グループごとに実行されるのは、その副照会の中に相関参照がある場合だけです。この相違については、 839 ページの『副選択の例』の例 6 および 7 を参照してください。

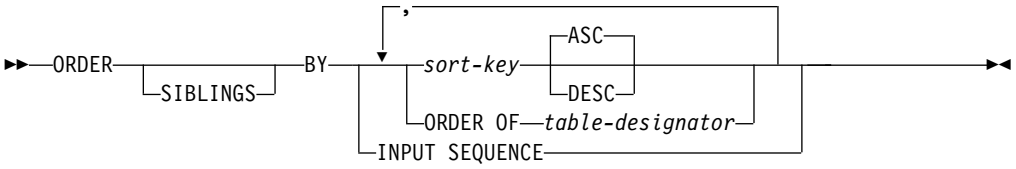
R のグループに対する相関参照では、グループ化列を識別しなければなりません。グループ化列を識別しない場合は、相関参照を集約関数の中に入れる必要があります。

GROUP BY を指定せずに HAVING を使用する場合は、選択リスト内のすべての列名を集約関数の中に入れなければなりません。

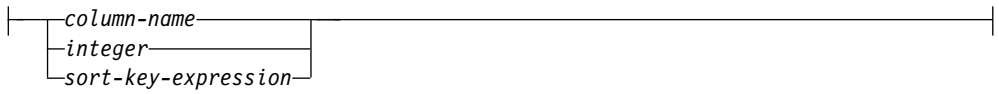
HAVING 文節を含むステートメントの実行時に *HEX 以外の照合順序が有効で、しかもその検索条件 に SBCS データ、混合データ、または Unicode データを持つオペランドが含まれている場合は、それらの述部の比較は、重み付けされた値を使用して行われます。この重み付けされた値は、述部の中のオペランドに対して該当の照合順序を適用して得られます。

ORDER BY 文節

ORDER BY 文節は、結果表の行の順序付けを指定します。



sort-key:



ORDER BY 文節を含む副選択は、ビューの最外部の全選択で指定することはできません。

注: 副選択の ORDER BY 文節は、照会によって戻される行順序には影響を与えません。ORDER BY 文節は、最外部の全選択で指定されている場合にのみ戻される行順序に影響を与えます。

副選択が小括弧で囲まれておらず、かつその副選択が最外部の全選択ではない場合は、ORDER BY 文節を指定することはできません。

ソート指定 (関連した昇順または降順の順序付けを指定した 1 つのソート・キー) が 1 つの場合は、行はその指定の値によって順に並べられます。ソート指定が複数ある場合は、行は、最初に示されたソート指定の値によって、次に 2 番目に示されたソート指定の値によって、以下同様の値によって順に並べられます。

副選択に階層照会文節 が含まれている場合、ORDER SIBLINGS BY を指定できます。これは、階層内の兄弟のみに順序が適用されることと、親の行は子の行の前にソートされることを指定します。

ORDER BY 文節を含むステートメントの実行時に *HEX 以外の照合順序が有効で、しかもその ORDER BY 文節に SBCS データ、混合データ、または Unicode データのソート指定が含まれている場合、それらのソート指定の比較は、重み付けされた値を使用して行われます。この重み付けされた値は、ソート指定の値に該当の照合順序を適用して得られます。

選択リスト内の名前のある列は、整数 または列名 のソート・キー で識別することができます。選択リスト内の名前のない列は、整数 または場合によっては選択リスト内の式と一致するソート・キー式 で識別することができます (ソート・キー式の詳細を参照)。結果列に名前がない場合については、 792 ページの『結果列の名前』を参照してください。 UNION 演算子を含む全選択の場合に、全選択における列の名前についての規則は、 841 ページの『全選択』を参照してください。

順序付けは、 57 ページの『第 2 章 言語エレメント』で説明した比較規則にしたがって行われます。 NULL 値は、他のどの値よりも上位に置かれます。ユーザー

の順序付けの指定によって完全な順序が判別できない場合は、最後に指定されたソート・キー の値が重複する行は、順序が不定になります。また、ORDER BY 文節を指定しなかった場合は、結果表の行の順序が不定になります。

ソート・キー の長さ属性の合計は 3.5 ギガバイトを超えてはなりません。

column-name
結果表の列を明確に識別するものでなければなりません。この列は DATALINK 列または XML 列であってはならず、ARRAY_AGG 関数の結果であってもなりません。明確な列参照の規則は、全選択の他の文節の場合と同じです。詳しくは、 166 ページの『あいまいさを避けるための列名修飾子』を参照してください。

全選択に UNION、UNION ALL、EXCEPT、または INTERSECT が含まれる場合は、列名は修飾できません。

column-name は、FROM 文節に指定されている表、ビュー、または *nested-table-expression* の列名も識別することができます。これには、暗黙的隠し列として定義された列も含まれます。副選択の選択リストに集約が含まれており、列名 がグループ化式 でない場合は、エラーになります。

integer
0 より大きく、結果表の列の数以下の値を指定しなければなりません。整数 *n* は、結果表の *n* 番目の列を指定します。この識別された列は DATALINK 列または XML 列であってはならず、ARRAY_AGG 関数の結果であってもなりません。

sort-key-expression
単純な 1 つの列名または無符号の整数定数ではない式。順序付けを適用する照会では、この形式のソート・キー を使用するために副選択にする必要があります。

全選択 に UNION、UNION ALL、EXCEPT、または INTERSECT が含まれている場合は、ソート・キー式 に RRN、RID、DATAPARTITIONNAME、DATAPARTITIONNUM、DBPARTITIONNAME、DBPARTITIONNUM、および HASHED_VALUE スカラー関数を入れることはできません。
sort-key-expression の結果は、DATALINK または XML であってはなりません。

副選択がグループ化されている場合は、*sort-key-expression* には、副選択の選択リスト内の式を使用したり、集約関数、定数、または変数を含めたりすることができます。

ASC
列の値を昇順に使用します。これはデフォルトです。

DESC
列の値を降順に使用します。

ORDER OF *table-designator*
表指定子 で使用されているのと同じ順序付けを、副選択の結果表にも適用することを指定します。表指定子 に一致する表参照が FROM 文節を指定する副選択のその文節内になければならず、その表参照はネストされた表の式 または 共通表式 を識別するものでなければなりません。適用される順序付けは、ネストされた表の式 または共通表式 の ORDER BY 文節の列が外部副選択 (または

全選択) に組み込まれ、それらの列が ORDER OF 文節に代わって指定される場合と同じです。 *nested-table-expression* または *common-table-expression* に ORDER BY 文節がない場合、順序は定義されません。

ORDER OF は、照会が以下を指定する場合には使用できません。

- 分散表
- 読み取りトリガーを指定する表
- 複数の物理ファイル・メンバー上に構築された論理ファイル

INPUT SEQUENCE
結果表が INSERT ステートメントの行の入力順序を反映することを示します。INPUT SEQUENCE の順序付けは、*FROM* 文節 内で INSERT ステートメントを指定する場合のみに指定できます。 INPUT SEQUENCE が指定されていても、入力データが配列されていない場合は、 INPUT SEQUENCE 文節は無視されます。

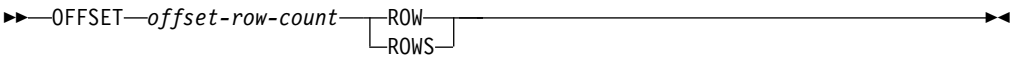
列アクセス制御は ORDER BY の操作に影響しません。順序は元の列値に基づきます。ただし、列マスクの適用後、最終結果表内のマスク値が元の列値の順序を反映しない可能性があります。

照会に以下の指定がある場合、*sort-key* は LOB であってはなりません。

- 分散表
- 読み取りトリガーを指定する表
- 複数の物理ファイル・メンバー上に構築された論理ファイル

offset-clause

offset-clause では、行が取得される前にスキップする行の数を設定します。これは、*offset-row-count* 行スキップされるまでアプリケーションが行の取得を開始しないことを、データベース・マネージャーに認識させます。 *offset-clause* が指定されない場合、デフォルトは OFFSET 0 ROWS と同等になります。 *offset-row-count* に中間結果表に行数よりも多くの行が指定される場合、中間結果表は空の結果表として扱われます。



offset-row-count

行を取得する前にスキップする行数を指定する式。*offset-row-count* に *scalar-fullselect*、列参照、表参照、ユーザー定義関数参照、および検査制約で制限と識別された組み込みスカラー関数を含めてはなりません。check-constraint を参照してください。式のデータ・タイプが BIGINT でない場合は、式の結果が BIGINT 値にキャストされます。 *offset-row-count* の値は、正数またはゼロでなければなりません。これを NULL 値にすることはできません。

スキップする行の予測される集合を決定するには、中間結果表内の各行のソート順序を一意的に識別するソート・キーを使用した ORDER BY 文節の指定が必要です。一部の行について重複するソート・キーが中間結果表に含まれると、行の順序は決定的ではありません。 ORDER BY 文節がない場合、中間結果表は、決定的順序ではありません。中間結果表の順序が決定的でないと、スキップされた行集合は予測不能になります。

offset-clause は、ビューの最外部の全選択で指定することはできません。

全選択の FROM 節に SQL データ変更ステートメントが含まれている場合は、スキップする行数にかかわりなく、すべての行が変更されます。

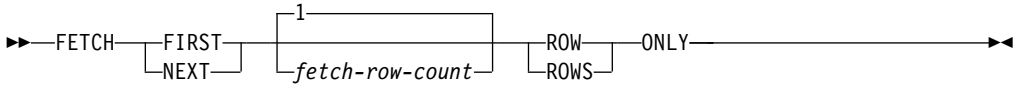
注

select-list **の制約事項:** *offset-clause* を含む全選択の *select-list* には、NEXT VALUE シーケンス式を含めてはなりません。また、非決定的な関数、外部アクションである関数、および SQL データを変更する関数を含めることもできません。

代替構文**:** 取得する最大行数の指定でスキップする行数を設定するための代替構文については、*fetch-clause* に関する注の項目を参照してください。

fetch-clause

fetch-clause は、取り出すことができる最大行数を設定します。



fetch-clause は、取り出すことができる最大行数を設定します。これは、中間結果表内の行数に関係なく、アプリケーションが最大 *fetch-row-count* 行までしか取得しないことを、データベース・マネージャーに認識させます。*fetch-row-count* 行を超えて取り出そうとすると、通常のデータの終わりと同様に処理されます。

fetch-clause は、ビューの最外部の全選択で指定することはできません。

取得する行の予測される集合を決定するには、中間結果表内の各行のソート順序を一意的に識別するソート・キーを使用した ORDER BY 文節の指定が必要です。一部の行について重複するソート・キーが中間結果表に含まれると、行の順序は決定的ではありません。 ORDER BY 節がない場合、中間結果表の順序は確定されません。中間結果表の順序が決定的でないと、取得された行集合は予測不能になります。*order-by-clause* と *fetch-clause* の両方を指定すると、 *fetch-clause* は、順序付けされたデータで処理されます。

fetch-row-count
取得する行の最大数を指定する式。 *fetch-row-count* に *scalar-fullselect*、列参照、表参照、ユーザー定義関数参照、および検査制約で制限と識別された組み込みスカラー関数を含めてはなりません。 check-constraint を参照してください。式のデータ・タイプが BIGINT でない場合は、式の結果が BIGINT 値にキャストされます。 *fetch-row-count* の値は、正数またはゼロでなければなりません。これを NULL 値にすることはできません。

結果表を指定の行数に限定することで、パフォーマンスが向上します。場合によっては、データベース・マネージャーは、指定された行数を判別したときに、照会の処理を停止します。*offset-clause* も指定された場合、データベース・マネージャーは、処理を停止するタイミングを判別する際にこのオフセット値も考慮します。

全選択に FROM 節の SQL データ変更ステートメントが入っている場合は、フェッチされる行の数の限度に関係なく、すべての行が変更されます。

行アクセス制御は副選択の許可 ID がアクセスできる行に影響するため、行アクセス制御は間接的に FETCH FIRST 節に影響する可能性があります。

注

代替構文:

- キーワード FIRST とキーワード NEXT は、同じ意味で使用されます。結果は変わりませんが、*offset-clause* を使用する場合は、キーワード NEXT を使用したほうが読みやすくなります。
- 他のデータベース製品で使用される SQL との互換性のために、以下がサポートされています。これらの代替は非標準であり、使用すべきではありません。

代替構文	同等の構文
LIMIT *x*	FETCH FIRST *x* ROWS ONLY
LIMIT *x* OFFSET *y*	OFFSET *y* ROWS FETCH FIRST *x* ROWS ONLY
LIMIT *y*, *x*	OFFSET *y* ROWS FETCH FIRST *x* ROWS ONLY

副選択の例

副選択はさまざまな方法で使用することができます。

例 1

表 EMPLOYEE から、すべての列および行を選択します。

```
SELECT * FROM EMPLOYEE
```

例 2

表 EMPPROJACT と EMPLOYEE を結合し、表 EMPPROJACT のすべての列を選択し、結果の各行に表 EMPLOYEE からの社員の姓 (LASTNAME) を加えます。

```
SELECT EMPPROJACT.*, LASTNAME
  FROM EMPPROJACT, EMPLOYEE
  WHERE EMPPROJACT.EMPNO = EMPLOYEE.EMPNO
```

例 3

表 EMPLOYEE と表 DEPARTMENT を結合します。誕生日 (BIRTHDATE) が 1930 年より前であるすべての社員の従業員番号 (EMPNO)、社員のラストネーム (LASTNAME)、部門番号 (表 EMPLOYEE の WORKDEPT と表 DEPARTMENT の DEPTNO)、および部門名 (DEPTNAME) を選択します。

```
SELECT EMPNO, LASTNAME, WORKDEPT, DEPTNAME
  FROM EMPLOYEE, DEPARTMENT
  WHERE WORKDEPT = DEPTNO
  AND YEAR(BIRTHDATE) < 1930
```

この副選択は、次のようにも書けます。

```
SELECT EMPNO, LASTNAME, WORKDEPT, DEPTNAME
  FROM EMPLOYEE INNER JOIN DEPARTMENT
  ON WORKDEPT = DEPTNO
  WHERE YEAR(BIRTHDATE) < 1930
```

例 4

表 EMPLOYEE において、同一のジョブ・コードを持つ行のグループごとに、ジョブ (JOB) と、給与 (SALARY) の最高額および最低額を選択します。ただし、対象となるグループは、複数の行があり、給与の最高額が 27000 以上であるものに限ります。

```
SELECT JOB, MIN(SALARY), MAX(SALARY)
  FROM EMPLOYEE
  GROUP BY JOB
  HAVING COUNT(*) > 1 AND MAX(SALARY) >= 27000
```

例 5

表 EMPPROJACT から、部門 (WORKDEPT) 'E11' の社員 (EMPNO) に関するすべての行を選択します。(社員の部門番号は、表 EMPLOYEE に示されています。)

```
SELECT * FROM EMPPROJACT
  WHERE EMPNO IN (SELECT EMPNO FROM EMPLOYEE
                  WHERE WORKDEPT = 'E11')
```

例 6

表 EMPLOYEE から、部門別給与 (SALARY) の最高額が全社員の平均給与より少ないすべての部門について、その部門番号 (WORKDEPT) と部門別給与 (SALARY) の最高額を選択します。

```
SELECT WORKDEPT, MAX(SALARY)
  FROM EMPLOYEE
  GROUP BY WORKDEPT
  HAVING MAX(SALARY) < (SELECT AVG(SALARY)
                        FROM EMPLOYEE)
```

この例では、HAVING 文節の副照会は一度だけ実行されることになります。

例 7

表 EMPLOYEE を使用して、部門別給与 (SALARY) の最高額が他のすべての部門の平均給与より少ない部門について、その部門番号 (WORKDEPT) と部門別給与 (SALARY) の最高額を選択します。

```
SELECT WORKDEPT, MAX(SALARY)
  FROM EMPLOYEE EMP_COR
  GROUP BY WORKDEPT
  HAVING MAX(SALARY) < (SELECT AVG(SALARY)
                        FROM EMPLOYEE
                        WHERE NOT WORKDEPT = EMP_COR.WORKDEPT)
```

例 6 とは対照的に、HAVING 文節の副照会は各グループごとに実行する必要があります。

例 8

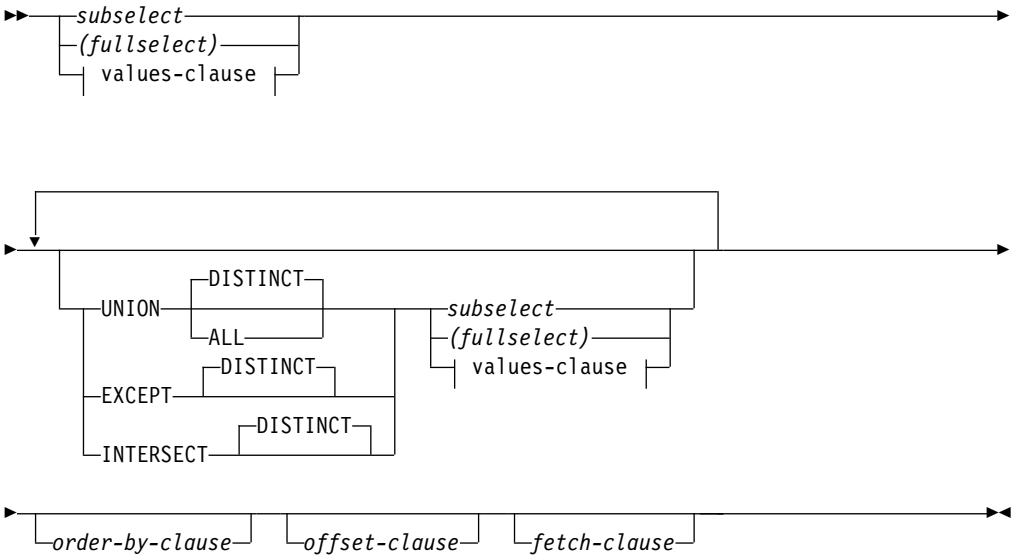
表 EMPLOYEE と EMPPROJACT を結合し、社員全員とそのプロジェクト番号をすべて選択します。現在割り当てられているプロジェクト番号がない社員も戻します。

```
SELECT EMPLOYEE.EMPNO, PROJNO
  FROM EMPLOYEE LEFT OUTER JOIN EMPPROJACT
  ON EMPLOYEE.EMPNO = EMPPROJACT.EMPNO
```

表 EMPLOYEE 内の社員で、表 EMPPROJACT にプロジェクト番号がない者も、EMPNO の値が入っている結果表の 1 行、および PROJNO の列の NULL 値を戻します。

全選択

全選択 は、*SELECT* ステートメント、マテリアライズ照会表の定義に関する ALTER TABLE ステートメント、CREATE TABLE ステートメント、CREATE VIEW ステートメント、DECLARE GLOBAL TEMPORARY TABLE ステートメント、INSERT ステートメント、遷移変数の SET ステートメント、SET VARIABLE ステートメント、 UPDATE ステートメント、割り当てステートメントのコンポーネントです。



values-clause:

values-row:

括弧で囲んだ全選択を副照会 といいます。例えば、副照会 は検索条件の中で使用することができます。

スカラー全選択 は括弧で囲んだ全選択で、単一の結果行および単一の結果列を戻します。全選択の結果に行が含まれない場合、NULL 値が戻されます。結果の中に複数の行がある場合には、エラーが戻されます。

全選択 は、結果表を指定するものです。 UNION、EXCEPT、または INTERSECT を使用しない場合は、指定した副選択または *values-clause* の結果が全選択の結果となります。

values-clause

結果表の行の各列ごとに式を使用して実際の値を指定することによって、結果表を派生させます。複数の行を指定することができます。

NULL は、複数の *values-row* 指定のみで使用することができ、同一列の少なくとも 1 行は NULL 以外でなければなりません。

Values-row は以下によって指定されます。

- 結果表の単一の列についての単一の式
- コンマで区切った *n* 個の式 (または NULL) を括弧で囲んだもの (*n* は結果表の列の数)

複数行からなる values-clause には、各 *values-row* に同数の式が必要です。

以下に、*values-clauses* の例とその意味を示します。

```
VALUES (1), (2), (3)           - 3 rows of 1 column
VALUES 1, 2, 3                 - 3 rows of 1 column
VALUES (1, 2, 3)               - 1 row of 3 columns
VALUES (1,21), (2,22), (3,23)  - 3 rows of 2 columns
```

values-clause は、 *n* 個の指定の *Values-row* RE_1 から RE_n (*n* は 2 以上) で構成され、以下と同等です。

RE_1 UNION ALL RE_2 ... UNION ALL RE_n

これは、各 *values-row* に対応する式が比較可能でなければならないことを示します。 *values-row* 内のすべての結果列には名前が付けられません。

UNION、EXCEPT、または INTERSECT

セット演算子の UNION、EXCEPT、INTERSECT は、関係演算子の結合、差、論理積にそれぞれ対応します。全選択は、結果表を指定するものです。セット演算子が使用されていない場合、全選択の結果は、指定された副選択の結果となります。セット演算子が使用されている場合は、結果表は、指定されたセット演算子に従って 2 つの結果表 (R1 と R2) の組み合わせから導き出されます。

UNION DISTINCT または UNION ALL

ALL オプションを付けずに UNION を指定すると、R1 または R2 にあるすべての行の集合から重複行を除去したものが結果表に入ります。ただし、どちらの場合も、UNION 表の各行は R1 または R2 のいずれかから得られた行です。

EXCEPT DISTINCT

結果は R1 のみに含まれるすべての行で構成され、この操作によって生じる重複行は除去されます。 DIFFERENCE の結果表の各行は、R1 に存在するが R2 には存在しない行です。

INTERSECT DISTINCT

結果は R1 と R2 の両方に含まれすべての行で構成され、重複行は除去されます。 INTERSECTION の結果表の行は、R1 と R2 の両方に存在する行です。

R1 および R2 内の *n* 番目の列に対応する式は、列マスクがある列を参照できます。セット演算の結果の *n* 番目の列は、R1 または R2 内のマスクされた値から導き出すことができます。

DISTINCT を使用すると、重複行の除去は、R1 および R2 内のマスクされない値に基づいて行われます。すべての行は R1 または R2 からのものであるため、以下の条件のうち 1 つ以上が発生した場合、セット演算の結果表内の出力値が変わる可能性があります。

- R1 の *n* 番目の列に対応する式は、列マスクがある列を参照しているが、R2 の *n* 番目の列に対応する式は参照していない (またはその逆)。
- R1 と R2 の *n* 番目の列に対応する式が、異なる列マスクがある列を参照している。
- 列マスク定義が、列マスクの定義の対象であるターゲット列と同じではない列を参照し、それらの列が DISTINCT 操作の一部ではない。列マスク定義ではターゲット表の他の列を参照しないようにすることをお勧めします。

例えば、R1 の行はマスクされた値から派生し、R2 の行はマスクされていない値から派生します。DISTINCT が結果表の行を R1 から選出する場合、マスクされた値が戻されます。DISTINCT が結果表の行を R2 から選出する場合、マスクされていない値が戻されます。

列に関する規則:

- R1 と R2 の列の数は同じでなければなりません。また、R1 の *n* 番目の列のデータ・タイプと R2 の *n* 番目の列のデータ・タイプには、互換性がなければなりません。文字ストリング値は、日付/時刻の値と互換性があります。
- UNION、UNION ALL、EXCEPT、または INTERSECT の結果の *n* 番目の列は、 R1 および R2 の *n* 番目の列から得られます。結果列の属性は、結果列に関する規則を使用して決定します。
- R1 の *n* 番目の列を指定すると、結果表の *n* 番目の列は、その結果列の名前になります。それ以外の場合は、その結果の列は無名となります。
- UNION、INTERSECT、または EXCEPT を指定するときは、どの列も DATALINK 列または XML 列であってはなりません。

オペランド列の有効な組み合わせと、結果列のデータ・タイプについては、 133 ページの『結果のデータ・タイプに関する規則』を参照してください。

EXCEPT と INTERSECT に関する制約事項: 照会で以下のいずれかを指定する場合は、VALUES、INTERSECT、EXCEPT を使用できません。

- 分散表
- 読み取りトリガーを指定する表
- 複数の物理ファイル・メンバー上に構築された論理ファイル

重複行: 2 つの行が重複していると見なされるのは、一方の行のそれぞれの値が、もう一方の行の対応する値にすべて等しい場合です。(重複を判別する際には、2 つの NULL 値は互いに等しいものと見なされます。)

演算子優先順位: UNION、UNION ALL、および INTERSECT は、結合セット演算です。ただし、UNION、UNION ALL、 EXCEPT、および INTERSECT を同一ステートメントで使用すると、操作の実行順序によって結果が異なります。括弧内

の操作が最初に実行されます。括弧で順序を指定しない場合は、操作は左から右の順に実行されます。ただし、INTERSECT 操作はすべて、UNION または EXCEPT 操作の前に実行されます。

セット演算の結果**:** 次の例では、表 R1 および R2 の値が左側に示されています。残りの見出しは、R1 および R2 に対するさまざまなセット演算の結果の値を示しています。

R1	R2	UNION ALL	UNION	EXCEPT	INTERSECT
1	1	1	1	2	1
1	1	1	2	5	3
1	3	1	3		4
2	3	1	4		
2	3	1	5		
2	3	2			
3	4	2			
4		2			
4		3			
5		3			
		3			
		3			
		3			
		4			
		4			
		4			
		5			

照合順序**:** UNION、EXCEPT、または INTERSECT キーワードを含むステートメントの実行時に *HEX 以外の照合順序が有効で、しかもその結果表に SBCS データ、混合データ、または Unicode データを持つ列が含まれている場合、それらの列の比較は、重み付けされた値を使用して行われます。この重み付けされた値は、それぞれの値に該当の照合順序を適用して得られます。

全選択の例

例 1

表 EMPLOYEE から、すべての列および行を選択します。

```
SELECT * FROM EMPLOYEE
```

例 2

EMPLOYEE 表で、部門番号 (WORKDEPT) が 'E' で始まる社員、または EMPPROJACT 表でプロジェクト番号 (PROJNO) が 'MA2100'、'MA2110'、または 'MA2112' に等しいプロジェクトに参画している社員全員の従業員番号 (EMPNO) をリストします。

```
SELECT EMPNO FROM EMPLOYEE
  WHERE WORKDEPT LIKE 'E%'
UNION
SELECT EMPNO FROM EMPPROJACT
  WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

例 3

例 2 と同じ照会を行いますが、重複行が除去されないように、UNION ALL を使用しています。

```
SELECT EMPNO FROM EMPLOYEE
  WHERE WORKDEPT LIKE 'E%'
UNION ALL
SELECT EMPNO FROM EMPPROJACT
  WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

例 4

例 2 と同じ照会を行うのに加えて、EMPLOYEE 表からの行に 'emp'、EMPPROJACT 表からの行に 'empprojact' の「タグ」を付けます。例 2 からの結果とは異なり、この照会は、関連の「タグ」 によって表を識別して、同じ EMPNO を 2 度以上戻すことがあります。

```
SELECT EMPNO, 'emp' FROM EMPLOYEE
  WHERE WORKDEPT LIKE 'E%'
UNION
SELECT EMPNO, 'empprojact' FROM EMPPROJACT
  WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

例 5

この EXCEPT の例では、T2 に含まれず T1 に含まれるすべての行が生成され、重複行は除去されます。

```
(SELECT * FROM T1)
     EXCEPT DISTINCT
(SELECT * FROM T2)
```

NULL 値が関係しない場合は、この例は次の例と同じ結果を戻します。

```
(SELECT DISTINCT *
   FROM T1
   WHERE NOT EXISTS (SELECT * FROM T2
                       WHERE  T1.C1 = T2.C1 AND T1.C2 = T2.C2 AND...) )
```

ここで、C1、C2 などは、T1 および T2 の列を表します。

例 6

この INTERSECT の例では、T1 と T2 の両方に含まれているすべての行が生成され、重複行は除去されます。

```
(SELECT * FROM T1)
     INTERSECT DISTINCT
(SELECT * FROM T2)
```

NULL 値が関係しない場合は、この例は次の例と同じ結果を戻します。

```
(SELECT DISTINCT *
   FROM T1
   WHERE EXISTS (SELECT * FROM T2
                   WHERE  T1.C1 = T2.C1 AND T1.C2 = T2.C2 AND...) )
```

ここで、C1、C2 などは、T1 および T2 の列を表します。

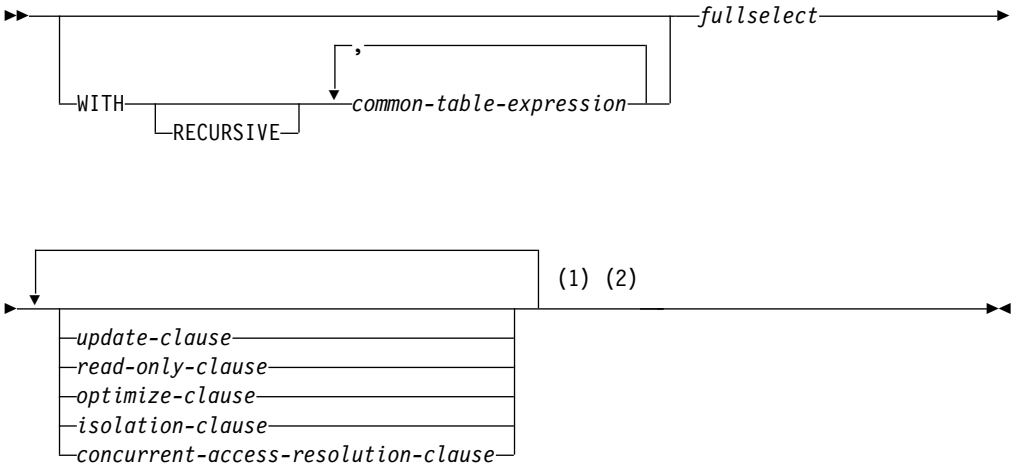
例 7

例 3 と同じ照会を行いますが、現在どの表にもない 2 人の従業員を追加して、それらの行に "new" というタグを付けます。

```
SELECT EMPNO, 'emp' FROM EMPLOYEE
  WHERE WORKDEPT LIKE 'E%'
UNION
SELECT EMPNO, 'empprojact' FROM EMPPROJACT
  WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
UNION
VALUES ('NEWAAA', 'new'), ('NEWBBB', 'new')
```

選択ステートメント

SELECT ステートメント は、1 つの照会形式であり、DECLARE CURSOR ステートメントまたは FOR ステートメントの中で直接指定できます。準備後に DECLARE CURSOR ステートメントの中で参照することも可能です。SQLJ の割り当て文節の中で直接指定することもできます。さらに、対話式に実行することも可能です。いずれの場合も、選択ステートメント で指定した表が、全選択の結果となります。



注:

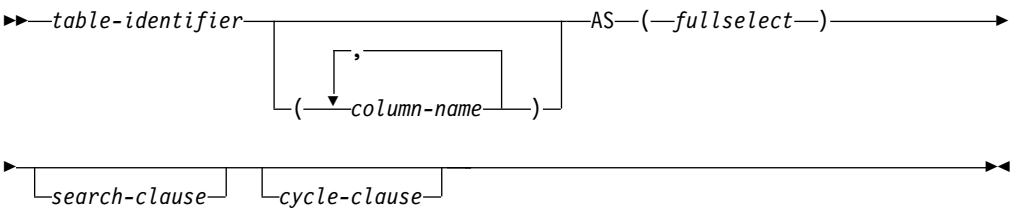
1 UPDATE 文節と READ-ONLY 文節は、同一の選択ステートメントで両方をともに指定することはできません。

2 各文節はそれぞれ 1 回だけ指定できます。

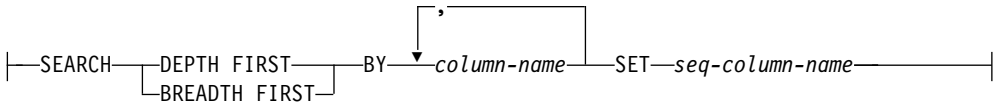
RECURSIVE
共通表式 が反復する可能性のあることを示しています。

共通表式

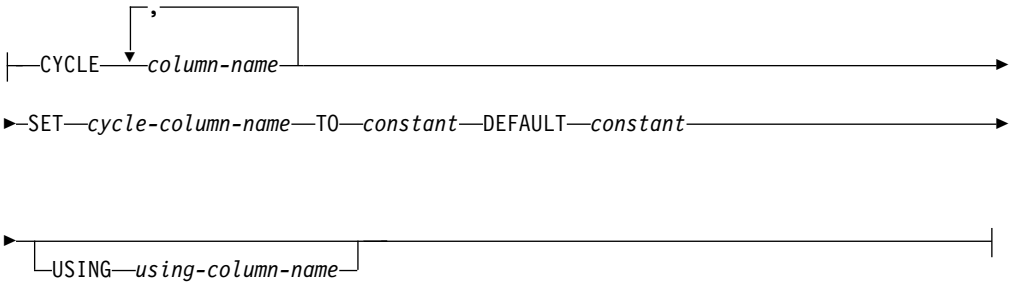
共通表式 を使用すると、次に続く全選択の FROM 文節で表として指定できる表 *ID* を持つ結果表を定義することができます。 1 つの WITH キーワードに続けて、複数の共通表式を指定することができます。指定された各共通表式は、後続の共通表式の FROM 文節でその名前を参照することもできます。



search-clause:



cycle-clause:



列名のリストを指定する場合は、そのリストは、全選択の結果表にある列の数と同じ数の列名で構成されている必要があります。各 *column-name* (列名) は、ユニークで、しかも非修飾でなければなりません。これらの列名を指定しない場合は、列名は、共通表式を定義するために使用される副選択の選択リストから取得されます。

共通表式の表 *ID* は、同じステートメント内の他の共通表式の表 *ID* とは異なっている必要があります。共通表式の表 *ID* は、全選択におけるどの FROM 文節の表名として指定してもかまいません。共通表式の表 *ID* は、同じ非修飾名を持つ既存の表、ビュー、または別名 (カタログ内の) をオーバーライドするか、トリガーに任意の表 *ID* を指定します。

同一ステートメントの中で複数の共通表式を定義した場合、それらが共通表式間で相互に参照し合うことはできません。循環参照 は、2 つの共通表式 *dt1* と *dt2* を作成したときに、*dt1* が *dt2* を参照し、*dt2* が *dt1* を参照していると起こります。

共通表式の表名 は、それを定義する選択ステートメント、INSERT ステートメント、または CREATE VIEW ステートメント内でのみ参照できます。

選択ステートメント、INSERT ステートメント、または CREATE VIEW ステートメントが非修飾表名を参照する場合、実際に参照されている表を判別するために以下の規則が適用されます。

- 非修飾名が、選択ステートメント で指定される 1 つ以上の共通表式名と一致する場合、名前は、有効範囲の最も内部にある共通表式を識別します。
- CREATE TRIGGER ステートメントおよび非修飾名が遷移表名と一致する場合、名前は遷移表を識別します。
- それ以外の場合、名前は永続表、一時表、またはデフォルトのスキーマに存在するビューを識別します。

共通表式 は、次の場合に使用できます。

- ビューの代わりとして (ビューを全般的に使用する必要がなく、位置指定の UPDATE または DELETE を使用しない場合に、ビューの作成を回避できます)
- スカラー全選択または非決定的な関数から派生する列によるグループ化を可能にするため
- 必要な結果表が変数に基づいているとき
- 同じ結果表を全選択 で共用するとき
- 再帰を使用して結果を得る必要がある場合

共通表式の *fullselect* の FROM 文節に自身への参照が含まれている場合、共通表式は反復表式 になります。反復を使用する照会は、部品表 (BOM)、予約システム、およびネットワーク計画などのアプリケーションをサポートするのに便利です。

再帰的共通表式 に当てはまる制約事項を以下にまとめます。

- 再帰サイクルの一部になっているそれぞれの全選択は、SELECT または SELECT ALL で開始する必要があります。SELECT DISTINCT は使用できません。
- UNION ALL セット演算子を指定しなければなりません。
- 列名 のリストは、共通表式 の表 *ID* の後に指定する必要があります。
- 最初の和集合の最初の全選択 (初期化全選択) の FROM 文節には、共通表式 自身の参照を含めてはなりません。
- 再帰サイクルを構成するそれぞれの *fullselect* は、集約関数、GROUP BY 文節、または HAVING 文節を含んでいてはなりません。
- それらの全選択 の FROM 文節には、再帰サイクルの一部になっている共通表式への参照を 1 つだけ組み込むことができます。
- 共通表式 で定義されている表を、共通表式 を定義する全選択 の副照会で参照することはできません。
- 共通表式 が右オペランドの場合、LEFT OUTER JOIN および FULL OUTER JOIN は使用できません。共通表式 が左オペランドの場合、RIGHT OUTER JOIN および FULL OUTER JOIN は使用できません。
- 再帰サイクルの一部になっている初期化全選択 以外の全選択 に ORDER BY 文節を組み込むことはできません。

反復全選択 で共通表式 の列名が参照される場合、結果列の属性は、結果列に関する規則を使用して決定します。詳しくは、 133 ページの『結果のデータ・タイプに関する規則』を参照してください。

search-clause

反復共通表式 の定義内の SEARCH 文節は、結果行が戻される順序の指定に使用されます。

SEARCH DEPTH FIRST

それぞれの親、または含まれている項目は、それを含む項目の前の結果内に現れます。

SEARCH BREADTH FIRST

兄弟項目は、従属項目の前にグループ化されます。

BY *column-name*,...

反復照会の親と子の関係を関連付ける列を識別します。それぞれの列名は、親の列を明確に識別するものでなければなりません。この列は、DATALINK 列または XML 列であってはなりません。明確な列参照の規則は、全選択の他の文節の場合と同じです。詳しくは、 166 ページの『あいまいさを避けるための列名修飾子』を参照してください。

列名 は、反復共通表式 の列名を識別するものでなければなりません。列名は修飾してはなりません。

SET *seq-column-name*

反復照会結果に現在行の序数を含む結果列の名前を指定します。順序列名 のデータ・タイプは BIGINT です。

順序列名 は、共通表式 を参照する外部全選択の ORDER BY 文節でのみ参照可能です。共通表式 を定義する全選択では、順序列名 を参照できません。

順序列名 は、使用列名 または循環列名 と同じであってはなりません。

cycle-clause

反復共通表式 の定義内の CYCLE 文節は、データの親と子の関係がループとなる際の、反復照会内の無限ループを防止するために使用されます。

CYCLE *column-name*,...

反復についての親/子の結合関係値を表す列のリストを指定します。照会からの新規行がある場合、循環が存在するかどうかを判別するために、反復照会結果内のこの行につながる既存の行に重複する値 (それらの列名ごと) がないかまず検査されます。

各列名 は、共通表式の結果列を識別するものでなければなりません。XML 列または DataLink 列以外の列を指定する必要があります。同じ列名を複数回指定することはできません。

SET *cycle-column-name*

反復照会内で循環が検出されたかどうかに基づいて設定される結果列の名前を指定します。

- 重複行が検出される場合 (データ内に循環が検出されたことを示す)、循環列名 は TO 定数 に設定されます。

- 重複行が検出されない場合 (データ内に循環が検出されなかったことを示す)、循環列名 は DEFAULT 定数 に設定されます。

循環列名 のデータ・タイプは CHAR(1) です。

行で循環データが検出されると、重複行は反復照会処理に戻されないためそれ以上反復は行われず、その照会の子の分岐は停止されます。提供された循環列名 をメインの全選択の結果セットに指定することにより、循環データが実際に存在するかどうかを判別することができ、必要な場合には訂正も行われます。

循環列名 は、使用列名 または順序列名 と同じであってはなりません。

共通表式 を定義する全選択で、循環列名 を参照することができます。

TO *constant*
データ内に循環が検出された場合に循環列 に割り当てる CHAR(1) 定数値を指定します。 TO 定数 は、DEFAULT 定数 と同じであってはなりません。

DEFAULT *constant*
データ内で循環が検出されなかった場合に循環列 に割り当てる CHAR(1) 定数値を指定します。 DEFAULT 定数 は、TO 定数 と同じであってはなりません。

USING *using-column-name*
CYCLE 列リストからの列で構成される一時的結果を識別します。一時的結果は、データベース・マネージャーが照会結果内の重複行を識別するために使用されます。

使用列名 は、循環列名 または順序列名 と同じであってはなりません。

再帰的共通表式を作成するときには、無限再帰サイクル (ループ) が生じる可能性があることに注意し、再帰サイクルが必ず終了するように記述してください。特にこのことは、関係するデータが周期的なものである場合に重要です。再帰的共通表式は、無限ループを回避する述部を組み込んでおく必要があります。次のものを再帰的共通表式に組み込むことをお勧めします。

- 反復 *fullselect* 内で、定数によって増分する整数列。
- 反復全選択 の WHERE 文節内で、「counter_col < constant」または「counter_col < :hostvar」という形式の述部。

再帰的共通表式でこの構文が見つからない場合は、警告が生成されます。

反復共通表式は、照会が以下を指定する場合には使用できません。

- 分散表
- 読み取りトリガーを指定する表
- 複数の物理ファイル・メンバー上に構築された論理ファイル

反復の例: 部品表

部品表 (BOM) アプリケーションは、多くのビジネス環境において一般的な要件となっています。 BOM アプリケーションの反復共通表式の機能を説明するために、部品、それに関連した副部品、および部品で必要な副部品の数量の表について考慮しましょう。

この例では、次のように表を作成します。

```
CREATE TABLE  PARTLIST
  ( PART     VARCHAR(8),
    SUBPART  VARCHAR(8),
    QUANTITY INTEGER )
```

この例で照会結果を出すために、PARTLIST 表に以下の値を取り込むと想定します。

```
PART     SUBPART  QUANTITY
-------- -------- -----------
   00       01         5
   00       05         3
   01       02         2
   01       03         3
   01       04         4
   01       06         3
   02       05         7
   02       06         6
   03       07         6
   04       08        10
   04       09        11
   05       10        10
   05       11        10
   06       12        10
   06       13        10
   07       14         8
   07       12         8
```

例 1: 単一レベルの部品展開

最初の例は、単一レベルの部品展開という例です。これは、「'01' によって識別される部品を作成するためにはどの部品が必要か?」という質問に答えます。リストには直接の副部品、副部品の副部品などが含まれます。部品が何度も使用される場合でも、その副部品は 1 度だけリストされます。

```
WITH RPL (PART, SUBPART, QUANTITY) AS
     (  SELECT ROOT.PART, ROOT.SUBPART, ROOT.QUANTITY
           FROM PARTLIST ROOT
           WHERE ROOT.PART = '01'
        UNION ALL
         SELECT CHILD.PART, CHILD.SUBPART, CHILD.QUANTITY
           FROM RPL PARENT, PARTLIST CHILD
           WHERE PARENT.SUBPART = CHILD.PART
      )
SELECT DISTINCT PART, SUBPART, QUANTITY
  FROM RPL
  ORDER BY PART, SUBPART, QUANTITY
```

上記の照会には、この照会の反復部分を表す共通表式 (名前 RPL によって識別される) が含まれています。これは、反復共通表式の基本的なエレメントを示しています。

初期化全選択とも呼ばれる UNION の第 1 オペランド (全選択) は、部品 '01' の直接の子を取得します。この全選択の FROM 文節はソース表を参照し、自己参照 (この場合は RPL) は行いません。この最初の全選択の結果は共通表式 RPL (反復 PARTLIST) に入ります。この例のように、UNION は常に UNION ALL でなければなりません。

UNION の第 2 オペランド (全選択) は、RPL を使用して、副部品の副部品を計算します。これは FROM 文節に、共通表式 RPL と、ソース表 (子) から RPL に含まれる現行結果の副部品 (親) の部品を結合したソース表を参照させることにより行います。この結果は RPL に戻されます。その後、UNION の第 2 オペランドは、子が存在しなくなるまで繰り返し使用されます。

この照会のメインの全選択の SELECT DISTINCT は、同じ部品/副部品が複数回リストされないようにします。

照会の結果は、次のようになります。

```
PART     SUBPART  QUANTITY
-------- -------- -----------
   01       02         2
   01       03         3
   01       04         4
   01       06         3
   02       05         7
   02       06         6
   03       07         6
   04       08        10
   04       09        11
   05       10        10
   05       11        10
   06       12        10
   06       13        10
   07       12         8
   07       14         8
```

結果から、部品 '01' は、'02' に至る ('06' まで続く) ことが分かります。さらに、部品 '06' へは 2 度 ('01' を介して直接 1 度、'02' を介してもう 1 度) 達していることも分かります。しかし出力では、そのサブコンポーネントは必要に応じて 1 度だけリストされています (これは SELECT DISTINCT の使用による結果です)。

例 2: 要約された部品展開

2 番目の例は、要約された部品展開です。ここで問題となるのは、部品 '01' の作成に必要な各部品の合計数量です。単一レベルの部品展開との主な違いは、数量の集約が必要であるという点です。最初の例は、部品に必要な副部品 (必要なときはいつでも) の数量を示しています。これは部品 '01' の作成に必要な副部品の数は示していません。

```
WITH RPL (PART, SUBPART, QUANTITY) AS
     (  SELECT ROOT.PART, ROOT.SUBPART, ROOT.QUANTITY
           FROM PARTLIST ROOT
           WHERE ROOT.PART = '01'
        UNION ALL
         SELECT PARENT.PART, CHILD.SUBPART, PARENT.QUANTITY*CHILD.QUANTITY
           FROM RPL PARENT, PARTLIST CHILD
           WHERE PARENT.SUBPART = CHILD.PART
      )
SELECT PART, SUBPART, SUM(QUANTITY) AS "Total QTY Used"
  FROM RPL
  GROUP BY PART, SUBPART
  ORDER BY PART, SUBPART
```

上記の照会では、反復共通表式の UNION の第 2 オペランドの選択リスト (名前 RPL によって識別される) が数量の集約を示しています。使用される副部品の数量を出すために、親の数量が子の親ごとの数量によって乗算されています。部品が別

の場所で複数回使用される場合、別の最終集約が必要となります。これは、共通表式 RPL に関するグループ化によって、またメインの全選択の選択リスト内の SUM 集約関数を使用して行われます。

照会の結果は、次のようになります。

```
PART     SUBPART  Total Qty Used
-------- -------- --------------
   01       02           2
   01       03           3
   01       04           4
   01       05          14
   01       06          15
   01       07          18
   01       08          40
   01       09          44
   01       10         140
   01       11         140
   01       12         294
   01       13         150
   01       14         144
```

出力の中の、副部品 '06' の行を考慮してみましょう。使用される合計数量の値 15 は、部品 '01' の数量 3 から直接と、そして部品 '02' の数量 6 (これは部品 '01' で 2 回必要になる) から来ています。

例 3: 深さの制御

照会に必要なものよりも多くのレベルの部品が表にあるとどうなるのだろう、という疑問が浮かぶかもしれません。つまり、「'01' で識別される部品を作成するために必要な最初の 2 つのレベルの部品はどれか?」という質問に答える照会がどのように作成されるか、ということです。分かりやすくするために、この例ではレベルが結果に組み込まれています。

```
WITH RPL ( LEVEL, PART, SUBPART, QUANTITY)
  AS ( SELECT 1, ROOT.PART, ROOT.SUBPART, ROOT.QUANTITY
         FROM PARTLIST ROOT
         WHERE ROOT.PART = '01'
       UNION ALL
       SELECT PARENT.LEVEL+1, CHILD.PART, CHILD.SUBPART, CHILD.QUANTITY
         FROM RPL PARENT, PARTLIST CHILD
         WHERE PARENT.SUBPART = CHILD.PART
         AND PARENT.LEVEL < 2
     )
SELECT PART, LEVEL, SUBPART, QUANTITY
  FROM RPL
```

この照会は例 1 と類似しています。オリジナルの部品からのレベルをカウントするために列 LEVEL が取り入れられました。初期化全選択の LEVEL 列の値は 1 に初期化されます。後続の全選択の親からのレベルは 1 ずつ増分されます。その後、結果のレベルの数値を制御するために、2 番目の全選択には、親レベルは 2 未満でなければならないという条件が組み込まれます。これにより、2 番目の全選択は第 2 レベルまでの子のみを処理することになります。

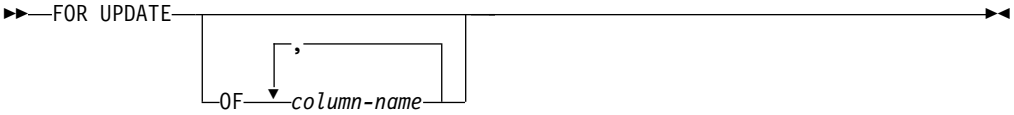
照会の結果は、次のようになります。

```
PART     LEVEL       SUBPART  QUANTITY
-------- ----------- -------- -----------
   01         1         02         2
   01         1         03         3
   01         1         04         4
```

```
01          1          06          3
02          2          05          7
02          2          06          6
03          2          07          6
04          2          08         10
04          2          09         11
06          2          12         10
06          2          13         10
```

UPDATE 文節

UPDATE 文節は、以後の位置指定 UPDATE ステートメントの割り当て文節内のターゲットとして指定できる列を識別します。各列名 は、それぞれ修飾のない名前でなければならず、全選択の最初の FROM 文節で識別されている表またはビューの 1 つの列を識別するものでなければなりません。この文節は、全選択の結果表が読み取り専用の場合は指定してはなりません。



column-name リストを使用して UPDATE 文節を指定し、拡張標識変数が使用不可の場合、*column-name* は更新可能な列でなければなりません。

列名 リストを指定せずに UPDATE 文節を指定する場合、暗黙の列名 リストが次のように決定されます。

- 拡張標識変数が使用可能な場合は、全選択の最初の FROM 文節で識別される表またはビューのすべての列。
- それ以外の場合は、全選択の最初の FROM 文節で識別される表またはビューのすべての更新可能な列。

SELECT ステートメント に UPDATE 文節を指定せず、その結果表が読み取り専用でない場合、結果として暗黙的に UPDATE 文節が使用されます。暗黙の列名 リストは、次のように決定されます。

- 拡張標識変数が使用可能な場合は、全選択の最初の FROM 文節で識別される表またはビューのすべての列が含まれます。
- それ以外の場合は、全選択の最初の FROM 文節で識別される表またはビューのすべての更新可能な列が含まれます。

全選択の結果表が読み取り専用である場合、(詳細は、 1353 ページの『DECLARE CURSOR』を参照)、または FOR READ ONLY 文節を使用する場合には、UPDATE 文節を指定してはなりません。

UPDATE 文節が使用されると、カーソルを参照する FETCH 操作は排他的行ロックが獲得されます。

READ-ONLY 文節

READ ONLY 文節は結果表が読み取り専用であり、そのため位置指定 UPDATE および DELETE ステートメントにカーソルを使用できないことを示します。

結果表の中には、最初から読み取り専用のものがあります。 (読み取り専用ビューに基づく表など。) そのような表についても FOR READ ONLY を指定することはできますが、この指定は効力をもちません。

更新や削除が許されている結果表の場合は、FOR READ ONLY を指定すると、データベース・マネージャーによるブロッキングが可能になり、排他的なロックが回避されるので、FETCH 操作のパフォーマンスが向上することがあります。例えば、FOR READ ONLY 文節または ORDER BY 文節のない動的 SQL ステートメントを含むプログラムでは、データベース・マネージャーは、カーソルを、UPDATE 文節が指定されている場合のようにオープンすることになります。

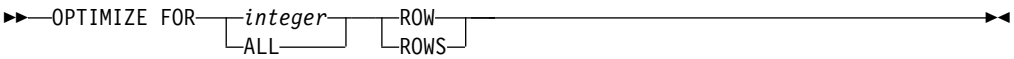
本来読み取り専用であるか、または FOR READ ONLY として指定されているかには関係なく、読み取り専用の結果表は、UPDATE または DELETE ステートメントで参照してはなりません。

選択データが他のどのジョブにからもロックされないようにするために、*ISOLATION* 文節 に USE AND KEEP EXCLUSIVE LOCKS のオプション構文を指定できます。これにより、行ロックの競合を引き起こさずに、選択データを後で更新または削除することができます。

代替構文**:** FOR READ ONLY の代わりに FOR FETCH ONLY を指定できます。

OPTIMIZE 文節

OPTIMIZE 文節 は、プログラムが整数 で指定された行数を超えて結果表から検索を行う意図はないことと想定するように、データベース・マネージャーに伝えます。この文節がない場合、またはキーワード ALL の指定がある場合は、データベース・マネージャーは、結果表の行をすべて検索するものと想定します。整数 行を最適化すると、パフォーマンスが向上する場合があります。データベース・マネージャーは、指定した行数に基づいて照会を最適化します。



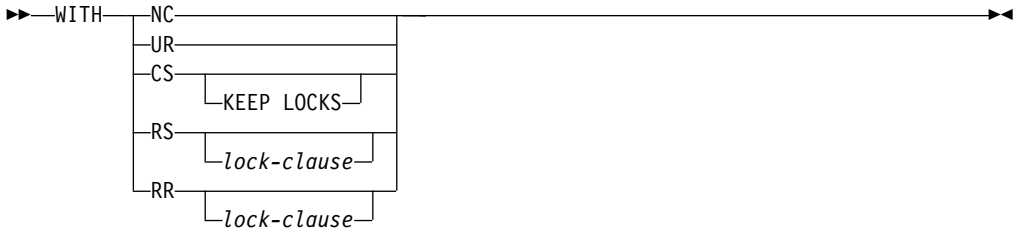
この文節によって、結果表や行を取り出す順序が変更されるわけではありません。任意の数の行を取り出すことができますが、指定の整数 回の取り出しの後は、パフォーマンスが下がる可能性があります。

integer の値は、正の整数 (ゼロを除く) でなければなりません。

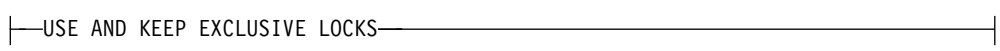
行アクセス制御は、副選択の権限 ID がアクセスできる行に影響するため、間接的に *OPTIMIZE* 文節 に影響します。

ISOLATION 文節

ISOLATION 文節 は、選択ステートメントを実行する分離レベルを指定します。



lock-clause:



RR 反復可能読み取り

USE AND KEEP EXCLUSIVE LOCKS
排他的行ロックが掛けられ、COMMIT または ROLLBACK ステートメントが実行されるまで保持されます。

RS 読み取り固定

USE AND KEEP EXCLUSIVE LOCKS
排他的行ロックが掛けられ、COMMIT または ROLLBACK ステートメントが実行されるまで保持されます。 USE AND KEEP EXCLUSIVE LOCKS 文節は、次の SQL ステートメントの *ISOLATION* 文節 でのみ使用できます。

- DECLARE CURSOR
- FOR
- *select-statement*
- SELECT INTO
- ATTRIBUTES ストリングの PREPARE

この文節は、カーソルが更新可能な場合は使用できません。

CS カーソル固定

KEEP LOCKS
KEEP LOCKS 文節は、獲得したどの読み取りロックも長期間保持することを指定するものです。通常は、読み取りロックは、次の行が読み取られると解除されます。 ISOLATION 文節がカーソルに関連付けられている場合は、ロックは、そのカーソルがクローズされるか、または COMMIT か ROLLBACK ステートメントが実行されるまで保持されます。関連付けられていない場合は、ロックは、この SQL ステートメントの完了まで保持されます。

UR 非コミット読み取り

NC コミットなし

ISOLATION 文節 を指定しなかった場合は、デフォルトの分離レベルが使用されます。ただし、非コミット読み取りのデフォルトの分離レベルは例外です。デフォルトの判別方法については、 32 ページの『分離レベル』を参照してください。

排他ロック: USE AND KEEP EXCLUSIVE LOCKS 文節を使用の際はご注意ください。これが指定されると、行に対して掛けられる排他的行ロックにより、作業単位の終了まで、COMMIT(*CS)、COMMIT(*RS)、および COMMIT(*RR) を実行するその他のユーザーが、それらの行に同時にアクセスすることができなくなります。 COMMIT(*NC) または COMMIT(*UR) を実行するユーザーの同時アクセスは可能です。

同義のキーワード : 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード NONE を NC の同義語として使用することができます。
- キーワード CHG を UR の同義語として使用することができます。
- キーワード ALL を RS の同義語として使用することができます。

concurrent-access-resolution-clause

concurrent-access-resolution-clause は、ステートメントで使用する同時アクセスの解決を指定します。

SKIP LOCKED DATA
選択ステートメント、検索 UPDATE ステートメント (MERGE ステートメント内の検索更新操作を含む)、または PREPARE ステートメント内で準備される検索 DELETE ステートメントが、互換性のないロックを他トランザクションが行に対して保持している場合には、その行をスキップすることを指定します。これらの行は、このステートメントで指定されたどのアクセス対象の表に属していても構いません。SKIP LOCKED DATA を使用できるのは、分離レベル NC、UR、CS、または RS が有効な場合に限定されます。

SKIP LOCKED DATA は、有効な分離レベルが RR であるときに指定されると、それは無視されます。

USE CURRENTLY COMMITTED
データが更新プロセスまたは削除プロセスの過程にあるときに、データベース・マネージャーが現時点でコミット済みのバージョンのデータを適用可能なスキャンに使用できるように指定します。挿入プロセスの過程にある行は、スキップできます。この節は、分離レベルが KEEP LOCKS 節を指定しない CS の場合に可能であれば適用され、その他の場合には無視されます。適用可能なスキャンには、読み取り専用スキャンが含まれます。

WAIT FOR OUTCOME
更新プロセスまたは削除プロセスにあるデータを検出するときに、コミットまたはロールバックを待機するように指定します。挿入プロセスの過程にあることが検出される行は、スキップされません。この節は、分離レベルが CS または RS のときに、可能であれば適用されます。分離レベル NC、UR、または RR が有効な場合は無視されます。

SELECT ステートメントの例

SELECT ステートメントはさまざまな方法で使用することができます。

例 1

表 EMPLOYEE から、すべての列および行を選択します。

```
SELECT * FROM EMPLOYEE
```

例 2

表 PROJECT から、プロジェクト名 (PROJNAME)、開始日付 (PRSTDATE)、および終了日付 (PRENDATE) を選択します。結果表を終了日付によって順序付けして、終了日付の最も早いプロジェクトが先頭になるようにしています。

```
SELECT PROJNAME, PRSTDATE, PRENDATE
  FROM PROJECT
  ORDER BY PRENDATE DESC
```

例 3

表 EMPLOYEE のすべての部門について、部門番号 (WORKDEPT) および給与 (SALARY) の部門別平均額を選択します。結果表は、部門別平均給与によって昇順に並べます。

```
SELECT WORKDEPT, AVG(SALARY)
  FROM EMPLOYEE
  GROUP BY WORKDEPT
  ORDER BY AVGSAL
```

例 4

UP_CUR という名前のカーソルを C プログラムで使用するように宣言します。これによって、PROJECT 表内の開始日付 (PRSTDATE) と終了日付 (PRENDATE) の列が更新されます。このプログラムでは、行ごとのプロジェクト番号 (PROJNO) の値と一緒に PRSTDATE と PRENDATE の両方の値を受け取らなければなりません。宣言では、照会のアクセス・パスを最大 2 行の検索に最適化するように指定しています。このように最適化しても、プログラムは結果表から 3 行以上検索することができます。ただし、3 行以上検索すると、パフォーマンスが下がる可能性があります。

```
EXEC SQL  DECLARE UP_CUR CURSOR FOR
            SELECT PROJNO, PRSTDATE, PRENDATE
              FROM PROJECT
              FOR UPDATE OF PRSTDATE, PRENDATE
              OPTIMIZE FOR 2 ROWS ;
```

例 5

分離レベルが読み取り固定 (RS) の表から項目を選択します。

```
SELECT NAME, SALARY
  FROM PAYROLL
  WHERE DEPT = 704
  WITH RS
```

例 6

この例は、式 SALARY+BONUS+COMM に TOTAL_PAY という名前を付けます。

```
SELECT SALARY+BONUS+COMM AS TOTAL_PAY
  FROM EMPLOYEE
  ORDER BY TOTAL_PAY
```

例 7

販売担当者の従業員数と給与、およびそれぞれの部門の平均給与と人数を調べます。平均給与が最高の部門の平均給与もリストします。

この場合、共通表式を使用すれば、DINFO ビューを正規ビューとして作成するオーバーヘッドを節約できます。全選択の残りのコンテキストから、ビューでは、販売担当者の部門の行だけを考慮すれば済みます。

```
WITH
   DINFO (DEPTNO, AVGSALARY, EMPCOUNT) AS
     (SELECT OTHERS.WORKDEPT, AVG(OTHERS.SALARY), COUNT(*)
        FROM EMPLOYEE OTHERS
        GROUP BY OTHERS.WORKDEPT),
   DINFOMAX AS
     (SELECT MAX(AVGSALARY) AS AVGMAX
        FROM DINFO)
SELECT THIS_EMP.EMPNO, THIS_EMP.SALARY, DINFO.AVGSALARY, DINFO.EMPCOUNT,
                          DINFOMAX.AVGMAX
  FROM EMPLOYEE THIS_EMP, DINFO, DINFOMAX
  WHERE THIS_EMP.JOB = 'SALESREP'
  AND THIS_EMP.WORKDEPT = DINFO.DEPTNO
```

例 8

2000 年の第 1 四半期の各月の最終金曜日の、カリフォルニア州の各加入者 (SNO) の平均料金を検索します。 SNO に基づいて結果をグループ化します。各 MONTHnn 表に、SNO、CHARGES、および DATE の列があります。 CUST 表には、SNO および STATE の列があります。

```
SELECT V.SNO, AVG( V.CHARGES)
  FROM CUST, LATERAL (
    SELECT SNO, CHARGES, DATE
    FROM MONTH1
    WHERE DATE BETWEEN '01/01/2000' AND '01/31/2000'
        UNION ALL
    SELECT SNO, CHARGES, DATE
    FROM MONTH2
    WHERE DATE BETWEEN '02/01/2000' AND '02/29/2000'
        UNION ALL
    SELECT SNO, CHARGES, DATE
    FROM MONTH3
    WHERE DATE BETWEEN '03/01/2000' AND '03/31/2000'
    ) AS  V (SNO, CHARGES, DATE)
  WHERE CUST.SNO=V.SNO
  AND CUST.STATE='CA'
  AND DATE IN ('01/28/2000','02/25/2000','03/31/2000')
  GROUP BY V.SNO
```

第 7 章 ステートメント

このセクションでは、SQL ステートメントの構文図、セマンティックの説明、規則、および使用例を示します。

これらのステートメントを次の表にリストします。

表 *69. SQL* スキーマ・ステートメント

SQL ステートメント	説明	ページ
ALTER FUNCTION (外部スカラー)	外部スカラー関数の記述を変更します。	884 ページの『ALTER FUNCTION (外部スカラー)』
ALTER FUNCTION (外部表)	外部表関数の記述を変更します。	890 ページの『ALTER FUNCTION (外部表)』
ALTER FUNCTION (SQL スカラー)	SQL スカラー関数の記述を変更します。	897 ページの『ALTER FUNCTION (SQL スカラー)』
ALTER FUNCTION (SQL 表)	SQL 表関数の記述を変更します。	906 ページの『ALTER FUNCTION (SQL 表)』
ALTER MASK	マスクの状態を変更します。	915 ページの『ALTER MASK』
ALTER PERMISSION	許可の状態を変更します。	917 ページの『ALTER PERMISSION』
ALTER PROCEDURE (外部)	外部プロシージャーの記述を変更します。	919 ページの『ALTER PROCEDURE (外部)』
ALTER PROCEDURE (SQL)	SQL プロシージャーの記述を変更します。	925 ページの『ALTER PROCEDURE (SQL)』
ALTER SEQUENCE	シーケンスの記述を変更します。	936 ページの『ALTER SEQUENCE』
ALTER TABLE	表の記述を変更します。	942 ページの『ALTER TABLE』
ALTER TRIGGER	トリガーの記述を変更します。	1004 ページの『ALTER TRIGGER』
COMMENT	オブジェクトの記述に対してコメントの追加または置換を行う。	1027 ページの『COMMENT』
CREATE ALIAS	別名を作成します。	1065 ページの『CREATE ALIAS』
CREATE FUNCTION	ユーザー定義の関数を作成します。	1070 ページの『CREATE FUNCTION』
CREATE FUNCTION (外部スカラー)	外部スカラー関数を作成します。	1076 ページの『CREATE FUNCTION (外部スカラー)』

表 69. SQL スキーマ・ステートメント (続き)

SQL ステートメント	説明	ページ
CREATE FUNCTION (外部表)	外部表関数を作成します。	1100 ページの『CREATE FUNCTION (外部表)』
CREATE FUNCTION (ソース派生)	別の既存のスカラー関数や集約関数にもとづいて、ユーザー定義の関数を作成します。	1122 ページの『CREATE FUNCTION (ソース派生)』
CREATE FUNCTION (SQL スカラー)	SQL スカラー関数を作成します。	1135 ページの『CREATE FUNCTION (SQL スカラー)』
CREATE FUNCTION (SQL 表)	SQL 表関数を作成します。	1151 ページの『CREATE FUNCTION (SQL 表)』
CREATE INDEX	表上の索引を作成します。	1166 ページの『CREATE INDEX』
CREATE MASK	表の列マスクを作成します。	1175 ページの『CREATE MASK』
CREATE PERMISSION	表の行の許可を作成します。	1182 ページの『CREATE PERMISSION』
CREATE PROCEDURE	プロシージャーを作成します。	1187 ページの『CREATE PROCEDURE』
CREATE PROCEDURE (外部)	外部プロシージャーを作成します。	1189 ページの『CREATE PROCEDURE (外部)』
CREATE PROCEDURE (SQL)	SQL プロシージャーを作成します。	1208 ページの『CREATE PROCEDURE (SQL)』
CREATE SCHEMA	スキーマ、およびそのスキーマ内の一連のオブジェクトを作成します。	1224 ページの『CREATE SCHEMA』
CREATE SEQUENCE	シーケンスを作成します。	1230 ページの『CREATE SEQUENCE』
CREATE TABLE	表を作成します。	1238 ページの『CREATE TABLE』
CREATE TRIGGER	トリガーを作成します。	1302 ページの『CREATE TRIGGER』
CREATE TYPE	タイプを作成します。	1328 ページの『CREATE TYPE (特殊)』
CREATE VARIABLE	グローバル変数を作成します。	1336 ページの『CREATE VARIABLE』
CREATE VIEW	1 つ以上の表あるいはビューの、1 つのビューを作成します。	1342 ページの『CREATE VIEW』

表 69. SQL スキーマ・ステートメント (続き)

SQL ステートメント	説明	ページ
DROP	別名、関数、索引、パッケージ、プロシージャー、スキーマ、シーケンス、表、トリガー、タイプ、変数、ビューまたは XSR オブジェクトを除去します。	1441 ページの『DROP』
GRANT (関数特権またはプロシージャー特権)	関数やプロシージャーに対する特権を付与します。	1517 ページの『GRANT (関数特権またはプロシージャー特権)』
GRANT (変数特権)	グローバル変数に関する特権を認可します。	1545 ページの『GRANT (変数特権)』
GRANT (パッケージ特権)	パッケージに関する特権を認可します。	1526 ページの『GRANT (パッケージ特権)』
GRANT (スキーマ特権)	スキーマに対する特権を付与する。	1529 ページの『GRANT (スキーマ特権)』
GRANT (シーケンス特権)	シーケンスに関する特権を認可します。	1532 ページの『GRANT (シーケンス特権)』
GRANT (表またはビューの特権)	表またはビューに関する特権を認可します。	1535 ページの『GRANT (表またはビューの特権)』
GRANT (タイプ特権)	タイプに関する特権を認可します。	1542 ページの『GRANT (タイプ特権)』
GRANT (XML スキーマ特権)	XML スキーマに関する特権を認可します。	1548 ページの『GRANT (XML スキーマ特権)』
LABEL	オブジェクトの記述に対してラベルの追加または置換を行う。	1570 ページの『LABEL』
RENAME	表、ビュー、または索引の名前を変更します。	1630 ページの『RENAME』
REVOKE (関数特権またはプロシージャー特権)	関数やプロシージャーに対する特権を取り消します。	1633 ページの『REVOKE (関数特権またはプロシージャー特権)』
REVOKE (変数特権)	グローバル変数に関する特権を取り消します。	1655 ページの『REVOKE (変数特権)』
REVOKE (パッケージ特権)	パッケージ内のステートメントを実行する特権を取り消します。	1641 ページの『REVOKE (パッケージ特権)』
REVOKE (スキーマ特権)	スキーマに対する特権を取り消す。	1644 ページの『REVOKE (スキーマ特権)』
REVOKE (シーケンス特権)	シーケンスに関する特権を取り消します。	1646 ページの『REVOKE (シーケンス特権)』
REVOKE (表またはビューの特権)	表またはビューに関する特権を取り消します。	1648 ページの『REVOKE (表またはビューの特権)』

表 69. SQL スキーマ・ステートメント (続き)

SQL ステートメント	説明	ページ
REVOKE (タイプ特権)	タイプを使用する特権を取り消します。	1652 ページの『REVOKE (タイプ特権)』
REVOKE (XML スキーマ特権)	XML スキーマに関する特権を取り消します。	1658 ページの『REVOKE (XML スキーマ特権)』
TRANSFER OWNERSHIP	オブジェクトの所有権を別の権限 ID に移します。	1744 ページの『TRANSFER OWNERSHIP』

表 70. SQL データ変更ステートメント

SQL ステートメント	説明	ページ
DELETE	表から 1 つ以上の行を削除します。	1404 ページの『DELETE』
INSERT	表に 1 行または複数行を挿入します。	1556 ページの『INSERT』
MERGE	ソース表のデータを使用して、ターゲット表を更新します。	1582 ページの『MERGE』
TRUNCATE	表からすべての行を削除する。	1747 ページの『TRUNCATE』
UPDATE	表の 1 つ以上の行にある、1 つ以上の列の値を更新します。	1750 ページの『UPDATE』

表 71. SQL データ・ステートメント

SQL ステートメント	説明	ページ
	すべての SQL データ変更ステートメント	表 70
ALLOCATE CURSOR	結果セット・ロケーター変数が示す結果セットにカーソルを割り振ります。	880 ページの『ALLOCATE CURSOR』
ASSOCIATE LOCATORS	プロシージャーが戻すそれぞれの結果セットに関する、結果セット・ロケーター値を入手します。	1007 ページの『ASSOCIATE LOCATORS』
CLOSE	カーソルをクローズします。	1025 ページの『CLOSE』
DECLARE CURSOR	SQL カーソルを定義します。	1353 ページの『DECLARE CURSOR』
FETCH	結果表の行にカーソルを位置付けます。また、結果表の 1 行または複数行の値を変数に割り当てることもできます。	1466 ページの『FETCH』
FREE LOCATOR	LOB ロケーター変数とその値との関連を除去します。	1475 ページの『FREE LOCATOR』

表 71. SQL データ・ステートメント (続き)

SQL ステートメント	説明	ページ
HOLD LOCATOR	作業単位が変わっても、LOB ロケーター変数が値との関連を保持できるようにします。	1551 ページの『HOLD LOCATOR』
LOCK TABLE	同時に実行されているプロセスによる表の変更を防止するか、または表の使用を防止します。	1580 ページの『LOCK TABLE』
OPEN	カーソルをオープンします。	1596 ページの『OPEN』
REFRESH TABLE	マテリアライズ照会表のデータをリフレッシュします。	1624 ページの『REFRESH TABLE』
SELECT	照会を実行します。	1669 ページの『SELECT』
SELECT INTO	変数に値を割り当てます。	1670 ページの『SELECT INTO』
SET 遷移変数	遷移変数に値を割り当てます。	SET 遷移変数
SET 変数	変数に値を割り当てます。	1737 ページの『SET 変数』
VALUES	照会を実行します。	1764 ページの『VALUES』
VALUES INTO	1 行だけで構成される結果表を指定し、それらの値を変数に割り当てます。	1765 ページの『VALUES INTO』

表 72. SQL トランザクション・ステートメント

SQL ステートメント	説明	ページ
COMMIT	作業単位を終了させ、その作業単位によって行われたデータベースの変更をコミットします。	1038 ページの『COMMIT』
RELEASE SAVEPOINT	作業単位内のセーブポイントを解放します。	1629 ページの『RELEASE SAVEPOINT』
ROLLBACK	作業単位を終了させ、その作業単位によって行われた、または指定したセーブポイント以降に行われたデータベースの変更をバックアウトします。	1661 ページの『ROLLBACK』
SAVEPOINT	作業単位内にセーブポイントをセットします。	1666 ページの『SAVEPOINT』
SET TRANSACTION	現行作業単位の分離レベルを変更します。	1733 ページの『SET TRANSACTION』

表 73. SQL 接続ステートメント

SQL ステートメント	説明	ページ
CONNECT (タイプ 1)	アプリケーション・サーバーに接続し、リモート作業単位のルール (規則) を確立します。	1053 ページの『CONNECT (タイプ 1)』

表 73. SQL 接続ステートメント (続き)

SQL ステートメント	説明	ページ
CONNECT (タイプ 2)	アプリケーション・サーバーに接続し、アプリケーション指向の分散作業単位のルール (規則) を確立します。	1059 ページの『CONNECT (タイプ 2)』
DISCONNECT	1 つ以上の接続をただちに終了させます。	1438 ページの『DISCONNECT』
RELEASE (接続)	1 つ以上の接続を解放ペンディング状態にします。	1626 ページの『RELEASE (接続)』
SET CONNECTION	既存の接続の 1 つを識別することによって、プロセスのアプリケーション・サーバーを確立する。	1673 ページの『SET CONNECTION』

表 74. SQL 動的ステートメント

SQL ステートメント	説明	ページ
ALLOCATE DESCRIPTOR	SQL 記述子を割り振ります。	882 ページの『ALLOCATE DESCRIPTOR』
コンパウンド (動的)	他のステートメントを 1 つの実行可能ルーチンの中にグループとしてまとめます。	1042 ページの『コンパウンド (動的)』
DEALLOCATE DESCRIPTOR	SQL 記述子の割り振りを解除します。	1352 ページの『DEALLOCATE DESCRIPTOR』
DESCRIBE	準備済みステートメントの結果列を記述します。	1413 ページの『DESCRIBE』
DESCRIBE CURSOR	カーソルを記述します。	1419 ページの『DESCRIBE CURSOR』
DESCRIBE INPUT	準備済みステートメントの入力パラメーター・マーカーを記述します。	1422 ページの『DESCRIBE INPUT』
DESCRIBE PROCEDURE	プロシージャーが戻す結果セットについて記述します。	1426 ページの『DESCRIBE PROCEDURE』
DESCRIBE TABLE	表またはビューの列を記述します。	1433 ページの『DESCRIBE TABLE』
EXECUTE	準備済みの SQL ステートメントを実行します。	1458 ページの『EXECUTE』
EXECUTE IMMEDIATE	SQL ステートメントを準備して、実行します。	1463 ページの『EXECUTE IMMEDIATE』
GET DESCRIPTOR	SQL 記述子から情報を取得します。	1476 ページの『GET DESCRIPTOR』
PREPARE	SQL ステートメントを実行できるように準備します。	1603 ページの『PREPARE』
SET DESCRIPTOR	SQL 記述子の項目を設定します。	1688 ページの『SET DESCRIPTOR』

表 *75. SQL* セッション・ステートメント

SQL ステートメント	説明	ページ
DECLARE GLOBAL TEMPORARY TABLE	宣言済み一時表を定義する。	1364 ページの『DECLARE GLOBAL TEMPORARY TABLE』
SET CURRENT DEBUG MODE	CURRENT DEBUG MODE 特殊レジスターに値を割り当てます。	1677 ページの『SET CURRENT DEBUG MODE』
SET CURRENT DECFLOAT ROUNDING MODE	CURRENT DECFLOAT ROUNDING MODE 特殊レジスターに値を割り当てます。	1679 ページの『SET CURRENT DECFLOAT ROUNDING MODE』
SET CURRENT DEGREE	CURRENT DEGREE 特殊レジスターに値を割り当てます。	1682 ページの『SET CURRENT DEGREE』
SET CURRENT IMPLICIT XMLPARSE OPTION	CURRENT IMPLICIT XMLPARSE OPTION 特殊レジスターに値を割り当てます。	1685 ページの『SET CURRENT IMPLICIT XMLPARSE OPTION』
SET CURRENT TEMPORAL SYSTEM_TIME	CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターに値を割り当てます	1687 ページの『SET CURRENT TEMPORAL SYSTEM_TIME』
SET ENCRYPTION PASSWORD	デフォルトの暗号化パスワードおよび暗号化パスワード・ヒントに値を割り当てます。	1693 ページの『SET ENCRYPTION PASSWORD』
SET PATH	CURRENT PATH 特殊レジスターに値を割り当てます。	1720 ページの『SET PATH』
SET SCHEMA	CURRENT SCHEMA 特殊レジスターに値を割り当てます。	1727 ページの『SET SCHEMA』
SET SESSION AUTHORIZATION	ジョブのユーザーと USER 特殊レジスターを変更します。	1730 ページの『SET SESSION AUTHORIZATION』

表 *76. SQL* 組み込みホスト言語ステートメント

SQL ステートメント	説明	ページ
BEGIN DECLARE SECTION	SQL 宣言セクションの開始を示します。	1013 ページの『BEGIN DECLARE SECTION』
CALL	プロシージャーを呼び出します。	1015 ページの『CALL』
DECLARE PROCEDURE	外部プロシージャーを定義します。	1388 ページの『DECLARE PROCEDURE』
DECLARE STATEMENT	準備済み SQL ステートメントを識別するための名前を宣言します。	1399 ページの『DECLARE STATEMENT』
DECLARE VARIABLE	ホスト変数に対して、デフォルト値以外のサブタイプまたは NORMALIZED を宣言します。	1401 ページの『DECLARE VARIABLE』
END DECLARE SECTION	SQL 宣言セクションの終わりを示します。	1457 ページの『END DECLARE SECTION』

表 76. SQL 組み込みホスト言語ステートメント (続き)

SQL ステートメント	説明	ページ
GET DIAGNOSTICS	直前に実行された SQL ステートメントに関する情報を取得します。	1489 ページの『GET DIAGNOSTICS』
INCLUDE	ソース・プログラムに宣言を挿入します。	1553 ページの『INCLUDE』
SET OPTION	SQL ステートメントの処理に関するオプションを設定します。	1696 ページの『SET OPTION』
SET RESULT SETS	プロシージャーの中の結果セットを識別します。	1724 ページの『SET RESULT SETS』
SIGNAL	エラー条件または警告条件を通知します。	1740 ページの『SIGNAL』
WHENEVER	SQL 戻りコードに基づいて、行うべきアクションを定義します。	1768 ページの『WHENEVER』

表 77. SQL 制御ステートメント

SQL ステートメント	説明	ページ
割り当てステートメント	出力パラメーターまたはローカル変数に値を割り当てます。	1782 ページの『割り当てステートメント』
CALL	プロシージャーを呼び出します。	1787 ページの『CALL ステートメント』
CASE	複数の条件に基づいて実行パスを選択します。	1789 ページの『ケース (CASE) ステートメント』
複合 (compound) ステートメント	他のステートメントを 1 つの SQL ルーチンの中にグループとしてまとめます。	1791 ページの『複合 (compound) ステートメント』
FOR	表のそれぞれの行ごとにステートメントを実行します。	1802 ページの『FOR ステートメント』
GET DIAGNOSTICS	直前に実行された SQL ステートメントに関する情報を取得します。	1804 ページの『GET DIAGNOSTICS ステートメント』
GOTO	SQL ルーチンまたはトリガーの中のユーザー定義のラベルに分岐します。	1812 ページの『GOTO ステートメント』
IF	条件の真理値に基づいて条件付きで実行します。	1814 ページの『IF ステートメント』
ITERATE	制御のフローをラベル付きループの先頭に戻します。	1820 ページの『ITERATE ステートメント』
LEAVE	ブロックまたはループを終了することによって実行を継続します。	1822 ページの『終了 (LEAVE) ステートメント』
LOOP	ステートメントの実行を繰り返します。	1824 ページの『ループ (LOOP) ステートメント』

表 77. *SQL* 制御ステートメント (続き)

SQL ステートメント	説明	ページ
REPEAT	ステートメントの実行を繰り返します。	1828 ページの『反復 (REPEAT) ステートメント』
RESIGNAL	エラー条件または警告条件を再通知します。	1830 ページの『再通知 (RESIGNAL) ステートメント』
RETURN	ルーチンから戻ります。	1835 ページの『戻り (RETURN) ステートメント』
SIGNAL	エラー条件または警告条件を通知します。	1838 ページの『通知 (SIGNAL) ステートメント』
WHILE	指定された条件が真である間、ステートメントの実行を繰り返します。	1843 ページの『WHILE ステートメント』

SQL ステートメントの呼び出し方法

この章で説明する SQL ステートメントは、実行可能 ステートメントと実行不能 ステートメントに類別されます。各ステートメントの説明の呼び出し の項に、そのステートメントが実行可能かを示しています。

実行可能ステートメント は、次のいずれかの方法で呼び出すことができます。

- アプリケーション・プログラムの中に組み込む。
- 動的に準備して実行する。
- 対話方式で呼び出す。

注**:** REXX に組み込まれたステートメントや RUNSQLSTM を使用して処理されるステートメントは、動的に準備され、実行されます。

ステートメントによっては、これらの方法をすべて使用できる場合もあれば、一部だけを使用できる場合もあります。各ステートメントの説明の呼び出し の項には、呼び出しにどのような方法を使用できるかを示しています。

実行不能ステートメント は、アプリケーション・プログラムの中に組み込むことだけが可能です。

アプリケーション・プログラムへのステートメントの組み込み

SQL ステートメントは、CRTSQLCBL、CRTSQLCBLI、CRTSQLCI、CRTSQLCPPI、CRTSQLPLI、CRTSQLRPG、または CRTSQLRPGI コマンドを使用することでプリコンパイラーに実行依頼されるソース・プログラムの中に組み込むことができます。このようなステートメントは、プログラムに組み込まれた と表現されます。

組み込みステートメントは、ホスト言語のステートメントを使用できる環境であれば、プログラム中のどこにでも入れることができます。そのステートメントが SQL

ステートメントであることを示すには、各組み込みステートメントそれぞれの前に、以下のように 1 つ以上のキーワードを置く必要があります。

- C、COBOL、PL/I、および RPG では、各組み込みステートメントそれぞれの前には、キーワード EXEC と SQL がなければなりません。
- Java では、各組み込みステートメントそれぞれの前には、キーワード #sql がなければなりません。
- REXX では、組み込みステートメントそれぞれの前には、キーワード EXECSQL がなければなりません。

実行可能ステートメント

アプリケーション・プログラムに組み込まれた実行可能ステートメントは、同じ場所にホスト言語のステートメントが指定されていればそのホスト言語ステートメントが実行されることになるすべての時点で、実行されます。つまり、ループの中にあるステートメントは、そのループが実行されるたびに実行されます。また、条件付き構造の中にあるステートメントは、その条件が満たされた場合にのみ実行されます。

組み込みステートメントでは、変数を参照することができます。組み込みステートメントでは、以下の 2 つの目的で変数を参照します。

- 入力として使用する (ステートメントの実行時に変数の現行値を使用する)。
- 出力として使用する (ステートメントの実行結果として、変数に新しい値を割り当てる)。

特に、式や述部内の変数の参照は、すべて、それらの変数の現行値によって実際に置換されます。つまり、それらの変数は入力として使用されます。その他の参照の扱い方については、ステートメントごとに個別に説明します。

SQL 戻り状態または SQL 戻りコードのテストで、すべての実行可能ステートメントをたどります。代わりに、WHENEVER ステートメント (このステートメント自体は実行不能ステートメント) を使用して、組み込みステートメントの実行直後の制御の流れを変えることができます。

SQL ステートメントで参照しているオブジェクトは、ステートメントを準備する時点では存在していなくても構いません。

実行不能ステートメント

組み込み実行不能ステートメントは、プリコンパイラーによってのみ処理されます。そのステートメントで検出されたすべてのエラーは、プリコンパイラーによって報告されます。実行不能ステートメントは、決して実行されることはありません。アプリケーション・プログラムの実行可能ステートメントの中に、実行不能ステートメントが入っている場合、その実行不能ステートメントはノー・オペレーションとして扱われます。したがって、そのようなステートメントに続けて、SQL 戻りコードのテストを行ってはなりません。

動的な準備と実行

アプリケーション・プログラムは、変数に入れられた文字ストリングの形式で動的に SQL ステートメントを構築することができます。通常、ステートメントは、プログラムで使用できるデータ (例えば、ワークステーションからの入力) から構築されます。

このようなステートメントは、(組み込まれた) ステートメント PREPARE を使用して実行の準備を行うことができ、(組み込まれた) ステートメント EXECUTE によって実行することができます。代わりに、(組み込まれた) ステートメント EXECUTE IMMEDIATE を使用して、1 つのステップでステートメントの準備と実行を行うことができます。 Java では、Statement、PreparedStatement、および CallableStatement クラスによってステートメントの実行の準備を行い、それぞれの execute() メソッドによって実行することができます。

動的に準備するステートメントにホスト変数への参照を組み込むことはできません。その代わりに、パラメーター・マーカーをそのステートメントに組み込むことができます。パラメーター・マーカーに関する規則については、 1603 ページの『PREPARE』を参照してください。準備されたステートメントが実行されると、パラメーター・マーカーは、EXECUTE ステートメントで指定された変数の現行値によって事実上置き換えられます。この置き換えに関する規則については、 1458 ページの『EXECUTE』を参照してください。準備済みのステートメントは、変数の異なる値を使用して、何回でも実行することができます。 EXECUTE IMMEDIATE では、パラメーター・マーカーを使用することはできません。

C、COBOL、PL/I、REXX、および RPG では、ステートメントが正常に実行されたか否かは、その EXECUTE (または EXECUTE IMMEDIATE) ステートメントの実行後に、独立型 SQLCODE または SQLSTATE に戻される値によって示されます。この SQL 戻りコードは、組み込みステートメントに関する上記の説明に従って検査してください。詳細については、 876 ページの『SQL 診断情報』の節を参照してください。 Java では、ステートメントの実行の成否は、Java 例外によって処理されます。

select ステートメントの静的呼び出し

選択ステートメント は、(実行不能) ステートメント DECLARE CURSOR の一環として組み込むことができます。

このようなステートメントは、該当のカーソルが、(組み込まれた) ステートメント OPEN によってオープンされるたびに実行されます。カーソルのオープン後、結果表は、FETCH ステートメントを繰り返して実行することにより 1 回に 1 行ずつ、または複数行 FETCH ステートメントを使用して、1 回に複数行を検索することができます。

このような方法で使用する場合、選択ステートメント には変数への参照を含めることができます。このような参照は、OPEN の実行時点における該当の変数の値によって事実上置き換えられます。

select ステートメントの動的呼び出し

アプリケーション・プログラムは、変数に入れられた文字ストリングの形式で選択ステートメント を動的に構築することができます。

一般的に、このようなステートメントはそのプログラムで使用可能なデータ (例えば、ワークステーションから得られる照会) から構築されます。構築されたステートメントは、(組み込まれた) ステートメント OPEN によってそのカーソルがオープンされるたびに実行されます。カーソルのオープン後、結果表は、FETCH ステートメントを繰り返して実行することにより 1 回に 1 行ずつ、または複数行 FETCH ステートメントを使用して、1 回に複数行を検索することができます。

このような方法で使用する場合、選択ステートメント には、変数に対する参照を含めてはなりません。ホスト変数に対する参照の代わりに、パラメーター・マーカーを入れることができます。パラメーター・マーカーに関する規則については、1603 ページの『PREPARE』を参照してください。パラメーター・マーカーは、実際には OPEN ステートメントで指定されている変数の値に置き換えられます。この置き換えに関する規則については、 1596 ページの『OPEN』を参照してください。

対話式呼び出し

ワークステーションから SQL ステートメントを入力する機能は、データベース・マネージャーのアーキテクチャーの一部です。このように入力されたステートメントは、対話式に出されたと言います。

パラメーター・マーカーや変数に対する参照は、アプリケーション・プログラムの文脈中でのみ意味を持つので、対話式に出すステートメントは、パラメーター・マーカーや変数への参照を含まない実行可能ステートメントでなければなりません。

Db2 for i データベースには、この機能に対応する構造化照会言語開始 (STRSQL) コマンド、照会管理機能開始 (STRQM) コマンド、System i ナビゲーターの SQL スクリプト実行サポートが用意されています。また、他のプロダクトを使用して SQL ステートメントを入力することもできます。

SQL 診断情報

データベース・マネージャーは、診断域を使用して、実行可能 SQL ステートメントの実行に関する状況情報と診断情報を保管します。GET DIAGNOSTICS または複合ステートメント 以外の SQL ステートメントが処理される場合、その SQL ステートメントを処理する前に、現行診断域がクリアされます。各 SQL ステートメントが処理されるときに、その SQL ステートメントの実行に関する情報は、1 つ以上の完了条件または例外条件として現行診断域に記録されます。

完了条件では、SQL ステートメントの正常完了、警告条件付きの完了、NOT FOUND 条件付きの完了のいずれかが示されます。例外条件は、ステートメントが失敗したことを示します。 GET DIAGNOSTICS ステートメントは、以前に実行された SQL ステートメントに関する条件およびその他の情報を診断域から戻すために、ほとんどの言語で使用できます。詳しくは、 1489 ページの『GET DIAGNOSTICS』を参照してください。さらに、条件情報は言語固有のメカニズムを通じて提供されます。

- SQL プロシージャー、SQL 関数、および SQL トリガーについては、 1779 ページの『SQL プロシージャー・ステートメント』を参照してください。
- ホスト言語については、『ホスト言語アプリケーションにおけるエラー条件と警告条件の検出と処理』を参照してください。

ホスト言語アプリケーションにおけるエラー条件と警告条件の検出と処理

各ホスト言語は、診断情報を処理するためのメカニズムを提供します。

- C、COBOL、および PL/I では、実行可能 SQL ステートメントを含むアプリケーション・プログラムは、少なくとも次のいずれか 1 つを用意しなければなりません。
  - 名前が SQLCA の構造体。
  - 名前が SQLSTATE の独立型の CHAR(5) (C の場合は CHAR(6)) 変数。
  - 名前が SQLCODE の独立型の整変数。

  独立型の SQLSTATE または SQLCODE は、ホスト構造体中で宣言されていてはなりません。独立型の SQLSTATE と SQLCODE の両者を用意しても構いません。

  INCLUDE SQLCA ステートメントの使用により、SQLCA を入手することができます。 SQLCA を用意する場合には、独立型の SQLSTATE または SQLCODE はいずれも用意することはできません。 SQLCA には、名前が SQLSTATE の文字ストリング変数、および名前が SQLCODE の整変数が含まれています。

  SQL 2003 コア標準に準拠するには、独立型の SQLSTATE を使用する必要があります。
- Java では、エラー状態の場合、SQLSTATE を取得するには getSQLState メソッドを使用し、 SQLCODE を取得するには getErrorCode メソッドを使用できます。
- REXX および RPG では、SQLCA が自動的に用意されます。

SQLSTATE

各 SQL ステートメント (GET DIAGNOSTICS または複合ステートメント以外) が実行された後、データベース・マネージャーは SQLSTATE を設定します。したがって、アプリケーション・プログラムでは、SQLCODE の代わりに SQLSTATE をテストすることによって、SQL ステートメントの実行の成否を検査することができます。

SQLSTATE はアプリケーション・プログラムに、共通エラー条件を示す共通コードを戻します。さらに、SQLSTATE は、アプリケーション・プログラムで特定のエラーまたはエラーのクラスをテストできるように設計されています。この体系は、すべてのデータベース・マネージャーで同一であり、ISO/ANSI SQL 2003 コア標準に基づいています。 SQLSTATE クラス、ならびにそれぞれの SQLCODE に関連付けられている SQLSTATE の全リストについては、「SQL メッセージおよびコード」トピック集を参照してください。

SQLCODE

各 SQL ステートメント (GET DIAGNOSTICS または複合ステートメント以外) が実行された後、データベース・マネージャーは SQLCODE を設定します。
SQLCODE は次のように設定されます。

- SQLCODE = 0 で、しかも SQLWARN0 がブランクの場合、実行は正しく行われました。
- SQLCODE = 100 の場合、データが見つかりませんでした。例えば、カーソルが結果表の最後の行より後にあることが原因で、FETCH ステートメントがデータを戻さない場合など。
- SQLCODE > 0 で、100 ではない場合、実行は警告付きで成功しました。
- SQLCODE = 0 で、しかも SQLWARN0 = 'W' の場合、警告が出されましたが、実行は正常に行われました。
- SQLCODE < 0 の場合、実行は失敗しました。

Db2 for i SQLCODE とそれに対応している SQLSTATE の全リストについては、「SQL メッセージおよびコード」トピック集を参照してください。

SQL のコメント

ほとんどのホスト言語の場合、静的 SQL ステートメントの中では、ホスト言語または SQL のコメントを使用することができます。 Java および REXX の場合、静的 SQL ステートメントの中では、ホスト言語または SQL のコメントを使用することはできません。

動的 SQL ステートメントの中に SQL コメントを組み込むことができます。

SQL コメントには、次の 2 つのタイプがあります。

単純コメント
: 単純コメントの前には、2 つのハイフンを連続で付けます。

ブラケット付きのコメント
: ブラケット付きコメントは、/* と */ で囲みます。

単純コメントは、次の規則に従って使用してください。

- 2 つのハイフンは同一行に置く必要があります。また、ハイフンとハイフンの間にスペースを入れることはできません。
- 単純コメントは、スペースを入れることができるところならば、どこからでも開始できます (ただし、区切りトークン内や 'EXEC' と 'SQL' との間は除きます)。
- 単純コメントは、次の行に続けることはできません。
- COBOL では、2 つのハイフンの前にスペースを 1 つ置く必要があります。

ブラケット付きコメントは、次の規則に従って使用してください。

- /* は、同一行に置く必要があります。また、間にスペースを入れることはできません。
- */ は、同一行に置く必要があります。また、間にスペースを入れることはできません。

- ブラケット付きコメントは、スペースを入れることができるところならば、どこからでも開始できます (ただし、区切りトークン内や 'EXEC' と 'SQL' との間は除きます)。
- ブラケット付きコメントは、次の行に続けることができます。
- ブラケット付きコメントは、他のブラケット付きコメント内にネストすることができます。

SQL ステートメントで (表の名前などの) 名前の前にコメントを組み込むと、ビュー、トリガー、変数、または MQT に保管されたテキスト内のオブジェクト名が、正しく維持されない原因となる場合があります。同様に、(SYSTRIGDEP などの) 従属関係ビューの行内の名前も正しく修飾されない可能性があります。

例 1

この例は、ステートメントの中に単純コメントを組み込む方法を示しています。

```
CREATE VIEW PRJ_MAXPER         -- PROJECTS WITH MOST SUPPORT PERSONNEL
   AS SELECT PROJNO, PROJNAME  -- NUMBER AND NAME OF PROJECT
      FROM PROJECT
      WHERE DEPTNO = 'E21'     -- SYSTEMS SUPPORT DEPT CODE
      AND PRSTAFF > 1
```

例 2

この例は、ステートメントの中にブラケット付きコメントを組み込む方法を示しています。

```
CREATE VIEW PRJ_MAXPER         /* PROJECTS WITH MOST SUPPORT
                                      PERSONNEL               */
   AS SELECT PROJNO, PROJNAME  /* NUMBER AND NAME OF PROJECT   */
      FROM PROJECT
      WHERE DEPTNO = 'E21'     /* SYSTEMS SUPPORT DEPT CODE    */
      AND PRSTAFF > 1
```

ALLOCATE CURSOR

ALLOCATE CURSOR ステートメントは、カーソルを定義し、それを結果セット・ロケーター変数に関連付けます。

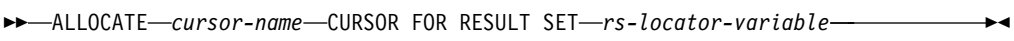
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができます。これは、動的に準備できる実行可能ステートメントです。これを対話式に発行することはできません。REXX で指定してはなりません。

権限

権限は不要です。

構文



説明

cursor-name
カーソルの名前を指定します。この名前は、ソース・プログラム内ですでに宣言されているカーソルを示すものであってはなりません。

CURSOR FOR RESULT SET *rs-locator-variable*
結果セット・ロケーター変数を宣言するための規則に従い、アプリケーション・プログラム内ですでに宣言された結果セット・ロケーター変数を指定します。

結果セット・ロケーター変数は ASSOCIATE LOCATORS ステートメントまたは DESCRIBE PROCEDURE SQL ステートメントで戻されたときに、有効な結果セット・ロケーター値を含んでいなければなりません。結果セット・ロケーター変数の値は、カーソルの割り当て時に使用されます。結果セット・ロケーターの値を後で変更しても、割り当てカーソルに影響はありません。結果セット・ロケーターの値は、そのソース・プログラム内で割り当てられる別のカーソルに使用されている値と同じであってはなりません。

規則

- 割り当てカーソルを使用する場合は、以下の規則が適用されます。
  - OPEN ステートメントを使用して割り当てカーソルをオープンすることはできません。
  - CLOSE ステートメントを使用して、割り当てカーソルをクローズすることができます。割り当てカーソルをクローズすると、ストアード・プロシージャー内に定義された関連するカーソルがクローズされます。
  - 各結果セットにつき、ただ 1 つのカーソルを割り当てることができます。
- CLOSQLCSR オプションを指定したプログラムでは、割り当てカーソルにも宣言カーソルの場合と同じ規則が当てはまります。CLOSQLCSR オプションは、CRTSQLxxx コマンドで指定するか、SET OPTION ステートメントを使用して指定できます。

- コミット操作を実行すると、プロシージャーで WITH HOLD が定義されていない割り当てカーソルのうち、*NONE 以外のコミット・レベルで実行されていない割り当てカーソルがクローズします。
- 割り当てカーソルをクローズすると、プロシージャー内の関連カーソルもクローズします。

例

カーソル C1 を定義し、結果セット・ロケーター変数 LOC1 と、SQL プロシージャーによって返される関連結果セットとにそのカーソルを関連付ける SQL プロシージャーの例を以下に示します。

```
ALLOCATE C1 CURSOR FOR RESULT SET LOC1;
```

ALLOCATE DESCRIPTOR

ALLOCATE DESCRIPTOR ステートメントは、SQL 記述子を割り振ります。

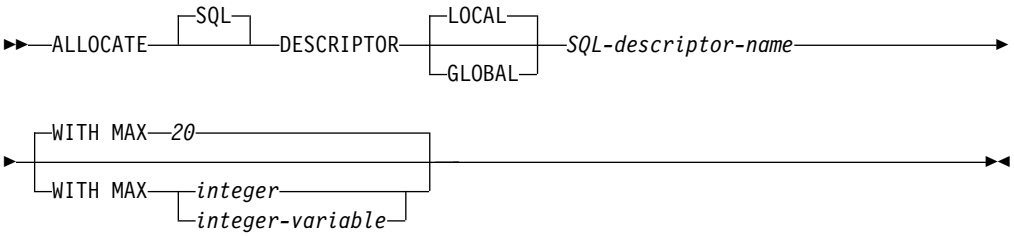
呼び出し

このステートメントは、アプリケーション・プログラム、SQL 関数、 SQL プロシージャー、またはトリガー内にのみ組み込むことができます。これを対話式に発行することはできません。これは実行可能ステートメントですが、動的に準備することはできません。REXX で指定してはなりません。

権限

権限は不要です。

構文



説明

LOCAL
記述子の名前の有効範囲はプログラム呼び出しのローカルであることを定義します。記述子がこの有効範囲外に知られることがなくなります。例えば、別個にコンパイルされた他のプログラムから呼び出されるプログラムでは、その呼び出し側プログラムによって割り振られた記述子を使用することはできません。また、記述子の有効範囲は、その記述子を含むプログラムが実行しているスレッドに限定されます。例えば、同じジョブの中にある別々の 2 つのスレッドで同じプログラムが実行している場合、2 番目のスレッドは、最初のスレッドによって割り振られた記述子を使用することができません。

GLOBAL
記述子の名前の有効範囲は SQL セッション全体であることを定義します。記述子は、同じデータベース接続を使用して実行するどのプログラムにも知られることになります。

SQL-descriptor-name
割り振る記述子の名前を指定します。名前は、指定した有効範囲を持つ既に存在する記述子と同一であってはなりません。

WITH MAX
記述子は指定された項目最大数をサポートするように割り振られます。この文節が指定されない場合、記述子は最大 20 項目を持つものとして割り振られます。

integer
割り振る項目の数を指定します。*integer* の値は、ゼロより大きく 8000 以下でなければなりません。

integer-variable
割り振る項目の数を含む整変数 (位取りがゼロの 10 進または数値変数) を指定します。この変数は、グローバル変数にすることはできません。*integer-variable* の値は、ゼロより大きく 8000 以下でなければなりません。

注

記述子持続性: ローカル記述子は、以下の CLOSQLCSR オプションに基づいて暗黙的に割り振り解除されます。

- ILE プログラムでは、CLOSQLCSR(*ENDACTGRP) が指定された場合 (デフォルト)、ローカル記述子は活動化グループが終了すると暗黙的に割り振り解除されます。 CLOSQLCSR(*ENDMOD) が指定された場合、ローカル記述子はモジュールの終了時に暗黙的に割り振り解除されます。
- OPM プログラムでは、CLOSQLCSR(*ENDPGM) が指定された場合 (デフォルト)、ローカル記述子はプログラムが終了すると暗黙的に割り振り解除されます。CLOSQLCSR(*ENDSQL) が指定された場合、ローカル記述子は呼び出しスタックの最初の SQL プログラムが終了すると暗黙的に割り振り解除されます。CLOSQLCSR(*ENDJOB) が指定された場合、ローカル記述子はジョブが終了すると暗黙的に割り振り解除されます。

グローバル記述子は、活動化グループが終了すると暗黙的に割り振り解除されます。

ローカル記述子とグローバル記述子の両方とも、DEALLOCATE DESCRIPTOR ステートメントを使用して明示的に割り振り解除できます。

例

20 項目を保持できる大きさの 'NEWDA' という記述子を割り振ります。

```
EXEC SQL ALLOCATE DESCRIPTOR 'NEWDA'
  WITH MAX 20
```

ALTER FUNCTION (外部スカラー)

ALTER FUNCTION (外部スカラー) ステートメントは、現行サーバーの外部スカラー関数を変更します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別される関数に対しては次のもの。
  - その関数に対する ALTER 特権、および
  - 関数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

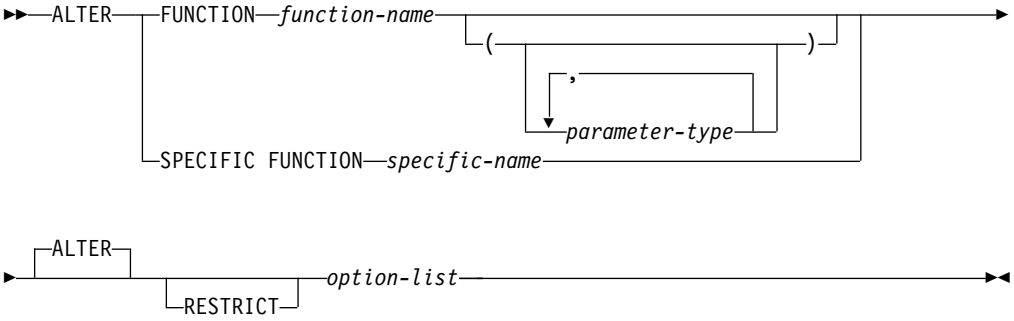
別の外部プログラムが指定されている場合、ステートメントの権限 ID によって保持される特権には、新規の外部スカラー関数を作成するために必要な同一特権が含まれていなければなりません。詳しくは、 1076 ページの『CREATE FUNCTION (外部スカラー)』を参照してください。

SECURED オプションが指定されるか、または関数が現在セキュアである場合は、以下のとおりです。

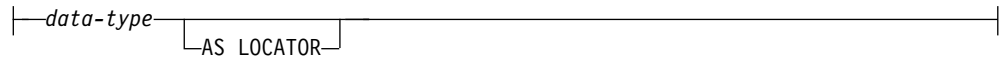
- このステートメントの許可 ID には、セキュリティー管理者権限 がなければなりません。 21 ページの『管理権限』を参照してください。

SQL 特権に対応するシステム権限の説明については、『関数またはプロシージャーへの権限を検査する際の対応するシステム権限』、『表またはビューへの権限を検査する際の対応するシステム権限』、および『特殊タイプへの権限を検査する際の対応するシステム権限』を参照してください。

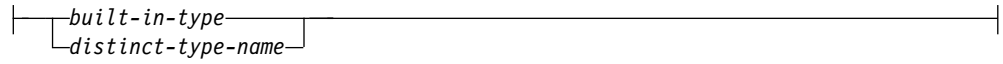
構文



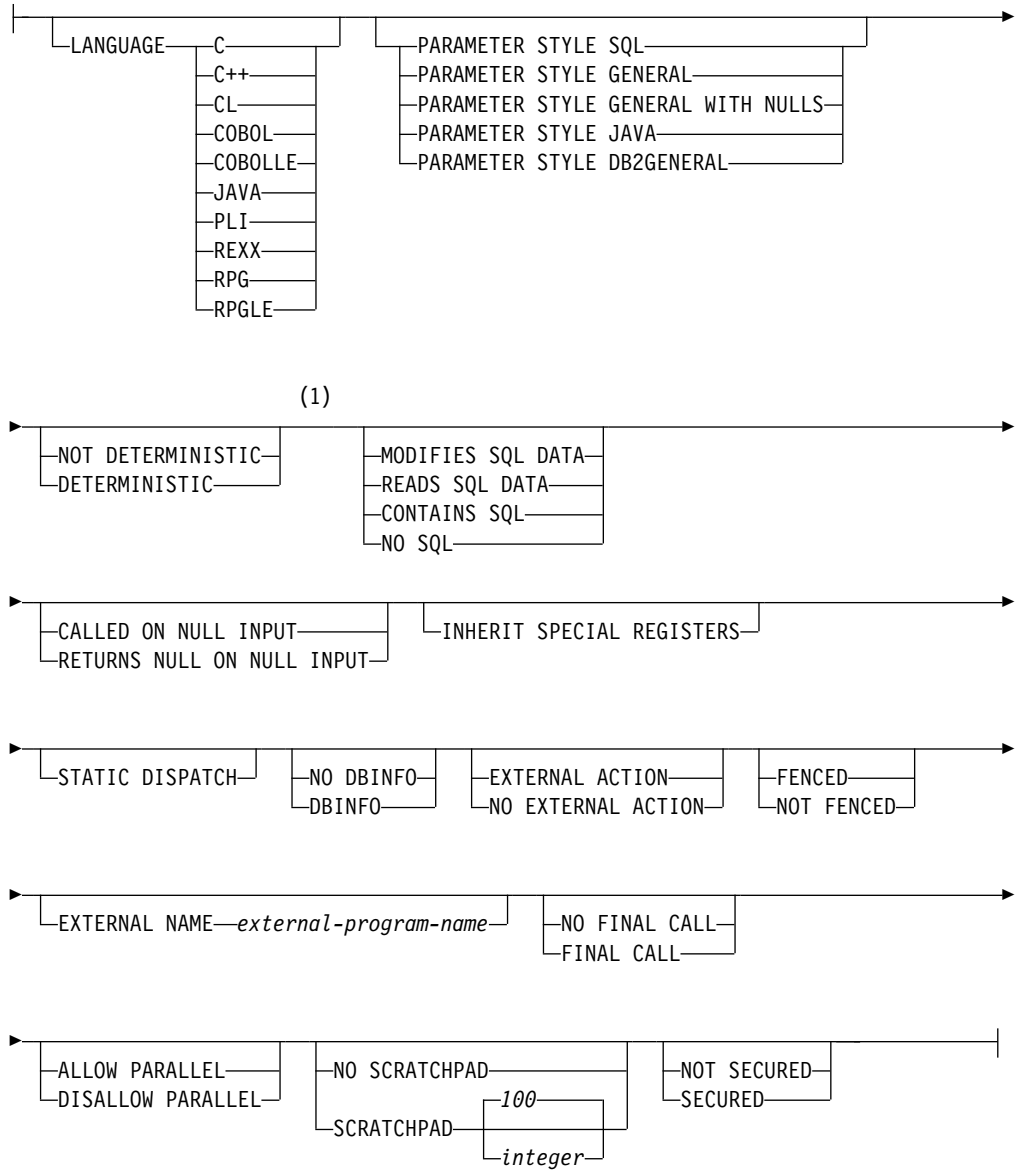
parameter-type:



data-type:



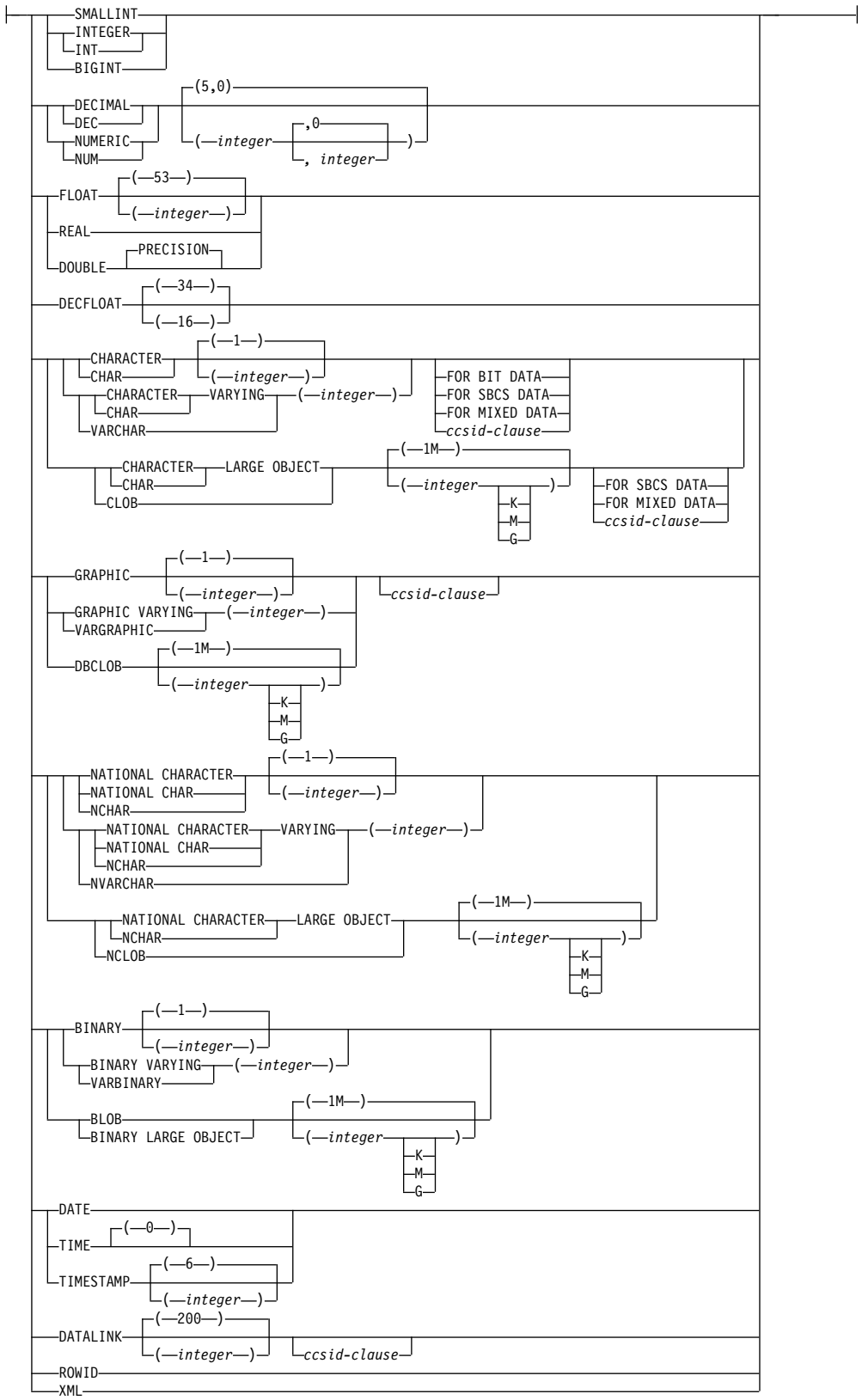
option-list:



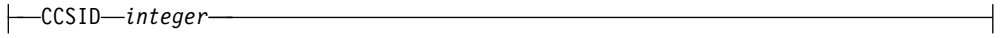
注:

1 *option-list* 内の文節は、どのような順番で指定してもかまいません。

built-in-type:



ccsid-clause:



説明

FUNCTION または SPECIFIC FUNCTION

変更する関数を識別します。 *function-name* は、現行サーバーに存在している外部スカラー関数を識別していなければなりません。これは、組み込み関数、ソース派生関数、または SQL 関数を識別することはできません。外部表関数は、外部スカラー関数に変更することはできません。

指定した関数が変更されます。関数の所有者は保持されます。関数が変更された時点で外部プログラムまたはサービス・プログラムが存在する場合、関数に対するすべての特権が保持されます。

FUNCTION *function-name*

関数を名前によって識別します。 *function-name* は厳密に 1 つの関数を示す必要があります。この関数には、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前の関数が複数ある場合、エラーが戻されます。

FUNCTION *function-name (parameter-type,...)*

関数を一意的に識別する関数シグニチャーによって、関数を識別します。 *function-name (parameter-type,...)* は、指定された関数シグニチャーを持つ関数を識別しなければなりません。指定されたパラメーターは、関数の作成時に指定された、対応する位置にあるデータ・タイプと一致していなければなりません。変更される特定の関数インスタンスを識別する場合、データ・タイプの数とデータ・タイプの論理連結が使用されます。データ・タイプの同義語は、一致として扱われます。デフォルトがあるパラメーターは、このシグニチャーに含まれていなければなりません。

function-name() を指定する場合、識別される関数にパラメーターを使用することはできません。

function-name

関数の名前を識別します。

(parameter-type,...)

関数のパラメーターを識別します。

非修飾の特殊タイプ名を指定する場合、データベース・マネージャーはその特殊タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、以下のいずれかを使用します。

- 中が空の括弧は、データ・タイプが一致しているか否かの判別時にデータベース・マネージャーによって属性が無視されることを示します。例えば、DEC() は、DEC(7,2) のデータ・タイプで定義された関数のパラメーターに一致するものとみなされます。ただし、精度値は特定のデータ・タイプ (REAL または DOUBLE) を示すため、中が空の括弧で FLOAT を指定することはできません。

- 長さ属性、精度属性、あるいは位取り属性に特定の値を指定した場合、その値は、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプが FLOAT の場合、突き合わせはデータ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度は指定された値に厳密に一致している必要はありません。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。暗黙の長さは、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

FOR DATA 文節または CCSID 文節の指定はオプショナルです。いずれの文節も指定しないと、データ・タイプが一致するかどうかを判定する場合に、データベース・マネージャーが属性を無視することを示します。どちらか一方の文節を指定する場合は、CREATE FUNCTION ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

AS LOCATOR
関数が、このパラメーターのロケーターを受け取るように定義されることを示します。 AS LOCATOR を指定する場合は、データ・タイプは LOB または XML、あるいは LOB または XML に基づく特殊タイプでなければなりません。

SPECIFIC FUNCTION *specific-name*
関数を特定名によって識別します。 *specific-name* は、現行サーバーに存在する特定の関数を示している必要があります。

ALTER *option-list*
関数の 1 つ以上のオプションが変更されることを示します。オプションが指定されない場合は、既存の関数定義での値が使用されます。各オプションの説明については、 1076 ページの『CREATE FUNCTION (外部スカラー)』を参照してください。

RESTRICT
任意のビュー、関数、プロシージャー、またはマテリアライズ照会表によって参照された場合は、関数が変更されないことを示します。

注

関数の定義または置換に関する一般考慮事項**:** 関数の定義に関する一般情報については、CREATE FUNCTION (外部スカラー) を参照してください。 ALTER FUNCTION (外部スカラー) を使用すると、関数の特権を保持したままで、個々の属性を変更できます。

NOT SECURED から **SECURED** への関数の変更**:** ALTER FUNCTION ステートメントが実行された後、関数はセキュアであると見なされます。Db2 は SECURED 属性を、ユーザー定義関数に対するすべての変更の監査手順をユーザーが確立したことを宣言するアサーションとして扱います。Db2 は、後続のすべての ALTER FUNCTION ステートメントがこの監査手順によってレビューされると想定します。

セキュアな関数内での他のユーザー定義関数の呼び出し: 行アクセス制御または列アクセス制御を使用している表を参照する SQL データ変更ステートメント内でセキュアなユーザー定義関数が参照されていて、そのセキュアなユーザー定義関数が他のユーザー定義関数を呼び出す場合、ネストされたユーザー定義関数はセキュアであるとは判定されません。こういったネストされた関数が機密データにアクセスする可能性がある場合、IBM i のデータベース・セキュリティー管理者機能の権限があるユーザーは、それらの関数がそのデータにアクセスすることを許可されていること、および、それらの関数に加えられるすべての変更に関して変更管理監査手順が確立されていることを確認する必要があります。

代替構文: 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード VARIANT と NOT VARIANT は、NOT DETERMINISTIC と DETERMINISTIC の同義語として使用することができます。
- キーワード NULL CALL と NOT NULL CALL は、CALLED ON NULL INPUT と RETURNS NULL ON NULL INPUT の同義語として使用できます。
- キーワード SIMPLE CALL は、GENERAL の同義語として使用できます。
- DB2GENERAL の同義語として、キーワード DB2GENRL を使用できます。
- SQL の同義語として、値 DB2SQL を使用できます。
- PARAMETER STYLE 文節のキーワード PARAMETER STYLE はオプションです。
- DETERMINISTIC の同義語として、キーワード IS DETERMINISTIC を使用できます。

例

MYFUNC 関数の定義を変更して、その関数の呼び出し時に起動する外部プログラムの名前を変更します。外部プログラムの名前は、PROG10B です。

```
ALTER FUNCTION MYFUNC
  EXTERNAL NAME PROG10B
```

ALTER FUNCTION (外部表)

ALTER FUNCTION (外部表) ステートメントは、現行サーバーの外部表関数を変更します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別される関数に対しては次のもの。
  - その関数に対する ALTER 特権、および
  - 関数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

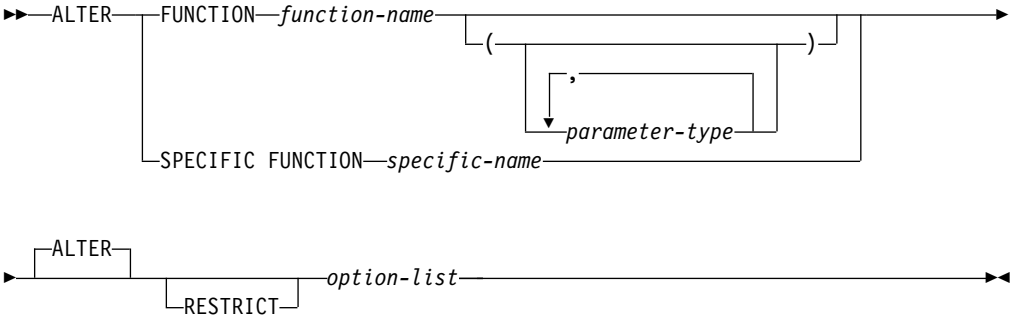
別の外部プログラムが指定されている場合、ステートメントの権限 ID によって保持される特権には、新規の外部表関数を作成するために必要な同一特権が含まれていなければなりません。詳しくは、 1100 ページの『CREATE FUNCTION (外部表)』を参照してください。

SECURED オプションが指定されるか、または関数が現在セキュアである場合は、以下のとおりです。

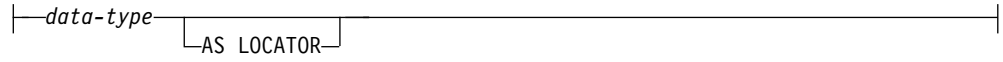
- このステートメントの許可 ID には、セキュリティー管理者権限 がなければなりません。 21 ページの『管理権限』を参照してください。

SQL 特権に対応するシステム権限の説明については、『関数またはプロシージャーへの権限を検査する際の対応するシステム権限』、『表またはビューへの権限を検査する際の対応するシステム権限』、および『特殊タイプへの権限を検査する際の対応するシステム権限』を参照してください。

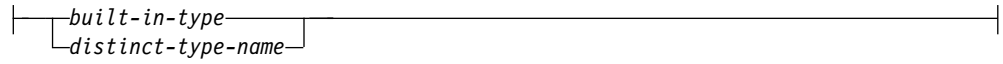
構文



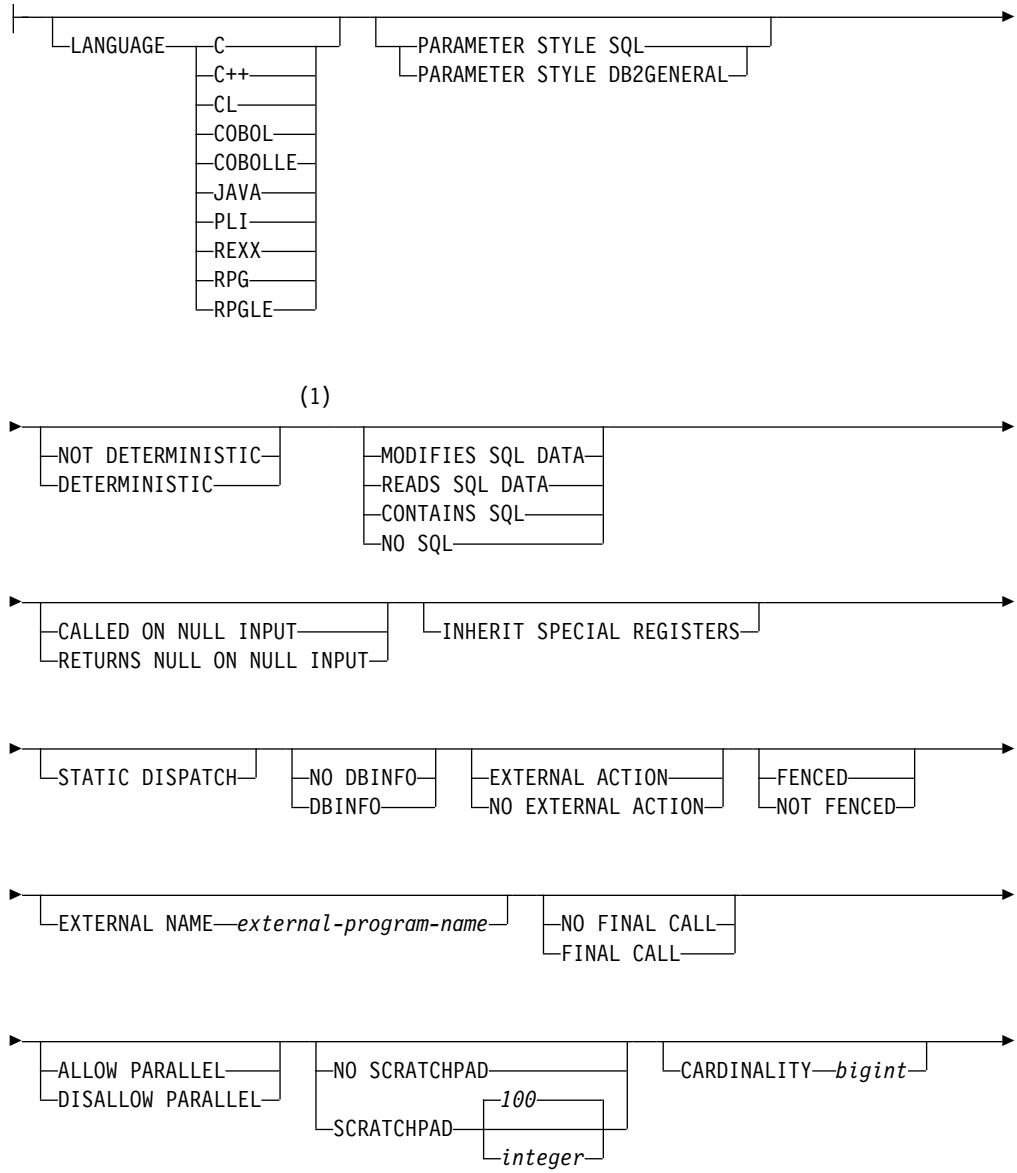
parameter-type:

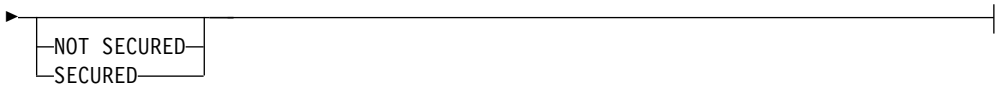


data-type:



option-list:

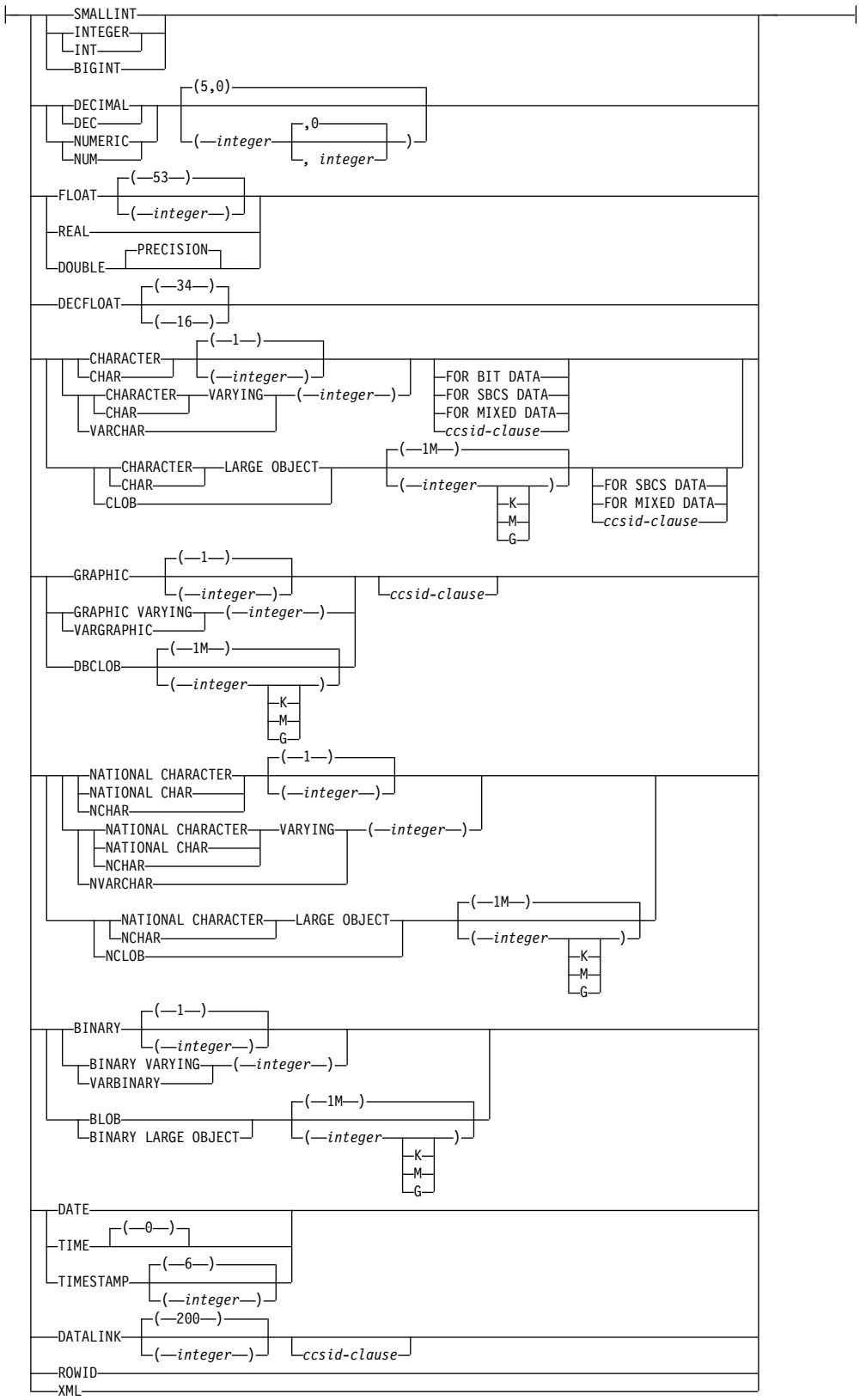




注:

1 *option-list* 内の文節は、どのような順番で指定してもかまいません。

built-in-type:



ccsid-clause:

```
├──CCSID──integer──────────────────────────────────────────────────────┤
```

説明

FUNCTION または SPECIFIC FUNCTION

変更する関数を識別します。 *function-name* は、現行サーバーに存在している外部表関数を識別していなければなりません。これは、組み込み関数、ソース派生関数、または SQL 関数を識別することはできません。外部スカラー関数は、外部表関数に変更することはできません。

指定した関数が変更されます。関数の所有者は保持されます。関数が変更された時点で外部プログラムまたはサービス・プログラムが存在する場合、関数に対するすべての特権が保持されます。

FUNCTION *function-name*

関数を名前によって識別します。 *function-name* は厳密に 1 つの関数を示す必要があります。この関数には、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前の関数が複数ある場合、エラーが戻されます。

FUNCTION *function-name (parameter-type,...)*

関数を一意的に識別する関数シグニチャーによって、関数を識別します。 *function-name (parameter-type,...)* は、指定された関数シグニチャーを持つ関数を識別しなければなりません。指定されたパラメーターは、関数の作成時に指定された、対応する位置にあるデータ・タイプと一致していなければなりません。変更される特定の関数インスタンスを識別する場合、データ・タイプの数とデータ・タイプの論理連結が使用されます。データ・タイプの同義語は、一致として扱われます。デフォルトがあるパラメーターは、このシグニチャーに含まれていなければなりません。

function-name() を指定する場合、識別される関数にパラメーターを使用することはできません。

function-name

関数の名前を識別します。

(parameter-type,...)

関数のパラメーターを識別します。

非修飾の特殊タイプ名を指定する場合、データベース・マネージャーはその特殊タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、以下のいずれかを使用します。

- 中が空の括弧は、データ・タイプが一致しているか否かの判別時にデータベース・マネージャーによって属性が無視されることを示します。例えば、DEC() は、DEC(7,2) のデータ・タイプで定義された関数のパラメーターに一致するものとみなされます。ただし、精度値は特定のデータ・タイプ (REAL または DOUBLE) を示すため、中が空の括弧で FLOAT を指定することはできません。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を指定した場合、その値は、CREATE FUNCTION ステートメントの中で暗黙的

または明示的に指定された値と正確に一致している必要があります。データ・タイプが FLOAT の場合、突き合わせはデータ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度は指定された値に厳密に一致している必要はありません。

- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。暗黙の長さは、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

FOR DATA 文節または CCSID 文節の指定はオプショナルです。いずれの文節も指定しないと、データ・タイプが一致するかどうかを判定する場合に、データベース・マネージャーが属性を無視することを示します。どちらか一方の文節を指定する場合は、CREATE FUNCTION ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

AS LOCATOR
関数が、このパラメーターのロケーターを受け取るように定義されることを示します。 AS LOCATOR を指定する場合は、データ・タイプは LOB または XML、あるいは LOB または XML に基づく特殊タイプでなければなりません。

SPECIFIC FUNCTION *specific-name*
関数を特定名によって識別します。 *specific-name* は、現行サーバーに存在する特定の関数を示している必要があります。

ALTER *option-list*
関数の 1 つ以上のオプションが変更されることを示します。オプションが指定されない場合は、既存の関数定義での値が使用されます。各オプションの説明については、 1100 ページの『CREATE FUNCTION (外部表)』を参照してください。

RESTRICT
任意のビュー、関数、プロシージャー、またはマテリアライズ照会表によって参照された場合は、関数が変更されないことを示します。

注

関数の定義または置換に関する一般考慮事項: 関数の定義に関する一般情報については、CREATE FUNCTION (外部表) を参照してください。 ALTER FUNCTION (外部表) を使用すると、関数の特権を保持したままで、個々の属性を変更できます。

NOT SECURED から **SECURED** への関数の変更: ALTER FUNCTION ステートメントが実行された後、関数はセキュアであると見なされます。Db2 は SECURED 属性を、ユーザー定義関数に対するすべての変更の監査手順をユーザーが確立したことを宣言するアサーションとして扱います。Db2 は、後続のすべての ALTER FUNCTION ステートメントがこの監査手順によってレビューされると想定します。

セキュアな関数内での他のユーザー定義関数の呼び出し: 行アクセス制御または列アクセス制御を使用している表を参照する SQL データ変更ステートメント内でセキュアなユーザー定義関数が参照されていて、そのセキュアなユーザー定義関数が他のユーザー定義関数を呼び出す場合、ネストされたユーザー定義関数はセキュアであるとは判定されません。こういったネストされた関数が機密データにアクセスする可能性がある場合、IBM i のデータベース・セキュリティー管理者機能の権限があるユーザーは、それらの関数がそのデータにアクセスすることを許可されていること、および、それらの関数に加えられるすべての変更に関して変更管理監査手順が確立されていることを確認する必要があります。

代替構文: 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード VARIANT と NOT VARIANT は、NOT DETERMINISTIC と DETERMINISTIC の同義語として使用することができます。
- キーワード NULL CALL と NOT NULL CALL は、CALLED ON NULL INPUT と RETURNS NULL ON NULL INPUT の同義語として使用できます。
- DB2GENERAL の同義語として、キーワード DB2GENRL を使用できます。
- SQL の同義語として、値 DB2SQL を使用できます。
- PARAMETER STYLE 文節のキーワード PARAMETER STYLE はオプションです。
- DETERMINISTIC の同義語として、キーワード IS DETERMINISTIC を使用できます。

例

カーディナリティーを 10,000 に設定するには、外部表関数の定義を変更してください。

```
ALTER FUNCTION GET_TABLE
  ALTER CARDINALITY 10000
```

ALTER FUNCTION (SQL スカラー)

ALTER FUNCTION (SQL スカラー) ステートメントは、現行サーバーの SQL スカラー関数を変更します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別される関数に対しては次のもの。
  - その関数に対する ALTER 特権、および
  - 関数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

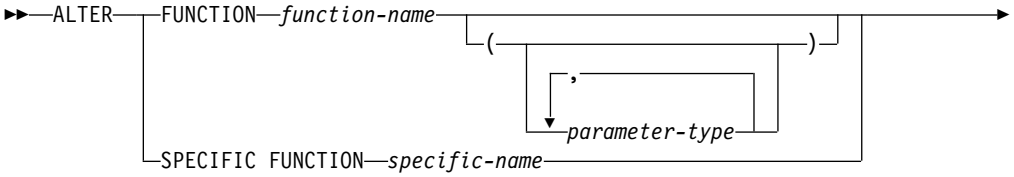
別の外部プログラムが指定されている場合、ステートメントの権限 ID によって保持される特権には、新規の外部スカラー関数を作成するために必要な同一特権が含まれていなければなりません。詳しくは、 1135 ページの『CREATE FUNCTION (SQL スカラー)』を参照してください。

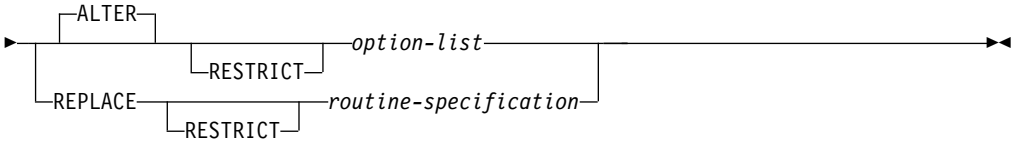
SECURED オプションが指定されるか、または関数が現在セキュアである場合は、以下のとおりです。

- このステートメントの許可 ID には、セキュリティー管理者権限 がなければなりません。 21 ページの『管理権限』を参照してください。

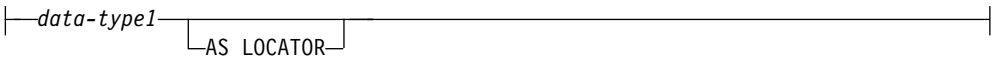
SQL 特権に対応するシステム権限の説明については、『関数またはプロシージャーへの権限を検査する際の対応するシステム権限』、『表またはビューへの権限を検査する際の対応するシステム権限』、および『特殊タイプへの権限を検査する際の対応するシステム権限』を参照してください。

構文

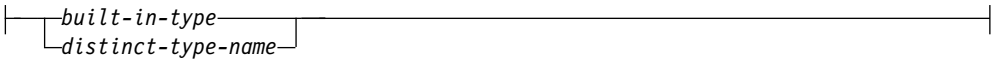




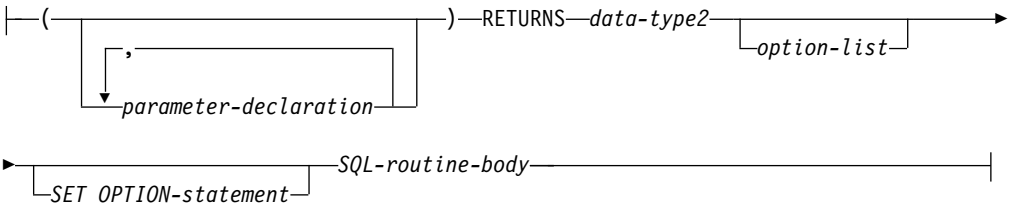
parameter-type:



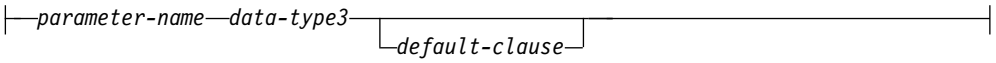
data-type1, data-type2,data-type3:



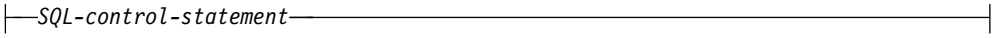
routine-specification:



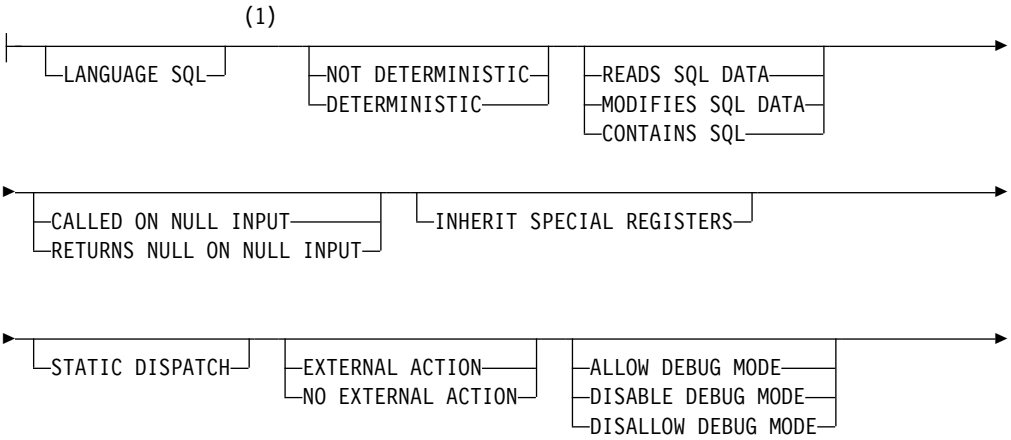
parameter-declaration:

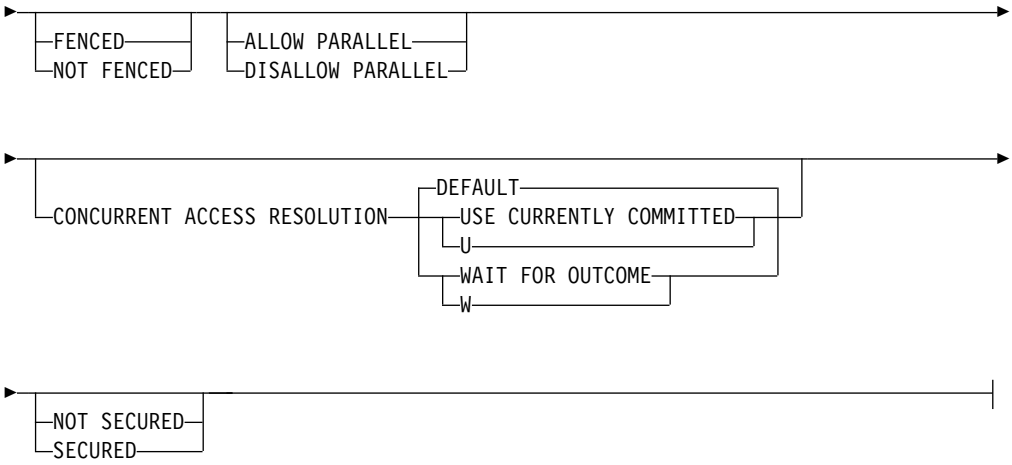


SQL-routine-body:



option-list:

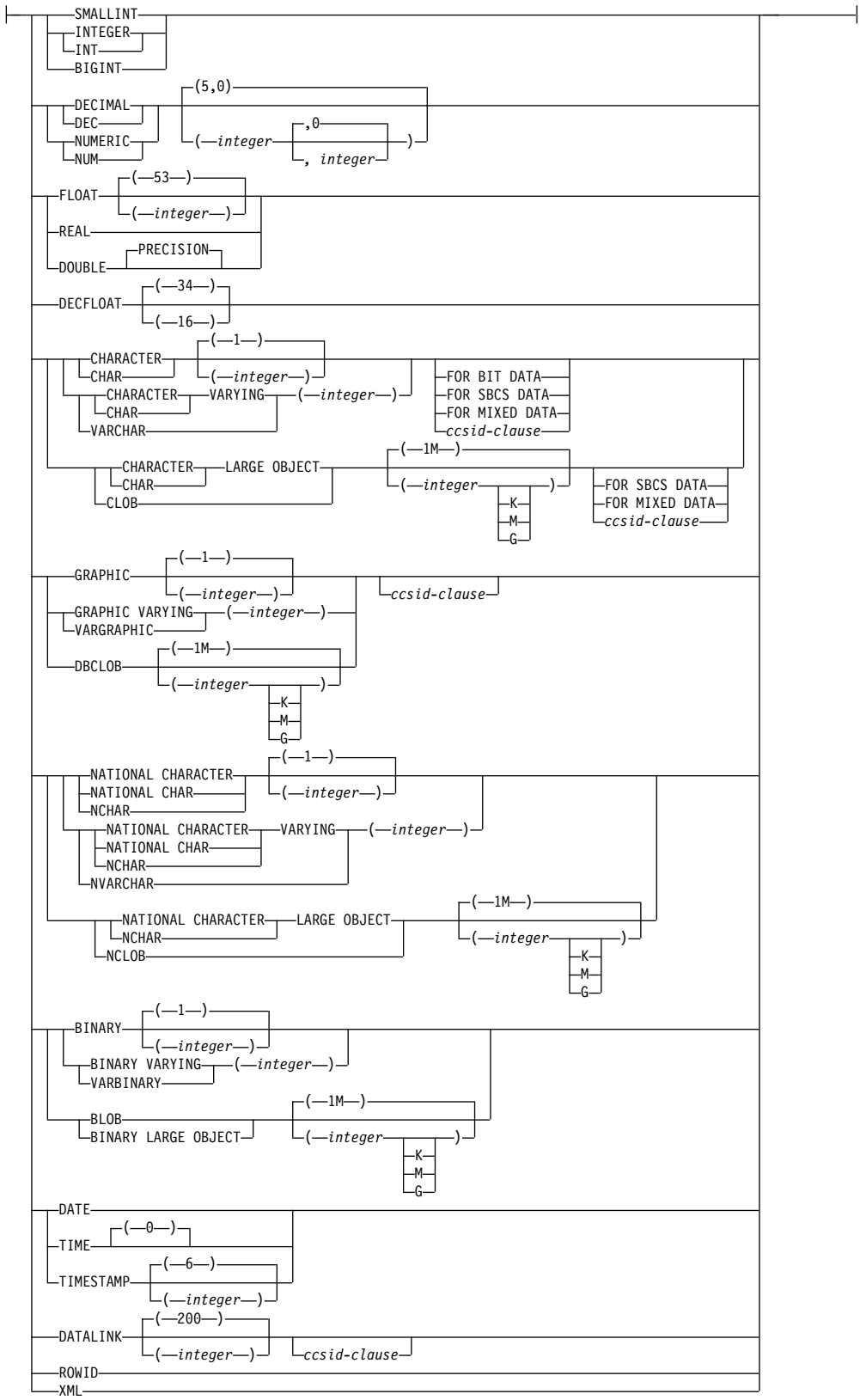




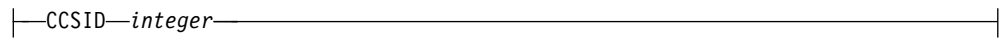
注:

1 *option-list* 内の文節は、どのような順番で指定してもかまいません。

built-in-type:



ccsid-clause:



default-clause:

説明

FUNCTION または **SPECIFIC FUNCTION**

変更する関数を識別します。この関数名 は、現行サーバーに存在している SQL スカラー関数を識別していなければなりません。

指定した関数が変更されます。関数の所有者と関数に関するすべての特権は、保持されます。

FUNCTION *function-name*

関数を名前によって識別します。 *function-name* は厳密に 1 つの関数を示す必要があります。この関数には、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前の関数が複数ある場合、エラーが戻されます。

FUNCTION *function-name (parameter-type,...)*

関数を一意的に識別する関数シグニチャーによって、関数を識別します。 *function-name (parameter-type,...)* は、指定された関数シグニチャーを持つ関数を識別しなければなりません。指定されたパラメーターは、関数の作成時に指定された、対応する位置にあるデータ・タイプと一致していなければなりません。変更される特定の関数インスタンスを識別する場合、データ・タイプの数とデータ・タイプの論理連結が使用されます。データ・タイプの同義語は、一致として扱われます。デフォルトがあるパラメーターは、このシグニチャーに含まれていなければなりません。

function-name() を指定する場合、識別される関数にパラメーターを使用することはできません。

function-name

関数の名前を識別します。

(parameter-type,...)

関数のパラメーターを識別します。

非修飾の特殊タイプ名を指定する場合、データベース・マネージャーはその特殊タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、以下のいずれかを使用します。

- 中が空の括弧は、データ・タイプが一致しているか否かの判別時にデータベース・マネージャーによって属性が無視されることを示します。例えば、DEC() は、DEC(7,2) のデータ・タイプで定義された関数のパラメーターに一致するものとみなされます。ただし、精度値は

特定のデータ・タイプ (REAL または DOUBLE) を示すため、中が空の括弧で FLOAT を指定することはできません。

- 長さ属性、精度属性、あるいは位取り属性に特定の値を指定した場合、その値は、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプが FLOAT の場合、突き合わせはデータ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度は指定された値に厳密に一致している必要はありません。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。暗黙の長さは、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

FOR DATA 文節または CCSID 文節の指定はオプショナルです。いずれの文節も指定しないと、データ・タイプが一致するかどうかを判定する場合に、データベース・マネージャーが属性を無視することを示します。どちらか一方の文節を指定する場合は、CREATE FUNCTION ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

AS LOCATOR
関数が、このパラメーターのロケーターを受け取るように定義されることを示します。 AS LOCATOR を指定する場合は、データ・タイプは LOB または XML、あるいは LOB または XML に基づく特殊タイプでなければなりません。

SPECIFIC FUNCTION *specific-name*
関数を特定名によって識別します。 *specific-name* は、現行サーバーに存在する特定の関数を示している必要があります。

ALTER *option-list*
関数の 1 つ以上のオプションが変更されることを示します。 ALTER FUNCTION ALTER *option-list* が指定されてオプションが指定されない場合は、既存の関数定義での値が使用されます。各オプションの説明については、1135 ページの『CREATE FUNCTION (SQL スカラー)』を参照してください。

REPLACE *routine-specification*
オプションおよびパラメーターを含む既存の関数定義を、このステートメントで指定したものに置き換えることを示します。関数を置き換えると、すべてのオプションの値が置き換えられます。オプションを指定しない場合は、新規 SQL スカラー関数が作成されるときと同じデフォルトが使用されます。詳しくは、1135 ページの『CREATE FUNCTION (SQL スカラー)』を参照してください。

ルーチンにコメントやラベルがある場合は、ルーチン定義からコメントやラベルが削除されます。

RESTRICT
任意の関数、マテリアライズ照会表、プロシージャー、トリガー、またはビューによって参照された場合は、関数が変更または置換されないことを示します。

(*parameter-declaration***,...)**
関数のパラメーターの数、および各パラメーターのデータ・タイプと名前を指定します。

SQL 関数に許可されるパラメーターの最大数は 2000 です。

parameter-name
パラメーター名を指定します。この名前は、関数の本体に含まれるパラメーターを参照するのに使用されます。この名前は、パラメーター・リスト内の他のパラメーター名 と同じものであってはなりません。

data-type3
入力パラメーターのデータ・タイプを指定します。CCSID が指定されている場合、関数に渡される前に、パラメーターはその CCSID に変換されます。CCSID が指定されていない場合は、CCSID は、関数の呼び出し時点における現行サーバーのデフォルトの CCSID によって決まります。

default-clause
パラメーターのデフォルト値を指定します。デフォルト値は、定数、特殊レジスター、グローバル変数、式、またはキーワード NULL にすることができます。式は、集約関数および列名を含まない、 196 ページの『式』で定義されている任意の式です。デフォルト値が指定されていない場合、パラメーターにデフォルト値がないため、呼び出し時に省略できません。式ストリングの最大長は 64K です。

デフォルトの式は、パラメーターのデータ・タイプに対して割り当ての互換性がなければなりません。

デフォルト式内でリスト中の数値定数を区切る区切り記号として使用するコンマの後には、スペースが 1 つ必要です。

配列タイプのパラメーターにデフォルトを指定することはできません。

RETURNS
関数の出力を指定します。

data-type2
出力のデータ・タイプと属性を指定します。

あらゆる組み込みデータ・タイプ (ただし、LONG VARCHAR または LONG VARGRAPHIC は除く) や特殊タイプを指定することができます。

CCSID が指定され、戻りデータの CCSID が異なる CCSID でコード化されている場合、データは指定された CCSID に変換されます。

CCSID が指定されていない場合、戻りデータの CCSID が異なる CCSID でコード化されている場合には、戻りデータはジョブの CCSID (グラフィック・ストリング戻り値の場合は、ジョブに関連したグラフィック CCSID) に変換されます。 変換時に文字が失われるのを防ぐために、関数から戻される文字をすべて表現できる CCSID を明示的に指定することを考慮してください。 これは、データ・タイプがグラフィック・ストリング・データの場合に特に重要です。 この場合、CCSID 1200 または 13488 (ユニコード・グラフィック・ストリング・データ) を使用することを考慮してください。

option-list
変更される関数のオプションのリスト。これらのオプションは、前記の ALTER *option-list* に記載したものと同じです。具体的なオプションを指定しない場合は、新規関数が作成されるときと同じデフォルトが使用されます。詳しくは、1135 ページの『CREATE FUNCTION (SQL スカラー)』を参照してください。

SET OPTION-statement
関数を作成するときに使用するオプションを指定します。例えば、デバッグ可能な関数を作成するときは、次のステートメントを含めることができます。

```
SET OPTION DBGVIEW = *SOURCE
```

詳しくは、 1696 ページの『SET OPTION』を参照してください。

オプション CNULRQD、COMPILEOPT、NAMING、および SQLCA は、ALTER FUNCTION ステートメントでは使用できません。デフォルト値式を処理するときには、オプション ALWCPYDTA、
CONACC、DATFMT、DATSEP、DECFLTRND、DECMPT、DECRESULT、DFTRDBCOL、LANGID、
SQLCURRULE、SQLPATH、SRTSEQ、TGTRLS、TIMFMT、および TIMSEP が使用されます。

SQL-routine-body
複合ステートメントも含め、単一の SQL ステートメントを指定します。 SQL 関数の定義についての詳細は、 1771 ページの『第 8 章 SQL 制御ステートメント』を参照してください。

CONNECT、SET CONNECTION、RELEASE、DISCONNECT、COMMIT、ROLLBACK および SET TRANSACTION ステートメントを実行するプロシージャーへの呼び出しは、関数内では使用できません。

SQL-routine-body は、RETURN ステートメントが少なくとも 1 つは含まれていなければならず、関数の呼び出し時に RETURN ステートメントが 1 つ実行される必要があります。

REPLACE キーワードを指定する ALTER PROCEDURE (SQL)、ALTER FUNCTION (SQL スカラー)、および ALTER FUNCTION (SQL 表) は、*SQL-routine-body* では使用できません。

注

関数の定義または置換に関する一般考慮事項**:** 関数の定義に関する一般情報については、CREATE FUNCTION (SQL スカラー) を参照してください。 ALTER FUNCTION (SQL スカラー) を使用すると、関数の特権を保持したままで、個々の属性を変更できます。

カスケード効果**:** REPLACE が RESTRICT なしで指定され、関数シグニチャーまたは結果データ・タイプが変更される場合、この関数を参照する関数、マテリアライズ照会表、プロシージャー、トリガー、およびビューからの結果は、予測不能なものになる可能性があります。参照されるオブジェクトはすべて再作成する必要があります。

難読化されたステートメント: 難読化されたステートメントを使用して作成した関数を変更できます。ステートメントを変更すると、カタログに保存されているエンコードされたバージョンのステートメントが変更されるため、 SQL ステートメントの最大長を超えることがあります。これが起こると、エラーが発行され、変更は失敗します。

NOT SECURED から **SECURED** への関数の変更: ALTER FUNCTION ステートメントが実行された後、関数はセキュアであると見なされます。Db2 は SECURED 属性を、ユーザー定義関数に対するすべての変更の監査手順をユーザーが確立したことを宣言するアサーションとして扱います。Db2 は、後続のすべての ALTER FUNCTION ステートメントがこの監査手順によってレビューされると想定します。

セキュアな関数内での他のユーザー定義関数の呼び出し: 行アクセス制御または列アクセス制御を使用している表を参照する SQL データ変更ステートメント内でセキュアなユーザー定義関数が参照されていて、そのセキュアなユーザー定義関数が他のユーザー定義関数を呼び出す場合、ネストされたユーザー定義関数はセキュアであるとは判定されません。こういったネストされた関数が機密データにアクセスする可能性がある場合、IBM i のデータベース・セキュリティー管理者機能の権限があるユーザーは、それらの関数がそのデータにアクセスすることを許可されていること、および、それらの関数に加えられるすべての変更に関して変更管理監査手順が確立されていることを確認する必要があります。

代替構文: 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード VARIANT と NOT VARIANT は、NOT DETERMINISTIC と DETERMINISTIC の同義語として使用することができます。
- キーワード NULL CALL と NOT NULL CALL は、CALLED ON NULL INPUT と RETURNS NULL ON NULL INPUT の同義語として使用できます。
- DETERMINISTIC の同義語として、キーワード IS DETERMINISTIC を使用できます。

例

SQL スカラー関数の定義を変更し、関数が deterministic 関数であることを指示します。

```
ALTER FUNCTION MY_UDF1
  DETERMINISTIC
```

ALTER FUNCTION (SQL 表)

ALTER FUNCTION (SQL 表) ステートメントは、現行サーバーの SQL 表関数を変更します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別される関数に対しては次のもの。
  - その関数に対する ALTER 特権、および
  - 関数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

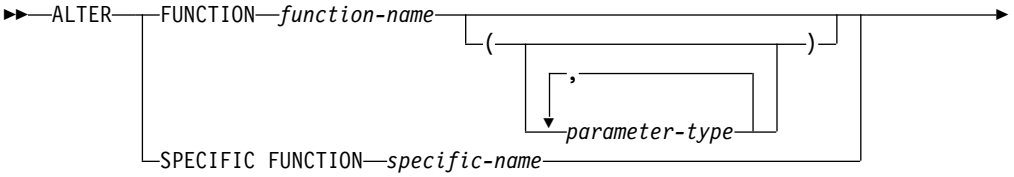
別の外部プログラムが指定されている場合、ステートメントの権限 ID によって保持される特権には、新規の外部表関数を作成するために必要な同一特権が含まれていなければなりません。詳しくは、 1151 ページの『CREATE FUNCTION (SQL 表)』を参照してください。

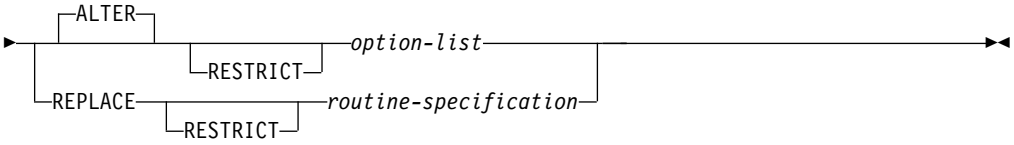
SECURED オプションが指定されるか、または関数が現在セキュアである場合は、以下のとおりです。

- このステートメントの許可 ID には、セキュリティー管理者権限 がなければなりません。 21 ページの『管理権限』を参照してください。

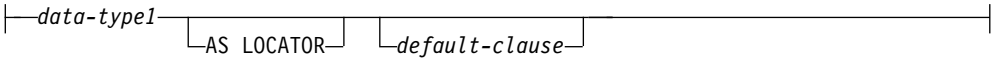
SQL 特権に対応するシステム権限の説明については、『関数またはプロシージャーへの権限を検査する際の対応するシステム権限』、『表またはビューへの権限を検査する際の対応するシステム権限』、および『特殊タイプへの権限を検査する際の対応するシステム権限』を参照してください。

構文

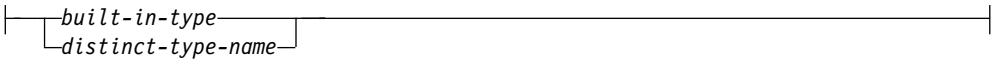




parameter-type:

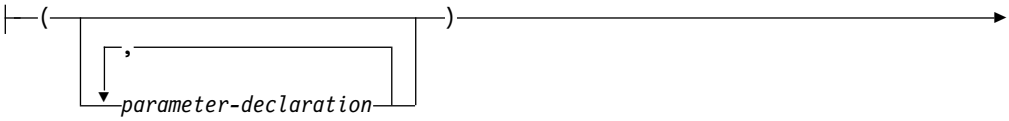


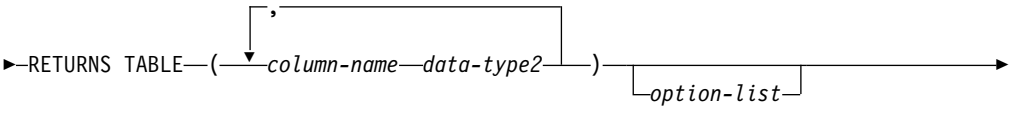
data-type1, data-type2, data-type3:

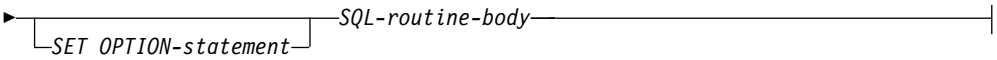


default-clause:

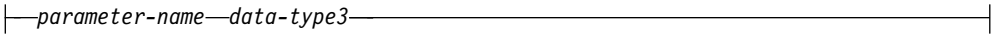
routine-specification:



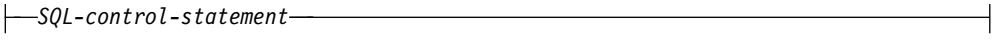




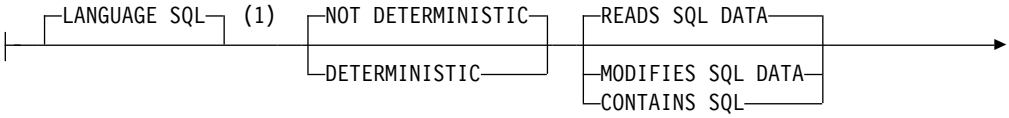
parameter-declaration:

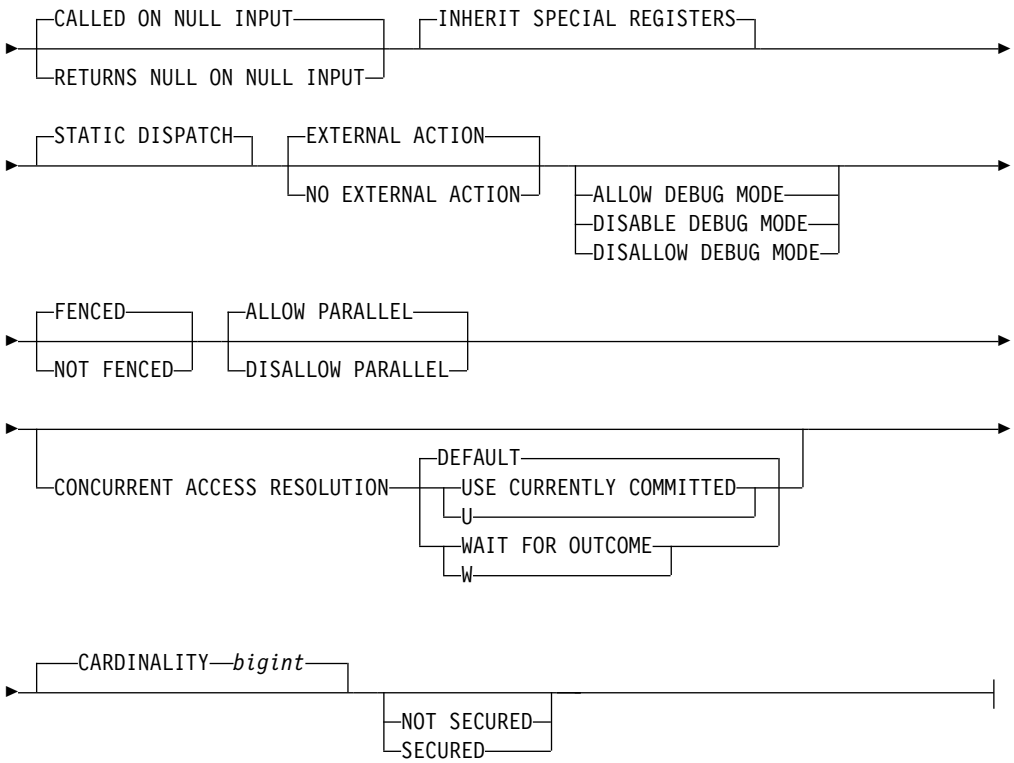


SQL-routine-body:



option-list:

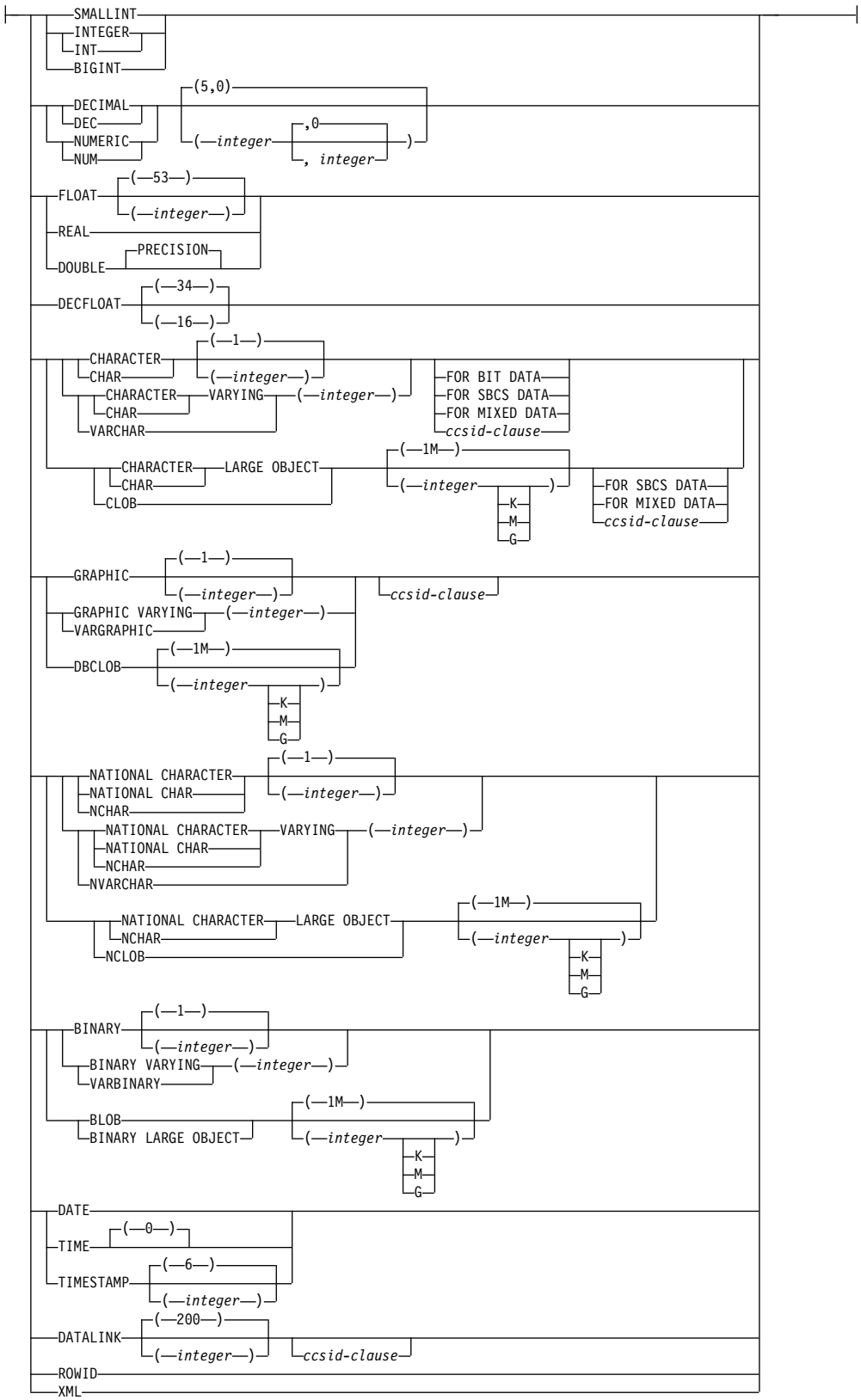




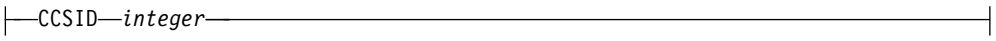
注:

1 *option-list* 内の文節は、どのような順番で指定してもかまいません。

built-in-type:



ccsid-clause:



説明

FUNCTION または SPECIFIC FUNCTION
変更する関数を識別します。 *function-name* は、現行サーバーに存在している SQL 表関数を識別していなければなりません。

指定した関数が変更されます。関数の所有者と関数に関するすべての特権は、保持されます。

FUNCTION *function-name*
関数を名前によって識別します。 *function-name* は厳密に 1 つの関数を示す必要があります。この関数には、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前の関数が複数ある場合、エラーが戻されます。

FUNCTION *function-name (parameter-type,...)*
関数を一意的に識別する関数シグニチャーによって、関数を識別します。 *function-name (parameter-type,...)* は、指定された関数シグニチャーを持つ関数を識別しなければなりません。指定されたパラメーターは、関数の作成時に指定された、対応する位置にあるデータ・タイプと一致していなければなりません。変更される特定の関数インスタンスを識別する場合、データ・タイプの数とデータ・タイプの論理連結が使用されます。データ・タイプの同義語は、一致として扱われます。デフォルトがあるパラメーターは、このシグニチャーに含まれていなければなりません。

function-name() を指定する場合、識別される関数にパラメーターを使用することはできません。

function-name
関数の名前を識別します。

(parameter-type,...)
関数のパラメーターを識別します。

非修飾の特殊タイプ名を指定する場合、データベース・マネージャーはその特殊タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、以下のいずれかを使用します。

- 中が空の括弧は、データ・タイプが一致しているか否かの判別時にデータベース・マネージャーによって属性が無視されることを示します。例えば、DEC() は、DEC(7,2) のデータ・タイプで定義された関数のパラメーターに一致するものとみなされます。ただし、精度値は特定のデータ・タイプ (REAL または DOUBLE) を示すため、中が空の括弧で FLOAT を指定することはできません。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を指定した場合、その値は、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプが FLOAT の場合、突き合わせはデータ・タイプ

(REAL または DOUBLE) に基づいて行われるので、精度は指定された値に厳密に一致している必要はありません。

- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。暗黙の長さは、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

FOR DATA 文節または CCSID 文節の指定はオプショナルです。いずれの文節も指定しないと、データ・タイプが一致するかどうかを判定する場合に、データベース・マネージャーが属性を無視することを示します。どちらか一方の文節を指定する場合は、CREATE FUNCTION ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

AS LOCATOR
関数が、このパラメーターのロケーターを受け取るように定義されることを示します。 AS LOCATOR を指定する場合は、データ・タイプは LOB または XML、あるいは LOB または XML に基づく特殊タイプでなければなりません。

SPECIFIC FUNCTION *specific-name*
関数を特定名によって識別します。 *specific-name* は、現行サーバーに存在する特定の関数を示している必要があります。

ALTER *option-list*
関数の 1 つ以上のオプションが変更されることを示します。 ALTER FUNCTION ALTER *option-list* が指定されてオプションが指定されない場合は、既存の関数定義での値が使用されます。各オプションの説明については、1151 ページの『CREATE FUNCTION (SQL 表)』を参照してください。

REPLACE *routine-specification*
オプションおよびパラメーターを含む既存の関数定義を、このステートメントで指定したものに置き換えることを示します。関数を置き換えると、すべてのオプションの値が置き換えられます。オプションを指定しない場合は、新規 SQL 表関数が作成されるときと同じデフォルトが使用されます。詳しくは、 1151 ページの『CREATE FUNCTION (SQL 表)』を参照してください。

ルーチンにコメントやラベルがある場合は、ルーチン定義からコメントやラベルが削除されます。

RESTRICT
任意の関数、マテリアライズ照会表、プロシージャー、トリガー、またはビューによって参照された場合は、関数が変更または置換されないことを示します。

(*parameter-declaration***,...)**
関数のパラメーターの数、および各パラメーターのデータ・タイプと名前を指定します。

SQL 関数に許可されるパラメーターの最大数は 2000 です。

parameter-name
パラメーター名を指定します。この名前は、関数の本体に含まれるパラメー

ターを参照するのに使用されます。この名前は、パラメーター・リスト内の他のパラメーター名 と同じものであってはなりません。

data-type3
入力パラメーターのデータ・タイプを指定します。CCSID が指定されている場合、関数に渡される前に、パラメーターはその CCSID に変換されます。CCSID が指定されていない場合は、CCSID は、関数の呼び出し時点における現行サーバーのデフォルトの CCSID によって決まります。

default-clause
パラメーターのデフォルト値を指定します。デフォルト値は、定数、特殊レジスター、グローバル変数、式、またはキーワード NULL にすることができます。式は、集約関数および列名を含まない、 196 ページの『式』で定義されている任意の式です。デフォルト値が指定されていない場合、パラメーターにデフォルト値がないため、呼び出し時に省略できません。式ストリングの最大長は 64K です。

デフォルトの式は、パラメーターのデータ・タイプに対して割り当ての互換性がなければなりません。

デフォルト式内でリスト中の数値定数を区切る区切り記号として使用するコンマの後には、スペースが 1 つ必要です。

配列タイプのパラメーターにデフォルトを指定することはできません。

RETURNS TABLE
関数の出力表を指定します。

パラメーターの数が *N* であるとすると、列の数は 2000-*N* 以下でなければなりません。

column-name
出力表の列の名前を指定します。同じ名前を何度も指定することはできません。

data-type2
出力のデータ・タイプと属性を指定します。

あらゆる組み込みデータ・タイプ (ただし、LONG VARCHAR または LONG VARGRAPHIC は除く) や特殊タイプを指定することができます。

CCSID が指定され、戻りデータの CCSID が異なる CCSID でコード化されている場合、データは指定された CCSID に変換されます。

CCSID が指定されていない場合、戻りデータの CCSID が異なる CCSID でコード化されている場合には、戻りデータはジョブの CCSID (グラフィック・ストリング戻り値の場合は、ジョブに関連したグラフィック CCSID) に変換されます。 変換時に文字が失われるのを防ぐために、関数から戻される文字をすべて表現できる CCSID を明示的に指定することを考慮してください。 これは、データ・タイプがグラフィック・ストリング・データの場合に特に重要です。 この場合、CCSID 1200 または 13488 (ユニコード・グラフィック・ストリング・データ) を使用することを考慮してください。

option-list
変更される関数のオプションのリスト。これらのオプションは、前記の ALTER

option-list に記載したものと同じです。具体的なオプションを指定しない場合は、新規関数が作成されるときと同じデフォルトが使用されます。詳しくは、1151 ページの『CREATE FUNCTION (SQL 表)』を参照してください。

SET OPTION-statement
関数を作成するときに使用するオプションを指定します。例えば、デバッグ可能な関数を作成するときは、次のステートメントを含めることができます。

```
SET OPTION DBGVIEW = *SOURCE
```

詳しくは、 1696 ページの『SET OPTION』を参照してください。

オプション CNULRQD、COMPILEOPT、NAMING、および SQLCA は、ALTER FUNCTION ステートメントでは使用できません。デフォルト値式を処理するときには、オプション ALWCPYDTA、
CONACC、DATFMT、DATSEP、DECFLTRND、DECMPT、DECRESULT、DFTRDBCOL、LANGID、
SQLCURRULE、SQLPATH、SRTSEQ、TGTRLS、TIMFMT、および TIMSEP が使用されます。

SQL-routine-body
複合ステートメントも含め、単一の SQL ステートメントを指定します。 SQL 関数の定義についての詳細は、 1771 ページの『第 8 章 SQL 制御ステートメント』を参照してください。

CONNECT、SET CONNECTION、RELEASE、DISCONNECT、COMMIT、ROLLBACK および SET TRANSACTION ステートメントを実行するプロシージャーへの呼び出しは、関数内では使用できません。

SQL-routine-body は、RETURN ステートメントが少なくとも 1 つは含まれていなければならず、関数の呼び出し時に RETURN ステートメントが 1 つ実行される必要があります。

REPLACE キーワードを指定する ALTER PROCEDURE (SQL)、ALTER FUNCTION (SQL スカラー)、および ALTER FUNCTION (SQL 表) は、*SQL-routine-body* では使用できません。

注

関数の定義または置換に関する一般考慮事項: 関数の定義に関する一般情報については、CREATE FUNCTION (SQL 表) を参照してください。 ALTER FUNCTION (SQL 表) を使用すると、関数の特権を保持したままで、個々の属性を変更できます。

カスケード効果: REPLACE が RESTRICT なしで指定され、関数シグニチャーまたは結果データ・タイプが変更される場合、この関数を参照する関数、マテリアライズ照会表、プロシージャー、トリガー、およびビューからの結果は、予測不能なものになる可能性があります。参照されるオブジェクトはすべて再作成する必要があります。

難読化されたステートメント: 難読化されたステートメントを使用して作成した関数を変更できます。ステートメントを変更すると、カタログに保存されているエンコ

ードされたバージョンのステートメントが変更されるため、 SQL ステートメントの最大長を超えることがあります。これが起こると、エラーが発行され、変更は失敗します。

NOT SECURED から **SECURED** への関数の変更**:** ALTER FUNCTION ステートメントが実行された後、関数はセキュアであると見なされます。Db2 は SECURED 属性を、ユーザー定義関数に対するすべての変更の監査手順をユーザーが確立したことを宣言するアサーションとして扱います。Db2 は、後続のすべての ALTER FUNCTION ステートメントがこの監査手順によってレビューされると想定します。

セキュアな関数内での他のユーザー定義関数の呼び出し**:** 行アクセス制御または列アクセス制御を使用している表を参照する SQL データ変更ステートメント内でセキュアなユーザー定義関数が参照されていて、そのセキュアなユーザー定義関数が他のユーザー定義関数を呼び出す場合、ネストされたユーザー定義関数はセキュアであるとは判定されません。こういったネストされた関数が機密データにアクセスする可能性がある場合、IBM i のデータベース・セキュリティー管理者機能の権限があるユーザーは、それらの関数がそのデータにアクセスすることを許可されていること、および、それらの関数に加えられるすべての変更に関して変更管理監査手順が確立されていることを確認する必要があります。

代替構文**:** 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード VARIANT と NOT VARIANT は、NOT DETERMINISTIC と DETERMINISTIC の同義語として使用することができます。
- キーワード NULL CALL と NOT NULL CALL は、CALLED ON NULL INPUT と RETURNS NULL ON NULL INPUT の同義語として使用できます。
- DETERMINISTIC の同義語として、キーワード IS DETERMINISTIC を使用できます。

例

カーディナリティーを 10,000 に設定するには、SQL 表関数の定義を変更してください。

```
ALTER FUNCTION GET_TABLE
  ALTER CARDINALITY 10000
```

ALTER MASK

ALTER MASK ステートメントは、現行サーバーに存在する列マスクを変更します。

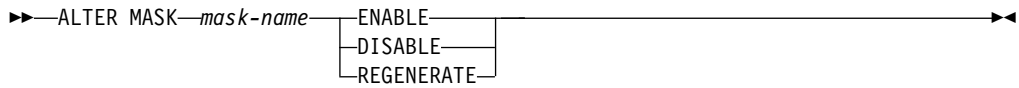
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの許可 ID には、セキュリティー管理者権限 がなければなりません。 21 ページの『管理権限』を参照してください。

構文



説明

mask-name
変更対象の列マスクを指定します。これは、現行サーバーに存在する列マスクを示すものでなければなりません。

ENABLE、DISABLE、または REGENERATE

ENABLE
列アクセス制御で列マスクを有効にするように指定します。表の列アクセス制御が現在活動化されていない場合、列マスクは、表の列アクセス制御が活動化されると有効になります。表の列アクセス制御が現在活動化されている場合、列マスクは直ちに有効になります。

有効にしようとするとエラーになる列マスクは、マスク定義にあるエラーがすべて解決されるまで有効にできません。これには、列マスクを除去し、変更した定義で再作成することが必要になることがあります。

列マスクが列アクセス制御に使用可能として既に定義されている場合、ENABLE は無視されます。

DISABLE
列アクセス制御のための列マスクを使用不可に設定することを指定します。表の列アクセス制御が現在活動化されていない場合、列マスクは、表の列アクセス制御が活動化されているときには、無効のままです。表の列アクセス制御が現在活動化されている場合、列マスクは使用不可になります。

列マスクが列アクセス制御には使用不可として既に定義されている場合、DISABLE は無視されます。

REGENERATE
列マスクを再生成することを指定します。カタログ内の列マスク定義が使用され、既存の従属関係と許可 (ある場合) は保持されます。列マスク定義

は、列マスクを作成したときと同じように再評価されます。列マスク定義で参照されるユーザー定義関数は、列マスクの作成時に解決されたのと同じセキュア UDF に解決される必要があります。

例

例 *1:* 列マスク M1 を使用可能に設定します。

```
ALTER MASK M1 ENABLE
```

例 *2:* 列マスク M1 を使用不可に設定します。

```
ALTER MASK M1 DISABLE
```

例 *3:* 列マスク M1 を再生成します。

```
ALTER MASK M1 REGENERATE
```

ALTER PERMISSION

ALTER PERMISSION ステートメントは、現行サーバーに存在する行の許可を変更します。

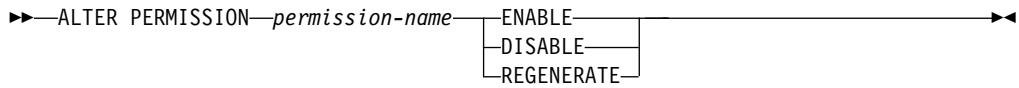
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの許可 ID には、セキュリティー管理者権限 がなければなりません。 21 ページの『管理権限』を参照してください。

構文



説明

permission-name
変更対象の行の許可を指定します。これは、現行サーバーに存在する行の許可を示すものでなければなりません。デフォルトの許可を指定することはできません。

ENABLE、DISABLE、または REGENERATE

ENABLE
行アクセス制御で行の許可を有効にすることを指定します。表の行アクセス制御が現在アクティブになっていない場合、行の許可は、表の行アクセス制御がアクティブになった時点で有効になります。表の行アクセス制御が現在アクティブになっている場合、行の許可は直ちに有効になります。

有効にしようとするとエラーになる行の許可は、許可定義にあるエラーがすべて解決されるまで有効にできません。これには、行の許可を除去し、変更した定義で再作成することが必要になることがあります。

行の許可が行アクセス制御に使用可能として既に定義されている場合、ENABLE は無視されます。

DISABLE
行アクセス制御で行の許可を無効にすることを指定します。表の行アクセス制御が現在活動化されていない場合、行の許可は、表の行アクセス制御が活動化されているときには無効のままです。表の行アクセス制御が現在活動化されている場合、行の許可は無効になります。

表の行アクセス制御が現在活動化されていて、すべての行の許可が無効になっている場合、デフォルトの行の許可 (表のどの行へのアクセスも許可されない) が使用されます。

行の許可が行アクセス制御には使用不可として既に定義されている場合、DISABLE は無視されます。

REGENERATE
行の許可を再生成することを指定します。カタログ内の行の許可定義が使用され、既存の従属関係と許可 (ある場合) は保持されます。行の許可定義は、行の許可を作成したときと同じように再評価されます。行の許可で参照されるユーザー定義関数は、行の許可の作成時に解決されたのと同じセキュア UDF に解決される必要があります。

例

例 *1:* 権限 P1 を使用可能に設定します。

```
ALTER PERMISSION P1 ENABLE
```

例 *2:* 権限 P1 を使用不可に設定します。

```
ALTER PERMISSION P1 DISABLE
```

例 *3:* 権限 P1 を再生成します。

```
ALTER PERMISSION P1 REGENERATE
```

ALTER PROCEDURE (外部)

ALTER PROCEDURE (外部) ステートメントは、現行サーバーで外部プロシージャーを変更します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

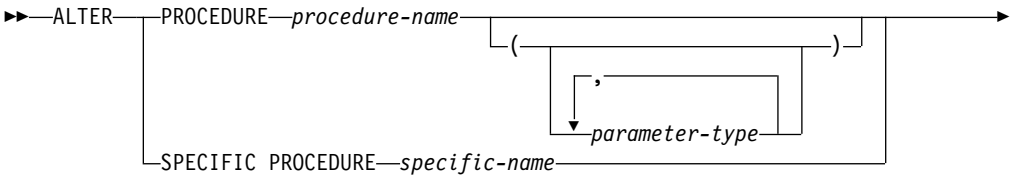
このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別されるプロシージャーに対しては次のもの。
  - そのプロシージャーに対する ALTER 特権、および
  - プロシージャーが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

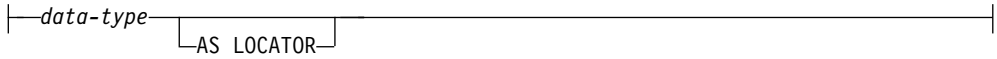
別の外部プログラムが指定されている場合、ステートメントの権限 ID によって保持される特権には、新規の外部プロシージャーを作成するために必要な同一特権が含まれていなければなりません。詳しくは、 1189 ページの『CREATE PROCEDURE (外部)』を参照してください。

SQL 特権に対応するシステム権限の説明については、『関数またはプロシージャーへの権限を検査する際の対応するシステム権限』、『表またはビューへの権限を検査する際の対応するシステム権限』、および『特殊タイプへの権限を検査する際の対応するシステム権限』を参照してください。

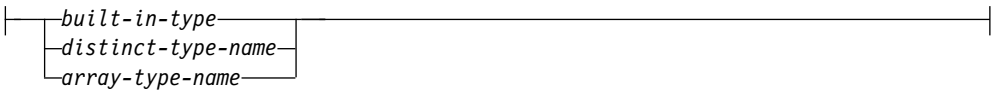
構文



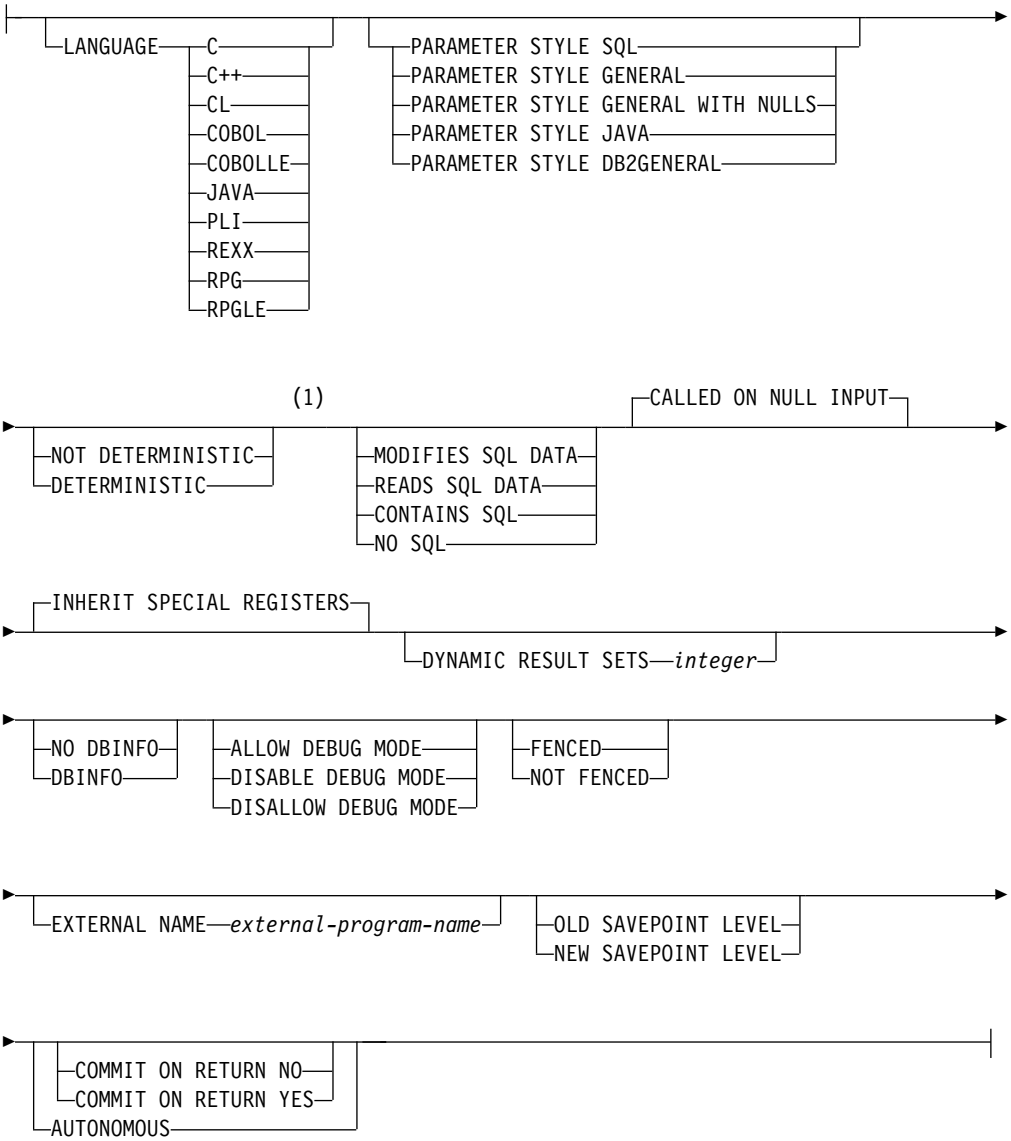
parameter-type:



data-type:



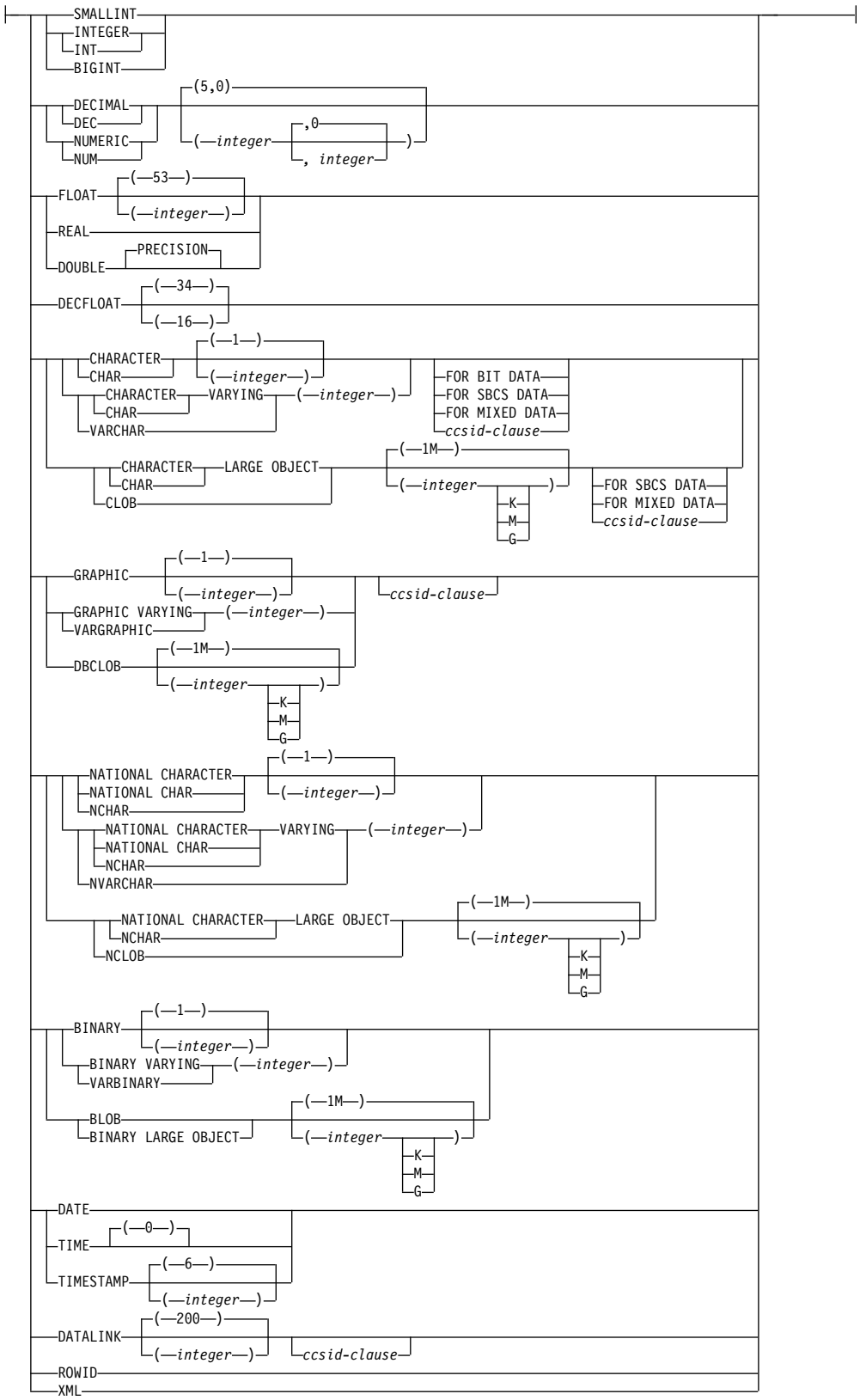
option-list:



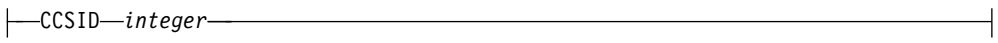
注:

1 *option-list* 内の文節は、どのような順番で指定してもかまいません。

built-in-type:



ccsid-clause:



説明

PROCEDURE または **SPECIFIC PROCEDURE**

変更するプロシージャーを識別します。*procedure-name* では、現行サーバーに存在する外部プロシージャーを指定する必要があります。

指定したプロシージャーが変更されます。プロシージャーの所有者は保持されます。プロシージャーが変更された時点で外部プログラムまたはサービス・プログラムが存在する場合、プロシージャーに対するすべての特権が保持されます。

PROCEDURE *procedure-name*

プロシージャーを名前によって識別します。 *procedure-name* では、ただ 1 つの外部プロシージャーを指定する必要があります。このプロシージャーには、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前のプロシージャーが複数ある場合、エラーが戻されます。

PROCEDURE *procedure-name (parameter-type,...)*

プロシージャーを一意的に識別するプロシージャー・シグニチャーによって、プロシージャーを識別します。 *procedure-name (parameter-type,...)* では、指定されたプロシージャー・シグニチャーを持つ 外部プロシージャーを識別する必要があります。指定されたパラメーターは、プロシージャーの作成時に指定された、対応する位置にあるデータ・タイプと一致していなければなりません。変更される特定のプロシージャー・インスタンスを識別する場合、データ・タイプの数とデータ・タイプの論理連結が使用されます。データ・タイプの同義語は、一致として扱われます。デフォルトがあるパラメーターは、このシグニチャーに含まれていなければなりません。

プロシージャー名 *()* を指定する場合、識別されるプロシージャーにパラメーターを使用することはできません。

procedure-name

プロシージャーの名前を識別します。

(parameter-type,...)

プロシージャーのパラメーターを識別します。

非修飾の特殊タイプ名または配列タイプ名を指定する場合、データベース・マネージャーはその特殊タイプまたは配列タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、以下のいずれかを使用します。

- 中が空の括弧は、データ・タイプが一致しているか否かの判別時にデータベース・マネージャーによって属性が無視されることを示します。例えば、DEC() は、DEC(7,2) のデータ・タイプで定義されたプロシージャーのパラメーターに一致するものとみなされます。ただし、精度値は特定のデータ・タイプ (REAL または DOUBLE) を示すため、中が空の括弧で FLOAT を指定することはできません。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を指定する場合、その値は、 CREATE PROCEDURE ステートメントの中で暗黙

的または明示的に指定された値と正確に一致している必要があります。データ・タイプが FLOAT の場合、突き合わせはデータ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度は指定された値に厳密に一致している必要はありません。

- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。暗黙の長さは、CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

FOR DATA 文節または CCSID 文節の指定はオプショナルです。いずれの文節も指定しないと、データ・タイプが一致するかどうかを判定する場合に、データベース・マネージャーが属性を無視することを示します。どちらか一方の文節を指定する場合は、CREATE PROCEDURE ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

AS LOCATOR
プロシージャーが、このパラメーターのロケーターを受け取るように定義されることを示します。 AS LOCATOR を指定する場合は、データ・タイプは LOB または XML、あるいは LOB または XML に基づく特殊タイプでなければなりません。

SPECIFIC PROCEDURE *specific-name*
プロシージャーを特定名によって識別します。 *specific-name* は、現行サーバーに存在している特定のプロシージャーを識別していなければなりません。

ALTER *option-list*
プロシージャーの 1 つ以上のオプションが変更されることを示します。オプションが指定されない場合は、既存のプロシージャー定義での値が使用されます。各オプションの説明については、 1189 ページの『CREATE PROCEDURE (外部)』を参照してください。

注

プロシージャーの定義または変更に関する一般考慮事項**:** プロシージャーの定義に関する一般情報については、CREATE PROCEDURE を参照してください。ALTER PROCEDURE (外部) を使用すると、プロシージャーに関する特権を保持したままで、個々の属性を変更できます。

代替構文**:** 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード VARIANT と NOT VARIANT は、NOT DETERMINISTIC と DETERMINISTIC の同義語として使用することができます。
- キーワード NULL CALL は、CALLED ON NULL INPUT の同義語として使用できます。
- DYNAMIC RESULT SET、RESULT SETS、および RESULT SET は、DYNAMIC RESULT SETS の同義語として使用できます。

- SQL の同義語として、値 DB2SQL を使用できます。

例

MYPROC プロシージャーの定義を変更して、そのプロシージャーの呼び出し時に起動する外部プログラムの名前を変更します。外部プログラムの名前は、PROG10A です。

```
ALTER PROCEDURE MYPROC
  EXTERNAL NAME PROG10A
```

ALTER PROCEDURE (SQL)

ALTER PROCEDURE (SQL) ステートメントは、現行サーバーでプロシージャーを変更します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

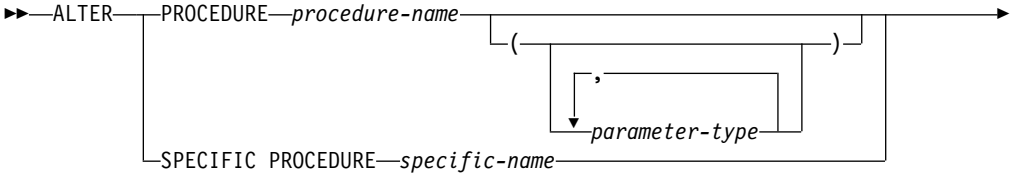
- ステートメント内で識別されるプロシージャーに対しては次のもの。
  - そのプロシージャーに対する ALTER 特権、および
  - プロシージャーが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

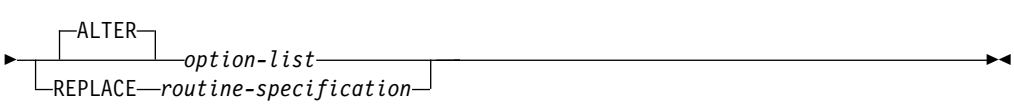
特殊タイプが *parameter-declaration* 内で参照される場合、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別された、それぞれの特殊タイプごとに、
  - その特殊タイプに対する USAGE 特権、および
  - 特殊タイプが含まれるスキーマに対する USAGE 特権
- 管理権限

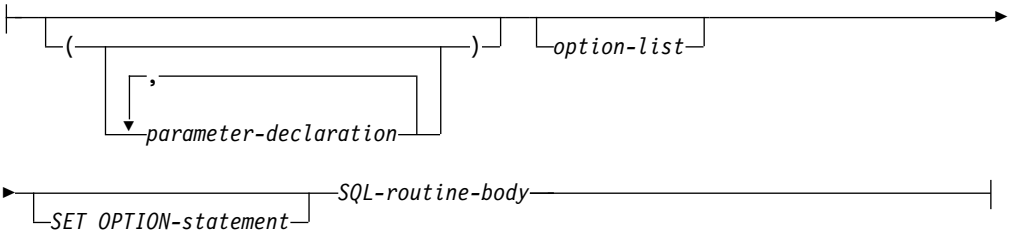
SQL 特権に対応するシステム権限の説明については、『関数またはプロシージャーへの権限を検査する際の対応するシステム権限』および 『特殊タイプへの権限を検査する際の対応するシステム権限』を参照してください。

構文

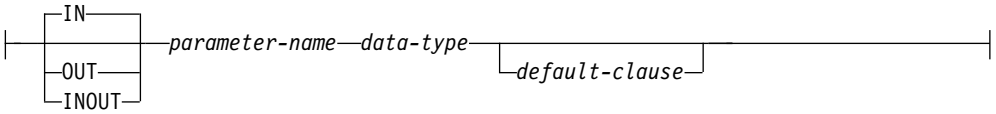




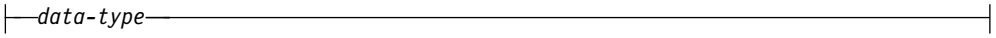
routine-specification:



parameter-declaration:



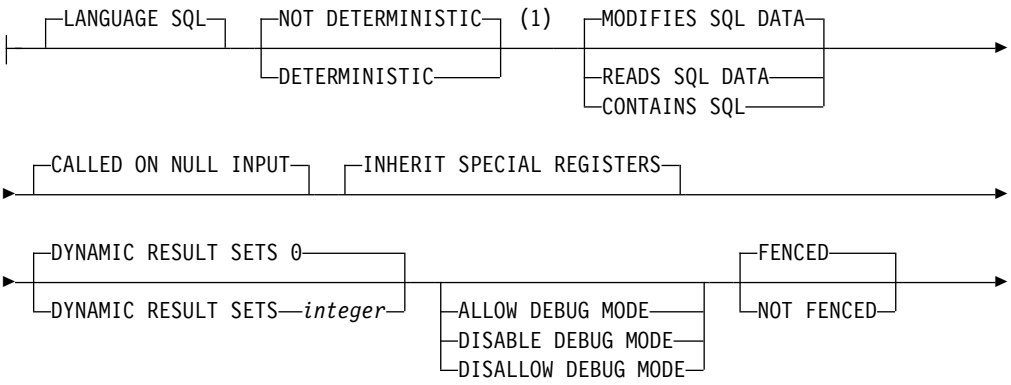
parameter-type:

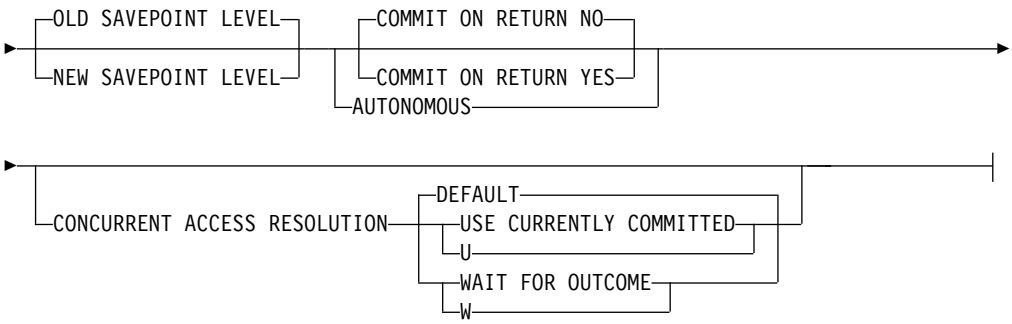


data-type:

default-clause:

option-list:

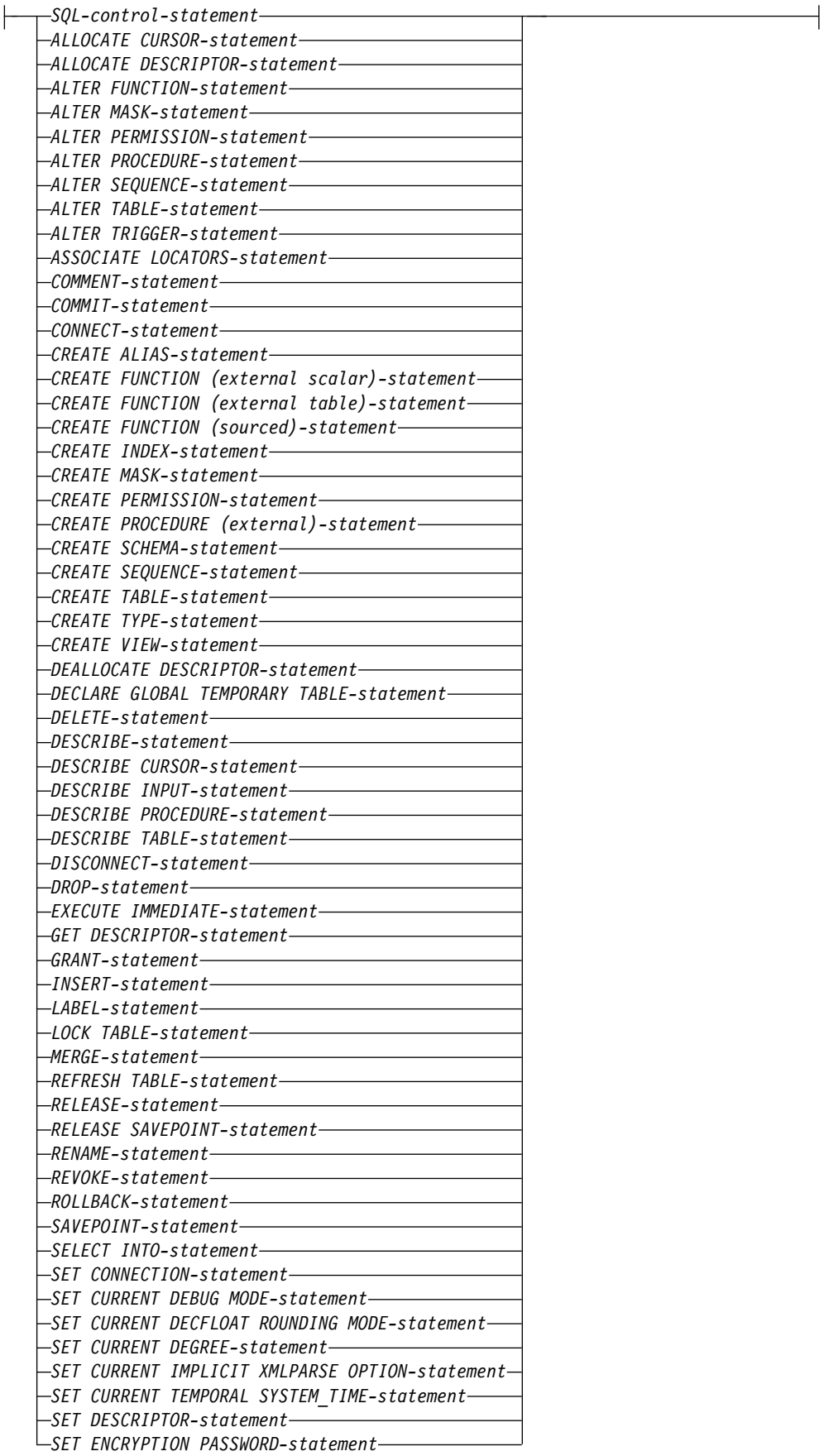




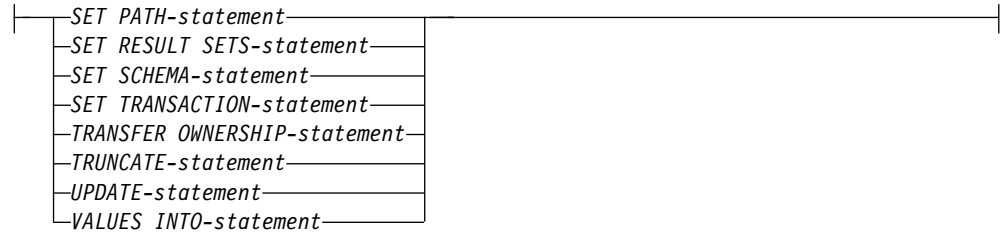
注:

1 *option-list* 内の文節は、どのような順番で指定してもかまいません。

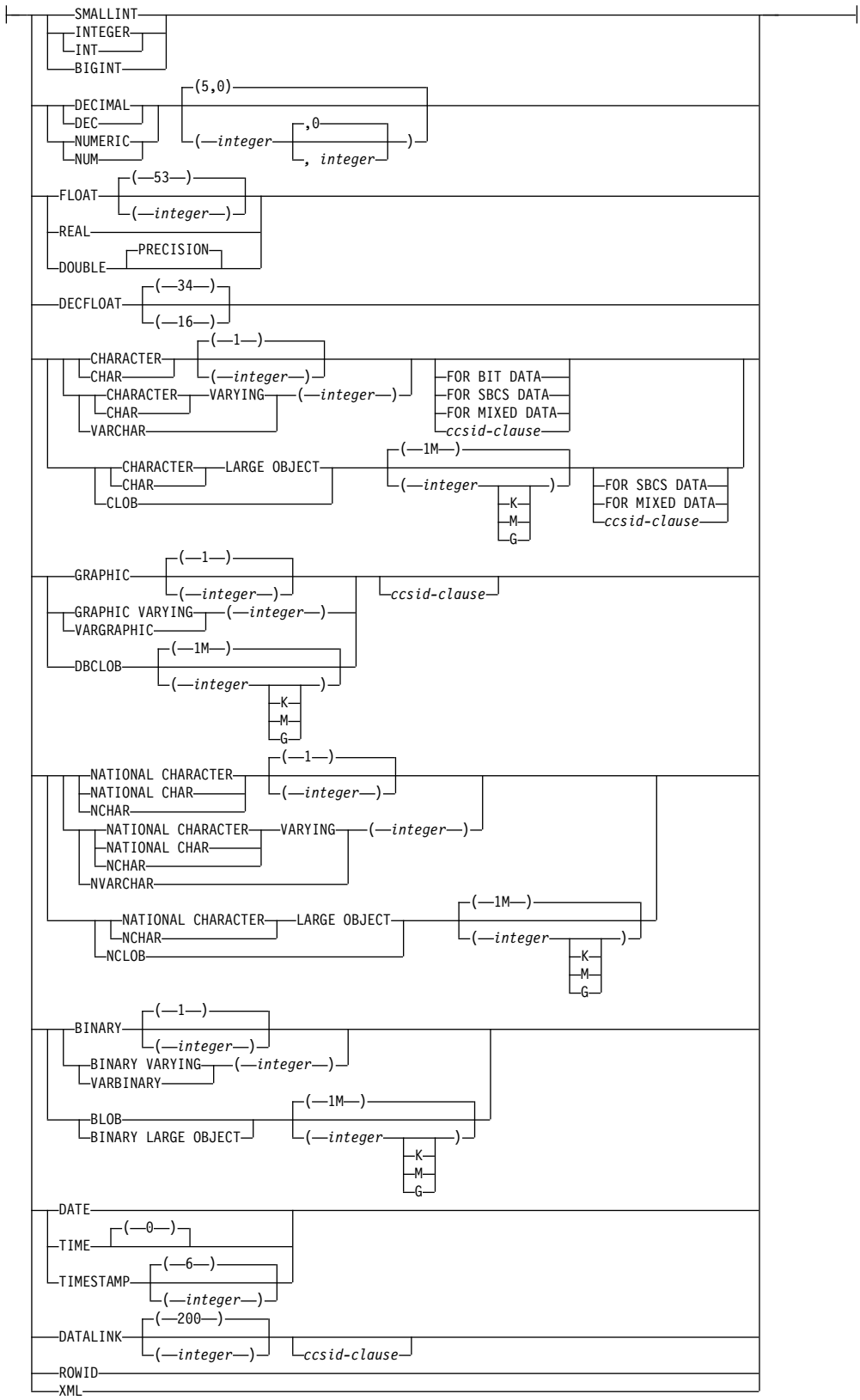
SQL-routine-body:


SQL-control-statement
ALLOCATE CURSOR-statement
ALLOCATE DESCRIPTOR-statement
ALTER FUNCTION-statement
ALTER MASK-statement
ALTER PERMISSION-statement
ALTER PROCEDURE-statement
ALTER SEQUENCE-statement
ALTER TABLE-statement
ALTER TRIGGER-statement
ASSOCIATE LOCATORS-statement
COMMENT-statement
COMMIT-statement
CONNECT-statement
CREATE ALIAS-statement
CREATE FUNCTION (external scalar)-statement
CREATE FUNCTION (external table)-statement
CREATE FUNCTION (sourced)-statement
CREATE INDEX-statement
CREATE MASK-statement
CREATE PERMISSION-statement
CREATE PROCEDURE (external)-statement
CREATE SCHEMA-statement
CREATE SEQUENCE-statement
CREATE TABLE-statement
CREATE TYPE-statement
CREATE VIEW-statement
DEALLOCATE DESCRIPTOR-statement
DECLARE GLOBAL TEMPORARY TABLE-statement
DELETE-statement
DESCRIBE-statement
DESCRIBE CURSOR-statement
DESCRIBE INPUT-statement
DESCRIBE PROCEDURE-statement
DESCRIBE TABLE-statement
DISCONNECT-statement
DROP-statement
EXECUTE IMMEDIATE-statement
GET DESCRIPTOR-statement
GRANT-statement
INSERT-statement
LABEL-statement
LOCK TABLE-statement
MERGE-statement
REFRESH TABLE-statement
RELEASE-statement
RELEASE SAVEPOINT-statement
RENAME-statement
REVOKE-statement
ROLLBACK-statement
SAVEPOINT-statement
SELECT INTO-statement
SET CONNECTION-statement
SET CURRENT DEBUG MODE-statement
SET CURRENT DECFLOAT ROUNDING MODE-statement
SET CURRENT DEGREE-statement
SET CURRENT IMPLICIT XMLPARSE OPTION-statement
SET CURRENT TEMPORAL SYSTEM_TIME-statement
SET DESCRIPTOR-statement
SET ENCRYPTION PASSWORD-statement

SQL-routine-body (続き):



built-in-type:



ccsid-clause:

```
├──CCSID──integer──────────────────────────────────────────────────────┤
```

説明

PROCEDURE または **SPECIFIC PROCEDURE**

変更するプロシージャーを識別します。プロシージャー名 は、現行サーバーに存在している SQL プロシージャーを識別していなければなりません。

指定したプロシージャーが変更されます。プロシージャーの所有者とプロシージャーに関するすべての特権は、保持されます。

PROCEDURE *procedure-name*

プロシージャーを名前によって識別します。 *procedure-name* は、ただ 1 つの SQL プロシージャーを識別していなければなりません。このプロシージャーには、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前のプロシージャーが複数ある場合、エラーが戻されます。

PROCEDURE *procedure-name (parameter-type,...)*

プロシージャーを一意的に識別するプロシージャー・シグニチャーによって、プロシージャーを識別します。 *procedure-name (parameter-type,...)* では、指定されたプロシージャー・シグニチャーを持つ SQL プロシージャーを識別する必要があります。指定されたパラメーターは、プロシージャーの作成時に指定された、対応する位置にあるデータ・タイプと一致していなければなりません。変更される特定のプロシージャー・インスタンスを識別する場合、データ・タイプの数とデータ・タイプの論理連結が使用されます。データ・タイプの同義語は、一致として扱われます。デフォルトがあるパラメーターは、このシグニチャーに含まれていなければなりません。

プロシージャー名 *()* を指定する場合、識別されるプロシージャーにパラメーターを使用することはできません。

procedure-name

プロシージャーの名前を識別します。

(parameter-type,...)

プロシージャーのパラメーターを識別します。

非修飾の特殊タイプ名または配列タイプ名を指定する場合、データベース・マネージャーはその特殊タイプまたは配列タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、以下のいずれかを使用します。

- 中が空の括弧は、データ・タイプが一致しているか否かの判別時にデータベース・マネージャーによって属性が無視されることを示します。例えば、DEC() は、DEC(7,2) のデータ・タイプで定義されたプロシージャーのパラメーターに一致するものとみなされます。ただし、精度値は特定のデータ・タイプ (REAL または DOUBLE) を示すため、中が空の括弧で FLOAT を指定することはできません。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を指定する場合、その値は、 CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要がありま

す。データ・タイプが FLOAT の場合、突き合わせはデータ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度は指定された値に厳密に一致している必要はありません。

- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。暗黙の長さは、CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

FOR DATA 文節または CCSID 文節の指定はオプショナルです。いずれの文節も指定しないと、データ・タイプが一致するかどうかを判定する場合に、データベース・マネージャーが属性を無視することを示します。どちらか一方の文節を指定する場合は、CREATE PROCEDURE ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

AS LOCATOR
プロシージャーが、このパラメーターのロケーターを受け取るように定義されることを示します。 AS LOCATOR を指定する場合は、データ・タイプは LOB または LOB に基づく特殊タイプでなければなりません。

SPECIFIC PROCEDURE *specific-name*
プロシージャーを特定名によって識別します。 *specific-name* は、現行サーバーに存在している特定のプロシージャーを識別していなければなりません。

ALTER *option-list*
プロシージャーの 1 つ以上のオプションが変更されることを示します。ALTER PROCEDURE ALTER *option-list* が指定されてオプションが指定されない場合は、既存のプロシージャー定義での値が使用されます。各オプションの説明については、 1208 ページの『CREATE PROCEDURE (SQL)』を参照してください。

REPLACE *routine-specification*
オプションおよびパラメーターを含む既存のプロシージャー定義を、このステートメントで指定したものに置き換えることを示します。プロシージャーを置き換えると、すべてのオプションの値が置き換えられます。オプションを指定しない場合は、新規 SQL プロシージャーが作成されるときと同じデフォルトが使用されます。詳しくは、 1208 ページの『CREATE PROCEDURE (SQL)』を参照してください。

ルーチンにコメントやラベルがある場合は、ルーチン定義からコメントやラベルが削除されます。

(*parameter-declaration***,...)**
プロシージャーのパラメーターの数、および各パラメーターのデータ・タイプと名前を指定します。プロシージャーに関するパラメーターは、入力専用、出力専用、または入出力両用に使用できます。

SQL プロシージャーに許されるパラメーターの最大数は 2000 です。

IN パラメーターが、プロシージャーへの入力パラメーターであることを指定します。

OUT
パラメーターが、プロシージャーから戻される出力パラメーターであることを示します。プロシージャー内でこのパラメーターが設定されていない場合は、NULL 値が戻されます。

INOUT
パラメーターが、このプロシージャー用の入出力両方のパラメーターであることを指定します。プロシージャー内でパラメーターが設定されなければ、入力値が戻されます。INOUT パラメーターがデフォルトと共に定義されていて、プロシージャーの呼び出し時にそのデフォルトが使用される場合、パラメーターに戻される値はありません。

parameter-name
SQL 変数として使用するパラメーターの名前を指定します。この名前は、このプロシージャー用の他の *parameter-name* と同じものであってはなりません。

data-type
パラメーターのデータ・タイプを指定します。 CCSID が指定されている場合、プロシージャーに渡される前に、パラメーターはその CCSID に変換されます。CCSID が指定されていない場合は、CCSID は、プロシージャーの呼び出し時点における現行サーバーのデフォルトの CCSID によって決まります。

default-clause
パラメーターのデフォルト値を指定します。デフォルト値は、定数、特殊レジスター、グローバル変数、式、またはキーワード NULL にすることができます。式は、集約関数および列名を含まない、 196 ページの『式』で定義されている任意の式です。デフォルト値が指定されていない場合、パラメーターにデフォルト値がないため、呼び出し時に省略できません。式ストリングの最大長は 64K です。

デフォルトの式で SQL データを変更することはできません。式は、パラメーターのデータ・タイプに対して割り当ての互換性がなければなりません。デフォルト式の中で参照されるすべてのオブジェクトは、プロシージャーが作成されるときに存在している必要があります。

デフォルト式内でリスト中の数値定数を区切る区切り記号として使用するコンマの後には、スペースが 1 つ必要です。

デフォルトを指定できません。

- OUT パラメーターに対して
- 配列タイプのパラメーターに対して

option-list
変更されるプロシージャーのオプションのリスト。これらのオプションは、前記の ALTER *option-list* に記載したものと同じです。具体的なオプションを指定しない場合は、新規プロシージャーが作成されるときと同じデフォルトが使用されます。 詳しくは、 1208 ページの『CREATE PROCEDURE (SQL)』を参照してください。

SET OPTION-statement
プロシージャーを作成するときに使用するオプションを指定します。例えば、デバッグ可能なプロシージャーを作成するときは、次のステートメントを含めることができます。

```
SET OPTION DBGVIEW = *SOURCE
```

詳しくは、 1696 ページの『SET OPTION』を参照してください。

オプション CLOSQLCSR、CNULRQD、COMPILEOPT、NAMING、SQLCA は、ALTER PROCEDURE ステートメントでは使用できません。デフォルト値式を処理するときには、オプション ALWCPYDTA、
CONACC、DATFMT、DATSEP、DECFLTRND、DECMPT、DECRESULT、DFTRDBCOL、LANGID、
SQLCURRULE、SQLPATH、SRTSEQ、TGTRLS、TIMFMT、および TIMSEP が使用されます。

SQL-routine-body
複合ステートメントも含め、単一の SQL ステートメントを指定します。SQL プロシージャーの定義に関する詳細については、 1771 ページの『第 8 章 SQL 制御ステートメント』 を参照してください。

CONNECT、SET CONNECTION、RELEASE、DISCONNECT、および SET TRANSACTION ステートメントは、リモート・アプリケーション・サーバー上で実行中のプロシージャー内で使用することはできません。 COMMIT および ROLLBACK ステートメントは、 ATOMIC SQL プロシージャーまたはリモート・アプリケーション・サーバーへの接続上で実行中のプロシージャー内で使用することはできません。

REPLACE キーワードを指定する ALTER PROCEDURE (SQL)、ALTER FUNCTION (SQL スカラー)、および ALTER FUNCTION (SQL 表) は、*SQL-routine-body* では使用できません。

注

プロシージャーの定義または置換に関する一般考慮事項**:** プロシージャーの定義に関する一般情報については、CREATE PROCEDURE を参照してください。ALTER PROCEDURE (SQL) を使用すると、プロシージャーに関する特権を保持したままで、個々の属性またはルーチン指定を変更できます。

ALTER PROCEDURE REPLACE に関する考慮事項**:** SQL プロシージャー定義が置き換えられる際、SQL は、組み込み SQL ステートメントと一緒に C ソース・コードが収められる一時ソース・ファイルを作成します。次いで、CRTPGM コマンドを使用して、プログラム・オブジェクトを作成します。プログラムの作成に使用される SQL オプションは、ALTER PROCEDURE (SQL) ステートメントの実行時に有効なオプションです。プログラムは、ACTGRP(*CALLER) を使用して作成します。

SQL プロシージャーが変更された場合、新規 *PGM または *SRVPGM オブジェクトが作成され、プロシージャーの属性は、作成されたプログラム・オブジェクトに保管されます。*PGM オブジェクトや *SRVPGM オブジェクトが保管された上でこのシステムや別のシステムに復元されると、カタログは、それらの属性を使用して自動的に更新されます。

特定名は、それが有効なシステム名である場合は、ソース・ファイルのメンバーの名前、ならびに、プログラム・オブジェクトの名前として使用されます。プロシージャー名が有効なシステム名でない場合は、固有名が生成されます。同じ名前のソース・ファイル・メンバーが既に存在している場合は、そのメンバーがオーバーレイされます。同じ名前のモジュールやプログラムが既に存在している場合、オブジェクトはオーバーレイされずに、固有名が生成されます。これらの固有名は、システム表名の生成に関する規則に従って生成されます。

ターゲット・リリースに関する考慮事項: SQL プロシージャー定義が置き換えられる場合、ターゲット・リリースは、ユーザーが明示的に別のターゲット・リリースを指定しないかぎり、ALTER ステートメントが実行される現行リリースになります。ターゲット・リリースは、SET OPTION ステートメントで TGTRLS キーワードを使用することで明示的に指定できます。 ALTER が RUNSQLSTM または CRTSQLxxx コマンドのソースで指定された場合、TGTRLS キーワードもそのコマンドに指定できます。

プロシージャー定義が置換されない場合、既存プロシージャーのターゲット・リリースが保持されます。ただし、プロシージャーのターゲット・リリース・レベルが、最も初期にサポートされたリリース・レベルより前である場合を除きます。この場合、ターゲット・リリースは、最も初期にサポートされたリリース・レベルに変更されます。

難読化されたステートメント: 難読化されたステートメントを使用して作成したプロシージャーを変更できます。ステートメントを変更すると、カタログに保存されているエンコードされたバージョンのステートメントが変更されるため、 SQL ステートメントの最大長を超えることがあります。これが起こると、エラーが発行され、変更は失敗します。

代替構文: 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード VARIANT と NOT VARIANT は、NOT DETERMINISTIC と DETERMINISTIC の同義語として使用することができます。
- キーワード NULL CALL は、CALLED ON NULL INPUT の同義語として使用できます。
- DYNAMIC RESULT SET、RESULT SETS、および RESULT SET は、DYNAMIC RESULT SETS の同義語として使用できます。

例

SQL プロシージャーからの戻り時に SQL の変更がコミットされるように SQL プロシージャーでの定義を変更します。

```
ALTER PROCEDURE UPDATE_SALARY_2
  ALTER COMMIT ON RETURN YES
```

ALTER SEQUENCE

ALTER SEQUENCE ステートメントを使用して、シーケンスを変更できます。

ALTER SEQUENCE ステートメントを使用して、シーケンスを以下のように変更できます。

- シーケンスを再始動する
- 将来のシーケンス値の間の増分を変更する
- 最小値または最大値を設定または除去する
- キャッシュ済みシーケンス番号の数を変更する
- シーケンスが循環するかどうかを決定する属性を変更する
- 要求の順序でシーケンス番号が生成されるかどうかを変更する

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別されるシーケンスに対しては次のもの。
  - シーケンスが含まれるスキーマに対する USAGE 特権
  - シーケンスについての ALTER 権限
- データベース管理者権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 次のシステム権限
  - Change Data Area (CHGDTAARA) コマンドに対する *USE
  - Retrieve Data Area (RTVDTAARA) コマンドに対する *USE
- 管理権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- SYSSEQOBJECTS カタログ表の場合
  - その表に対する UPDATE 特権、および
  - スキーマ QSYS2 に対する USAGE 特権
- 管理権限

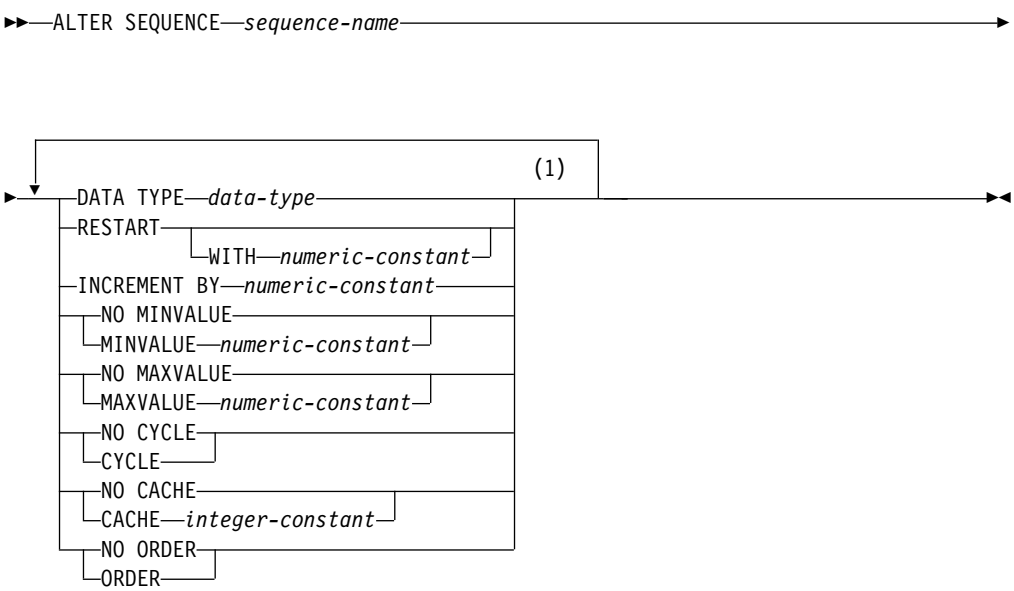
特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別される特殊タイプに対しては次のもの。
  - その特殊タイプに対する USAGE 特権、および

– 特殊タイプが含まれるスキーマに対する USAGE 特権
- 管理権限

SQL 特権に対応するシステム権限については、『シーケンスへの権限を検査する際の対応するシステム権限』、『表またはビューへの権限を検査する際の対応するシステム権限』、および『特殊タイプへの権限を検査する際の対応するシステム権限』を参照してください。

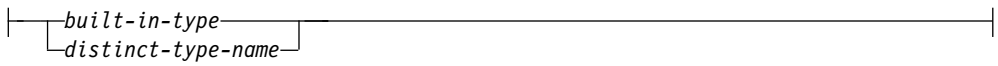
構文



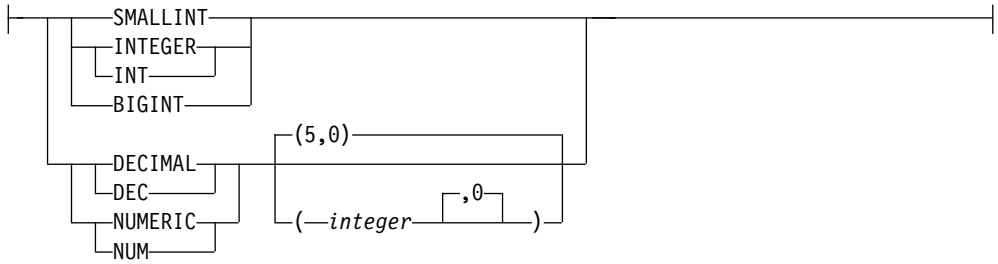
注:

1 同じ文節を複数回指定することはできません。

data-type:



built-in-type:



説明

sequence-name
変更されるシーケンスを識別します。 この名前は、現行サーバーに既に存在するシーケンスを示すものでなければなりません。

DATA TYPE *data-type*
シーケンス値に使用する新規のデータ・タイプを指定します。データ・タイプは、厳密に位取りがゼロの数値タイプ (SMALLINT、INTEGER、BIGINT、DECIMAL、または NUMERIC)、またはソース・タイプが厳密に位取りがゼロの数値タイプであるユーザー定義の特殊タイプにすることができます。

ALTER SEQUENCE ステートメントによって変更されない既存の START WITH、INCREMENT BY、MINVALUE、および MAXVALUE 属性のそれぞれに、新規データ・タイプに関連したデータ・タイプの列に割り当て可能な値が含まれている必要があります。

built-in-type
シーケンスの内部表示のベースとして使用される新規の組み込みデータ・タイプを指定します。データ・タイプが DECIMAL または NUMERIC である場合、精度は 63 以下、位取りは 0 でなければなりません。各組み込みデータ・タイプについての詳細は、 1238 ページの『CREATE TABLE』を参照してください。

プラットフォーム間でのアプリケーションの移植性を保つには、NUMERIC データ・タイプの代わりに DECIMAL を使用します。

distinct-type-name
シーケンスの新規データ・タイプが、特殊タイプ (ユーザー定義のデータ・タイプ) であることを指定します。ソース・タイプが DECIMAL または NUMERIC である場合、シーケンスの精度は該当する特殊タイプのソース・タイプの精度になります。ソース・タイプの精度は 63 以下、また位取りは 0 でなければなりません。スキーマ名なしの特殊タイプを指定すると、その特殊タイプ名は、SQL パス上のスキーマを検索することで解決されます。

RESTART
シーケンスを再開始します。数値定数 を指定していない場合は、このシーケンスを最初に作成した CREATE SEQUENCE ステートメントとに暗黙的または明示的に指定されている開始値から、シーケンスが再開始されます。

WITH *numeric-constant*
指定した値でシーケンスを再始動します。小数点の右側にゼロ以外の数字がないことを条件として、シーケンスに関連したデータ・タイプの列に割り当てることのできる任意の正または負の値を指定できます。

INCREMENT BY*numeric-constant*
シーケンスの連続した値の間隔を指定します。小数点の右側にゼロ以外の数字がないことと、長精度整数定数の値を超えないことを条件として、シーケンスに関連したデータ・タイプの列に割り当てることのできる任意の正または負の値を指定できます。

この値が負の場合は、降順になります。この値が 0 または正の場合は、ALTER ステートメントの後が昇順になります。

NO MINVALUE または MINVALUE
降順シーケンスが値の生成を循環または停止する最小値、あるいは最大値に達した後、昇順シーケンスが循環する最小値を指定します。

NO MINVALUE
昇順シーケンスの場合、値はオリジナルの開始値です。降順シーケンスの場合、シーケンスに関連するデータ・タイプ (および精度 (DECIMAL または NUMERIC の場合)) の最小値です。

MINVALUE*numeric-constant*
このシーケンス用として生成される最小値を示す数値定数を指定します。小数点の右側にゼロ以外の数字がないことを条件として、シーケンスに関連したデータ・タイプの列に割り当てることのできる任意の正または負の値を指定できます。値は、最大値またはそれより小さい値でなければなりません。

NO MAXVALUE または MAXVALUE
昇順シーケンスが値の生成を循環または停止する最大値、あるいは最小値に達した後、降順シーケンスが循環する最大値を指定します。

NO MAXVALUE
昇順シーケンスの場合、シーケンスに関連するデータ・タイプ (および精度 (DECIMAL または NUMERIC の場合)) の最大値です。降順シーケンスの場合、値はオリジナルの開始値です。

MAXVALUE *numeric-constant*
このシーケンス用として生成される最大値を示す数値定数を指定します。小数点の右側にゼロ以外の数字がないことを条件として、シーケンスに関連したデータ・タイプの列に割り当てることのできる任意の正または負の値を指定できます。値は、最小値またはそれより大きい値でなければなりません。

CYCLE または NO CYCLE
シーケンスの最大値または最小値に達した後も、このシーケンスについて値を生成し続けるかどうかを指定します。

NO CYCLE
シーケンスの最大値または最小値に達した後は、このシーケンスについて値を生成しないことを指定します。

CYCLE
最大値または最小値に達した後も、このシーケンスの値を生成し続けることを指定します。このオプションを使用した場合は、昇順シーケンスがシーケンスの最大値に達した後では、最小値が生成されます。降順シーケンスがシーケンスの最小値に達した後は、最大値が生成されます。シーケンスの最大値と最小値によって、循環に使用される範囲が決まります。

CYCLE が有効な場合、データベース・マネージャーによって 1 つのシーケンスに対して重複する値が生成される可能性があります。

CACHE または NO CACHE
事前割り振りの値をメモリー内に保持するかどうかを指定します。値を事前に割り振ってキャッシュに保管しておくと、NEXT VALUE シーケンス式のパフォーマンスが向上します。

CACHE *integer-constant*
事前割り振りされてメモリーに保持されるシーケンス値の最大数を指定しま

す。値を事前割り振りしてキャッシュに保管することにより、シーケンスの値が生成されるときの同期入出力が減少します。

システム障害のような特定の状態になると、キャッシュに保管されていてコミット済みステートメントでまだ使用されていないシーケンス値はすべて失われるため、その後使用されることはありません。 CACHE オプションに指定する値は、こうした状態で失われる可能性のあるシーケンス値の最大数です。

最小値は 2 です。

NO CACHE
シーケンスの値を事前割り振りしないことを指定します。これによって、システム障害のような状態になっても値が失われることはなくなります。このオプションを指定すると、シーケンスの値はキャッシュに保管されません。この場合は、シーケンスの新しい値が要求されるたびに同期入出力が発生します。

NO ORDER または ORDER
シーケンス番号を要求の順序どおりに生成する必要があるかどうかを指定します。

NO ORDER
シーケンス番号を要求の順序どおりに生成する必要がないことを指定します。

ORDER
シーケンス番号を要求の順序どおりに生成するように指定します。ORDER を指定すると、 NO ORDER を指定した場合よりも NEXT VALUE シーケンス式のパフォーマンスが低下します。

注

シーケンスの変更:

- 今後のシーケンス番号だけが ALTER SEQUENCE ステートメントによって影響を受けます。
- シーケンスが変更されると、キャッシュ済みの値はすべて失われます。
- シーケンスを再始動した後、または CYCLE に変更した後、生成される値が以前にそのシーケンスに対して生成された値と重複する可能性があります。

代替構文: 以下のキーワードは、他の Db2 製品の旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード NOMINVALUE、NOMAXVALUE、NOCYCLE、NOCACHE、および NOORDER を、 NO MINVALUE、NO MAXVALUE、NO CYCLE、NO CACHE、および NO ORDER の同義語として使用することができます。

例

数値なしで RESTART を指定する理由として考えられるのは、シーケンスを START WITH 値にリセットすることです。この例では、1 から表の行数までの数値を生成し、一時表を使用して表に追加した列にその数値を挿入しています。

```
ALTER SEQUENCE ORG_SEQ RESTART

DECLARE GLOBAL TEMPORARY TABLE TEMP_ORG AS
  (SELECT NEXT VALUE FOR ORG_SEQ, ORG.*
  FROM ORG) WITH DATA

INSERT INTO TEMP_ORG
  SELECT NEXT VALUE FOR ORG_SEQ, ORG.*
  FROM ORG
```

以降使用する時には、すべての結果行に番号が付けられて結果が返されます。

```
ALTER SEQUENCE ORG_SEQ RESTART

SELECT NEXT VALUE FOR ORG_SEQ, ORG.*
  FROM ORG
```

ALTER TABLE

ALTER TABLE ステートメントは表の定義を変更します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメントで識別される表に対して、
  - その表の ALTER 特権、および
  - 表が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

外部キーを定義する場合、ステートメントの権限 ID が保持する特権には、親表に関して少なくとも次の 1 つが含まれていなければなりません。

- 該当の表に対する REFERENCES 特権またはオブジェクト管理権限。
- 指定された親キーの各列に対する REFERENCES 特権。
- データベース管理者権限

フィールド・プロシージャーを定義する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- 次のシステム権限
  - プログラムに対する *EXECUTE システム権限、および
  - プログラムが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

選択ステートメント を指定する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つがこれらの文節で指定された表またはビューにおいて含まれなければなりません。

- 表またはビューに対する SELECT 特権
- データベース管理者権限

特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別された、それぞれの特殊タイプごとに、
  - その特殊タイプに対する USAGE 特権、および
  - 特殊タイプが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

ACTIVATE または DEACTIVATE 節が指定される場合は、次のとおりです。

- このステートメントの許可 ID には、セキュリティー管理者権限 がなければなりません。 21 ページの『管理権限』を参照してください。

行の許可がアクティブになっている表から DELETE ROWS でパーティションを削除するには、このステートメントの権限 ID には次の 1 つ以上が含まれなければなりません。

- 表についての *OBJOPR および *OBJEXIST システム権限。
- データベース管理者権限

パーティションをアタッチするには、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つも含まれなければなりません。

- ステートメントで識別されるソース表に対して、
  - SELECT 特権、および
  - *OBJEXIST システム権限、および
  - ソース表が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

パーティションをデタッチするには、ステートメントの権限 ID によって保持される特権にも、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメントで識別されるソース表に対して、
  - SELECT、DELETE、ALTER 特権、および
  - ソース表が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

パーティションをデタッチするには、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

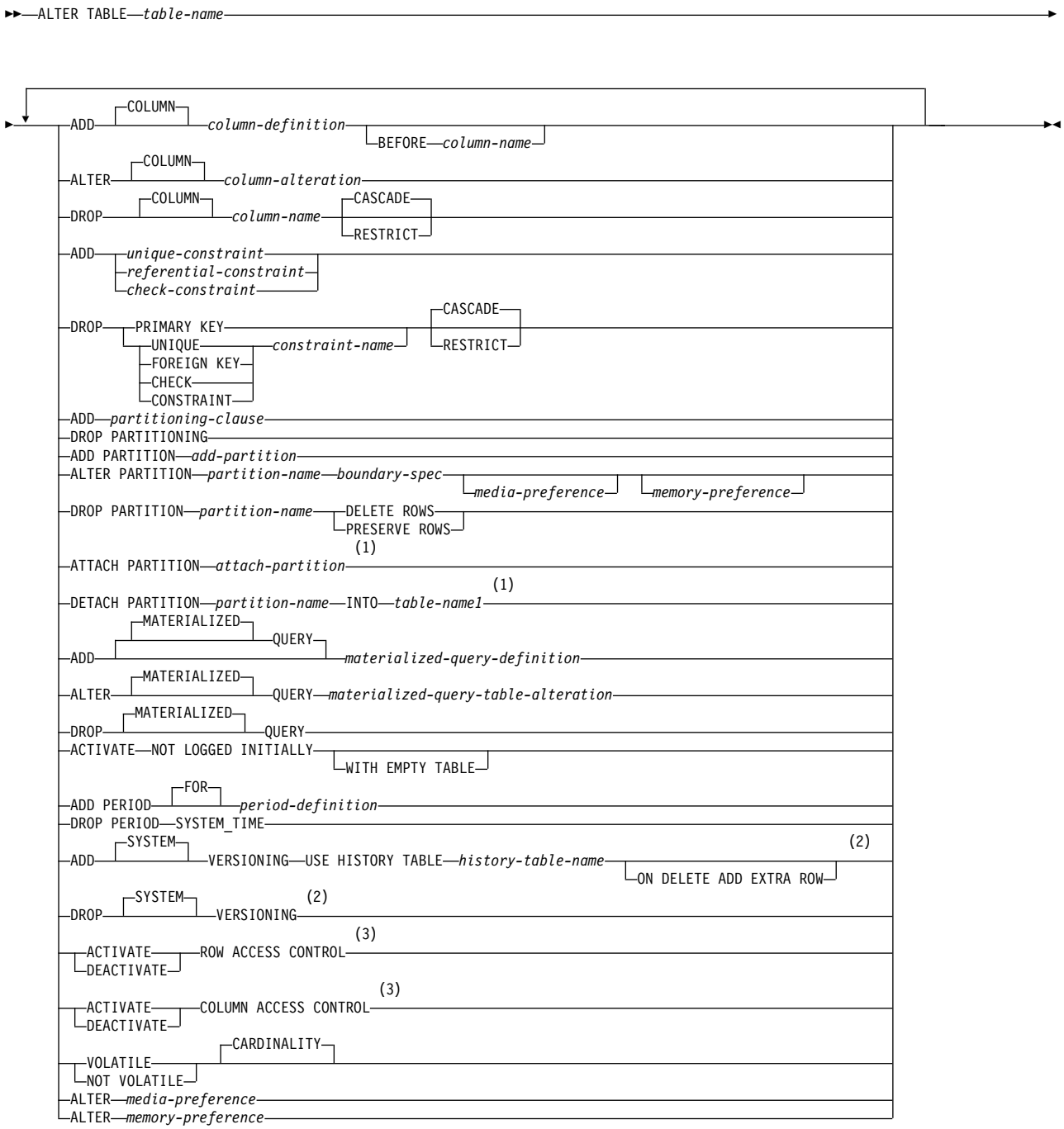
- スキーマ内に作成する特権。詳しくは、スキーマ内で作成する必要のある権限を参照してください。
- データベース管理者権限

表をシステム期間テンポラル表になるように変更する場合 (ADD VERSIONING 節を使用)、関連する履歴表にも変更が発生するような変更を 1 つ以上含む、システム期間テンポラル表への変更を行う場合、または、DROP VERSIONING を実行する場合、ステートメントの許可 ID の保持する特権に、以下の権限も少なくとも 1 つ含まれている必要があります。

- 次のシステム権限
  - 履歴表の ALTER 特権、および
  - 履歴表が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

SQL 特権に対応するシステム権限の説明については、『表またはビューへの権限を検査する際の対応するシステム権限』および『特殊タイプへの権限を検査する際の対応するシステム権限』を参照してください。

構文



注:

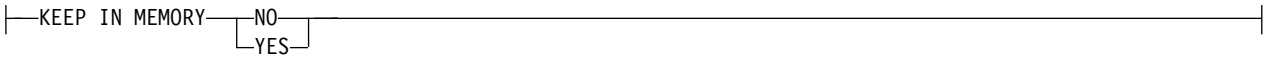
1 ATTACH PARTITION または DETACH PARTITION を、他の節と一緒に指定することはできません。

2 ADD VERSIONING または DROP VERSIONING を、他の節と一緒に指定することはできません。

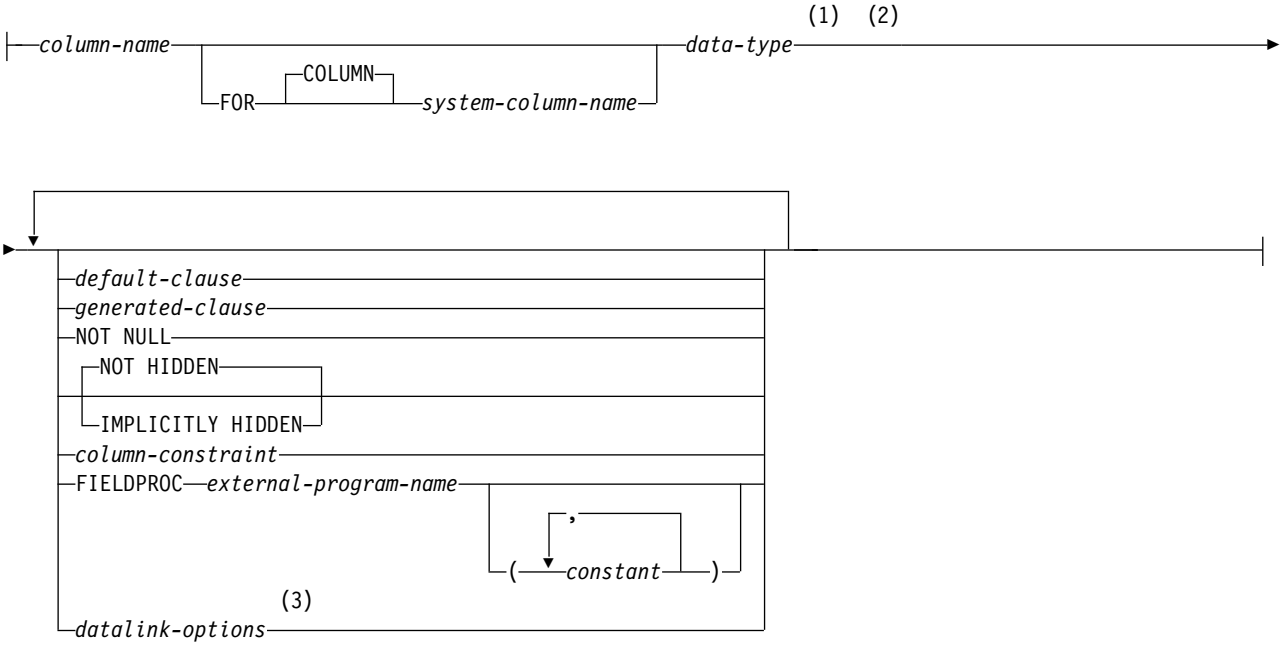
3 ACTIVATE または DEACTIVATE 節が指定される場合、ACTIVATE および DEACTIVATE 以外の節を ALTER TABLE ステートメントに指定することは許可されません。各文節を複数回指定してはいけません。

media-preference:

memory-preference:



column-definition:



data-type:

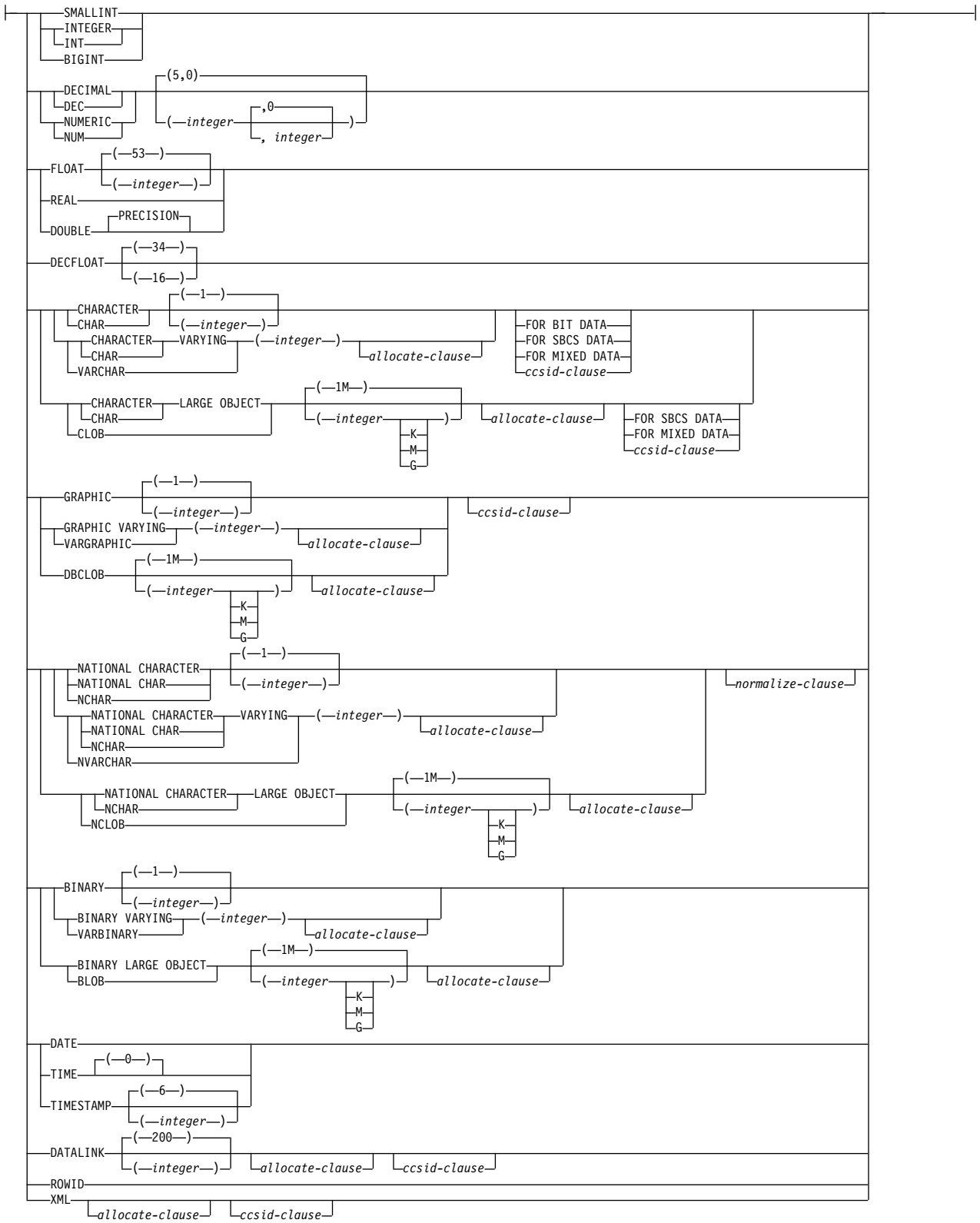
注:

1 *data-type* は、行変更タイム・スタンプ列、行開始列、行終了列、およびトランザクション開始 ID 列のためのオプションです。

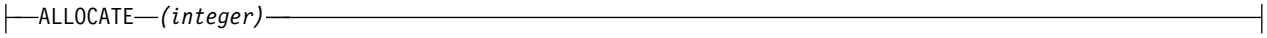
2 同じ文節を複数回指定することはできません。

3 データ・リンク・オプション は、DATALINK、および DATALINK をソースとする特殊タイプに対してのみ指定することができます。

built-in-type:


SMALLINT
INTEGER
INT
BIGINT
DECIMAL
DEC
NUMERIC
NUM
(5,0)
(integer
,0
, integer
)
FLOAT
(53)
(integer)
REAL
DOUBLE
PRECISION
DECFLOAT
(34)
(16)
CHARACTER
CHAR
(1)
(integer)
CHARACTER VARYING
CHAR VARYING
VARCHAR
allocate-clause
FOR BIT DATA
FOR SBCS DATA
FOR MIXED DATA
ccsid-clause
CHARACTER LARGE OBJECT
CHAR LARGE OBJECT
CLOB
(1M)
K
M
G
GRAPHIC
GRAPHIC VARYING
VARGRAPHIC
DBCLOB
NATIONAL CHARACTER
NATIONAL CHAR
NCHAR
NATIONAL CHARACTER VARYING
NATIONAL CHAR VARYING
NVARCHAR
NCLOB
normalize-clause
BINARY
BINARY VARYING
VARBINARY
BINARY LARGE OBJECT
BLOB
DATE
TIME
(0)
TIMESTAMP
(6)
DATALINK
(200)
ROWID
XML

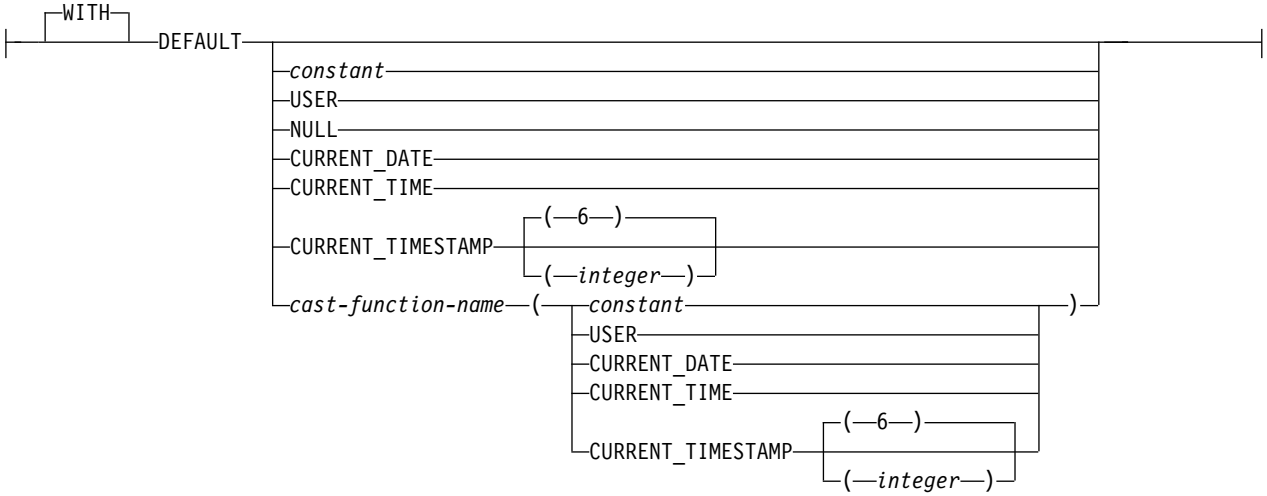
allocate-clause:



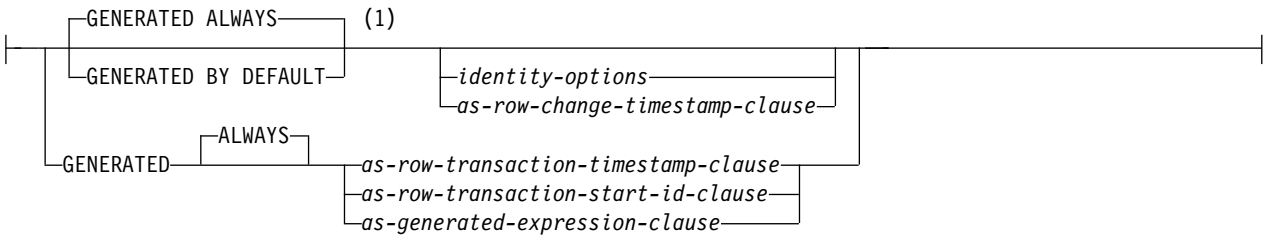
ccsid-clause:

normalize-clause:

default-clause:



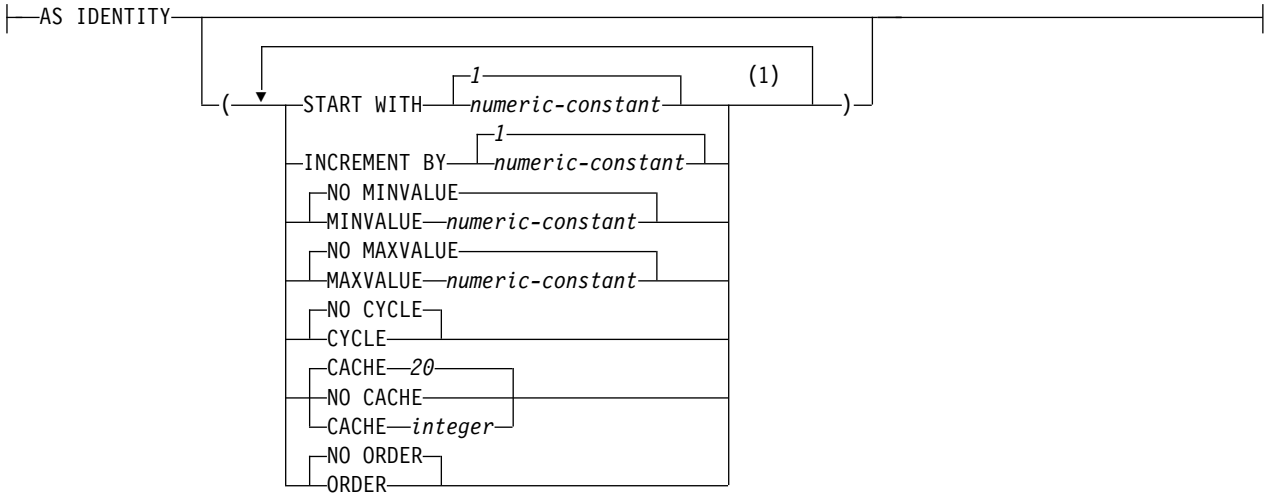
generated-clause:



注:

1 GENERATED を指定できるのは、列のデータ・タイプが ROWID (または ROWID データ・タイプに基づく特殊タイプ) であるか、列が ID 列であるか、または列が行変更タイム・スタンプである場合のみです。

identity-options:

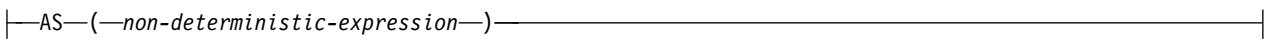


as-row-change-timestamp-clause:

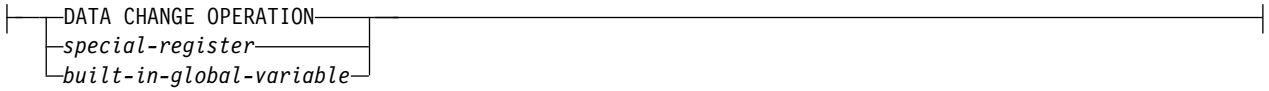
as-row-transaction-timestamp-clause:

as-row-transaction-start-id-clause:

as-generated-expression-clause:



non-deterministic-expression:



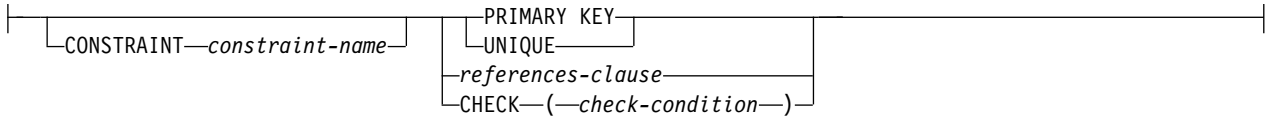
注:

1 同じ文節を複数回指定することはできません。

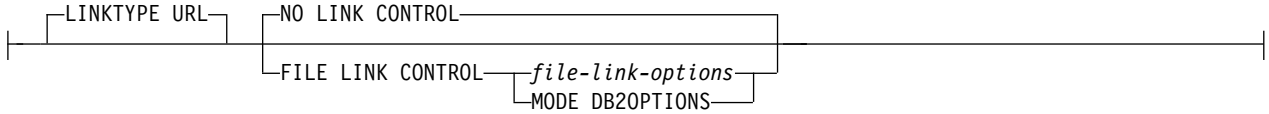
special-register:

built-in-global-variable:

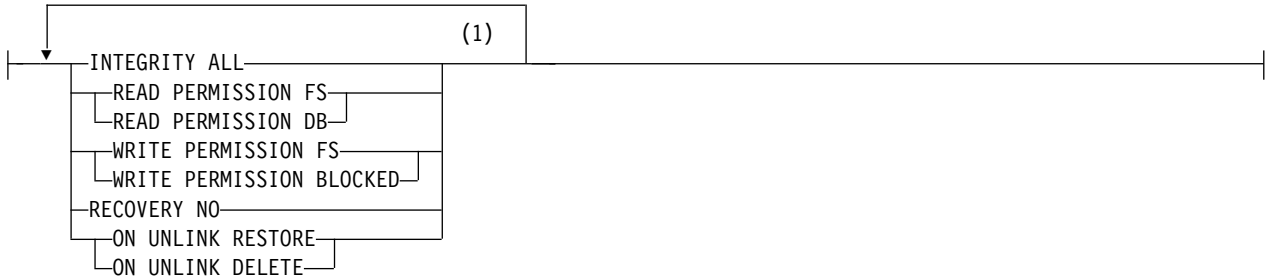
column-constraint:



datalink-options:



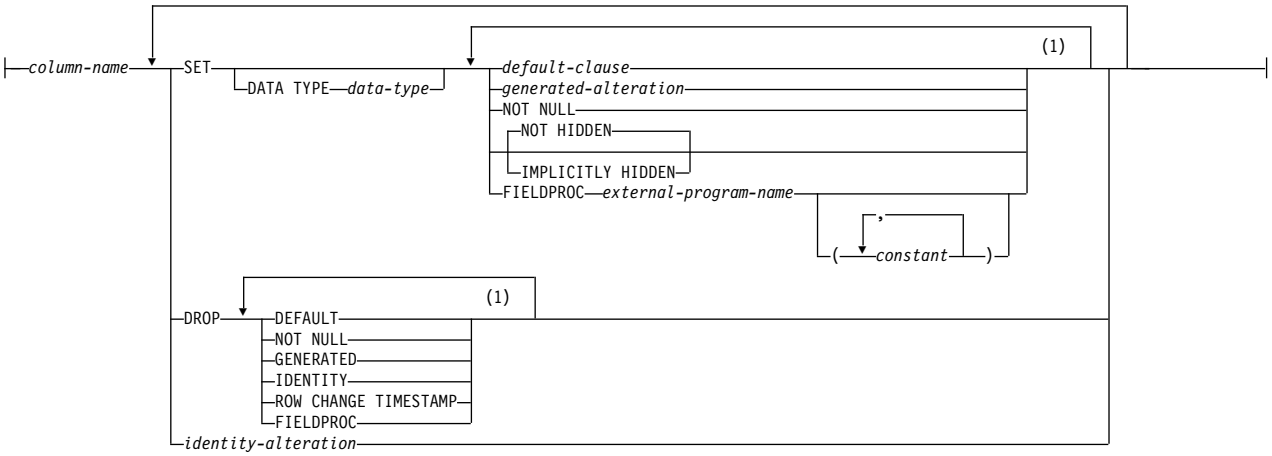
file-link-options:



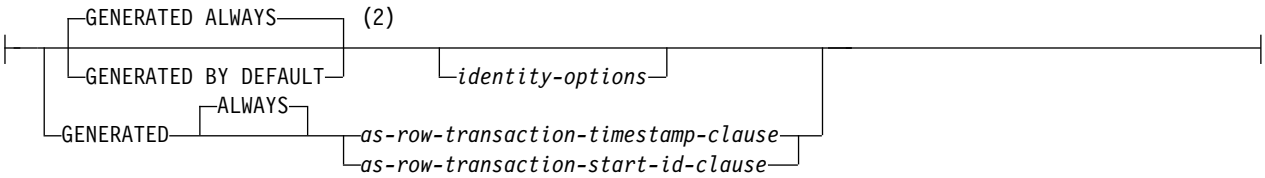
注:

1 5 つの *file-link-options* をすべて指定する必要がありますが、指定する順序は任意です。

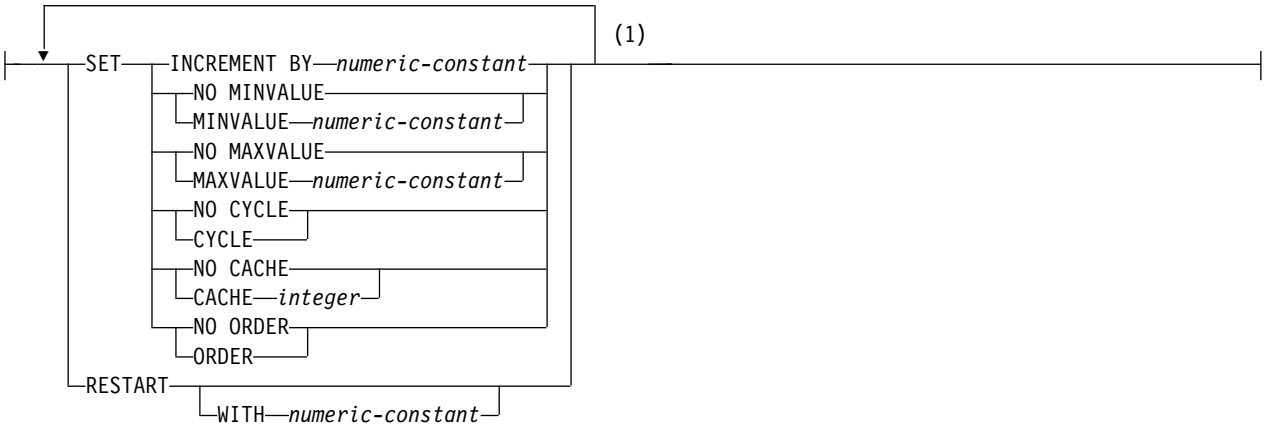
column-alteration:



generated-alteration:



identity-alteration:

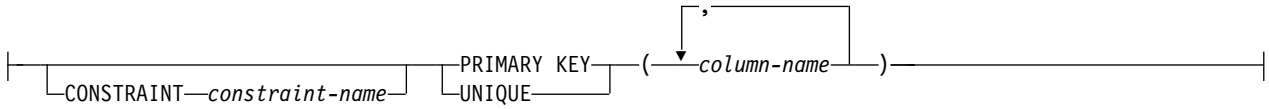


注:

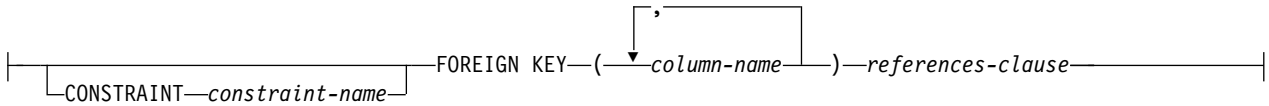
1 同じ文節を複数回指定することはできません。

2 GENERATED は次の場合にのみ指定できます。すなわち、列のデータ・タイプが ROWID (または ROWID データ・タイプに基づく特殊タイプ) である場合、列が ID 列である場合、*identity-options* が指定された場合、*as-row-transaction-timestamp-clause* が指定された場合、*as-row-transaction-start-id-clause* が指定された場合、または、列が行変更タイム・スタンプである場合です。

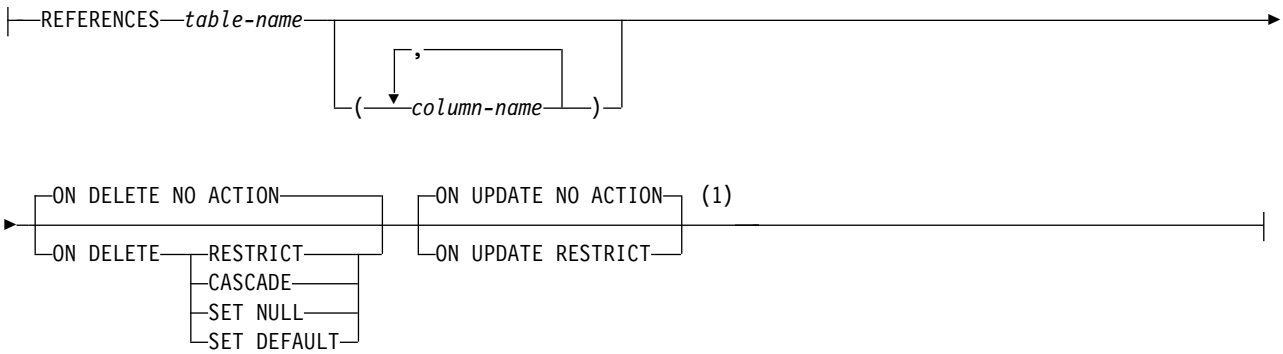
unique-constraint:



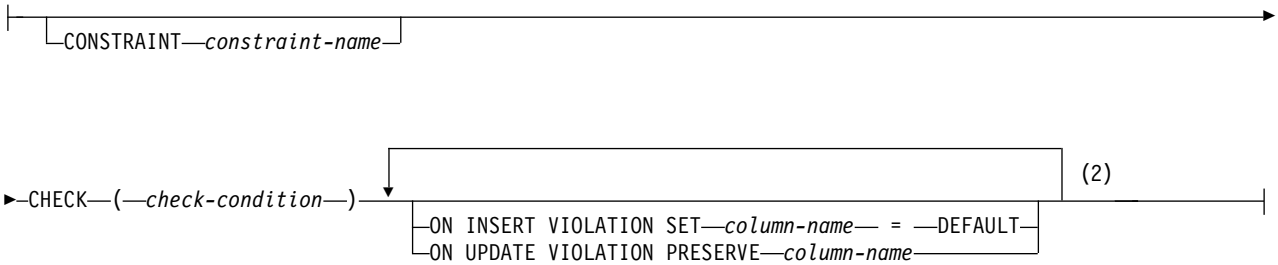
referential-constraint:



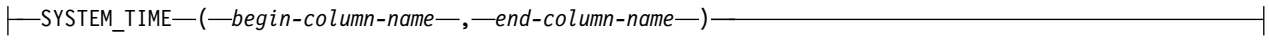
references-clause:



check-constraint:



period-definition:

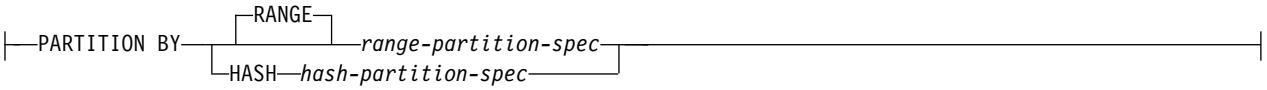


注:

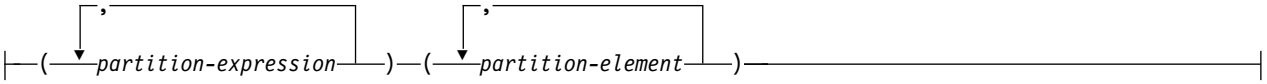
1 ON DELETE と ON UPDATE 文節は、どの順序で指定しても構いません。

2 同じ文節を複数回指定することはできません。

partitioning-clause:

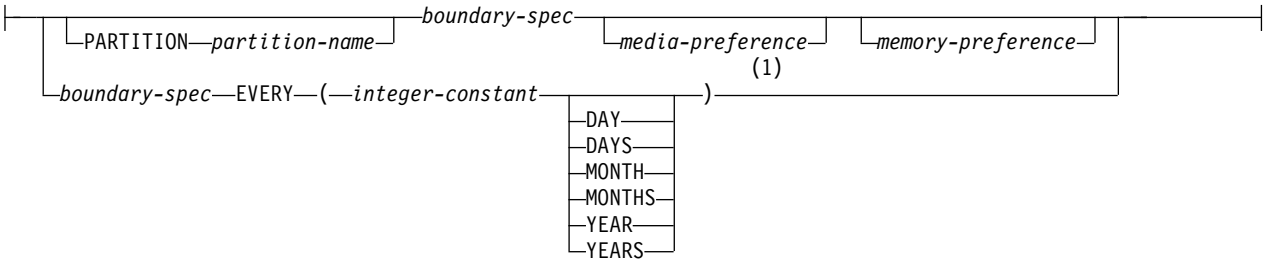


range-partition-spec:



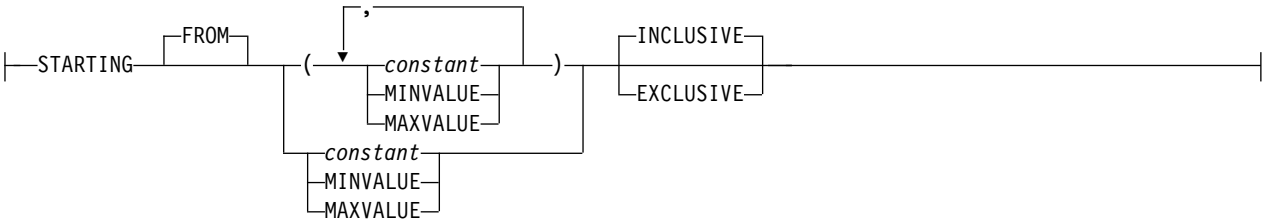
partition-expression:

partition-element:



boundary-spec:

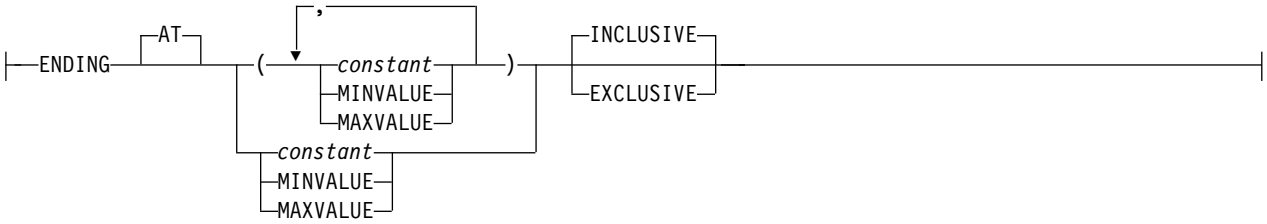
starting-clause:



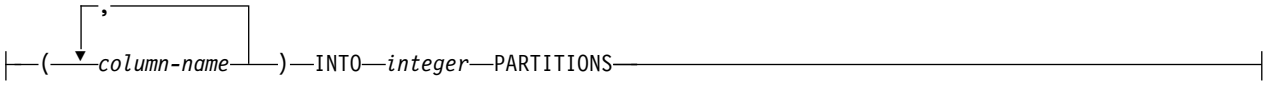
注:

1 *partition-element* のこの構文が有効なのは、数値データ・タイプまたは日時データ・タイプの *partition-expression* が 1 つだけ存在する場合です。

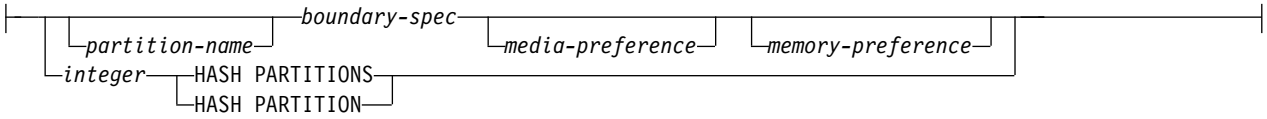
ending-clause:



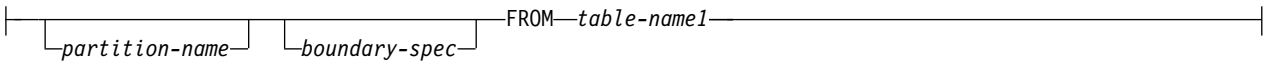
hash-partition-spec:



add-partition:

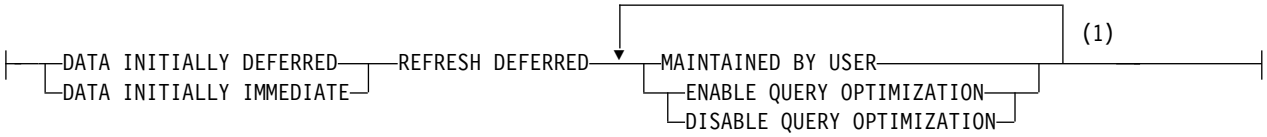


attach-partition:



materialized-query-definition:

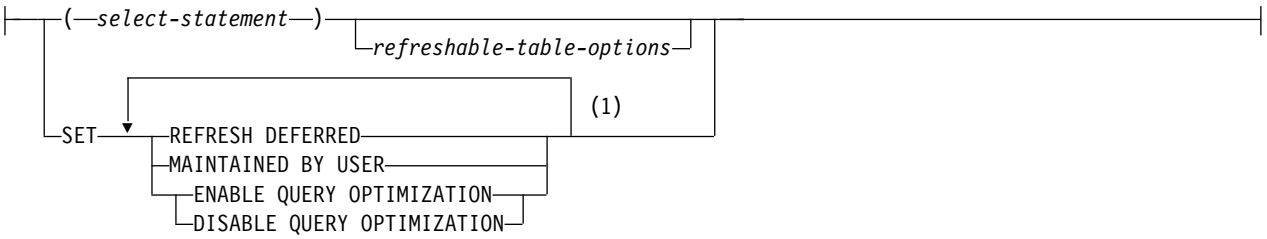
refreshable-table-options:



注:

1 同じ文節を複数回指定することはできません。MAINTAINED BY USER を指定しなければなりません。

materialized-query-table-alteration:



注:

1 同じ文節を複数回指定することはできません。

説明

table-name
変更する表を指定します。*table-name* は、 現行サーバーに存在する表を識別していなければなりません。この表は、ビュー、カタログ表、または宣言済み一時表であってはなりません。表名 がマテリアライズ照会表または履歴表を識別する場合、ADD 列定義、ALTER 列変更、または DROP COLUMN は使用できません。*table-name* がマテリアライズ照会表またはテンポラル表を識別する場合、 ATTACH PARTITION および DETACH PARTITION は使用できません。

ADD COLUMN *column-definition*

表に列を追加します。この表は履歴表にすることはできません。 表に行がある場合は、列の値はすべて、そのデフォルト値に設定されます。ただし、列が ROWID 列、または生成列である場合を除きます。ROWID 列および生成列のデフォルト値は、データベース・マネージャーが生成します。表に既に *n* 個の列がある場合、新規の列の順番は *n*+1 になります。*n*+1 の値は 8000 以下でなければなりません。

1 つの表には、1 つの ROWID と生成列の各タイプの 1 つのみを含められます。

FILE LINK CONTROL を指定した DataLink 列は、CASCADE の削除規則を伴う参照制約で従属表である表や、システム期間テンポラル表には追加できません。ROWID 列は、システム期間テンポラル表に追加できません。

新しい列を追加した結果、すべての列の行バッファー・バイト・カウントの合計が、32766 (VARCHAR 列、VARBINARY 列または VARGRAPHIC 列を指定する場合は 32740) を超えてしまう場合は、列の追加はできません。さらに、LOB 列または XML 列を指定してある場合は、挿入または更新の時点で、すべての列のバイト・カウントの合計が 3 758 096 383 を超えていてはなりません。データ・タイプごとの列のバイト・カウントについては、 1295 ページの『最大行サイズ 』を参照してください。

表がシステム期間テンポラル表である場合、関連する履歴表にも列が追加されます。履歴表に追加された列については以下の属性がコピーされます。

- 列名およびシステム列名
- データ・タイプ

- 長さ、精度、および位取り
- FOR BIT DATA、FOR SBCS DATA、FOR MIXED DATA 属性
- CCSID
- 割り振り属性
- NULL 属性
- 非表示属性
- フィールド手順

column-name
表に追加する列の名前を指定します。表の複数の列や、表のシステム列名 に同じ名前を使用してはなりません。SYSTEM_TIME という名前の列は、期間を持つ表に追加できません。列名 は修飾しません。

FOR COLUMN *system-column-name*
列の IBM i 名を指定します。表の複数の列名 またはシステム列名 に、同じ名前を使用してはなりません。

システム列名 が指定されず、また列名 が有効なシステム列名 でない場合には、システム列名が生成されます。システム列名の生成方法に関する詳細については、 1297 ページの『列名の生成の規則』を参照してください。

data-type
列のデータ・タイプを指定します。このデータ・タイプは、組み込みデータ・タイプまたは特殊タイプにすることができます。

built-in-type
組み込みデータ・タイプを指定します。組み込みタイプの説明については、1238 ページの『CREATE TABLE』を参照してください。

distinct-type-name
列のデータ・タイプが特殊タイプになるよう指定します。この列の長さ、精度、および位取りは、それぞれ、特殊タイプのソースとなっているタイプの長さ、精度、および位取りと同じになります。スキーマ名なしの特殊タイプを指定すると、その特殊タイプ名は、SQL パス上のスキーマを検索することで解決されます。

DEFAULT
列のデフォルト値を指定します。この文節は、同じ列定義 で複数回指定することはできません。次のタイプの列に対しては、システムがデフォルト値を生成するため、DEFAULT を指定できません。

- ROWID 列
- ID 列
- 行変更タイム・スタンプ列
- 行開始列
- 行終了列
- トランザクション開始 ID 列
- 生成式列

XML 列のデフォルトは、NULL です。ただし、NOT NULL を指定した場合、デフォルトはありません。

DEFAULT キーワードの後に値が指定されていないか、DEFAULT 節が指定されていない場合、次のようになります。

- 列が生成列である場合、デフォルトは生成列のタイプによって決まります。これらの値については、 958 ページの『GENERATED』 のキーワードを参照してください。
- 列が NULL 可能の場合、デフォルト値は NULL 値になります。
- 列が NULL 可能でない場合、デフォルト値は列のデータ・タイプによって決まります。

データ・タイプ	デフォルト値
数値	0
固定長文字またはグラフィック・ストリング	ブランク
固定長バイナリー・ストリング	16 進ゼロ
可変長ストリング	0 のストリング長
日付	既存の行の場合、0001 年 1 月 1 日に対応する日付。追加した行の場合は、現在の日付。
時刻	既存の行の場合は、0 時、0 分、0 秒に対応する時刻。追加した行の場合は、現在の時刻。
タイム・スタンプ	既存の行の場合は、0001 年 1 月 1 日に対応する日付、および 0 時 0 分 0 秒 0 小数秒に対応する時刻。追加した行の場合は、現在のタイム・スタンプ。
データ・リンク	DLVALUE('','URL','') に対応する値
特殊タイプ	特殊タイプの対応するソース・タイプのデフォルト値

NOT NULL および DEFAULT を列の定義 から省いた場合、DEFAULT NULL の暗黙の指定が取られます。

constant
その列のデフォルト値としての定数を指定します。これは、 113 ページの『割り当ておよび比較』で説明している割り当て規則に従って、その列に割り当てることができる値を表す定数にする必要があります。浮動小数点定数または 10 進浮動小数点定数は、SMALLINT、INTEGER、BIGINT、DECIMAL、または NUMERIC 列に使用してはなりません。10 進定数には、小数点より右方に、その列に指定された位取りより多くの桁を含めてはなりません。

USER
INSERT または UPDATE の時点での USER 特殊レジスターの値をその列のデフォルト値として指定します。 列のデータ・タイプは、USER 特殊レジスターの長さ属性と同じかそれより大きい長さ属性を持つ CHAR または VARCHAR でなければなりません。既存の行の場合、値は ALTER TABLE ステートメントの処理時の USER 特殊レジスターの値になります。

NULL
その列のデフォルト値として NULL を指定します。 NOT NULL を指定する場合は、同じ列の定義 内で DEFAULT NULL を指定してはなりません。

CURRENT_DATE
現在の日付を列のデフォルト値として指定します。 CURRENT_DATE を指定する場合は、列のデータ・タイプは DATE または DATE に基づく特殊タイプでなければなりません。

CURRENT_TIME
現在の時刻を列のデフォルト値として指定します。 CURRENT_TIME を指定する場合は、列のデータ・タイプは TIME または TIME に基づく特殊タイプでなければなりません。

CURRENT_TIMESTAMP または **CURRENT_TIMESTAMP(***integer***)**
現在のタイム・スタンプを列のデフォルト値として指定します。 CURRENT_TIMESTAMP を指定する場合は、列のデータ・タイプは TIMESTAMP または TIMESTAMP に基づく特殊タイプでなければなりません。デフォルトとして使用される CURRENT_TIMESTAMP 特殊レジスターのタイム・スタンプ精度は、この特殊レジスターに指定された精度に関係なく、常に列のタイム・スタンプ精度と一致します。

cast-function-name
この形式のデフォルト値は、特殊タイプやデータ・タイプ、BINARY、VARBINARY、BLOB、CLOB、DBCLOB、DATE、TIME、または TIMESTAMP として定義された列でのみ使用することができます。次の表は、これらのキャスト関数 の許可されている使用法を示します。

データ・タイプ	キャスト関数名
BINARY、VARBINARY、BLOB、CLOB、または DBCLOB に基づく特殊タイプ N	BINARY、VARBINARY、BLOB、CLOB、または DBCLOB *
DATE、TIME、または TIMESTAMP に基づく特殊タイプ N	N (N の作成時に生成されたユーザー定義のキャスト関数) ** あるいは DATE、TIME、または TIMESTAMP *
他のデータ・タイプに基づく特殊タイプ	N (N の作成時に生成されたユーザー定義のキャスト関数) **
BINARY、VARBINARY、BLOB、CLOB、または DBCLOB	BINARY、VARBINARY、BLOB、CLOB、または DBCLOB *
DATE、TIME、または TIMESTAMP	DATE、TIME、または TIMESTAMP *
注: * 関数には、QSYS2 の暗黙的または明示的スキーマ名のデータ・タイプ (または、特殊タイプのソース・タイプ) の名前と一致する名前を指定する必要があります。 ** 関数には、列の特殊タイプの名前と一致する名前を指定する必要があります。スキーマ名で修飾する場合は、特殊タイプのスキーマ名と同じ名前を指定する必要があります。修飾しない場合は、関数の解析から得られるスキーマ名は、特殊タイプのスキーマ名と同じ名前にする必要があります。	

constant
定数を引数として指定します。この定数は、特殊タイプのソース・タイプの定数、あるいは、特殊タイプでない場合は、データ・タイプの定数の規則に準拠する必要があります。BINARY、VARBINARY、 BLOB、

CLOB、DBCLOB、DATE、TIME、および TIMESTAMP 関数の場合は、この定数をストリング定数にする必要があります。

USER
INSERT または UPDATE の時点での USER 特殊レジスターの値をその列のデフォルト値として指定します。列の特殊タイプのソース・タイプのデータ・タイプは、USER の長さ属性と同じかそれより大きい長さ属性を持つ CHAR または VARCHAR でなければなりません。既存の行の場合、値は ALTER TABLE ステートメントの処理時の USER 特殊レジスターの値になります。

CURRENT_DATE
現在の日付を列のデフォルト値として指定します。 CURRENT_DATE を指定する場合、列の特殊タイプのソース・タイプのデータ・タイプは、DATE にする必要があります。

CURRENT_TIME
現在の時刻を列のデフォルト値として指定します。 CURRENT_TIME を指定する場合、列の特殊タイプのソース・タイプのデータ・タイプは、TIME にする必要があります。

CURRENT_TIMESTAMP または **CURRENT_TIMESTAMP(***integer***)**
現在のタイム・スタンプを列のデフォルト値として指定します。 CURRENT_TIMESTAMP を指定する場合、列の特殊タイプのソース・タイプのデータ・タイプは、TIMESTAMP にする必要があります。デフォルトとして使用される CURRENT_TIMESTAMP 特殊レジスターのタイム・スタンプ精度は、この特殊レジスターに指定された精度に関係なく、常に列のタイム・スタンプ精度と一致します。

指定した値が無効である場合、エラーが戻されます。

GENERATED
データベース・マネージャーが列の値を生成することを示します。その列が ID 列 (AS IDENTITY 文節で定義されたもの)、行変更タイム・スタンプ列、行開始列、行終了列、トランザクション開始 ID 列、または生成式列であると見なされる場合、GENERATED を指定できます。また、列のデータ・タイプが ROWID (または ROWID に基づく特殊タイプ) である場合も、GENERATED を指定できます。その他の場合は、GENERATED を指定してはなりません。ID 列、ROWID 列、または行変更タイム・スタンプ列をシステム期間テンポラル表に追加することはできません。

列が NULL 可能な場合、既存の行にある列の値として NULL 値が割り当てられます。それ以外の場合は、既存の行にある列の値は、以下のように生成列のタイプによって決まります。

- IDENTITY では、各行の識別値を生成します。
- ROW CHANGE TIMESTAMP では、ALTER TABLE ステートメントのタイム・スタンプに対応する値を使用
- ROW BEGIN では、0001 年 1 月 1 日に対応する日付と、0 時 0 分 0 秒 0 小数秒に対応する時刻を使用
- ROW END では、9999 年 12 月 30 日に対応する日付と、0 時 0 分 0 秒 0 小数秒に対応する時刻を使用

- TRANSACTION START ID では、0001 年 1 月 1 日に対応する日付と、0 時 0 分 0 秒 0 小数秒に対応する時刻を使用
- 生成式は、数値列の場合は 0、可変長文字ストリング列の場合は長さ 0 のストリング、固定長文字ストリング列の場合はブランクを使用します。

ALWAYS
行の挿入時または更新時にデフォルト値を生成しなければならない場合に、データベース・マネージャーが常に列の値を生成することを指定します。ALWAYS は推奨値です。

BY DEFAULT
行の挿入時または更新時にデフォルト値を生成しなければならない場合に、明示的な値が指定されていない限り、データベース・マネージャーが列の値を生成することを指定します。

ROWID 列の場合は、データベース・マネージャーは指定された値を使用しますが、その値は、既にデータベース・マネージャーか Db2 for i によって生成されている有効な固有の行 ID の値でなければなりません。

ID 列または行変更タイム・スタンプ列の場合は、データベース・マネージャーは指定された値を挿入または更新しますが、その ID 列または行変更タイム・スタンプ列がユニーク制約を持っているか、その ID 列または行変更タイム・スタンプ列を単独で指定するユニーク索引を持っている場合を除き、その値がその列の固有な値であるかどうかの検査は行いません。

AS IDENTITY
列が表の識別列であることを指定します。 1 つの表は識別列を 1 つだけ持つことができます。識別列は、分散表内で使用することはできません。AS IDENTITY を指定できるのは、列のデータ・タイプが、厳密に位取りがゼロの数値タイプ (SMALLINT、 INTEGER、BIGINT、DECIMAL、または位取りがゼロの NUMERIC、またはこれらのデータ・タイプに基づく特殊タイプ) である場合だけです。 DECIMAL または NUMERIC データ・タイプが指定された場合、精度は 31 以下でなければなりません。

識別列は、暗黙的に NOT NULL になります。識別属性の説明については、1238 ページの『CREATE TABLE』内の AS IDENTITY 文節を参照してください。

FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
列がタイム・スタンプであり、値はデータベース・マネージャーによって生成されることを指定します。データベース・マネージャーは、行が挿入されるたびに各行の列に値を生成し、また列が更新されるたびに各行に値を生成します。行変更タイム・スタンプ列に生成される値は、その行の挿入または更新の時刻に対応するタイム・スタンプです。 1 つの SQL ステートメントを指定して複数の行が挿入される場合、行変更タイム・スタンプ列の値は、各行が挿入された時点を反映するために行ごとに異なる可能性があります。生成された値が固有である保証はありません。

表には、行変更タイム・スタンプ列を 1 つだけ指定できます。データ・タイプを指定する場合は、精度 6 の TIMESTAMP であるか、または、精度 6 の TIMESTAMP に基づく特殊タイプでなければなりません。行変更タイム・スタンプ列の場合、DEFAULT 節を指定することはできず、NOT NULL でなければなりません。

AS ROW BEGIN

列がタイム・スタンプ・データを含み、値がデータベース・マネージャーによって生成されることを指定します。データベース・マネージャーは、行が挿入されるたびに各行の列に値を生成し、また列が更新されるたびに各行に値を生成します。生成される値は、最新のトランザクションに関連付けられている開始時刻に対応するタイム・スタンプです。単一 SQL ステートメントで複数の行が挿入される場合、トランザクション開始タイム・スタンプ列の値は各行で同じになります。

システム期間テンポラル表の場合、行開始列の値は、トランザクション全体にわたり固有になるようにデータベース・マネージャーによって生成されます。関連した履歴表に挿入される行の終了タイム・スタンプ値が開始タイム・スタンプ値より大きくなるように、タイム・スタンプ値が調整される可能性があります。これは、競合するトランザクションがシステム期間テンポラル表の同じ行を更新しているときに行われる場合があります。このタイム・スタンプ値の調整を行うには、SYSTIME_PERIOD_ADJ QAQQINI オプションを *ADJUST に設定する必要があります。単一の SQL トランザクション内で複数の行が挿入または更新され、調整が必要ではない場合、行開始列の値はすべての行において同じになり、別のトランザクションでその列のために生成された値とは異なる固有の値になります。

行開始列は、システム期間テンポラル表で使用するためのもので、SYSTEM_TIME 期間の最初の列として必要です。 1 つの表は 1 つの行開始列しか持てません。*data-type* が指定されない場合、列は TIMESTAMP(12) として定義されます。*data-type* を指定する場合は、TIMESTAMP(12) でなければなりません。この列には DEFAULT 節を指定できないため、NOT NULL として定義する必要があります。行開始列は更新できません。

既存の行の場合、列の値は、0001 年 1 月 1 日に対応する日付、および 0 時 0 分 0 秒 0 小数秒に対応する時刻に設定されます。

AS ROW END

これを指定すると、行が挿入されるとき、または行内のいずれかの列が更新されるときには常に、データベース・マネージャーによって列のデータ・タイプの値が割り当てられます。割り当てられる値は TIMESTAMP '9999-12-30-00.00.00.000000000000'です。システム期間テンポラル表では、行が削除されると、履歴行の行終了列の値に、行がいつ削除されたかが反映されます。単一の SQL ステートメントで複数の行が削除される場合、履歴行の列の値は同じになります。

行終了列は、システム期間テンポラル表で使用するためのもので、SYSTEM_TIME 期間の 2 番目の列として必要です。 表には 1 つの行終了列のみを含めることができます。*data-type* が指定されない場合、列は TIMESTAMP(12) として定義されます。*data-type* を指定する場合は、TIMESTAMP(12) でなければなりません。この列には DEFAULT 節を指定できないため、NOT NULL として定義する必要があります。行終了列は更新できません。

既存の行の場合、列の値は、9999 年 12 月 30 日に対応する日付、および 0 時 0 分 0 秒 0 小数秒に対応する時刻に設定されます。

AS TRANSACTION START ID

これを指定すると、行が表に挿入されるとき、または行のいずれかの列が更新さ

れるときには常に、データベース・マネージャーによって値が割り当てられます。データベース・マネージャーは、トランザクションごとに固有のタイム・スタンプ値、または NULL 値を割り当てます。トランザクション開始 ID 列が NULL 可能で、値を調整する必要がない行開始列が表にある場合には、その列に NULL 値が割り当てられます。それ以外の場合は、値が生成されます。単一の SQL トランザクション内で複数の行が挿入または更新される場合、トランザクション開始 ID 列の値はすべての行において同じになり、別のトランザクションでその列のために生成された値とは異なる固有の値になります。

トランザクション開始 ID 列は、システム期間テンポラル表で使用するためのもので、システム期間テンポラル表に必要です。 1 つの表は 1 つのトランザクション開始 ID 列しか持てません。*data-type* が指定されない場合、列は TIMESTAMP(12) として定義されます。*data-type* を指定する場合は、TIMESTAMP(12) でなければなりません。トランザクション開始 ID 列には DEFAULT 節を指定できません。トランザクション開始 ID 列は更新できません。

既存の行で、列が NULL 可能である場合、NULL 値が列の値として割り当てられます。それ以外の場合、列の値は、0001 年 1 月 1 日に対応する日付、および 0 時 0 分 0 秒 0 小数秒に対応する時刻のタイム・スタンプに設定されます。

DATA CHANGE OPERATION

挿入された各行、列が更新されたすべての行、および履歴表が ON DELETE ADD EXTRA ROW で定義されている場合にシステム期間テンポラル表から削除されたすべての行に、データベース・マネージャーが値を生成することを指定します。列には、以下のいずれかの値が含まれます。

I 挿入操作

U 更新操作

D 削除操作

data-type が指定されない場合、列は CHAR(1) として定義されます。*data-type* を指定する場合は、CHAR(1) でなければなりません。この列には、DEFAULT 文節もフィールド・プロシージャーも指定できません。

既存の行で、列が NULL 可能である場合、NULL 値が列の値として割り当てられます。それ以外の場合、列の値はブランクに設定されます。

special-register

挿入された各行、列が更新されたすべての行、および履歴表が ON DELETE ADD EXTRA ROW で定義されている場合にシステム期間テンポラル表から削除されたすべての行に、データベース・マネージャーによって特殊レジスターの値が割り当てられることを指定します。データ変更ステートメントの時点での特殊レジスターの値が使用されます。単一の SQL ステートメントを使用して複数の行が変更される場合、列の値は、すべての行で同じになります。

data-type は、以下の表に従って定義する必要があります。

特殊レジスター	列のデータ・タイプ
CURRENT CLIENT_ACCTNG	VARCHAR(255)
CURRENT CLIENT_APPLNAME	VARCHAR(255)

特殊レジスター	列のデータ・タイプ
CURRENT CLIENT_PROGRAMID	VARCHAR(255)
CURRENT CLIENT_USERID	VARCHAR(255)
CURRENT CLIENT_WRKSTNNAME	VARCHAR(255)
CURRENT SERVER	VARCHAR(18)
SESSION_USER	VARCHAR(128)
USER	VARCHAR(18)

この列には、DEFAULT 文節もフィールド・プロシージャーも指定できません。

既存の行で、列が NULL 可能である場合、NULL 値が列の値として割り当てられます。それ以外の場合、列の値は、長さ 0 のストリングに設定されます。

built-in-global-variable
挿入された各行、列が更新されたすべての行、および履歴表が ON DELETE ADD EXTRA ROW で定義されている場合にシステム期間テンポラル表から削除されたすべての行に、データベース・マネージャーによって組み込みグローバル変数の値が割り当てられることを指定します。データ変更ステートメントの時点での組み込みグローバル変数の値が使用されます。単一の SQL ステートメントを使用して複数の行が変更される場合、列の値は、すべての行で同じになります。

data-type は、以下の表に従って定義する必要があります。

組み込みグローバル変数	列のデータ・タイプ
QSYS2.JOB_NAME	VARCHAR(28)
QSYS2.SERVER_MODE_JOB_NAME	VARCHAR(28)
SYSIBM.CLIENT_HOST	VARCHAR(255)
SYSIBM.CLIENT_IPADDR	VARCHAR(128)
SYSIBM.CLIENT_PORT	INTEGER
SYSIBM.PACKAGE_NAME	VARCHAR(128)
SYSIBM.PACKAGE_SCHEMA	VARCHAR(128)
SYSIBM.PACKAGE_VERSION	VARCHAR(64)
SYSIBM.ROUTINE_SCHEMA	VARCHAR(128)
SYSIBM.ROUTINE_SPECIFIC_NAME	VARCHAR(128)
SYSIBM.ROUTINE_TYPE	CHAR(1)

この列には、DEFAULT 文節もフィールド・プロシージャーも指定できません。

既存の行で、列が NULL 可能である場合、NULL 値が列の値として割り当てられます。それ以外の場合、列の値は、数値列の場合は 0、可変長文字ストリング列の場合は長さ 0 のストリング、固定長文字ストリング列の場合はブランクに設定されます。

NOT NULL
列に NULL 値が入るのを防止します。NOT NULL を指定しないと、列に NULL 値が入ってもよいことを暗黙指定することになります。列定義で NOT

NULL を指定する場合は、列が ID 列でない限り、DEFAULT も指定する必要があります。行変更タイム・スタンプ列、行開始列、および行終了列には、NOT NULL が必要です。

NOT HIDDEN
列が SQL ステートメントの表の暗黙的参照に組み込まれることを示します。これはデフォルトです。

IMPLICITLY HIDDEN
名前で明示的に参照されない限り、列は SQL ステートメントから不可視であることを示します。例えば、SELECT * は、結果に隠し列を含みません。表には、少なくとも 1 つの IMPLICITLY HIDDEN でない列が含まれていなければなりません。

column-constraint
列定義 の列制約 は、単一の列から成る制約を定義するための簡便な手段です。列 C の定義に列制約の指定がある場合、その効果は、C が識別された唯一の列である固有制約、参照制約または検査制約が指定されている場合と同一です。

CONSTRAINT *constraint-name*
制約名 は、既に ALTER TABLE ステートメントで指定され、かつ既に現行サーバーに存在している制約名と同じであってはなりません。

この文節の指定がない場合、固有制約の名前がデータベース・マネージャーによって生成されます。

PRIMARY KEY
これは、1 つの列からなる基本キーを定義する簡便な手段です。列 C の定義に PRIMARY KEY を指定した場合、その効果は、別個の文節として PRIMARY KEY(C) 文節を指定したのと同一です。

この文節は、複数の列定義 で指定してはなりません。また列定義に UNIQUE 文節の指定がある場合には、この文節を指定してはなりません。この列は、LOB 列、DATALINK 列、または XML 列であってはなりません。ソート・シーケンスを指定する場合、列にフィールド・プロシージャーを含めることはできません。

基本キーを追加すると、CHECK 制約が暗黙的に追加され、その基本キーを構成する列で NULL を使用することはできないという規則が適用されます。

UNIQUE
これは、1 つの列からなるユニーク制約を定義する簡便な手段です。列 C の定義に UNIQUE の指定がある場合、その効果は、別個の文節として UNIQUE(C) 文節が指定された場合と同一です。

この文節は、1 つの列定義で複数回指定することはできません。また、列定義 で PRIMARY KEY が指定されている場合には、この文節を指定してはなりません。この列は、LOB 列、DATALINK 列、または XML 列であってはなりません。ソート・シーケンスを指定する場合、列にフィールド・プロシージャーを含めることはできません。

references-clause
列定義の *REFERENCES* 文節 は、1 つの列からなる外部キーを定義する簡便な手段です。列 C の定義に *REFERENCES* 文節 の指定がある場合、そ

の効果は、C が識別された唯一の列である FOREIGN KEY 文節の一環としてその REFERENCES 文節が指定されている場合と同一です。表が宣言済みグローバル一時表、分散表、または履歴表である場合は、参照文節 を使用することはできません。この列は、行変更タイム・スタンプ列であってはなりません。

CHECK(*check-condition***)**
これを指定すると、単一の列だけを参照するという検査条件 の検査制約を簡略式で定義することができます。したがって、列 C の列定義で CHECK を指定した場合、検査制約の検査条件 では、C 以外の列を参照することができなくなります。結果は、検査制約を別個の文節として指定した場合と同じです。

CHECK 制約の中で、FILE LINK CONTROL 列を持つ ROWID、XML、または DATALINK を参照することはできません。その他の制限事項については、 972 ページの『ADD 検査制約』を参照してください。

FIELDPROC
列のフィールド・プロシージャー出口ルーチンとして、外部プログラム名 を指定します。SQL が含まれていない ILE プログラムを指定する必要があります。サービス・プログラムを指定することはできません。

このフィールド・プロシージャーは列の値のエンコードとデコードを行います。列に値が挿入されるときは、フィールド・プロシージャーに渡されてエンコードされてから挿入されます。列に入っている値が使用されるときは、フィールド・プロシージャーに渡されてデコードされてから使用されます。

また、フィールド・プロシージャーは、ALTER TABLE ステートメントの処理中にも呼び出されます。この呼び出しの場合、プロシージャーは Db2 に列のフィールド記述を提供します。フィールド記述は、エンコードされた値のデータ特性を定義します。一方、列で指定する情報では、デコードされた値のデータ特性を定義します。

constant
フィールド・プロシージャーの呼び出し時にフィールド・プロシージャーに渡すパラメーターを指定します。パラメーター・リストはオプションです。

ROWID または DATALINK の列、あるいは ROWID または DATALINK に基づく特殊タイプの列でフィールド・プロシージャーを定義することはできません。この列は、ID 列、行変更タイム・スタンプ列、行開始列、行終了列、トランザクション開始 ID 列、および生成式列であってはなりません。この列に、CURRENT DATE、CURRENT TIME、CURRENT TIMESTAMP、USER のいずれかのデフォルト値が入っていてはなりません。チェック条件でこの列を参照することはできません。ただし、NULL 述部で参照する場合は例外です。この列が外部キーの一部になっている場合は、対応する親キー列でも同じフィールド・プロシージャーを使用する必要があります。フィールド・プロシージャーを作成する方法の詳細については、『SQL プログラミング』を参照してください。

datalink-options
DATALINK 列に関連したオプションを指定します。データ・リンク・オプション については、 1238 ページの『CREATE TABLE』を参照してください。

BEFORE *column-name*
新しい列をどの列の前に追加するのかを指定します。名前を修飾してはなりません。また、表の既存の列を識別しなければなりません。 BEFORE 文節を指定しない場合は、行の末尾に列が追加されます。

ALTER COLUMN *column-alteration*

既存の ID 列の属性など、列の定義を変更します。指定した属性だけが変更されます。それ以外のものは、未変更のままになります。表がシステム期間テンポラル表である場合は、対応する履歴表に変更が加えられます。ただし、ID や行変更タイム・スタンプなどの生成列属性はコピーされません。

column-name
変更したい列を識別します。名前を修飾してはなりません。また、表の既存の列を識別しなければなりません。 この名前で識別する列は、同じ ALTER TABLE ステートメントで追加または除去しようとしている列であってはなりません。

SET DATA TYPE *data-type*
変更したい列の新しいデータ・タイプを指定します。新しいデータ・タイプには、その列の既存のデータ・タイプとの互換性を保持させる必要があります。データ・タイプの互換性に関する詳細については、 113 ページの『割り当ておよび比較』を参照してください。以下の変更は許可されません。

- 数値データ・タイプを文字ストリング・データ・タイプへ
- 文字ストリング・データ・タイプを数値データ・タイプへ
- 日時データ・タイプを文字ストリング・データ・タイプへ
- 日時データ・タイプを異なる日時データ・タイプへ

XML 列の場合は、CCSID だけを変更できます。

指定した長さ、精度、および位取りは、既存の長さ、精度、および位取りに比較して、大きい場合も、小さい場合も、同じである場合もあります。ただし、新しい長さ、精度、または位取りの方が小さい場合は、切り捨てまたは数値変換エラーが起こる場合があります。

指定した列にデフォルト値が入れられており、新しいデフォルト値を指定しない場合は、既存のデフォルト値で、 113 ページの『割り当ておよび比較』で説明している割り当て規則に従ってその列に割り当てることができる値を表す必要があります。

行変更タイム・スタンプ列を、6 以外の精度のタイム・スタンプに変更することはできません。

行開始列、行終了列、またはトランザクション開始 ID 列を、12 以外の精度のタイム・スタンプに変更することはできません。

生成式列を必要な定義と異なるデータ・タイプおよび長さに変更することはできません。

表がシステム期間テンポラル表である場合、関連した履歴表でも列が変更されます。表がシステム期間テンポラル表である場合、ストリング長を短くする、精度を低くする、などのデータ損失が発生する可能性がある方法で列を変更することはできません。

列を固有キー、基本キー、または外部キーで指定する場合は、それらのキーの列の長さの新しい合計は 32766-*n* を超えてはなりません。ここで、*n* はヌルになることができる、指定した列の数です。

属性を変更すると、列への割り当てに関する規則に従って、列内の既存の値が新しい列属性に変換されます。ただし、ストリング値は切り捨てられるという例外を伴います。

列のデータ・タイプ属性の変更が、表に定義されている行の許可または列マスクに影響を及ぼすことがあります。列のデータ・タイプ属性が変更されると、行の許可および列マスクは、新しい列属性を使用して再評価されます。この再評価処理でエラーが検出される場合、 ALTER ステートメントは失敗します。

この列を参照する別の表に行の許可または列マスクが定義されている場合、行の許可または列マスクは、使用されるまで、または ALTER REGENERATE のオブジェクトになるまで再評価されません。再評価でエラーがあると再生成、再生成が失敗するか、または、列マスクまたは行の許可を使用する必要のある最初のステートメントが失敗するかのどちらかの結果になります。エラーを修正するために、行の許可または列マスクを除去して再作成する必要がある場合があります。

SET *default-clause*
変更したい列の新しいデフォルト値を指定します。指定するデフォルト値は、113 ページの『割り当ておよび比較』で説明している割り当て規則に従って、その列に割り当てることができる値を表す必要があります。

SET GENERATED ALWAYS または **GENERATED BY DEFAULT**
データベース・マネージャーが列の値を生成することを示します。列が ID 列 (AS IDENTITY 文節で定義されたもの)、行変更タイム・スタンプ列と見なされる場合、または列のデータ・タイプが ROWID (または ROWID に基づく特殊タイプ) である場合は、GENERATED を指定することができます。行開始列、行終了列、トランザクション開始 ID 列、および生成式列に使用できるのは、GENERATED ALWAYS のみです。それ以外の場合は、GENERATED を指定してはなりません。

AS IDENTITY
この列を表の ID 列に変更することを指定します。1 つの表は識別列を 1 つだけ持つことができます。識別列は、分散表内で使用することはできません。AS IDENTITY を指定できるのは、列のデータ・タイプが、厳密に位取りがゼロの数値タイプ (SMALLINT、 INTEGER、BIGINT、DECIMAL、または位取りがゼロの NUMERIC、またはこれらのデータ・タイプに基づく特殊タイプ) である場合だけです。 DECIMAL または NUMERIC データ・タイプが指定された場合、精度は 31 以下でなければなりません。

列は NULL 可能であってはなりません。列に明示的なデフォルト値があると、そのデフォルト値は削除されます。識別属性の説明については、 1238 ページの『CREATE TABLE』内の AS IDENTITY 文節を参照してください。

as-row-transaction-timestamp-clause
この列を表の行開始列または行終了列に変更することを指定します。この列は、TIMESTAMP(12) でなければなりません。デフォルトを持つことはできず、NULL 可能であってはなりません。

as-row-transaction-start-id-clause
この列を表のトランザクション開始 ID 列に変更することを指定します。この列は、TIMESTAMP(12) でなければならず、デフォルトを持つことはできません。

SET NOT NULL
列に NULL 値を含めることはできないことを指定します。表の既存の行にあるこの列の値は、すべて NULL 以外でなければなりません。指定した列にデフォルト値があり、新しいデフォルト値を指定しない場合は、既存のデフォルト値は NULL であってはなりません。 SET NULL の DELETE 規則を伴う参照制約の外部キーで列が識別され、その外部キーに他の NULL 可能列がない場合は、SET NOT NULL は使用できません。

SET NOT HIDDEN または **IMPLICITLY HIDDEN**
列の隠し属性を指定します。

NOT HIDDEN
列が SQL ステートメントの表の暗黙的参照に組み込まれることを示します。

IMPLICITLY HIDDEN
名前で明示的に参照されない限り、列は SQL ステートメントから不可視であることを示します。例えば、SELECT * は、結果に隠し列を含みません。表には、少なくとも 1 つの IMPLICITLY HIDDEN でない列が含まれていなければなりません。

SET FIELDPROC
列のフィールド・プロシージャー出口ルーチンとして、外部プログラム名 を指定します。SQL が含まれていない ILE プログラムを指定する必要があります。サービス・プログラムを指定することはできません。

このフィールド・プロシージャーは列の値のエンコードとデコードを行います。列に値が挿入されるときは、フィールド・プロシージャーに渡されてエンコードされてから挿入されます。列に入っている値が使用されるときは、フィールド・プロシージャーに渡されてデコードされてから使用されます。

また、フィールド・プロシージャーは、ALTER TABLE ステートメントの処理中にも呼び出されます。この呼び出しの場合、プロシージャーは Db2 に列のフィールド記述を提供します。フィールド記述は、エンコードされた値のデータ特性を定義します。一方、列で指定する情報では、デコードされた値のデータ特性を定義します。

constant
フィールド・プロシージャーの呼び出し時にフィールド・プロシージャーに渡すパラメーターを指定します。パラメーター・リストはオプションです。

ROWID または DATALINK の列、あるいは ROWID または DATALINK に基づく特殊タイプの列でフィールド・プロシージャーを定義することはできません。この列は、ID 列、行変更タイム・スタンプ列、行開始列、行終了列、トランザクション開始 ID 列、および生成式列であってはなりません。この列に、CURRENT DATE、CURRENT TIME、CURRENT TIMESTAMP、USER のいずれかのデフォルト値が入っていてはなりません。フィールドのエンコード形式とデコード形式では、NULL 可能性属性が一致している必要があります。チェック条件でこの列を参照することはできません。ただし、NULL 述部で参照す

る場合は例外です。この列が外部キーの一部になっている場合は、対応する親キー列でも同じフィールド・プロシージャーを使用する必要があります。フィールド・プロシージャーの例については、『SQL プログラミング』トピック集を参照してください。

DROP DEFAULT
列の現行デフォルト値を除去します。指定される列は、次のとおりです。

- デフォルト値を持っていなければならず、ヌル属性が NOT NULL であってはなりません。あるいは、
- 列のデフォルトが DEFAULT NULL で定義された場合は NULL のデフォルト値を持つことができます。

この新規のデフォルト値は NULL 値になります。

DROP NOT NULL
列の NOT NULL 属性を除去し、その列が NULL 値をもてるようにします。デフォルト値の指定がない場合、またはデフォルト値がまだ存在していない場合は、新しいデフォルト値は NULL 値になります。表の基本キーで列を指定する場合、もしくはその列が ID 列、行変更タイム・スタンプ列、行開始列、行終了列、または ROWID である場合は、DROP NOT NULL は使用できません。

DROP GENERATED
列の生成された属性をドロップします。この列は、ID 列、行変更タイム・スタンプ列、行開始列、行終了列、トランザクション開始 ID 列、または生成式列でなければなりません。バージョン管理がアクティブな場合、行開始列、行終了列、およびトランザクション開始 ID 列の生成属性を除去することはできません。

DROP IDENTITY
列の識別属性を除去して、列を単純な数値データ・タイプ列にします。列が ID 列でない場合は、DROP IDENTITY は使用できません。

DROP ROW CHANGE TIMESTAMP
列の行変更タイム・スタンプ属性をドロップし、その列を単純なタイム・スタンプ列にします。列が行変更タイム・スタンプ列でない場合は、DROP ROW CHANGE TIMESTAMP は使用できません。

DROP FIELDPROC
列のフィールド・プロシージャーを除去します。列でフィールド・プロシージャーが定義されていなければ、DROP FIELDPROC を使用できません。

identity-alteration
列の識別属性を変更します。列は指定した表に存在しなければならず、また、既に IDENTITY 属性を使用して定義されていなければなりません。属性の説明については、AS IDENTITY を参照してください。

RESTART
ID 列に入れる次の値を指定します。 WITH 数値定数 を指定していない場合は、この ID 列を最初に作成したときに暗黙的または明示的に指定されている開始値から、シーケンスが再開始されます。RESTART を指定しても、元の START WITH 値は変更されません。

WITH *numeric-constant*
この列に入れる次の値として数値定数 を使用することを指定します。

numeric-constant は、この列に割り当てることのできる正または負の値で、小数点の右側にゼロ以外の数字がない厳密な数値定数でなければなりません。

DROP COLUMN

識別された列を表から除去します。

column-name
除去したい列を識別します。列名は修飾してはなりません。この名前は、指定した表の列を識別するものでなければなりません。以下のものを示す名前であってはなりません。

- この ALTER TABLE ステートメントで既に追加または変更された列
- 表の唯一の列
- 表の列の中で非表示であるものを除いた最後の列
- パーティション表または分散表のパーティション・キー
- システム期間テンポラル表の列
- 履歴表の列
- 期間の定義で参照される列

列が除去されると、その列に対して定義された列マスクはすべて除去されます。列が、別の列の行の許可または列マスクの定義で参照されている場合、その列を除去することはできません。

CASCADE
除去される列に従属しているビュー、索引、トリガー、または制約もすべて除去されることを指定します。[80]

RESTRICT
ビュー、索引、トリガー、マテリアライズ照会表、または制約が列に依存している場合、その列を除去できないことを指定します。 [80]

ある制約で参照されている列がすべて同一の ALTER TABLE ステートメントで除去される場合は、その除去が RESTRICT で妨げられることはありません。

ADD ユニーク制約

CONSTRAINT *constraint-name*
制約の名前を指定します。制約名 は、既に ALTER TABLE ステートメントで指定され、かつ既に現行サーバーに存在している制約名と同じであってはなりません。制約名は、スキーマ内で固有でなければなりません。

指定しない場合は、固有の制約名がデータベース・マネージャーによって生成されます。

UNIQUE (*column-name***,...)**
識別された列で構成されるユニーク制約を定義します。それぞれの *column-name* は、該当の表の列を識別する非修飾の名前でなければなりません。同じ列を複数回指定することはできません。この列は、LOB 列、DATALINK 列、または XML 列であってはなりません。ソート・シーケンス

80. トリガーは、UPDATE OF 列リストまたはトリガー・アクション内の任意の場所で参照されている場合、列に従属します。

を指定する場合、列にフィールド・プロシージャーを含めることはできません。指定できる列の数は 120 を超えてはならず、その長さの合計は 32766-*n* を超えてはなりません。ここで、*n* は NULL が許されると指定された列の数です。表が履歴表である場合は、UNIQUE を使用できません。

指定する列のセットは、その表の他の UNIQUE 制約または PRIMARY KEY に指定されている列のセットと同じであってはなりません。例えば、UNIQUE(A,B) は、UNIQUE(B,A) あるいは PRIMARY KEY(A,B) が該当の表に既に存在する場合には許されません。一連の列に指定されている既存のどの非 NULL 値も固有の値にする必要があります。複数の NULL 値が許されます。

識別された列に固有索引が既に存在する場合は、その索引が固有制約索引として指定されます。それ以外の場合は、固有キーの固有性をサポートするために固有索引が作成されます。固有索引は、別個のシステム論理ファイルとしてではなく、システム物理ファイルの一部として作成されます。

PRIMARY KEY (*column-name***,...)**

指定した列で構成される基本キーを定義します。それぞれの *column-name* は、該当の表の列を識別する非修飾の名前でなければなりません。同じ列を複数回指定することはできません。この列は、LOB 列、DATALINK 列、または XML 列であってはなりません。ソート・シーケンスを指定する場合、列にフィールド・プロシージャーを含めることはできません。指定する列の数は 120 を超えてはならず、それらの列の長さの合計は 32766 を超えてはなりません。該当の表に既に基本キーが存在していてはなりません。表が履歴表である場合は、PRIMARY KEY を使用できません。

指定する列は、その表の他の UNIQUE 制約に指定されている列と同じであってはなりません。例えば、PRIMARY KEY(A,B) は、その表に UNIQUE(B,A) が既に存在する場合には許されません。指定した一連の列の既存の値は、固有でなければなりません。

基本キーを追加すると、CHECK 制約が暗黙的に追加され、その基本キーを構成するどの列でも NULL を使用することはできないという規則が適用されます。

指定した列に固有索引が既に存在している場合には、その索引は、基本索引として扱われます。このような場合以外は、基本キーの固有性をサポートする基本索引が作成されます。固有索引は、別個のシステム論理ファイルとしてではなく、システム物理ファイルの一部として作成されます。

ADD referential-constraint

CONSTRAINT *constraint-name*

制約の名前を指定します。制約名 は、現行サーバーに既に存在している制約を識別するものであってはなりません。

指定しない場合は、固有の制約名がデータベース・マネージャーによって生成されます。

FOREIGN KEY

参照制約を定義します。表が宣言済みグローバル一時表、分散表、または履歴表である場合は、FOREIGN KEY を使用することはできません。

以下の説明で T1 は、変更したい表を表しています。

(*column-name***,...)**
参照制約の外部キーは、指定した列で構成されます。各 *column-name* は、T1 の列を指定する非修飾名でなければなりません。同じ列を複数回指定することはできません。この列は、LOB 列、DATALINK 列、または XML 列であってはならず、また行変更タイム・スタンプ列であってもなりません。ソート・シーケンスを指定する場合、列にフィールド・プロシージャーを含めることはできません。指定できる列の数は 120 を超えてはならず、その長さの合計は 32766-*n* を超えてはなりません。ここで、*n* は NULL が許可されるように指定された列の数です。

REFERENCES *table-name*
REFERENCES 文節で指定する表名 は、現行サーバー上に存在する基本表を示すものでなければなりませんが、カタログ表、宣言済み一時表、分散表、または履歴表を示すものであってはなりません。親がパーティション表の場合は、親のユニーク制約を適用するユニーク索引がパーティション化されていてはなりません。この表は、制約関係における親表と呼ばれます。

参照制約の外部キー、親キー、および親表が表にある既存の参照制約の外部キー、親キー、および親表と同じである場合は、その参照制約は重複 します。 重複する参照制約は許されますが、お勧めできません。

以下の説明で、T2 は親表を示しています。

(*column-name***,...)**
参照制約の親キーは、ここで指定する列によって構成されます。各 *column-name* は、T2 の列を指定する非修飾名でなければなりません。同じ列を複数回指定することはできません。この列は、LOB 列、DATALINK 列、または XML 列であってはならず、また行変更タイム・スタンプ列であってもなりません。ソート・シーケンスを指定する場合、列にフィールド・プロシージャーを含めることはできません。指定できる列の数は 120 を超えてはならず、その長さの合計は 32766-*n* を超えてはなりません。ここで、*n* は NULL が許されると指定された列の数です。

この列名のリストは、T2 の基本キーまたは T2 に存在する UNIQUE 制約の列名のリストと同一でなければなりません。名前はどのような順序で指定しても構いません。例えば、(A,B) を指定すると、UNIQUE(B,A) として定義された固有制約はこの要件を満たします。列名のリストの指定がない場合、T2 は基本キーを持たなければなりません。列名のリストの省略は、基本キーの列の暗黙の指定を意味しています。

指定した外部キーは、T2 の親キーと同じ数の列を持たなければなりません。外部キーの *n* 番目の列と親キーの *n* 番目の列の記述は、同一のデータ・タイプ、長さ、CCSID、および FIELDPROC を持つ必要があります。

表が空である場合を除き、表の使用に先立って、外部キーの値の妥当性を検査する必要があります。外部キーの値は、ALTER TABLE ステートメントの実行中に妥当性を検査されます。したがって、外部キーの NULL 以外の値はすべて、T2 の親キーの値と一致している必要があります。

FOREIGN KEY 文節によって指定する参照制約は、T2 が親で、T1 が従属の関係を定義します。

ON DELETE
親表の行が削除される時点で、従属表について行うアクションを指定します。可能なアクションには以下の 5 つがあります。

- NO ACTION (デフォルト)
- RESTRICT
- CASCADE
- SET NULL
- SET DEFAULT

外部キーの列に NULL が許される列がある場合を除いて、SET NULL を指定してはなりません。 T1 が更新トリガーを持つ場合には、SET NULL および SET DEFAULT を指定してはなりません。

T1 が削除トリガーを持つ場合には、CASCADE を指定してはなりません。

FILE LINK CONTROL を指定した DATALINK 列が T1 に含まれる場合には、CASCADE を指定してはなりません。

削除規則は、T2 の行が DELETE または波及削除操作の対象で、しかもその行が T1 に従属する行を持っている場合に適用されます。 *p* は、そのような T2 の行を表すと想定します。

- RESTRICT または NO ACTION を指定した場合、エラーが生じ、行の削除は行われません。
- CASCADE を指定すると、 T1 の *p* の従属行に削除操作が伝搬します。
- SET NULL が指定された場合、T1 の *p* のそれぞれの従属行の外部キーの NULL 可能な列が NULL 値に設定されます。
- SET DEFAULT を指定した場合、T1 の *p* の各従属行の外部キーの各列は、そのデフォルト値に設定されます。

ON UPDATE
親表の行が更新される時点で、従属表で行うアクションを指定します。

更新規則は、T2 の行が UPDATE または波及更新操作の対象で、しかもその行が T1 に従属行を持つ場合に適用されます。 *p* は、そのような T2 の行を表すと想定します。

- RESTRICT または NO ACTION を指定した場合、エラーが生じ、行の更新は行われません。

ADD 検査制約

CONSTRAINT *constraint-name*
制約の名前を指定します。制約名 は、現行サーバーに既に存在している制約を識別するものであってはなりません。制約名は、スキーマ内で固有でなければなりません。

指定しない場合は、固有の制約名がデータベース・マネージャーによって生成されます。

CHECK (*check-condition***)**
検査制約を定義します。表のどの行についても、検査条件 は真または不明のいずれかでなければなりません。

検査条件 は、検索条件 です。ただし、下記の条件は除きます。

- 参照できるのは表の列だけであり、列名を修飾してはなりません。
- *check-condition* で指定する式の結果を、ROWID、XML の各データ・タイプ、または FILE LINK CONTROL を伴う DATALINK データ・タイプにすることはできません。
- 次のいずれも含めることはできません。
  - 副照会
  - 集約関数
  - 変数
  - グローバル変数
  - パラメーター・マーカー
  - シーケンス参照
  - LOB を含む複合式 (連結など)
  - OLAP の指定
  - ROW CHANGE 式
  - IS JSON、JSON_EXISTS、または REGEXP_LIKE の述部
  - 特殊レジスター
  - deterministic でない関数
  - 特殊タイプの作成に伴って暗黙に生成された関数以外のユーザー定義関数
  - 以下の組み込みスカラー関数

ATAN2	DLURLSCHEME	JSON_TO_BSON	ROUND_TIMESTAMP
BSON_TO_JSON	DLURLSERVER	JSON_VALUE	RPAD
CARDINALITY	DLVALUE	LOCATE_IN_STRING	SCORE
CONTAINS	ENCRYPT_AES	LPAD	SOUNDEX
CURDATE	ENCRYPT_RC2	MAX_CARDINALITY	TABLE_NAME
CURTIME	ENCRYPT_TDES	MONTHNAME	TABLE_SCHEMA
DATAPARTITIONNAME	GENERATE_UNIQUE	MONTHS_BETWEEN	TIMESTAMP_FORMAT
DATAPARTITIONNUM	GETHINT	NEXT_DAY	TIMESTAMPDIFF
DAYNAME	HASH	NOW	TRUNC_TIMESTAMP
DBPARTITIONNAME	HASH_MD5	OVERLAY	VARCHAR_FORMAT
DECRYPT_BINARY	HASH_SHA1	RAISE_ERROR	VERIFY_GROUP_FOR_USER
DECRYPT_BIT	HASH_SHA256	RAND	WEEK_ISO
DECRYPT_CHAR	HASH_SHA512	REGEXP_COUNT	WRAP
DECRYPT_DB	IDENTITY_VAL_LOCAL	REGEXP_INSTR	XMLPARSE
DIFFERENCE	INSERT	REGEXP_REPLACE	XMLVALIDATE
DLURLCOMPLETE [1]	JSON_ARRAY	REGEXP_SUBSTR	XSLTRANSFORM
DLURLPATH	JSON_OBJECT	REPEAT	
DLURLPATHONLY	JSON_QUERY	REPLACE	

[1] FILE LINK CONTROL と READ PERMISSION DB の属性が指定されたデータ・リンクの場合。

ON INSERT VIOLATION

挿入される行の *check-condition* が false の場合の処置を指定します。この節が指定されていないと、挿入の *check-condition* が false の場合はエラーが発生します。

SET *column-name* **= DEFAULT**

挿入操作で提供される値の代わりに、*column-name* のデフォルト値が表に挿入されます。

column-name は *check-condition* 内で参照されている必要があります。

ON UPDATE VIOLATION
更新される行に対する *check-condition* が false の場合の処置を指定します。この節が指定されていないと、更新の *check-condition* が false の場合はエラーが発生します。

PRESERVE *column-name*
column-name の現行値は、更新操作によって提供される値で置き換えられるのではなく、表に残ります。

column-name は *check-condition* 内で参照されている必要があります。

検索条件の詳細については、 275 ページの『検索条件』を参照してください。

DROP

PRIMARY KEY
基本キーの定義、およびその基本キーが親キーである参照制約すべてを除去します。該当の表は、基本キーを持っていなければなりません。

UNIQUE *constraint-name*
制約名 で指定した固有制約、およびこの固有制約に依存する参照制約すべてを除去します。制約名 は、該当の表の固有制約を識別していなければなりません。DROP UNIQUE は、PRIMARY KEY 固有制約を除去することはありません。

FOREIGN KEY *constraint-name*
制約名が *constraint-name* の参照制約をドロップします。制約名 は、該当の表が従属している参照制約を識別していなければなりません。

CHECK *constraint-name*
検査制約 *constraint-name* をドロップします。制約名 では、表の検査制約を識別する必要があります。

CONSTRAINT *constraint-name*
制約 *constraint-name* をドロップします。制約名 では、表の固有、参照、または検査制約を識別する必要があります。この制約が、PRIMARY KEY または UNIQUE の制約である場合、その基本キーまたは固有キーが親キーである参照制約もすべて除去されます。

CASCADE
固有制約に関して、除去される制約に従属している参照制約があれば、それもすべて除去されることを指定します。

RESTRICT
固有制約に関して、それに従属している参照制約がある場合は、その固有制約は除去できないことを指定します。

ADD パーティション化文節

パーティション化されていない表をパーティション化された表に変更します。指定した表が分散表または既にパーティション化された表である場合は、エラーが戻されます。 DDS で作成された物理ファイルをパーティション化することはできません。パーティション化文節 の説明については、 1238 ページの『CREATE TABLE』を参照してください。

データを含むパーティション化されていない表をパーティション化された表に変更する場合、データ・パーティション間のデータ移動が必要になります。範囲パーティションを使用する場合、表内のすべての既存のデータは、指定された範囲パーティションに割り当て可能でなければなりません。

DROP PARTITIONING

パーティション化された表をパーティション化されていない表に変更します。指定した表が既にパーティション化されていない表である場合は、エラーが戻されます。

データを含むパーティション化された表をパーティション化されていない表に変更する場合、データ・パーティション間のデータ移動が必要になります。

ADD PARTITION *add-partition*

1 つ以上のパーティションをパーティション化された表に追加します。指定する表は、パーティション表でなければなりません。パーティションの数は、256 以下でなければなりません。

データを含むパーティション化された表のハッシュ・パーティションの数を変更する場合、データ・パーティション間のデータ移動が必要になります。

partition-name
パーティションに名前を付けます。パーティション名 で、表内に既に存在するデータ・パーティションを識別してはなりません。

この文節の指定がない場合、固有のパーティション名がデータベース・マネージャーによって生成されます。

boundary-spec
範囲パーティションの境界を指定します。指定した表が範囲パーティション化された表でない場合、エラーが戻されます。開始文節 および終了文節 の両方を必ず指定する必要があります。境界スペック の説明については、 1238 ページの『CREATE TABLE』を参照してください。

整数 HASH PARTITIONS
追加するハッシュ・パーティションの数を指定します。指定した表がハッシュ・パーティション化された表でない場合は、エラーが戻されます。

ALTER PARTITION

範囲パーティション化された表のパーティションの境界を変更します。指定した表が範囲パーティション化された表でない場合、エラーが戻されます。

データを含む表の複数のパーティションの境界を変更する場合、データ・パーティション間でデータを移動する必要があります。表内のすべての既存のデータは、指定された範囲パーティションに割り当て可能でなければなりません。

partition-name
変更するパーティションの名前を指定します。パーティション名 で、表内に存在するデータ・パーティションを識別しなければなりません。

boundary-spec
範囲パーティションの新規の境界を指定します。開始文節 および終了文節 の両方を必ず指定する必要があります。境界スペック の説明については、 1238 ページの『CREATE TABLE』を参照してください。

DROP PARTITION

パーティション化された表のパーティションを除去します。指定した表がパーティション化された表でない場合、エラーが戻されます。パーティション化された表の最後に残ったパーティションを指定すると、エラーが戻されます。 PRESERVE ROWS オプションが指定されている場合とパーティションが空である場合を除き、システム期間テンポラル表からパーティションをドロップすることはできません。

partition-name
除去するパーティションの名前を指定します。パーティション名 で、表内に存在するデータ・パーティションを識別しなければなりません。

DELETE ROWS
指定したパーティションのすべてのデータを廃棄することを指定します。パーティションに保管されているすべてのデータは、削除トリガーを処理せずに表から削除されます。

PRESERVE ROWS
指定したパーティションのすべてのデータを残りのパーティションに移動することによって、削除または挿入トリガーを処理せずにそのデータを保存することを指定します。指定した表が範囲パーティション化された表である場合、PRESERVE ROWS を指定してはなりません。ハッシュ・パーティションを除去すると、残りのデータ・パーティション間でデータの移動が必要になります。

ATTACH PARTITION

他の表を新規のデータ・パーティションとしてアタッチします。アタッチされる表のデータは、アタッチ先の表の新規パーティションになります。データ移動は一切ありません。指定したターゲット表がパーティション化された表でない場合、エラーが戻されます。ターゲット表に既に最大数のパーティションがある場合、エラーが戻されます。表がハッシュ・パーティション化されている場合、エラーが戻されます。

アタッチされる表で、行レベルのアクセス制御または列レベルのアクセス制御のいずれかがアクティブになっている場合、アタッチ先の表の同じ制御をアクティブにする必要があります。行権限や列マスクは、アタッチされる表からターゲット表に自動的に引き継がれるわけではありません。列マスクおよび行権限は、必ずしも両方の表で厳密に同じである必要はありません。ただし、セキュリティーの観点では、同じであるのが最善です。しかし、アタッチされる表の行レベルのアクセス制御がアクティブになっている場合には、アタッチ先の表でも行レベルのアクセス制御をアクティブにする必要があります。同様に、アタッチする表の列レベルのアクセス制御がアクティブになっており、少なくとも 1 つの列マスクが使用可能になっている場合には、アタッチ先の表でも列レベルのアクセス制御をアクティブにし、対応する列の列マスクを使用可能にする必要があります。

partition-name
データ・パーティションの名前を指定します。表の他のデータ・パーティション

と同じ名前を指定することはできません。この文節が指定されず、ソース表がパーティション化されている場合は、ソース表のパーティション名が使用されます。その他の場合は、この文節が指定されていないと、固有のパーティション名がデータベース・マネージャーによって生成されます。

boundary-spec
新規パーティションの境界を指定します。 *boundary-spec* が指定されていない場合は、次のようになります。

- ソース表は、単一パーティションでパーティション化された表でなければなりません。そのパーティションの *boundary-spec* が暗黙的に使用されます。
- ソース表のパーティション・キーの数とパーティション・キー名は、ターゲット表のものと同じでなければなりません。

範囲は、暗黙的であれ明示的であれ、既存のデータ・パーティションの範囲と重なり合ってはなりません。境界スペック の説明については、 1238 ページの『CREATE TABLE』を参照してください。 *boundary-spec* が指定された場合、ソース表のデータは、指定された範囲に適合する必要があります。そうでないと、後続の SQL 操作が予測不能になる可能性があります。

FROM *table-name1*
新規パーティションのデータのソースとして使用する表を指定します。表は、ターゲット表と同じリレーショナル・データベースに存在しなければなりません。指定した表がパーティション化された表でない場合、エラーが戻されます。表がハッシュ・パーティション化されている場合、エラーが戻されます。表は、ビュー、マテリアライズ照会表、宣言済みグローバル一時表、システム期間テンポラル表、履歴表、およびシステム表であってはなりません。表に NOT LOGGED INITIALLY 属性があってはなりません。*table-name1* の表定義は、複数のパーティションを持つことができず、以下の形で、変更される表と一致していなくてはなりません。

- 列数が同じ。
- 表内の同じ位置 (順に並べた時の同じ位置) にある列のデータ・タイプ、属性、フィールド・プロシージャー、生成式、デフォルト値、および CCSID が同じ。
- 表内の同じ位置 (順に並べた時の同じ位置) にある列の NULL 可能特性が同じ。
- ターゲット表に行変更タイム・スタンプ列がある場合、それに対応するソース表の列は、行変更タイム・スタンプ列でなければなりません。
- 可変長または LOB データ・タイプの場合、表内の同じ位置 (順に並べた時の同じ位置) にある列の ALLOCATE 長が同じ。

table-name1 のデータが正しくアタッチされた後は、DROP TABLE *table-name1* CASCADE と同等の操作が実行され、この表はデータを持たなくなり、データベースから削除されます。

DETACH PARTITION

パーティション化された表のパーティションをデタッチします。指定した表がパーティション化された表でない場合、エラーが戻されます。 1 つのパーティションを含む新しいパーティション化された表 *table-name1* が作成され、デタッチされたパーティション・データが入ります。 *partition-name* を変更される表の最後に残った

データ・パーティションにすることはできません。変更する表は、システム期間テンポラル表や、強制される参照制約の親表であってはなりません。また、変更する表がハッシュ・パーティションを使用してはなりません。

ソース表が ID 列を含む場合、作成された表で対応する列は ID 列になりません。ソース表のその他の列属性はすべて、作成された表の対応する列に定義されます。ソース表に制約がある場合、作成された表に同様の制約は追加されません。

行レベルのアクセス制御または列レベルのアクセス制御のいずれかがアクティブである表からパーティションがデタッチされる場合、デタッチされたデータ用に作成される新しい表では、デタッチされたデータを保護するために、自動的に行レベルのアクセス制御がアクティブになります。この新しい表に対して適切な行権限を定義するか、表の行レベルのアクセス制御を非アクティブにするまで、この表に直接アクセスしても行は返されません。

ソース表に対する特権は、作成された表に伝搬されません。

partition-name
デタッチするパーティションの名前を指定します。 *partition-name* は、表内に存在するデータ・パーティションを識別しなければなりません。

table-name1
表の名前を指定します。暗黙的または明示的修飾子も含め、この名前で、現行サーバーに既に存在している別名、ファイル、索引、表、またはビューを識別することはできません。

SQL 名が指定されている場合、表は、暗黙的または明示的修飾子で指定しているスキーマ内に作成されます。

システム名が指定されている場合、表名は、修飾子で指定しているスキーマ内に作成されます。修飾されない場合:

- CURRENT SCHEMA 特殊レジスターの値が *LIBL である場合、表は、現行ライブラリー (*CURLIB) 内に作成されます。
- そうでない場合、表は現行スキーマ内に作成されます。

指定されたパーティション表がジャーナルに記録された場合は、新しい表も同じジャーナルに記録されます。そうでない場合は、新しい表もジャーナルに記録されません。

ADD MATERIALIZED QUERY マテリアライズ照会定義

基本表をマテリアライズ照会表に変更します。指定した表が既にマテリアライズ照会表である場合、あるいはその表が他のマテリアライズ照会表で参照されている場合は、エラーが戻されます。

選択ステートメント
表の基礎となる照会を定義します。既存の表の列は、以下の特性を満たしていなければなりません。

- 表の列の数は、選択ステートメント の結果列の数と同じでなければなりません。
- 表の各列の列属性は、選択ステートメント 内の対応する結果列の列属性と互換性がなければなりません。

マテリアライズ照会表の選択ステートメント には、変更する表の参照、変更する表に対するビュー、または他のマテリアライズ照会表を含めてはなりません。マテリアライズ照会表の選択ステートメント を指定することについての詳細は、 1238 ページの『CREATE TABLE』を参照してください。

refreshable-table-options
基本表をマテリアライズ照会表に変更するためのマテリアライズ照会表オプションを指定します。

DATA INITIALLY DEFERRED
表内のデータを ALTER TABLE ステートメントの一部として検証しないことを指定します。 REFRESH TABLE ステートメントを使用して、マテリアライズ照会表にあるデータが、その表を基礎とする照会の結果と同じになることを確証できます。

DATA INITIALLY IMMEDIATE
ALTER TABLE ステートメントの処理の一部として、データを照会の結果から表内に挿入することを指定します。

REFRESH DEFERRED
表内のデータを REFRESH TABLE ステートメントを使用していつでもリフレッシュできるように指定します。表内のデータは、REFRESH TABLE ステートメントの処理時または最後に更新された時のスナップショットとしての照会の結果のみを反映します。

MAINTAINED BY USER
マテリアライズ照会表がユーザーによって保守されるように指定します。ユーザーは表に対して INSERT、DELETE、UPDATE、または REFRESH TABLE ステートメントを使用できます。

ENABLE QUERY OPTIMIZATION または **DISABLE QUERY OPTIMIZATION**
このマテリアライズ照会表を照会の最適化に使用できるかどうかを指定します。

ENABLE QUERY OPTIMIZATION
マテリアライズ照会表を照会最適化に使用できます。

DISABLE QUERY OPTIMIZATION
マテリアライズ照会表は照会最適化に使用されません。それでもその表を直接照会することはできます。

select-statement 内で直接または間接に参照されている表で行レベルまたは列レベルのアクセス制御がアクティブになっていて、変更される表では行アクセス制御がアクティブになっていない場合、変更される表に対して行アクセス制御が暗黙的にアクティブ化されます。これにより、マテリアライズ照会表の内容への直接アクセスが制限されます。表を明示的に参照する照会では、表内にデータがないことを示す警告が戻されます。マテリアライズ照会表にアクセスするには、適切な行権限を作成します。またはふさわしいようであれば、マテリアライズ照会表に対する ALTER TABLE DEACTIVATE ROW ACCESS CONTROL を発行して、行レベルの保護を除去します。

ALTER MATERIALIZED QUERY マテリアライズ照会表変更

マテリアライズ照会表の属性を変更します。表名でマテリアライズ照会表を識別する必要があります。

選択ステートメント
表の基礎となる照会を定義します。既存の表の列は、以下の特性を満たしていなければなりません。

- 表の列の数は、選択ステートメント の結果列の数と同じでなければなりません。
- 表の各列の列属性は、選択ステートメント 内の対応する結果列の列属性と互換性がなければなりません。

マテリアライズ照会表の選択ステートメント には、変更する表の参照、変更する表に対するビュー、または他のマテリアライズ照会表を含めてはなりません。マテリアライズ照会表の選択ステートメント を指定することについての詳細は、 1238 ページの『CREATE TABLE』を参照してください。

refreshable-table-options
基本表をマテリアライズ照会表に変更するためのマテリアライズ照会表オプションを指定します。

DATA INITIALLY DEFERRED
表内のデータを ALTER TABLE ステートメントの一部としてリフレッシュも検証もしないことを指定します。 REFRESH TABLE ステートメントを使用して、マテリアライズ照会表にあるデータが、その表を基礎とする照会の結果と同じになることを確証できます。

DATA INITIALLY IMMEDIATE
ALTER TABLE ステートメントの処理の一部として、データを照会の結果から表内に挿入することを指定します。

REFRESH DEFERRED
表内のデータを REFRESH TABLE ステートメントを使用していつでもリフレッシュできるように指定します。表内のデータは、REFRESH TABLE ステートメントの処理時または最後に更新された時のスナップショットとしての照会の結果のみを反映します。

MAINTAINED BY USER
マテリアライズ照会表がユーザーによって保守されるように指定します。ユーザーは表に対して INSERT、DELETE、UPDATE、または REFRESH TABLE ステートメントを使用できます。

ENABLE QUERY OPTIMIZATION または **DISABLE QUERY OPTIMIZATION**
このマテリアライズ照会表を照会の最適化に使用できるかどうかを指定します。

ENABLE QUERY OPTIMIZATION
マテリアライズ照会表を照会最適化に使用できます。

DISABLE QUERY OPTIMIZATION
マテリアライズ照会表は照会最適化に使用されません。それでもその表を直接照会することはできます。

SET *refreshable-table-alteration*
表を保守する方法、または表を照会最適化に使用するかどうかを変更します。

MAINTAINED BY USER
マテリアライズ照会表がユーザーによって保守されるように指定します。ユーザーは表に対して INSERT、DELETE、UPDATE、または REFRESH TABLE ステートメントを使用できます。

REFRESH DEFERRED
表内のデータを REFRESH TABLE ステートメントを使用していつでもリフレッシュできるように指定します。表内のデータは、REFRESH TABLE ステートメントの処理時または最後に更新された時のスナップショットとしての照会の結果のみを反映します。

ENABLE QUERY OPTIMIZATION または **DISABLE QUERY OPTIMIZATION**
このマテリアライズ照会表を照会の最適化に使用できるかどうかを指定します。

ENABLE QUERY OPTIMIZATION
マテリアライズ照会表を照会最適化に使用できます。

DISABLE QUERY OPTIMIZATION
マテリアライズ照会表は照会最適化に使用されません。それでもその表を直接照会することはできます。

DROP MATERIALIZED QUERY

マテリアライズ照会表を変更して、その表がマテリアライズ照会表ではなくなるようにします。*table-name* によって指定される表は、マテリアライズ照会表として定義されている必要があります。列の定義は変更されませんが、この表はもはや照会最適化には使用できなくなり、 REFRESH TABLE ステートメントでの使用は無効になります。表に対する行レベルと列レベルのアクセスは、現在のアクティブ状態または非アクティブ状態のままになります。

ACTIVATE NOT LOGGED INITIALLY

この現行作業単位に対して表の NOT LOGGED INITIALLY 属性を活動化します。

このステートメントによって変更された表に対して同一作業単位内の INSERT、DELETE、または UPDATE ステートメントによって行われた変更は、ログ (ジャーナル) に記録されません。

現行作業単位の完了時に NOT LOGGED INITIALLY 属性が非活動化され、後続の作業単位で表に対して行われるすべての操作はログ (ジャーナル) に記録されます。

表名 に対するデータ変更操作が保留中の場合、または表名 を参照するカーソルがコミット下で現在オープン状態である場合、トランザクション内の ACTIVATE NOT LOGGED INITIALLY は許可されません。

表がシステム期間テンポラル表または履歴表の場合、ACTIVATE NOT LOGGED INITIALLY は使用できません。表に FILE LINK CONTROL が指定された DATALINK 列 がある場合や、表が分離レベル No Commit (NC) で実行されている場合、それは無視されます。

WITH EMPTY TABLE
表に現在あるすべてのデータが除去されます。この ALTER ステートメントが発行された作業単位がロールバックされると、表データは元の状態に戻らなくなります。このアクションが要求されると、影響を受ける表に定義された DELETE トリガーは起動しません。

マテリアライズ照会表、または参照制約内の親には、WITH EMPTY TABLE を指定することはできません。表が分離レベル No Commit (NC) を指定されて実行されている場合は、WITH EMPTY TABLE は無視されます。

WHERE 文節を使用しない DELETE ステートメントは一般に ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE と同等以上のパフォーマンスを示すうえ、ROLLBACK を使用して表の行の削除をロールバックすることを可能にします。

ADD PERIOD FOR *period-definition*

期間定義を表に追加します。

SYSTEM_TIME (*begin-column-name***, *end-column-name*)**
システム期間を SYSTEM_TIME という名前で定義します。表内に名前 SYSTEM_TIME の列があってはなりません。表に指定できる SYSTEM_TIME 期間は 1 つのみです。*begin-column-name* は ROW BEGIN として定義し、*end-column-name* は ROW END として定義する必要があります。

DROP PERIOD SYSTEM_TIME

SYSTEM_TIME 期間を表からドロップします。表がシステム期間テンポラル表である場合、期間 SYSTEM_TIME はドロップできません。

ADD VERSIONING USE HISTORY TABLE *history-table-name*

これを指定すると、表はシステム期間テンポラル表になります。表はシステム期間テンポラル表かまたは履歴表として既に定義されていてはなりません。表に SYSTEM_TIME 期間とトランザクション開始 ID 列が定義されている必要があります。表をマテリアライズ照会表および分散表にすることはできません。また、表に ROWID 列および FILE LINK CONTROL を指定した DATALINK 列が含まれてはなりません。この表および履歴表で、NOT LOGGED INITIALLY 属性をアクティブにすることはできません。

この表の行の履歴バージョンは、データベース・マネージャーによって保持されます。関連履歴表は、表の履歴行を保管するために使用されます。データベース・マネージャーは、行が表に挿入された時期とそれが更新または削除された時期を示す追加情報を記録します。システム期間テンポラル表の行が更新されると、その行の直前のバージョンが保持されます。システム期間テンポラル表のデータが削除されると、その行の古いバージョンが履歴レコードとして挿入されます。

この表に対する参照には、戻すデータのバージョンを指示する期間指定を含めることができます。

history-table-name
システム期間テンポラル表の履歴行を保持する履歴表を指定します。*history-table-name* は、現行サーバーに存在する表を指定するものでなければな

らず、カタログ表、既存のシステム期間テンポラル表、既存の履歴表、宣言済みのグローバル一時表、マテリアライズ照会表、およびビューのいずれでもありません。

指定する履歴表は、ID 列、行変更タイム・スタンプ列、行開始列、行終了列、トランザクション開始 ID 列、または期間を含むものであってはなりません。この履歴表は、参照制約の一部であることも、固有キー制約や基本キー制約を持つこともできません。

表が履歴表として一度定義されると、その表に直接の挿入および更新は実行できません。削除は可能です。

ON DELETE ADD EXTRA ROW

行がシステム期間テンポラル表から削除されると、削除された行が履歴表に追加されることを示します。この追加の履歴行は、システム期間テンポラル表の期間指定の照会に対して戻されません。この行が追加される際に、行開始列、行終了列、および生成式列の値が生成されます。

ON DELETE ADD EXTRA ROW 節は、システム期間テンポラル表に生成式列が含まれる場合に使用することを想定しています。追加行の生成式列には、履歴表の追加行に至った削除操作に関する情報が含まれます。

システム期間テンポラル表と指定した履歴表は、同じスキーマ内にあって、列の数と順序が同じでなければなりません。 2 つの表の対応する列の以下の属性は同じでなければなりません。

- 名前およびシステム列名
- データ・タイプ
- 長さ、精度、および位取り
- FOR BIT DATA、FOR SBCS DATA、FOR MIXED DATA 属性
- CCSID
- NULL 属性
- 非表示属性
- フィールド手順
- 日時の形式と区切り記号
- ソート順序と言語 ID

履歴表にデータが含まれている場合、データが履歴行を正確に表すようにしてください。データが履歴行の結果を正確に表さない場合、テンポラル照会が予期しない結果になる可能性があります。

行アクセス制御または列アクセス制御がシステム期間テンポラル表においてアクティブであるが、行アクセス制御が履歴表でアクティブでない場合、データベース・マネージャーは自動的に、履歴表での行アクセス制御をアクティブにし、デフォルトの行権限を履歴表に作成します。

DROP VERSIONING

表がもはやシステム期間テンポラル表でないことを指定します。表はシステム期間テンポラル表でなければなりません。この表では、履歴データの記録と保守が行われなくなります。列の定義および表のデータは変更されませんが、表はシステム期

間テンポラル表としては扱われません。 SYSTEM_TIME 期間は保持されます。これ以降、この表を参照する照会では、表の SYSTEM_TIME 期間指定を指定することはできません。システム期間テンポラル表と、関連する履歴表との関係は解除されます。履歴表はドロップされず、履歴表の内容は影響を受けません。

ACTIVATE ROW ACCESS CONTROL または DEACTIVATE ROW ACCESS CONTROL

有効な行の許可が表内のアクセス可能な行のセットを制御するために Db2 によって適用されるかどうかを指定します。

ACTIVATE ROW ACCESS CONTROL

表の行アクセス制御を活動化することを指定します。表が別名である場合は、基本表の行アクセス制御が活動化されます。宣言されたグローバル一時表または QTEMP 中の表に対して行アクセス制御を活動化することはできません。

有効な行の許可が存在していて、行の許可の定義に指定された権限 ID またはグループ・プロファイルにアクセス権を認めている場合を除いて、デフォルトの行の許可が暗黙的に作成され、表のどの列へのアクセスも許可されません。そのような行の許可がまだ存在していない状態で表を参照する照会では、表内にデータがないことを示す警告が戻されます。

表に対するトリガーが存在する場合、そのトリガーは SECURED 属性を指定して定義される必要があり、READ トリガーであってはなりません。

ビューに関して、NOT SECURED 属性を指定して定義された INSTEAD OF トリガーが存在する場合、そのビューの定義内で、この表を参照してはなりません。

この表がデータ操作ステートメントで参照されるとき、その表に対して作成されたすべての有効な行権限 (デフォルトの行権限を含む) が Db2 によって適用され、表の中でアクセス可能な行セットが制御されます。活動化しようとするとエラーになる許可は、許可定義にあるエラーがすべて解決されるまで活動化できません。これには、許可を除去し、変更した定義で再作成することが必要になることがあります。

マテリアライズ照会表が、行レベルのアクセス制御が活動化されている表に (直接的に、またはビューを介して間接的に) 依存しており、そのマテリアライズ照会表の行レベルのアクセス制御がまだ活動化されていない場合、そのマテリアライズ照会表に対して行レベルのアクセス制御が暗黙的に活動化されます。これにより、マテリアライズ照会表の内容への直接アクセスが制限されます。行に対するそのような行権限が定義される前に表を照会で明示的に参照すると、表にデータがないことを示す警告が返されます。マテリアライズ照会表にアクセスするには、適切な行権限を作成します。またはふさわしいようであれば、マテリアライズ照会表に対する ALTER TABLE DEACTIVATE ROW ACCESS CONTROL を発行して、行レベルの保護を除去します。

この表において行アクセス制御が既にアクティブとして定義されている場合、ACTIVATE ROW ACCESS CONTROL は無視されます。

表がシステム期間テンポラル表で、行アクセス制御が履歴表でまだアクティブでない場合、データベース・マネージャーは自動的に、履歴表での行アクセス制御をアクティブにし、履歴表に対してデフォルトの行権限を作成します。

DEACTIVATE ROW ACCESS CONTROL
表の行アクセス制御を非活動化することを指定します。この表がデータ操作ステートメントで参照されるとき、その表に対して定義された有効な行権限は Db2 によって適用されないため、表の中でアクセス可能な行セットは制御されません。

DEACTIVATE ROW ACCESS CONTROL は、行アクセス制御をこの表に対して活動化しないよう既に定義されている場合は、無視されます。

ACTIVATE COLUMN ACCESS CONTROL または DEACTIVATE COLUMN ACCESS CONTROL

有効な列マスクが、表から戻された列値をマスクするために Db2 によって適用されるかどうかを指定します。

ACTIVATE COLUMN ACCESS CONTROL
表の列アクセス制御を活動化することを指定します。表が別名である場合は、基本表の列アクセス制御が活動化されます。

表に対するトリガーが存在する場合、そのトリガーは SECURED 属性を指定して定義される必要があり、READ トリガーであってはなりません。

ビューに関して、NOT SECURED 属性を指定して定義された INSTEAD OF トリガーが存在する場合、そのビューの定義内で、この表を参照してはなりません。

表へのアクセスは制限されませんが、この表がデータ操作ステートメントで参照されるときには、表に対して作成されたすべての有効な列マスクが Db2 によって適用されて、照会の最終結果表で参照される列について戻される値のマスクや、データ変更ステートメントで使用される新しい値の判別が行われます。活動化しようとするとエラーになる列マスクは、マスク定義にあるエラーがすべて解決されるまで活動化できません。これには、列マスクを除去し、変更した定義で再作成することが必要になることがあります。

列レベルのアクセス制御が活動化中の表に依存している (直接的に、またはビューを介して間接的に) マテリアライズ照会表があり、そのマテリアライズ照会表では行レベルのアクセス制御がまだ活動化されていない場合、そのマテリアライズ照会表に対して行レベルのアクセス制御が暗黙的に活動化されます。これにより、マテリアライズ照会表の内容への直接アクセスが制限されます。行に対するそのような行権限が定義される前に表を照会で明示的に参照すると、表にデータがないことを示す警告が返されます。マテリアライズ照会表にアクセスするには、適切な行権限を作成します。またはふさわしいようであれば、マテリアライズ照会表に対する ALTER TABLE DEACTIVATE ROW ACCESS CONTROL を発行して、行レベルの保護を除去します。

ACTIVATE COLUMN ACCESS CONTROL は、表に対して列アクセス制御を活動化するよう既に定義されている場合は無視されます。

表がシステム期間テンポラル表で、行アクセス制御が履歴表でまだアクティブでない場合、データベース・マネージャーは自動的に、履歴表での行アクセス制御をアクティブにし、履歴表に対してデフォルトの行権限を作成します。

DEACTIVATE COLUMN ACCESS CONTROL
表の列アクセス制御を非活動化することを指定します。表がデータ操作ステート

メント内で参照されるときに、表に対して定義されている有効な列マスクは Db2 によって適用されず、照会の最終結果表内で参照される列について戻される値が制御されたり、データ変更ステートメントで新しい値が使用可能かどうかが判別されることはありません。

DEACTIVATE COLUMN ACCESS CONTROL は、列アクセス制御をこの表に対して活動化しないよう既に定義されている場合は、無視されます。

VOLATILE または NOT VOLATILE

表表名 のカーディナリティーを実行時に大きく変えることができるかをオプティマイザーに示します。揮発性は表の行数に適用され、表そのものに適用されるわけではありません。デフォルトは NOT VOLATILE です。

VOLATILE
実行時に表名 のカーディナリティーを空から大規模に大きく変えることができることを指定します。オプティマイザーは表にアクセスするとき、可能であれば通常は索引を使用します。

NOT VOLATILE
table-name のカーディナリティーが揮発性でないことを指定します。この表を参照するアクセス・プランは、アクセス・プランが構築された時点の表のカーディナリティーに基づいたものになります。

media-preference

表またはパーティションの優先ストレージ・メディアを指定します。

UNIT ANY
どのストレージ・メディアも優先しません。使用可能なストレージ・メディアであればどのストレージ・メディアからでも表またはパーティションのストレージが割り振られます。表で UNIT ANY を指定すると、パーティションで指定する *media-preference* が使用されます。現時点で表またはパーティションがソリッド・ステート・ディスクのストレージに存在する場合に、他のメディアが使用可能な状態になっていれば、その表またはパーティションが非同期モードで他のメディアに移動することもあります。

UNIT SSD
ソリッド・ステート・ディスク・ストレージ・メディアを優先します。ソリッド・ステート・ディスクのストレージ・メディアが使用可能になっていれば、そのソリッド・ステート・ディスクのストレージ・メディアから表またはパーティションのストレージが割り振られます。表で UNIT SSD を指定すると、パーティションで指定する *media-preference* は無視されます。現時点で表またはパーティションがソリッド・ステート・ディスクのストレージに存在しない場合に、ソリッド・ステート・ディスクのストレージ・メディアが使用可能な状態になっていれば、その表またはパーティションが非同期モードでソリッド・ステート・ディスクのストレージ・メディアに移動することもあります。

memory-preference

KEEP IN MEMORY
表のデータが照会で使用されるときに、データを主記憶域プールに入れるかどうかを指定します。

NO データは主記憶域プールに入れられません。

YES
データは主記憶域プールに入れられます。

注

列参照**:** ALTER TABLE ステートメントの ADD、ALTER、または DROP COLUMN 文節では、1 つの列を *1* 回 だけ参照できます。ただし、同じ ALTER TABLE ステートメント内で制約を追加またはドロップする場合は、同じ列を複数回参照できます。

操作の順序**:** ALTER TABLE ステートメント内での操作の順序は、次のとおりです。

- 期間の除去
- 制約の除去
- マテリアライズ照会表の除去
- パーティションの除去
- パーティション化の除去
- RESTRICT オプションが指定された列の除去
- 他のすべての列定義の変更
  - CASCADE オプションが指定された列の除去
  - 列除去属性の変更 (例、DROP DEFAULT)
  - 列変更属性の変更
  - 列追加属性の変更
  - 列の追加
- パーティションの変更
- マテリアライズ照会表の追加または変更
- パーティションまたはパーティション化の追加
- 制約の追加
- 期間の追加

上記の各段階内で、ユーザーが文節を指定する順序は、文節が実行される順序になります。ただし、これには例外が 1 つあります。列のいずれかが除去される場合は、操作は列定義の追加または変更が行われる前に論理的に実行されます。

QTEMP 考慮事項**:** ビューまたは別のジョブの QTEMP 内の論理ファイルが、変更される表に従属している場合は、それらはいずれも ALTER TABLE ステートメントの結果として除去されます。このようなオブジェクトの除去は、その変更がカーディナリティーの変更、メディア・プリファレンスの変更、メモリー・プリファレンスの変更、制約の追加または削除、行アクセス制御および列アクセス制御のアクティブ化または非アクティブ化である場合や、ACTIVATE NOT LOGGED INITIALLY が指定されている場合には行われません。

権限検査**:** 権限検査が実行されるのは、変更中の表および ALTER TABLE ステートメント内で明示的に参照されるオブジェクト (例えば、全選択で参照された表など) に対してのみです。それ以外のオブジェクトには ALTER TABLE ステートメント

でアクセスできますが、それらのオブジェクトに対する権限はまったく必要ありません。例えば、除去される表に存在しているビューに対しても、除去される表を参照制約によって参照する従属表に対しても、権限はまったく必要ありません。

バックアップの推奨: 表を変更するにあたっては、あらかじめその表と従属ビュー、および論理ファイルの現行バックアップをとっておくことをぜひお勧めします。

パフォーマンスの考慮事項: 次のパフォーマンスに関する考慮事項は、表に対して列の追加、変更、または除去を行う際に ALTER TABLE ステートメントに適用されます。

- 表内のデータはコピーできます。[81]

  列を追加および除去する場合は、データをコピーする必要があります。

  列を変更する場合は、通常、データをコピーする必要があります。ただし、変更の内容が単に次のような場合は、データをコピーする必要はありません。

  - VARCHAR または VARBINARY 列の長さ属性が増やされる場合に、現行の長さ属性が 20 よりも大きい。
  - VARGRAPHIC 列の長さ属性が増やされる場合に、現行の長さ属性が 10 よりも大きい。
  - VARCHAR または VARBINARY 列の割り振られた長さが変更される場合に、現行および新規の割り振られた長さが共に 20 より小さいか同じである。
  - VARGRAPHIC 列の割り振られた長さが変更される場合に、現行および新規の割り振られた長さが共に 10 より小さいか同じである。
  - 列の CCSID が変更される場合に、古い CCSID と新規の CCSID との間で変換が不要である。例えば、1 つの CCSID が 65535 である場合、データ変換は不要。
  - デフォルト値が変更される場合に、デフォルト値の長さが現行の割り振られた長さより大きくない。
  - DROP DEFAULT が指定されている。
  - DROP NOT NULL が指定されているのに、表の変更が完了した後も、少なくとも 1 つの NULL 可能列がその表の中に依然として存在している。

  フィールド・プロシージャーが定義されている列を変更するには、そのフィールド・プロシージャーを 2 回実行しなければならない場合があります。
- 索引の再作成が必要になる可能性があります。[82]

  列が表に追加される場合や列が除去または変更されるときにそれらの列が索引キー内で参照されない場合は、索引を再作成する必要はありません。

  索引や制約のキー内で使用される列を変更する場合は、通常、その索引を再作成する必要があります。ただし、次のような場合は、索引の再作成は不要です。

---

81. 記憶域が不足していて完全なコピーを作成できない場合は、必要なフリー・ストレージが約 16 ～ 32 メガバイトだけで済むという特殊コピーが実行されます。

82. 再作成が必要な索引は、すべて、データベース・サーバー・ジョブによって非同期で再作成されます。

- VARCHAR、VARBINARY、または VARGRAPHIC キーの長さ属性が増やされる場合。
- 列の CCSID が変更される場合に、古い CCSID と新規の CCSID との間で変換が不要である。例えば、1 つの CCSID が 65535 である場合。

生成列の追加**:** 既存の表に行変更タイム・スタンプ列、行開始列、行終了列、トランザクション開始 ID 列、または生成式列を追加する場合は、変更操作の実行時に初期値が生成されます。

暗黙的に隠された列に対する考慮事項**:** 暗黙的に隠されたものとして定義された列は、ALTER ステートメントで明示的に参照できます。例えば、暗黙的に隠された列は、参照制約またはチェック制約の一部として、あるいはマテリアライズ照会表定義として指定できます。

マテリアライズ照会表の変更**:** 基本表が ALTER TABLE ステートメントによって最初にマテリアライズ照会表に変更されたときの分離レベルが、マテリアライズ照会表の分離レベルになります。

ある表を照会最適化が使用可能なマテリアライズ照会表に変更すると、その表を最適化で使用することが可能になります。このため、表内のデータが正しいことを確認してください。 DATA INITIALLY IMMEDIATE 文節を使用して、表の変更時にデータをリフレッシュすることができます。

従属するビュー、索引、およびマテリアライズ照会表に対する **ATTACH PARTITION** の影響**:**

- ソース表のビューと DDS 作成論理ファイルは除去されます。
- ターゲット表を参照するビューに、新しいパーティションが含まれるようになりました。
- アタッチの前にすべてのパーティションを参照していた DDS 作成論理ファイルに、新しいパーティションが含まれるようになりました。
- ソース表のマテリアライズ照会表は除去されます。
- ターゲット表のマテリアライズ照会表は保持されますが、新しいパーティションに関連付けられたデータを含むように、ユーザーがマテリアライズ照会表を更新する必要があります。
- ターゲット表のパーティション化索引に対応する、ソース表のパーティション化 (複数のパーティションにわたらない) 索引は、索引の論理ページ・サイズが同じである限り、ターゲット表のパーティション化索引の一部として保持されます。論理ページ・サイズが異なる場合、そのパーティションの索引は除去されます。
- ターゲット表のパーティション化索引に対応しない、ソース表のパーティション化 (複数のパーティションにわたらない) 索引は、除去されます。
- ソース表の索引に対応しない、ターゲット表のパーティション化 (複数のパーティションにわたらない) 索引は、新しいパーティションを含むように変更されます。
- ターゲット表の非パーティション化 (複数のパーティションにわたる) 索引は、新しいパーティションを含むように再作成されます。

従属するビュー、索引、およびマテリアライズ照会表に対する **DETACH PARTITION** の影響**:**

- ソース表のビューに、デタッチされたパーティションは含まれなくなります。
- デタッチの前にすべてのパーティションを参照していた DDS 作成論理ファイルは、残りのパーティションを含みます。
- ソース表のマテリアライズ照会表は保持されますが、デタッチされたパーティションに関連付けられたデータを削除するように、ユーザーがマテリアライズ照会表を更新する必要があります。
- ソース表のパーティション化 (複数のパーティションにわたらない) 索引は、デタッチされたパーティション索引を削除するように変更されます。
- ソース表の非パーティション化 (複数のパーティションにわたる) 索引は、再作成されます。

行アクセス制御と列アクセス制御の使用に関する考慮事項:

明示的にアクティブ化される行アクセス制御: 表の行アクセス制御をアクティブにするために、ACTIVATE ROW ACCESS CONTROL 節が使用されます。活動化されると、表のデフォルトの行の許可が暗黙的に作成され、表のどの行にもアクセスできなくなります。ただし、許可の定義で指定された許可 ID のアクセス権限を提供する別の行の許可が、後で使用可能になり、存在する場合を除きます。デフォルトの行の許可は、常に使用可能です。

表がデータ操作ステートメントで参照される場合、その表に作成されていたすべての使用可能な行の許可 (デフォルトの行の許可を含む) は、表のどの行をアクセス可能とするかを制御するために、Db2 によって暗黙的に適用されます。行アクセス制御検索条件は、使用可能なそれぞれの行の許可で検索条件に論理 OR 演算子を適用することによって派生されます。この派生された検索条件は、述部、グループ化、順序付けなどのユーザー指定の操作が処理される前に、表に対するフィルターとして動作します。この得られた検索条件は、権限定義で指定した許可 ID でこの表の特定の行にアクセスすることを許可します。使用可能な行の許可を適用することによるフェッチ操作への影響の詳細については、副選択の説明を参照してください。使用可能な行の許可を適用することによるデータ変更操作への影響の詳細については、データ変更ステートメントを参照してください。

行アクセス制御は、DEACTIVATE ROW ACCESS CONTROL 文節を使用して行アクセス制御の実施を停止するまで実施されます。

表に対して行アクセス制御がアクティブになるときに作成される暗黙的な行の許可: ACTIVATE ROW ACCESS CONTROL 節を使用して表の行アクセス制御をアクティブにするとき、Db2 は表のデフォルトの行権限を暗黙的に作成します。デフォルトの行の許可では、表に対するすべてのアクセスができなくなります。暗黙的に作成された行の許可は、基本表と同じスキーマ内にあり、名前は QIBM_DEFAULT_*system-table-name_system_schema_name* 形式です。

デフォルトの行権限は、常に使用可能になっています。

行アクセス制御が非活動化されるか、または表がドロップされると、デフォルトの行の許可はドロップされます。

列アクセス制御のアクティブ化: 表の列のアクセス制御をアクティブにするために、ACTIVATE COLUMN ACCESS CONTROL 節が使用されます。表に対するアクセ

スは制限されませんが、データ操作ステートメントでこの表が参照される場合は、この表に対して作成されている使用可能なすべての列マスクが適用され、照会の最終結果表で参照される列値のマスクや、データ変更ステートメントで使用される新しい値の判別が行われます。

列マスクを使用して列値をマスクする場合、それらのマスクによって最終結果表の値が決まります。列マスクがある列 (列名の単純参照であるか、式に組み込まれている) が最外部の選択リスト中に出現する場合、列に列マスクが適用されて最終結果表の値が生成されます。列が最外部の選択リスト中に出現しないものの、最終結果表に関与する (例えば、ネストされた表式またはビュー内に出現する) 場合、最終結果表で使用できるように、ネストされた表式またはビューの結果表に戻される列に、列マスクが適用されます。

列マスクの適用は、ステートメント内の他の節 (WHERE、GROUP BY、HAVING、SELECT DISTINCT、および ORDER BY など) の操作に干渉しません。最終結果表に戻される行は同じままです。ただし、結果の行の値は、列マスクによってマスクされている場合があります。そのため、マスクされた列が ORDER BY ソート・キーにも出現する場合、順序は元の列値に基づくものとなり、最終結果表でマスクされた値にその順序が反映されない可能性があります。同様に、マスクされた値は、SELECT DISTINCT によって強制適用される固有性が反映されない可能性があります。マスクされる列が式に組み込まれていると、列マスクは式の評価が行われる前に列に適用されるため、式の結果が異なる場合があります。例えば、列 SSN に列マスクを適用すると、マスク値に対して DISTINCT 操作が行われるため、集約関数 COUNT(DISTINCT SSN) の結果が変わることがあります。一方、照会の式が、列マスク定義で列値のマスクに使用された式と同じである場合、照会の式の結果は変わらない可能性があります。例えば、照会の式が 'XXX-XX-' || SUBSTR(SSN, 8, 4) で、列マスク定義に同じ式が指定されているとします。この特定の例では、ユーザーは照会の式を列 SSN で置き換えて、同じ式が 2 回評価されるのを回避できます。

以下に、照会の結果の列値をマスクするために Db2 が列マスクを使用するコンテキストを示します。特定の制約事項が、いくつかのコンテキストに適用されることがあります。この制約事項は、分離リストで説明されています。

- SELECT または SELECT INTO ステートメントの最外部 SELECT 節。あるいは、列が最外部の選択リスト内に出現しないものの最終結果表に関与する場合は、対応するネストされた表式、共通表式、または列が出現するビューの最外部 SELECT 節。
- INSERT、UPDATE、または MERGE ステートメントの新しい値を導き出すために使用される最外部 SELECT 節。
- 上記ステートメントの最外部 SELECT 節の中、 SET 変数割り当てステートメントの右辺、VALUES INTO ステートメントの右辺、または VALUES ステートメントの右辺に出現する、スカラー全選択式。

列マスクは作成されますが、これが使用される可能性があるコンテキストのすべてが認識されるわけではありません。最終結果表内の列値をマスクするために、列マスク定義は Db2 によってステートメントにマージされます。列マスク定義は、ステートメントのコンテキストに取り込まれると、ステートメント内の特定の SQL セマンティクスと競合することがあります。したがって、状態によっては、ステー

トメントと列マスクの適用とを組み合わせることによりエラーが戻されることがあります。以下に、エラーが戻される可能性のある状態を説明します。

- 副選択の FROM 文節で再帰的共通表式を参照し、最終結果表の派生に再帰的共通表式の結果を使用する場合、列マスクは、再帰的共通表式の全選択で参照される列には適用できません。
- ユーザー定義関数が NOT SECURED オプションで定義されている場合、この関数の引数は、列マスクが使用可能にされ、かつその表の列アクセス制御が活動化される列を参照していてはなりません。この規則は、ステートメントの任意の場所で参照されるユーザー定義関数に適用されます。

上記のエラー状態を回避するには、以下のいずれかのアクションを行う必要があります。

- ステートメントの上記コンテキストを変更するか、ステートメントから削除する
- 列マスクを使用不可にする
- 列マスクをドロップし、定義を変更して、列マスクを再作成する
- 表の列アクセス制御を非活動化する

状況によっては、ステートメントに SELECT DISTINCT または GROUP BY が含まれていて、SELECT DISTINCT または GROUP BY の結果を直接または間接に導出する列に列マスクが適用される場合、ステートメントが戻す結果が一定でないことがあります。該当するのは、以下のような状況です。

- 列マスク定義が、列マスクが適用される先の列と同じ表にある他の列を参照している。
- 列が組み込みスカラー関数 (COALESCE、IFNULL、NULLIF、MAX、MIN、LOCATE など) の引数で参照されている。
- 列が集約関数の引数で参照されている。
- 列が式に組み込まれ、その式に非 deterministic 関数かまたは外部アクションのある関数が含まれている。

列が NULL 可能ではない場合、おそらく、列の列マスクの定義で列の NULL 値は考慮されません。外部結合の NULL 埋め込み表の場合、表の列アクセス制御がアクティブにされた後で、最終結果表の列値が NULL になる可能性があります。列マスクにより NULL 値をマスクできることを確実にするため、Db2 では、外部結合の NULL 埋め込み表の場合には列マスク定義に 1 番目の WHEN 文節として WHEN target-column IS NULL THEN NULL が追加されます。これにより、NULL 値が常に NULL 値として強制的にマスクされます。NULL 可能列の場合、これにより、NULL 値を他の値としてマスクすることができなくなります。例 4 に、この追加される WHEN 文節を示します。

INSERT、UPDATE、および MERGE では、新しい行の値を導出するときに列が参照されている場合に、その列に対して有効な列マスクがあれば、マスクされた値が新しい値を導出するために使用されます。オブジェクト表でも列アクセス制御がアクティブになっている場合、新しい値を導出するために適用される列マスクは、定数や式ではなく、列自体を戻す必要があります。列マスクを列に適用した結果が列自体にならない場合、新しい値を挿入または更新に使用できず、エラーが戻されます。新規の値を派生させるために列マスクの適用に使用される規則は、照会の最終

結果表について前述したのと同じ規則に従います。挿入可能性と更新可能性に影響する列マスクの使用方法については、データ変更ステートメントを参照してください。

列マスクを適用できるのは、基本表の列のみです。ビュー、ネストされた表式、または共通表式の列が最終結果表に関係する場合、上記のエラー状態が、関係するビュー、ネストされた表式、または共通表式の内部で発生する可能性があります。

列アクセス制御は、XMLTABLE 組み込み関数には影響を与えません。XMLTABLE 関数の入力が、列マスクのある列の場合、列マスクは適用されません。

列アクセス制御は、DEACTIVATE COLUMN ACCESS CONTROL 文節を使用して列アクセス制御の実施を停止するまで実施されます。

列マスクとトリガー遷移変数**:** OLD ROW および OLD TABLE 遷移変数の値に、マスクされた値が含まれることはありません。

SET 遷移変数 割り当てステートメントは、マスクされたデータを変数に割り当てることができます。列に違反チェック制約が存在しない場合、マスクされたデータが列で挿入または更新され、エラーは発行されません。

行または列のアクセス制御の強制適用の停止**:** 表の行アクセス制御の強制適用を停止するために、DEACTIVATE ROW ACCESS CONTROL 節が使用されます。デフォルトの行の許可はドロップされます。停止された後は、データ操作ステートメントで表が参照される場合、明示的に作成された行の許可は適用されません。この表は、付与された特権に基づいてアクセスできます。

DEACTIVATE COLUMN ACCESS CONTROL 文節は、表の列アクセス制御の実施を停止する場合に使用します。停止された後は、データ操作ステートメントで表が参照される場合、列マスクは適用されません。最終結果表には、マスクされない列値が使用されます。明示的に作成された行の許可または列マスク (ある場合) は、そのままですが有効ではありません。

行と列のアクセス制御のセキュア・トリガー**:** データベース保全性のためにトリガーが使用されます。そのため、行と列のアクセス制御 (セキュリティー) とデータベース保全性の間でバランスが必要です。有効な行の許可および列マスクは、遷移変数および遷移表の初期値には適用されません。トリガー表に適用される行アクセス制御と列アクセス制御も、トリガー本体内で参照されている遷移変数または遷移表に対しては無視されます。トリガー・アクションの SQL ステートメントが遷移変数と遷移表に含まれる機密データにアクセスするときのセキュリティー上の心配をなくすため、トリガーは SECURED オプションを指定して作成または変更する必要があります。トリガーがセキュアでない場合、トリガー表に対して行および列のアクセス制御を実施することはできません。

行と列のアクセス制御におけるセキュアなユーザー定義関数**:** 行権限または列マスクの定義でユーザー定義関数を参照する場合、機密データが引数として関数に渡される可能性があるため、その関数は SECURED オプションを指定して変更する必要があります。

Db2 は SECURED オプションを、ユーザー定義関数に対するすべての変更の変更制御監査手順をユーザーが確立したことを宣言するアサーションと見なします。ま

た、このような制御監査手順はユーザー定義関数のすべてのバージョンに適していると想定し、また後続のすべての ALTER FUNCTION ステートメント、または外部プログラムに対する変更は監査プロセスで確認されると想定します。

行と列のアクセス制御が適用されないデータベース操作**:** 行と列のアクセス制御を使用するためにデータベース保全性を犠牲にすることがないようにしてください。主キー、ユニーク・キー、索引、チェック制約、および参照整合性に関係する列は、行と列のアクセス制御が適用されないようにする必要があります。こういった列に対して列マスクを定義することはできますが、それらの列マスクは、キー作成プロセス中、または、制約または RI の強制適用プロセス中には適用されません。

システム期間テンポラル表の定義**:** システム期間テンポラル表の定義には、以下の事柄が含まれます。

- SYSTEM_TIME という名前のシステム期間。これは、行開始列と行終了列を使用して定義されます。 959 ページの『AS ROW BEGIN』、 960 ページの『AS ROW END』、および 982 ページの『ADD PERIOD FOR period-definition』を参照してください。
- トランザクション開始 ID 列。 960 ページの『AS TRANSACTION START ID』 を参照してください。
- システム期間データ・バージョン管理定義。これは、関連した履歴表の名前が含まれる ADD VERSIONING アクションを使用する、後続の ALTER TABLE ステートメントで指定されます。 982 ページの『ADD VERSIONING USE HISTORY TABLE history-table-name』 を参照してください。

トランザクション開始 **ID** 列に関する考慮事項**:** トランザクション開始 ID 列で NULL 値が許可され、行開始列が存在し、その行開始列の値が他のトランザクションで生成された行開始列の値とは異なる固有の値になっている場合、トランザクション開始 ID 列には NULL 値が含まれます。列に NULL 値が含まれる可能性があるため、その列から値を取り出すときには、以下のいずれかの方式を使用することをお勧めします。

- COALESCE (transaction_start_id_col, row_begin_col)
- CASE WHEN transaction_start_id_col IS NOT NULL THEN transaction_start_id_col ELSE row_begin_col END

システム期間テンポラル表と、行および列のアクセス制御に関する考慮事項**:** 行および列のアクセス制御は、システム期間テンポラル表と、関連する履歴表の両方に定義できます。

- システム期間テンポラル表へのアクセス時、システム期間テンポラル表に定義された行および列のアクセス規則は、行がシステム期間テンポラル表に格納されているか履歴表に格納されているかに関係なく、システム期間テンポラル表から戻される行のすべてに適用されます。履歴表に定義された行および列のアクセス規則は、適用されません。
- 履歴表に直接アクセスする際は、履歴表に定義された行および列のアクセス規則が適用されます。

システム期間テンポラル表が定義されていて、システム期間テンポラル表の行アクセス制御または列アクセス制御がアクティブであり、履歴表の行アクセス制御がま

だアクティブでない場合、データベース・マネージャーは自動的に、履歴表での行アクセス制御をアクティブにし、履歴表に対してデフォルトの行権限を作成します。

履歴表の管理: 許可ユーザーの場合、履歴表に限定的な変更を直接行うことができます。表からの行の削除、パーティションの追加および除去を行えます。

読み取り専用カーソルおよび読み取り専用ビュー: 読み取り専用カーソルまたは読み取り専用ビューを判別するために使用されるルールは、行アクセス制御および列アクセス制御の影響を受けません。使用可能な列マスクの適用の影響は、実行時まで不明です。したがって、書き込み可能カーソルまたは書き込み可能ビューに対するデータ変更操作は、実行時に依然として失敗することがあります。

代替構文: 以下の構文は、旧リリースとの互換性を維持するためにサポートされています。この構文は標準構文ではないので、使用しないようにしてください。

- INLINE LENGTH は ALLOCATE の同義語です。
- ADD 制約が ALTER TABLE ステートメントの最初の文節である場合は、ADD キーワードは任意指定ですが、指定することを強くお勧めします。それ以外の場合は、ADD キーワードは必須です。
- 参照制約内の FOREIGN KEY キーワードの後に制約名 (CONSTRAINT キーワードなし) を指定することもできます。
- PART は PARTITION の同義語です。
- PARTITION パーティション名 の代わりに、 PARTITION パーティション番号 を指定できます。パーティション番号で、表の既存のパーティションまたは ALTER TABLE ステートメントで以前に指定したパーティションを識別することはできません。

  partition-number を指定しない場合は、データベース・マネージャーによって固有のパーティション番号が生成されます。
- VALUES は ENDING AT の同義語です。
- SET MATERIALIZED QUERY AS DEFINITION ONLY は DROP MATERIALIZED QUERY の同義語です。
- SET SUMMARY AS DEFINITION ONLY は DROP MATERIALIZED QUERY の同義語です。
- SET MATERIALIZED QUERY AS (選択ステートメント) は、ADD MATERIALIZED QUERY (選択ステートメント) の同義語です。
- SET SUMMARY AS (選択ステートメント) は、ADD MATERIALIZED QUERY (選択ステートメント) の同義語です。

カスケード効果

列を追加しても、SQL ビュー、マテリアライズ照会表、または大部分の論理ファイルに対するカスケード効果はありません。例えば、表に列を追加しても、従属するビューに列は追加されません (これらのビューが SELECT * 文節を指定して作成された場合でも)。

列を追加しても、その新しい列を組み込むために SQL トリガーが再作成されることはありません。

列を除去または変更した場合は、数種類のカスケード効果が生じる可能性があります。 表 78 に列を除去した場合のカスケード効果を示します。

表 *78.* 列の除去によるカスケード効果

操作	**RESTRICT** 効果	**CASCADE** 効果
ビューによって参照した列の除去	列の除去は許されません。	ビューおよびそのビューに従属しているビューすべてが除去されます。
非ビュー論理ファイルで参照する列の除去	この除去は可能で、次の場合に列が論理ファイルから除去されます。 • 論理ファイルが、変更しようとしているファイルと形式を共用する。また、 • 除去された列が、キー・フィールドとして使用されなかったり、選択/省略仕様で使用されない。また、 • 形式が、論理ファイルにおいて別のベースオン・ファイルで再使用されない。 それ以外の場合は、列の除去は許されません。	この除去は可能で、次の場合に列が論理ファイルから除去されます。 • 論理ファイルが、変更しようとしているファイルと形式を共用する。また、 • 除去された列が、キー・フィールドとして使用されなかったり、選択仕様や省略仕様で使用されない。また、 • 形式が、論理ファイルにおいて別のベースオン・ファイルで再使用されない。 それ以外の場合、論理ファイルは除去されます。
索引でキーの一部として、WHERE 節で、INCLUDE 節で、あるいは、列の ADD 節で明示的または暗黙的な列として参照された列の除去	索引の除去は不可能です。	索引は除去されます。
キーが存在する物理ファイルのキーで参照されている列の削除 (キーが主キーではない場合)	その物理ファイルは、キーが存在しない物理ファイルに変更されます。	その物理ファイルは、キーが存在しない物理ファイルに変更されます。

83. 列を物理ファイルに追加する場合、列は、その物理ファイルの形式を共用する論理ファイルにも追加されます (ただし、その形式がその論理ファイルにおいて別の基準ファイルで再使用されない場合に限ります)。

表 78. 列の除去によるカスケード効果 (続き)

操作	RESTRICT 効果	CASCADE 効果
固有制約の中で参照した列の除去	固有制約の中で参照した列がすべて同じ ALTER COLUMN ステートメントで除去され、しかもその固有制約が参照制約によって参照されていない場合は、それらの列および制約は除去されます。(したがって、この制約を満たすために使用された索引もすべて除去されます。) 例えば、列 A が除去され、固有制約 UNIQUE(A) または PRIMARY KEY(A) が存在し、この固有制約を参照する参照制約がない場合は、この操作が許されます。 それ以外の場合は、列の除去は許されません。	固有制約は、その固有制約を参照する参照制約と同じように、除去されます。 (したがって、それらの制約によって使用された索引もすべて除去されます。)
参照制約の中で参照した列の除去	参照制約の中で参照した列がすべて同時に除去される場合は、それらの列および制約は除去されます。 (したがって、外部キーによって使用された索引も除去されます。) 例えば、列 B が削除され、参照制約 FOREIGN KEY (A) が存在する場合は、この操作が許されます。 それ以外の場合は、列の除去は許されません。	参照制約は除去されます。(したがって、外部キーによって使用された索引も除去されます。)
SQL トリガーの中で参照した列の除去	列の除去は許されません。	SQL トリガーは除去されます。
MQT の中で参照した列の除去	列の除去は許されません。	MQT は除去されます。

表 79 に列の変更によるカスケード効果をリストしてあります。 (次の表で列の変更という場合は、データ・タイプ、精度、位取り、長さ、または NULL 可能性特性の変更を意味します。)

表 79. 列の変更によるカスケード効果

操作	効果
ビューによって参照した列の変更	変更が許されます。 表に従属しているビューが再作成されます。ビューの再作成時には、新しい列属性が使用されます。

表 79. 列の変更によるカスケード効果 (続き)

操作	効果
非ビュー論理ファイルで参照する列の変更	変更が許されます。 表に従属している非ビュー論理ファイルが再作成されます。論理ファイルが、変更しようとしているファイルと形式を共用する場合、および、形式が論理ファイルにおいて別のベースオン・ファイルで再使用されない場合は、論理ファイルの再作成時に新規の列属性が使用されます。 それ以外の場合は、論理ファイルの再作成時に新規の列属性は使用されません。代わりに、現行の論理ファイル属性が使用されます。
索引のキーで参照される列の変更	変更が許されます。(したがって、通常、索引は再作成されます。)
固有制約の中で参照した列の変更	変更が許されます。(したがって、通常、索引は再作成されます。) ユニーク制約が参照制約によって参照されている場合は、外部キーの属性がそのユニーク制約の属性 (フィールド・プロシージャーを含む) と一致しなくなります。その制約は定義済み検査保留状態に置かれます。
参照制約の中で参照した列の変更	変更が許されます。 • 参照制約が定義済み検査保留状態にある場合は、変更が許され、その制約を使用可能状態に置く試みがなされます。 (したがって、通常、固有制約を満たすために使用した索引は再作成されます。) • 参照制約が使用可能状態にある場合は、その制約は定義済み検査保留状態に置かれます。
SQL トリガーの中で参照した列の変更	トリガーは再作成されます。
MQT の中で参照した列の変更	MQT が再作成されて新規属性が組み込まれます。

例

例 *1:* 1 文字の長さの RATING という名前の新しい列を、DEPARTMENT 表に追加します。

```
ALTER TABLE DEPARTMENT
  ADD RATING CHAR
```

例 *2:* PICTURE_THUMBNAIL という名前の新しい列を EMPLOYEE 表に追加します。 PICTURE_THUMBNAIL を最大 1K 文字の長さの BLOB 列として作成します。

```
ALTER TABLE EMPLOYEE
  ADD PICTURE_THUMBNAIL  BLOB(1K)
```

例 *3:* 次の列を持つ新たな表 EQUIPMENT が作成されていると想定します。

EQUIP_NO
INT

EQUIP_DESC
VARCHAR(50)

LOCATION
VARCHAR(50)

EQUIP_OWNER
CHAR(3)

表 EQUIPMENT に参照制約を追加して、所有者 (EQUIP_OWNER) が、表 DEPARTMENT に存在する部門番号 (DEPTNO) でなければならないようにします。ある部門が表 DEPARTMENT から除去された場合は、その部門が所有していた備品すべての所有者 (EQUIP_OWNER) の値は、所有者未定 (または NULL) に設定される必要があります。制約に、名前 DEPTQUIP を指定しています。

```
ALTER TABLE EQUIPMENT
  FOREIGN KEY DEPTQUIP (EQUIP_OWNER)
    REFERENCES DEPARTMENT
    ON DELETE SET NULL
```

列 EQUIP_OWNER のデフォルト値を 'ABC' に変更します。

```
ALTER TABLE EQUIPMENT
  ALTER COLUMN EQUIP_OWNER
  SET DEFAULT 'ABC'
```

列 LOCATION を除去します。ビュー、索引、または制約がその列に対して構築されている場合は、それらもすべて除去します。

```
ALTER TABLE EQUIPMENT
  DROP COLUMN LOCATION CASCADE
```

SUPPLIER と呼ばれる新しい列が追加され、LOCATION と呼ばれる既存の列が除去され、新しい列 SUPPLIER に対する固有制約が追加され、基本キーが既存の列 EQUIP_NO に対して構築されるように、表を変更します。

```
ALTER TABLE EQUIPMENT
  ADD COLUMN SUPPLIER INT
  DROP COLUMN LOCATION
  ADD UNIQUE SUPPLIER
  ADD PRIMARY KEY EQUIP_NO
```

列 EQUIP_DESC が可変長列であることに注意してください。25 という長さが割り振られている場合、次の ALTER TABLE ステートメントはその割り振られた長さを変更しません。

```
ALTER TABLE EQUIPMENT
  ALTER COLUMN EQUIP_DESC
  SET DATA TYPE VARCHAR(60)
```

例 *4:* EMPLOYEE 表を変更します。各従業員の給与と歩合の合計が $30,000 を超えていなければならない、という定義済みの REVENUE という名前の表検査制約を追加します。

```
ALTER TABLE EMPLOYEE
  ADD CONSTRAINT REVENUE
  CHECK (SALARY + COMM > 30000)
```

例 *5:* EMPLOYEE 表を変更します。前に定義した制約 REVENUE を除去します。

```
ALTER TABLE EMPLOYEE
  DROP CONSTRAINT REVENUE
```

例 *6:* EMPLOYEE 表を変更します。列 PHONENO を変更して、電話番号として最大 20 文字まで受け入れます。

```
ALTER TABLE EMPLOYEE
  ALTER COLUMN PHONENO SET DATA TYPE VARCHAR (20)
```

例 *7:* 基本表 TRANSCOUNT をマテリアライズ照会表に変更します。選択ステートメント の結果は、既存の表内の列と一致する列のセット (同じ列数および互換性のある属性) を提供する必要があります。

```
ALTER TABLE TRANSCOUNT
  ADD MATERIALIZED QUERY
  (SELECT  ACCTID, LOCID, YEAR, COUNT(*) AS CNT
     FROM TRANS
     GROUP BY ACCTID, LOCID, YEAR )
  DATA INITIALLY DEFERRED
  REFRESH DEFERRED
  MAINTAINED BY USER
```

例 *8:* バージョン管理を可能にするように CUSTOMER 表を変更します。

まず、バージョン管理に必要な列と期間定義を、行変更の ID 追跡に使用される 2 つの列とともに追加します。

```
ALTER TABLE CUSTOMER
  ADD COLUMN SYSTEM_START TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN
  ADD COLUMN SYSTEM_END TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END
  ADD COLUMN TRANSACTION_ID TIMESTAMP(12) GENERATED ALWAYS AS TRANSACTION START ID
  ADD COLUMN AUDIT_TYPE_CHANGE CHAR(1) GENERATED ALWAYS AS (DATA CHANGE OPERATION)
  ADD COLUMN AUDIT_USER VARCHAR(128) GENERATED ALWAYS AS (SESSION_USER)
  ADD PERIOD FOR SYSTEM_TIME (SYSTEM_START, SYSTEM_END);
```

次に、履歴表を作成します。

```
CREATE TABLE CUSTOMER_HISTORY LIKE CUSTOMER;
```

最後に、表のバージョン管理関係を定義します。

```
ALTER TABLE CUSTOMER ADD VERSIONING USE HISTORY TABLE CUSTOMER_HISTORY
                       ON DELETE ADD EXTRA ROW;
```

列アクセス制御の例

例 *1:* SELECT DISTINCT ステートメントは、CUSTOMER 表にあるデータに基づいて、SALARY 値が 100,000 の行を 1 つ戻します。SALARY 値をマスクするために、列マスク SALARY_MASK が作成されます。CUSTOMER 表の列アクセス制御が活動化されると、SALARY 列に列マスクが適用されます。「MGR」権限 ID を持つユーザーが SELECT DISTINCT ステートメントを発行します。SELECT DISTINCT ステートメントは、重複の除去は SALARY 列のマスクされない値に基づいているため、依然として 1 つの行を戻しますが、その行で戻される値は、マスクされた SALARY 値 (125,000 か 110,000 のいずれか) に基づきます。

CUSTOMER 表は、以下のようになっています。

SALARY	COMMISSION	EMPID
100,000	25,000	123456
100,000	10,000	654321

```
CREATE MASK SALARY_MASK ON CUSTOMER
   FOR COLUMN SALARY RETURN
      CASE WHEN(SESSION_USER = 'MGR')
              THEN SALARY + COMMISSION
           ELSE SALARY
      END
   ENABLE;

COMMIT;

ALTER TABLE CUSTOMER
   ACTIVATE COLUMN ACCESS CONTROL;

COMMIT;

SELECT DISTINCT SALARY FROM CUSTOMER;
```

例 *2:* SELECT DISTINCT ステートメントは、T1 表および T2 表のデータに基づき、COALESCE 関数を使用して T1.C1 値が 1 の 1 つの行を戻します。T1.C1 の値をマスクするために列マスク C1_MASK が作成されます。表 T1 の列アクセス制御が活動化されると、表 T1 の列 C1 に列マスクが適用されます。「EMP」権限 ID を持つユーザーが SELECT DISTINCT ステートメントを発行します。SELECT DISTINCT ステートメントは、重複の除去は COALESCE 関数からの T1.C1 のマスクされない値に基づいているため、依然として 1 つの行を戻しますが、この行で戻される値は、COALESCE 関数からの T1.C1 のマスクされた値に基づきます。戻される値は、2 または 3 のいずれかです。

```
INSERT INTO T1(C1) VALUES(1);
INSERT INTO T1(C1) VALUES(1);

INSERT INTO T2(C1) VALUES(2);
INSERT INTO T2(C1) VALUES(3);

CREATE MASK C1_MASK ON T1
   FOR COLUMN C1 RETURN
      CASE WHEN(SESSION_USER = 'EMP')
         THEN NULL
         ELSE C1
      END
   ENABLE;

COMMIT;

ALTER TABLE T1
   ACTIVATE COLUMN ACCESS CONTROL;

COMMIT;

SELECT DISTINCT COALESCE(T1.C1, T2.C1) FROM T1, T2;
```

例 *3:* CUSTOMER 表のデータに基づくと、カリフォルニア (CA) とイリノイ (IL) 州の最大収入は同じで、50,000 です。このため、SELECT DISTINCT ステートメントは 1 つの行を戻します。収入値をマスクするために列マスク INCOME_MASK が作成されます。CUSTOMER 表の列アクセス制御が活動化されると、INCOME 列に列マスクが適用され、MAX 集約関数が評価されます。ただし、列マスク INCOME_MASK は、イリノイ州では収入値 0 を 100,000 としてマスクします。このため、イリノイ州では最大収入は 100,000 になりますが、カリフォルニア州の最大収入は 50,000 のままです。SELECT DISTINCT ステートメントの述部で X.B が使用されています。このため、MAX(INCOME) 関数のオリジナルの INCOME 値とオリジナルの結果を保存しておく必要があります。したがって、SELECT

DISTINCT ステートメントは依然として 1 つの行を戻しますが、この行の値は deterministic でない場合があります。つまり、この値は 'CA' 行からの 50,000、または 'IL' 行からの 100,000 である場合があります。

CUSTOMER 表は、以下のようになっています。

STATE	INCOME
CA	40,000
CA	50,000
IL	0
IL	10,000
IL	50,000

```
CREATE MASK INCOME_MASK ON CUSTOMER
   FOR COLUMN INCOME RETURN
      CASE WHEN(INCOME = 0)
         THEN 100000
         ELSE INCOME
      END
   ENABLE;

COMMIT;

ALTER TABLE CUSTOMER
   ACTIVATE COLUMN ACCESS CONTROL;

COMMIT;

SELECT DISTINCT B FROM
   (SELECT STATE, MAX(INCOME) FROM CUSTOMER
      GROUP BY STATE)
      X(A, B)
   WHERE B > 10000;
```

例 *4:* 式 INCOME + RAND() は、RAND 関数が非 deterministic 関数であるため、非 deterministic です。SELECT DISTINCT ステートメントは、CUSTOMER 表のデータに基づいて 2 つの特殊行を戻す可能性が高くなります。しかし、1 つの行のみを戻すことがあります。収入値をマスクするために列マスク INCOME_MASK が作成されます。CUSTOMER 表の列アクセス制御が活動化されると、INCOME 列に列マスクが適用されます。これによって、両方の行のマスク値が同じになります。RAND 関数は非 deterministic 関数であるため、SELECT DISTINCT ステートメントは、依然として 2 つの特殊行を戻す可能性が高いですが、1 行のみを戻すことがあります。RAND 関数によって生じた不確実性によって SELECT DISTINCT ステートメントの結果が非 deterministic になります。

CUSTOMER 表は、以下のようになっています。

STATE	INCOME
CA	40,000
CA	50,000

```
CREATE MASK INCOME_MASK ON CUSTOMER
   FOR COLUMN INCOME RETURN
      CASE WHEN(INCOME = 40,000)
         THEN 50000
```

```
         ELSE INCOME
      END
   ENABLE;

COMMIT;

ALTER TABLE CUSTOMER
   ACTIVATE COLUMN ACCESS CONTROL;

COMMIT;

SELECT DISTINCT A FROM
   (SELECT INCOME + RAND() FROM CUSTOMER)
      X(A)
   WHERE A > 10000;
```

例 *5:* 市が SJ、SFO、または OKLD の場合に州と市の名前を示す値を戻すために、CUSTOMER 表の STATE 列に対して列マスク STATE_MASK が作成されます。これ以外の場合、市は戻されず、州のみが戻されます。CUSTOMER 表の列アクセス制御が活動化されると、STATE 列を使用して結果をグループ化する SELECT ステートメントが発行されます。ただし、列マスク STATE_MASK で参照される CITY 列はグループ化列でないため、列マスク STATE_MASK がこのステートメントには適切でないことを示すエラーが戻されます。

CUSTOMER 表は、以下のようになっています。

STATE	CITY	INCOME
CA	SJ	40,000
CA	SC	30,000
CA	SB	60,000
CA	SFO	80,000
CA	OKLD	50,000
CA	SJ	70,000
NY	NY	50,000

```
CREATE MASK STATE_MASK ON CUSTOMER
   FOR COLUMN STATE RETURN
      CASE WHEN(CITY = 'SJ')
         THEN CITY||', '||STATE
        WHEN(CITY = 'SFO')
          THEN CITY||', '||STATE
        WHEN(CITY = 'OKLD')
          THEN CITY||', '||STATE
        ELSE STATE
      END
   ENABLE;

COMMIT;

ALTER TABLE CUSTOMER
   ACTIVATE COLUMN ACCESS CONTROL;

COMMIT;

SELECT STATE, AVG(INCOME) FROM CUSTOMER
   GROUP BY STATE
   HAVING STATE = 'CA';
```

ALTER TRIGGER

ALTER TRIGGER ステートメントは、現行サーバーにあるトリガーの記述を変更します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

ENABLE または DISABLE を指定する場合、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

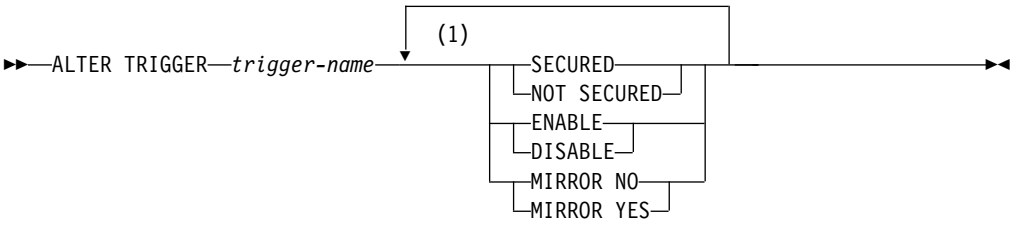
- このステートメントで指定されたトリガーに対して:
  - 物理ファイル変更トリガー (CHGPFTRG) コマンドに対するシステム権限 *USE
  - トリガーが定義されている表またはビューに対する ALTER 特権
  - トリガーが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

SECURED オプションが指定されているか、または、NOT SECURED オプションが指定されていてトリガーが現在セキュアである場合、以下のとおりです。

- このステートメントの許可 ID には、セキュリティー管理者権限 がなければなりません。 21 ページの『管理権限』を参照してください。

SQL 特権に対応するシステム権限については『表またはビューへの権限を検査する際の対応するシステム権限』を参照してください。

構文



注:

1 同じ文節を複数回指定しないでください。

説明

trigger-name

変更するトリガーを識別します。トリガー名 は、現行サーバーに存在しているトリガーを識別していなければなりません。SQL トリガーまたはネイティブ・トリガーのいずれかを指定できます。

SECURED または **NOT SECURED**
トリガーが行および列のアクセス制御に対してセキュアであるどうかを指定します。トリガー表に対して行アクセス制御または列アクセス制御が活動化されている場合、トリガーを SECURED から NOT SECURED に変更するとエラーが戻されます。

SECURED
トリガーが行アクセス制御と列アクセス制御においてセキュアであると見なされることを指定します。

トリガーのサブジェクト表が行アクセス制御または列アクセス制御を使用している場合、そのトリガーには SECURED を指定する必要があります。また、トリガーがビューに対して作成され、そのビュー定義内の 1 つ以上の基礎表が行アクセス制御または列アクセス制御を使用している場合も、そのトリガーに SECURED を指定する必要があります。

NOT SECURED
トリガーが行アクセス制御と列アクセス制御においてセキュアでないと見なされることを指定します。

NOT SECURED は、そのサブジェクト表が行アクセス制御または列アクセス制御を使用しているトリガーに指定してはなりません。また、NOT SECURED は、ビュー定義にある 1 つ以上の基礎表が行アクセス制御または列アクセス制御を使用しているビューに対して作成されたトリガーにも指定してはなりません。

ENABLE または **DISABLE**
トリガーをどの状態に変更するのかを指定します。

ENABLE
トリガーは該当するデータ変更操作中に起動されます。

DISABLE
トリガーは該当するデータ変更操作中に起動されません。

MIRROR NO または **MIRROR YES**
ミラー保護環境でトリガーを起動する場所を指定します。ミラーリングがアクティブではない場合、このオプションは無視されます。

MIRROR NO
ミラー保護環境で、トリガーはソース・ノードでのみ起動されます。

MIRROR YES
ミラー保護環境で、トリガーは、ミラー保護されたペアの両ノードで起動されます。

MIRROR YES は、INSTEAD OF トリガーや、MODE DB2SQL として定義されたトリガーには使用できません。

注

NOT SECURED から **SECURED** へのトリガーの変更**:** ALTER TRIGGER ステートメントが実行された後、トリガーはセキュアであると見なされます。Db2 は SECURED 属性を、トリガー本体のすべてのアクティビティーに対する監査手順をユーザーが確立したことを宣言するアサーションとして扱います。セキュア・トリガーがユーザー定義関数を参照する場合、Db2 では妥当性検査を実行せずに、この

ようなユーザー定義関数がセキュアであると想定します。このような関数が機密データにアクセスする可能性がある場合、セキュリティー管理者権限を持つユーザーは、それらの関数がそのデータにアクセスすることを許可されていること、それらの関数のための監査手順が整っていること、および、後続のすべての ALTER FUNCTION ステートメントがこの監査手順によってレビューされることを確認する必要があります。

遷移変数値および行アクセス制御と列アクセス制御**:** 遷移変数および遷移表には行アクセス制御と列アクセス制御は適用されません。トリガー表に行アクセス制御または列アクセス制御が適用されている場合、遷移変数および遷移表の初期値には行の許可および列マスクは適用されません。トリガー表に適用されている行アクセス制御および列アクセス制御は、トリガー本体において参照されている遷移変数および遷移表に対して、またはトリガー本体内で呼び出されるユーザー定義関数に引数として渡される遷移変数および遷移表に対しては無視されます。SQL ステートメントが遷移変数または遷移表の機密データにアクセスすることに関してセキュリティー上の問題がないことを確実にするため、SECURED オプションを使用してトリガーを作成してください。トリガーがセキュアでない場合、トリガー表に対して行アクセス制御および列アクセス制御を実施することはできません。

例

例 *1:* トリガー TRIGGER1 の定義をセキュアに変更:

```
ALTER TRIGGER TRIGGER1
  SECURED
```

例 *2:* トリガー TRIGGER1 の定義を非セキュアに変更:

```
ALTER TRIGGER TRIGGER1
  NOT SECURED
```

ASSOCIATE LOCATORS

ASSOCIATE LOCATORS ステートメントは、プロシージャーが戻すそれぞれの結果セットについての結果セット・ロケーター変数を入手します。

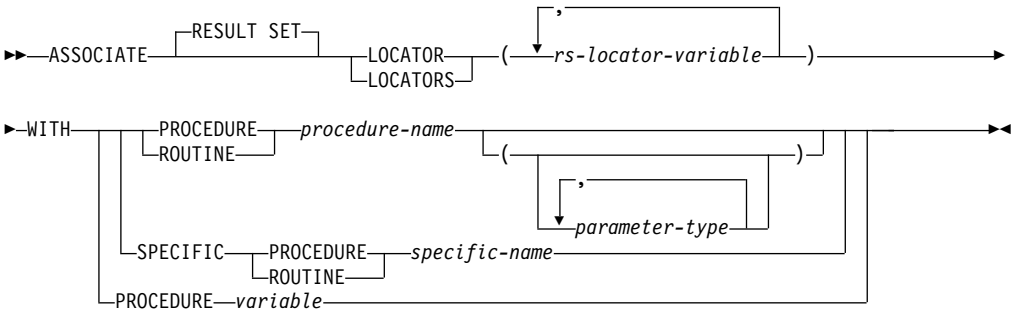
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができます。これは、動的に準備できる実行可能ステートメントです。これを対話式に発行することはできません。REXX で指定してはなりません。

権限

権限は不要です。

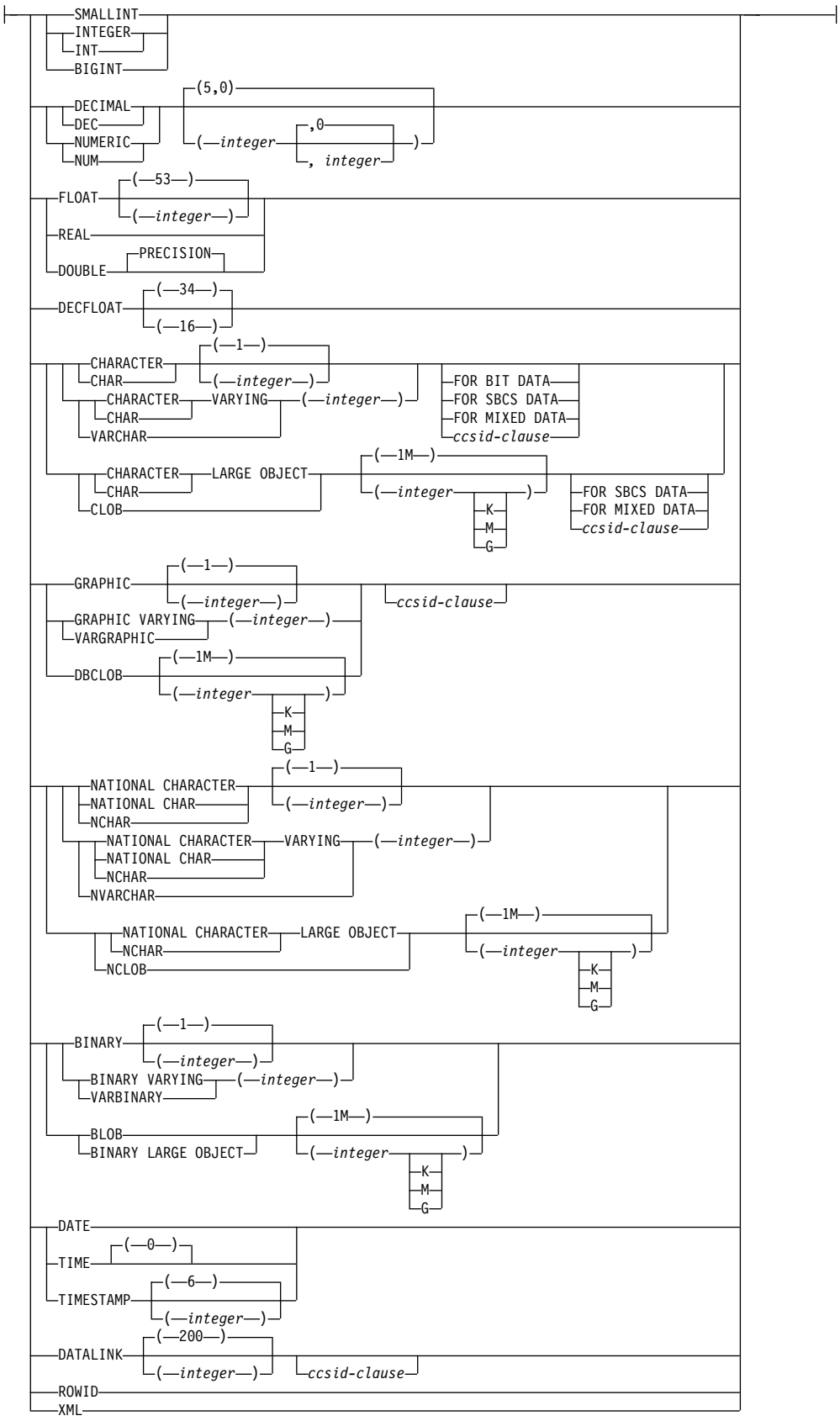
構文



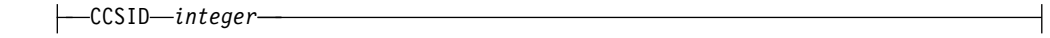
parameter-type:

data-type:

built-in-type:


SMALLINT
INTEGER
INT
BIGINT
DECIMAL
DEC
NUMERIC
NUM
(5,0)
(integer
,0
, integer
)
FLOAT
(53)
(integer)
REAL
DOUBLE
PRECISION
DECFLOAT
(34)
(16)
CHARACTER
CHAR
(1)
(integer)
CHARACTER VARYING
CHAR VARYING
VARCHAR
FOR BIT DATA
FOR SBCS DATA
FOR MIXED DATA
ccsid-clause
CHARACTER LARGE OBJECT
CHAR LARGE OBJECT
CLOB
(1M)
K
M
G
GRAPHIC
GRAPHIC VARYING
VARGRAPHIC
DBCLOB
NATIONAL CHARACTER
NATIONAL CHAR
NCHAR
NATIONAL CHARACTER VARYING
NATIONAL CHAR VARYING
NCHAR VARYING
NVARCHAR
NATIONAL CHARACTER LARGE OBJECT
NCHAR LARGE OBJECT
NCLOB
BINARY
BINARY VARYING
VARBINARY
BLOB
BINARY LARGE OBJECT
DATE
TIME
(0)
TIMESTAMP
(6)
DATALINK
(200)
ROWID
XML

ccsid-clause:



説明

rs-locator-variable
結果セット・ロケーター変数を宣言するための規則に従い、すでに宣言された規則セット・ロケーター変数を指定します。

WITH PROCEDURE *procedure-name* または ***variable***
1 つ以上の結果セットを戻したプロシージャーを識別します。ASSOCIATE LOCATORS ステートメントを実行する際のプロシージャー名は、リクエスターがすでに SQL CALL ステートメントを使用して呼び出したプロシージャーを示している必要があります。

PROCEDURE または **SPECIFIC PROCEDURE**
プロシージャーを識別します。このプロシージャー名 は、現行サーバーに存在しているプロシージャーを識別していなければなりません。

PROCEDURE *procedure-name*
プロシージャーを名前によって識別します。プロシージャー名 は、ただ 1 つのプロシージャーを識別していなければなりません。このプロシージャーには、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前のプロシージャーが複数ある場合、エラーが戻されます。

PROCEDURE *procedure-name (parameter-type, ...)*
プロシージャーを一意的に識別するプロシージャー・シグニチャーによって、プロシージャーを識別します。 *procedure-name (parameter-type,...)* では、指定されたプロシージャー・シグニチャーを持つプロシージャーを識別する必要があります。指定されたパラメーターは、プロシージャーの作成時に指定された、対応する位置にあるデータ・タイプと一致していなければなりません。ラベルを付ける対象のプロシージャー・インスタンスを識別するには、データ・タイプの数とデータ・タイプの論理連結を使用します。データ・タイプの同義語は、一致として扱われます。デフォルトがあるパラメーターは、このシグニチャーに含まれていなければなりません。

プロシージャー名 () を指定する場合、識別されるプロシージャーにパラメーターを使用することはできません。

procedure-name
プロシージャーの名前を識別します。

(parameter-type, ...)
プロシージャーのパラメーターを識別します。

非修飾の特殊タイプ名または配列タイプ名を指定する場合、データベース・マネージャーはその特殊タイプまたは配列タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、以下のいずれかを使用します。

- 空の括弧は、データベース・マネージャーがデータ・タイプの一致の判別に際して属性を無視することを示します。例えば、DEC() は、DEC(7,2) のデータ・タイプで定義されたプロシージャーのパラメーターに一致するものとみなされます。ただし、FLOAT に空の括弧を指定することはできません。これは、そのパラメーター値が特定のデータ・タイプ (REAL または DOUBLE) を示しているからです。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を指定する場合、その値は、 CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプが FLOAT の場合、突き合わせはデータ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度は指定された値に厳密に一致している必要はありません。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。暗黙の長さは、CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

FOR DATA 文節または CCSID 文節の指定はオプショナルです。いずれの文節も指定しないと、データ・タイプが一致するかどうかを判定する場合に、データベース・マネージャーが属性を無視することを示します。どちらか一方の文節を指定する場合は、CREATE PROCEDURE ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

AS LOCATOR
プロシージャーが、このパラメーターのロケーターを受け取るように定義されることを示します。 AS LOCATOR を指定する場合は、データ・タイプは LOB または XML、あるいは LOB または XML に基づく特殊タイプでなければなりません。 AS LOCATOR を指定した場合、FOR SBCS DATA または FOR MIXED DATA を指定してはなりません。

SPECIFIC PROCEDURE *specific-name*
プロシージャーを特定名によって識別します。特定名 は、現行サーバーに存在している特定のプロシージャーを識別していなければなりません。

variable
プロシージャー名または特定名を含んでいる変数を指定します。*variable* を指定した場合:

- その変数は、文字ストリング変数またはユニコード・グラフィック・ストリング変数でなければなりません。この変数は、グローバル変数にすることはできません。
- 標識変数を伴っていてはなりません。
- 変数内に含まれる名前は左寄せでなければならず、その長さが変数の長さより短い場合は、右側にブランクを埋め込まなければなりません。

- 名前は、区切り文字付きの名前でない限り、大文字でなければなりません。

この名前の 1 つのプロシージャーのみが CALL ステートメントを使用して呼び出された場合、変数はプロシージャー名として使用されます。この名前の複数のプロシージャーが呼び出された場合、変数は特定名として使用されます。

注

ロケーター値の割り当て。SET RESULT SETS ステートメントがプロシージャーで実行される場合は、その SET RESULT SETS ステートメントが結果セットを識別します。ロケーター値は、SET RESULT SETS ステートメントで指定した順序で、記述子域の項目または SQLDA の SQLVAR 項目に割り当てられます。プロシージャーで SET RESULT SETS ステートメントが実行されなかった場合は、実行時に関連カーソルがオープンした順序でロケーター値がロケーター変数に割り当てられます。ロケーター値がロケーター変数に割り当てられる順序は、DESCRIBE PROCEDURE ステートメントの結果としてロケーター値が SQL 記述子域または SQLDA 内の項目に配置される順序と同じです。

ロケーター値は、呼び出し側のアプリケーションに制御が戻されるとクローズされるカーソルには提供されません。カーソルがクローズされ、呼び出し側のアプリケーションに戻る前に再度オープンされた場合は、プロシージャー結果セットにロケーター値が戻される順序を決めるために、カーソルに対して最新に実行された OPEN CURSOR ステートメントが使用されます。例えば、プロシージャー P1 が 3 つのカーソル A、B、C をオープンして、カーソル B をクローズし、呼び出し側のアプリケーションに戻る前にカーソル B に対して別の OPEN CURSOR ステートメントを実行したとします。次の ASSOCIATE LOCATORS ステートメントに割り当てられるロケーター値の順序は、A、C、B になります。

ASSOCIATE RESULT SET LOCATORS (:loc1, :loc2, :loc3) WITH PROCEDURE P1;

複数のロケーターを結果セットに関連付けることができます。同じプロシージャーに対して、異なる結果セット・ロケーター変数を指定して複数の ASSOCIATE LOCATORS ステートメントを発行することにより、それぞれの結果セットに複数のロケーターを関連付けることができます。

- ASSOCIATE LOCATORS ステートメントに指定された結果セット・ロケーター変数の数が、プロシージャーから戻される結果セットの数より少ない場合、ステートメントに指定されたロケーター変数すべてに値が割り当てられ、警告が出されます。例えば、プロシージャー P1 が存在し、4 つの結果セットを戻すとします。次の ASSOCIATE LOCATORS ステートメントはそれぞれ、最初の結果セットに関する情報とともに、すべての結果セットに関する情報を得るために十分なロケーターがなかったことを示す警告を戻します。

  CALL P1;

  ASSOCIATE RESULT SET LOCATORS (:loc1) WITH PROCEDURE P1; -> loc1 is assigned a value for first result set, and a warning is returned

ASSOCIATE RESULT SET LOCATORS (:loc2) WITH PROCEDURE P1; -> loc2 is assigned a value for first result set, and a warning is returned

ASSOCIATE RESULT SET LOCATORS (:loc3) WITH PROCEDURE P1; -> loc3 is assigned a value for first result set, and a warning is returned

ASSOCIATE RESULT SET LOCATORS (:loc4) WITH PROCEDURE P1; -> loc4 is assigned a value for first result set, and a warning is returned

- ASSOCIATE LOCATORS ステートメントにリストされている結果セット・ロケーター変数の数が、プロシージャーによって戻されるロケーターの数より多い場合は、余分のロケーター変数に値 0 が割り当てられます。

同じプロシージャーに対する複数の呼び出し: 同じプログラムから同じプロシージャーに対して複数の呼び出しを実行すると、先行の呼び出しの結果セットが失われます。ただし、プロシージャーに対する後続の CALL の前に ASSOCIATE LOCATOR ステートメントを実行する場合は、例外です。ASSOCIATE LOCATORS ステートメントでは、最新の CALL を参照します。

```
EXEC SQL CALL P1; /* Returns 2 result sets */

EXEC SQL CALL P1; /* Returns 2 result sets, result sets from first invocation are closed */

EXEC SQL ASSOCIATE RESULT SET LOCATORS (:a, :b) WITH PROCEDURE P1; /* Refers to second call */

EXEC SQL CALL P1; /* Returns 2 result sets */

EXEC SQL ASSOCIATE RESULT SET LOCATORS (:c, :d) WITH PROCEDURE P1; /* Refers to third call */

/* The following statements process the result sets from the second call */
EXEC SQL ALLOCATE C1 CURSOR FOR RESULT SET :a;
EXEC SQL ALLOCATE C2 CURSOR FOR RESULT SET :b;
EXEC SQL FETCH C1 INTO :h1;
EXEC SQL CLOSE C1;
EXEC SQL FETCH C2 INTO :h2;
EXEC SQL CLOSE C2;

/* The following statements process the result sets from the third call */
EXEC SQL ALLOCATE C3 CURSOR FOR RESULT SET :c;
EXEC SQL ALLOCATE C4 CURSOR FOR RESULT SET :d;
EXEC SQL FETCH C3 INTO :h1;
EXEC SQL CLOSE C3;
EXEC SQL FETCH C4 INTO :h2;
EXEC SQL CLOSE C4;
```

RETURN TO CLIENT プロシージャー: RETURN TO CLIENT プロシージャーの結果セットは、呼び出しスタックの最高位のプロシージャーと関連付けられます。そういった結果セットにロケーターを関連付けるには、呼び出しスタックの最高位のプロシージャーのプロシージャー名を指定する必要があります。

例

3 つの結果セットを返すプロシージャー P1 の結果セット・ロケーターを割り振ります。

```
ASSOCIATE RESULT SET LOCATORS (:loc11, :loc2, :loc3) WITH PROCEDURE P1;
```

BEGIN DECLARE SECTION

BEGIN DECLARE SECTION ステートメントは、SQL 宣言セクションの開始を示します。SQL 宣言セクションには、プログラム内の SQL ステートメントでホスト変数として使用できる有資格ホスト変数の宣言が含まれます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、実行可能ステートメントではありません。 Java、RPG または REXX で指定してはなりません。

権限

権限は不要です。

構文

説明

BEGIN DECLARE SECTION ステートメントを使用して、SQL 宣言セクションの始まりを示します。これは、ホスト言語の規則に従って変数宣言を置ける場所であれば、アプリケーション・プログラム内のどこにでもコーディングすることができます。ホスト構造体の宣言の内部に、コーディングすることはできません。SQL 宣言セクションは、 1457 ページの『END DECLARE SECTION』で説明されているように END DECLARE SECTION ステートメントで終了します。

BEGIN DECLARE SECTION と END DECLARE SECTION ステートメントは、対にして使用しなければなりません。また、これらのステートメントをネストすることはできません。

DECLARE VARIABLE および INCLUDE ステートメント以外の SQL ステートメントは、宣言セクション内にコーディングしてはなりません。

プログラムに SQL 宣言セクションの指定がある場合は、その SQL 宣言セクションで宣言されている変数だけがホスト変数として使用できます。プログラムに SQL 宣言セクションの指定がない場合は、そのプログラムの中の変数はすべてがホスト変数として使用できます。

RPG および REXX 以外のホスト言語では、そのソース・プログラムが SQL の IBM SQL 規格に準拠するように、SQL 宣言セクションを指定する必要があります。SQL 宣言セクションは、C++ のすべてのホスト変数に必須です。 SQL 宣言セクションは、変数に対する最初の参照よりも前にコーディングされている必要があります。Java および RPG では、これらのステートメントを使用せずにホスト変数が宣言され、また REXX ではホスト変数は宣言されません。

SQL 宣言セクションの外側で宣言されている変数の名前は、SQL 宣言セクション内で宣言されている変数と同じ名前であってはなりません。

複数の SQL 宣言セクションを、プログラムに指定することができます。

例

例 *1:* C プログラムで、ホスト変数の hv_smint (SMALLINT)、hv_vchar24 (VARCHAR(24))、および hv_double (DOUBLE) を定義します。

```
EXEC SQL  BEGIN DECLARE SECTION;
  static short                          hv_smint;
  static struct {
           short hv_vchar24_len;
           char  hv_vchar24_value[24];
         }                              hv_vchar24;
  static double                         hv_double;
EXEC SQL  END DECLARE SECTION;
```

例 *2:* COBOL プログラムで、ホスト変数 HV-SMINT (smallint)、HV-VCHAR24 (varchar(24))、および HV-DEC72 (dec(7,2)) を定義します。

```
WORKING-STORAGE SECTION.
      EXEC SQL BEGIN DECLARE SECTION  END-EXEC.
01  HV-SMINT              PIC S9(4)       BINARY.
01  HV-VCHAR24.
    49 HV-VCHAR24-LENGTH  PIC S9(4)       BINARY.
    49 HV-VCHAR24-VALUE   PIC X(24).
01  HV-DEC72              PIC S9(5)V9(2)  PACKED-DECIMAL.
      EXEC SQL END DECLARE SECTION  END-EXEC.
```

CALL

CALL ステートメントはプロシージャーを呼び出します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 呼び出すプロシージャーが SQL プロシージャーである場合
  - そのプロシージャーに対する EXECUTE 特権。および、
  - SQL プロシージャーが含まれるスキーマに対する USAGE 特権
- 呼び出すプロシージャーが Java 外部プロシージャーである場合
  - Java クラスを含む統合ファイル・システム・ファイルに対する読み取り権限 (*R)
  - 統合ファイル・システム・ファイルを検出するためにアクセスする必要があるすべてのディレクトリーに対する読み取りおよび実行権限 (*RX)
- 呼び出すプロシージャーが REXX 外部プロシージャーである場合
  - そのプロシージャーに関連するソース・ファイルに対する *OBJOPR、*READ、および *EXECUTE システム権限
  - ソース・ファイルが含まれるスキーマに対する USAGE 特権、および
  - CL コマンドに対する *USE システム権限
- 呼び出すプロシージャーが外部プロシージャーではあるが、REXX または Java 外部プロシージャーではない場合
  - そのプロシージャーに関連するプログラムまたはサービス・プログラムに対する *EXECUTE システム権限、および
  - そのプロシージャーに関連するプログラムまたはサービス・プログラムが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

グローバル変数を IN パラメーターまたは INOUT パラメーターとして参照する場合は、ステートメントの権限 ID が保持する特権に、以下の少なくとも 1 つが含まれていなければなりません。

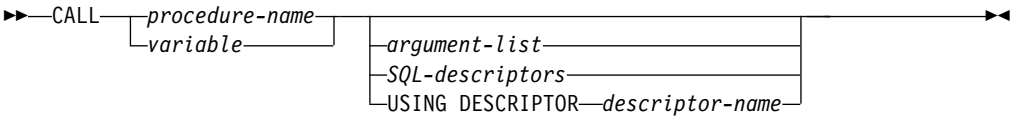
- グローバル変数に対する READ 特権
- データベース管理者権限

グローバル変数を OUT パラメーターまたは INOUT パラメーターとして参照する場合は、ステートメントの権限 ID が保持する特権に、以下の少なくとも 1 つが含まれていなければなりません。

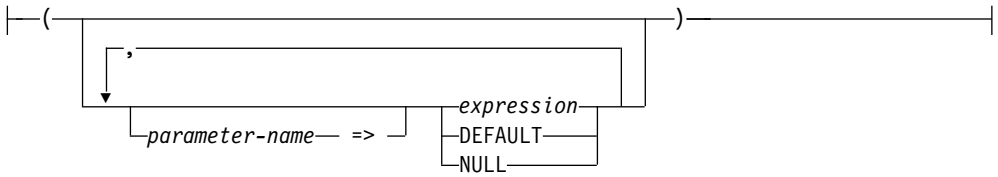
- グローバル変数に対する WRITE 特権
- データベース管理者権限

SQL 特権に対応するシステム権限については、『関数またはプロシージャーへの権限を検査する際の対応するシステム権限』を参照してください。

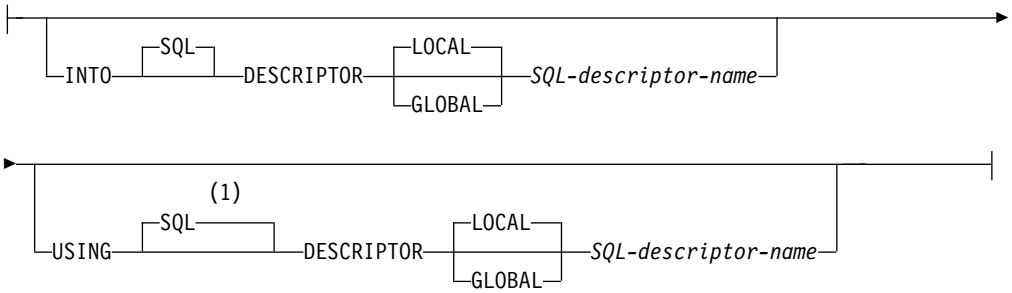
構文



argument-list:



SQL-descriptors:



注:

1 SQL 記述子が USING 文節で指定されて INTO 文節が指定されない場合、USING DESCRIPTOR は許可されないため USING SQL DESCRIPTOR を指定しなければなりません。

説明

procedure-name または *variable*

指定した *procedure-name*、または *variable* に含まれているプロシージャー名によって、呼び出したいプロシージャーを識別します。識別されたプロシージャーは、現行サーバーに存在していなければなりません。

変数 を指定する場合、

- その変数は、文字ストリング変数またはユニコード・グラフィック・ストリングでなければなりません。この変数は、グローバル変数にすることはできません。
- 標識変数を伴っていてはなりません。
- 変数内に含まれるプロシージャー名は左寄せでなければならず、その長さが変数の長さより短い場合は、右側にブランクを埋め込まなければなりません。

- プロシージャーの名前は、区切り文字付きの名前でない限り、大文字でなければなりません。

プロシージャー名は、修飾されなかった場合、パラメーターのパスと番号に基づいて暗黙的に修飾されます。詳しくは、 72 ページの『非修飾オブジェクト名の修飾』を参照してください。

プロシージャー名が、DECLARE PROCEDURE ステートメントによって定義されたプロシージャーを識別し、かつ、現行サーバーが Db2 for i 製品である場合は、次のようになります。

- その DECLARE PROCEDURE ステートメントによって、外部プログラム、言語、および呼び出し規則の名前が決まります。
- そのプロシージャーのパラメーターの属性が、DECLARE PROCEDURE ステートメントによって定義されます。

DECLARE PROCEDURE がなく、プロシージャー名が示しているプロシージャーが CREATE PROCEDURE ステートメントで定義されたものであり、現行サーバーが Db2 for i である場合、次のようになります。

- その CREATE PROCEDURE ステートメントによって、外部プログラム、言語、および呼び出し規則の名前が決まります。
- そのプロシージャーのパラメーターの属性が、CREATE PROCEDURE ステートメントによって定義されます。

上記以外の場合、

- 現行サーバーが外部プログラム、言語、および呼び出し規則の名前を決定します。
- 現行サーバーが Db2 for i である場合、
  - 外部プログラム名は、外部プロシージャー名と同じであると想定されます。
  - そのプログラムのプログラム属性情報が認識可能な言語を示している場合には、その言語が使用されます。それ以外の場合は、言語は C であると見なされます。
  - 呼び出し規則は GENERAL と見なされます。
- アプリケーション・リクエスターは、変数またはパラメーター・マーカーであるパラメーターはすべて INOUT であると想定します。変数以外のパラメーターは、すべて IN であると見なされます。
- パラメーターの実際の属性は、現行サーバーによって決定されます。

  現行サーバーが Db2 for i である場合、パラメーターの属性は、CALL ステートメント上に指定された引数の属性と同じになります。[84]

argument-list

パラメーターとしてプロシージャーに渡される値のリストを識別します。 n 番目の名前付きでない引数が、プロシージャー内の n 番目のパラメーターに対応します。

(CREATE PROCEDURE または DECLARE PROCEDURE ステートメントを使用して) OUT として定義された各パラメーターは、変数として指定する必要

84. 10 進定数の場合、先行ゼロは、引数の属性を決定する際に有効になります。通常は、先行ゼロは有効数字ではありません。

があります。OUT パラメーターにはデフォルトを指定できません。INOUT パラメーターにデフォルトが使用される場合、そのデフォルト式は、プロシージャーへの入力用のパラメーターを初期化するために使用されます。プロシージャーの終了時には、このパラメーターに対して値は何も戻されません。

プロシージャーが呼び出されるときには、デフォルト値を持つよう定義されていないすべてのパラメーターに対して引数が指定される必要があります。名前付き構文を使用して、ある引数がパラメーターに代入される場合、それに続く引数もすべて名前付き構文を使用して代入される必要があります 。

日付、時刻、またはタイム・スタンプの特殊レジスター値への引数リスト内での参照では、デフォルト式に 1 つのクロック読み取りが使用され、明示的引数内での参照には別のクロック読み取りが使用されます。

いずれかの引数が配列の場合、他のすべての引数はリテラル、特殊レジスター、NULL、または DEFAULT でなければなりません。

アプリケーション・リクエスターは、変数であるパラメーターがすべて Java 以外の INOUT パラメーターであると想定します。ここでは、モードが変数参照で明示的に指定されている場合を除いて、変数であるパラメーターはすべて IN であると想定されています。変数以外のパラメーターは、すべて入力パラメーターであると見なされます。パラメーターの実際の属性は、現行サーバーによって決定されます。

parameter-name
引数値が割り当てられるパラメーターの名前。この名前は、プロシージャーに定義されているパラメーター名と一致しなければなりません。名前付き引数は、引数リスト内に指定される順序とは無関係に、同じ名前を持つパラメーターに対応します。引数が、名前によってパラメーターに割り当てられる場合、それ以降の引数もすべて名前によって割り当てられる必要があります。

名前付き引数は、一回だけ (暗黙的または明示的に) 指定される必要があります。

名前付き引数は、CREATE PROCEDURE ステートメントを使用して定義されたプロシージャーの呼び出しでのみ許可されます。

expression
集約関数または列名を含まない、 196 ページの『式』で説明されているタイプの *expression*。拡張標識変数が使用可能である場合、DEFAULT および UNASSIGNED の拡張標識変数値は、その式に対して使用してはなりません。

DEFAULT
CREATE PROCEDURE ステートメントに定義されたデフォルトがプロシージャーへの引数として使用されることを指定します。パラメーターのデフォルトが定義されていない場合は、NULL 値が使用されます。

NULL
NULL 値をプロシージャーへの引数として指定します。

SQL-descriptors
SQL 記述子が CALL で指定される場合、IN および INOUT パラメーターを持つプロシージャーでは SQL 記述子を USING 文節で指定する必要があり、

OUT または INOUT パラメーターを持つプロシージャーでは SQL 記述子を INTO 文節で指定する必要があります。プロシージャーのすべてのパラメーターが INOUT パラメーターである場合は、両方の文節に同じ記述子を使用できます。詳しくは、『CALL での複数の SQL 記述子』を参照してください。

INTO

CALL ステートメントとともに使用される出力変数の有効な記述を含む SQL 記述子を識別します。 CALL ステートメントを実行する前に、ALLOCATE DESCRIPTOR ステートメントを使用して記述子を割り振らなければなりません。記述子ヘッダーの COUNT フィールドは、プロシージャーの OUT および INOUT パラメーターの数を反映して設定する必要があります。ステートメントの処理時に使用される変数について、TYPE のほか、DATETIME_INTERVAL_CODE、LENGTH、DB2_CCSID、PRECISION、SCALE の該当する項目情報を設定する必要があります。

LOCAL

記述子の名前の有効範囲はプログラム呼び出しのローカルであることを指定します。情報は、このローカル有効範囲で既知の記述子から戻されます。

GLOBAL

記述子の名前の有効範囲は SQL セッション全体であることを指定します。情報は、同じデータベース接続を使用して実行するどのプログラムにも既知の記述子から戻されます。

SQL-descriptor-name

SQL 記述子の名前を指定します。名前は、指定の有効範囲を持つ既存の記述子を識別するものでなければなりません。

SQL 記述子に入る情報の説明については、 1476 ページの『GET DESCRIPTOR』を参照してください。

USING

CALL ステートメントとともに使用される入力変数の有効な記述を含む SQL 記述子を識別します。 CALL ステートメントを実行する前に、ALLOCATE DESCRIPTOR ステートメントを使用して記述子を割り振らなければなりません。記述子ヘッダーの COUNT フィールドは、プロシージャーの IN および INOUT パラメーターの数を反映して設定する必要があります。ステートメントの処理時に使用される変数について、TYPE のほか、
DATETIME_INTERVAL_CODE、LENGTH、DB2_CCSID、PRECISION、SCALE の該当する項目情報を設定する必要があります。入力変数の DATA 項目と INDICATOR 項目 (NULL が使用される場合) を設定する必要があります。

LOCAL

記述子の名前の有効範囲はプログラム呼び出しのローカルであることを指定します。

GLOBAL

記述子の名前の有効範囲は SQL セッション全体であることを指定します。

SQL-descriptor-name
SQL 記述子の名前を指定します。名前は、指定の有効範囲を持つ既存の記述子を識別するものでなければなりません。

SQL 記述子内の情報の説明については、 1688 ページの『SET DESCRIPTOR』を参照してください。

USING DESCRIPTOR *descriptor-name*
SQLDA を識別します。この SQLDA には、変数の有効な記述が入っていなければなりません。

CALL ステートメントの処理に先立って、SQLDA の以下のフィールドをセットしておく必要があります。(REXX の場合は、規則が異なります。詳しくは、「組み込み SQL プログラミング」トピック集を参照してください。)

- SQLN (SQLDA に用意する SQLVAR のオカレンスの数を示します。)
- SQLDABC (SQLDA 用に割り振る記憶域のバイト数を示します。)
- SQLD (ステートメントを処理するときに、SQLDA で使用する変数の個数を指示します。)
- SQLVAR の各オカレンス (変数の属性を指示します。)

SQLDA の記憶域は、SQLVAR のオカレンスをすべて収容するのに十分な大きさで割り振らなければなりません。したがって、SQLDABC の値は、16 + SQLN*(80) よりも大きいか、または等しくなければなりません。ここで、80 は SQLVAR の 1 つのオカレンスの長さです。LOB または特殊タイプが指定された場合には、各パラメーター・マーカーごとに 2 つの SQLVAR 項目が必要であり、SQLN はパラメーター・マーカー数の 2 倍にセットしなければなりません。

SQLD には、ゼロ以上で SQLN 以下の値をセットしなければなりません。この値は、CALL ステートメント内のパラメーター数と同じでなければなりません。 SQLDA によって *n* 番目に記述される変数は、準備済みステートメントの *n* 番目のパラメーター・マーカーに対応します。(SQLDA の説明については、 1877 ページの『付録 D. SQLDA (SQL 記述子域)』を参照してください。)

RPG/400 には、ポインターを設定する機能が用意されていないことに注意してください。 SQLDA はポインターを使用して適切な変数を見つけるので、ユーザーは、RPG/400 アプリケーションの外側でそのようなポインターを設定する必要があります。

Java プログラムでは、CALL ステートメントで USING DESCRIPTOR 文節を使用できません。

注

パラメーターの割り当て**:** CALL ステートメントが実行されると、その各パラメーターの値は、プロシージャーの対応するパラメーターに (ストレージ割り当て規則を使用して) 割り当てられます。デフォルト値を持つように定義されているパラメーター値は、プロシージャーを呼び出すときの引数リストから除外することが可能です。制御は、ホスト言語の呼び出し規則に従って、プロシージャーに渡されます。プロシージャーの実行が完了すると、プロシージャーの各パラメーターの値が (SQL パラメーターのストレージ割り当て規則に基づいて、そうでなければ取得割り

当て規則に基づいて) CALL ステートメントの対応するパラメーターに OUT または INOUTとして割り当てられます。割り当て規則の詳細については、 113 ページの『割り当ておよび比較』を参照してください。

グローバル変数**:** グローバル変数を指定することもできます。INOUT または OUT のパラメーターとしてグローバル変数を使用すると、そのグローバル変数が変更されます。

プロシージャー・シグニチャー**:** プロシージャーは、そのスキーマ、プロシージャー名、およびパラメーター数によって識別されます。これはプロシージャー・シグニチャーと呼ばれ、データベース内でユニークである必要があります。プロシージャーごとにパラメーターの数が違っていれば、 1 つのスキーマに同じ名前のプロシージャーが複数存在しても構いません。

SQL パス**:** プロシージャーは、修飾名 (スキーマおよびプロシージャー名) を参照して呼び出すことができます。修飾名の後に、括弧に囲まれた引数のオプションのリストが続きます。また、スキーマ名を指定せずにプロシージャーを呼び出すことも可能であり、その場合は、同じ数のパラメーターを持つ異なるスキーマのプロシージャーが選択可能になります。この場合、SQL パスを使用してプロシージャー解決を支援します。

プロシージャー解決**:** プロシージャー解決がどのように実行されるのかについて詳しくは、 80 ページの『プロシージャー解決』を参照してください。

プロシージャー内のカーソルと準備済みステートメント**:** 結果セット・カーソルでない、呼び出し先プロシージャー内でオープンされたカーソルは、プロシージャーの終了時にすべてクローズされ、呼び出し先プロシージャー内で準備されたステートメントはすべて破棄されます。プロシージャーの終了時にカーソルをオープンしたままにしておくために、CLOSQLCSR(*ENDACTGRP) を使用することもできます。詳しくは、 1696 ページの『SET OPTION』を参照してください。

プロシージャーの結果セット**:** プロシージャーから結果セットを返す方法は、3 つあります。

- プロシージャー内で SET RESULT SETS ステートメントが実行される場合、プロシージャー内で実行された最後の SET RESULT SETS ステートメントが結果セットを識別します。結果セットは、SET RESULT SETS ステートメントで指定した順序で戻されます。
- SET RESULT SETS ステートメントがプロシージャーで実行されない場合
  - WITH RETURN 文節でカーソルが指定されていない場合、プロシージャーがオープンし、オープン状態のまま戻す各カーソルが、それぞれ 1 つの結果セットを識別します。結果セットは、カーソルがオープンされた順序で戻されます。
  - WITH RETURN 文節でカーソルが指定されている場合、WITH RETURN 文節で定義されたカーソルのうち、プロシージャーがオープンし、オープン状態のまま戻す各カーソルが、それぞれ 1 つの結果セットを識別します。結果セットは、カーソルがオープンされた順序で戻されます。

オープン・カーソルを使用して結果セットが戻される場合、現行カーソル位置から始まる行が戻されます。

カーソル結果セットが SQL 配列タイプを参照している場合、結果セットの処理時に配列参照が原因で正しくない結果になる可能性がある場合はエラーが戻されます。

プロシージャー内のロック**:** 呼び出されたプロシージャーで獲得されたすべてのロックは、作業単位の終了まで保存されます。

プロシージャーからのエラー**:** プロシージャーは、他の SQL ステートメントのように SQLSTATE を使用してエラー (または警告) を戻すことができます。アプリケーションは、プロシージャーの呼び出し時に生じ得る SQLSTATE に留意する必要があります。生じ得る SQLSTATE はプロシージャーをコード化する方法によって異なります。プロシージャーは、プロシージャーの実行時にデータベース・マネージャーで問題が起きた場合、 '38' または '39' で始まる SQLSTATE を戻すこともできます。このため、アプリケーションは、 CALL ステートメントの発行の結果生じる可能性のあるエラー SQLSTATE を処理する準備ができていなければなりません。

CALL ステートメントのネスティング**:** プロシージャーとして実行しているプログラム、ユーザー定義の関数、またはトリガーで、CALL ステートメントを出すことができます。プロシージャー、ユーザー定義の関数、またはトリガーでプロシージャー、ユーザー定義の関数、またはトリガーを呼び出すと、その呼び出しはネストされるものであると見なされます。プロシージャーと関数のネストのレベル数には制限は設けられていませんが、トリガーの場合は、最大 200 レベルだけしかネストできません。

プロシージャーが何らかの照会の結果セットを戻す場合、その結果セットは、プロシージャーの呼び出し元に戻されます。 SQL CALL ステートメントがネストされた場合、その結果セットは、直前のネスト・レベルのプログラムだけに表示されます。例えば、クライアント・プログラムがプロシージャー PROCA を呼び出すと、そのプロシージャーは、プロシージャー PROCB を呼び出します。 PROCA だけが PROCB によって戻された結果セットをアクセスすることができます。クライアント・プログラムは、照会の結果セットをアクセスすることはできません。

SQL や外部プロシージャーが呼び出されると、そのプロシージャーの作成時に定義された SQL データ・アクセスに対して属性が設定されます。それらの属性に使用できる値は、次のとおりです。

```
NONE
CONTAINS
READS
MODIFIES
```

次のような場合に、2 番目のプロシージャーが現行プロシージャーの実行中に呼び出されるとエラーが出されます。

- 呼び出されたプロシージャーに SQL が使用されているが、呼び出しプロシージャーで SQL が許可されていない。
- 呼び出されたプロシージャーが SQL データを読み取っているが、呼び出しプロシージャーで SQL データの読み取りを許可していない。
- 呼び出されたプロシージャーが SQL データを変更したが、呼び出しプロシージャーで SQL データの変更を許可していない。

REXX プロシージャー: 呼び出す外部プロシージャーが REXX である場合、そのプロシージャーは、CREATE PROCEDURE または DECLARE PROCEDURE ステートメントを使用して宣言する必要があります。

変数は、REXX プロシージャー内では CALL ステートメントに使用することはできません。その代わりとして、CALL は、パラメーター・マーカーを使用して PREPARE と EXECUTE のオブジェクトにする必要があります。

CALL での複数の SQL 記述子: CALL で SQL 記述子が指定され、プロシージャーが IN または INOUT パラメーターと OUT または INOUT パラメーターを持つ場合は、2 つの記述子を指定する必要があります。 SQL 記述子に割り振る必要のある変数の数は、SQL 記述子の属性がどのように設定されるかと、パラメーターの各タイプの数によって異なります。

- 入力 SQL 記述子の属性が DESCRIBE INPUT を使用して設定されていて、出力 SQL 記述子の属性が DESCRIBE (OUTPUT) を使用して設定されている場合、SQL 記述子はプロシージャーを呼び出す前に、現行サーバーでの実際のプロシージャー定義に一致する属性を持つことになります。この場合、出力 SQL 記述子には、OUT および INOUT パラメーターごとに変数が 1 つずつ含まれることになります。同様に、入力 SQL 記述子には、IN および INOUT パラメーターごとに変数が 1 つずつ含まれることになります。

  これが、CALL ステートメントで複数の SQL 記述子を指定する場合の推奨技法です。
- それ以外の場合は、プロシージャーを呼び出す前は現行サーバーでの実際のプロシージャー定義が不明であるため、プロシージャーが呼び出される時点では各パラメーターは INOUT であると見なされます。これは、両方の SQL 記述子を指定する必要があり、各パラメーターが INOUT であると見なされるために両方の SQL 記述子が同数の変数を持つ必要があり、出力 SQL 記述子内の各変数の TYPE、DATETIME_INTERVAL_CODE、LENGTH、DB2_CCSID、PRECISION、SCALE は入力 SQL 記述子内の対応する変数と厳密に同じでなければならないことを意味します。こうしないと、エラーが戻されます。

また、複数の SQL 記述子が指定された場合、入力 SQL 記述子内の INOUT パラメーターに関連した DATA または INDICATOR 項目が、プロシージャーが呼び出されると変更される場合があります。したがって、このような入力 SQL 記述子では、別のステートメントで使用する前に、DATA および INDICATOR 項目を再設定するために SET DESCRIPTOR が必要な場合があります。

例

例 1: プロシージャー PGM1 を呼び出し、2 つのパラメーターを渡します。

```
CALL PGM1 (:hv1,:hv2)
```

例 2: C において、INOUT_SQLDA という名前の SQLDA を使用して SALARY_PROC と呼ばれるプロシージャーを呼び出します。

```
struct sqlda *INOUT_SQLDA;

/* Setup code for SQLDA variables goes here */

CALL SALARY_PROC USING DESCRIPTOR :*INOUT_SQLDA;
```

例 *3:* 以下のステートメントを使用して、 Java プロシージャーをデータベース内で定義します。

```
CREATE PROCEDURE PARTS_ON_HAND (IN  PARTNUM  INTEGER,
                                OUT COST     DECIMAL(7,2),
                                OUT QUANTITY INTEGER)
                LANGUAGE JAVA PARAMETER STYLE JAVA
                EXTERNAL NAME 'parts!onhand';
```

Java アプリケーションは、以下のコード・フラグメントを使用して、接続コンテキスト 'ctx' においてこのプロシージャーを呼び出します。

```
...
int        variable1;
BigDecimal variable2;
Integer    variable3;
...
#sql [ctx] {CALL PARTS_ON_HAND(:IN variable1, :OUT variable2, :OUT variable3)};
...
```

このアプリケーション・コードのフラグメントは、CALL ステートメントで指定された *procedure-name* がデータベースにあり、外部名 'parts!onhand' を持っているため、クラス *parts* にある Java メソッド *onhand* を呼び出します。

例 *4:* リレーショナル・データベース BRANCHRDB2 に対してプロシージャー PGM2 を呼び出し、1 つのパラメーターを渡します。

```
CALL BRANCHRDB2.SCHEMA3.PGM2 (:hv1)
```

例 *5:* 以下のプロシージャーが存在するものと想定します。

```
CREATE PROCEDURE update_order(
                    IN IN_POID BIGINT,
                    IN IN_CUSTID BIGINT DEFAULT GLOBAL_CUST_ID,
                    IN NEW_STATUS VARCHAR(10) DEFAULT NULL,
                    IN NEW_ORDERDATE DATE DEFAULT NULL,
                    IN NEW_COMMENTS VARCHAR(1000)DEFAULT NULL)...
```

また、グローバル変数 GLOBAL_CUST_ID が 1002 という値に設定されているものとします。カスタマー 1002 の注文 5000 の状況を「出荷済み」に変更するためにプロシージャーを呼び出します。残りの引数をデフォルトの NULL 値にしておくことで、その他の注文データは現状のままとすることができます。

```
CALL update_order (5000, NEW_STATUS => 'Shipped')
```

1001 という ID を持つカスタマーから連絡があり、購買注文 5002 の品物を受け取って満足していると示されます。注文を更新します。

```
CALL update_order (5002,
          IN_CUSTID => 1001,
          NEW_STATUS => 'Received',
          NEW_COMMENTS => 'Customer satisfied with the order.')
```

CLOSE

CLOSE ステートメントは、カーソルをクローズします。カーソルのオープン時に結果表が作成された場合は、その表は破棄されます。

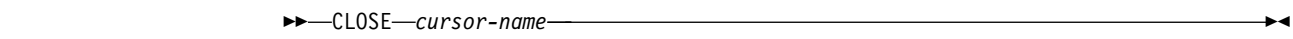
呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。これは実行可能ステートメントですが、動的に準備することはできません。Java では指定できません。

権限

権限は不要です。カーソルの使用に必要な権限については、 1353 ページの『DECLARE CURSOR』を参照してください。

構文



説明

cursor-name
: クローズするカーソルを識別します。DECLARE CURSOR ステートメントの項で説明されているように、 *cursor-name* は、宣言されたカーソルを指定しなければなりません。 CLOSE ステートメントは、オープン状態にあるカーソルに対して実行しなければなりません。

注

暗黙的なカーソル・クローズ**:** 以下の時点では、プログラム内のすべてのカーソルはクローズ状態にあります。

- プログラムが呼び出されたとき。
  - CLOSQLCSR(*ENDPGM) が指定されている場合、プログラムが呼び出されるたびに、すべてのカーソルがクローズ状態になります。
  - CLOSQLCSR(*ENDSQL) が指定されている場合、1 つの SQL プログラムが呼び出しスタックに残っている間は、プログラムが初めて呼び出されるときに限って、すべてのカーソルがクローズ状態になります。
  - CLOSQLCSR(*ENDJOB) が指定されている場合、ジョブ内でプログラムが最初に呼び出されたときに限って、すべてのカーソルがクローズ状態になります。
  - CLOSQLCSR(*ENDMOD) が指定されている場合、モジュールが開始されるたびに、すべてのカーソルがクローズ状態になります。
  - CLOSQLCSR(*ENDACTGRP) が指定されている場合、活動化グループ内で、プログラム中のモジュールが最初に開始されたときに、すべてのカーソルがクローズ状態になります。

- HOLD オプションの指定がない COMMIT または ROLLBACK ステートメントを実行して、プログラムから新規の作業単位を開始したとき。 HOLD オプションを指定して宣言されたカーソルは、COMMIT ステートメントではクローズされません。

注: Db2 for i データベース・マネージャーは、照会をインプリメントするためにファイルをオープンします。このファイルのクローズは、SQL CLOSE ステートメントとは別に行うことができます。詳しくは、「SQL プログラミング」を参照してください。

パフォーマンスのためのカーソルのクローズ: カーソルを、できるだけ早い時機に明示的にクローズすることによって、パフォーマンスを向上させることができます。

プロシージャーに関する考慮事項: クローズされずに呼び出し側プログラムに戻ったプロシージャー内のカーソルには、特殊な規則が適用されます。詳しくは、 1015 ページの『CALL』を参照してください。

例

COBOL プログラムでカーソル C1 を使用し、EMPPROJACT 表の最初の 4 つの列から一度に 1 行ずつ値を取り出して、その値を次のホスト変数に入れます。

- EMP (CHAR(6))
- PRJ (CHAR(6))
- ACT (SMALLINT)
- TIM (DECIMAL(5,2))

最後にカーソルをクローズします。

```
EXEC SQL  BEGIN DECLARE SECTION  END-EXEC.
  77  EMP              PIC X(6).
  77  PRJ              PIC X(6).
  77  ACT              PIC S9(4) BINARY.
  77  TIM              PIC S9(3)V9(2) PACKED-DECIMAL.
EXEC SQL  END DECLARE SECTION  END-EXEC.
.
.
.

EXEC SQL  DECLARE C1 CURSOR FOR
            SELECT EMPNO, PROJNO, ACTNO, EMPTIME
              FROM EMPPROJACT                       END-EXEC.

EXEC SQL  OPEN C1  END-EXEC.

EXEC SQL  FETCH C1 INTO :EMP, :PRJ, :ACT, :TIM  END-EXEC.

IF SQLSTATE = '02000'
  PERFORM DATA-NOT-FOUND
ELSE
  PERFORM GET-REST-OF-ACTIVITY UNTIL SQLSTATE IS NOT EQUAL TO '00000'.

EXEC SQL  CLOSE C1  END-EXEC.

GET-REST-OF-ACTIVITY
EXEC SQL  FETCH C1 INTO :EMP, :PRJ, :ACT, :TIM  END-EXEC.
.
.
.
```

COMMENT

COMMENT ステートメントは、種々のデータベース・オブジェクトのカタログ記述にコメントを追加したり、置換したりします。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、あるいは対話式に実行することもできます。これは、動的に準備できる実行可能ステートメントです。

権限

表、ビュー、別名、列、タイプ、パッケージ、シーケンス、変数、または XSR オブジェクトに対してコメントを付けるには、ステートメントの権限 ID が保持する特権に、次のうち少なくとも 1 つを含める必要があります。

- ステートメント内の表、ビュー、別名、タイプ、パッケージ、シーケンス、変数、または XSR オブジェクトの場合
  - その表、ビュー、別名、タイプ、パッケージ、シーケンス、変数、または XSR オブジェクトに対する ALTER 特権、および
  - その表、ビュー、別名、索引、タイプ、パッケージ、シーケンス、変数、XSR オブジェクトが入っているスキーマに対する USAGE 特権
- 管理権限

制約またはトリガーに対してコメントを付けるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

- ステートメント内の制約またはトリガーのサブジェクト表に対して、
  - 対象表に対する ALTER 特権
  - 対象表が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

索引に対してコメントを付けるには、ステートメントの権限 ID が保持する特権に、次のうち少なくとも 1 つを含める必要があります。

- ステートメントで識別される索引に対して、
  - その索引についての *OBJALTER システム権限
  - 索引が含まれるスキーマに対する USAGE 特権
- 管理権限

関数に対してコメントを付けるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

- SYSFUNCS および SYSROUTINES カタログ・ビューと表の場合
  - SYSROUTINES に対する UPDATE 特権
  - SYSFUNCS に対する *OBJOPR システム権限、および
  - スキーマ QSYS2 に対する USAGE 特権
- 管理権限

プロシージャーに対してコメントを付けるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

- SYSPROCS および SYSROUTINES カタログ・ビューと表の場合:
  - SYSROUTINES に対する UPDATE 特権
  - SYSPROCS に対する *OBJOPR システム権限、および
  - スキーマ QSYS2 に対する USAGE 特権
- 管理権限

パラメーターに対してコメントを付けるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

- SYSPARMS カタログ表の場合
  - その表に対する UPDATE 特権、および
  - スキーマ QSYS2 に対する USAGE 特権
- 管理権限

マスクまたは許可に対してコメントを付けるには、次が必要です。

- ステートメントの権限 ID には、 IBM i のデータベース・セキュリティー管理者機能を実行する権限が付与されている必要があります。 21 ページの『管理権限』を参照してください。

シーケンスに対してコメントを付けるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要もあります。

- Change Data Area (CHGDTAARA)、CL コマンドに対する *USE 権限
- 管理権限

変数に対してコメントを付けるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要もあります。

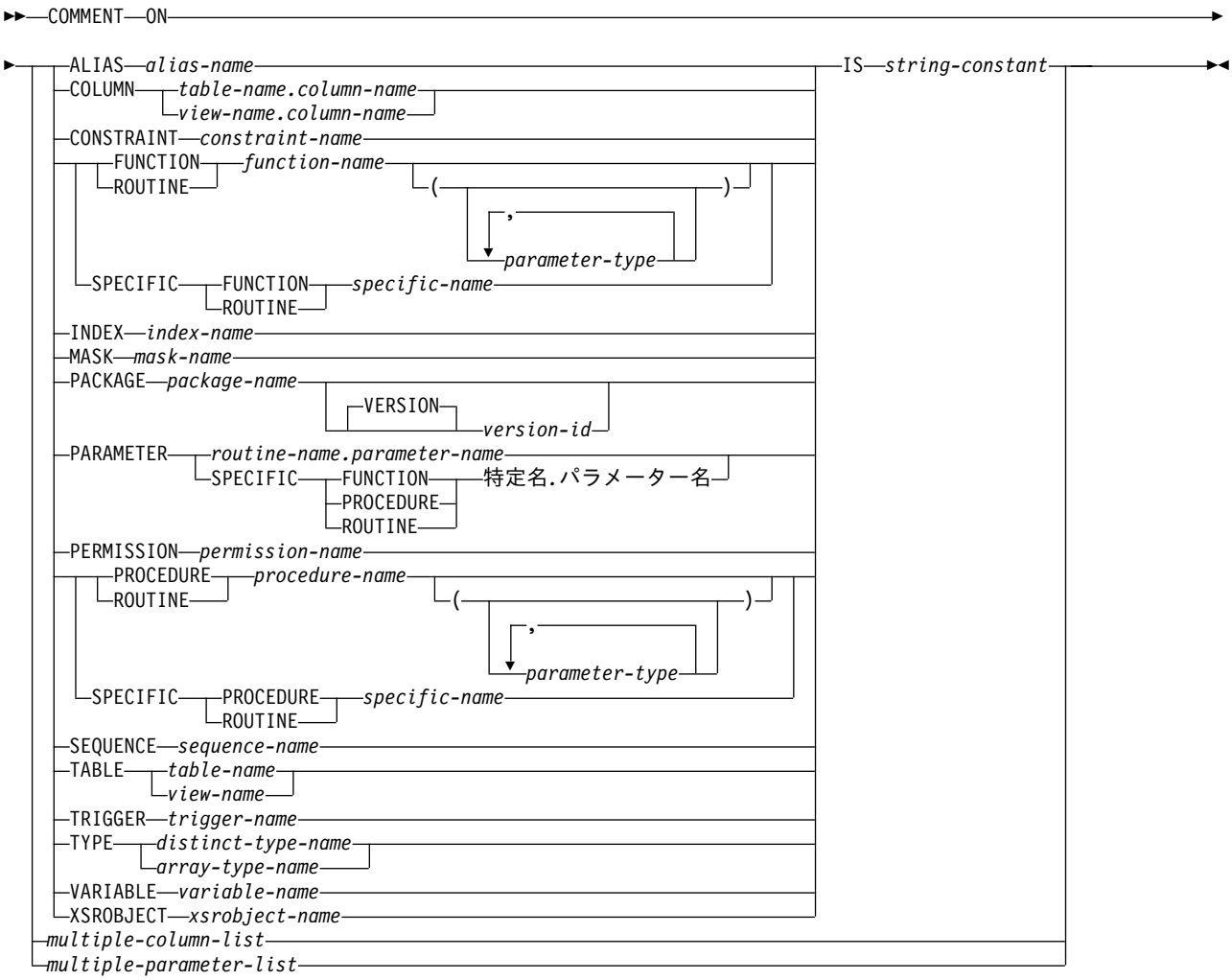
- SYSVARIABLES カタログ表の場合
  - その SYSVARIABLES に対する UPDATE 特権、および
  - スキーマ QSYS2 に対する USAGE 特権
- 管理権限

XSR オブジェクトに対してコメントを付けるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要もあります。

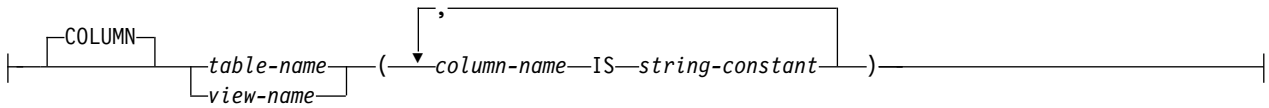
- XSROBJECTS カタログ表の場合
  - その XSROBJECTS に対する UPDATE 特権、および
  - スキーマ QSYS2 に対する USAGE 特権
- 管理権限

SQL 特権に対応するシステム権限の説明については、 表またはビューへの権限を検査する際の対応するシステム権限、 ユーザー定義タイプへの権限を検査する際の対応するシステム権限、 シーケンスへの権限を検査する際の対応するシステム権限、 変数への権限を検査する際の対応するシステム権限、 パッケージへの権限を検査する際の対応するシステム権限、および XSR オブジェクトへの権限を検査する際の対応するシステム権限を参照してください。

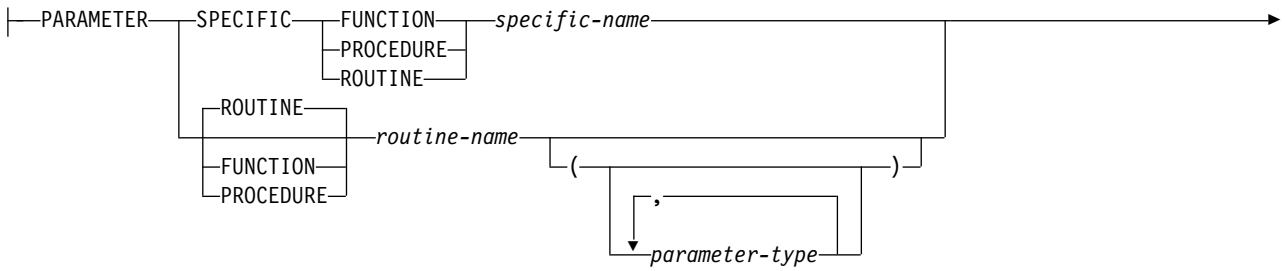
構文



multiple-column-list:



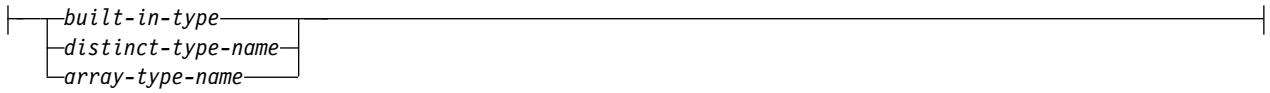
multiple-parameter-list:



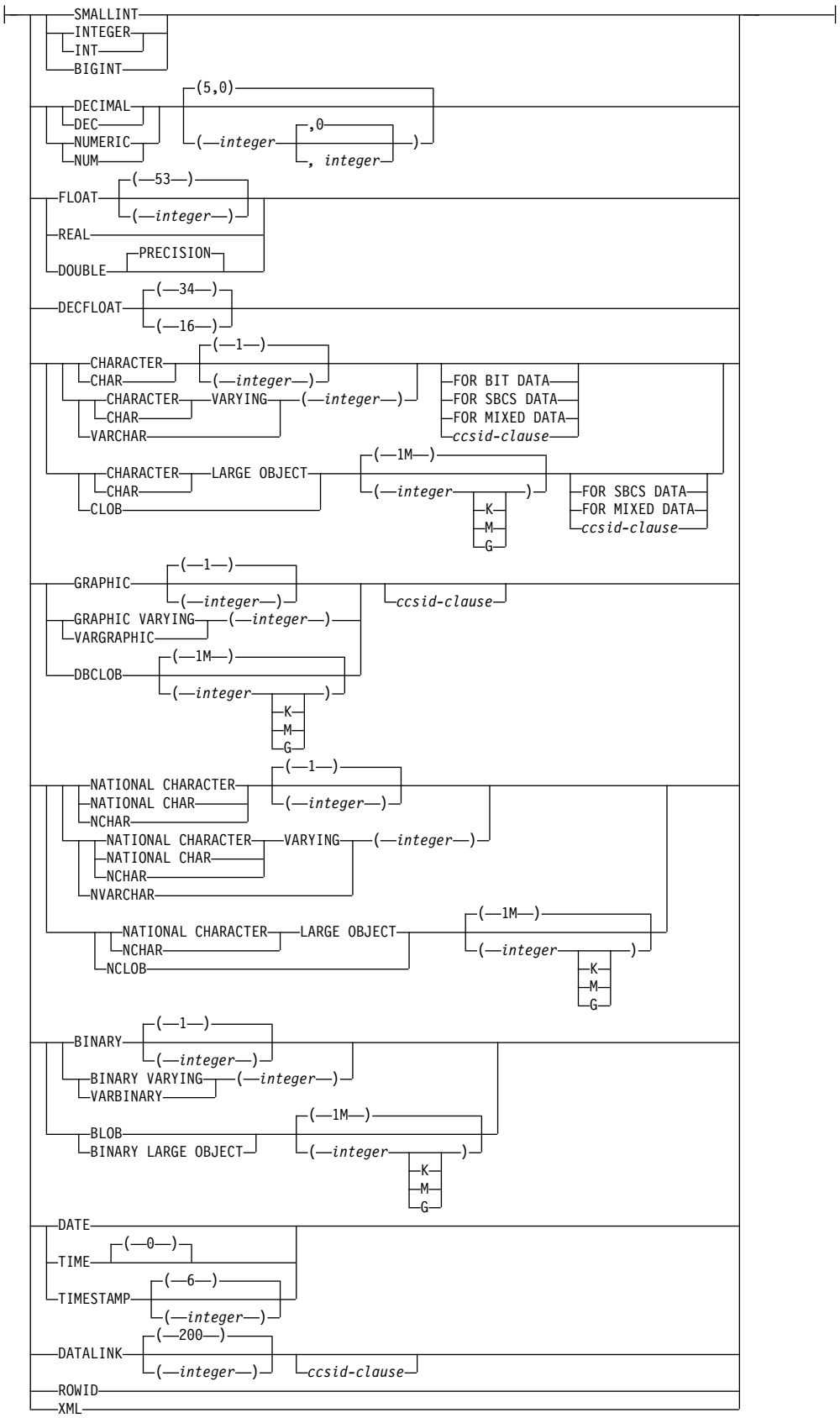
COMMENT

parameter-type:

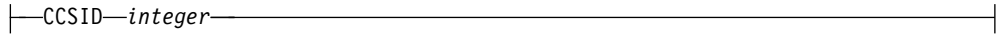
data-type:



built-in-type:


SMALLINT
INTEGER
INT
BIGINT
DECIMAL
DEC
NUMERIC
NUM
(5,0)
(integer
,0
, integer
)
FLOAT
(53)
(integer)
REAL
DOUBLE
PRECISION
DECFLOAT
(34)
(16)
CHARACTER
CHAR
(1)
(integer)
CHARACTER VARYING
CHAR VARYING
VARCHAR
FOR BIT DATA
FOR SBCS DATA
FOR MIXED DATA
ccsid-clause
CHARACTER LARGE OBJECT
CHAR LARGE OBJECT
CLOB
(1M)
K
M
G
GRAPHIC
VARGRAPHIC
GRAPHIC VARYING
DBCLOB
NATIONAL CHARACTER
NATIONAL CHAR
NCHAR
NATIONAL CHARACTER VARYING
NATIONAL CHAR VARYING
NVARCHAR
NATIONAL CHARACTER LARGE OBJECT
NCLOB
BINARY
BINARY VARYING
VARBINARY
BLOB
BINARY LARGE OBJECT
DATE
TIME
(0)
TIMESTAMP
(6)
DATALINK
(200)
ROWID
XML

ccsid-clause:



説明

ALIAS *alias-name*
コメントの適用対象である別名を識別します。この別名 は、現行サーバーに存在している別名を示すものでなければなりません。

COLUMN
列に対してコメントの追加、またはコメントの置き換えを行うことを指定します。

table-name.column-name または *view-name.column-name*
コメントの追加、または置き換えを行う列を識別します。表名 またはビュー名 は、現行サーバーにある表またはビューを示すものでなければなりませんが、宣言済み一時表を示すものであってはなりません。列名 は、その表またはビューの列を識別するものでなければなりません。

CONSTRAINT
制約に対してコメントの追加や置換を行うことを指定します。

constraint-name
コメントを適用する制約を識別します。 *constraint-name* は、現行サーバーに存在する制約を識別する必要があります。

FUNCTION または **SPECIFIC FUNCTION**
コメントの適用対象である関数を識別します。この関数は現行サーバーに存在していなければならず、ユーザー定義関数である必要があります。関数は、それぞれその名前、関数シグニチャー、あるいは特定名によって識別することができます。

FUNCTION *function-name*
関数を名前によって識別します。 *function-name* は厳密に 1 つの関数を示す必要があります。この関数には、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前の関数が複数ある場合、エラーが戻されます。

FUNCTION *function-name (parameter-type, ...)*
関数を一意的に識別する関数シグニチャーによって、関数を識別します。 *function-name (parameter-type,...)* は、指定された関数シグニチャーを持つ関数を識別しなければなりません。指定されたパラメーターは、関数の作成時に指定された、対応する位置にあるデータ・タイプと一致していなければなりません。コメントする関数インスタンスを識別する場合、データ・タイプの数とデータ・タイプの論理連結が使用されます。データ・タイプの同義語は、一致として扱われます。デフォルトがあるパラメーターは、このシグニチャーに含まれていなければなりません。

function-name () を指定する場合、識別される関数にパラメーターを使用することはできません。

function-name
関数の名前を識別します。

(parameter-type, ...)
関数のパラメーターを識別します。

非修飾の特殊タイプ名を指定する場合、データベース・マネージャーはその特殊タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、以下のいずれかを使用します。

- 空の括弧は、データベース・マネージャーがデータ・タイプの一致の判別に際して属性を無視することを示します。例えば、DEC() は、DEC(7,2) のデータ・タイプで定義された関数のパラメーターに一致するものとみなされます。ただし、FLOAT に空の括弧を指定することはできません。これは、そのパラメーター値が特定のデータ・タイプ (REAL または DOUBLE) を示しているからです。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を指定した場合、その値は、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプが FLOAT の場合、突き合わせはデータ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度は指定された値に厳密に一致している必要はありません。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。暗黙の長さは、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

FOR DATA 文節または CCSID 文節の指定はオプショナルです。いずれの文節も指定しないと、データ・タイプが一致するかどうかを判定する場合に、データベース・マネージャーが属性を無視することを示します。どちらか一方の文節を指定する場合は、CREATE FUNCTION ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

AS LOCATOR
関数が、このパラメーターのロケーターを受け取るように定義されることを示します。 AS LOCATOR を指定する場合は、データ・タイプは LOB または XML、あるいは LOB または XML に基づく特殊タイプでなければなりません。 AS LOCATOR を指定した場合、FOR SBCS DATA または FOR MIXED DATA を指定してはなりません。

SPECIFIC FUNCTION *specific-name*
関数を特定名によって識別します。 *specific-name* は、現行サーバーに存在する特定の関数を示している必要があります。

INDEX *index-name*
コメントが適用される索引を指定します。この索引名 は、現行サーバーに存在している索引を示すものでなければなりません。

MASK *mask-name*
コメントを適用するマスクを指定します。マスク名 は、現行サーバーに存在するマスクを示していなければなりません。

PACKAGE *package-name*
コメントを付けるパッケージを識別します。このパッケージ名 は、現行サーバーに存在しているパッケージを識別していなければなりません。[85]

VERSION *version-id*
バージョン *ID* は、作成時にパッケージに割り当てられたバージョン ID です。バージョン *ID* を指定しない場合、バージョン ID として NULL ストリングが使用されます。

PARAMETER
パラメーターに対してコメントの追加や置換を行うことを指定します。

ルーチン名.パラメーター名
コメントが適用されるパラメーターを識別します。このパラメーターの指定対象には、プロシージャーや関数があります。ルーチン名 では、現行サーバーに存在しているプロシージャーや関数を識別し、パラメーター名 では、そのプロシージャーや関数のパラメーターを識別する必要があります。

特定名.パラメーター名
コメントが適用されるパラメーターを識別します。このパラメーターの指定対象には、プロシージャーや関数があります。特定名 では、現行サーバーに存在しているプロシージャーや関数を識別し、パラメーター名 では、そのプロシージャーや関数のパラメーターを識別する必要があります。

PERMISSION *permission-name*
コメントを適用する権限を指定します。許可名 は、現行サーバーに存在する許可を示していなければなりません。

PROCEDURE または **SPECIFIC PROCEDURE**
コメントが適用されるプロシージャーを識別します。このプロシージャー名 は、現行サーバーに存在しているプロシージャーを識別していなければなりません。

PROCEDURE *procedure-name*
プロシージャーを名前によって識別します。プロシージャー名 は、ただ 1 つのプロシージャーを識別していなければなりません。このプロシージャーには、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前のプロシージャーが複数ある場合、エラーが戻されます。

PROCEDURE *procedure-name (parameter-type, ...)*
プロシージャーを一意的に識別するプロシージャー・シグニチャーによって、プロシージャーを識別します。 *procedure-name (parameter-type,...)* では、指定されたプロシージャー・シグニチャーを持つプロシージャーを識別する必要があります。指定されたパラメーターは、プロシージャーの作成時に指定された、対応する位置にあるデータ・タイプと一致していなければなりません。コメントする対象のプロシージャー・インスタンスを識別するために、データ・タイプの数とデータ・タイプの論理連結が使用されます。データ・タイプの同義語は、一致として扱われます。デフォルトがあるパラメーターは、このシグニチャーに含まれていなければなりません。

85. 識別されたパッケージがバージョン *ID* を持っている場合、コメントは 176 バイトに制限されます。

プロシージャー名 () を指定する場合、識別されるプロシージャーにパラメーターを使用することはできません。

procedure-name
プロシージャーの名前を識別します。

(parameter-type, ...)
プロシージャーのパラメーターを識別します。

非修飾の特殊タイプ名または配列タイプ名を指定する場合、データベース・マネージャーはその特殊タイプまたは配列タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、以下のいずれかを使用します。

- 空の括弧は、データベース・マネージャーがデータ・タイプの一致の判別に際して属性を無視することを示します。例えば、DEC() は、DEC(7,2) のデータ・タイプで定義されたプロシージャーのパラメーターに一致するものとみなされます。ただし、FLOAT に空の括弧を指定することはできません。これは、そのパラメーター値が特定のデータ・タイプ (REAL または DOUBLE) を示しているからです。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を指定する場合、その値は、 CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプが FLOAT の場合、突き合わせはデータ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度は指定された値に厳密に一致している必要はありません。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。暗黙の長さは、CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

FOR DATA 文節または CCSID 文節の指定はオプショナルです。いずれの文節も指定しないと、データ・タイプが一致するかどうかを判定する場合に、データベース・マネージャーが属性を無視することを示します。どちらか一方の文節を指定する場合は、CREATE PROCEDURE ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

AS LOCATOR
プロシージャーが、このパラメーターのロケーターを受け取るように定義されることを示します。 AS LOCATOR を指定する場合は、データ・タイプは LOB または XML、あるいは LOB または XML に基づく特殊タイプでなければなりません。 AS LOCATOR を指定した場合、FOR SBCS DATA または FOR MIXED DATA を指定してはなりません。

SPECIFIC PROCEDURE *specific-name*
プロシージャーを特定名によって識別します。特定名 は、現行サーバーに存在している特定のプロシージャーを識別していなければなりません。

SEQUENCE *sequence-name*
コメントが適用されるシーケンスを識別します。シーケンス名 は、現行サーバーに存在するシーケンスを示すものでなければなりません。

TABLE *table-name* または *view-name*
コメントを付ける表またはビューを識別します。表名 またはビュー名 は、現行サーバーにある表またはビューを示すものでなければなりませんが、宣言済み一時表を示すものであってはなりません。

TRIGGER *trigger-name*
コメントが適用されるトリガーを識別します。トリガー名 は、現行サーバーに存在しているトリガーを識別していなければなりません。

TYPE *distinct-type-name* または *array-type-name*
コメントを適用する特殊タイプまたは配列タイプを識別します。特殊タイプ名 または 配列タイプ名 は、現行サーバーに存在するタイプを示すものでなければなりません。

VARIABLE *variable-name*
コメントを適用する変数を識別します。この変数名 は、現行サーバーに存在している変数を識別していなければなりません。

XSROBJECT *xsrobject-name*
コメントを適用する XSR オブジェクトを識別します。この*XSR* オブジェクト名 は、現行サーバーに存在しているXSR オブジェクトを示すものでなければなりません。

IS この後に、付加または置換するコメントを指定します。

string-constant
2000 文字 (シーケンスの場合は 500 文字) 以内であれば、どんな文字ストリング定数でも構いません。

multiple-column-list

表またはビューの複数の列に対してコメントを行うには、その表またはビューを指定してから、以下の形式のリストを括弧で囲んで指定します。

```
(column-name IS string-constant,
 列名 IS ストリング定数, ...)
```

列名は修飾してはなりません。それぞれの名前は、指定した表またはビューの列を識別しなければならず、その表またはビューは、現行サーバーに存在していなければなりません。

multiple-parameter-list

プロシージャーや関数の複数のパラメーターに対してコメントを付けるには、プロシージャー名、関数名、または特定名を指定してから、次の形式でリストを括弧で囲んで指定します。

```
(parameter-name IS string-constant,
 パラメーター名 IS ストリング定数, ...)
```

パラメーター名は修飾することはできません。また、それぞれの名前では、指定されたプロシージャーや関数のパラメーターを識別し、しかも、そのプロシージャーや関数は現行サーバーに入れておく必要があります。

注

代替構文: 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- PACKAGE の同義語として、キーワード PROGRAM を使用することができます。
- キーワード DATA TYPE または DISTINCT TYPE を TYPE の同義語として使用することができます。

例

例 *1:* 表 EMPLOYEE に関するコメントを挿入します。

```
COMMENT ON TABLE EMPLOYEE
  IS 'Reflects first quarter 2000 reorganization'
```

例 *2:* ビュー EMP_VIEW1 に関するコメントを挿入します。

```
COMMENT ON TABLE EMP_VIEW1
  IS 'View of the EMPLOYEE table without salary information'
```

例 *3:* 表 EMPLOYEE の列 EDLEVEL に関するコメントを挿入します。

```
COMMENT ON COLUMN EMPLOYEE.EDLEVEL
  IS 'Highest grade level passed in school'
```

例 *4:* DEPARTMENT 表内の 2 列に対してコメントを入力します。

```
COMMENT ON DEPARTMENT
  (MGRNO IS 'EMPLOYEE NUMBER OF DEPARTMENT MANAGER',
   ADMRDEPT IS 'DEPARTMENT NUMBER OF ADMINISTERING DEPARTMENT')
```

例 *5:* パッケージ PAYROLL に関するコメントを挿入します。

```
COMMENT ON PACKAGE PAYROLL
  IS 'This package is used for distributed payroll processing.'
```

COMMIT

COMMIT ステートメントは、作業単位を終了させ、その作業単位によって行われたデータベースの変更をコミットします。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

トリガー・プログラムとその対象となるプログラムが同じコミットメント定義のもとで実行される場合、トリガーでは COMMIT は許されません。リモート・アプリケーション・サーバーへの分散作業単位接続時に呼び出されるプロシージャー、または ATOMIC として定義されているプロシージャーでは、COMMIT を使用することはできません。関数内では COMMIT を使用できません。

権限

権限は不要です。

構文

説明

COMMIT ステートメントは、そのステートメントが実行される作業単位を終了させ、新たな作業単位を開始します。このステートメントは、対象の作業単位の中で SQL スキーマ・ステートメント (DROP SCHEMA を除く) および SQL データ変更ステートメントにより行われたすべての変更をコミットします。 SQL スキーマ・ステートメントおよび SQL データ変更ステートメントについて詳しくは、865 ページの『第 7 章 ステートメント』を参照してください。

解除保留状態の接続は終了します。

WORK
COMMIT WORK は、COMMIT と同じ効果を持ちます。

HOLD
リソースを保持するように指示します。これを指定すると、HOLD オプションで宣言されているかどうかに関わらず、現在オープンされているカーソルはクローズされません。その作業単位の過程で獲得したリソースはすべて保留されます。特定の行およびオブジェクトについて、その作業単位の中で暗黙的に獲得されたロックは、解放されます。

クローズされないカーソルに必要なオブジェクト・レベルのロックを除き、暗黙に獲得されたロックは、すべて解放されます。

保留されていないロケーターはすべて解放されます。保留ロケーターについて詳しくは、 1551 ページの『HOLD LOCATOR』を参照してください。

注

推奨されるコーディング方法**:** 明示的な COMMIT または ROLLBACK ステートメントを、アプリケーション・プロセスの最後にコーディングしてください。アプリケーション環境に応じて、暗黙的なコミットまたはロールバック操作のいずれかが、アプリケーション・プロセスの終わりに実行されます。このため、移植可能なアプリケーションでは、明示的な COMMIT または ROLLBACK が許可された環境で実行が終了する前に、COMMIT または ROLLBACK を明示的に実行する必要があります。

暗黙的な COMMIT または ROLLBACK を以下の環境下で実行することができます。

- デフォルトの活動化グループの場合
  - デフォルトの活動化グループのもとで実行されているアプリケーションが終了する時点で、暗黙の COMMIT は実行されません。デフォルトの活動化グループのもとで実行されるプログラムには、対話式 SQL、Query Manager、および非 ILE プログラムなどがあります。
  - 作業をコミットするには、COMMIT を出す必要があります。
- デフォルト以外の活動化グループで、コミットメント定義の有効範囲がその活動化グループの場合
  - その活動化グループが正常終了する時点で、そのコミットメント定義は暗黙のうちにコミットされます。
  - その活動化グループが異常終了する場合、コミットメント定義は暗黙のうちにロールバックされます。
- 活動化グループがどのようなタイプであっても、コミットメント定義の有効範囲がジョブであれば、暗黙のコミットが行われることはありません。

コミットの影響**:** HOLD を使用せずにコミットすると、以下のエラーの原因となります。

- 解除保留状態の接続は終了します。

  既存の接続の場合
  - 位置指定 UPDATE または DELETE ステートメントを実行するのに先立って、 FETCH ステートメントが必要になる場合でも、 WITH HOLD 文節を使用して宣言されたすべてのオープン・カーソルは保存され、その現在位置は保守されます。
  - 分離レベル NC でオープンされたものを含めて、WITH HOLD 節なしで宣言されたすべてのオープン・カーソルはクローズされます。
- 保留されていないすべての LOB ロケーターが解放されます。ロケーターが WITH HOLD プロパティーを持つカーソルを通して検索された LOB 値に関連付けられている場合にも、これが当てはまることに注意してください。
- LOCK TABLE ステートメントによって獲得されたすべてのロックは解放されます。クローズされないカーソルに必要なロックを除き、暗黙に獲得されたロックはすべて解放されます。

行ロックの制限: 1 つの作業単位には、最大 4,000,000 行までの処理を組み込むことができます。これには、SELECT や FETCH ステートメントの過程で検索された行[86]、ならびに、INSERT、DELETE、および UPDATE ステートメントの一部として挿入、削除、または更新された行も含まれます。[87]

影響されないステートメント: コミットおよびロールバック操作が DROP SCHEMA ステートメントに影響することはありません。したがって、このステートメントは、COMMIT(*CHG)、COMMIT(*CS)、 COMMIT(*ALL)、または COMMIT(*RR) も指定しているアプリケーション・プログラムでは使用できません。

COMMIT 制約事項: 2 次スレッドのユーザー定義関数のコミットまたはロールバックは使用できません。

コミットメント定義: SQL で使用されるコミットメント定義は、次のように決められます。

- SQL を呼び出すプログラムの活動化グループが既に活動化グループ・レベルのコミットメント定義を使用している場合、SQL はそのコミットメント定義を使用します。
- SQL を呼び出すプログラムの活動化グループがジョブ・レベルのコミットメント定義を使用している場合、SQL はそのジョブ・レベルのコミットメント定義を使用します。
- SQL を呼び出すプログラムの活動化グループがその時点でコミットメント定義を使用していない場合、ジョブ・コミットメント定義が開始されていれば、SQL はそのジョブ・コミットメント定義を使用します。
- SQL を呼び出すプログラムの活動化グループがその時点でコミットメント定義を使用しておらず、しかもジョブ・コミットメント定義が開始されていない場合、SQL は暗黙的にコミットメント定義を開始します。SQL は、以下の指定を伴うコミットメント制御開始 (STRCMTCTL) コマンドを使用します。
  - CMTSCOPE(*ACTGRP) パラメーター
  - CRTSQLxxx、STRSQL、または RUNSQLSTM のいずれかのコマンドで指定された COMMIT オプションに基づく LCKLVL パラメーター REXX では、LCKLVL パラメーターは、SET OPTION ステートメントのコミット・オプションに基づきます。

例

C プログラムの中で、EMPLOYEE 表のある従業員 (EMPNO) から別の従業員へ、ある金額の手数料 (COMM) を移します。一方の行から手数料の金額を引いた上

86. この制限には、次のものも含まれます。
- 高水準言語のファイル処理機能によるコミットメント制御のもとでオープンされたファイルに基づいてアクセスまたは変更された行。
- トリガー、または CASCADE、SET NULL、あるいは SET DEFAULT 参照保全削除規則の結果として削除、更新、または挿入された行。

87. COMMIT(*CHG) または COMMIT(*CS) を指定した場合は例外で、これらの行は行数の合計には含まれません。

で、その金額をもう一方の行に加えます。この両方の操作が正常に完了するまで、データベースへの永続的な変更を行わないように、COMMIT ステートメントを使用します。

```
void main ()
  {

     EXEC SQL  BEGIN DECLARE SECTION;
     decimal(5,2) AMOUNT;
     char FROM_EMPNO[7];
     char TO_EMPNO[7];
     EXEC SQL  END DECLARE SECTION;
     EXEC SQL  INCLUDE SQLCA;
     EXEC SQL  WHENEVER SQLERROR GOTO SQLERR;
     ...
     EXEC SQL  UPDATE EMPLOYEE
               SET COMM = COMM - :AMOUNT
               WHERE EMPNO = :FROM_EMPNO;
     EXEC SQL  UPDATE EMPLOYEE
               SET COMM = COMM + :AMOUNT
               WHERE EMPNO = :TO_EMPNO;
   FINISHED:
     EXEC SQL  COMMIT WORK;
     return;

   SQLERR:
      ...
     EXEC SQL  WHENEVER SQLERROR CONTINUE;  /* continue if error on rollback */
     EXEC SQL  ROLLBACK WORK;
     return;
 }
```

コンパウンド (動的)

複合 (動的) ステートメントは、他のステートメントを一緒にグループ化して 1 つの実行可能ルーチンにします。複合ステートメントによって、SQL 変数、カーソル、および条件ハンドラーを宣言することができます。

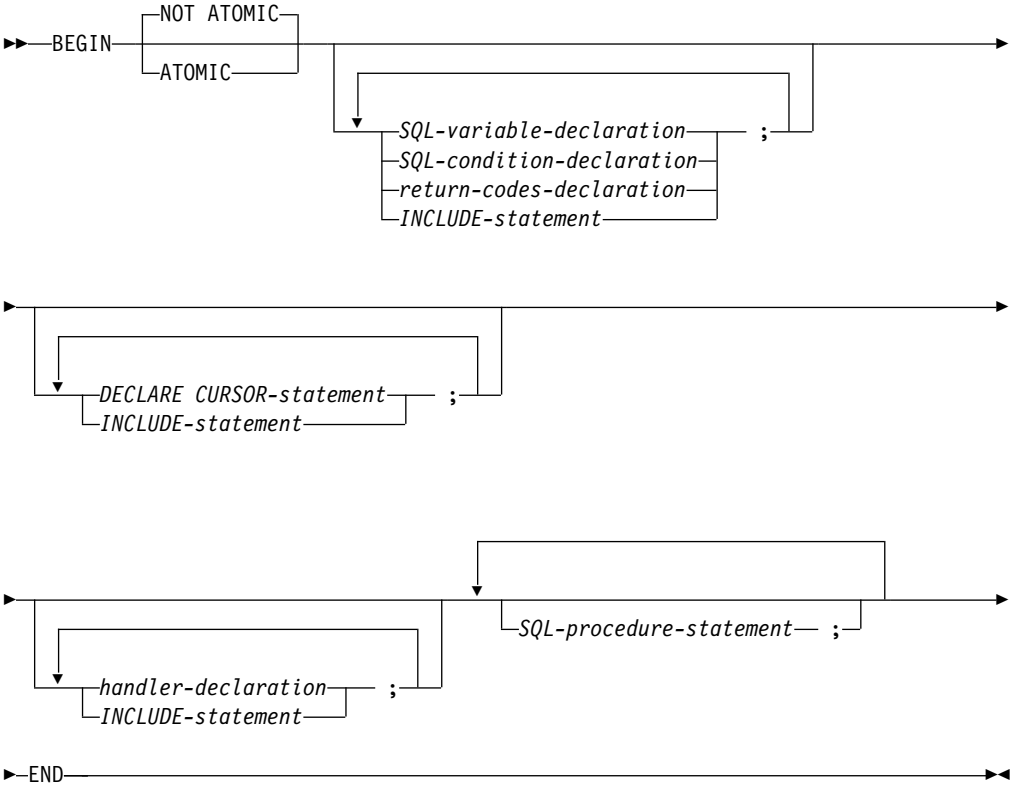
呼び出し

このステートメントは、対話式に発行することができます。このステートメントは、動的に準備できる実行可能ステートメントです。これをアプリケーション・プログラムに組み込むことはできません。

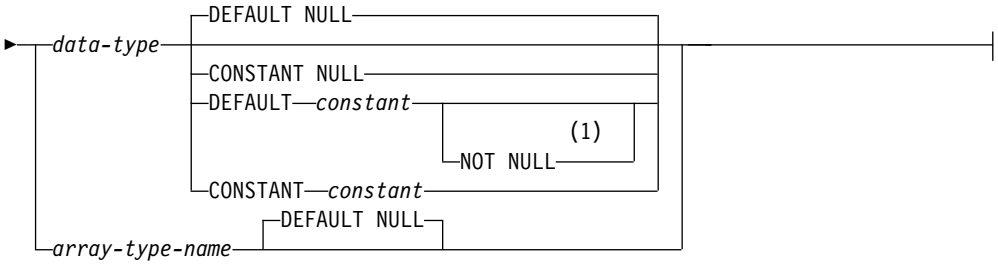
権限

ステートメントの許可 ID によって保持される特権には、複合 (動的) ステートメントに指定されている SQL ステートメントを呼び出すために必要なすべての特権も含まれていなければなりません。

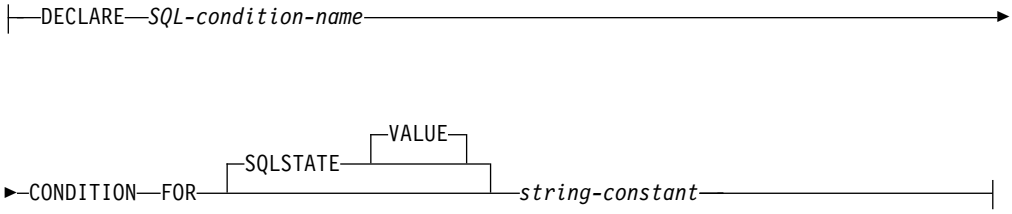
構文



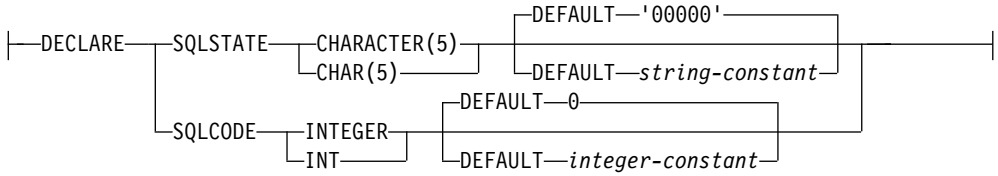
SQL-variable-declaration:



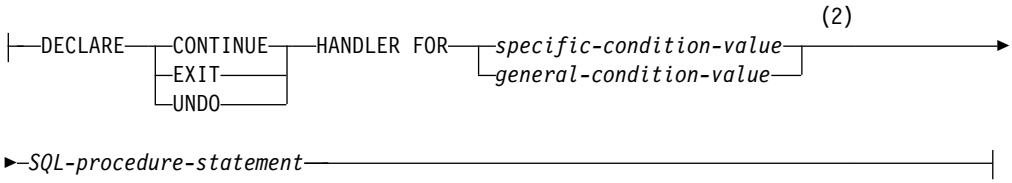
SQL-condition-declaration:



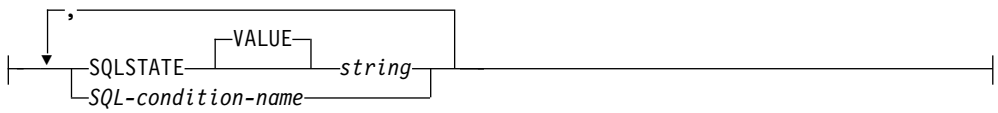
return-codes-declaration:



handler-declaration:



specific-condition-value:



general-condition-value:

data-type:

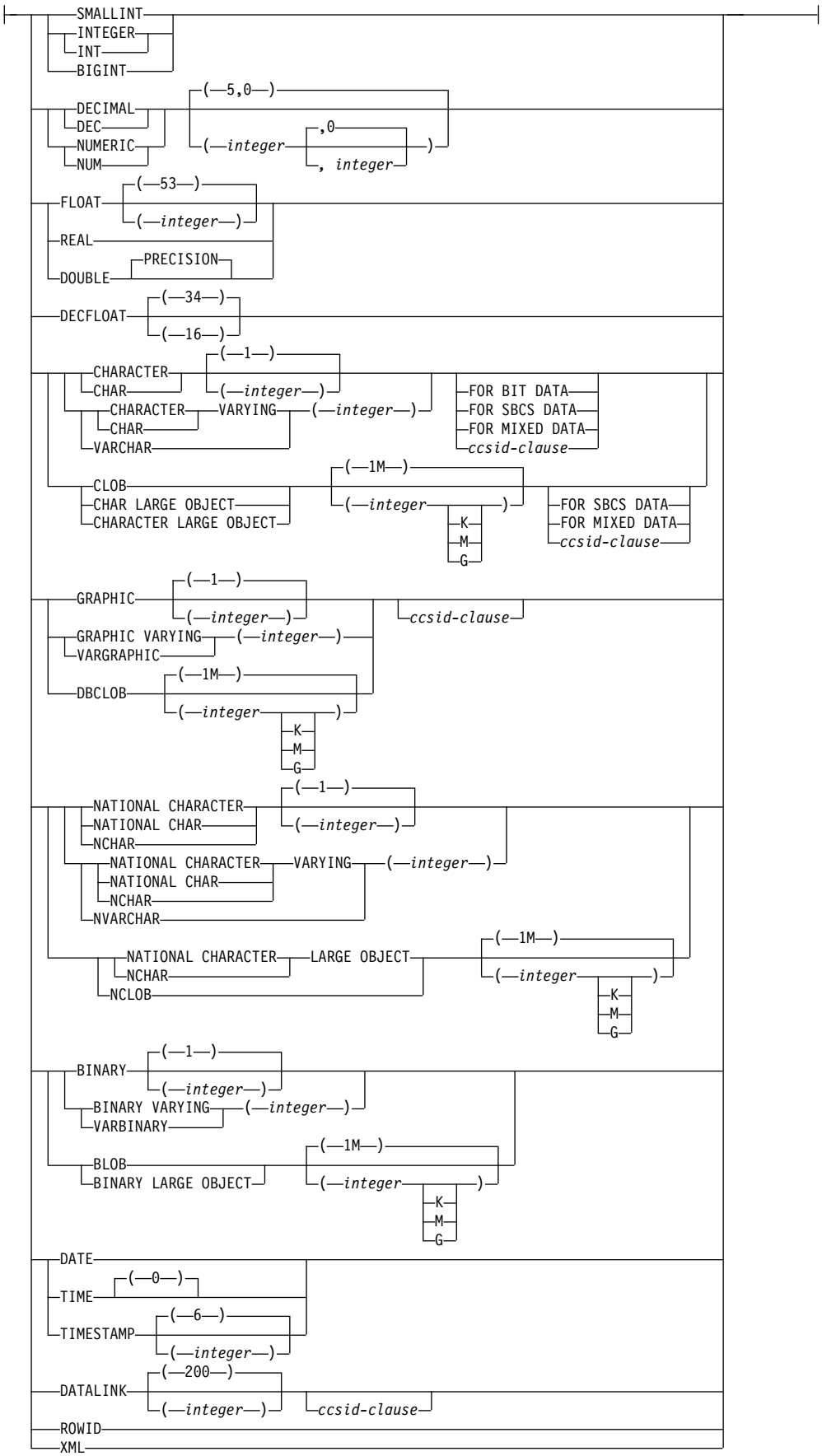
注:

1 DEFAULT 文節と NOT NULL 文節は、どちらの順序で指定しても構いません。

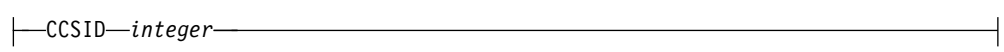
2 特定条件値 と一般条件値 を同一のハンドラー宣言に同時に指定することはできません。

3 同じ文節を複数回指定することはできません。

built-in-type:


SMALLINT
INTEGER
INT
BIGINT
DECIMAL
DEC
NUMERIC
NUM
(5,0)
(integer)
,0
, integer
FLOAT
(53)
(integer)
REAL
DOUBLE
PRECISION
DECFLOAT
(34)
(16)
CHARACTER
CHAR
(1)
(integer)
CHARACTER VARYING
CHAR VARYING
VARCHAR
FOR BIT DATA
FOR SBCS DATA
FOR MIXED DATA
ccsid-clause
CLOB
CHAR LARGE OBJECT
CHARACTER LARGE OBJECT
(1M)
K
M
G
GRAPHIC
GRAPHIC VARYING
VARGRAPHIC
DBCLOB
NATIONAL CHARACTER
NATIONAL CHAR
NCHAR
NATIONAL CHARACTER VARYING
NVARCHAR
NCLOB
NATIONAL CHARACTER LARGE OBJECT
BINARY
BINARY VARYING
VARBINARY
BLOB
BINARY LARGE OBJECT
DATE
TIME
(0)
TIMESTAMP
(6)
DATALINK
(200)
ROWID
XML

ccsid-clause:



説明

ATOMIC
ATOMIC は、複合 (動的) ステートメントの未処理の例外条件によって複合ステートメントがロールバックされることを示します。ATOMIC が指定される場合、その複合ステートメント内に、COMMIT、ROLLBACK (ROLLBACK TO SAVEPOINT は指定可能)、SET CURRENT DEBUG MODE、SET CURRENT DEGREE、SET CURRENT DECFLOAT ROUNDING MODE、SET CURRENT IMPLICIT XMLPARSE OPTION、SET CURRENT TEMPORAL SYSTEM_TIME、SET PATH、および SET SCHEMA ステートメントを指定することはできません。

NOT ATOMIC
NOT ATOMIC は、複合 (動的) ステートメントで未処理の例外条件が発生しても、その複合ステートメントをロールバックしません。

SQL-variable-declaration
複合 (動的) ステートメントに対してローカルな変数を宣言します。

SQL-variable-name
ローカル SQL 変数の名前を定義します。データベース・マネージャーは、区切り文字のない SQL 変数名はすべて大文字に変換します。同じ複合 (動的) ステートメントの内部にある別の SQL 変数と同じ名前を指定しないでください。ただし、その複合ステートメント の内部でネストされている複合 (動的) ステートメント内の宣言は例外です。SQL 変数に列名と同じ名前を付けないでください。同一ステートメント内に同名の列が複数個ある場合に、SQL 変数名がどのように解決されるかについては、 1773 ページの『SQL パラメーターおよび変数の参照』を参照してください。変数名は 'SQL' で始まっていてはなりません。

data-type
変数のデータ・タイプを指定します。データ・タイプの説明については、1238 ページの『CREATE TABLE』を参照してください。

データ・タイプ がグラフィック・ストリング・データ・タイプの場合は、UTF-16 または UCS-2 データを指す CCSID 1200 または 13488 を指定してください。 CCSID が指定されていない場合は、グラフィック・ストリング変数の CCSID は、そのジョブに関連付けられている DBCS CCSID です。

配列タイプ名
この SQL 変数は、CREATE TYPE (配列) ステートメントで定義する配列である、ということを指定します。

DEFAULT *constant* または **NULL**
SQL 変数のデフォルト値を定義します。指定された定数は、 113 ページの『割り当ておよび比較』で説明している割り当て規則に従って、その変数に割り当てることができる値を表している必要があります。変数は複合 (動的) ステートメントが呼び出されるときに初期化されます。デフォルト値の指定

がない場合は、SQL 変数は NULL に初期化されます。 XML タイプの SQL 変数でデフォルト値を指定することはできません。

NOT NULL
SQL 変数に NULL 値が入るのを防ぎます。NOT NULL を指定しないことは、その変数がヌルであってもよいことを意味します。XML タイプの SQL 変数で NOT NULL を指定することはできません。

CONSTANT *constant* **または NULL**
SQL 変数に変更できない固定値があることを指定します。 CONSTANT を使用して定義された SQL 変数は、すべての割り当て操作のターゲットとして使用できません。指定された定数は、 113 ページの『割り当ておよび比較』で説明している割り当て規則に従って、その変数に割り当てることができる値を表している必要があります。

SQL-condition-declaration
条件名と対応する SQLSTATE 値を宣言します。

SQL-condition-name
条件の名前を指定します。条件名は、複合 (動的) ステートメント内で固有でなければなりません。ただし、その複合ステートメント内部でネストされた複合ステートメント 内の宣言は除きます。

FOR SQLSTATE *string-constant*
この条件に関連する SQLSTATE を指定します。 ストリング定数は、5 文字で指定しなければなりません。 SQLSTATE クラス (最初の 2 文字) は「00」であってはなりません。

return-codes-declaration
GET DIAGNOSTICS や空の複合ステートメント 以外の SQL ステートメントの実行後に、診断域の最初の条件で設定される SQLSTATE と SQLCODE という特殊な SQL 変数を宣言します。

SQLSTATE および SQLCODE 特殊変数は、GET DIAGNOSTICS 以外の直前の SQL ステートメントを処理した結果得られる SQL 戻りコードを取得する手段としてのみ使用されます。SQLSTATE 変数および SQLCODE 変数を使用する意図がある場合は、直ちにその値を別の SQL 変数に保管して、次の SQL ステートメントの実行後に戻される SQL 戻りコードによって、その値が置き換えられるのを防ぐ必要があります。SQLSTATE を処理するハンドラーが定義される場合、割り当てがそのハンドラーの最初のステートメントである場合、割り当てステートメントを使用してその SQLSTATE (または関連した SQLCODE) 値を別の SQL 変数に保管することができます。

これらの変数に値を割り当てることは、禁止されてはいませんが、お勧めしません。これらの特殊変数への割り当ては、条件ハンドラーによって無視されます。SQLSTATE および SQLCODE 特殊変数は NULL に設定できません。

DECLARE CURSOR ステートメント
複合 (動的) ステートメントでカーソルを宣言します。カーソル名は、複合 (動的) ステートメント内で固有でなければなりません。ただし、その複合ステートメント内部でネストされた複合ステートメント 内の宣言は除きます。

カーソル名 は、それが宣言されている複合 (動的) ステートメント内でのみ参照することができます。これには、その複合 (動的) ステートメント内部でネストされた複合ステートメント も含みます。

カーソルをオープンする場合は OPEN ステートメントを、カーソルを使用して行を読み取る場合は FETCH ステートメントを使用します。オープンされたカーソルは、複合 (動的) ステートメントの終わりでクローズされます。

カーソルの宣言について詳しくは、 1353 ページの『DECLARE CURSOR』を参照してください。

handler-declaration

ハンドラー、つまり複合 (動的) ステートメント内で例外条件または完了条件が発生したときに実行される *SQL* プロシージャー・ステートメント を指定します。

条件ハンドラー宣言は、同じ条件値または SQLSTATE 値を複数回参照することはできず、また、1 つの SQLSTATE 値と、それと同じ SQLSTATE 値を表す条件名を参照することもできません。 SQLSTATE 値のリストおよび詳細については、「SQL メッセージおよびコード」トピック集を参照してください。

さらに、1 つの複合 (動的) ステートメントで複数の条件ハンドラーが宣言される場合、2 つの条件ハンドラー宣言が以下のものを指定することはできません。

- 同一の一般条件カテゴリー、または
- 同一の特定条件 (SQLSTATE 値として、またはそれと同一の値を表す条件名として)

条件ハンドラーがアクティブであるのは、複合 (動的) ステートメント内 (ネストされた複合ステートメント を含む) でハンドラー宣言 の後にある *SQL* プロシージャー・ステートメント のセットに対してです。

ある条件のハンドラーは、ネストされた複合ステートメントの複数のレベルに存在することがあります。例えば、複合 (動的) ステートメント *n1* に別の複合ステートメント *n2* が含まれ、それにさらに別の複合ステートメント *n3* が含まれている場合を想定してください。例外条件が *n3* 内で生じると、 *n3* 内のアクティブなハンドラーが最初にその条件を処理することを許可されます。適切なハンドラーが *n3* に存在しない場合、その条件は *n2* に再通知されて、 *n2* 内のアクティブなハンドラーがその状態を処理できるようになります。適切なハンドラーが *n2* に存在しない場合、その条件は *n1* に再通知されて、 *n1* 内のアクティブなハンドラーがその状態を処理できるようになります。 *n1* 内に適切なハンドラーがない場合、その状態は未処理と見なされます。

条件ハンドラーには以下の 3 つのタイプがあります。

CONTINUE

条件ハンドラーが活動化され、正常に完了した後、例外を起こしたステートメントの後の SQL ステートメントに制御が戻されることを指定します。 IF、 CASE、 FOR、 WHILE、または REPEAT の中で比較を実行しているときにエラーが発生すると、それに対応する END IF、 END CASE、 END FOR、 END WHILE、または END REPEAT の後のステートメントに制御が戻されます。

EXIT
条件ハンドラーが活動化され、正常に完了した後は、複合 (動的) ステートメントの末尾へ制御を戻すことを指定します。

UNDO
条件ハンドラーが活動化されると、複合 (動的) ステートメントによって加えられた変更がロールバックされることを指定します。このハンドラーが正常に完了すると、複合 (動的) ステートメントの終わりに制御が戻されます。 UNDO を指定する場合は、ATOMIC を指定する必要があります。

このハンドラーが活動化される条件は以下のとおりです。

SQLSTATE *string*
特定の SQLSTATE が発生したときにハンドラーを呼び出すことを指定します。SQLSTATE クラス (最初の 2 文字) は「00」であってはなりません。

SQL-condition-name
条件名に関連した特定の SQLSTATE が発生したときにハンドラーを呼び出すことを指定します。*SQL* 条件名は、*SQL* 条件宣言 の中で既に定義されていなければなりません。

SQLEXCEPTION
例外条件が発生したときにハンドラーを呼び出すことを指定します。例外条件は、最初の 2 文字が '00'、'01'、または '02' ではない SQLSTATE 値によって表されます。

SQLWARNING
警告条件が発生したときにハンドラーを呼び出すことを指定します。警告条件は、最初の 2 文字が '01' である SQLSTATE 値によって表されます。

NOT FOUND
NOT FOUND 条件が発生したときにハンドラーを呼び出すことを指定します。 NOT FOUND 条件は、最初の 2 文字が '02' である SQLSTATE 値によって表されます。

SQL-procedure-statement
1779 ページの『SQL プロシージャー・ステートメント』 で定義されているような SQL ステートメントまたは SQL 制御ステートメント。SET RESULT SETS および SET SESSION AUTHORIZATION SQL ステートメントは許可されません。

注

複合 (動的) 内容: 複合 (動的) ステートメント内で使用できる構成体について詳しくは、 1771 ページの『第 8 章 SQL 制御ステートメント』を参照してください。

ネストされた複合ステートメント: 複合ステートメント を複合 (動的) ステートメント内でネストすることができます。ネストされた複合ステートメントを使用して、変数定義、条件名、条件ハンドラー、およびカーソルの有効範囲を複合 (動的) ステートメント内のステートメントのサブセットに指定することができます。これにより、各 *SQL* プロシージャー・ステートメントに対する処理を単純化できます。ネストされた複合ステートメントのサポートにより、条件ハンドラーの宣言内で複合ステートメントを使用できるようになります。

複合 (動的) ステートメントの実行: 複合 (動的) ステートメントが動的に準備され、実行されるとき、複合ステートメント内のステートメントは静的ステートメントとして処理されます。複合ステートメントのステートメントが埋め込まれた一時プログラムが作成され、実行されます。プログラム名は QTEMP.QCMPDnnnnn です (nnnnn はジョブの固有番号)。

CURRENT PATH および **CURRENT SCHEMA:** 現行スキーマおよび現行パスは、準備され、実行される複合 (動的) ステートメントに適用されます。複合ステートメント内でこれらの特殊レジスターに加えられた変更は、同じ複合ステートメント内にある後続のステートメント解決には影響しません。例えば、次のようになります。

```
SET SCHEMA prodlib;
SET stmt =  'BEGIN  SET SCHEMA datalib;  INSERT INTO test_table VALUES(1);  END';
PREPARE compound_stmt FROM stmt;
EXECUTE compound_stmt;
```

複合ステートメント compound_stmt が準備され、実行されると、 INSERT は、DATALIB.TEST_TABLE ではなく PRODLIB.TEST_TABLE に解決されます。

ほとんどのエンド・ユーザー SQL スクリプト・インターフェースは、動的 SQL を使用して SQL ステートメントを実行します。例えば、 IBM i Navigator の SQL スクリプトの実行での以下の 2 つのステートメントの実行は、論理的には上記の例と等価です。 INSERT は、DATALIB.TEST_TABLE ではなく PRODLIB.TEST_TABLE に解決されます。

```
SET SCHEMA prodlib;
BEGIN
  SET SCHEMA datalib;
  INSERT INTO test_table VALUES(1);
END;
```

条件ハンドラー: 複合 (動的) ステートメント内の条件ハンドラーは、外部 SQL アプリケーション・プログラムで使用される WHENEVER ステートメントとほぼ同じです。例外、警告、または Not Found 条件が発生すると自動的に制御を得るように、条件ハンドラーを定義できます。条件ハンドラーの本体には、その条件ハンドラーが活動化されるときに実行されるコードが含まれています。SQL ステートメントの処理のためにデータベース・マネージャーが戻す例外、警告、または Not Found 条件の結果として、条件ハンドラーを活動化することができます。または、ルーチン本体内で SIGNAL または RESIGNAL ステートメントが発行された結果を、活動化の条件とすることができます。

条件ハンドラーは複合 (動的) ステートメント内で宣言され、その複合 (動的) ステートメント内のすべての条件ハンドラー宣言の後にある *SQL* プロシージャー・ステートメント のセットに対してアクティブです。もっと具体的には、条件ハンドラー宣言 H の有効範囲は、H が表示される複合 (動的) ステートメント内に含まれている条件ハンドラー宣言の後に続く *SQL* プロシージャー・ステートメント のリストです。したがって、H の有効範囲には、条件ハンドラー H の本体中に含まれるステートメントは入りません。つまり、条件ハンドラーは条件ハンドラー自体の本体中で生じる条件を処理できないことになります。同様に、同一の複合 (動的) ステートメント中で H1 と H2 という 2 つの条件ハンドラーが宣言されている場合、H1 は H2 の本体中で生じる条件を処理できず、H2 は H1 の本体中で生じる条件を処理できません。

条件ハンドラーの宣言は、それを活動化する条件、条件ハンドラーのタイプ (CONTINUE、EXIT、または UNDO)、およびハンドラーのアクションを指定します。条件ハンドラーのタイプにより、ハンドラー・アクションが正常に完了した後に制御がどこに戻されるかが決まります。

条件ハンドラーの活動化: *SQL プロシージャー・ステートメント* の処理で、正常終了以外の条件が発生すると、その条件を処理できる条件ハンドラーが有効範囲内にある場合、その条件を処理するためにその条件ハンドラーが活動化されます。

ネストされた複合ステートメント を持つ複合 (動的) ステートメント内で、ある特定の条件を処理することができる条件ハンドラーが、ネストされた複合ステートメントのいくつかのレベルに存在する可能性があります。活動化される条件ハンドラーは、条件が検出された有効範囲の最も内側で宣言される条件ハンドラーです。そのネスト・レベルの複数の条件ハンドラーが条件を処理できる場合、活動化される条件ハンドラーは、その複合ステートメントで宣言された最も適したハンドラーです。

最も適したハンドラーとは、該当の例外条件または完了条件の SQLSTATE に最も一致度の高い複合ステートメントに定義されているハンドラーです。

例えば、最も内側の複合ステートメントが SQLSTATE 22001 に対する特定のハンドラーとともに、SQLEXCEPTION に対するハンドラーも宣言する場合、SQLSTATE 22001 が検出されるときに、SQLSTATE 22001 に対する特定ハンドラーが最も適したハンドラーです。この場合、その特定ハンドラーが活動化されます。

条件ハンドラーがアクティブ化されると、その条件ハンドラーのアクションが実行されます。ハンドラー・アクションが正常に完了するか、未処理の警告を出して完了する場合、診断域がクリアされ、条件ハンドラーのタイプ (CONTINUE、EXIT、または UNDO ハンドラー) により、どこに制御が戻されるかが決まります。さらに、ハンドラーが正常に完了するか、未処理の警告で完了する場合、SQLSTATE および SQLCODE SQL 変数がクリアされます。

ハンドラー・アクションが正常に完了しないときに、ハンドラー・アクションで検出された条件に対して適切なハンドラーが存在する場合、その条件ハンドラーが活動化されます。それ以外の場合、条件ハンドラー内で検出される条件は処理されません。

未処理条件**:** 条件が検出されるときに、その条件に対して適したハンドラーが存在しない場合、その条件は処理されません。

- 未処理の条件が例外条件である場合、失敗したステートメントを含んでいる複合ステートメントは、未処理例外条件により終了します。
- 未処理条件が警告または Not Found 条件である場合、次のステートメントに移って処理が続けられます。次の SQL ステートメントの処理により、診断域内の未処理条件に関する情報が上書きされ、未処理条件の形跡が存在しなくなることに注意してください。

ネストされた複合ステートメントと一緒に **SIGNAL** または **RESIGNAL** ステートメントを使用する場合の考慮事項: 条件ハンドラーで指定された *SQL プロシージャー・ステートメント* が、例外 SQLSTATE を持つ SIGNAL または RESIGNAL ス

テートメントのどちらかである場合、複合 (動的) ステートメントは、指定された例外を出して終了します。これは、この条件ハンドラー、または同じ複合 (動的) ステートメント内の別の条件ハンドラーが CONTINUE を指定する場合であっても起こります。これらの条件ハンドラーがこの例外の有効範囲内にないからです。複合 (動的) ステートメント内に複合ステートメント がネストされている場合、複合 (動的) ステートメント内の条件ハンドラーは、複合ステートメント 内からの例外を処理できます。これは、それらの条件ハンドラーは例外の有効範囲内にあるからです。

SQL 変数中の **NULL** 値: SQL 変数の値が NULL で、標識変数が許可されていない SQL ステートメント (CONNECT あるいは DESCRIBE など) に使用されている場合、エラーが戻されます。

コミットされていない変更: コミットされていないトランザクション作業の有効範囲は動的複合ステートメントが実行されている接続の活動化グループに限定されます。

オープン・カーソルへの影響: 何らかの理由で複合 (動的) ステートメントが終了した後、その複合ステートメント内で宣言されているすべてのオープン・カーソルはクローズされます。

SQLSTATE および **SQLCODE SQL** 変数に関する考慮事項**:** 複合 (動的) ステートメント自体は、SQLSTATE および SQLCODE SQL 変数に影響を与えません。ただし、複合ステートメント内にある SQL ステートメントは、SQLSTATE 変数および SQLCODE SQL 変数に影響する可能性があります。複合ステートメントの終わりで、SQLSTATE および SQLCODE SQL 変数は、その複合ステートメント内で実行され、SQLSTATE および SQLCODE SQL 変数を変更した最後の SQL ステートメントの結果を反映します。SQLSTATE および SQLCODE 変数が複合 (動的) ステートメント内で変更されなかった場合、それらの変数には、複合 (動的) ステートメントが起動されたときと同じ値が含まれます。

CONNECT (タイプ 1)

CONNECT (タイプ 1) ステートメントは、リモート作業単位の規則を使用して、アプリケーション・プロセス内の活動化グループを識別されたアプリケーション・サーバーに接続します。次に、このサーバーはその活動化グループの現行サーバーになります。このタイプの CONNECT ステートメントが使用されるのは、RDBCNNMTH(*RUW) が CRTSQLxxx コマンドで指定されている場合です。

これら 2 つのタイプのステートメントの相違点については、 1865 ページの『CONNECT (タイプ 1) と CONNECT (タイプ 2) の相違点』を参照してください。接続状態の詳細については、 52 ページの『アプリケーション指向の分散作業単位』を参照してください。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことと、対話式に呼び出すことだけが可能です。これは実行可能ステートメントですが、動的に準備することはできません。Java または REXX では指定できません。

CONNECT は、トリガーおよび関数で使用できません。

権限

このステートメントの権限 ID によって保持される特権には、通信レベルのセキュリティーが含まれていなければなりません。 (「分散データベース・プログラミング」のセキュリティーに関するセクションを参照してください。)

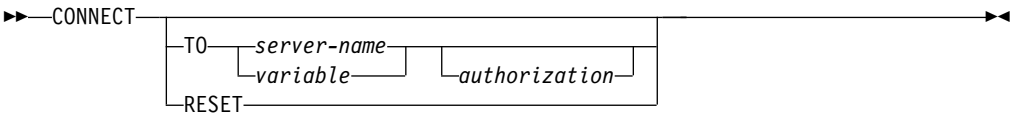
アプリケーション・サーバーが Db2 for i である場合は、次の場合に該当しない限り、ステートメントの発行者のユーザー・プロファイルをアプリケーション・サーバー・システム上でも有効なユーザー・プロファイルにする必要があります。

- ユーザーが指定されている。この場合は、USER 文節で、アプリケーション・サーバー・システム上の有効なユーザー・プロファイルを指定する必要があります。
- TCP/IP が、アプリケーション・サーバーのサーバー権限記入項目で使用されている。この場合は、サーバー権限記入項目で、アプリケーション・サーバー・システム上の有効なユーザー・プロファイルを指定する必要があります。

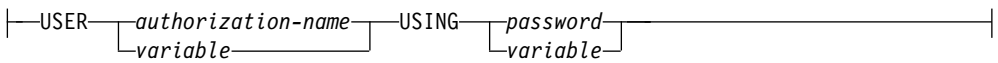
ステートメントでグローバル変数を参照する場合は、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別されるグローバル変数に対して、
  - そのグローバル変数に対する READ 特権
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

構文



authorization:



説明

TO *server-name* または *variable*
指定したサーバー名、または指定した変数に入っているサーバー名によってアプリケーション・サーバーを識別します。スキーマ名で修飾すれば、グローバル変数を使用することもできます。変数を指定する場合、

- CHAR または VARCHAR ホスト変数でなければなりません。
- 標識変数を伴っていてはなりません。
- サーバー名は、その変数内で左寄せし、通常 ID の形成の規則に従っていなければなりません。
- サーバー名の長さが、変数の長さよりも短い場合、右側をブランクで埋めなければなりません。

この CONNECT ステートメントが実行される時点で、指定したサーバー名または変数に入っているサーバー名は、ローカル・ディレクトリーに記述されているアプリケーション・サーバーを識別していなければなりません。また、その活動化グループは、接続可能状態でなければなりません。

サーバー名 がローカル・リレーショナル・データベースの場合、指定する権限名 は、ジョブのユーザーでなければなりません。 指定された権限名 がジョブのユーザーと異なっていると、エラーが発生し、アプリケーションは接続されない状態のままになります。

USER *authorization-name* または *variable*
アプリケーション・サーバーへの接続に使用される権限名を識別します。スキーマ名で修飾すれば、グローバル変数を使用することもできます。

変数 を指定する場合、

- CHAR または VARCHAR ホスト変数でなければなりません。
- 標識変数を伴っていてはなりません。
- 権限名は、その変数に左寄せして入れ、権限名の命名規則に従っていなければなりません。
- 権限名の長さが、変数の長さよりも短い場合には、右側をブランクで埋めなければなりません。
- サーバー名の値には小文字を含めてはなりません。

USING *password* または *variable*
アプリケーション・サーバーへの接続に使用されるパスワードを識別します。

パスワードをリテラルとして指定する場合、そのリテラルは文字ストリングでなければなりません。最大長は 128 文字です。左寄せしなければなりません。リテラル形式のパスワードは、静的 SQL または REXX では許可されていません。

変数 を指定する場合、

- この変数は、グローバル変数にすることはできません。
- CHAR または VARCHAR ホスト変数でなければなりません。
- 標識変数を伴っていてはなりません。
- パスワードは、変数に左寄せして入れなければなりません。
- パスワードの長さが変数の長さよりも短い場合には、右側をブランクで埋めなければなりません。

RESET

CONNECT RESET は、CONNECT TO x と同等です。ここで、x はローカル・サーバー名です。

CONNECT (オペランドの指定なし)

この形式の CONNECT ステートメントは、現行サーバーについての情報を戻し、接続状態、オープン・カーソル、準備されたステートメント、またはロックに対してまったく影響を与えません。接続情報は、SQL 診断域 (または SQLCA) にある接続情報項目に戻されます。

注

接続成功: CONNECT ステートメントが正常に完了した場合 :

- オープン・カーソルはすべてクローズされ、準備されたステートメントはすべて破棄され、すべてのロックは現行接続から解放されます。
- その活動化グループは、現行および休止の接続 (接続されている場合) すべてから切り離され、指定されたアプリケーション・サーバーに接続されます。
- 接続したアプリケーション・サーバーの名前が、特殊レジスター CURRENT SERVER に入れられます。
- アプリケーション・サーバーについての情報は、 SQL 診断領域の接続情報項目に入れられます。
- アプリケーション・サーバーに関する情報も、SQLCA のフィールド SQLERRP および SQLERRD(4) に入れられます。そのアプリケーション・サーバーが IBM リレーショナル・データベース・プロダクトである場合は、フィールド SQLERRP の中の情報は、*pppvvrrm* の形式をとります。ただし、
  - *ppp* は、製品を次のように識別します。
    - ARI (Db2 UDB サーバー (VM および VSE 版) の場合)
    - DSN (Db2 for z/OS の場合)
    - QSQ (Db2 for i の場合)
    - 他のすべての Db2 プロダクトの場合は SQL
  - *vv* は、2 桁のバージョン ID です (例えば、'09' など)。
  - *rr* は、2 桁のリリース ID です (例えば、'01' など)。
  - *m* は、「0」のような 1 文字の修正レベルを示します。

例えば、アプリケーション・サーバーがバージョン 9 の Db2 for z/OS の場合は、SQLERRP の値は「DSN09010」になります。

SQLCA の SQLERRD(4) フィールドには、アプリケーション・サーバーがコミット可能な更新の実行を許可するか否かを示す値が入ります。 CONNECT (タイプ 1) ステートメントの場合、SQLERRD(4) には、常に値 1 が入ります。値 1 は、コミット可能な更新を行うことができること、ならびに、接続が次のようなものであることを示します。

- 無保護会話を使用する[88]、または
- アプリケーション・リクエスターのドライバー・プログラムとの接続に *RUW 接続方式を使用する、または
- *RUW 接続方式を使用するローカル接続である。

- 接続についての追加情報は SQLCA のフィールド SQLERRMC に入れられます。 1867 ページの『付録 C. SQLCA (SQL 連絡域)』を参照してください。

接続失敗: この CONNECT ステートメントの実行が成功しなかった場合、 SQL 診断域にある DB2_MODULE_DETECTING_ERROR 条件情報項目 (または SQLCA の SQLERRP フィールド) には、エラーを検出したアプリケーション・リクエスターのモジュールの名前が入れられます。モジュール名の最初の 3 文字は、プロダクトを識別しています。例えば、アプリケーション・リクエスターが Db2 LUW である場合、最初の 3 文字は 'SQL' になります。

活動化グループが接続可能状態でないために、CONNECT ステートメントが正常に実行されない場合は、その活動化グループの接続状態は変更されません。

CONNECT ステートメントが他の何らかの理由で正常に実行されない場合は、次のようになります。

- その活動化グループは接続可能でありながら、未接続状態のままです。
- オープン・カーソルはすべてクローズされ、準備されたステートメントはすべて破棄され、すべてのロックは現行または休止の接続すべてから解放されます。

接続可能で未接続状態のアプリケーションのみが、CONNECT または SET CONNECTION ステートメントを実行できます。

暗黙接続:

- デフォルト活動化グループで実行される場合、次の状態にあれば、SQL プログラムは、暗黙のうちにリモートのリレーショナル・データベースに接続します。
  - その活動化グループが接続可能状態にある。
  - プログラム・スタックの最初の SQL プログラムの最初の SQL ステートメントが実行される。
- デフォルト以外の活動化グループで実行されている SQL プログラムは、その活動化グループに関する最初の SQL プログラムの最初の SQL ステートメントを実行する時に、リモートのリレーショナル・データベースへ暗黙に接続します。

88. ネットワーク接続と SQL 接続との間で混同が生じる可能性を減少させるため、本書では、TCP/IP ならびに APPC を介するネットワーク接続に適用させる場合に「会話」という用語を使用します。ただし、正式には、これは APPC 接続だけに適用されます。

注: 活動化グループにより実行される最初の SQL ステートメントを CONNECT ステートメントにするのは、望ましい方法です。

APPC を RDB との接続に使用している場合、暗黙の接続では、常にアプリケーション・リクエスター・ジョブの権限名 を送信しますが、パスワードは送信しません。 アプリケーション・サーバー・ジョブの権限名 が異なる場合や、パスワードを送る必要がある場合は、明示的な接続ステートメントを使用する必要があります。

TCP/IP を RDB との接続に使用している場合、暗黙の接続は上記の制約に拘束されることはありません。 ADDSVRAUTE コマンドと他の -SVRAUTE コマンドを使用すると、暗黙 (または明示) の CONNECT の実行元である所定のユーザーに対して、該当の RDB との接続で使用されるリモート権限名とパスワードを指定することが可能になります。

パスワードが ADDSVRAUTE コマンドや CHGSVRAUTE コマンドで保管されるようにするには、QRETSVRSEC システム値をデフォルト値の '0' ではなく、'1' に設定する必要があります。これらのコマンドを DRDA 接続に使用している場合に非常に重要なことは、SERVER パラメーターに RDB 名の値を英大文字で入力しなければならないと認識しておくことです。詳細については、タイプ 2 の CONNECT の節の例 2 を参照してください。

暗黙の接続の詳細については、SQL プログラミングのトピック集を参照してください。ユーザー・プロファイルとリレーショナル・データベースとの接続が一度確立されると、同じユーザー・プロファイルと同じリレーショナル・データベースとの以後の接続では、パスワード (指定がある場合) の妥当性が再検証されない場合があります。パスワードの妥当性の再検証は、会話がまだ活動状態であるかどうかによって異なります。詳細に関しては、分散データベース・プログラミングのトピック集を参照してください。

接続状態: 接続状態については、 50 ページの『リモート作業単位の接続管理』を参照してください。 CONNECT ステートメントは、連続して使用しても正常に実行されます。これは、CONNECT ステートメントは接続可能状態からその活動化グループを除去しないからです。

アプリケーション・グループの現行または休止接続に対する CONNECT は、次のように行われます。

- サーバー名によって識別された接続が、CONNECT (タイプ 1) ステートメントを使用して確立されていた場合には、どのようなアクションも取られません。カーソルはクローズされず、準備されたステートメントは破棄されず、またロックは解放されません。
- サーバー名によって識別された接続が、CONNECT (タイプ 2) ステートメントを使用して確立されていた場合、その CONNECT ステートメントは、他の CONNECT ステートメントと同様に実行されます。

CONNECT の前に CONNECT、COMMIT、DISCONNECT、SET CONNECTION、 RELEASE、または ROLLBACK 以外の SQL ステートメントがある場合は、その CONNECT は正常に実行できません。エラーを防ぐためには、CONNECT ステートメントを実行する前に、コミットまたはロールバックの操作を実行してください。

前の現行または休止接続が、保護会話を使用して確立されていた場合には、CONNECT (タイプ 1) ステートメントは失敗します。 CONNECT (タイプ 2) ステートメントを使用するか、または保護会話を使用したその接続を解放し、コミットを正しく行うことによって終了しなければなりません。

リモート・リレーショナル・データベースへの接続およびローカル・ディレクトリーについて詳しくは、「SQL プログラミング」および「分散データベース・プログラミング」を参照してください。

SET SESSION AUTHORIZATION: SET SESSION AUTHORIZATION ステートメントがスレッド内で実行された場合、接続ステートメントより前に SYSTEM_USER 値が SESSION_USER と同じでなければ、ローカル・サーバーへの CONNECT は失敗します。

これには、ACTGRP(*NEW) を指定するプログラムを呼び出すことによる暗黙的な接続が含まれます。

例

例 *1:* C プログラムで、アプリケーション・サーバー TOROLAB に接続します。

```
EXEC SQL  CONNECT TO TOROLAB;
```

例 *2:* C プログラムで、名前が変数 APP_SERVER (VARCHAR(18)) に保管されているアプリケーション・サーバーに接続します。接続が正常に完了した後で、接続したアプリケーション・サーバーのプロダクト ID を、変数 PRODUCT にコピーします。

```
void main ()
  {
    char product[9] = " ";
    EXEC SQL BEGIN DECLARE SECTION;
    char APP_SERVER[19];
    char username[11];
    char userpass[129];
    EXEC SQL END DECLARE SECTION;
    EXEC SQL INCLUDE SQLCA;
    strcpy(APP_SERVER,"TOROLAB");
    strcpy(username,"JOE");
    strcpy(userpass,"XYZ1");
    EXEC SQL CONNECT TO :APP_SERVER
             USER :username USING :userpass;
    if (strncmp(SQLSTATE, "00000", 5) )
      { EXEC SQL GET DIAGNOSTICS CONDITION 1
          product = DB2_PRODUCT_ID;  }
    ...
    return;
  }
```

CONNECT (タイプ 2)

CONNECT (タイプ 2) ステートメントは、アプリケーション指向分散作業単位の規則を使用して、アプリケーション・プロセス内の活動化グループを識別されたアプリケーション・サーバーに接続します。次に、このサーバーはその活動化グループの現行サーバーになります。このタイプの CONNECT ステートメントは、CRTSQLxxx コマンドに RDBCNNMTH(*DUW) の指定があった場合に使用します。

これら 2 つのタイプのステートメントの相違点については、 1865 ページの『CONNECT (タイプ 1) と CONNECT (タイプ 2) の相違点』を参照してください。接続状態の詳細については、 52 ページの『アプリケーション指向の分散作業単位』を参照してください。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、または対話式に呼び出すことができます。これは実行可能ステートメントですが、動的に準備することはできません。Java または REXX では指定できません。

CONNECT は、トリガーおよび関数で使用できません。

権限

このステートメントの権限 ID によって保持される特権には、通信レベルのセキュリティーが含まれていなければなりません。 (「分散データベース・プログラミング」トピック集のセキュリティーに関するセクションを参照してください。)

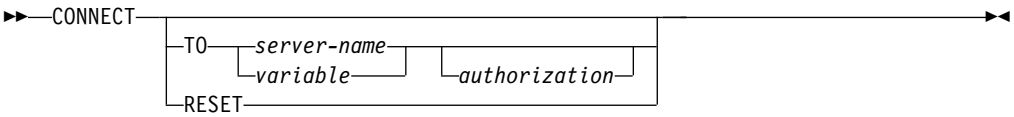
アプリケーション・サーバーが Db2 for i である場合は、次の場合に該当しない限り、ステートメントの発行者のプロファイル ID をアプリケーション・サーバー・システム上でも有効なユーザー・プロファイルにする必要があります。

- USER が指定されている。 USER が指定されている場合は、USER 文節で、アプリケーション・サーバー・システム上の有効なユーザー・プロファイルを指定する必要があります。
- TCP/IP が、アプリケーション・サーバーのサーバー権限記入項目で使用されている。この場合は、サーバー権限記入項目で、アプリケーション・サーバー・システム上の有効なユーザー・プロファイルを指定する必要があります。

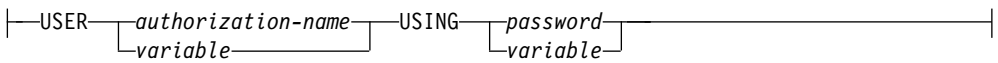
ステートメントでグローバル変数を参照する場合は、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別されるグローバル変数に対して、
  - そのグローバル変数に対する READ 特権
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

構文



authorization:



説明

TO *server-name* または *variable*

指定したサーバー名、または指定した変数に入っているサーバー名によってアプリケーション・サーバーを識別します。スキーマ名で修飾すれば、グローバル変数を使用することもできます。変数を指定する場合、

- CHAR または VARCHAR ホスト変数でなければなりません。
- 標識変数を伴っていてはなりません。
- サーバー名は、その変数内で左寄せし、通常 ID の形成の規則に従っていなければなりません。
- サーバー名の長さが、変数の長さよりも短い場合、右側をブランクで埋めなければなりません。
- サーバー名の値には小文字を含めてはなりません。

この CONNECT ステートメントが実行される場合、指定したサーバー名または変数に入っているサーバー名は、ローカル・ディレクトリーに記述されているアプリケーション・サーバーを識別する必要があります。

以下の説明で S は、指定したサーバー名または変数に入っているサーバー名を表しています。 S は、該当のアプリケーション・プロセスの既存の接続を識別していてはなりません。

USER *authorization-name* または *variable*

アプリケーション・サーバーへの接続に使用される権限名を識別します。スキーマ名で修飾すれば、グローバル変数を使用することもできます。

変数 を指定する場合、

- CHAR または VARCHAR ホスト変数でなければなりません。
- 標識変数を伴っていてはなりません。権限名は、その変数に左寄せして入れ、権限名の命名規則に従っていなければなりません。
- 権限名の長さが、変数の長さよりも短い場合には、右側をブランクで埋めなければなりません。

USING *password* または *variable*

アプリケーション・サーバーへの接続に使用されるパスワードを識別します。

パスワードをリテラルとして指定する場合、そのリテラルは文字ストリングでなければなりません。最大長は 128 文字です。左寄せしなければなりません。リテラル形式のパスワードは、静的 SQL または REXX では許可されていません。

変数 を指定する場合、

- この変数は、グローバル変数にすることはできません。
- CHAR または VARCHAR ホスト変数でなければなりません。
- 標識変数を伴っていてはなりません。
- パスワードは、変数に左寄せして入れなければなりません。
- パスワードの長さが変数の長さよりも短い場合には、右側をブランクで埋めなければなりません。

RESET

CONNECT RESET は、CONNECT TO *x* と同等です。ここで、*x* はローカル・サーバー名です。

CONNECT (オペランドの指定なし)

この形式の CONNECT ステートメントは、現行サーバーについての情報を戻し、接続状態、オープン・カーソル、準備されたステートメント、またはロックに対してまったく影響を与えません。接続情報は、SQL 診断域 (または SQLCA) にある接続情報項目に戻されます。

さらに、SQL 診断域にある DB2_CONNECTION_STATUS 接続情報項目 (または SQLCA のフィールド SQLERRD(3)) には、該当の作業単位の接続の状況を示す値が入れられます。次の値のいずれかが入れられます。

- 1 - この作業単位の接続では、コミット可能な更新を行うことができる。
- 2 - この作業単位の接続では、コミット可能な更新を行うことはできない。

注

接続成功**:** CONNECT ステートメントが正常に完了した場合 :

- アプリケーション・サーバー S への接続が作成され、現行および保留状態に置かれます。それ以前の接続は (存在する場合)、休止状態になります。
- S が特殊レジスター CURRENT SERVER に入れられます。
- アプリケーション・サーバーについての情報は、 SQL 診断領域の接続情報項目に入れられます。
- アプリケーション・サーバー S に関する情報も、SQLCA のフィールド SQLERRP および SQLERRD(4) に入れられます。そのアプリケーション・サーバーが IBM リレーショナル・データベース・プロダクトである場合は、フィールド SQLERRP の中の情報は、*pppvvrrm* の形式をとります。ただし、
  - *ppp* は、製品を次のように識別します。
    - ARI (Db2 UDB サーバー (VM および VSE 版) の場合)
    - DSN (Db2 for z/OS の場合)
    - QSQ (Db2 for i の場合)
    - 他のすべての Db2 プロダクトの場合は SQL
  - *vv* は、2 桁のバージョン ID です (例えば、'09' など)。

- *rr* は、2 桁のリリース ID です (例えば、'01' など)。
- *m* は、「0」のような 1 文字の修正レベルを示します。

例えば、アプリケーション・サーバーがバージョン 9 の Db2 for z/OS の場合は、SQLERRP の値は「DSN09010」になります。

SQLCA の SQLERRD(4) フィールドには、アプリケーション・サーバー S がコミット可能な更新の実行を許可するかどうかを示す値が入ります。以下は、この CONNECT に関して SQLCA のフィールド SQLERRD(4) に入れられる値とその意味を示しています。

- 1 - コミット可能な更新を行うことができる。会話は、無保護会話です。[88]
- 2 - コミット可能な更新は行うことができない。会話は、無保護会話です。
- 3 - コミット可能な更新を行うことができるか否かは不明である。会話は、保護会話です。
- 4 - コミット可能な更新を行うことができるか否かは不明である。会話は、無保護会話です。
- 5 - コミット可能な更新を行うことができるか否かは不明である。接続は、ローカル接続、またはアプリケーション・リクエスターのドライバー・プログラムとの接続です。

- 接続についての追加情報は SQLCA のフィールド SQLERRMC に入れられます。 1867 ページの『付録 C. SQLCA (SQL 連絡域)』を参照してください。

接続失敗**:** CONNECT ステートメントが正常に実行されなかった場合は、その活動化グループの接続状態、およびその接続の状態は変わりません。

暗黙接続**:** 暗黙接続では、常にアプリケーション・リクエスター・ジョブの権限名が送信され、パスワードは送信されません。アプリケーション・サーバー・ジョブの権限名 が異なる場合や、パスワードを送る必要がある場合は、明示的な接続ステートメントを使用する必要があります。

TCP/IP を RDB との接続に使用している場合、暗黙の接続は上記の制約に拘束されることはありません。 ADDSVRAUTE コマンドと他の -SVRAUTE コマンドを使用すると、暗黙 (または明示) の CONNECT の実行元である所定のユーザーに対して、該当の RDB との接続で使用されるリモート権限名とパスワードを指定することが可能になります。

パスワードが ADDSVRAUTE コマンドや CHGSVRAUTE コマンドで保管されるようにするには、QRETSVRSEC システム値をデフォルト値の '0' ではなく、'1' に設定する必要があります。これらのコマンドを DRDA 接続に使用している場合に非常に重要なことは、SERVER パラメーターに RDB 名の値を英大文字で入力しなければならないと認識しておくことです。詳細については、タイプ 2 の CONNECT の節の例 2 を参照してください。

暗黙的な接続について詳しくは、「SQL プログラミング」を参照してください。ユーザー・プロファイルとリレーショナル・データベースとの接続が一度確立されると、同じユーザー・プロファイルと同じリレーショナル・データベースとの以後の接続では、パスワード (指定がある場合) の妥当性が再検証されない場合がありま

す。パスワードの妥当性の再検証は、会話がまだ活動状態であるかどうかによって異なります。詳しくは、「分散データベース・プログラミング」を参照してください。

SET SESSION AUTHORIZATION: SET SESSION AUTHORIZATION ステートメントがスレッド内で実行された場合、接続ステートメントより前に SYSTEM_USER 値が SESSION_USER と同じでなければ、ローカル・サーバーへの CONNECT は失敗します。

これには、ACTGRP(*NEW) を指定するプログラムを呼び出すことによる暗黙的な接続が含まれます。

例

例 *1:* SQL ステートメントを TOROLAB および SVLLAB で実行します。最初の CONNECT ステートメントは TOROLAB との接続を確立し、2 番目の CONNECT ステートメントはその接続を休止状態にします。

```
EXEC SQL  CONNECT TO TOROLAB;

  (TOROLAB にあるオブジェクトを参照するステートメントを実行)

EXEC SQL  CONNECT TO SVLLAB;

  (SVLLAB にあるオブジェクトを参照するステートメントを実行)
```

例 *2:* リモート・サーバーに接続してユーザー ID およびパスワードを指定し、そのユーザーとして作業を行ってから、今度は別のユーザーとして接続し、さらに作業を行います。

```
EXEC SQL  CONNECT TO SVLLAB USER :AUTHID USING :PASSWORD;

  (サーバー上のデータにアクセスする SQL ステートメントを実行)

EXEC SQL COMMIT;

  (AUTHID および PASSWORD を新規の値に設定する)

EXEC SQL  CONNECT TO SVLLAB USER :AUTHID USING :PASSWORD;

  (サーバー上のデータにアクセスする SQL ステートメントを実行)
```

例 *3:* ユーザー JOE は、パスワードが SHIBBOLETH のユーザー ID ANONYMOUS によって TOROLAB3 に接続し、SQL ステートメントを実行したいとします。 TOROLAB3 の RDB ディレクトリー項目では、その接続タイプに *IP を指定します。

アプリケーションを実行する前に、ある種のセットアップを行う必要があります。

IBM i オペレーティング・システムでサーバーのセキュリティー情報を保存できるようにするために、以下のコマンドをまだ実行していなければ、まずこのコマンドを実行する必要があります。

```
CHGSYSVAL SYSVAL(QRETSVRSEC) VALUE('1')
```

このコマンドによって、必須のサーバー権限項目が追加されます。

```
ADDSVRAUTE USRPRF(JOE) SERVER(TOROLAB3) USRID(ANONYMOUS) +
           PASSWORD(SHIBBOLETH)
```

JOE のユーザー・プロファイルのもとで実行されるこのステートメントは、この時点で、必要な接続を確立します。

```
EXEC SQL CONNECT TO TOROLAB3;
(TOROLAB3 にあるオブジェクトを参照するステートメントを実行)
```

CREATE ALIAS

CREATE ALIAS ステートメントは、現行またはリモート・サーバーにあるデータベース・ファイルの表、表のパーティション、ビュー、またはメンバーの別名を定義します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- スキーマ内に作成する特権。詳しくは、スキーマ内で作成する必要のある権限を参照してください。
- データベース管理者権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 次のシステム権限
  - DDM ファイル作成 (CRTDDMF) コマンドに対する *USE
- データベース管理者権限

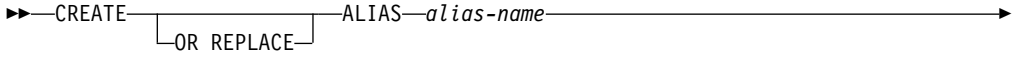
SQL 名が指定され、別名が作成されるライブラリーと同じ名前のユーザー・プロファイルが存在し、しかも、その名前がステートメントの権限 ID と異なっている場合、ステートメントの権限 ID が保持している特権には、少なくとも次のいずれか 1 つを含める必要があります。

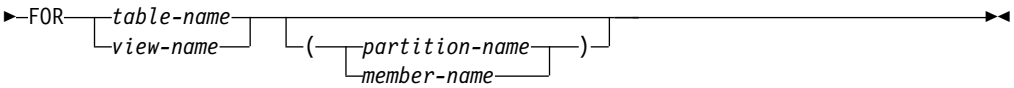
- その名前を持つユーザー・プロファイルに対する *ADD システム権限
- データベース管理者権限

既存の別名に置き換えるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

- 次のシステム権限
  - 別名に対する *OBJMGT システム権限
  - この別名を削除するために必要な全権限
- データベース管理者権限

構文





説明

OR REPLACE
現行サーバーにこの別名の定義が存在する場合に、その定義を置き換える、という動作を指定します。実際には、カタログで既存の定義を削除してから新しい定義に置き換える、という動作になりますが、例外として、この別名に対して与えられていた特権は影響を受けません。現行サーバーにこの別名の定義が存在しなければ、このオプションは無視されます。

alias-name
別名を指定します。暗黙的または明示的修飾子も含め、この名前は、現行サーバーに既に存在している索引、表、ビュー、別名、またはファイルと同じ名前にすることはできません。

別名 を修飾する場合は、2 部構成または 3 部構成の名前にすることができます。スキーマ名として、システム・スキーマは使用しないでください。3 部構成の名前の名前を使用する場合は、最初の部分でリレーショナル・データベース・ディレクトリーにあるリレーショナル・データベース名を指定する必要があります。

リモート・リレーショナル・データベースへの接続およびローカル・ディレクトリーについて詳しくは、「SQL プログラミング」および「分散データベース・プログラミング」を参照してください。

SQL 名が指定されている場合、別名は、暗黙的または明示的修飾子で指定しているスキーマ内に作成されます。
システム名が指定されている場合、別名は、修飾子で指定しているスキーマ内に作成されます。修飾されていない場合、別名は、その別名の作成の対象である表かビューと同じスキーマ内に作成されます。表が修飾されず、しかも別名の作成時に存在していなかった場合:

- CURRENT SCHEMA 特殊レジスターの値が *LIBL である場合、別名は、現行ライブラリー (*CURLIB) 内に作成されます。
- そうでない場合、別名は現行スキーマ内に作成されます。

別名が有効なシステム名でない場合、Db2 for i は、システム名を生成します。名前の生成に関する規則については、 1298 ページの『表名の生成の規則』を参照してください。

FOR 表名またはビュー名
別名の定義の対象である現行またはリモート・サーバーの表やビューを識別します。別名を指定することはできません (別名で別の別名を参照することはできません)。ただし、リモート・サーバーにある別名を参照する名前であれば、指定できます。

非ローカルのリレーショナル・データベース名を使用して 3 部構成の名前を指定する場合は、*alias-name* を *table-name* または *view-name* と同じ名前にする必要があります。

表名 やビュー名 では、別名の作成時に存在していた表やビューを識別する必要はありません。表やビューが別名の作成時に存在していない場合は、警告が戻されます。別名の使用時に表やビューが存在していない場合は、エラーが戻されます。

SQL 名が指定されており、表名 またはビュー名 が修飾されていない場合の修飾子は、暗黙の修飾子です。詳しくは、 62 ページの『命名規則』を参照してください。

システム名が指定されており、表名 やビュー名 が修飾されておらず、しかも別名の作成時に存在していない場合、表名 やビュー名 は、別名が作成されるライブラリーによって修飾されます。

partition-name
パーティション化された表のパーティションを識別します。

パーティションを指定した場合、別名を SQL スキーマ・ステートメント内で使用することはできません。パーティションを指定しなかった場合、表内のすべてのパーティションが別名に組み込まれます。

パーティション名が指定される場合、 3 部構成の名前を FOR 節に指定してはなりません。3 部構成の名前が指定され、別名のリレーショナル・データベースと異なるリレーショナル・データベースを示している場合、別名を使用しようとすると失敗します。

メンバー名
データベース・ファイルのメンバーを識別します。メンバー名が指定されず、表がパーティション化された表ではない場合、*FIRST が使用されます。メンバー名が指定されず、表がパーティション化された表である場合、すべてのメンバー (パーティション) が使用されます。

メンバーを指定した場合、ほとんどの SQL スキーマ・ステートメント内では別名を使用することはできません。これは、CREATE PROCEDURE 内、CREATE FUNCTION 内、および *as-result-table* 節を指定する CREATE TABLE 内で使用できます。

メンバー名が指定される場合、3 部構成の名前を FOR 節に指定してはなりません。3 部構成の名前が指定され、別名のリレーショナル・データベースと異なるリレーショナル・データベースを示している場合、別名を使用しようとすると失敗します。

注

別名参照**:** システム名または SQL 名を参照するように別名を定義することができます。システム名は作成処理中に生成されるため、一般的には SQL 名を指定することが推奨されます。

ただし、別名が 3 部構成の名前への参照を指定していて、別名がネイティブ・コマンドまたはネイティブ・アクセスで DDM ファイルとして使用される予定の場合、指定される名前はシステム名でなければなりません。

データベース・ファイル・オーバーライド (OVRDBF) CL コマンドも使用すれば、データベース・マネージャーは、データベース・ファイルの個々のメンバーを処理することができるようになります。しかし、表のパーティションまたはデータベー

ス・ファイルのメンバーに対する別名を作成する方が、オーバーライドを実行する必要がなくなるため、簡単でしかもパフォーマンスも向上します。

別名の属性 **:** 別名は特殊形式の DDM ファイルとして作成されます。 SQL では、別名と通常の DDM ファイルの両方を使用できますが、その別名または DDM ファイルで非ローカルのリレーショナル・データベース名を指定する場合は、*table-name* または *view-name* で別名と同じ名前を指定する必要があります。

分散表を介して作成される別名は、現行サーバー上でのみ作成されます。 分散表についての詳細は、Db2 マルチシステム (Multisystem) を参照してください。

別名の所有権**:** SQL 名が指定されている場合、

- 作成した別名が入れられるスキーマと同じ名前のユーザー・プロファイルが存在する場合、別名の所有者 はそのユーザー・プロファイルです。
- その他の場合は、別名の所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

システム名を指定した場合は、別名の所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

別名の権限 **:** SQL 名を使用する場合は、別名は、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成されます。システム名を使用する場合、別名は、スキーマの作成権限 (CRTAUT) パラメーターによって決められる *PUBLIC に対する権限を使用して作成されます。

別名の所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、その別名に対する権限が与えられます。

REPLACE の規則**:** REPLACE によって別名を再作成する場合は、以下のようになります。

- 既存のコメントまたはラベルは破棄されます。
- 権限を持つユーザーは維持されます。オブジェクト所有者は変更される可能性があります。
- 現在のジャーナル監査は保持されます。

パッケージと **3** 部構成の別名**:** アプリケーションがリモート・オブジェクトと DRDA アクセスのために 3 部構成の名前を別名として使用する場合は、そのアプリケーション・プログラムのパッケージが、その 3 部構成の名前で指定されているそれぞれの場所に存在していなければなりません。パッケージを明示的に作成するには、CRTSQLPKG CL コマンドを使用します。3 部構成の名前を別名として参照するときに、パッケージが存在していなければ、データベース・マネージャーが暗黙的にパッケージを作成しようとします。

代替構文**:** 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード SYNONYM は、ALIAS の同義語として使用することができます。

例

例 1: PROJECT 表に対して、CURRENT_PROJECTS という別名を作成します。

```
CREATE ALIAS CURRENT_PROJECTS
  FOR PROJECT
```

例 2: SALES 表の JANUARY パーティションに対して、SALES_JANUARY という別名を作成します。 SALES 表には、12 のパーティション (1 年のそれぞれの月ごとに 1 つずつ) があります。

```
CREATE ALIAS SALES_JANUARY
  FOR SALES(JANUARY)
```

例 3: リレーショナル・データベース USARDB にあるスキーマ REPORTS の SALES 表の別名として、REPORTS.SALES という別名を作成します。

```
CREATE ALIAS REPORTS.SALES
  FOR USARDB.REPORTS.SALES
```

CREATE FUNCTION

CREATE FUNCTION ステートメントは、現行サーバーでユーザー定義関数を定義します。

定義できる関数のタイプは以下のとおりです。

- 外部スカラー

  このタイプの関数は、C または Java などのプログラミング言語で書かれ、スカラー値を戻します。この外部プログラムは、現行サーバーで定義されている関数により、その関数の各種属性に基づいて参照されます。 1076 ページの『CREATE FUNCTION (外部スカラー)』を参照してください。
- 外部表

  このタイプの関数は、C または Java などのプログラミング言語で書かれ、一組の行を戻します。この外部プログラムは、現行サーバーで定義されている関数により、その関数の各種属性に基づいて参照されます。 1100 ページの『CREATE FUNCTION (外部表)』を参照してください。
- ソース化

  このタイプの関数は、既に現行サーバーに存在している他の関数 (組み込み、外部、ソース化、または SQL) を呼び出すことによりインプリメントされます。ソース化関数は、スカラーの結果、または集約関数の結果を戻します。 1122 ページの『CREATE FUNCTION (ソース派生)』を参照してください。この関数は、基礎となっているソース関数の属性を継承します。
- **SQL** スカラー

  このタイプの関数は SQL のみで書かれるもので、スカラー値を戻します。関数本体は、関数の各種属性と一緒に現行サーバーで定義されます。 1135 ページの『CREATE FUNCTION (SQL スカラー)』を参照してください。
- **SQL** 表

  このタイプの関数は SQL のみで書かれるもので、一組の行を戻します。関数本体は、関数の各種属性と一緒に現行サーバーで定義されます。 1151 ページの『CREATE FUNCTION (SQL 表)』を参照してください。

注

スキーマおよび関数名の選択**:** 修飾付き関数名を指定する場合は、スキーマ名 はシステム・スキーマ (6 ページの『スキーマ』を参照) のいずれかであってはなりません。関数名が修飾されていない場合、それはデフォルトのスキーマ名で暗黙的に修飾されます。

非修飾名は、区切り文字付き ID として指定される場合でも、システム使用のために予約されている以下の名前のいずれかであってはなりません。

=	<	>	>=
<=	<>	¬=	¬<
¬<	!=	!<	!>
ALL	FALSE	POSITION	XMLAGG

AND	FOR	RID	XMLATTRIBUTES
ANY	FROM	RRN	XMLCOMMENT
ARRAY_AGG	HASHED_VALUE	SELECT	XMLCONCAT
BETWEEN	IN	SIMILAR	XMLDOCUMENT
BOOLEAN	IS	SOME	XMLELEMENT
CASE	LIKE	STRIP	XMLFOREST
CAST	MATCH	SUBSTRING	XMLGROUP
CHECK	NODENAME	TABLE	XMLNAMESPACES
DATAPARTITIONNAME	NODENUMBER	THEN	XMLPARSE
DATAPARTITIONNUM	NOT	TRIM	XMLPI
DBPARTITIONNAME	NULL	TRUE	XMLROW
DBPARTITIONNUM	ONLY	TYPE	XMLSERIALIZE
DISTINCT	OR	UNIQUE	XMLTEXT
EXCEPT	OVERLAPS	UNKNOWN	XMLVALIDATE
EXISTS	PARTITION	WHEN	XSLTRANSFORM
EXTRACT			

パラメーターの定義**:** 関数の入力パラメーターは括弧内のリストとして指定されます。

CREATE FUNCTION で許容されているパラメーターの最大数は 2000 です。

関数には入力パラメーターがなくても構いません。この場合は、次のように、中が空の 1 組の括弧をコーディングする必要があります。

```
CREATE FUNCTION WOOFER()
```

関数の結果のデータ・タイプは、その関数の RETURNS 文節で指定されます。

- パラメーターのデータ・タイプの選択**:** 関数の入力および結果パラメーターのデータ・タイプを選択する場合は、それらのパラメーターの値に影響を与える可能性のあるプロモーションの規則を考慮する必要があります。 107 ページの『データ・タイプのプロモーション』を参照してください。例えば、関数の入力引数の 1 つである定数には、その関数で予期していたデータ・タイプとは別の組み込みデータ・タイプが指定される場合があり、さらに重要なことに、その定数は、予期されていたデータ・タイプにプロモートできない可能性があります。プロモーションの規則に従って、次のデータ・タイプを使用することをお勧めします。
  - SMALLINT に代わって INTEGER
  - REAL に代わって DOUBLE
  - CHAR に代わって VARCHAR
  - GRAPHIC に代わって VARGRAPHIC

  Db2 for i 以外のプラットフォーム間で関数を移植可能にするために、以下のデータ・タイプ名は使用しないでください。これらは、異なるプラットフォームでは表記が異なる場合があります。
  - FLOAT。この代わりに、DOUBLE や REAL を使用すること。
  - NUMERIC。この代わりに、DECIMAL を使用すること。
- パラメーターに **AS LOCATOR** を指定する**:** 値の代わりにロケーターを渡すことにより、関数との間で受け渡しするバイト数を削減できることがあります。これは、パラメーターの値が非常に大きい場合に便利です。 AS LOCATOR 文節は、実際の値の代わりにパラメーターの値へのロケーターを渡すことを指定しま

す。 AS LOCATOR を指定するのは、LOB データ・タイプまたは XML データ・タイプ、あるいは LOB データ・タイプまたは XML データ・タイプに基づく特殊タイプのパラメーターに対してのみであり、しかも、LANGUAGE JAVA が有効になっていない場合に限られます。

AS LOCATOR 文節によって、データ・タイプがプロモート可能かどうかの決定が影響を受けることも、関数解決に使用される関数シグニチャーが影響を受けることもありません。

SQL 関数には、AS LOCATOR は指定できません。

スキーマ内で関数の固有性を判別する: それぞれの関数の関数シグニチャーが固有であれば、スキーマ内で複数の関数に同じ名前を指定することができます。関数シグニチャーは、入力パラメーターの数およびデータ・タイプと結合された修飾関数名です。名前、スキーマ名、パラメーターの数、およびそれぞれのパラメーターのデータ・タイプ (データ・タイプの長さ、精度、位取り、または CCSID といった他の属性に関係なく) の組み合わせで、現行サーバー上に存在しているユーザー定義の関数を識別してはなりません。戻りタイプは、関数の固有性の判別に影響しません。異なる 2 つのスキーマのそれぞれに、それらに対応しているすべてのデータ・タイプに同じデータ・タイプが指定されている同じ名前の 1 つの関数を入れることができるということです。ただし、それらに対応しているすべてのデータ・タイプに、同じデータ・タイプが指定されている同じ名前の関数を 2 つ入れることはできません。

対応しているデータ・タイプが一致しているか否かの判別時に、データベース・マネージャーは、比較において、長さ、精度、または位取りの属性はどれも考慮に入れません。データベース・マネージャーは、データ・タイプの同義語を一致と見なします。例えば、REAL と FLOAT、ならびに DOUBLE と FLOAT を一致と見なします。したがって、CHAR(8) と CHAR(35) は同じであると見なされ、同様に、DECIMAL(11,2) と DECIMAL(4,3) は同じであると見なされます。さらに、文字タイプとグラフィック・タイプは同じであると見なされます。例えば、以下は同じタイプであると見なされます。 CHAR と GRAPHIC、VARCHAR と VARGRAPHIC、および CLOB と DBCLOB。 CHAR(13) と GRAPHIC(8) は同じタイプであると見なされます。作成中の関数のシグニチャーが、同じ名前とスキーマを持つ既存のユーザー定義関数のシグニチャーと重複している場合、エラーが戻されます。

次のステートメントを実行して、同じスキーマ内に 4 つの関数を作成すると想定します。 2 番目と 4 番目のステートメントは、1 番目と 3 番目のステートメントが作成した関数と重複している関数を作成するので失敗します。

```
CREATE FUNCTION PART (INT, CHAR(15) ...
CREATE FUNCTION PART (INTEGER, CHAR(40) ...

CREATE FUNCTION ANGLE (DECIMAL(12,2)) ...
CREATE FUNCTION ANGLE (DEC(10,7)) ...
```

関数に特定の名前を指定する: 名前もスキーマも同じである (ただしパラメーター・リストは異なる) 複数の関数を定義するときは、特定名も指定することをお勧めします。関数のソース化、除去、または関数へのコメントの付加を行うときに、特定名を使用して、その関数を一意的に識別することができます。ただし、この関数をその特定名で呼び出すことはできません。

この特定名は、暗黙的または明示的にスキーマ名で修飾されます。 CREATE FUNCTION でスキーマ名が指定されていない場合、特定名は関数名 (関数名) の明示的または暗黙的なスキーマ名と同じ名前になります。スキーマ名が指定されている場合、特定名は関数名の明示的または暗黙的なスキーマ名と同じにしなければなりません。スキーマ名も含め、この名前は、別のプロシージャーや現行サーバーに存在しているプロシージャーの特定名を示すものであってはなりません。

特定名を指定しなかった場合、その特定名は、関数名に設定されます。この特定名の関数やプロシージャーが既に存在している場合は、固有表名の生成に使用される規則にほぼ準拠した固有名が生成されます。

組み込み関数の拡張またはオーバーライド:

組み込み関数の拡張またはオーバーライドが必要な場合を除き、ユーザー定義の関数に組み込み関数と同じ名前を付けることはお勧めできません。

- 既存の組み込み関数の機能の拡張:

  組み込み関数と同じ名前の新しいユーザー定義の関数と、固有の関数シグニチャーを作成します。例えば、組み込み数値タイプの代わりに特殊タイプ MONEY を入力として受け入れる、組み込み関数 ROUND に似たユーザー定義の関数が必要になったとします。この場合、ROUND という名前の新しいユーザー定義関数のシグニチャーは、組み込み関数 ROUND がサポートするどの関数シグニチャーとも異なるものになります。
- 組み込み関数をオーバーライドする:

  既存の組み込み関数のいずれかと名前もシグニチャーも同じである新しいユーザー定義の関数を作成します。この新しい関数は、その組み込み関数の対応するパラメーターと同じ名前およびデータ・タイプを持ちますが、インプリメントされるロジックが異なります。例えば、組み込み関数 ROUND に類似しているが、丸めの規則が異なるユーザー定義の関数が必要になったとします。この場合、ROUND という名前の新しいユーザー定義関数のシグニチャーは、組み込み関数 ROUND がサポートするシグニチャーのどれかと同じになります。

  組み込み関数をオーバーライドした後に、SQL パスの中で新しい関数のスキーマがシステム・スキーマの前に表示されると、データベース・マネージャーは、組み込み関数ではなくユーザー定義関数を選択する可能性があります。非修飾関数名を使用しているアプリケーションが、前回はその名前の組み込み関数を使用して正常に実行されたのに、今回は失敗するという状況が生じることがあります。さらに事態が悪化すると、一見して正常に実行されたように見えるのに、実際には、データベース・マネージャーがその組み込み関数ではなくユーザー定義の関数の方を選択したため、意図しない結果が生じるという状況が発生することもあります。

  組み込み集約関数をソースにしたユーザー定義関数の呼び出し時に DISTINCT キーワードを渡すこともできます。例えば、組み込みの AVG 関数をソースにした MY_AVG というユーザー定義関数があるとします。そのユーザー定義関数を `MY_AVG (DISTINCT expression)` で呼び出すことも可能です。その場合は、基礎になっている組み込みの AVG 関数が DISTINCT キーワードで呼び出されることになります。

関数内の特殊レジスター**:** 呼び出し側の特殊レジスターの設定値は呼び出し時に関数によって継承され、呼び出し側への戻りにおいてリストアされます。 SQL ステートメントを実行できる関数内で特殊レジスターが変更されることもありますが、それらの変更は呼び出し元には影響しません。

セキュア関数の作成**:** Db2 は SECURED 属性を、ユーザー定義関数に対するすべての変更の監査手順をユーザーが確立したことを宣言するアサーションとして扱います。Db2 は、そのような監査制御手順が後続のすべての ALTER FUNCTION ステートメントに対して確立していると想定します。

セキュアな関数内での他のユーザー定義関数の呼び出し**:** 行アクセス制御または列アクセス制御を使用している表を参照する SQL ステートメント内で、セキュアなユーザー定義関数が参照されていて、そのセキュアなユーザー定義関数が他のユーザー定義関数を呼び出す場合、ネストされたユーザー定義関数はセキュアであるとは判定されません。それらのネストされた関数が機密データにアクセスする可能性がある場合、セキュリティー管理者権限を持つ認可されたユーザーは、それらの関数がそのデータにアクセスすることを許可されていること、および、それらの関数に加えられるすべての変更に関して変更管理監査手順が確立されていることを確認する必要があります。

MODIFIES SQL DATA および **EXTERNAL ACTION** 関数**:** MODIFIES SQL DATA または EXTERNAL ACTION 関数が最外部の選択リスト以外で呼び出されると、使用されるアクセス・プランによって関数を呼び出す回数が変わるため、結果は予測できません。

関数および借用権限**:** 隔離される関数は、分離スレッドで実行します。NOT FENCED 関数 (隔離されない関数) に対して ALLOW PARALLEL が指定される場合、ALLOW PARALLEL 関数も分離スレッドで実行する可能性があります。分離スレッドで実行する関数は、呼び出し側アプリケーションによって指定されることのある借用権限の下では実行しません。

リストアの考慮事項**:** 関数に関連するプログラムまたはサービス・プログラムが保管された後でリストアされ、オブジェクトが関数作成時に関数属性で更新されていた場合、保管されていた属性がリストア中に処理され、変更される可能性があります。

プログラムまたはサービス・プログラムの「保管されたライブラリー」(SAVLIB) が「復元先ライブラリー」(RSTLIB) と異なる場合、関数のスキーマ名、固有スキーマ名、および外部名は、リストアの結果として変更される可能性があります。

- 保管された関数スキーマ名と保管されたオブジェクトのライブラリー名が一致する場合、関数スキーマは「復元先ライブラリー」(RSTLIB) に変更されます。それ以外の場合、関数スキーマ名は、保管された関数スキーマ名です。
- 固有スキーマ名は、常に関数スキーマ名と同じです。
- 保管された EXTERNAL NAME ライブラリーと保管されたオブジェクトのライブラリー名が一致する場合、EXTERNAL NAME ライブラリーは「復元先ライブラリー」(RSTLIB) に変更されます。それ以外の場合、EXTERNAL NAME ライブラリーは、保管されたライブラリー名です。保管された EXTERNAL NAME ライブラリーが *LIBL の場合は、変更されません。

同じ関数シグニチャーがカタログ内に既に存在する場合:

- 外部プログラム名またはサービス・プログラム名が、カタログ内に既に存在する名前と同じである場合、そのプロシージャーについてのカタログ内の情報は、保管された属性 (固有名を含む) で置換されます。
- それ以外の場合、保管された属性はリストアされず、警告 (SQL9015) が発行されます。

同じ固有名が既にカタログ内に存在する場合、警告が発行され、新しい固有名が生成されます。それ以外の場合、関数の固有名が保持されます。

CREATE FUNCTION (外部スカラー)

CREATE FUNCTION (外部スカラー) ステートメントは、現行サーバー上に外部スカラー関数を定義します。ユーザー定義の外部スカラー関数は、呼び出されるたびに単一値を戻します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- SYSFUNCS カタログ・ビューと SYSPARMS カタログ表の場合 [89]:
  - 該当の表に対する INSERT 特権、および
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

外部プログラムやサービス・プログラムが存在している場合、このステートメントの権限 ID が保持する特権には、少なくとも次のいずれか 1 つを含める必要があります。

- SQL ステートメントで参照された外部プログラムやサービス・プログラムの場合
  - その外部プログラムやサービス・プログラムが含まれるスキーマに対する USAGE 特権
  - その外部プログラムやサービス・プログラムに対するシステム権限の *EXECUTE。
  - そのプログラムやサービス・プログラムに対するシステム権限の *CHANGE。システムには、プログラム・オブジェクトを更新し、関数を別のシステムに保管/復元するために必要な情報を入れる場合にこの権限が必要となります。ユーザーにこの権限が与えられていない場合、関数は同じように作成されますが、プログラム・オブジェクトは更新されません。
- データベース管理者権限

SQL 名が指定され、関数が作成されるライブラリーと同じ名前のユーザー・プロファイルが存在し、しかも、その名前がステートメントの権限 ID と異なっている場合、ステートメントの権限 ID が保持している特権には、少なくとも次のいずれか 1 つを含める必要があります。

- その名前を持つユーザー・プロファイルに対する *ADD システム権限
- データベース管理者権限

特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

89. GRTOBJAUT CL コマンドを使用してこれらの特権を付与する必要があります。

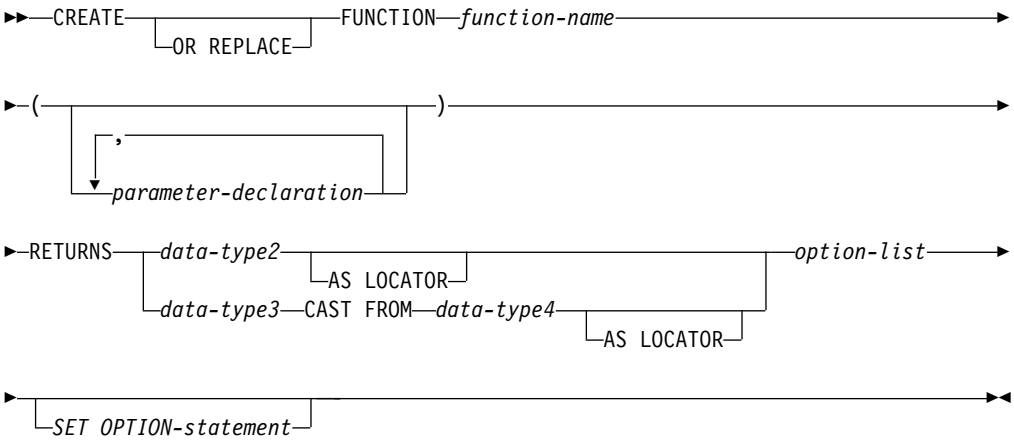
- ステートメント内で識別された、それぞれの特殊タイプごとに、
  - その特殊タイプに対する USAGE 特権、および
  - 特殊タイプが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

既存の関数に置き換えるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

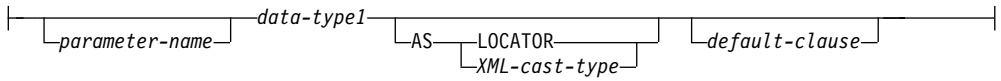
- 次のシステム権限
  - この関数に関連したサービス・プログラム・オブジェクトに対する *OBJMGT システム権限
  - この関数を削除するために必要な全権限
  - SYSFUNCS カタログ・ビューと SYSPARMS カタログ表に対する *READ システム権限
- データベース管理者権限

SQL 特権に対応するシステム権限については、『表またはビューへの権限を検査する際の対応するシステム権限』および 『特殊タイプへの権限を検査する際の対応するシステム権限』を参照してください。

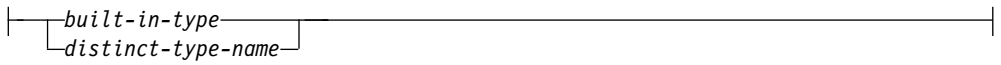
構文



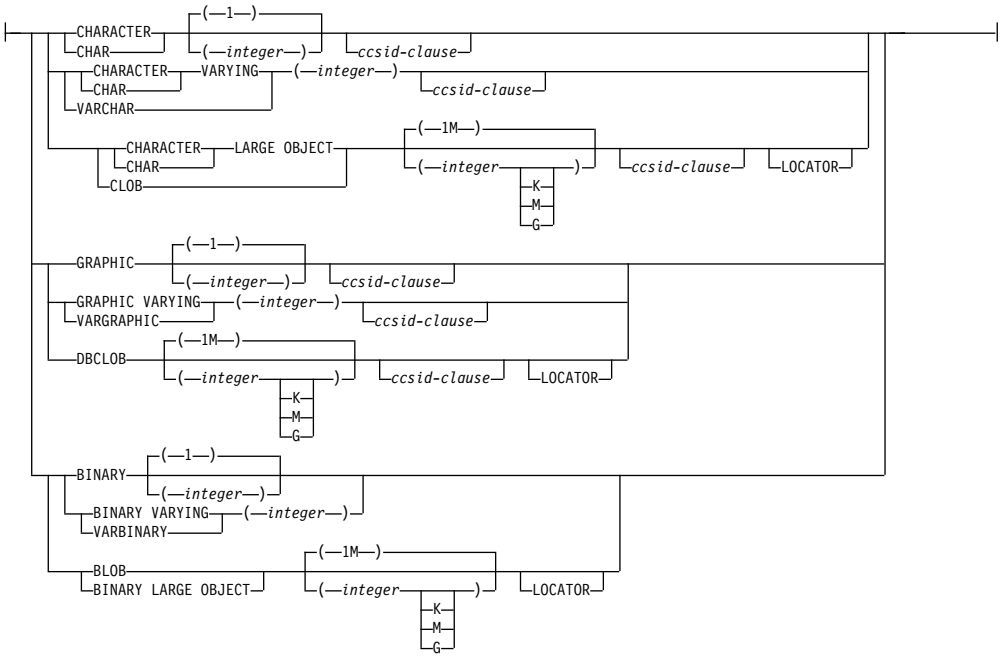
parameter-declaration:



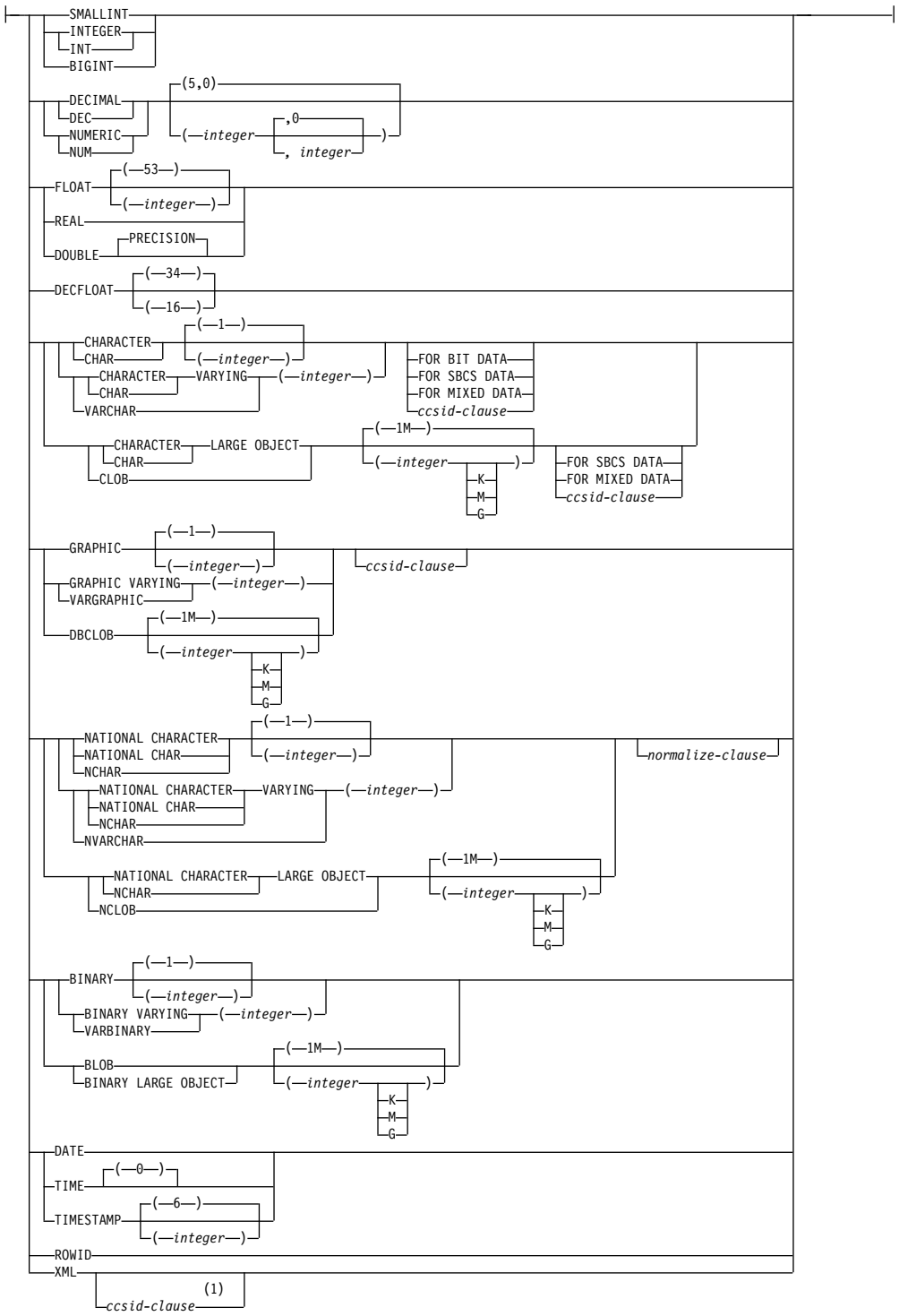
data-type1, data-type2, data-type3, data-type4:



XML-cast-type:



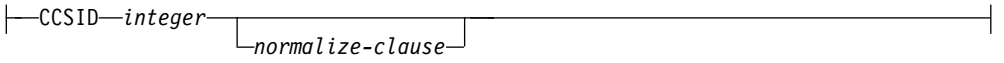
built-in-type:



注:

1 XML の *ccsid-clause* は、*data-type2* および *data-type3* でのみ使用できます。

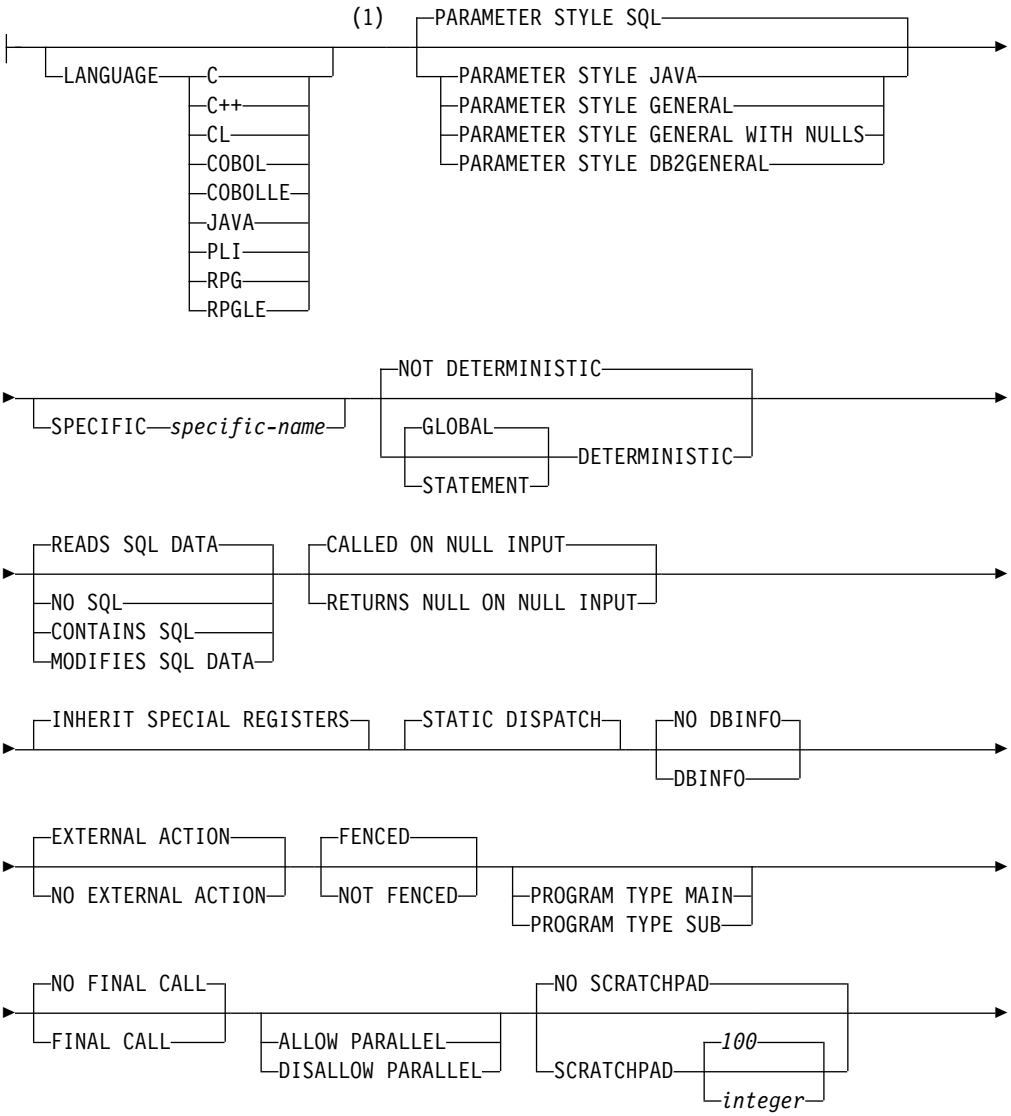
ccsid-clause:

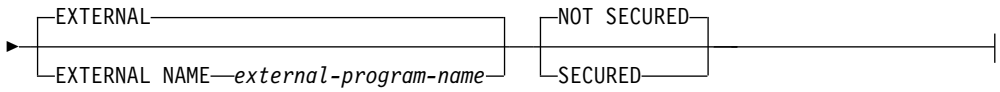


normalize-clause:

default-clause:

option-list:





注:

1 この文節とこの後の option-list にある文節は、どのような順番で指定してもかまいません。各文節を指定できるのは、それぞれ 1 回だけです。

説明

OR REPLACE

現行サーバーにこの関数の定義が存在する場合に、その定義を置き換える、という動作を指定します。実際には、カタログで既存の定義を削除してから新しい定義に置き換える、という動作になりますが、例外として、この関数に対して与えられていた特権は影響を受けません。現行サーバーにこの関数の定義が存在しなければ、このオプションは無視されます。既存の関数を置き換えるには、新しい定義の *specific-name* および *function-name* が古い定義の *specific-name* および *function-name* と同じであるか、新しい定義のシグニチャーが古い定義のシグニチャーと一致している必要があります。そうでなければ、新しい関数が作成されます。

function-name

ユーザー定義関数の名前を指定します。名前、スキーマ名、パラメーターの数、およびそれぞれのパラメーターのデータ・タイプ (データ・タイプの長さ、精度、位取り、または CCSID の属性に関係なく) の組み合わせで、現行サーバー上に存在しているユーザー定義の関数を識別してはなりません。

SQL 命名の場合、関数は、暗黙または明示修飾子で指定されたスキーマ内に作成されます。

システム命名の場合、関数は、修飾子で指定されたスキーマ内に作成されます。修飾子を指定しなかった場合:

- CURRENT SCHEMA 特殊レジスターの値が *LIBL である場合、関数は、現行ライブラリー (*CURLIB) 内に作成されます。
- そうでない場合、関数は現行スキーマ内に作成されます。

通常、それぞれの関数の関数シグニチャーが固有であれば、複数の関数に同じ名前を指定することができます。

一部の関数名は、システムが使用するために予約されています。 詳しくは、1070 ページの『CREATE FUNCTION』の『スキーマおよび関数名の選択』を参照してください。

(*parameter-declaration,...***)**

関数の入力パラメーターの数とそれぞれのパラメーターのデータ・タイプを指定します。各 *parameter-declaration* は、関数の入力パラメーターを指定します。

パラメーターの最大数は、言語のタイプによって異なります。

- JAVA と ILE プログラムおよびサービス・プログラムの場合、最大数は 2000 です。
- OPM プログラムの場合、最大数は 90 です。

パラメーターの最大数は、その言語で許されるパラメーターの最大数によってさらに制限される可能性があります。関数は、ゼロまたはそれ以上の入力パラメーターを持つことができます。関数が受け取ると予期しているパラメーターには、それぞれリスト内に 1 個の項目が必要です。関数のパラメーターはすべて入力パラメーターで、ヌル可能です。JAVA の場合は、DECIMAL タイプと NUMERIC タイプ以外の数値パラメーターがヌル可能です。CALLED ON NULL INPUT 関数でそのようなパラメーターの入力がヌル値になると、実行時エラーになります。詳しくは、 1070 ページの『CREATE FUNCTION』の『パラメーターの定義』を参照してください。

parameter-name
パラメーター名を指定します。必須ではありませんが、各パラメーターにパラメーター名を指定することができます。この名前は、パラメーター・リスト内の他のパラメーター名 と同じものであってはなりません。

data-type1
入力パラメーターのデータ・タイプを指定します。このデータ・タイプは、組み込みデータ・タイプまたは特殊タイプにすることができます。この変数は、配列タイプにすることはできません。

built-in-type
組み込みデータ・タイプを指定します。それぞれの組み込みデータ・タイプの詳細については、 1238 ページの『CREATE TABLE』を参照してください。データ・タイプによってはすべての言語ではサポートされていないものもあります。 SQL データ・タイプとホスト言語データ・タイプのマッピングについての詳細は、「組み込み SQL プログラミング」トピック集を参照してください。組み込みデータ・タイプの仕様は、ユーザー定義関数の作成に使用する言語に対応していれば、指定することができます。

distinct-type-name
ユーザー定義特殊タイプを指定します。パラメーターの長さ、精度、または位取り属性は、特殊タイプのソース・タイプの属性 (CREATE TYPE で指定された属性) と同じになります。特殊タイプの作成についての詳細は、 1328 ページの『CREATE TYPE (特殊)』を参照してください。

特殊タイプの名前が修飾されていない場合、データベース・マネージャーは、SQL パス上のスキーマを検索することでそのスキーマ名を解決します。

CCSID が指定されている場合、関数に渡される前に、パラメーターはその CCSID に変換されます。CCSID が指定されていない場合は、CCSID は、関数の呼び出し時点における現行サーバーのデフォルトの CCSID によって決まります。

日付、時刻、およびタイム・スタンプの各パラメーターは、ISO 形式の文字ストリングとしてプロシージャーに渡されます。

XML タイプのパラメーターでは、*XML-cast-type* 文節または AS LOCATOR 文節を指定する必要があります。

AS LOCATOR

実際の値の代わりにパラメーターの値へのロケーターを関数に渡すことを指定します。 AS LOCATOR は、LOB または XML データ・タイプまたは LOB または XML データ・タイプに基づく特殊タイプのパラメーターの場合に限り使用するようにしてください。AS LOCATOR を指定した場合、FOR SBCS DATA または FOR MIXED DATA を指定してはなりません。

AS LOCATOR 文節について詳しくは、 1070 ページの『CREATE FUNCTION』の『パラメーターに AS LOCATOR を指定する』を参照してください。

AS *XML-cast-type*

XML タイプまたは XML タイプに基づく特殊タイプのパラメーターのデータ・タイプとしてこの関数に渡すデータ・タイプを指定します。LOCATOR を指定する場合は、パラメーターが実際の値ではなく値のロケーターになります。

CCSID 値を指定する場合に、グラフィック・データ・タイプとして指定できるのは、Unicode CCSID 値に限られます。CCSID 値を指定しない場合は、この関数の作成時に、SQL_XML_DATA_CCSID QAQQINI オプションの設定に基づいて CCSID が設定されます。デフォルトの CCSID は 1208 です。このオプションの説明については、 101 ページの『XML 値』を参照してください。

default-clause

パラメーターのデフォルト値を指定します。デフォルト値は、定数、特殊レジスター、グローバル変数、式、またはキーワード NULL にすることができます。式は、集約関数および列名を含まない、 196 ページの『式』で定義されている任意の式です。デフォルト値が指定されていない場合、パラメーターにデフォルト値がないため、呼び出し時に省略できません。式ストリングの最大長は 64K です。

デフォルトの式は、パラメーターのデータ・タイプに対して割り当ての互換性がなければなりません。

デフォルト式内でリスト中の数値定数を区切る区切り記号として使用するコンマの後には、スペースが 1 つ必要です。

デフォルト式の中で参照されるすべてのオブジェクトは、関数が作成されるときに存在している必要があります。関数が起動されると、デフォルトは起動側の権限を使用して評価されます。

配列タイプのパラメーターにデフォルトを指定することはできません。

RETURNS

関数の結果のデータ・タイプを指定します。この文節は、オプションの CAST FROM 文節との関連で考慮してください。

data-type2

出力のデータ・タイプと属性を指定します。

あらゆる組み込みデータ・タイプ (ただし、LONG VARCHAR、LONG VARGRAPHIC、または DataLink は除く) や特殊タイプ (データ・リンクをベースとしていない) を指定することができます。配列タイプを指定することはできません。

CCSID が指定されている場合

- AS LOCATOR が指定されていない場合、戻される結果はその CCSID でコード化されていると想定されます。
- AS LOCATOR が指定され、ロケーターが指しているデータの CCSID が異なる CCSID でコード化されている場合、データは指定された CCSID に変換されます。

CCSID が指定されず、ビューの最外部の選択リストで関数が参照されない場合、以下のようになります。

- AS LOCATOR が指定されていない場合、戻される結果は、ジョブの CCSID (グラフィック・ストリング戻り値の場合は、ジョブに関連したグラフィック CCSID) でコード化されていると想定されます。
- AS LOCATOR が指定され、ロケーターが指しているデータの CCSID が異なる CCSID でコード化されている場合、ロケーターが指しているデータは、ジョブの CCSID に変換されます。変換時に文字が失われるのを防ぐために、関数から戻される文字をすべて表現できる CCSID を明示的に指定することを考慮してください。 これは、データ・タイプがグラフィック・ストリング・データの場合に特に重要です。 この場合、CCSID 1200 または 13488 (ユニコード・グラフィック・ストリング・データ) を使用することを考慮してください。

CCSID が指定されず、ビューの最外部の選択リストで関数が参照される場合、以下のようになります。

- AS LOCATOR が指定されていない場合、戻される結果は、関連するビュー列の CCSID でコード化されていると想定されます。
- AS LOCATOR が指定され、ロケーターが指しているデータの CCSID が異なる CCSID でコード化されている場合、ロケーターが指しているデータは、関連するビュー列の CCSID に変換されます。変換時に文字が失われるのを防ぐために、関数から戻される文字をすべて表現できる CCSID を明示的に指定することを考慮してください。 これは、データ・タイプがグラフィック・ストリング・データの場合に特に重要です。 この場合、CCSID 1200 または 13488 (ユニコード・グラフィック・ストリング・データ) を使用することを考慮してください。

AS LOCATOR

これを指定すると、関数は、実際の値ではなく、値のロケーターを戻します。 AS LOCATOR は、関数の結果が LOB または XML データ・タイプまたは LOB または XML データ・タイプに基づく特殊タイプのパラメーターを持っている場合に限り使用するようにしてください。AS LOCATOR を指定した場合、FOR SBCS DATA または FOR MIXED DATA を指定してはなりません。

AS LOCATOR 文節について詳しくは、 1070 ページの『CREATE FUNCTION』の『パラメーターに AS LOCATOR を指定する』を参照してください。

data-type3 **CAST FROM** *data-type4*

関数のデータ・タイプと属性 (データ・タイプ *4*)、および、その結果が呼び出しステートメントに戻されるデータ・タイプ (データ・タイプ *3*) を指定します。これら 2 つのデータ・タイプは、異なっていても構いません。

例えば、次の定義の場合、関数は DOUBLE の値を戻しますが、データベース・マネージャーはそれを DECIMAL の値に変換してから、呼び出し元ステートメントに渡します。

```
CREATE FUNCTION SQRT (DECIMAL15,0))
  RETURNS DECIMAL(15,0)
  CAST FROM DOUBLE
  ...
```

data-type4 の値は、XML タイプまたは特殊タイプであってはならず、*data-type3* にキャスト可能である必要があります。 データ・タイプ *3* の値は、任意の組み込みデータ・タイプや特殊タイプにすることができます。 (データ・タイプのキャストについては、 109 ページの『データ・タイプ間のキャスト』を参照してください。)

CCSID については、上記のデータ・タイプ *2* の説明を参照してください。

AS LOCATOR
これを指定すると、関数は、実際の値ではなく、値のロケーターを戻します。 AS LOCATOR は、関数の結果が LOB データ・タイプまたは LOB データ・タイプに基づく特殊タイプのパラメーターを持っている場合に限り使用するようにしてください。 AS LOCATOR を指定した場合、FOR SBCS DATA または FOR MIXED DATA を指定してはなりません。

AS LOCATOR 文節について詳しくは、 1070 ページの『CREATE FUNCTION』の『パラメーターに AS LOCATOR を指定する』を参照してください。

LANGUAGE
関数本体を作成する言語インターフェース規則を指定します。すべてのプログラムがサーバーの環境で実行するように設計しなければなりません。

LANGUAGE を指定しなかった場合、その LANGUAGE は、関数の作成時に、外部プログラムと関連したプログラム属性情報から決定されます。次の場合、プログラムの言語は C であると想定されます。

- プログラムに関連したプログラム属性情報で、認識可能な言語を識別しない。
- プログラムが見つからない。

C 外部プログラムは C で作成されます。

C++
外部プログラムは C++ で作成されます。

CL 外部プログラムは CL または ILE CL で作成されます。

COBOL
外部プログラムは COBOL で作成されます。

COBOLLE
外部プログラムは ILE COBOL で作成されます。

JAVA
外部プログラムは JAVA で作成されます。このユーザー定義の関数はデー

タベース・マネージャーにより呼び出されます。この関数は、指定された Java クラスの共通静的メソッドでなければなりません。

LANGUAGE JAVA を指定する場合は、EXTERNAL NAME 文節で有効な *external-java-routine-name* の値を指定する必要があります。
SCRATCHPAD、FINAL CALL、または DBINFO を指定する場合は、LANGUAGE JAVA を指定しないでください。

PLI
外部プログラムは PL/I で作成されます。

RPG
外部プログラムは RPG で作成されます。

RPGLE
外部プログラムは ILE RPG で作成されます。

PARAMETER STYLE
関数にパラメーターを渡し、関数から値を戻すために使用する規則を指定します。

SQL
適用可能なパラメーターはすべて渡されます。これらのパラメーターは、次の順序で配列されるように定義されます。

- 関数に対して指定される入力パラメーター用の *n* 個のパラメーター。
- 関数の結果を表すパラメーター。
- 入力パラメーターに対する標識変数用の *n* 個のパラメーター。
- 結果の標識変数を表すパラメーター。
- SQLSTATE の CHAR(5) 出力パラメーター。戻される SQLSTATE は、関数が成功したかどうかを示します。戻される SQLSTATE は、以下のいずれかです。
  - 外部プログラムで実行された最後の SQL ステートメントからの SQLSTATE
  - 外部プログラムによって割り当てられた SQLSTATE

  ユーザーは、関数からエラーまたは警告を戻すために、外部プログラム内で SQLSTATE を任意の有効な値にセットすることができます。
- 完全修飾関数名の VARCHAR(517) 入力パラメーター。
- 特定の名前の VARCHAR(128) 入力パラメーター。
- メッセージ・テキストの VARCHAR(1000) 出力パラメーター。

  制御が呼び出し側プログラムに戻された場合、SQLCA の SQLERRMC フィールドの 6 番目のトークンにメッセージ・テキストが入っています。入手できるのは、メッセージ・テキストの一部分だけです。SQLERRMC 内のメッセージ・データのレイアウトについては、メッセージ・ファイル QSQLMSG 内のメッセージ SQL0443 の置換データ記述を参照してください。完全なメッセージ・テキストは、GET DIAGNOSTICS ステートメントを使用して検索することができます。詳しくは、 1489 ページの『GET DIAGNOSTICS』を参照してください。
- 0 から 3 個のオプション・パラメーター

- CREATE FUNCTION ステートメントで SCRATCHPAD を指定した場合、スクラッチパッドの構造 (後に CHAR(n) が続く INTEGER からなる) 入出力パラメーター。
- CREATE FUNCTION ステートメントで FINAL CALL を指定した場合、呼び出しタイプの INTEGER 入力パラメーター。
- CREATE FUNCTION ステートメントで DBINFO を指定した場合、dbinfo 構造体の構造。

これらのパラメーターは、指定の LANGUAGE に基づいて渡されます。例えば、言語が C または C++ であれば、VARCHAR パラメーターはヌル終了ストリングとして渡されます。 渡されるパラメーターについての詳細は、ライブラリー QSYSINC の該当するソース・ファイル内の組み込み sqludf を参照してください。例えば、C の場合、sqludf は QSYSINC/H で見つかります。

DB2GENERAL

このパラメーター・スタイルは、Java クラスでメソッドとして定義されている外部関数にパラメーターを渡し、外部関数から値を戻すための規則を指定するのに使用します。適用可能なパラメーターはすべて渡されます。これらのパラメーターは、次の順序で配列されるように定義されます。

- 最初の N 個のパラメーターは、CREATE FUNCTION ステートメント上に指定される入力パラメーターです。
- 関数の結果を表すパラメーター。

DB2GENERAL は、LANGUAGE が JAVA の場合にのみ許されます。

GENERAL

適用可能なパラメーターはすべて渡されます。これらのパラメーターは、次の順序で配列されるように定義されます。

- 最初の N 個のパラメーターは、CREATE FUNCTION ステートメント上に指定される入力パラメーターです。

結果は、関数を戻す C 値の値として戻されることに注意してください。例えば、次のようになります。

```
return_val func(parameter-1, parameter-2, ...)
```

GENERAL は、EXTERNAL NAME でサービス・プログラムを識別する場合にのみ許可されます。

GENERAL WITH NULLS

適用可能なパラメーターはすべて渡されます。これらのパラメーターは、次の順序で配列されるように定義されます。

- 最初の N 個のパラメーターは、CREATE FUNCTION ステートメント上に指定される入力パラメーターです。
- 標識変数配列に追加の引数が渡されます。
- 結果の標識変数を表すパラメーター。

結果は、関数を戻す C 値の値として戻されることに注意してください。例えば、次のようになります。

```
return_val func(parameter-1, parameter-2, ...)
```

GENERAL WITH NULLS は、EXTERNAL NAME でサービス・プログラムを識別する場合にのみ許可されます。

JAVA
この関数で、Java 言語および ISO/IEC FCD 9075-13:2003「*Information technology - Database languages - SQL - Part 13: Java Routines and Types (SQL/JRT)*」仕様に準拠するパラメーター引き渡し規則を使用することを指定します。適用可能なパラメーターはすべて渡されます。これらのパラメーターは、次の順序で配列されるように定義されます。

- 最初の N 個のパラメーターは、CREATE FUNCTION ステートメント上に指定される入力パラメーターです。

結果は、関数を戻す C 値の値として戻されることに注意してください。例えば、次のようになります。

```
return_val func(parameter-1, parameter-2, ...)
```

JAVA は、LANGUAGE が JAVA の場合にのみ許されます。

パラメーターを渡す方法は、外部関数の言語によって決まります。例えば、C では、VARCHAR または CHAR パラメーターは NULL 文字で終了するストリングとして渡されます。詳しくは、「SQL プログラミング」のトピック集を参照してください。 Java ルーチンについては、「IBM Developer Kit for Java」のトピック集を参照してください。

SPECIFIC *specific-name*
関数の固有名を指定します。特定名の詳細については、 1070 ページの『CREATE FUNCTION』の『関数に特定の名前を指定する』を参照してください。

GLOBAL DETERMINISTIC または **STATEMENT DETERMINISTIC** または **NOT DETERMINISTIC**
関数が同じ入力引数を指定して呼び出されるたびに、その関数が同じ結果を戻すかどうかを指定します。デフォルトは NOT DETERMINISTIC です。

NOT DETERMINISTIC
関数が同じ入力引数を指定して呼び出されるたびに、その関数が同じ結果を戻さない場合があることを指定します。関数は、結果に影響を与えるいくつかの状態値によって変わってきます。データベース・マネージャーは、SQL ステートメントの最適化時にこの情報を使用します。非 deterministic 関数の例として、乱数を生成する関数があります。

非決定的関数を並列タスクで実行すると、間違った結果が返される可能性があります。このような関数には DISALLOW PARALLEL 文節を指定してください。

NOT DETERMINISTIC は、特殊レジスター、非決定的関数、またはシーケンスに対する参照がこの関数に含まれている場合に指定してください。

GLOBAL DETERMINISTIC
関数が同じ入力引数を指定して呼び出されるたびに、その関数が常に同じ結果を戻すかを指定します。データベース・マネージャーは、SQL ステートメントの最適化時にこの情報を使用します。照会オプティマイザーは、グローバル deterministic なスカラー関数の結果を キャッシュに入れるよう選

択する場合があります。[90]グローバル deterministic 関数の例としては、入力引数の平方根を計算する関数があります。

STATEMENT DETERMINISTIC
関数が同じ入力引数を使用して呼び出されるたびに、同じ結果を戻さない可能性があるが、単一 SQL ステートメント内での関数の複数の呼び出しは、deterministic と見なされることを指定します。照会オプティマイザーは、ステートメント deterministic なスカラー関数の結果をキャッシュに入れません。[91]ステートメント deterministic 関数の例としては、通貨変換を実行する関数があります。

CONTAINS SQL、READS SQL DATA、MODIFIES SQL DATA、または NO SQL
関数が実行できる SQL ステートメントおよびネストされたルーチンの種別を指定します。データベース・マネージャーは、関数と、関数からローカルで呼び出すすべてのルーチンが発行する SQL ステートメントが、この指定と整合しているかどうかを検査します。ネストされたリモート・ルーチンが呼び出された場合、検査は実行されません。各ステートメントの分類については、 1855 ページの『付録 B. SQL ステートメントの特性』を参照してください。デフォルトは、READS SQL DATA です。このオプションは、パラメーター・デフォルト式では無視されます。

READS SQL DATA
この関数が、データ・アクセス種別 READS SQL DATA、CONTAINS SQL、または NO SQL のステートメントを実行できるように指定します。この関数は、データの変更を行う SQL ステートメントは実行できません。

NO SQL
この関数が、データ・アクセス種別 NO SQL の SQL ステートメントのみを実行できるように指定します。

CONTAINS SQL
この関数が、データ・アクセス種別 CONTAINS SQL または NO SQL の SQL ステートメントのみを実行できるように指定します。この関数は、データの読み取りまたは変更を行う SQL ステートメントを実行できません。

MODIFIES SQL DATA
この関数は、どの関数でもサポートされないステートメントを除くすべての SQL ステートメントを実行できます。

RETURNS NULL ON NULL INPUT または CALLED ON NULL INPUT
入力引数のいずれかが実行時にヌルである場合に関数を呼び出すかどうかを指定します。CALLED ON NULL INPUT がデフォルトです。

RETURNS NULL ON INPUT
入力引数のいずれかが NULL である場合に関数を呼び出さないことを指定します。結果は NULL 値です。

---

90. 関数の結果に機密データが含まれる場合、結果への不用意なアクセスを防止するため、STATEMENT DETERMINISTIC または DETERMINISTIC_UDF_SCOPE QAQQINI オプションの使用を検討するか、関数 NOT DETERMINISTIC を作成してください。詳細については、「データベース パフォーマンスおよび Query 最適化」トピック集を参照してください。

91. DETERMINISTIC_UDF_SCOPE QAQQINI オプションを使用して、GLOBAL DETERMINISTIC 関数にこの同じ動作を使用することができます。詳細については、「データベース パフォーマンスおよび Query 最適化」トピック集を参照してください。

CALLED ON NULL INPUT
引数値のいずれかまたは全部がヌルである場合に関数を呼び出すことを指定します。この指定は、ヌル引数値のテストを行うように関数をコーディングする必要があります。関数は NULL または非 NULL 値を戻すことができます。

INHERIT SPECIAL REGISTERS
特殊レジスターの既存の値は、関数に入った後に継承されることを指定します。

STATIC DISPATCH
関数を静的にディスパッチすることを指定します。すべての関数が静的にディスパッチされます。

NO DBINFO または DBINFO
関数を呼び出すときに、追加の状況情報を渡すかどうかを指定します。デフォルトは NO DBINFO です。

NO DBINFO
補足情報を渡さないことを指定します。

DBINFO
データベース・マネージャーは、状況情報が入っている構造体を関数に渡す必要があることを指定します。 表 80 は、DBINFO 構造体の説明を示しています。 DBINFO 構造体についての詳しい情報は、ライブラリー QSYSINC 内の該当するソース・ファイルの組み込み sqludf に入っています。例えば、C の場合、sqludf は QSYSINC/H で見つかります。

DBINFO は、PARAMETER STYLE SQL または PARAMETER STYLE DB2GENERAL でのみ許可されます。

表 *80. DBINFO* フィールド

フィールド	データ・タイプ	説明
リレーショナル・データベース	VARCHAR(128)	現行サーバーの名前
権限 ID	VARCHAR(128)	実行時権限 ID

表 80. DBINFO フィールド (続き)

フィールド	データ・タイプ	説明
CCSID 情報	INTEGER INTEGER INTEGER INTEGER INTEGER INTEGER INTEGER INTEGER INTEGER INTEGER CHAR(8)	ジョブの CCSID 情報。3 つの CCSID が 3 セット戻されます。各セット内の 3 つの CCSID を識別する情報は、次のとおりです。 • SBCS CCSID • DBCS CCSID • 混合 CCSID 3 セットの CCSID の後に、3 セットの CCSID のうちのどのセットが該当するかを示す整数と 8 バイトの予約済みスペースが続きます。 CCSID の各設定は、異なるエンコード・スキーム (EBCDIC、ASCII、およびユニコード) のためのものです。 CREATE FUNCTION ステートメントのパラメーターの 1 つとして明示的に CCSID を指定していない場合は、入力ストリングは、関数の実行時にジョブの CCSID でコード化されるものと見なされます。入力ストリングの CCSID がパラメーターの CCSID と同じではない場合は、この外部関数に渡される入力ストリングは、外部プログラムの呼び出しの前に変換されます。
ターゲット列	VARCHAR(128) VARCHAR(128) VARCHAR(128)	ユーザー定義の関数が UPDATE ステートメントの SET 文節の右側に指定されている場合、次の情報がターゲット列を識別します。 • スキーマ名 • 基本表名 • 列名 ユーザー定義の関数が UPDATE ステートメントの SET 文節の右側に指定されなかった場合、これらのフィールドはブランクになります。
バージョンとリリース	CHAR(8)	データベース・マネージャーのバージョン、リリース、および修正レベル。
プラットフォーム	INTEGER	サーバーのプラットフォーム・タイプ。

EXTERNAL ACTION または NO EXTERNAL ACTION

関数が、データベース・マネージャーが管理しないオブジェクトの状態を変更するアクションを行うかどうかを指定します。外部アクションの例には、メッセージの送信やストリーム・ファイルへのレコードの書き込みなどがあります。デフォルトは EXTERNAL ACTION です。

EXTERNAL ACTION

関数が、データベース・マネージャーが管理しないオブジェクトの状態を変更するアクションを行うことができるかを指定します。したがって、関数は、それぞれの連続関数呼び出しで呼び出す必要があります。 EXTERNAL ACTION は、この関数に、外部アクションを持つ他の関数に対する参照が含まれている場合に、指定してください。

NO EXTERNAL ACTION
関数は外部アクションを行いません。この関数は、連続した各関数呼び出しごとに呼び出す必要はありません。

NO EXTERNAL ACTION 関数は、連続した各関数呼び出しごとに呼び出されない場合があるので、EXTERNAL ACTION 関数よりもパフォーマンスが向上する可能性があります。

FENCED または NOT FENCED
データベース・マネージャー環境から分離した環境で外部関数を実行するかどうかを指定します。 FENCED がデフォルトです。

FENCED
この関数は別のスレッドで実行されます。

FENCED 関数では、関数の呼び出し間で SQL カーソルをオープン状態のままにすることができません。ただし、あるスレッド内のカーソルは他のスレッド内のカーソルから独立しています。このことは、カーソル名の競合が起きる可能性を低くしています。

NOT FENCED
この関数は、呼び出し元の SQL ステートメントと同じスレッド内で実行できます。

NOT FENCED 関数では、関数の呼び出し間で SQL カーソルをオープン状態のままにすることができます。カーソルをオープン状態のままにしておくことができるため、関数の呼び出し間でカーソル位置も同じに維持されます。ただし、UDF が呼び出し元の SQL ステートメントおよび他の NOT FENCED UDF と同じスレッド内で実行されるためにカーソル名が競合する場合があります。

NOT FENCED 関数は、通常 FENCED 関数よりもパフォーマンスが良好です。

PROGRAM TYPE MAIN または PROGRAM TYPE SUB
他の製品との互換性を備えるために、このパラメーターが許可されています。これは、ルーチンの外部プログラムが、プログラム (*PGM) であるか、サービス・プログラム (*SRVPGM) のプロシージャーであるかを示します。

PROGRAM TYPE MAIN
ルーチンがプログラムのメインエントリー・ポイントとして実行することを指定します。外部プログラムは、*PGM オブジェクトでなければなりません。

PROGRAM TYPE SUB
ルーチンがサービス・プログラムのプロシージャーとして実行することを指定します。外部プログラムは、*SRVPGM オブジェクトでなければなりません。

NO FINAL CALL または FINAL CALL
関数の最終呼び出し を行うかどうかを指定します。最終呼び出しによって、関数は、獲得済みのすべてのシステム・リソースを解放することができます。最終呼び出しは、SCRATCHPAD キーワードを定義した関数でシステム・リソースを獲得し、そのシステム・リソースをスクラッチパッドに格納するときに使用できます。デフォルトは NO FINAL CALL です。

NO FINAL CALL
関数に対する最終呼び出しを行わないことを指定します。関数は、呼び出しのタイプを指定する追加の引数を受け取りません。

FINAL CALL
関数に対する最終呼び出しを行うことを指定します。この関数は、最終呼び出しとその他の呼び出しを区別するために、呼び出しのタイプを示す追加の引数を受け取ります。

FINAL CALL は、PARAMETER STYLE SQL または PARAMETER STYLE DB2GENERAL でのみ許可されます。

呼び出しには以下のタイプがあります。

First Call (初回呼び出し**)**
この SQL ステートメントでの関数に対するこの参照についての、関数に対する初回呼び出しを示します。初回呼び出しは通常呼び出しです。 SQL 引数が渡され、関数は結果を戻すものと見なされます。

Normal Call (通常呼び出し**)**
SQL 引数が渡され、関数が結果を戻すことを指定します。

Final Call (最終呼び出し**)**
この関数に対する最後の呼び出しであり、これにより関数はリソースを解放できることを指定します。最終呼び出しは通常呼び出しではありません。エラーが起きた場合は、データベース・マネージャーは最終呼び出しを行おうとします。

最終呼び出しが発生するのは以下の場合です。

- ステートメントの終わり : カーソル指向ステートメント用のカーソルがクローズされたとき、またはステートメントの実行が完了したとき。
- 並列タスクの終わり : 並列タスクにより関数が実行されたとき。
- トランザクションの終わり : ステートメント処理の正常終了が発生しなかったとき。例えば、何らかの理由によりアプリケーションのロジックがカーソルのクローズをバイパスした場合。

最終呼び出しを使用する関数には、並列タスクが関数を実行した場合に誤った結果を受け取るものもあります。例えば、関数が最終呼び出しごとに注釈を送信する場合、各関数ごとにではなく、各並列タスクごとに 1 つの注釈が送信されます。並列して実行した場合に正しく動作しない関数には、DISALLOW PARALLEL 文節を指定してください。

WITH HOLD として定義されているカーソルがオープン状態にあるときにコミット操作が発生した場合は、カーソルがクローズされるかアプリケーションが終了した時点で、最終呼び出しが行われます。並列タスクの終わりにコミットが発生した場合は、WITH HOLD として定義されているカーソルがオープン状態にあってもなくても、最終呼び出しが行われます。

FINAL CALL ではコミット可能な操作は行ってはなりません。FINAL CALL は、COMMIT 操作の一部として呼び出されたクローズ中に実行される可能性があるからです。

ALLOW PARALLEL または **DISALLOW PARALLEL**
関数を並列で実行できるかどうかを指定します。

NOT DETERMINISTIC、EXTERNAL ACTION、FINAL CALL、MODIFIES SQL DATA、SCRATCHPAD のいずれか 1 つ以上の文節を指定すると、デフォルトは DISALLOW PARALLEL になります。それ以外の場合は、ALLOW PARALLEL がデフォルトです。

ALLOW PARALLEL
データベース・マネージャーが関数の並列処理を考慮できることを指定します。データベース・マネージャーは、この関数を呼び出す SQL ステートメントまたはこの関数の中で実行する SQL ステートメントで並列処理を使用しなければならない、というわけではありません。

ALLOW PARALLEL の指定に関する注意点については、NOT DETERMINISTIC、EXTERNAL ACTION、MODIFIES SQL DATA、SCRATCHPAD、FINAL CALL の説明を参照してください。

DISALLOW PARALLEL
データベース・マネージャーが関数の並列処理を使用してはならないことを指定します。

NO SCRATCHPAD または **SCRATCHPAD**
関数に、静的メモリー域が必要か否かを指定します。

NO SCRATCHPAD
これを指定すると、関数では、持続メモリー域を必要としなくなります。

SCRATCHPAD *integer*
これを指定すると、関数には、持続メモリー域の長さ整数が必要となります。この整数に指定できる範囲は、1 から 16,000,000 です。メモリー域を指定しなかった場合、その区域のサイズは 100 バイトになります。パラメーター・スタイル SQL を指定すると、ポインターは、静的ストレージを指す必須パラメーターに続いて渡されます。ALLOW PARALLEL を指定すると、メモリー域は、ステートメント内のそれぞれのユーザー定義の関数参照に割り振られます。DISALLOW PARALLEL を指定すると、1 つのメモリー域だけが関数に割り振られます。

スクラッチパッドの有効範囲は SQL です。SQL ステートメント内の関数の参照ごとに、1 つのスクラッチパッドが存在します。例えば、関数 UDFX が SCRATCHPAD キーワードによって定義されていると想定した場合、次の SQL ステートメント内では、UDFX の 3 つの参照に対して 3 つのスクラッチパッドが割り振られます。

```
SELECT A, UDFX(A)
  FROM TABLEB
  WHERE UDFX(A) > 103 OR UDFX(A) < 19
```

関数が並列タスクのもとで実行されている場合、SQL ステートメント内の関数のそれぞれの参照の並列タスクごとに、1 つのスクラッチパッドが割り振られます。これによって、予測不能の結果が生じる可能性があります。例えば、関数で、呼び出される回数をカウントするためにスクラッチパッドを

使用した場合、そのカウントは、SQL ステートメントではなく、並列タスクによって行われた呼び出し回数を反映します。並列性を使用して正しく動作しない関数には、DISALLOW PARALLEL 文節を指定してください。

SCRATCHPAD は、PARAMETER STYLE SQL または PARAMETER STYLE DB2GENERAL でのみ許可されます。

EXTERNAL

CREATE FUNCTION ステートメントを使用して、外部プログラミング言語によるコードに基づいた新規関数を定義するように指定します。

外部プログラム名 の指定がない場合、外部プログラム名は該当の関数名と同じであると見なされます。

NAME *external-program-name*

SQL ステートメント内でこの関数が呼び出されたときに実行するプログラム、サービス・プログラム、または Java クラスを指定します。この名前は、関数が呼び出される時点でアプリケーション・サーバー上に存在しているプログラム、サービス・プログラム、または Java クラスを示すものでなければなりません。命名オプションが *SYS であり、その名前が修飾されていない場合:

- 関数の呼び出し時に、現行パスを使用して該当のプログラムを検索します。
- COMMENT、GRANT、LABEL、REVOKE の各操作をその関数で実行する時点で、*LIBL を使用して対象のプログラムまたはサービス・プログラムを検索します。

この名前の妥当性は、アプリケーション・サーバーで検査されます。名前の形式が正しくない場合、エラーが戻されます。

このプログラム、サービス・プログラム、または Java クラスは、関数の作成時に存在している必要はありませんが、関数の呼び出し時には存在している必要があります。

CONNECT、SET
CONNECTION、RELEASE、DISCONNECT、COMMIT、ROLLBACK および SET TRANSACTION ステートメントは、関数の外部プログラム内で使用することはできません。

NOT SECURED または **SECURED**

関数が行アクセス制御と列アクセス制御においてセキュアであると見なされるかどうかを指定します。

NOT SECURED

関数が行アクセス制御と列アクセス制御において非セキュアであると見なされることを指定します。これはデフォルトです。

表でアクティブな列アクセス制御が使用されている場合、関数の呼び出し時に、関数の引数が、列マスクが有効になっている列を参照してはなりません。

SECURED

関数が行アクセス制御と列アクセス制御においてセキュアであると見なされることを指定します。このオプションは、C、C++、ILE RPG、ILE COBOL、ILE CL、または Java で作成された外部関数に対して使用で

きます。プログラムまたはサービス・プログラムは、CREATE FUNCTION ステートメントが実行されるときに存在していなければなりません。

関数は、行の許可または列マスク内で参照される場合は、セキュアとして定義される必要があります。

SET OPTION-statement
パラメーター・デフォルトに使用されるオプションを指定します。各オプションのデフォルト値は、作成時に有効だったオプションによって異なります。詳しくは、 1696 ページの『SET OPTION』を参照してください。

デフォルト値式を処理するときには、オプション ALWCPYDTA、CONACC、DATFMT、DATSEP、DECFLTRND、DECMPT、DECRESULT、DFTRDBCOL、
LANGID、SQLCURRULE、SQLPATH、SRTSEQ、TGTRLS、TIMFMT、および TIMSEP が使用されます。オプション
CNULRQD、CNULIGN、COMPILEOPT、EXTIND、NAMING、SQLCA は、CREATE FUNCTIONステートメントでは使用できません。他のオプションは、受け入れられますが、無視されます。

注

ユーザー定義関数の定義に関する一般考慮事項**:** ユーザー定義関数の定義に関する一般情報については、 1070 ページの『CREATE FUNCTION』を参照してください。

REPLACE の規則**:** REPLACE によって外部関数を再作成する場合は、以下のようになります。

- 既存のコメントまたはラベルは破棄されます。
- 別の外部プログラムを指定する場合、
  - 権限を持つユーザーは新しいプログラムにコピーされません。
  - ジャーナル監査は変更されません。
- 上記以外の場合、
  - 権限を持つユーザーは維持されます。オブジェクト所有者は変更されない可能性があります。
  - 現在のジャーナル監査は変更されません。

関数の作成**:** ILE 外部プログラムまたはサービス・プログラムに関連した外部関数が作成されると、その関数に関連したプログラムやサービス・プログラムのオブジェクトへの関数属性の保管が試行されます。 *PGM または *SRVPGM オブジェクトが保管された後、このシステムまたは別のシステムに復元されると、属性が使用されてカタログが更新されます。

外部関数の場合は、次の制約の範囲内で属性を保管することができます。

- 外部プログラム・ライブラリーは、QSYS であってはなりません。
- 外部プログラムは、CREATE FUNCTION ステートメントの発行時に存在していなければなりません。

システム命名が指定され、外部プログラム名が修飾されない場合は、外部プログラムはライブラリー・リストで検出されなければなりません。

- 外部プログラムは、ILE *PGM オブジェクトか *SRVPGM オブジェクトにする必要があります。
- 外部プログラムには、32 ルーチンの属性が既に入っていてはなりません。

オブジェクトを更新できない場合でも、関数は作成されます。

関数の呼び出し**:** 外部関数が呼び出されると、その関数は、外部プログラムやサービス・プログラムの作成時に指定された活動化グループであれば、どの活動化グループ内でも実行します。ただし、通常は、関数が呼び出し側プログラムと同じ活動化グループ内で実行するように ACTGRP(*CALLER) を使用します。ACTGRP(*NEW) は使用できません。

LANGUAGE JAVA の関数は、常にデフォルトの活動化グループ (*DFTACTGRP) で実行されます。したがって、MODIFIES SQL DATA Java 関数を作成するときには、注意が必要です。その Java 関数による変更は、デフォルトの活動化グループで実行されるので、呼び出し側が新しい活動化グループ (*NEW) で実行されていると、トランザクションの問題が発生する可能性があります。

Java 関数に関する注釈**:** Java 関数を実行するためには、システムに IBM IBM Developer Kit for Java (5770-JV1) をインストールしておく必要があります。インストールされていないと、SQLCODE -443 が戻され、CPDB521 メッセージがジョブ・ログに入ります。

Java 関数の実行中にエラーが発生すると、SQLCODE -443 が戻されます。エラーによっては、関数が実行されていたジョブのジョブ・ログに他のメッセージが入っている場合があります。

代替構文**:** 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード VARIANT と NOT VARIANT は、NOT DETERMINISTIC と DETERMINISTIC の同義語として使用することができます。
- キーワード NULL CALL と NOT NULL CALL は、CALLED ON NULL INPUT と RETURNS NULL ON NULL INPUT の同義語として使用できます。
- キーワード SIMPLE CALL は、GENERAL の同義語として使用できます。
- DB2GENERAL の同義語として、キーワード DB2GENRL を使用できます。
- SQL の同義語として、値 DB2SQL を使用できます。
- PARAMETER STYLE 文節のキーワード PARAMETER STYLE はオプションです。
- DETERMINISTIC の同義語として、キーワード IS DETERMINISTIC を使用できます。

例

例 *1:* C で書かれた外部関数プログラムが以下のロジックをインプリメントする必要があるとします:

```
rslt   = 2 * input - 4
```

入力引数の 1 つが NULL である場合にのみ、関数は NULL 値を戻す必要があります。関数呼び出しを回避し、入力値が NULL の場合に NULL の結果を得るための最も簡単な方法は、CREATE FUNCTION ステートメント上に RETURNS NULL ON NULL INPUT と指定することです。以下のステートメントは特定名 MINENULL1 を使用して、関数を定義します。

```
CREATE FUNCTION NTEST1 (SMALLINT)
    RETURNS SMALLINT
    EXTERNAL NAME NTESTMOD
    SPECIFIC MINENULL1
    LANGUAGE C
    DETERMINISTIC
    NO SQL
    FENCED
    PARAMETER STYLE SQL
    RETURNS NULL ON NULL INPUT
    NO EXTERNAL ACTION
```

プログラム・コードは次のとおりです。

```
void nudft1
  (int *input,                /* ptr to input argument         */
   int *output,               /* ptr to output argument        */
   short *input_ind,          /* ptr to input indicator        */
   short *output_ind,         /* ptr to output indicator       */
   char sqlstate[6],  /* sqlstate                      */
   char fname[140],   /* fully qualified function name */
   char finst[129],   /* function specific name        */
   char msgtext[71])  /* msg text buffer               */
{
  if (*input_ind == -1)
    *output_ind = -1;
  else
{
    *output = 2*(*input)-4;
    *output_ind = 0;
  }
  return;
}
```

例 *2:* ユーザーが CENTER という名前の外部関数を定義すると想定します。関数プログラムは、C で作成されます。以下のステートメントは関数を定義し、データベース・マネージャーが関数の特定名を生成できるようにします。関数本体を含むプログラムの名前は、関数の名前と同一であるため、EXTERNAL 文節には「NAME *external-program-name*」は含まれません。

```
CREATE FUNCTION CENTER (INTEGER, FLOAT)
   RETURNS FLOAT
   LANGUAGE C
   DETERMINISTIC
   NO SQL
   PARAMETER STYLE SQL
   NO EXTERNAL ACTION
```

例 *3:* ユーザー McBride (データベース管理者権限を持っている) が SMITH スキーマ内に CENTER という名前の外部関数を作成すると想定します。 McBride

は、この関数に特定名 FOCUS98 を指定しようとしています。関数プログラムでは、ある種の一回限りの初期設定を実行し、結果を保管するためにスクラッチパッドを使用します。関数プログラムでは、DOUBLE データ・タイプが指定された値を戻します。ユーザー McBride によって書き込まれた以下のステートメントは関数を定義し、関数の呼び出し時に、DECIMAL(8,4) というデータ・タイプが指定された値が戻されるようにします。

```
CREATE FUNCTION SMITH.CENTER (DOUBLE, DOUBLE, DOUBLE)
   RETURNS DECIMAL(8,4)
   CAST FROM DOUBLE
   EXTERNAL NAME CMOD
   SPECIFIC FOCUS98
   LANGUAGE C
   DETERMINISTIC
   NO SQL
   FENCED
   PARAMETER STYLE SQL
   NO EXTERNAL ACTION
   SCRATCHPAD
   NO FINAL CALL
```

例 *4:* 以下の例では、ストリング内の最初の母音の位置を戻す Java ユーザー定義関数を定義します。ユーザー定義関数は Java で書かれており、隔離して実行されるクラス JAVAUDFS の FINDVWL メソッドです。

```
CREATE FUNCTION FINDV (VARCHAR(32000))
   RETURNS INTEGER
   FENCED
   LANGUAGE JAVA
   PARAMETER STYLE JAVA
   EXTERNAL NAME 'JAVAUDFS.FINDVWL'
   NO EXTERNAL ACTION
   CALLED ON NULL INPUT
   DETERMINISTIC
   NO SQL
```

CREATE FUNCTION (外部表)

CREATE FUNCTION (外部表) ステートメントは、現行サーバー上に外部表関数を定義します。この外部表関数は、結果表を戻します。

外部ユーザー定義表関数 を副選択の FROM 文節の中で使用できます。この表関数は、呼び出しのたびに 1 行ずつを返すことによって、表を副選択に返します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、あるいは対話式に実行することもできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- SYSFUNCS カタログ・ビューと SYSPARMS カタログ表の場合
  - 該当の表に対する INSERT 特権、および
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

外部プログラムやサービス・プログラムが存在している場合、このステートメントの権限 ID が保持する特権には、少なくとも次のいずれか 1 つを含める必要があります。

- SQL ステートメントで参照された外部プログラムやサービス・プログラムの場合
  - その外部プログラムやサービス・プログラムが含まれるスキーマに対する USAGE 特権
  - その外部プログラムやサービス・プログラムに対するシステム権限の *EXECUTE。
  - そのプログラムやサービス・プログラムに対するシステム権限の *CHANGE。システムには、プログラム・オブジェクトを更新し、関数を別のシステムに保管/復元するために必要な情報を入れる場合にこの権限が必要となります。ユーザーにこの権限が与えられていない場合、関数は同じように作成されますが、プログラム・オブジェクトは更新されません。
- データベース管理者権限

SQL 名が指定され、関数が作成されるライブラリーと同じ名前のユーザー・プロファイルが存在し、しかも、その名前がステートメントの権限 ID と異なっている場合、ステートメントの権限 ID が保持している特権には、少なくとも次のいずれか 1 つを含める必要があります。

- その名前を持つユーザー・プロファイルに対する *ADD システム権限
- データベース管理者権限

特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別された、それぞれの特殊タイプごとに、

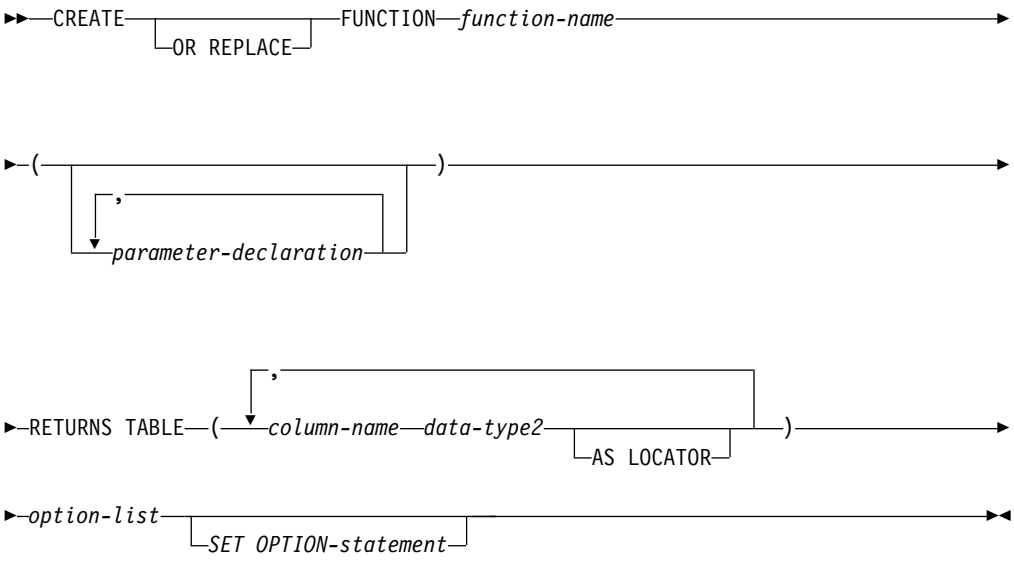
  - その特殊タイプに対する USAGE 特権、および
  - 特殊タイプが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

既存の関数に置き換えるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

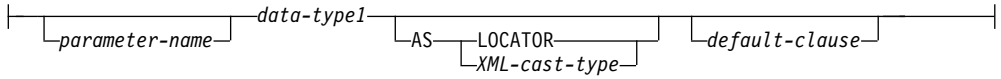
- 次のシステム権限
  - この関数に関連したサービス・プログラム・オブジェクトに対する *OBJMGT システム権限
  - この関数を削除するために必要な全権限
  - SYSFUNCS カタログ・ビューと SYSPARMS カタログ表に対する *READ システム権限
- データベース管理者権限

SQL 特権に対応するシステム権限については、『表またはビューへの権限を検査する際の対応するシステム権限』および 『特殊タイプへの権限を検査する際の対応するシステム権限』を参照してください。

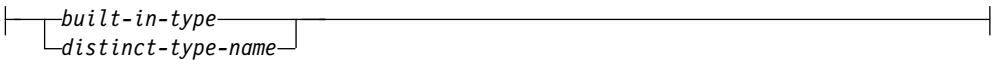
構文



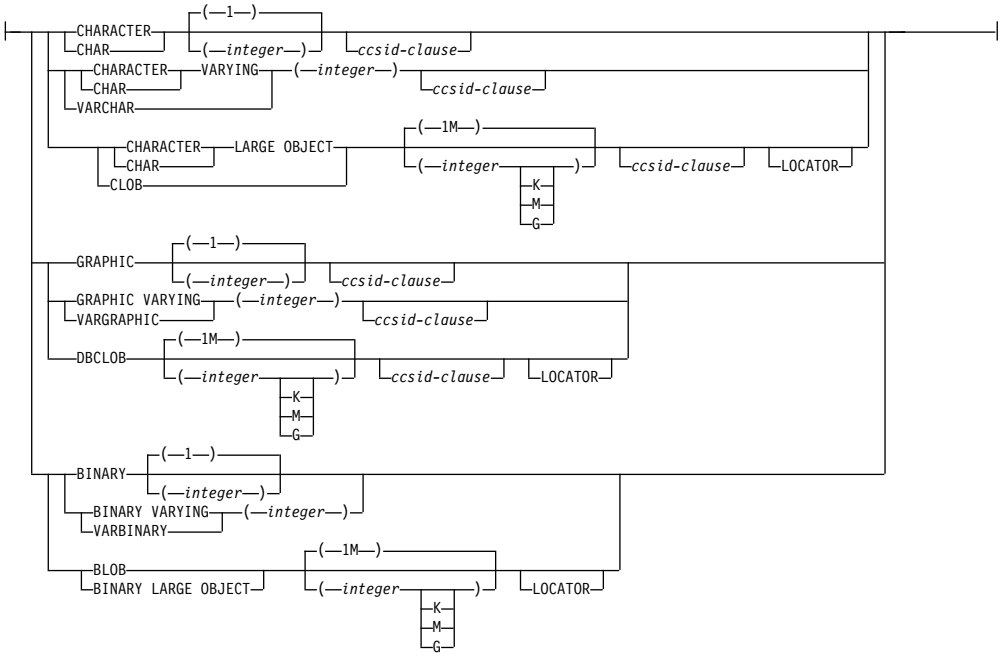
parameter-declaration:



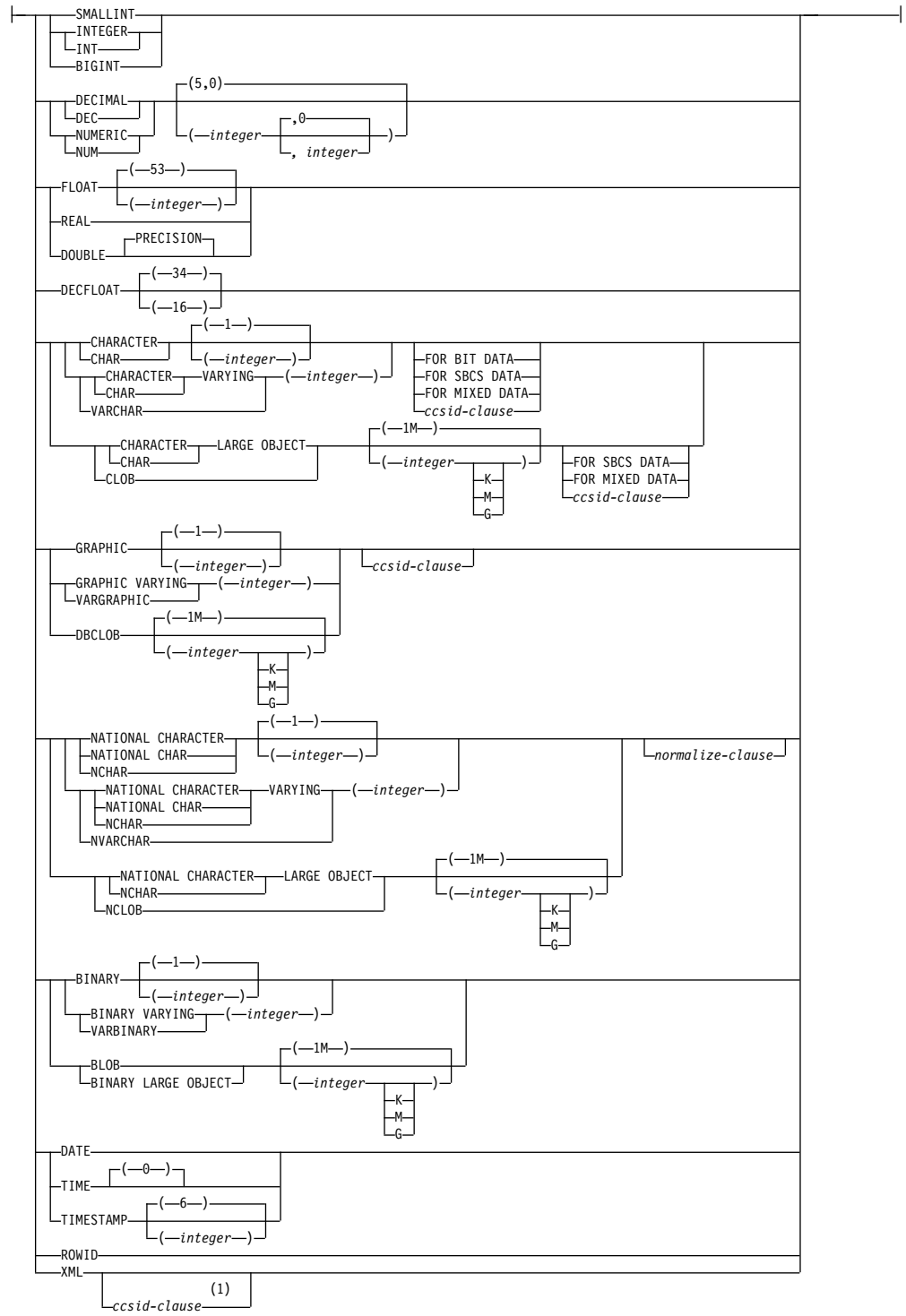
data-type1, data-type2:



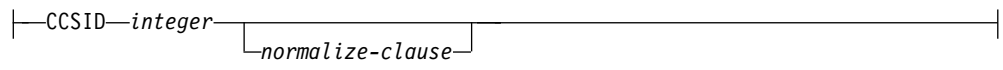
XML-cast-type:



built-in-type:



ccsid-clause:



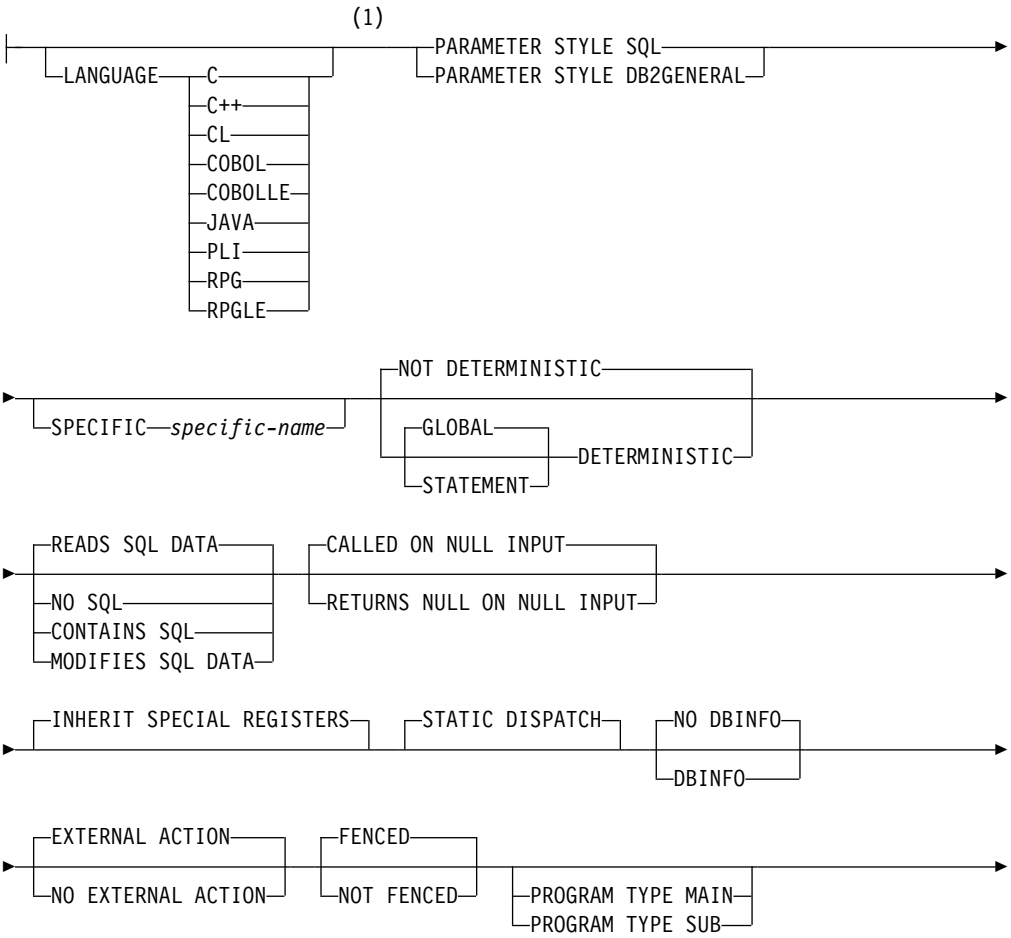
normalize-clause:

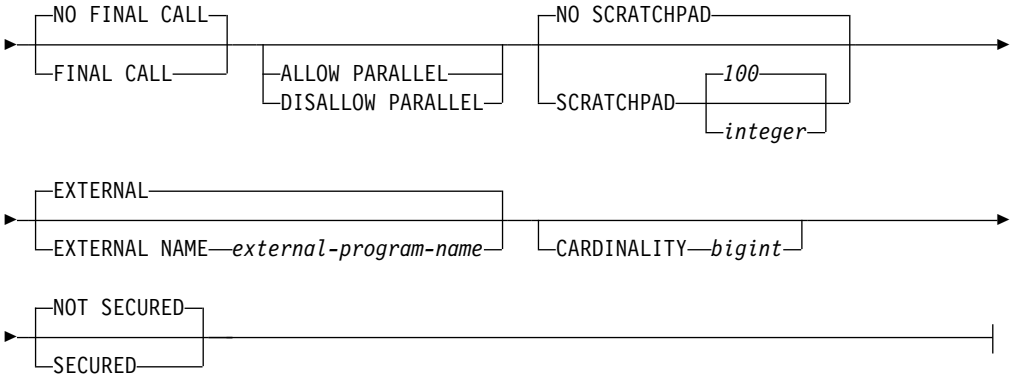
default-clause:

注:

1 XML の *ccsid-clause* は、*data-type2* でのみ使用できます。

option-list:





注:

1 この文節とこの後の option-list にある文節は、どのような順番で指定してもかまいません。各文節を指定できるのは、それぞれ 1 回だけです。

説明

OR REPLACE

現行サーバーにこの関数の定義が存在する場合に、その定義を置き換える、という動作を指定します。実際には、カタログで既存の定義を削除してから新しい定義に置き換える、という動作になりますが、例外として、この関数に対して与えられていた特権は影響を受けません。現行サーバーにこの関数の定義が存在しなければ、このオプションは無視されます。既存の関数を置き換えるには、新しい定義の *specific-name* および *function-name* が古い定義の *specific-name* および *function-name* と同じであるか、新しい定義のシグニチャーが古い定義のシグニチャーと一致している必要があります。そうでなければ、新しい関数が作成されます。

function-name

ユーザー定義関数の名前を指定します。名前、スキーマ名、パラメーターの数、およびそれぞれのパラメーターのデータ・タイプ (データ・タイプの長さ、精度、位取り、または CCSID の属性に関係なく) の組み合わせで、現行サーバー上に存在しているユーザー定義の関数を識別してはなりません。

SQL 命名の場合、関数は、暗黙または明示修飾子で指定されたスキーマ内に作成されます。

システム命名の場合、関数は、修飾子で指定されたスキーマ内に作成されます。修飾子を指定しなかった場合:

- CURRENT SCHEMA 特殊レジスターの値が *LIBL である場合、関数は、現行ライブラリー (*CURLIB) 内に作成されます。
- そうでない場合、関数は現行スキーマ内に作成されます。

通常、それぞれの関数の関数シグニチャーが固有であれば、複数の関数に同じ名前を指定することができます。

一部の関数名は、システムが使用するために予約されています。 詳しくは、1070 ページの『CREATE FUNCTION』の『スキーマおよび関数名の選択』を参照してください。

(*parameter-declaration,...***)**

関数の入力パラメーターの数とそれぞれのパラメーターのデータ・タイプを指定します。各 *parameter-declaration* は、関数の入力パラメーターを指定します。

パラメーターの最大数は、言語のタイプによって異なります。

- JAVA と ILE プログラムおよびサービス・プログラムの場合、最大数は 2000 です。
- OPM プログラムの場合、最大数は 90 です。

パラメーターの最大数は、その言語で許されるパラメーターの最大数によってさらに制限される可能性があります。関数は、ゼロまたはそれ以上の入力パラメーターを持つことができます。関数が受け取ると予期しているパラメーターには、それぞれリスト内に 1 個の項目が必要です。関数のパラメーターはすべて入力パラメーターで、ヌル可能です。JAVA の場合は、DECIMAL タイプと NUMERIC タイプ以外の数値パラメーターがヌル可能です。CALLED ON NULL INPUT 関数でそのようなパラメーターの入力がヌル値になると、実行時エラーになります。詳しくは、 1070 ページの『CREATE FUNCTION』の『パラメーターの定義』を参照してください。

parameter-name

パラメーター名を指定します。必須ではありませんが、各パラメーターにパラメーター名を指定することができます。この名前は、パラメーター・リスト内の他のパラメーター名 と同じものであってはなりません。

data-type1

入力パラメーターのデータ・タイプを指定します。このデータ・タイプは、組み込みデータ・タイプまたは特殊タイプにすることができます。この変数は、配列タイプにすることはできません。

built-in-type

組み込みデータ・タイプを指定します。それぞれの組み込みデータ・タイプの詳細については、 1238 ページの『CREATE TABLE』を参照してください。データ・タイプによってはすべての言語ではサポートされていないものもあります。 SQL データ・タイプとホスト言語データ・タイプのマッピングについての詳細は、「組み込み SQL プログラミング」トピック集を参照してください。組み込みデータ・タイプの仕様は、ユーザー定義関数の作成に使用する言語に対応していれば、指定することができます。

distinct-type-name

ユーザー定義特殊タイプを指定します。パラメーターの長さ、精度、または位取り属性は、特殊タイプのソース・タイプの属性 (CREATE TYPE で指定された属性) と同じになります。特殊タイプの作成について詳しくは、 1328 ページの『CREATE TYPE (特殊)』を参照してください。

特殊タイプの名前が修飾されていない場合、データベース・マネージャーは、SQL パス上のスキーマを検索することでそのスキーマ名を解決します。

PARAMETER STYLE JAVA を指定する場合は、ラージ・オブジェクト (LOB) データ・タイプのパラメーターはサポートされません。

CCSID が指定されている場合、関数に渡される前に、パラメーターはその CCSID に変換されます。CCSID が指定されていない場合は、CCSID は、関数の呼び出し時点における現行サーバーのデフォルトの CCSID によって決まります。

日付、時刻、およびタイム・スタンプの各パラメーターは、ISO 形式の文字ストリングとしてプロシージャーに渡されます。

XML タイプのパラメーターでは、*XML-cast-type* 文節または AS LOCATOR 文節を指定する必要があります。

AS LOCATOR
これを指定すると、入力パラメーターは、実際の値ではなく、値のロケーターになります。AS LOCATOR は、入力パラメーターに LOB または XML データ・タイプや LOB または XML データ・タイプをベースとする特殊タイプが指定されている場合にのみ、指定することができます。AS LOCATOR を指定した場合、FOR SBCS DATA または FOR MIXED DATA を指定してはなりません。

AS LOCATOR 文節について詳しくは、1070 ページの『CREATE FUNCTION』の『パラメーターに AS LOCATOR を指定する』を参照してください。

AS *XML-cast-type*
XML タイプまたは XML タイプに基づく特殊タイプのパラメーターのデータ・タイプとしてこの関数に渡すデータ・タイプを指定します。LOCATOR を指定する場合は、パラメーターが実際の値ではなく値のロケーターになります。

CCSID 値を指定する場合に、グラフィック・データ・タイプとして指定できるのは、Unicode CCSID 値に限られます。CCSID 値を指定しない場合は、この関数の作成時に、SQL_XML_DATA_CCSID QAQQINI オプションの設定に基づいて CCSID が設定されます。デフォルトの CCSID は 1208 です。このオプションの説明については、101 ページの『XML 値』を参照してください。

default-clause
パラメーターのデフォルト値を指定します。デフォルト値は、定数、特殊レジスター、グローバル変数、式、またはキーワード NULL にすることができます。式は、集約関数および列名を含まない、196 ページの『式』で定義されている任意の式です。デフォルト値が指定されていない場合、パラメーターにデフォルト値がないため、呼び出し時に省略できません。式ストリングの最大長は 64K です。

デフォルトの式は、パラメーターのデータ・タイプに対して割り当ての互換性がなければなりません。

デフォルト式内でリスト中の数値定数を区切る区切り記号として使用するコンマの後には、スペースが 1 つ必要です。

デフォルト式の中で参照されるすべてのオブジェクトは、関数が作成されるときに存在している必要があります。関数が起動されると、デフォルトは起動側の権限を使用して評価されます。

配列タイプのパラメーターにデフォルトを指定することはできません。

RETURNS TABLE

この関数の出力が表であることを指定します。この文節の後に、結果表の列の名前とデータ・タイプのリストを括弧で囲んで記述します。

結果列の最大数は、言語のタイプによって異なります。パラメーターの数が *N* であるとします。

- JAVA と ILE プログラムおよびサービス・プログラムの場合、列の数は 8000-*N* 以下でなければなりません。
- OPM プログラムの場合、列の数は 125 - *N* 以下でなければなりません。

column-name

出力表の列の名前を指定します。同じ名前を何度も指定することはできません。

data-type2

列のデータ・タイプを指定します。列はNULL 可能です。

あらゆる組み込みデータ・タイプ (ただし、LONG VARCHAR、LONG VARGRAPHIC、または DataLink は除く) や特殊タイプ (データ・リンクをベースとしていない) を指定することができます。配列タイプを指定することはできません。

DATE または TIME を指定した場合は、表関数は ISO 形式の日付または時刻を戻す必要があります。

CCSID が指定されている場合

- AS LOCATOR が指定されていない場合、戻される結果はその CCSID でコード化されていると想定されます。
- AS LOCATOR が指定され、ロケーターが指しているデータの CCSID が異なる CCSID でコード化されている場合、データは指定された CCSID に変換されます。

CCSID が指定されていない場合

- AS LOCATOR が指定されていない場合、戻される結果は、ジョブの CCSID (グラフィック・ストリング戻り値の場合は、ジョブに関連したグラフィック CCSID) でコード化されていると想定されます。
- AS LOCATOR が指定され、ロケーターが指しているデータの CCSID が異なる CCSID でコード化されている場合、ロケーターが指しているデータは、ジョブの CCSID に変換されます。変換時に文字が失われるのを防ぐために、関数から戻される文字をすべて表現できる CCSID を明示的に指定することを考慮してください。 これは、データ・タイプがグラフィック・ストリング・データの場合に特に重要です。 この場合、CCSID 1200 または 13488 (ユニコード・グラフィック・ストリング・データ) を使用することを考慮してください。

AS LOCATOR

これを指定すると、関数は、実際の値ではなく、該当の列の値に対するロケーターを戻します。 AS LOCATOR は、LOB または XML データ・タイプまたは LOB または XML データ・タイプに基づく特殊タイプに限り指定できます。AS LOCATOR を指定した場合、FOR SBCS DATA または FOR MIXED DATA を指定してはなりません。

AS LOCATOR 文節について詳しくは、 1070 ページの『CREATE FUNCTION』の『パラメーターに AS LOCATOR を指定する』を参照してください。

LANGUAGE

言語文節は、外部プログラムの言語を指定します。

LANGUAGE を指定しなかった場合、その LANGUAGE は、関数の作成時に、外部プログラムと関連したプログラム属性情報から決定されます。次の場合、プログラムの言語は C であると想定されます。

- プログラムに関連したプログラム属性情報で、認識可能な言語を識別しない。
- プログラムが見つからない。

C 外部プログラムは C で作成されます。

C++
外部プログラムは C++ で作成されます。

CL 外部プログラムは CL または ILE CL で作成されます。

COBOL
外部プログラムは COBOL で作成されます。

COBOLLE
外部プログラムは ILE COBOL で作成されます。

JAVA
外部プログラムは JAVA で作成されます。データベース・マネージャーは、ユーザー定義関数を、Java クラス内のメソッドとして呼び出します。

PLI
外部プログラムは PL/I で作成されます。

RPG
外部プログラムは RPG で作成されます。

RPGLE
外部プログラムは ILE RPG で作成されます。

PARAMETER STYLE

関数にパラメーターを渡し、関数から値を戻すために使用する規則を指定します。

DB2GENERAL
このパラメーター・スタイルは、Java クラスでメソッドとして定義されている外部関数にパラメーターを渡し、外部関数から値を戻すための規則を指定するのに使用します。適用可能なパラメーターはすべて渡されます。これらのパラメーターは、次の順序で配列されるように定義されます。

- 最初の N 個のパラメーターは、CREATE FUNCTION ステートメント上に指定される入力パラメーターです。
- その後の M 個のパラメーターは、RETURNS TABLE 文節に指定されている、この関数の結果列です。

DB2GENERAL は、LANGUAGE が JAVA の場合にのみ許されます。

SQL

適用可能なパラメーターはすべて渡されます。これらのパラメーターは、次の順序で配列されるように定義されます。

- 最初の N 個のパラメーターは、CREATE FUNCTION ステートメント上に指定される入力パラメーターです。
- その後の M 個のパラメーターは、RETURNS TABLE 文節に指定されている、この関数の結果列です。
- 入力パラメーターの標識変数を表す N 個のパラメーター。
- RETURNS TABLE 文節に指定されているこの関数の結果列の標識変数を表す M 個のパラメーター。
- SQLSTATE の CHAR(5) 出力パラメーター。戻される SQLSTATE は、関数が成功したかどうかを示します。戻される SQLSTATE は、以下のいずれかです。
  - 外部プログラムで実行された最後の SQL ステートメントからの SQLSTATE
  - 外部プログラムによって割り当てられた SQLSTATE

  ユーザーは、関数からエラーまたは警告を戻すために、外部プログラム内で SQLSTATE を任意の有効な値にセットすることができます。
- 完全修飾関数名の VARCHAR(517) 入力パラメーター。
- 特定の名前の VARCHAR(128) 入力パラメーター。
- メッセージ・テキストの VARCHAR(1000) 出力パラメーター。
- CREATE FUNCTION ステートメントで SCRATCHPAD を指定した場合、スクラッチパッドの構造 (後に CHAR(n) が続く INTEGER からなる) 入出力パラメーター。
- 呼び出しタイプを示す INTEGER 入力パラメーター。
- CREATE FUNCTION ステートメントで DBINFO を指定した場合、dbinfo 構造体の構造。

これらのパラメーターは、指定の LANGUAGE に基づいて渡されます。例えば、言語が C または C++ であれば、VARCHAR パラメーターはヌル終了ストリングとして渡されます。 渡されるパラメーターについての詳細は、ライブラリー QSYSINC の該当するソース・ファイル内の組み込み sqludf を参照してください。例えば、C の場合、sqludf は QSYSINC/H で見つかります。

パラメーターを渡す方法は、外部関数の言語によって決まります。例えば、C では、VARCHAR または CHAR パラメーターは NULL 文字で終了するストリングとして渡されます。詳しくは、「SQL プログラミング」のトピック集を参照してください。 Java ルーチンについては、「IBM Developer Kit for Java」のトピック集を参照してください。

SPECIFIC *specific-name*

関数の固有名を指定します。特定名の詳細については、 1070 ページの『CREATE FUNCTION』の『関数に特定の名前を指定する』を参照してください。

GLOBAL DETERMINISTIC または **STATEMENT DETERMINISTIC** または **NOT DETERMINISTIC**

関数が同じ入力引数を指定して呼び出されるたびに、その関数が同じ結果を戻すかどうかを指定します。デフォルトは NOT DETERMINISTIC です。

NOT DETERMINISTIC

関数が同じ入力引数を指定して呼び出されるたびに、その関数が同じ結果を戻さない場合があることを指定します。関数は、結果に影響を与えるいくつかの状態値によって変わってきます。データベース・マネージャーは、SQL ステートメントの最適化時にこの情報を使用します。表関数の結果表に影響するような方法で、特殊レジスター、非決定的な関数、シーケンスのいずれかを参照する表関数は、非決定的な表関数の例です。

GLOBAL DETERMINISTIC

関数が同じ入力引数を指定して呼び出されるたびに、その関数が常に同じ結果表を戻すかを指定します。データベース・マネージャーは、SQL ステートメントの最適化時にこの情報を使用します。照会オプティマイザーは、グローバル deterministic な関数の結果をキャッシュに入れるよう選択することができます。

STATEMENT DETERMINISTIC

関数が同じ入力引数を使用して呼び出されるたびに、同じ結果を戻さない可能性があるが、単一 SQL ステートメント内での関数の複数の呼び出しは、deterministic と見なされることを指定します。照会オプティマイザーは、ステートメント deterministic な関数の結果をキャッシュに入れません。[92]

CONTAINS SQL、**READS SQL DATA**、**MODIFIES SQL DATA**、または **NO SQL**

関数が実行できる SQL ステートメントおよびネストされたルーチンの種別を指定します。データベース・マネージャーは、関数と、関数からローカルで呼び出すすべてのルーチンが発行する SQL ステートメントが、この指定と整合しているかどうかを検査します。ネストされたリモート・ルーチンが呼び出された場合、検査は実行されません。各ステートメントの分類については、 1855 ページの『付録 B. SQL ステートメントの特性』を参照してください。デフォルトは、READS SQL DATA です。このオプションは、パラメーター・デフォルト式では無視されます。

READS SQL DATA

この関数が、データ・アクセス種別 READS SQL DATA、CONTAINS SQL、または NO SQL のステートメントを実行できるように指定します。この関数は、データの変更を行う SQL ステートメントは実行できません。

NO SQL

この関数が、データ・アクセス種別 NO SQL の SQL ステートメントのみを実行できるように指定します。

CONTAINS SQL

この関数が、データ・アクセス種別 CONTAINS SQL または NO SQL の SQL ステートメントのみを実行できるように指定します。この関数は、データの読み取りまたは変更を行う SQL ステートメントを実行できません。

92. DETERMINISTIC_UDF_SCOPE QAQQINI オプションを使用して、GLOBAL DETERMINISTIC 関数にこの同じ動作を使用することができます。詳細については、「データベース パフォーマンスおよび Query 最適化」トピック集を参照してください。

MODIFIES SQL DATA
この関数は、どの関数でもサポートされないステートメントを除くすべての SQL ステートメントを実行できます。

RETURNS NULL ON NULL INPUT または CALLED ON NULL INPUT
入力引数のいずれかが実行時にヌルである場合に関数を呼び出すかどうかを指定します。CALLED ON NULL INPUT がデフォルトです。

RETURNS NULL ON NULL INPUT
入力引数のいずれかがヌルである場合に関数を呼び出さないことを指定します。結果は、表に行がない、空表になります。

CALLED ON NULL INPUT
引数値のいずれかがヌルである場合に関数を呼び出すことを指定します。この指定は、ヌル引数値のテストを行うように関数をコーディングする必要があります。関数は、その論理によっては、空の表を戻す場合があります。

INHERIT SPECIAL REGISTERS
特殊レジスターの既存の値は、関数に入った後に継承されることを指定します。

STATIC DISPATCH
関数を静的にディスパッチすることを指定します。すべての関数が静的にディスパッチされます。

NO DBINFO または DBINFO
関数を呼び出すときに、追加の状況情報を渡すかどうかを指定します。デフォルトは NO DBINFO です。

NO DBINFO
補足情報を渡さないことを指定します。

DBINFO
データベース・マネージャーは、状況情報が入っている構造体を関数に渡す必要があることを指定します。 表 81 は、DBINFO 構造体の説明を示しています。 DBINFO 構造体についての詳しい情報は、ライブラリー QSYSINC 内の該当するソース・ファイルの sqludf に入っています。例えば、C の場合、sqludf は QSYSINC/H で見つかります。

表 *81. DBINFO* フィールド

フィールド	データ・タイプ	説明
リレーショナル・データベース	VARCHAR(128)	現行サーバーの名前
権限 ID	VARCHAR(128)	実行時権限 ID

表 81. DBINFO フィールド (続き)

フィールド	データ・タイプ	説明
CCSID 情報	INTEGER INTEGER INTEGER INTEGER INTEGER INTEGER INTEGER INTEGER INTEGER INTEGER CHAR(8)	ジョブの CCSID 情報。3 つの CCSID が 3 セット戻されます。各セット内の 3 つの CCSID を識別する情報は、次のとおりです。 • SBCS CCSID • DBCS CCSID • 混合 CCSID 3 セットの CCSID の後に、3 セットの CCSID のうちのどのセットが該当するかを示す整数と 8 バイトの予約済みスペースが続きます。 CCSID の各設定は、異なるエンコード・スキーム (EBCDIC、ASCII、およびユニコード) のためのものです。 CREATE FUNCTION ステートメントのパラメーターの 1 つとして明示的に CCSID を指定していない場合は、入力ストリングは、関数の実行時にジョブの CCSID でコード化されるものと見なされます。入力ストリングの CCSID がパラメーターの CCSID と同じではない場合は、この外部関数に渡される入力ストリングは、外部プログラムの呼び出しの前に変換されます。
ターゲット列	VARCHAR(128) VARCHAR(128) VARCHAR(128)	ユーザー定義の関数が UPDATE ステートメントの SET 文節の右側に指定されている場合、次の情報がターゲット列を識別します。 • スキーマ名 • 基本表名 • 列名 ユーザー定義の関数が UPDATE ステートメントの SET 文節の右側に指定されなかった場合、これらのフィールドはブランクになります。
バージョンとリリース	CHAR(8)	データベース・マネージャーのバージョン、リリース、および修正レベル。
プラットフォーム	INTEGER	サーバーのプラットフォーム・タイプ。
表関数の列リスト項目の数。	SMALLINT	下記の「表関数の列リスト」フィールドに指定されている表関数列リスト内のゼロでない項目の数。
予約済み	CHAR(24)	将来の利用のため予約済み。

表 *81. DBINFO* フィールド (続き)

フィールド	データ・タイプ	説明
表関数の列リスト	ポインター (16 バイト)	このフィールドは、データベース・マネージャーが動的に割り振る短整数の配列を指すポインターです。「表関数の列リスト項目の数」フィールドに n を指定した場合、最初の n 個の項目のみが対象になります。n は、0 またはそれより大きく、この関数の RETURNS TABLE 文節で定義した結果列の数と同じかまたはそれより小さい値です。個々の値は、このステートメントが表関数から取得する必要がある列の順序番号に対応しています。つまり、1 の値は最初に定義された結果列を意味し、2 は 2 番目に定義された結果列を意味します (以下同様)。これらの値はどのような順序になっていても構いません。 SELECT COUNT(*) FROM TABLE(TF(...)) AS QQ のように、実際の列値を必要としない照会を指定するステートメントの場合は、n はゼロになることがあるという点に注意してください。 この配列により最適化が可能になる場合があります。それは、この関数が、表関数のすべての結果列のすべての値を戻す必要がなくなるからです。特定のコンテキストでは一部の値しか必要とされないことがあり、配列内で番号により識別されている列がこれらの値に相当します。この最適化により関数ロジックが複雑になることもあるので、定義されているすべての列を戻すことを選択することもできます。

EXTERNAL ACTION または **NO EXTERNAL ACTION**

関数が、データベース・マネージャーが管理しないオブジェクトの状態を変更するアクションを行うかどうかを指定します。外部アクションの例には、メッセージの送信やストリーム・ファイルへのレコードの書き込みなどがあります。デフォルトは EXTERNAL ACTION です。

EXTERNAL ACTION

関数が、データベース・マネージャーが管理しないオブジェクトの状態を変更するアクションを行うことができるかを指定します。したがって、関数は、それぞれの連続関数呼び出しで呼び出す必要があります。 EXTERNAL ACTION は、この関数に、外部アクションを持つ他の関数に対する参照が含まれている場合に、指定してください。

NO EXTERNAL ACTION

関数は外部アクションを行いません。この関数は、連続した各関数呼び出しごとに呼び出す必要はありません。

NO EXTERNAL ACTION 関数は、連続した各関数呼び出しごとに呼び出されない場合があるので、EXTERNAL ACTION 関数よりもパフォーマンスが向上する可能性があります。

FENCED または **NOT FENCED**

データベース・マネージャー環境から分離した環境で外部関数を実行するかどうかを指定します。 FENCED がデフォルトです。

FENCED

この関数は別のスレッドで実行されます。

FENCED 関数では、関数の呼び出し間で SQL カーソルをオープン状態のままにすることができません。ただし、あるスレッド内のカーソルは他のスレッド内のカーソルから独立しています。このことは、カーソル名の競合が起きる可能性を低くしています。

NOT FENCED
この関数は、呼び出し元の SQL ステートメントと同じスレッド内で実行できます。

NOT FENCED 関数では、関数の呼び出し間で SQL カーソルをオープン状態のままにすることができます。カーソルをオープン状態のままにしておくことができるため、関数の呼び出し間でカーソル位置も同じに維持されます。ただし、UDF が呼び出し元の SQL ステートメントおよび他の NOT FENCED UDF と同じスレッド内で実行されるためにカーソル名が競合する場合があります。

NOT FENCED 関数は、通常 FENCED 関数よりもパフォーマンスが良好です。

PROGRAM TYPE MAIN または **PROGRAM TYPE SUB**
他の製品との互換性を備えるために、このパラメーターが許可されています。これは、ルーチンの外部プログラムが、プログラム (*PGM) であるか、サービス・プログラム (*SRVPGM) のプロシージャーであるかを示します。

PROGRAM TYPE MAIN
ルーチンがプログラムのメインエントリー・ポイントとして実行することを指定します。外部プログラムは、*PGM オブジェクトでなければなりません。

PROGRAM TYPE SUB
ルーチンがサービス・プログラムのプロシージャーとして実行することを指定します。外部プログラムは、*SRVPGM オブジェクトでなければなりません。

NO FINAL CALL または **FINAL CALL**
関数に対する独立した最初の呼び出し と最終呼び出し を行うかどうかを指定します。 呼び出しのタイプを区別するために、関数は呼び出しのタイプを指定する追加引数を受け取ります。表関数の場合は、FINAL CALL と NO FINAL CALL のどちらが有効になっているかにかかわりなく、*call-type* 引数が常に存在します。その引数では、初回呼び出し、オープン呼び出し、フェッチ呼び出し、クローズ呼び出し、最終呼び出しのいずれかを指定します。

NO FINAL CALL の場合、データベース・マネージャーは、表関数に対して、オープン、フェッチ、クローズの 3 つのタイプの呼び出しだけを実行します。しかし、FINAL CALL が指定されている場合は、オープン、取り出しおよびクローズに加えて、表関数に対して最初の呼び出しと最終呼び出しを行うことができます。

最終呼び出しによって、関数は、獲得済みのすべてのシステム・リソースを解放することができます。最終呼び出しは、SCRATCHPAD キーワードを定義した関数でシステム・リソースを獲得し、そのシステム・リソースをスクラッチパッドに格納するときに使用できます。デフォルトは NO FINAL CALL です。

NO FINAL CALL
関数に対して独立した初回呼び出しと最終呼び出しを実行しない、という動作を指定します。ただし、関数に対して、オープン呼び出し、フェッチ呼び出し、クローズ呼び出しは実行します。さらに、表関数は、呼び出しのタイプを指定した追加の引数を常に受け取ります。

FINAL CALL
関数に対して独立した初回呼び出しと最終呼び出しを実行する、という動作を指定します。これは、スクラッチパッドをどの時点で再初期化するかも制御します。

呼び出しには以下のタイプがあります。

First Call (初回呼び出し**)**
この SQL ステートメントでの関数に対するこの参照についての、関数に対する初回呼び出しを示します。

Open Call (オープン呼び出し**)**
この SQL での表関数結果をオープンするための呼び出しを指定します。

Fetch Call (フェッチ呼び出し**)**
この SQL ステートメントで表関数から行をフェッチするための呼び出しを指定します。

Close Call (クローズ呼び出し**)**
この SQL での表関数結果をクローズするための呼び出しを指定します。

Final Call (最終呼び出し**)**
この関数に対する最後の呼び出しであり、これにより関数はリソースを解放できることを指定します。エラーが起きた場合は、データベース・マネージャーは最終呼び出しを行おうとします。

最終呼び出しが発生するのは以下の場合です。

- ステートメントの終わり : カーソル指向ステートメント用のカーソルがクローズされたとき、またはステートメントの実行が完了したとき。
- トランザクションの終わり : ステートメント処理の正常終了が発生しなかったとき。例えば、何らかの理由によりアプリケーションのロジックがカーソルのクローズをバイパスした場合。

WITH HOLD として定義されているカーソルがオープン状態にあるときにコミット操作が発生した場合は、カーソルがクローズされるかアプリケーションが終了した時点で、最終呼び出しが行われます。

FINAL CALL ではコミット可能な操作は行ってはなりません。FINAL CALL は、COMMIT 操作の一部として呼び出されたクローズ中に実行される可能性があるからです。

ALLOW PARALLEL または **DISALLOW PARALLEL**
関数を並列で実行できるかどうかを指定します。

NOT DETERMINISTIC、EXTERNAL ACTION、FINAL CALL、MODIFIES SQL DATA、SCRATCHPAD のいずれか 1 つ以上の文節を指定すると、デフォルトは DISALLOW PARALLEL になります。それ以外の場合は、ALLOW PARALLEL がデフォルトです。

ALLOW PARALLEL
データベース・マネージャーが関数の並列処理を考慮できることを指定します。データベース・マネージャーは、この関数を呼び出す SQL ステートメントまたはこの関数の中で実行する SQL ステートメントで並列処理を使用しなければならない、というわけではありません。

ALLOW PARALLEL の指定に関する注意点については、NOT DETERMINISTIC、EXTERNAL ACTION、MODIFIES SQL DATA、SCRATCHPAD、FINAL CALL の説明を参照してください。

DISALLOW PARALLEL
データベース・マネージャーが関数の並列処理を使用してはならないことを指定します。

NO SCRATCHPAD または **SCRATCHPAD**
関数に、静的メモリー域が必要か否かを指定します。

NO SCRATCHPAD
これを指定すると、関数では、持続メモリー域を必要としなくなります。

SCRATCHPAD *integer*
これを指定すると、関数には、持続メモリー域の長さ整数が必要となります。この整数に指定できる範囲は、1 から 16,000,000 です。メモリー域を指定しなかった場合、その区域のサイズは 100 バイトになります。パラメーター・スタイル SQL を指定すると、ポインターは、静的ストレージを指す必須パラメーターに続いて渡されます。この関数には、メモリー域が 1 つだけ割り振られます。

スクラッチパッドの有効範囲は SQL です。SQL ステートメント内の関数の参照ごとに、1 つのスクラッチパッドが存在します。例えば、関数 UDFX が SCRATCHPAD キーワードによって定義されていると想定した場合、次の SQL ステートメント内では、UDFX の 2 つの参照に対して 2 つのスクラッチパッドが割り振られます。

```
SELECT A.C1, B.C1
  FROM TABLE(UDFX(:hv1)) AS A, TABLE(UDFX(:hv1)) AS B
```

EXTERNAL
CREATE FUNCTION ステートメントを使用して、外部プログラミング言語によるコードに基づいた新規関数を定義するように指定します。

外部プログラム名 の指定がない場合、外部プログラム名は該当の関数名と同じであると見なされます。

NAME *external-program-name*
SQL ステートメント内でこの関数が呼び出されたときに実行するプログラム、サービス・プログラム、または Java クラスを指定します。この名前は、関数が呼び出される時点でアプリケーション・サーバー上に存在しているプログラム、サービス・プログラム、または Java クラスを示すものでなければなりません。命名オプションが *SYS であり、その名前が修飾されていない場合:

- 関数の呼び出し時に、現行パスを使用して該当のプログラムを検索します。
- COMMENT、GRANT、LABEL、REVOKE の各操作をその関数で実行する時点で、*LIBL を使用して対象のプログラムまたはサービス・プログラムを検索します。

この名前の妥当性は、アプリケーション・サーバーで検査されます。名前の形式が正しくない場合、エラーが戻されます。

このプログラム、サービス・プログラム、または Java クラスは、関数の作成時に存在している必要はありませんが、関数の呼び出し時には存在している必要があります。

CONNECT、SET
CONNECTION、RELEASE、DISCONNECT、COMMIT、ROLLBACK および SET TRANSACTION ステートメントは、関数の外部プログラム内で使用することはできません。

CARDINALITY *bigint*
この関数が戻すものとして予期される行数の見積もりを指定します。この見積もりは、データベース・マネージャーが最適化を行う際に使用されます。 *bigint* は、 0 から 9 223 372 036 854 775 807 までの範囲内でなければなりません。データベース・マネージャーは、CARDINALITY が指定されていない場合は有限値を想定します。

呼び出されるたびに行を戻して表終了状態を戻すことのない表関数は、無限カーディナリティーを持ちます。データを戻す前に結果としての表終了状態を必要とする照会がこのような関数を呼び出すと、その照会は中断しない限り戻りません。表終了状態を戻すことのない表関数を、DISTINCT、GROUP BY、または ORDER BY を伴う照会で使用すべきではありません。

NOT SECURED または **SECURED**
関数が行アクセス制御と列アクセス制御においてセキュアであると見なされるかどうかを指定します。

NOT SECURED
関数が行アクセス制御と列アクセス制御において非セキュアであると見なされることを指定します。これはデフォルトです。

表でアクティブな列アクセス制御が使用されている場合、関数の呼び出し時に、関数の引数が、列マスクが有効になっている列を参照してはなりません。

SECURED
関数が行アクセス制御と列アクセス制御においてセキュアであると見なされることを指定します。このオプションは、C、C++、ILE RPG、ILE COBOL、ILE CL、または Java で作成された外部関数に対して使用できます。プログラムまたはサービス・プログラムは、CREATE FUNCTION ステートメントが実行されるときに存在していなければなりません。

関数は、行の許可または列マスク内で参照される場合は、セキュアとして定義される必要があります。

SET OPTION-statement
パラメーター・デフォルトに使用されるオプションを指定します。各オプション

のデフォルト値は、作成時に有効だったオプションによって異なります。詳しくは、 1696 ページの『SET OPTION』を参照してください。

デフォルト値式を処理するときには、オプション
ALWCPYCTA、CONACC、DATFMT、DATSEP、DECFLTRND、DECMPT、DECRESULT、DFTRDBCOL、
LANGID、SQLCURRULE、SQLPATH、SRTSEQ、TGTRLS、TIMFMT、および TIMSEP が使用されます。オプション
CNULRQD、CNULIGN、COMPILEOPT、EXTIND、NAMING、SQLCA は、CREATE FUNCTIONステートメントでは使用できません。他のオプションは、受け入れられますが、無視されます。

注

ユーザー定義関数の定義に関する一般考慮事項**:** ユーザー定義関数の定義に関する一般情報については、 1070 ページの『CREATE FUNCTION』を参照してください。

REPLACE の規則**:** REPLACE によって外部関数を再作成する場合は、以下のようになります。

- 既存のコメントまたはラベルは破棄されます。
- 別の外部プログラムを指定する場合、
  - 権限を持つユーザーは新しいプログラムにコピーされません。
  - ジャーナル監査は変更されません。
- 上記以外の場合、
  - 権限を持つユーザーは維持されます。オブジェクト所有者は変更されない可能性があります。
  - 現在のジャーナル監査は変更されません。

関数の作成**:** ILE 外部プログラムまたはサービス・プログラムに関連した外部関数が作成されると、その関数に関連したプログラムやサービス・プログラムのオブジェクトへの関数属性の保管が試行されます。 *PGM または *SRVPGM オブジェクトが保管された後、このシステムまたは別のシステムに復元されると、属性が使用されてカタログが更新されます。

外部関数の場合は、次の制約の範囲内で属性を保管することができます。

- 外部プログラム・ライブラリーは、SYSIBM、QSYS、または QSYS2 であってはなりません。
- 外部プログラムは、CREATE FUNCTION ステートメントの発行時に存在していなければなりません。

  システム命名が指定され、外部プログラム名が修飾されない場合は、外部プログラムはライブラリー・リストで検出されなければなりません。
- 外部プログラムは、ILE *PGM オブジェクトか *SRVPGM オブジェクトにする必要があります。
- 外部プログラムには、32 ルーチンの属性が既に入っていてはなりません。

オブジェクトを更新できない場合でも、関数は作成されます。

関数の呼び出し: 外部関数が呼び出されると、その関数は、外部プログラムやサービス・プログラムの作成時に指定された活動化グループであれば、どの活動化グループ内でも実行します。ただし、通常は、関数が呼び出し側プログラムと同じ活動化グループ内で実行するように ACTGRP(*CALLER) を使用します。
ACTGRP(*NEW) は使用できません。

LANGUAGE JAVA の関数は、常にデフォルトの活動化グループ (*DFTACTGRP) で実行されます。したがって、MODIFIES SQL DATA Java 関数を作成するときには、注意が必要です。その Java 関数による変更は、デフォルトの活動化グループで実行されるので、呼び出し側が新しい活動化グループ (*NEW) で実行されていると、トランザクションの問題が発生する可能性があります。

Java 関数に関する注釈: Java 関数を実行するためには、システムに IBM IBM Developer Kit for Java (5770-JV1) をインストールしておく必要があります。インストールされていないと、SQLCODE -443 が戻され、CPDB521 メッセージがジョブ・ログに入ります。

Java 関数の実行中にエラーが発生すると、SQLCODE -443 が戻されます。エラーによっては、関数が実行されていたジョブのジョブ・ログに他のメッセージが入っている場合があります。

代替構文: 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード VARIANT と NOT VARIANT は、NOT DETERMINISTIC と DETERMINISTIC の同義語として使用することができます。
- キーワード NULL CALL と NOT NULL CALL は、CALLED ON NULL INPUT と RETURNS NULL ON NULL INPUT の同義語として使用できます。
- DB2GENERAL の同義語として、値 DB2GENRL を使用できます。
- PARAMETER STYLE 文節のキーワード PARAMETER STYLE はオプションです。
- DETERMINISTIC の同義語として、キーワード IS DETERMINISTIC を使用できます。
- PARAMETER STYLE SQL の同義語として、キーワード PARAMETER STYLE DB2SQL を使用できます。

例

以下の例で作成する表関数は、テキスト管理システム内にある既知の各文書を示す単一の文書 ID 列が入った行を 1 つずつ戻します。最初のパラメーターは特定のサブジェクト・エリアに対応し、2 番目のパラメーターには特定のストリングが入ります。

単一セッションのコンテキストでは、この UDF は常に同じ表を戻すので、DETERMINISTIC として定義されています。 RETURNS 文節が DOCMATCH からの出力を定義している点に注意してください。各表関数について、FINAL CALL

を指定する必要があります。 DOCMATCH の場合の出力のサイズは大きく変化しますが、代表的な値は CARDINALITY 20 なので、最適化プログラムを支援するためにこの値が指定されています。

```
CREATE FUNCTION DOCMATCH (VARCHAR(30), VARCHAR(255))
  RETURNS TABLE (DOCID CHAR(16))
  EXTERNAL NAME 'MYLIB/RAJIV(UDFMATCH)'
  LANGUAGE C
  PARAMETER STYLE SQL
  NO SQL
  DETERMINISTIC
  NO EXTERNAL ACTION
  NOT FENCED
  SCRATCHPAD
  FINAL CALL
  DISALLOW PARALLEL
  CARDINALITY 20
```

CREATE FUNCTION (ソース派生)

この CREATE FUNCTION (ソース化された) ステートメントは、現行サーバーで、他の既存のスカラー関数または集約関数に基づいてユーザー定義の関数を定義します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- スキーマ内に作成する特権。詳しくは、スキーマ内で作成する必要のある権限を参照してください。
- データベース管理者権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- SYSFUNCS カタログ・ビューと SYSPARMS カタログ表の場合
  - 該当の表に対する INSERT 特権、および
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

ソース関数がユーザー定義の関数である場合は、そのソース関数に対して、このステートメントの権限 ID に、少なくとも次のいずれか 1 つを含める必要があります。

- その関数に対する EXECUTE 特権
- データベース管理者権限

ソース化関数を作成するには、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つを含める必要があります。

- 次のシステム権限
  - サービス・プログラム作成 (CRTSRVPGM) コマンドに対する *USE
  - プログラム作成 (CRTPGM) コマンドに対する *USE
- データベース管理者権限

SQL 名が指定され、関数が作成されるライブラリーと同じ名前のユーザー・プロファイルが存在し、しかも、その名前がステートメントの権限 ID と異なっている場合、ステートメントの権限 ID が保持している特権には、少なくとも次のいずれか 1 つを含める必要があります。

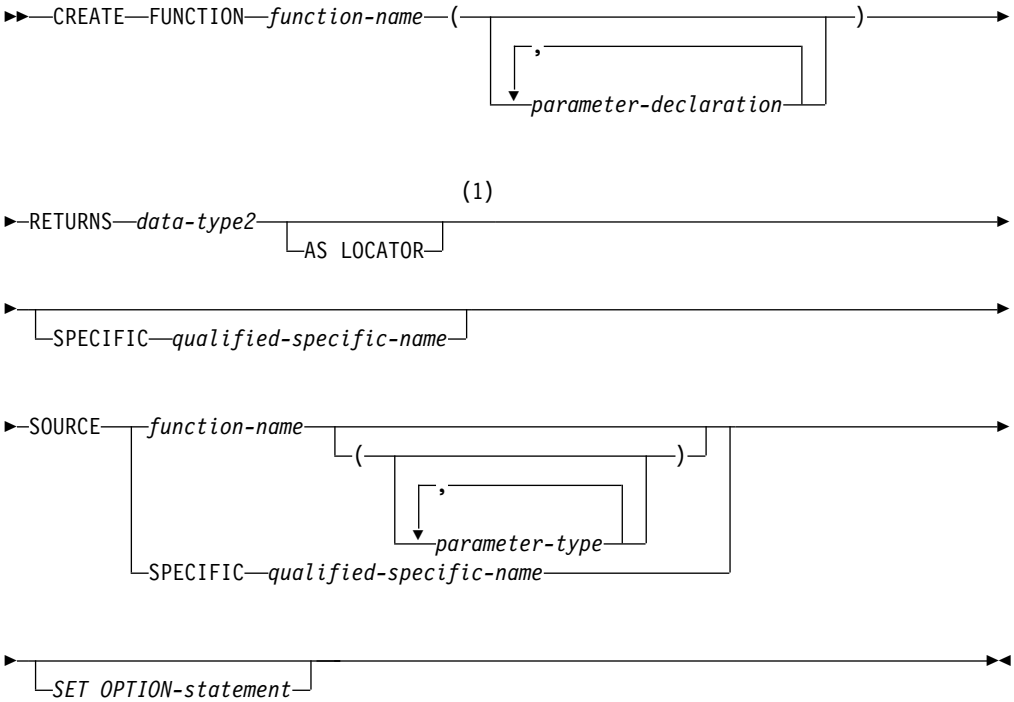
- その名前を持つユーザー・プロファイルに対する *ADD システム権限
- データベース管理者権限

特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別された、それぞれの特殊タイプごとに、
  - その特殊タイプに対する USAGE 特権、および
  - 特殊タイプが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

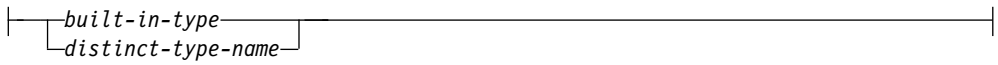
SQL 特権に対応するシステム権限の説明については、『表またはビューへの権限を検査する際の対応するシステム権限』、『関数またはプロシージャーへの権限を検査する際の対応するシステム権限』、および『特殊タイプへの権限を検査する際の対応するシステム権限』を参照してください。

構文



parameter-declaration:

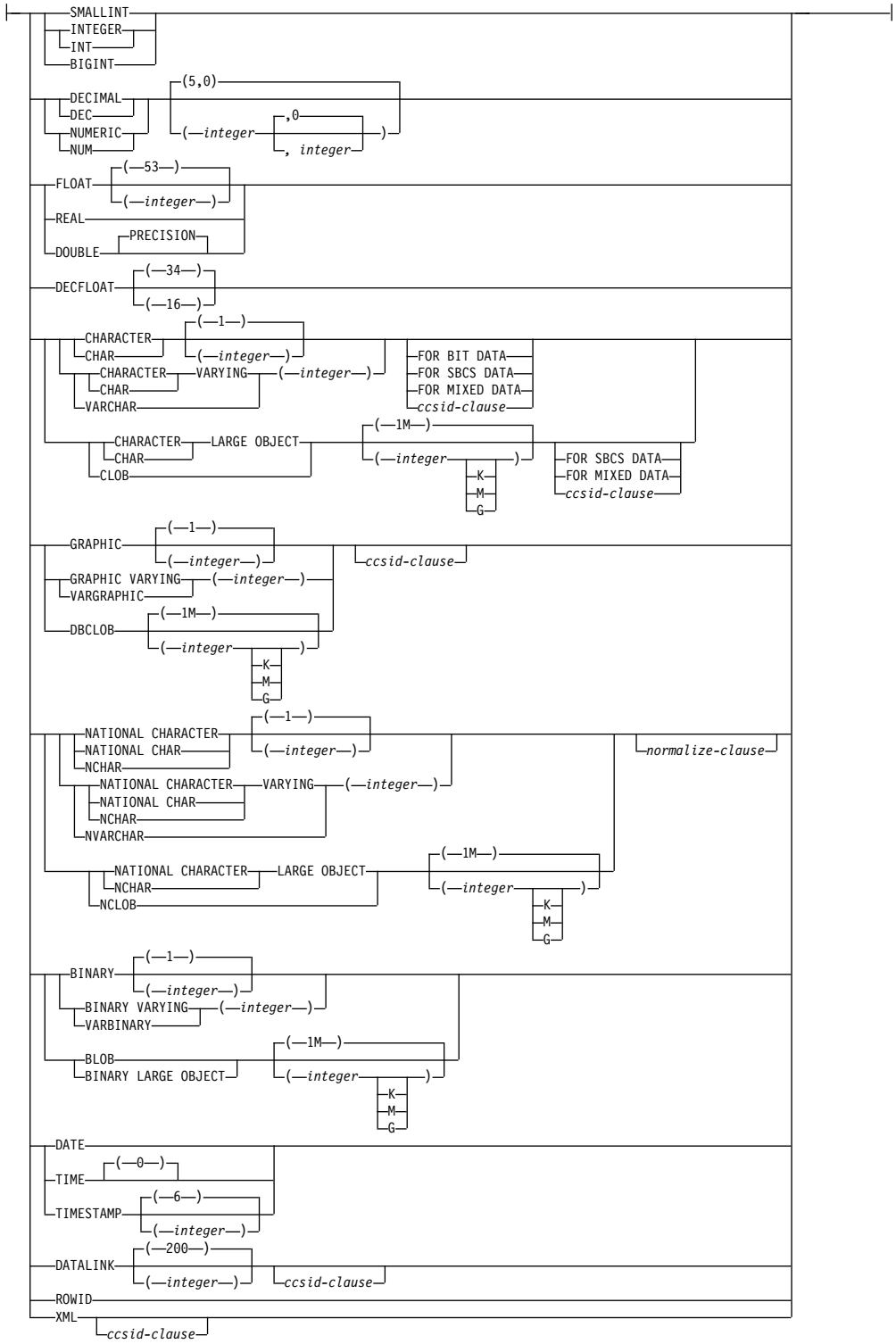
data-type1, data-type2, data-type3:



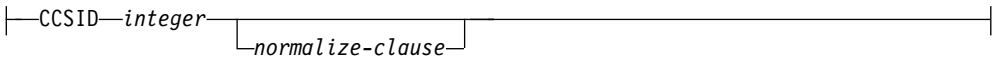
注:

1 RETURNS、SPECIFIC、および SOURCE 文節は、どのような順序で指定しても構いません。

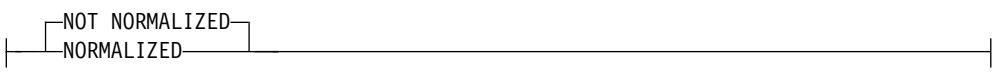
built-in-type:



ccsid-clause:



normalize-clause:



default-clause:

parameter-type:

説明

function-name

ユーザー定義の関数の名前を指定します。名前、スキーマ名、パラメーターの数、およびそれぞれのパラメーターのデータ・タイプ (データ・タイプの長さ、精度、位取り、または CCSID の属性に関係なく) の組み合わせで、現行サーバー上に存在しているユーザー定義の関数を識別してはなりません。

SQL 命名の場合、関数は、暗黙または明示修飾子で指定されたスキーマ内に作成されます。

システム命名の場合、関数は、修飾子で指定されたスキーマ内に作成されます。修飾子を指定しなかった場合:

- CURRENT SCHEMA 特殊レジスターの値が *LIBL である場合、関数は、現行ライブラリー (*CURLIB) 内に作成されます。
- そうでない場合、関数は現行スキーマ内に作成されます。

特殊タイプを指定した既存関数の使用を可能にするために、関数がその既存関数をソースとして作成される場合、その名前は、その既存関数と同じ名前にすることができます。通常、それぞれの関数の関数シグニチャーが固有であれば、複数の関数に同じ名前を指定することができます。

一部の関数名は、システムが使用するために予約されています。 詳しくは、1070 ページの『CREATE FUNCTION』の『スキーマおよび関数名の選択』を参照してください。

(*parameter-declaration,...***)**

関数の入力パラメーターの数とそれぞれのパラメーターのデータ・タイプを指定します。各 *parameter-declaration* は、関数の入力パラメーターを指定します。最

大 2000 のパラメーターを指定することができます。関数は、ゼロまたはそれ以上の入力パラメーターを持つことができます。関数が受け取ると予期しているパラメーターには、それぞれリスト内に 1 個の項目が必要です。関数のパラメーターはすべて入力パラメーターで、ヌル可能です。JAVA の場合は、DECIMAL タイプと NUMERIC タイプ以外の数値パラメーターがヌル可能です。CALLED ON NULL INPUT 関数でそのようなパラメーターの入力がヌル値になると、実行時エラーになります。詳しくは、 1070 ページの『CREATE FUNCTION』の『パラメーターの定義』を参照してください。

parameter-name
パラメーター名を指定します。必須ではありませんが、各パラメーターにパラメーター名を指定することができます。この名前は、パラメーター・リスト内の他のパラメーター名 と同じものであってはなりません。

data-type1
パラメーターのデータ・タイプを指定します。このデータ・タイプは、組み込みデータ・タイプまたは特殊データ・タイプにすることができます。この変数は、配列タイプにすることはできません。

SOURCE 文節で指定された関数の対応するパラメーターのタイプにキャスト可能であれば、任意の有効な SQL データ・タイプを使用できます (詳細は、 109 ページの『データ・タイプ間のキャスト』を参照してください)。ただし、この検査は関数の呼び出し時にエラーが発生しないことを保証するものではありません。詳しくは、『ソース化されたユーザー定義関数の呼び出しに関する考慮事項』を参照してください。

built-in-type
入力パラメーターのデータ・タイプは組み込みデータ・タイプです。それぞれの組み込みデータの詳細については、 1238 ページの『CREATE TABLE』を参照してください。

distinct-type-name
入力パラメーターのデータ・タイプは特殊タイプです。パラメーターの長さ、精度、または位取り属性は、特殊タイプのソース・タイプの属性 (CREATE TYPE で指定された属性) と同じになります。詳しくは、1328 ページの『CREATE TYPE (特殊)』を参照してください。

スキーマ名なしの特殊タイプを指定すると、データベース・マネージャーは、 SQL パス上のスキーマを検索することでそのスキーマ名を解決します。

外部関数にソース化された関数には、データ・リンクは使用できません。

CCSID が指定されている場合、関数に渡される前に、パラメーターはその CCSID に変換されます。CCSID が指定されていない場合は、CCSID は、関数の呼び出し時点における現行サーバーのデフォルトの CCSID によって決まります。

AS LOCATOR
これを指定すると、入力パラメーターは、実際の値ではなく、値のロケーターになります。 AS LOCATOR は、入力パラメーターに LOB データ・タイプや LOB データ・タイプをベースとする特殊タイプが指定されている場

合にのみ、指定することができます。 AS LOCATOR を指定した場合、FOR SBCS DATA または FOR MIXED DATA を指定してはなりません。

AS LOCATOR 文節について詳しくは、 1070 ページの『CREATE FUNCTION』の『パラメーターに AS LOCATOR を指定する』を参照してください。

default-clause

パラメーターのデフォルト値を指定します。デフォルト値は、定数、特殊レジスター、グローバル変数、式、またはキーワード NULL にすることができます。式は、集約関数および列名を含まない、 196 ページの『式』で定義されている任意の式です。デフォルト値が指定されていない場合、パラメーターにデフォルト値がないため、呼び出し時に省略できません。式ストリングの最大長は 64K です。

デフォルトの式は、パラメーターのデータ・タイプに対して割り当ての互換性がなければなりません。

デフォルト式内でリスト中の数値定数を区切る区切り記号として使用するコンマの後には、スペースが 1 つ必要です。

デフォルト式の中で参照されるすべてのオブジェクトは、関数が作成されるときに存在している必要があります。

配列タイプのパラメーターにデフォルトを指定することはできません。

RETURNS

関数の結果を指定します。

data-type2

列のデータ・タイプを指定します。この列はNULL 可能です。可能なデータ・タイプは、組み込みデータ・タイプ (LONG VARCHAR、LONG VARGRAPHIC、DataLink を除く) または特殊タイプ (DataLink をベースとしないもの) です。 配列タイプにすることはできません。

ソース関数の結果タイプからキャスト可能なものであれば、有効な SQL データ・タイプを使用できます。 (データ・タイプのキャストについては、109 ページの『データ・タイプ間のキャスト』を参照してください) ただし、この検査はこの新しい関数の呼び出し時にエラーが発生しないことを保証するものではありません。詳しくは、『ソース化されたユーザー定義関数の呼び出しに関する考慮事項』を参照してください。

AS LOCATOR

これを指定すると、関数は、実際の値ではなく、値のロケーターを戻します。 AS LOCATOR は、関数の出力に LOB データ・タイプや LOB データ・タイプをベースとする特殊タイプが指定されている場合にのみ、指定することができます。 AS LOCATOR を指定した場合、FOR SBCS DATA または FOR MIXED DATA を指定してはなりません。 AS LOCATOR 文節は、SQL 関数をソースとする関数に使用することはできません。

AS LOCATOR 文節について詳しくは、 1070 ページの『CREATE FUNCTION』の『パラメーターに AS LOCATOR を指定する』を参照してください。

SPECIFIC *qualified-specific-name*
関数の固有名を指定します。特定名の詳細については、 1070 ページの『CREATE FUNCTION』の『関数に特定の名前を指定する』を参照してください。

SOURCE
定義中の新規の関数がソース化関数になることを指定します。ソース化関数 *(sourced function)* は、別の関数 (ソース関数*; source function*) によってインプリメントされます。関数は、現行サーバーに存在するスカラー関数または集約関数であり、次に示すいずれかのタイプの関数であることが必要です。

- CREATE FUNCTION ステートメントによって定義された関数
- CREATE TYPE ステートメントによって生成された cast 関数
- 組み込み関数

ソース関数が組み込み関数でない場合、特定の関数を識別するには、名前、関数シグニチャー、または特定名を使用できます。

ソース関数が組み込み関数である場合は、組み込み関数の関数シグニチャーを SOURCE 文節に指定する必要があります。ソース関数は、次の組み込み関数のいずれかであってはなりません (特定の構文が示されている場合は、示された形式のみが指定できません)。:

- ARRAY_AGG
- 複数の引数を指定する場合の BINARY
- 複数の引数を指定する場合の BLOB
- CARDINALITY
- 複数の引数を指定する場合の CHAR
- 複数の引数を指定する場合の CLOB
- COALESCE
- CONTAINS
- DATAPARTITIONNAME
- DATAPARTITIONNUM
- 複数の引数を指定する場合の DBCLOB
- DBPARTITIONNAME
- DBPARTITIONNUM
- 複数の引数を指定する場合の DECFLOAT
- 複数の引数を指定する場合の DECIMAL
- 複数の引数を指定する場合の DECRYPT_BIN
- 複数の引数を指定する場合の DECRYPT_BINARY
- 複数の引数を指定する場合の DECRYPT_BIT
- 複数の引数を指定する場合の DECRYPT_CHAR
- 複数の引数を指定する場合の DECRYPT_DB
- EXTRACT
- 複数の引数を指定する場合の GRAPHIC
- HASH_VALUES

- HASHED_VALUE
- LAND
- LOR
- MAX
- MAX_CARDINALITY
- MIN
- NODENAME
- NODENUMBER
- PARTITION
- PERCENTILE_CONT
- PERCENTILE_DISC
- POSITION
- RAISE_ERROR
- RID
- RRN
- SCORE
- 複数の引数が指定されている SECOND
- STRIP
- SUBSTRING
- 第 2 引数が整数である場合の TIMESTAMP
- 3 つ以上の引数が指定されている TIMESTAMP_FORMAT
- 複数の引数を指定する場合の TRANSLATE
- TRIM
- TRIM_ARRAY
- VALUE
- 複数の引数を指定する場合の VARBINARY
- 複数の引数を指定する場合の VARCHAR
- 複数の引数を指定する場合の VARGRAPHIC
- 3 つ以上の引数が指定されている VERIFY_GROUP_FOR_USER
- XMLAGG
- XMLATTRIBUTES
- XMLCOMMENT
- XMLCONCAT
- XMLDOCUMENT
- XMLELEMENT
- XMLFOREST
- XMLGROUP
- XMLNAMESPACES
- XMLPARSE
- XMLPI

- XMLROW
- XMLSERIALIZE
- XMLTEXT
- XMLVALIDATE
- XOR
- XSLTRANSFORM
- 複数の引数を指定する場合の ZONED

function-name
ソース関数として使用する関数を関数名で識別します。ソース関数のパラメーターはいくつでも定義できます。指定したスキーマまたは暗黙スキーマの中に、指定した名前を使用して定義された関数が複数ある場合は、エラーが戻されます。

非修飾の関数名 が指定されている場合、SQL パスを使用して関数を探します。この名前の関数をただ 1 つ含み、ユーザーが EXECUTE 権限を持っている最初のスキーマがデータベース・マネージャーによって選択されます。関数が見つからない場合や、この名前の関数が複数存在するスキーマをデータベース・マネージャーが検出した場合は、エラーが返されます。

function-name (parameter-type, ...)
ソース関数として使用する関数を、関数を一意的に識別する関数シグニチャーで識別します。 *function-name (parameter-type,...)* は、現行サーバーにおいて指定されたシグニチャーを持つ関数を識別する必要があります。指定されたパラメーターは、関数の作成時に指定された、対応する位置にあるデータ・タイプと一致していなければなりません。関数インスタンスを識別する場合、データ・タイプの数とデータ・タイプの論理連結が使用されます。データ・タイプの同義語は、一致として扱われます。デフォルトがあるパラメーターは、このシグニチャーに含まれていなければなりません。

function-name() を指定する場合、識別される関数にパラメーターを使用することはできません。

組み込み関数をソース関数として使用するには、この構文のバリエーションを使用する必要があります。

function-name
ソース関数の名前を識別します。非修飾名が指定されている場合、SQL パスのスキーマが検索されます。指定しなければ、データベース・マネージャーは指定したスキーマ内で関数を検索します。

パラメーター・タイプ,...
関数のパラメーターを識別します。

非修飾の特殊タイプ名を指定する場合、データベース・マネージャーはその特殊タイプのスキーマ名を解決するための SQL パスを検索します。

このコンテキストで指定する一部のデータ・タイプには、空の括弧を使用できます。長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、以下のいずれかの指定方法を使用します。

- 空の括弧は、データベース・マネージャーがデータ・タイプの一致の判別に際して属性を無視することを示します。例えば、DEC() は、DEC(7,2) のデータ・タイプで定義された関数のパラメーターに一致するものとみなされます。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を指定した場合、その値は、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプが FLOAT の場合、突き合わせはデータ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度は指定された値に厳密に一致している必要はありません。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。暗黙の長さは、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

サブタイプまたは CCSID 属性のあるデータ・タイプの場合は、FOR DATA 文節または CCSID 文節の指定はオプションです。いずれの文節も指定しないと、データ・タイプが一致するかどうかを判定する場合に、データベース・マネージャーが属性を無視することを示します。どちらか一方の文節を指定する場合は、CREATE FUNCTION ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

AS LOCATOR
関数が、このパラメーターのロケーターを受け取るように定義されることを示します。 AS LOCATOR を指定する場合は、データ・タイプは LOB または LOB に基づく特殊タイプでなければなりません。 AS LOCATOR を指定した場合、FOR SBCS DATA または FOR MIXED DATA を指定してはなりません。 AS LOCATOR を指定して、長さが明示的に指定された場合、データ・タイプの長さは無視されます。

AS LOCATOR 文節について詳しくは、 1070 ページの『CREATE FUNCTION』の『パラメーターに AS LOCATOR を指定する』を参照してください。

SPECIFIC *qualified-specific-name*
ソース関数として使用する関数を特定名で識別します。この修飾特定名は、指定されたスキーマに存在している特定の関数を識別していなければなりません。

SET OPTION-statement
パラメーター・デフォルトに使用されるオプションを指定します。各オプションのデフォルト値は、作成時に有効だったオプションによって異なります。詳しくは、 1696 ページの『SET OPTION』を参照してください。デフォルト値式を処理するときには、オプション
ALWCPYDTA、CONACC、DATFMT、DATSEP、DECFLTRND、DECMPT、DECRESULT、DFTRDBCOL、
LANGID、SQLCURRULE、SQLPATH、SRTSEQ、TGTRLS、TIMFMT、および TIMSEP が使用されます。オプション

CNULRQD、CNULIGN、COMPILEOPT、EXTIND、NAMING、SQLCA は、CREATE FUNCTIONステートメントでは使用できません。他のオプションは、受け入れられますが、無視されます。

作成しようとしている関数の入力パラメーターの数は、ソース関数のパラメーターの数と同じにする必要があります。それぞれの入力パラメーターのデータ・タイプが、ソース関数の対応パラメーターと同じでなかったり、その対応パラメーターにキャストできない場合は、エラーが発生します。ソース関数の最終結果のデータ・タイプは、ソース化関数の結果と一致させるか、あるいは、その結果にキャストできるようにする必要があります。

CCSID が指定され、戻りデータの CCSID が異なる CCSID でコード化されている場合、データは指定された CCSID に変換されます。

CCSID が指定されていない場合、戻りデータの CCSID が異なる CCSID でコード化されている場合には、戻りデータはジョブの CCSID (グラフィック・ストリング戻り値の場合は、ジョブに関連したグラフィック CCSID) に変換されます。 変換時に文字が失われるのを防ぐために、関数から戻される文字をすべて表現できる CCSID を明示的に指定することを考慮してください。 これは、データ・タイプがグラフィック・ストリング・データの場合に特に重要です。 この場合、CCSID 1200 または 13488 (ユニコード・グラフィック・ストリング・データ) を使用することを考慮してください。

注

ユーザー定義関数の定義に関する一般考慮事項: ユーザー定義関数の定義に関する一般情報については、 1070 ページの『CREATE FUNCTION』を参照してください。

関数の所有権: SQL 名が指定されている場合、

- 作成した関数が入れられるスキーマと同じ名前のユーザー・プロファイルが存在する場合、関数の所有者 はそのユーザー・プロファイルです。
- その他の場合は、関数の所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

システム名を指定した場合は、関数の所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

関数の権限: SQL 名を使用する場合は、関数は、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成されます。 システム名を使用する場合、関数は、スキーマの作成権限 (CRTAUT) パラメーターによって決められる *PUBLIC に対する権限を使用して作成されます。

関数の所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、その関数に対する権限が与えられます。

ソース化されたユーザー定義関数の呼び出しに関する考慮事項: ソース化関数が呼び出されると、関数のそれぞれの引数が関数に定義された関連パラメーターに割り当

てられます。次に、その値は基礎となる関数の対応するパラメーターのデータ・タイプに (必要に応じて) キャストされます。割り当てまたはキャストのいずれかで、エラーが発生する可能性があります。例: 関数のパラメーターのデータ・タイプ、および長さ、または精度の属性と一致する関数への入力に渡された引数は、基礎となるソース関数の対応するパラメーターの長さが短かったり、精度が低かったりすると、キャストできない可能性があります。基礎となる関数の対応するパラメーターの属性以下の属性を使用して、ソース化関数のパラメーターのデータ・タイプを定義することをお勧めします。

基礎となる関数の結果は、ソース化関数の RETURNS データ・タイプに割り当てられます。基礎となる関数の RETURNS データ・タイプは、ソース関数の RETURNS データ・タイプにキャストできない場合があります。これが生じる可能性があるのは、新規のソース関数の RETURNS データ・タイプが、基礎となる関数の RETURNS データ・タイプより長さが短かったり、精度が低い場合です。例えば、以下の関数が存在すると想定して関数 A を呼び出した場合に、エラーが発生する場合があります。関数 A は INTEGER を戻します。関数 B は SMALLINT を戻すよう定義されているソース化関数であり、その定義は関数 A を SOURCE 文節で参照します。基礎となる関数の RETURNS データ・タイプを定義する属性以上の属性を使用して、ソース化関数の RETURNS データ・タイプを定義することをお勧めします。

関数がユーザー定義関数に基づく場合の考慮事項**:** ソース化関数を外部スカラー関数をもとにして直接的または間接的に作成する場合、そのソース化関数は、その外部スカラー関数の属性を継承します。これには、ソース化関数のいくつかの層が含まれる場合があります。例えば、関数 A が関数 B をソースとし、関数 B は関数 C をソースとしているとします。また、関数 C は外部スカラー関数であるとします。関数 A と B は、関数 C のすべての属性を継承します。

ソース派生関数は、それが直接的に基づいているユーザー定義関数からセキュア属性を継承します。組み込み関数に基づいているソース派生関数は、常にセキュアです。

関数の作成**:** ソース化関数が作成されると、その関数を表す小さなサービス・プログラム・オブジェクトが作成されます。このサービス・プログラムが別のシステムに保管および復元されると、 CREATE FUNCTION ステートメントの属性は自動的にそのシステム上のカタログに追加されます。

例

例 1: 特殊タイプ HATSIZE が定義され、組み込みデータ・タイプ INTEGER に基づいていると想定します。異なる部門の平均の帽子サイズを計算するために、AVG 関数を定義することができます。組み込み関数 AVG をベースにしたソース化関数を作成します。

```
CREATE FUNCTION AVG (HATSIZE)
  RETURNS HATSIZE
  SOURCE AVG (INTEGER)
```

ソース関数は組み込み関数であるため、 SOURCE 文節の構文には明示的なパラメーター・リストが含まれます。

特殊タイプ HATSIZE が作成された際に、2 つのキャスト関数が生成されました。これにより、HATSIZE を引数用に INTEGER にキャストし、 INTEGER を関数の結果用に HATSIZE にキャストすることができます。

例 *2:* Smith が外部スカラー関数 CENTER を自分のスキーマに作成した後で、この関数の使用が必要になります。ただし、この関数の呼び出し時に、1 つの INTEGER 引数と 1 つの DOUBLE 引数ではなく、2 つの INTEGER 引数を受け入れさせる必要があります。CENTER をベースにしたソース化関数を作成します。

```
CREATE FUNCTION MYCENTER (INTEGER, INTEGER)
   RETURNS DOUBLE
   SOURCE SMITH.CENTER (INTEGER, DOUBLE);
```

CREATE FUNCTION (SQL スカラー)

CREATE FUNCTION (SQL スカラー) ステートメントは、現行サーバー上に SQL 関数を作成します。この関数は、単一の結果を戻します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、あるいは対話式に実行することもできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- スキーマ内に作成する特権。詳しくは、スキーマ内で作成する必要のある権限を参照してください。
- データベース管理者権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- SYSFUNCS カタログ・ビューと SYSPARMS カタログ表の場合
  - 該当の表に対する INSERT 特権、および
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

このステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- 次のシステム権限
  - サービス・プログラム作成 (CRTSRVPGM) コマンドに対する *USE
- データベース管理者権限

特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別された、それぞれの特殊タイプごとに、
  - その特殊タイプに対する USAGE 特権、および
  - 特殊タイプが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

SECURED 属性が指定されるか、または、関数がセキュアで OR REPLACE が指定される場合は、次のとおりです。

- このステートメントの許可 ID には、セキュリティー管理者権限 がなければなりません。 21 ページの『管理権限』を参照してください。

既存の関数に置き換えるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

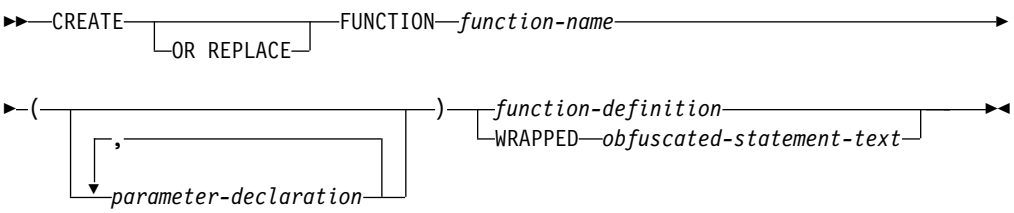
- 次のシステム権限

- この関数に関連したサービス・プログラム・オブジェクトに対する *OBJMGT システム権限
- この関数を削除するために必要な全権限
- SYSFUNCS カタログ・ビューと SYSPARMS カタログ表に対する *READ システム権限

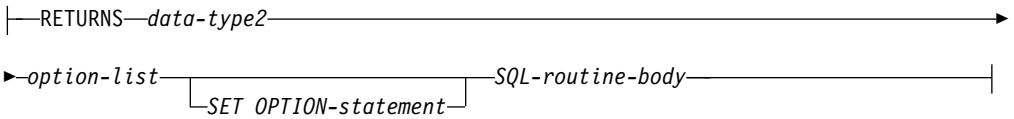
- データベース管理者権限

SQL 特権に対応するシステム権限については、『表またはビューへの権限を検査する際の対応するシステム権限』および 『特殊タイプへの権限を検査する際の対応するシステム権限』を参照してください。

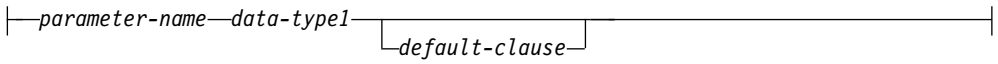
構文



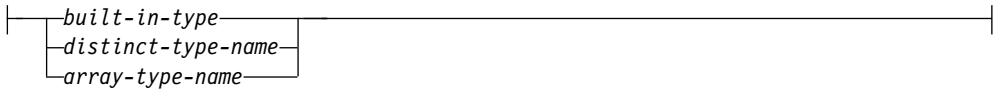
function-definition:



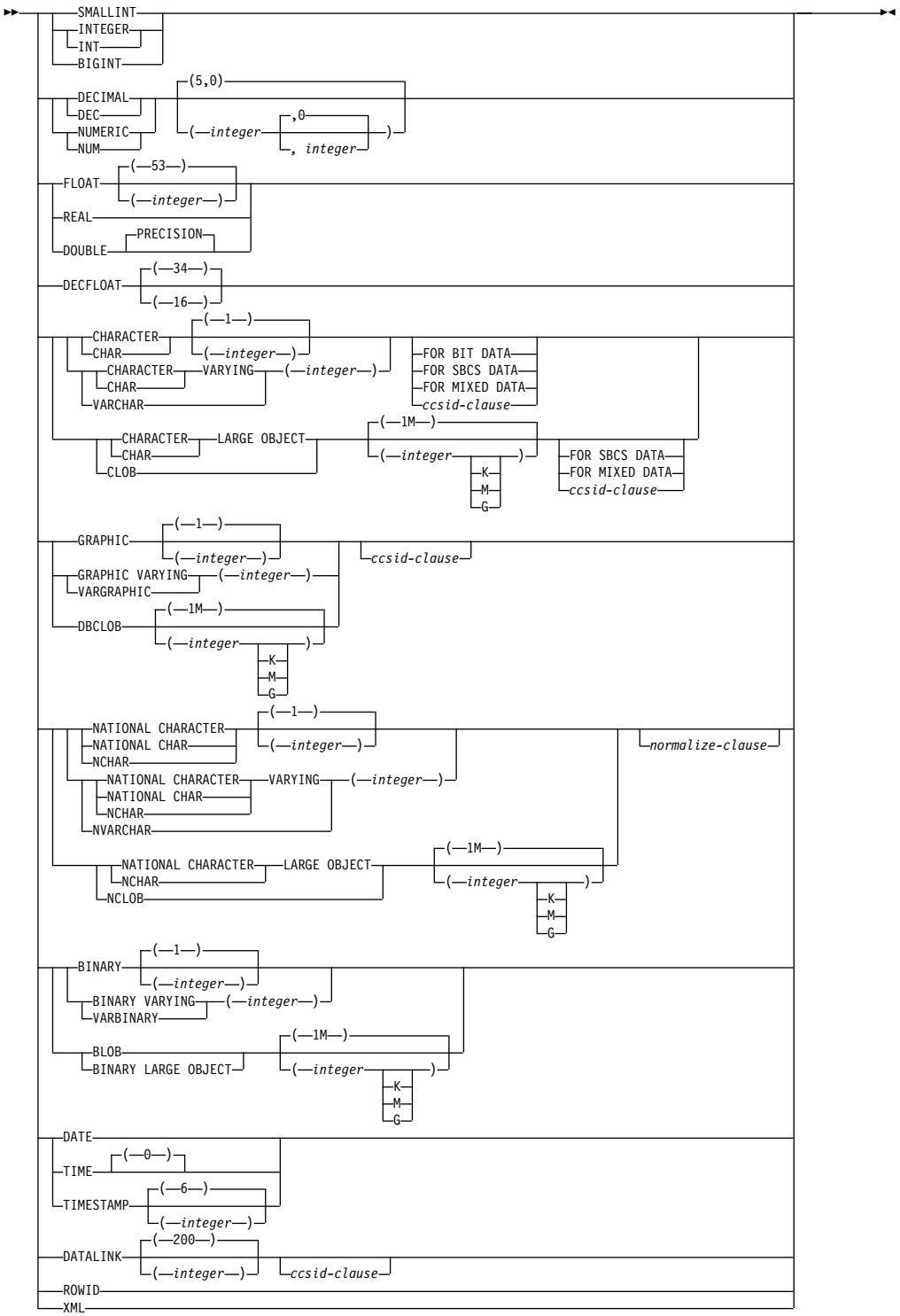
parameter-declaration:



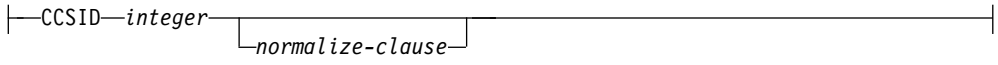
data-type1, data-type2:



built-in-type



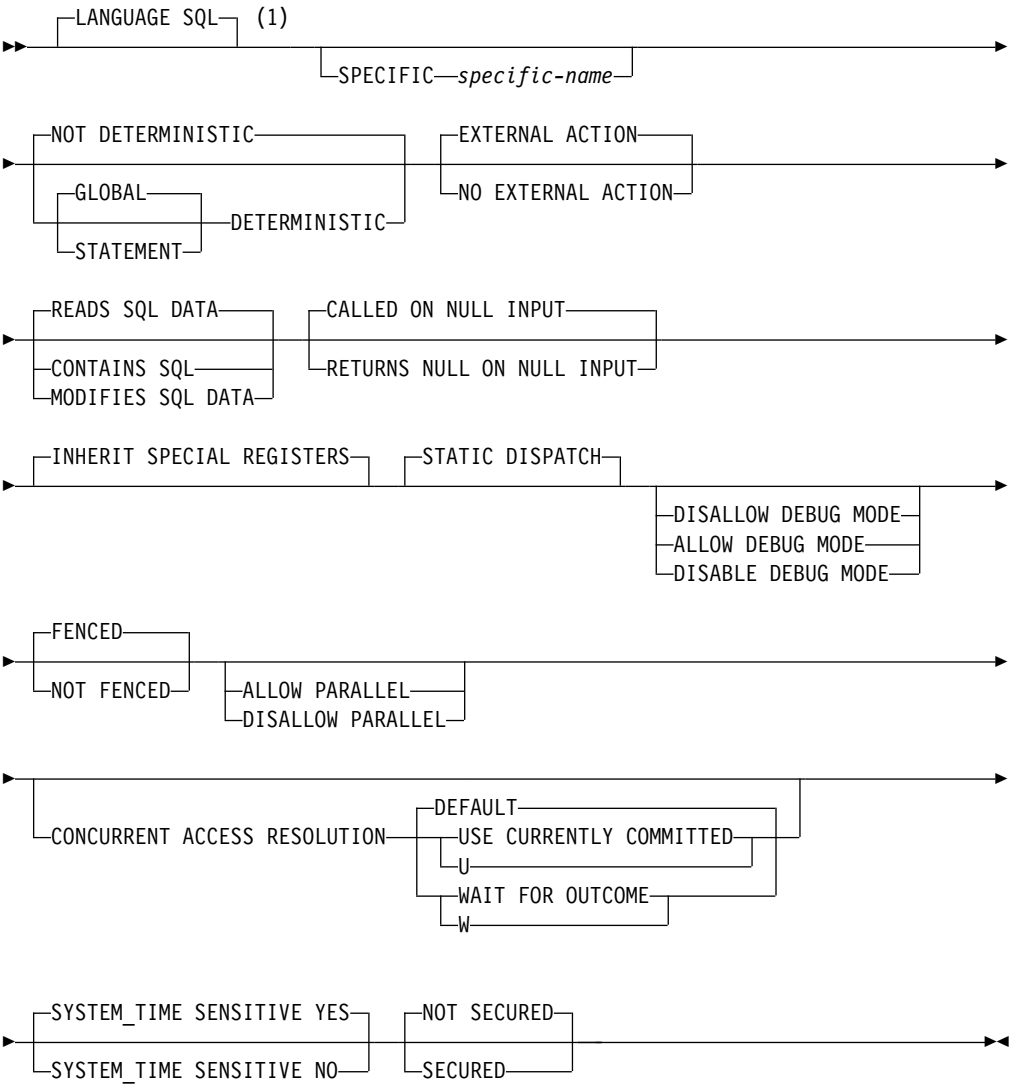
ccsid-clause:



normalize-clause:

default-clause:

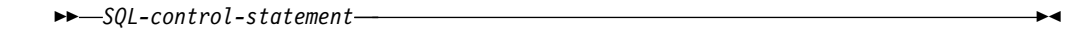
option-list



注:

1 この文節とこの後の option-list にある文節は、どのような順番で指定してもかまいません。各文節を指定できるのは、それぞれ 1 回だけです。

SQL-routine-body



説明

OR REPLACE
現行サーバーにこの関数の定義が存在する場合に、その定義を置き換える、という動作を指定します。実際には、カタログで既存の定義を削除してから新しい定義に置き換える、という動作になりますが、例外として、この関数に対して与えられていた特権は影響を受けません。現行サーバーにこの関数の定義が存在しなければ、このオプションは無視されます。既存の関数を置き換えるには、新しい定義の *specific-name* および *function-name* が古い定義の *specific-name* および *function-name* と同じであるか、新しい定義のシグニチャーが古い定義のシグニチャーと一致している必要があります。そうでなければ、新しい関数が作成されます。

function-name
ユーザー定義の関数の名前を指定します。名前、スキーマ名、パラメーターの数、およびそれぞれのパラメーターのデータ・タイプ (データ・タイプの長さ、精度、位取り、または CCSID の属性に関係なく) の組み合わせで、現行サーバー上に存在しているユーザー定義の関数を識別してはなりません。

SQL 命名の場合、関数は、暗黙または明示修飾子で指定されたスキーマ内に作成されます。

システム命名の場合、関数は、修飾子で指定されたスキーマ内に作成されます。修飾子を指定しなかった場合:

- CURRENT SCHEMA 特殊レジスターの値が *LIBL である場合、関数は、現行ライブラリー (*CURLIB) 内に作成されます。
- そうでない場合、関数は現行スキーマ内に作成されます。

通常、それぞれの関数の関数シグニチャーが固有であれば、複数の関数に同じ名前を指定することができます。

一部の関数名は、システムが使用するために予約されています。 詳しくは、1070 ページの『CREATE FUNCTION』の『スキーマおよび関数名の選択』を参照してください。

(*parameter-declaration,...***)**
関数の入力パラメーターの数とそれぞれのパラメーターのデータ・タイプを指定します。各 *parameter-declaration* は、関数の入力パラメーターを指定します。最大 2000 のパラメーターを指定することができます。関数は、ゼロまたはそれ以上の入力パラメーターを持つことができます。関数が受け取ると予期しているパラメーターには、それぞれリスト内に 1 個の項目が必要です。関数のパラメーターはすべて入力パラメーターで、ヌル可能です。詳しくは、1070 ページの『CREATE FUNCTION』の『パラメーターの定義』を参照してください。

parameter-name
パラメーター名を指定します。この名前は、関数の本体に含まれるパラメーターを参照するのに使用されます。この名前は、パラメーター・リスト内の他のパラメーター名 と同じものであってはなりません。

data-type1
入力パラメーターのデータ・タイプを指定します。このデータ・タイプは、組み込みデータ・タイプまたは特殊データ・タイプにすることができます。

built-in-type
組み込みデータ・タイプを指定します。それぞれの組み込みデータ・タイプの詳細については、 1238 ページの『CREATE TABLE』を参照してください。

distinct-type-name
特殊タイプを指定します。パラメーターの長さ、精度、または位取り属性は、特殊タイプのソース・タイプの属性 (CREATE TYPE で指定された属性) と同じになります。特殊タイプの作成についての詳細は、1328 ページの『CREATE TYPE (特殊)』を参照してください。

特殊タイプの名前が修飾されていない場合、データベース・マネージャーは、SQL パス上のスキーマを検索することでそのスキーマ名を解決します。

配列タイプ名
配列タイプを指定します。

配列タイプの名前が修飾されていない場合、データベース・マネージャーは、SQL パス上のスキーマを検索することでそのスキーマ名を解決します。

CCSID が指定されている場合、関数に渡される前に、パラメーターはその CCSID に変換されます。CCSID が指定されていない場合は、CCSID は、関数の呼び出し時点における現行サーバーのデフォルトの CCSID によって決まります。

配列タイプ、XML タイプ、または LOB タイプのパラメーターは、読み取り専用です。

default-clause
パラメーターのデフォルト値を指定します。デフォルト値は、定数、特殊レジスター、グローバル変数、式、またはキーワード NULL にすることができます。式は、集約関数および列名を含まない、 196 ページの『式』で定義されている任意の式です。デフォルト値が指定されていない場合、パラメーターにデフォルト値がないため、呼び出し時に省略できません。式ストリングの最大長は 64K です。

デフォルトの式は、パラメーターのデータ・タイプに対して割り当ての互換性がなければなりません。

デフォルト式内でリスト中の数値定数を区切る区切り記号として使用するコンマの後には、スペースが 1 つ必要です。

デフォルト式の中で参照されるすべてのオブジェクトは、関数が作成されるときに存在している必要があります。

配列タイプのパラメーターにデフォルトを指定することはできません。

RETURNS
関数の結果を指定します。

data-type2
この関数で返す式を指定します。式の結果のデータ・タイプは、ストレージ割り当て規則に基づいて、RETURNS 文節で定義するデータ・タイプに代入できるデータ・タイプでなければなりません。詳しくは、 113 ページの『割り当ておよび比較』を参照してください。

あらゆる組み込みデータ・タイプ (ただし、LONG VARCHAR または LONG VARGRAPHIC は除く)、特殊タイプ、または配列タイプを指定することができます。

CCSID が指定され、戻りデータの CCSID が異なる CCSID でコード化されている場合、データは指定された CCSID に変換されます。

CCSID が指定されていない場合、戻りデータの CCSID が異なる CCSID でコード化されている場合には、戻りデータはジョブの CCSID (グラフィック・ストリング戻り値の場合は、ジョブに関連したグラフィック CCSID) に変換されます。変換時に文字が失われるのを防ぐために、関数から戻される文字をすべて表現できる CCSID を明示的に指定することを考慮してください。 これは、データ・タイプがグラフィック・ストリング・データの場合に特に重要です。 この場合、CCSID 1200 または 13488 (ユニコード・グラフィック・ストリング・データ) を使用することを考慮してください。

LANGUAGE SQL
これは SQL 関数であることを指定します。

SPECIFIC *specific-name*
関数の固有名を指定します。特定名の詳細については、 1070 ページの『CREATE FUNCTION』の『関数に特定の名前を指定する』を参照してください。

GLOBAL DETERMINISTIC または **STATEMENT DETERMINISTIC** または **NOT DETERMINISTIC**
関数が同じ入力引数を指定して呼び出されるたびに、その関数が同じ結果を戻すかどうかを指定します。デフォルトは NOT DETERMINISTIC です。

NOT DETERMINISTIC
関数が同じ入力引数を指定して呼び出されるたびに、その関数が同じ結果を戻さない場合があることを指定します。関数は、結果に影響を与えるいくつかの状態値によって変わってきます。データベース・マネージャーは、SQL ステートメントの最適化時にこの情報を使用します。非 deterministic 関数の例として、乱数を生成する関数があります。

非決定的関数を並列タスクで実行すると、間違った結果が返される可能性があります。このような関数には DISALLOW PARALLEL 文節を指定してください。

NOT DETERMINISTIC は、特殊レジスター、非決定的関数、またはシーケンスに対する参照がこの関数に含まれている場合に指定してください。

GLOBAL DETERMINISTIC
関数が同じ入力引数を指定して呼び出されるたびに、その関数が常に同じ結果を戻すかを指定します。データベース・マネージャーは、SQL ステートメントの最適化時にこの情報を使用します。照会オプティマイザーは、グローバル deterministic なスカラー関数の結果を キャッシュに入れるよう選

択する場合があります。[93]グローバル deterministic 関数の例としては、入力引数の平方根を計算する関数があります。

STATEMENT DETERMINISTIC
関数が同じ入力引数を使用して呼び出されるたびに、同じ結果を戻さない可能性があるが、単一 SQL ステートメント内での関数の複数の呼び出しは、deterministic と見なされることを指定します。照会オプティマイザーは、ステートメント deterministic なスカラー関数の結果をキャッシュに入れません。[94]ステートメント deterministic 関数の例としては、通貨変換を実行する関数があります。

EXTERNAL ACTION または NO EXTERNAL ACTION
関数が、データベース・マネージャーが管理しないオブジェクトの状態を変更するアクションを行うかどうかを指定します。外部アクションの例には、メッセージの送信やストリーム・ファイルへのレコードの書き込みなどがあります。デフォルトは EXTERNAL ACTION です。

EXTERNAL ACTION
関数が、データベース・マネージャーが管理しないオブジェクトの状態を変更するアクションを行うことができるかを指定します。したがって、関数は、それぞれの連続関数呼び出しで呼び出す必要があります。 EXTERNAL ACTION は、この関数に、外部アクションを持つ他の関数に対する参照が含まれている場合に、指定してください。

NO EXTERNAL ACTION
関数は外部アクションを行いません。この関数は、連続した各関数呼び出しごとに呼び出す必要はありません。

NO EXTERNAL ACTION 関数は、連続した各関数呼び出しごとに呼び出されない場合があるので、EXTERNAL ACTION 関数よりもパフォーマンスが向上する可能性があります。

CONTAINS SQL、READS SQL DATA、または MODIFIES SQL DATA
関数が実行できる SQL ステートメントおよびネストされたルーチンの種別を指定します。データベース・マネージャーは、関数と、関数からローカルで呼び出すすべてのルーチンが発行する SQL ステートメントが、この指定と整合しているかどうかを検査します。ネストされたリモート・ルーチンが呼び出された場合、検査は実行されません。各ステートメントの分類については、 1855 ページの『付録 B. SQL ステートメントの特性』を参照してください。デフォルトは、READS SQL DATA です。このオプションは、すべてのパラメーター・デフォルト式に適用されます。

READS SQL DATA
この関数が、データ・アクセス種別 READS SQL DATA、CONTAINS SQL、または NO SQL のステートメントを実行できるように指定します。この関数は、データの変更を行う SQL ステートメントは実行できません。

---

93. 関数の結果に機密データが含まれる場合、結果への不用意なアクセスを防止するため、STATEMENT DETERMINISTIC または DETERMINISTIC_UDF_SCOPE QAQQINI オプションの使用を検討するか、関数 NOT DETERMINISTIC を作成してください。詳細については、「データベース パフォーマンスおよび Query 最適化」トピック集を参照してください。

94. DETERMINISTIC_UDF_SCOPE QAQQINI オプションを使用して、GLOBAL DETERMINISTIC 関数にこの同じ動作を使用することができます。詳細については、「データベース パフォーマンスおよび Query 最適化」トピック集を参照してください。

CONTAINS SQL
この関数が、データ・アクセス種別 CONTAINS SQL または NO SQL の SQL ステートメントのみを実行できるように指定します。この関数は、データの読み取りまたは変更を行う SQL ステートメントを実行できません。

MODIFIES SQL DATA
この関数は、どの関数でもサポートされないステートメントを除くすべての SQL ステートメントを実行できます。

RETURNS NULL ON NULL INPUT または CALLED ON NULL INPUT
入力引数のいずれかが実行時にヌルである場合に関数を呼び出すかどうかを指定します。

RETURNS NULL ON INPUT
入力引数のいずれかが NULL である場合に関数を呼び出さないことを指定します。結果は NULL 値です。

CALLED ON NULL INPUT
引数値のいずれかまたは全部がヌルである場合に関数を呼び出すことを指定します。この指定は、ヌル引数値のテストを行うように関数をコーディングする必要があります。関数は NULL または非 NULL 値を戻すことができます。

INHERIT SPECIAL REGISTERS
特殊レジスターの既存の値は、関数に入った後に継承されることを指定します。

STATIC DISPATCH
関数を静的にディスパッチすることを指定します。すべての関数が静的にディスパッチされます。

DISALLOW DEBUG MODE、ALLOW DEBUG MODE、または DISABLE DEBUG MODE
関数を Unified Debugger でデバッグできるように作成するかどうかを示します。DEBUG MODE を指定する場合は、SET OPTION ステートメント内の DBGVIEW オプションを指定してはなりません。

DISALLOW DEBUG MODE
関数は Unified Debugger でデバッグできません。関数の DEBUG MODE 属性が DISALLOW の場合、後で関数を変更してデバッグ・モード属性を変えることができます。

ALLOW DEBUG MODE
関数は Unified Debugger でデバッグすることができます。関数の DEBUG MODE 属性が ALLOW の場合、後で関数を変更してデバッグ・モード属性を変えることができます。

DISABLE DEBUG MODE
関数は Unified Debugger でデバッグできません。関数の DEBUG MODE 属性が DISABLE の場合、後で関数を変更してデバッグ・モード属性を変えることはできません。

関数に FENCED または ALLOW PARALLEL が指定されている場合、DEBUG MODE オプションは無視されます。DISALLOW DEBUG MODE が使用されます。

DEBUG MODE を指定せずに SET OPTION ステートメント内の DBGVIEW オプションを指定した場合、関数を Unified Debugger でデバッグすることはできませんが、システム・デバッグ機能を使用してデバッグできる場合があります。 DEBUG MODE オプションも DBGVIEW オプションも指定しない場合は、CURRENT DEBUG MODE 特殊レジスターでのデバッグ・モードが使用されます。

FENCED または NOT FENCED
データベース・マネージャー環境から分離した環境で SQL 関数を実行するかどうかを指定します。 FENCED がデフォルトです。

FENCED
この関数は別のスレッドで実行されます。

FENCED 関数では、関数の呼び出し間で SQL カーソルをオープン状態のままにすることができません。ただし、あるスレッド内のカーソルは他のスレッド内のカーソルから独立しています。このことは、カーソル名の競合が起きる可能性を低くしています。

NOT FENCED
この関数は、呼び出し元の SQL ステートメントと同じスレッド内で実行できます。

NOT FENCED 関数では、関数の呼び出し間で SQL カーソルをオープン状態のままにすることができます。カーソルをオープン状態のままにしておくことができるため、関数の呼び出し間でカーソル位置も同じに維持されます。ただし、UDF が呼び出し元の SQL ステートメントおよび他の NOT FENCED UDF と同じスレッド内で実行されるためにカーソル名が競合する場合があります。

NOT FENCED 関数は、通常 FENCED 関数よりもパフォーマンスが良好です。

ALLOW PARALLEL または DISALLOW PARALLEL
関数を並列で実行できるかどうかを指定します。

NOT DETERMINISTIC、EXTERNAL ACTION、MODIFIES SQL DATA のいずれか 1 つ以上の文節を指定すると、デフォルトは DISALLOW PARALLEL になります。それ以外の場合は、ALLOW PARALLEL がデフォルトです。

ALLOW PARALLEL
データベース・マネージャーが関数の並列処理を考慮できることを指定します。データベース・マネージャーは、この関数を呼び出す SQL ステートメントまたはこの関数の中で実行する SQL ステートメントで並列処理を使用しなければならない、というわけではありません。

ALLOW PARALLEL の指定に関する注意点については、NOT DETERMINISTIC、EXTERNAL ACTION、MODIFIES SQL DATA の説明を参照してください。

DISALLOW PARALLEL
データベース・マネージャーが関数の並列処理を使用してはならないことを指定します。

CONCURRENT ACCESS RESOLUTION
データベース・マネージャーが更新プロセスの過程にあるデータを待つかどうかを指定します。DEFAULT がデフォルトです。

DEFAULT
この関数に関する並行アクセスの解決方法を明示的に設定しないことを指定します。この関数の呼び出し時に有効だった値が使用されます。

WAIT FOR OUTCOME
データベース・マネージャーが更新プロセスの過程にあるデータのコミットまたはロールバックを待つ、という動作を指定します。

USE CURRENTLY COMMITTED
データベース・マネージャーが更新プロセスの過程にあるデータを検出したときに現時点でのコミット済みバージョンのデータを使用する、という動作を指定します。

読み取りトランザクションと削除/更新トランザクションの間でロックの競合が発生した場合に、この文節の適用対象になるのは、分離レベル CS のスキャンです (CS KEEP LOCKS のスキャンは対象になりません)。

SYSTEM_TIME SENSITIVE
静的および動的 SQL ステートメントの両方でのシステム期間テンポラル表への参照が、CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターの値の影響を受けるかどうかを決定します。YES がデフォルトです。

YES
システム期間テンポラル表への参照は、CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターの値の影響を受けます。

NO システム期間テンポラル表の参照は、CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターの値の影響を受けません。

NOT SECURED または **SECURED**
関数が行アクセス制御と列アクセス制御においてセキュアであると見なされるかどうかを指定します。

NOT SECURED
関数が行アクセス制御と列アクセス制御において非セキュアであると見なされることを指定します。これはデフォルトです。

表でアクティブな列アクセス制御が使用されている場合、関数の呼び出し時に、関数の引数が、列マスクが有効になっている列を参照してはなりません。

SECURED
関数が行アクセス制御と列アクセス制御においてセキュアであると見なされることを指定します。

関数は、行の許可または列マスク内で参照される場合は、セキュアとして定義される必要があります。

WRAPPED *obfuscated-statement-text*
エンコードされた関数定義を指定します。WRAP スカラー関数を使用して CREATE FUNCTION ステートメントをエンコードできます。

SET OPTION-statement
関数を作成するときに使用するオプションを指定します。これらのオプションは、すべてのデフォルト値式にも適用されます。 例えば、デバッグ可能な関数を作成するときは、次のステートメントを含めることができます。

```
SET OPTION DBGVIEW = *SOURCE
```

各オプションのデフォルト値は、作成時に有効だったオプションによって異なります。詳細については、 1696 ページの『SET OPTION』を参照してください。

オプション CNULRQD、CNULIGN、COMPILEOPT、NAMING、SQLCA は、CREATE FUNCTIONステートメントでは使用できません。デフォルト値式を処理するときには、オプション ALWCPYDTA、CONACC、DATFMT、DATSEP、DECFLTRND、DECMPT、DECRESULT、DFTRDBCOL、LANGID、SQLCURRULE、SQLPATH、SRTSEQ、TGTRLS、TIMFMT、および TIMSEP が使用されます。

SQL-routine-body
単一の *SQL-procedure-statement* (複合ステートメントを含む) を指定します。SQL 関数の定義についての詳細は、 1771 ページの『第 8 章 SQL 制御ステートメント』を参照してください。

CONNECT、SET CONNECTION、RELEASE、DISCONNECT、COMMIT、ROLLBACK および SET TRANSACTION ステートメントを実行するプロシージャーへの呼び出しは、関数内では使用できません。

SQL-routine-body が複合ステートメントである場合は、そのステートメントには RETURN ステートメントが少なくとも 1 つは含まれていなければならず、関数の呼び出し時に RETURN ステートメントが 1 つ実行される必要があります。

REPLACE キーワードを指定する ALTER PROCEDURE (SQL)、ALTER FUNCTION (SQL スカラー)、および ALTER FUNCTION (SQL 表) は、*SQL-routine-body* では使用できません。

注

ユーザー定義関数の定義に関する一般考慮事項**:** ユーザー定義関数の定義に関する一般情報については、 1070 ページの『CREATE FUNCTION』を参照してください。

SQL パスと関数解決**:** 関数本体にある関数呼び出しは、CREATE FUNCTION ステートメントの有効な SQL パス (関数作成後も変わらない SQL パス) に基づいて解決されます。

関数の所有権**:** SQL 名が指定されている場合、

- 作成した関数が入れられるスキーマと同じ名前のユーザー・プロファイルが存在する場合、関数の所有者 はそのユーザー・プロファイルです。
- その他の場合は、関数の所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

システム名を指定した場合は、関数の所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

関数の権限**:** SQL 名を使用する場合は、関数は、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成されます。 システム名を使用する場合、関数は、スキーマの作成権限 (CRTAUT) パラメーターによって決められる *PUBLIC に対する権限を使用して作成されます。

関数の所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、その関数に対する権限が与えられます。

REPLACE の規則**:** REPLACE によって関数を再作成する場合は、以下のようになります。

- 既存のコメントまたはラベルは破棄されます。
- 権限を持つユーザーは維持されます。オブジェクト所有者は変更される可能性があります。
- 現在のジャーナル監査は保持されます。

関数が置き換えられ、関数シグニチャーまたは結果データ・タイプが変更される場合、その関数を参照する、任意の関数、マテリアライズ照会表、プロシージャー、トリガー、およびビューからの結果は、予測不能なものになる可能性があります。参照されるオブジェクトはすべて再作成する必要があります。

関数の作成**:** SQL 関数が作成される際、データベース・マネージャーは、組み込み SQL ステートメントと一緒に C ソース・コードが収められる一時ソース・ファイルを作成します。次いで、CRTSRVPGM コマンドを使用して、*SRVPGM オブジェクトが作成されます。サービス・プログラムの作成に使用される SQL オプションは、CREATE FUNCTION ステートメントの実行時に有効なオプションです。サービス・プログラムは、ACTGRP(*CALLER) を使用して作成します。

SQL 関数が作成されると、関数の属性は、作成されたサービス・プログラム・オブジェクトに保管されます。*SRVPGM オブジェクトが保管された後、このシステムまたは別のシステムに復元されると、属性が使用されてカタログが更新されます。

ソース・ファイル・メンバーと *SRVPGM オブジェクトの判別には、特定名が使用されます。特定名が有効なシステム名ならば、その特定名がメンバーやプログラムの名前として使用されます。メンバーは、既に存在している場合、オーバーレイされます。指定されたライブラリー内にプログラムが既に存在している場合は、システム表名の生成に関する規則を使用して固有名が生成されます。特定名が有効なシステム名でない場合は、システム表名の生成に関する規則を使用して固有名が生成されます。

関数の呼び出し**:** SQL 関数が呼び出されると、その関数は呼び出し側プログラムの活動化グループ内で実行します。

選択ステートメント の選択リスト で関数が指定され、その関数が EXTERNAL ACTION または MODIFIES SQL DATA を指定している場合、関数は、戻される

各行に対してだけ呼び出されます。それ以外の場合は、選択されていない行に対して UDF が呼び出されることもあります。

インライン関数: SQL スカラー関数がインライン化された場合、照会の一部として関数を呼び出す代わりに、関数の RETURN ステートメントの式 を照会自体にコピーする (インラインで書き込む) こともできます。そのような関数のことをインライン関数 といいます。以下の場合に、スカラー関数はインライン関数 です。

- SQL 関数はグローバル deterministic です。
- *SQL-routine-body* に RETURN ステートメントだけが入っています。
- 配列タイプの入力パラメーターがありません。
- 結果のデータ・タイプが XML でも配列タイプでもありません。
- 関数で参照されるすべてのオブジェクトが、関数の作成時に存在しています。
- *SQL-routine-body* に、入力パラメーターを参照する共通表式が含まれません。
- *SQL-routine-body* に、入力パラメーターを参照し、前に LATERAL キーワードがないネストされた表の式が含まれません。

インライン関数 は、以下の場合にのみ、照会にコピー (インラインで記述) されます。

- 照会が SQL 照会エンジン (SQE) に適格です。
- 関数がオブジェクトを参照し、関数と照会の権限属性が、次のいずれかの条件に基づいて互換です。
  - 関数がユーザーの権限 (*USER) で実行されるよう定義されています。
  - 照会が所有者の権限 (*OWNER) で実行され、照会の所有者が関数の所有者と同じです。
  - 照会がユーザーの権限 (*USER) で実行され、そのユーザーまたはユーザーのグループ・プロファイルが関数の所有者と同じです。

  注 : 関数が FENCED として定義されている場合、照会は借用権限を使用してはなりません。照会が所有者の権限 (*OWNER) で実行され、関数がユーザーの権限 (*USER) で実行される場合、照会の所有者はこのユーザーまたはユーザーのグループ・プロファイルと同じでなければなりません。

関数がインライン化されると、関数の作成時に指定されたオプションの中で、以下のオプションが無視されます。

- PARALLEL または NOT PARALLEL
- MODIFIES SQL DATA
- コミットメント制御レベル
- CONCURRENT ACCESS RESOLUTION
- ALWCPYDTA
- ATOMIC または NOT ATOMIC

関数がインラインで記述されていて、特殊レジスターへの参照を含んでいる場合、その特殊レジスターの値は、照会内の同じ特殊レジスターへの他の参照と同じになります。

関数から配列を戻す: 戻りタイプが配列タイプの関数は、以下のいずれかのコンテキストでのみ、SQL 関数または SQL プロシージャーのルーチン本体から呼び出すことができます。

- SELECT INTO の選択リスト
- DECLARE CURSOR の選択リスト
- SET ステートメントの右側にあるスカラー副選択の選択リスト

配列タイプを戻す関数が含まれている選択リスト は DISTINCT キーワードを使用できません。この選択リストを含んでいる照会は、UNION、EXCEPT、および INTERSECT を使用できません。

難読化されたステートメント: CREATE FUNCTION ステートメントは、難読化された形式で実行できます。難読化されたステートメントでは、 WRAPPED キーワードの前にある関数名とパラメーターのみが判読可能です。ステートメントの残りは、判読できないが、難読化されたステートメントをサポートするデータベース・サーバーによってデコード可能なように、エンコードされます。難読化されたステートメントは、WRAP スカラー関数を呼び出すことによって生成できます。難読化されたステートメントから関数が作成されるときに指定されるデバッグ・オプションはすべて無視されます。難読化されたステートメントから作成された関数を難読化がサポートされていないリリースにリストアすることはできません。

デフォルト値の設定: 関数のパラメーターがデフォルト値を指定して定義されていれば、その関数の呼び出し時にパラメーターはデフォルト値に設定されますが、それは、対応する引数に値が与えられていない場合か、引数が DEFAULT と指定されている場合に限ります。

従属オブジェクト: SQL ルーチンは、*SQL-routine-body* で参照されるオブジェクトに従属します。従属オブジェクトの名前は、カタログ・ビュー SYSROUTINEDEP に保管されます。 *SQL-routine-body* のオブジェクト参照が完全修飾名である場合、または非修飾名が現行スキーマによって SQL 命名で修飾されている場合、SYSROUTINEDEP 内のオブジェクトのスキーマ名は、指定された名前か現行スキーマの値に設定されます。それ以外の場合は、スキーマ名は、特定のスキーマ名に設定されません。非修飾の関数名、変数名、およびタイプ名のスキーマ名は CURRENT PATH になります。名前を実際のスキーマ名に設定しないと、DROP ステートメントおよび ALTER ステートメントは、ルーチンが変更中または除去中のオブジェクトに従属しているかどうかを判別することができません。

代替構文: 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード VARIANT と NOT VARIANT は、NOT DETERMINISTIC と DETERMINISTIC の同義語として使用することができます。
- キーワード NULL CALL と NOT NULL CALL は、CALLED ON NULL INPUT と RETURNS NULL ON NULL INPUT の同義語として使用できます。
- DETERMINISTIC の同義語として、キーワード IS DETERMINISTIC を使用できます。

例

例 *1:* 既存の組み込み関数 SIN と COS を使用して値のタンジェントを返すスカラー関数を定義します。

```
CREATE FUNCTION TAN
     (X DOUBLE)
  RETURNS DOUBLE
  LANGUAGE SQL
  CONTAINS SQL
  NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN SIN(X)/COS(X)
```

パラメーター名 (X) が関数 TAN の入力パラメーターに指定されていることに注意してください。パラメーター名は関数の本体で使用され、入力パラメーターを参照します。 SIN 関数および COS 関数の呼び出し時に、 TAN ユーザー定義関数の本体でパラメーター X を入力として渡します。

例 *2: mm/dd/yyyy* の後に最大 3 文字のストリングを追加した形式で日付を返すスカラー関数を定義します。

```
CREATE FUNCTION BADPARM
     (INP1 DATE,
      USA  VARCHAR(3))
  RETURNS VARCHAR(20)
  LANGUAGE SQL
  CONTAINS SQL
  NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN CHAR(INP1,USA)CONCAT USA
```

以下のステートメントでこの関数を呼び出すとします。

```
SELECT BADPARM(BIRTHDATE,'ISO')
    FROM EMPLOYEE WHERE EMPNO='000010'
```

結果は「08/24/1933ISO」です。ここで注目したいのは、BADPARM 関数の入力パラメーターとして、パラメーター名 (INP1 と USA) が指定されていることです。確かに USA という名前の入力パラメーターは存在しますが、CHAR 関数のパラメーター・リストに含まれている USA のインスタンスは、USA という名前のパラメーターとしてではなく、組み込みの CHAR 関数のキーワード・パラメーターとして取り込まれます。

CREATE FUNCTION (SQL 表)

CREATE FUNCTION (SQL 表) ステートメントは、現行サーバー上に SQL 表関数を作成します。その関数は単一の結果表を戻します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、あるいは対話式に実行することもできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- スキーマ内に作成する特権。詳しくは、スキーマ内で作成する必要のある権限を参照してください。
- データベース管理者権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- SYSFUNCS カタログ・ビューと SYSPARMS カタログ表の場合
  - 該当の表に対する INSERT 特権、および
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

このステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- 次のシステム権限
  - サービス・プログラム作成 (CRTSRVPGM) コマンドに対する *USE
- データベース管理者権限

特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別された、それぞれの特殊タイプごとに、
  - その特殊タイプに対する USAGE 特権、および
  - 特殊タイプが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

SECURED 属性が指定されるか、または、関数がセキュアで OR REPLACE が指定される場合は、次のとおりです。

- このステートメントの許可 ID には、セキュリティー管理者権限 がなければなりません。 21 ページの『管理権限』を参照してください。

既存の関数に置き換えるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

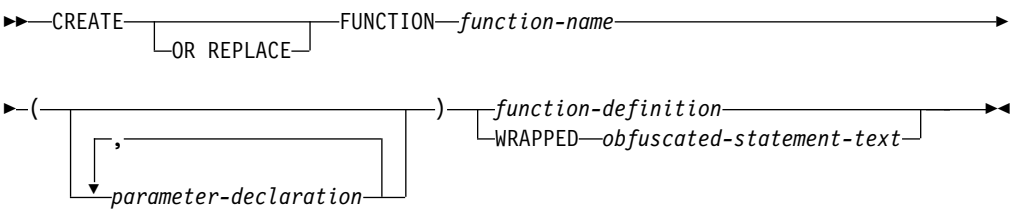
- 次のシステム権限

- この関数に関連したサービス・プログラム・オブジェクトに対する *OBJMGT システム権限
- この関数を削除するために必要な全権限
- SYSFUNCS カタログ・ビューと SYSPARMS カタログ表に対する *READ システム権限

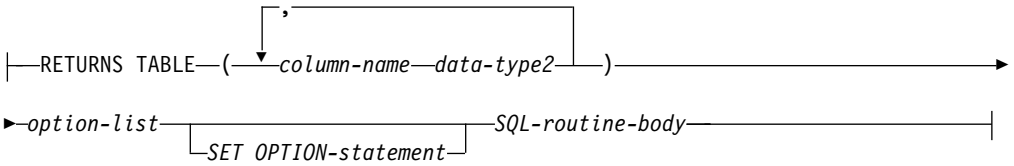
• データベース管理者権限

SQL 特権に対応するシステム権限については、『表またはビューへの権限を検査する際の対応するシステム権限』および 『特殊タイプへの権限を検査する際の対応するシステム権限』を参照してください。

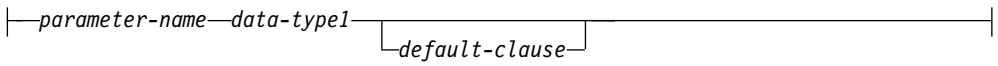
構文



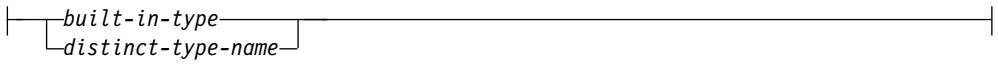
function-definition:



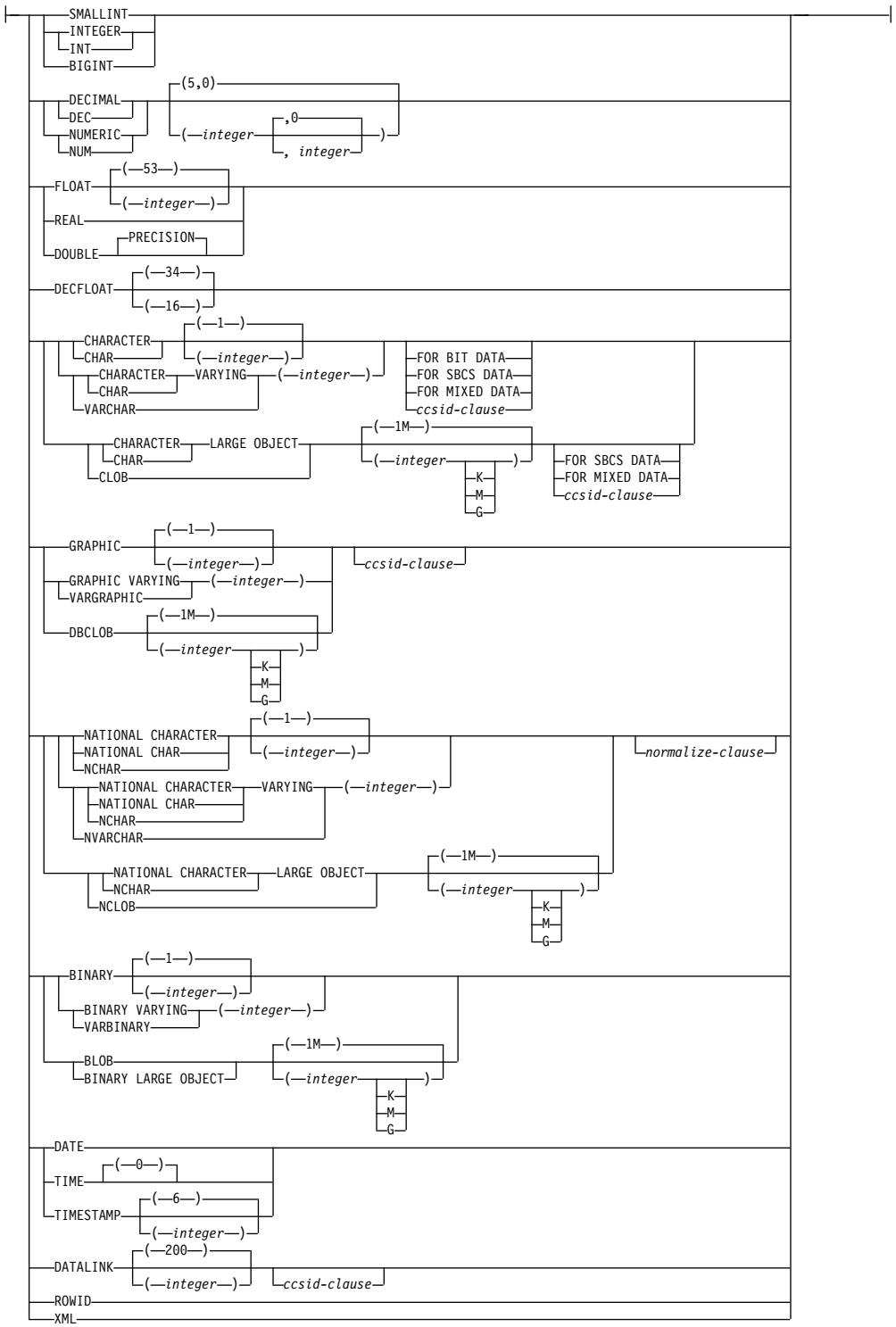
parameter-declaration:



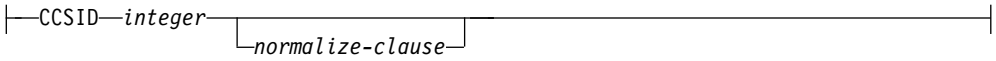
data-type1, data-type2:



built-in-type:



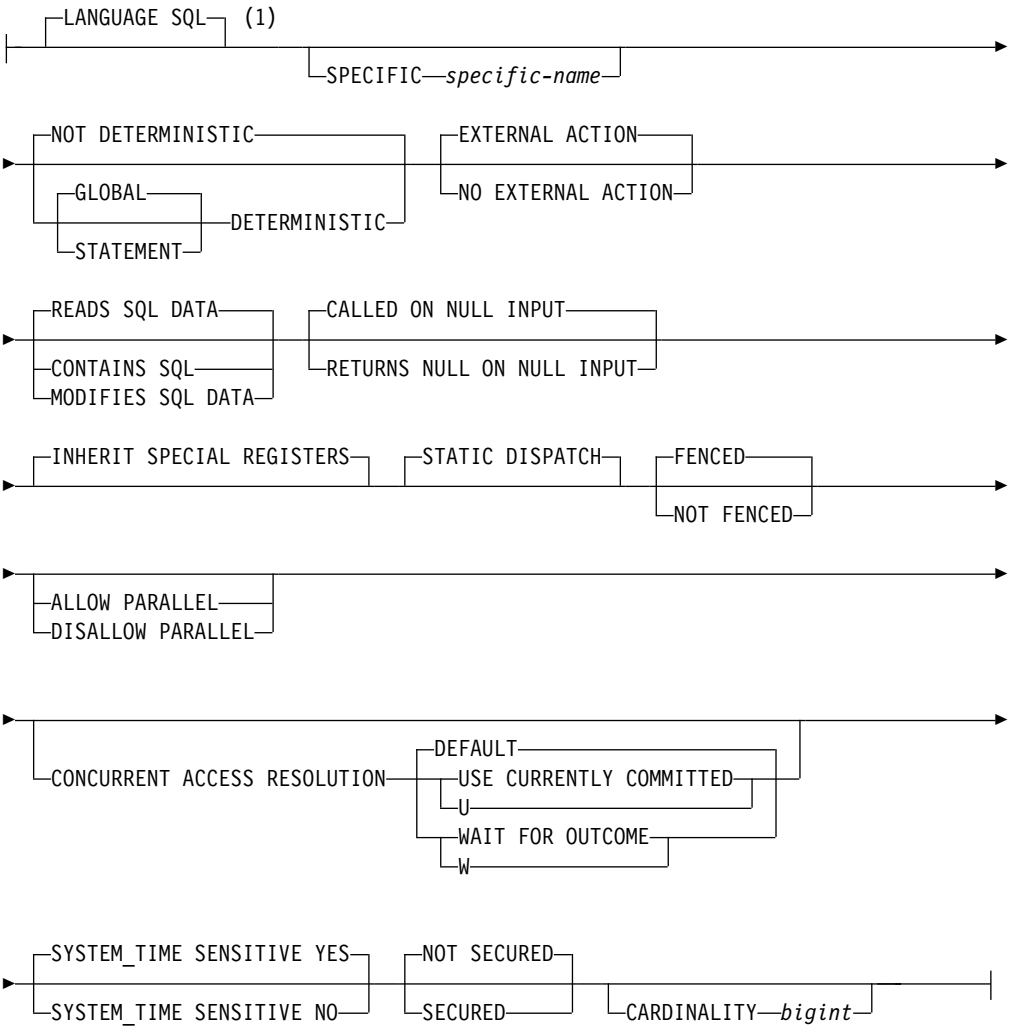
ccsid-clause:



normalize-clause:

default-clause:

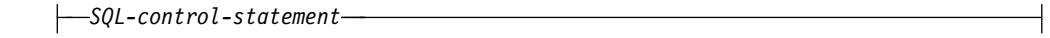
option-list:



注:

1 この文節とこの後の option-list にある文節は、どのような順番で指定してもかまいません。各文節を指定できるのは、それぞれ 1 回だけです。

SQL-routine-body:



説明

OR REPLACE
現行サーバーにこの関数の定義が存在する場合に、その定義を置き換える、という動作を指定します。実際には、カタログで既存の定義を削除してから新しい定義に置き換える、という動作になりますが、例外として、この関数に対して与えられていた特権は影響を受けません。現行サーバーにこの関数の定義が存在しなければ、このオプションは無視されます。既存の関数を置き換えるには、新しい定義の *specific-name* および *function-name* が古い定義の *specific-name* および *function-name* と同じであるか、新しい定義のシグニチャーが古い定義のシグニチャーと一致している必要があります。そうでなければ、新しい関数が作成されます。

function-name
ユーザー定義の関数の名前を指定します。名前、スキーマ名、パラメーターの数、およびそれぞれのパラメーターのデータ・タイプ (データ・タイプの長さ、精度、位取り、または CCSID の属性に関係なく) の組み合わせで、現行サーバー上に存在しているユーザー定義の関数を識別してはなりません。

SQL 命名の場合、関数は、暗黙または明示修飾子で指定されたスキーマ内に作成されます。

システム命名の場合、関数は、修飾子で指定されたスキーマ内に作成されます。修飾子を指定しなかった場合:

- CURRENT SCHEMA 特殊レジスターの値が *LIBL である場合、関数は、現行ライブラリー (*CURLIB) 内に作成されます。
- そうでない場合、関数は現行スキーマ内に作成されます。

通常、それぞれの関数の関数シグニチャーが固有であれば、複数の関数に同じ名前を指定することができます。

一部の関数名は、システムが使用するために予約されています。 詳しくは、1070 ページの『CREATE FUNCTION』の『スキーマおよび関数名の選択』を参照してください。

(*parameter-declaration,...***)**
関数の入力パラメーターの数とそれぞれのパラメーターのデータ・タイプを指定します。各 *parameter-declaration* は、関数の入力パラメーターを指定します。最大 2000 のパラメーターを指定することができます。関数は、ゼロまたはそれ以上の入力パラメーターを持つことができます。関数が受け取ると予期しているパラメーターには、それぞれリスト内に 1 個の項目が必要です。関数のパラメーターはすべて入力パラメーターで、ヌル可能です。詳しくは、1070 ページの『CREATE FUNCTION』の『パラメーターの定義』を参照してください。

parameter-name
パラメーター名を指定します。この名前は、関数の本体に含まれるパラメーターを参照するのに使用されます。この名前は、パラメーター・リスト内の他のパラメーター名 と同じものであってはなりません。

data-type1
入力パラメーターのデータ・タイプを指定します。このデータ・タイプは、組み込みデータ・タイプまたは特殊データ・タイプにすることができます。

built-in-type
組み込みデータ・タイプを指定します。それぞれの組み込みデータ・タイプの詳細については、 1238 ページの『CREATE TABLE』を参照してください。

distinct-type-name
特殊タイプを指定します。パラメーターの長さ、精度、または位取り属性は、特殊タイプのソース・タイプの属性 (CREATE TYPE で指定された属性) と同じになります。特殊タイプの作成についての詳細は、1328 ページの『CREATE TYPE (特殊)』を参照してください。

特殊タイプの名前が修飾されていない場合、データベース・マネージャーは、SQL パス上のスキーマを検索することでそのスキーマ名を解決します。

CCSID が指定されている場合、関数に渡される前に、パラメーターはその CCSID に変換されます。CCSID が指定されていない場合は、CCSID は、関数の呼び出し時点における現行サーバーのデフォルトの CCSID によって決まります。

default-clause
パラメーターのデフォルト値を指定します。デフォルト値は、定数、特殊レジスター、グローバル変数、式、またはキーワード NULL にすることができます。式は、集約関数および列名を含まない、 196 ページの『式』で定義されている任意の式です。デフォルト値が指定されていない場合、パラメーターにデフォルト値がないため、呼び出し時に省略できません。式ストリングの最大長は 64K です。

デフォルトの式は、パラメーターのデータ・タイプに対して割り当ての互換性がなければなりません。

デフォルト式内でリスト中の数値定数を区切る区切り記号として使用するコンマの後には、スペースが 1 つ必要です。

デフォルト式の中で参照されるすべてのオブジェクトは、関数が作成されるときに存在している必要があります。

RETURNS TABLE
関数の出力表を指定します。

パラメーターの数が *N* であるとすると、列の数は 8000-*N* 以下でなければなりません。

column-name
出力表の列の名前を指定します。同じ名前を何度も指定することはできません。

data-type2
出力のデータ・タイプと属性を指定します。

あらゆる組み込みデータ・タイプ (ただし、LONG VARCHAR または LONG VARGRAPHIC は除く) や特殊タイプを指定することができます。関数が呼び出されると、結果は、(記憶域割り当て規則を使用して) これらのデータ・タイプに割り当てられます。

CCSID が指定され、戻りデータの CCSID が異なる CCSID でコード化されている場合、データは指定された CCSID に変換されます。

CCSID が指定されず、関数がビューで参照されない場合、戻りデータの CCSID が異なる CCSID でコード化されている場合には、戻りデータはジョブの CCSID (グラフィック・ストリング戻り値の場合は、ジョブに関連したグラフィック CCSID) に変換されます。変換時に文字が失われるのを防ぐために、関数から戻される文字をすべて表現できる CCSID を明示的に指定することを考慮してください。 これは、データ・タイプがグラフィック・ストリング・データの場合に特に重要です。 この場合、CCSID 1200 または 13488 (ユニコード・グラフィック・ストリング・データ) を使用することを考慮してください。

CCSID が指定されず、関数がビューで参照される場合、戻りデータは、関連するビュー列の CCSID に変換されます。変換時に文字が失われるのを防ぐために、関数から戻される文字をすべて表現できる CCSID を明示的に指定することを考慮してください。 これは、データ・タイプがグラフィック・ストリング・データの場合に特に重要です。 この場合、CCSID 1200 または 13488 (ユニコード・グラフィック・ストリング・データ) を使用することを考慮してください。

LANGUAGE SQL
これは SQL 関数であることを指定します。

SPECIFIC *specific-name*
関数の固有名を指定します。特定名の詳細については、 1070 ページの『CREATE FUNCTION』の『関数に特定の名前を指定する』を参照してください。

GLOBAL DETERMINISTIC または **STATEMENT DETERMINISTIC** または **NOT DETERMINISTIC**
関数が同じ入力引数を指定して呼び出されるたびに、その関数が同じ結果を戻すかどうかを指定します。デフォルトは NOT DETERMINISTIC です。

NOT DETERMINISTIC
関数が同じ入力引数を指定して呼び出されるたびに、その関数が同じ結果を戻さない場合があることを指定します。関数は、結果に影響を与えるいくつかの状態値によって変わってきます。データベース・マネージャーは、SQL ステートメントの最適化時にこの情報を使用します。表関数の結果表に影響するような方法で、特殊レジスター、非決定的な関数、シーケンスのいずれかを参照する表関数は、非決定的な表関数の例です。

GLOBAL DETERMINISTIC
関数が同じ入力引数を指定して呼び出されるたびに、その関数が常に同じ結果表を戻すかを指定します。データベース・マネージャーは、SQL ステートメントの最適化時にこの情報を使用します。照会オプティマイザーは、グローバル deterministic な関数の結果をキャッシュに入れるよう選択することができます。

STATEMENT DETERMINISTIC
関数が同じ入力引数を使用して呼び出されるたびに、同じ結果を戻さない可能性があるが、単一 SQL ステートメント内での関数の複数の呼び出しは、deterministic と見なされることを指定します。照会オプティマイザーは、ステートメント deterministic な関数の結果をキャッシュに入れません。[95]

EXTERNAL ACTION または NO EXTERNAL ACTION
関数が、データベース・マネージャーが管理しないオブジェクトの状態を変更するアクションを行うかどうかを指定します。外部アクションの例には、メッセージの送信やストリーム・ファイルへのレコードの書き込みなどがあります。デフォルトは EXTERNAL ACTION です。

EXTERNAL ACTION
関数が、データベース・マネージャーが管理しないオブジェクトの状態を変更するアクションを行うことができるかを指定します。したがって、関数は、それぞれの連続関数呼び出しで呼び出す必要があります。 EXTERNAL ACTION は、この関数に、外部アクションを持つ他の関数に対する参照が含まれている場合に、指定してください。

NO EXTERNAL ACTION
関数は外部アクションを行いません。この関数は、連続した各関数呼び出しごとに呼び出す必要はありません。

NO EXTERNAL ACTION 関数は、連続した各関数呼び出しごとに呼び出されない場合があるので、EXTERNAL ACTION 関数よりもパフォーマンスが向上する可能性があります。

CONTAINS SQL、READS SQL DATA、または MODIFIES SQL DATA
関数が実行できる SQL ステートメントおよびネストされたルーチンの種別を指定します。データベース・マネージャーは、関数と、関数からローカルで呼び出すすべてのルーチンが発行する SQL ステートメントが、この指定と整合しているかどうかを検査します。ネストされたリモート・ルーチンが呼び出された場合、検査は実行されません。各ステートメントの分類については、 1855 ページの『付録 B. SQL ステートメントの特性』を参照してください。デフォルトは、READS SQL DATA です。このオプションは、すべてのパラメーター・デフォルト式に適用されます。

READS SQL DATA
この関数が、データ・アクセス種別 READS SQL DATA、CONTAINS SQL、または NO SQL のステートメントを実行できるように指定します。この関数は、データの変更を行う SQL ステートメントは実行できません。

CONTAINS SQL
この関数が、データ・アクセス種別 CONTAINS SQL または NO SQL の SQL ステートメントのみを実行できるように指定します。この関数は、データの読み取りまたは変更を行う SQL ステートメントを実行できません。

MODIFIES SQL DATA
この関数は、どの関数でもサポートされないステートメントを除くすべての SQL ステートメントを実行できます。

---

95. DETERMINISTIC_UDF_SCOPE QAQQINI オプションを使用して、GLOBAL DETERMINISTIC 関数にこの同じ動作を使用することができます。詳細については、「データベース パフォーマンスおよび Query 最適化」トピック集を参照してください。

RETURNS NULL ON NULL INPUT または CALLED ON NULL INPUT
入力引数のいずれかが実行時にヌルである場合に関数を呼び出すかどうかを指定します。

RETURNS NULL ON NULL INPUT
入力引数のいずれかがヌルである場合に関数を呼び出さないことを指定します。結果は、表に行がない、空表になります。 RETURNS NULL ON NULL INPUT がデフォルトです。

CALLED ON NULL INPUT
引数値のいずれかがヌルである場合に関数を呼び出すことを指定します。この指定は、ヌル引数値のテストを行うように関数をコーディングする必要があります。関数は、その論理によっては、空の表を戻す場合があります。

INHERIT SPECIAL REGISTERS
特殊レジスターの既存の値は、関数に入った後に継承されることを指定します。

STATIC DISPATCH
関数を静的にディスパッチすることを指定します。すべての関数が静的にディスパッチされます。

FENCED または NOT FENCED
データベース・マネージャー環境から分離した環境で SQL 関数を実行するかどうかを指定します。 FENCED がデフォルトです。

FENCED
この関数は別のスレッドで実行されます。

FENCED 関数では、関数の呼び出し間で SQL カーソルをオープン状態のままにすることができません。ただし、あるスレッド内のカーソルは他のスレッド内のカーソルから独立しています。このことは、カーソル名の競合が起きる可能性を低くしています。

NOT FENCED
この関数は、呼び出し元の SQL ステートメントと同じスレッド内で実行できます。

NOT FENCED 関数では、関数の呼び出し間で SQL カーソルをオープン状態のままにすることができます。カーソルをオープン状態のままにしておくことができるため、関数の呼び出し間でカーソル位置も同じに維持されます。ただし、UDF が呼び出し元の SQL ステートメントおよび他の NOT FENCED UDF と同じスレッド内で実行されるためにカーソル名が競合する場合があります。

NOT FENCED 関数は、通常 FENCED 関数よりもパフォーマンスが良好です。

ALLOW PARALLEL または DISALLOW PARALLEL
関数を並列で実行できるかどうかを指定します。

NOT DETERMINISTIC、EXTERNAL ACTION、MODIFIES SQL DATA の 1 つ以上の文節が指定された場合、またはこれがパイプライン表関数である場合、デフォルトは DISALLOW PARALLEL になります。それ以外の場合は、ALLOW PARALLEL がデフォルトです。

ALLOW PARALLEL
データベース・マネージャーが関数の並列処理を考慮できることを指定します。データベース・マネージャーは、この関数を呼び出す SQL ステートメントまたはこの関数の中で実行する SQL ステートメントで並列処理を使用しなければならない、というわけではありません。

ALLOW PARALLEL の指定に関する注意点については、NOT DETERMINISTIC、EXTERNAL ACTION、MODIFIES SQL DATA の説明を参照してください。

DISALLOW PARALLEL
データベース・マネージャーが関数の並列処理を使用してはならないことを指定します。

CONCURRENT ACCESS RESOLUTION
データベース・マネージャーが更新プロセスの過程にあるデータを待つかどうかを指定します。DEFAULT がデフォルトです。

DEFAULT
この関数に関する並行アクセスの解決方法を明示的に設定しないことを指定します。この関数の呼び出し時に有効だった値が使用されます。

WAIT FOR OUTCOME
データベース・マネージャーが更新プロセスの過程にあるデータのコミットまたはロールバックを待つ、という動作を指定します。

USE CURRENTLY COMMITTED
データベース・マネージャーが更新プロセスの過程にあるデータを検出したときに現時点でのコミット済みバージョンのデータを使用する、という動作を指定します。

読み取りトランザクションと削除/更新トランザクションの間でロックの競合が発生した場合に、この文節の適用対象になるのは、分離レベル CS のスキャンです (CS KEEP LOCKS のスキャンは対象になりません)。

SYSTEM_TIME SENSITIVE
静的および動的 SQL ステートメントの両方でのシステム期間テンポラル表への参照が、CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターの値の影響を受けるかどうかを決定します。YES がデフォルトです。

YES
システム期間テンポラル表への参照は、CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターの値の影響を受けます。

NO システム期間テンポラル表の参照は、CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターの値の影響を受けません。

NOT SECURED または **SECURED**
関数が行アクセス制御と列アクセス制御においてセキュアであると見なされるかどうかを指定します。

NOT SECURED
関数が行アクセス制御と列アクセス制御において非セキュアであると見なされることを指定します。これはデフォルトです。

表でアクティブな列アクセス制御が使用されている場合、関数の呼び出し時に、関数の引数が、列マスクが有効になっている列を参照してはなりません。

SECURED
関数が行アクセス制御と列アクセス制御においてセキュアであると見なされることを指定します。

関数は、行の許可または列マスク内で参照される場合は、セキュアとして定義される必要があります。

WRAPPED *obfuscated-statement-text*
エンコードされた関数定義を指定します。WRAP スカラー関数を使用して CREATE FUNCTION ステートメントをエンコードできます。

CARDINALITY *bigint*
このオプションの文節は、最適化を目的として、この関数が戻すものとして予期される行数の見積もりを指定します。整数の有効な値の範囲は、0 から 9 223 372 036 854 775 807 です。

表関数に対して CARDINALITY 文節を指定しない場合は、データベース・マネージャーは、デフォルトとして特定の有限値を想定します。

呼び出されるたびに行を戻して表終了状態を戻すことのない表関数は、無限カーディナリティーを持ちます。データを戻す前に結果としての表終了状態を必要とする照会がこのような関数を呼び出すと、その照会は中断しない限り戻りません。

SET OPTION-statement
関数を作成するときに使用するオプションを指定します。これらのオプションは、すべてのデフォルト値式にも適用されます。 例えば、デバッグ可能な関数を作成するときは、次のステートメントを含めることができます。

```
SET OPTION DBGVIEW = *SOURCE
```

各オプションのデフォルト値は、作成時に有効だったオプションによって異なります。詳しくは、 1696 ページの『SET OPTION』を参照してください。

オプション CNULRQD、CNULIGN、COMPILEOPT、NAMING、SQLCA は、CREATE FUNCTIONステートメントでは使用できません。
CLOSQLCSR(*ENDACTGRP) は、常に SQL 表関数に対して使用されます。デフォルト値式を処理するときには、オプション
ALWCPYDTA、CONACC、DATFMT、DATSEP、DECFLTRND、DECMPT、DECRESULT、DFTRDBCOL、LANGID、SQLCURRULE、
SQLPATH、SRTSEQ、TGTRLS、TIMFMT、および TIMSEP が使用されます。

SQL-routine-body
複合ステートメントも含め、単一の SQL ステートメントを指定します。 SQL 関数の定義についての詳細は、 1771 ページの『第 8 章 SQL 制御ステートメント』を参照してください。

非パイプライン表関数は、厳密に 1 つの RETURN ステートメントを含む必要があります。パイプライン表関数は、少なくとも 1 つの RETURN ステートメントを含む必要があります。関数を呼び出すときに、RETURN ステートメントを実行する必要があります。

CONNECT、SET CONNECTION、RELEASE、DISCONNECT、COMMIT、ROLLBACK および SET TRANSACTION ステートメントを実行するプロシージャーへの呼び出しは、関数内では使用できません。

REPLACE キーワードを指定する ALTER PROCEDURE (SQL)、ALTER FUNCTION (SQL スカラー)、および ALTER FUNCTION (SQL 表) は、*SQL-routine-body* では使用できません。

注

ユーザー定義関数の定義に関する一般考慮事項**:** ユーザー定義関数の定義に関する一般情報については、 1771 ページの『第 8 章 SQL 制御ステートメント』を参照してください。

関数の所有権**:** SQL 名が指定されている場合、

- 作成した関数が入れられるスキーマと同じ名前のユーザー・プロファイルが存在する場合、関数の所有者 はそのユーザー・プロファイルです。
- その他の場合は、関数の所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

システム名を指定した場合は、関数の所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

関数の権限**:** SQL 名を使用する場合は、関数は、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成されます。 システム名を使用する場合、関数は、スキーマの作成権限 (CRTAUT) パラメーターによって決められる *PUBLIC に対する権限を使用して作成されます。

関数の所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、その関数に対する権限が与えられます。

パイプライン関数と非パイプライン関数**:** SQL 表関数には、2 つのタイプがあります。 *SQL-routine-body* 内に PIPE ステートメントを含まない表関数は、非パイプライン表関数です。これには、表を戻す 1 個の RETURN ステートメントが含まれます。パイプライン表関数は、戻り値のない 1 個以上の RETURN ステートメントと 0 個以上 PIPE ステートメントが *SQL-routine-body* 内に含まれている表関数です。これは、一度に 1 行を表に戻します。この 2 つのタイプの表関数は、まったく同じ方法で呼び出されます。

PIPE ステートメントは、表関数からの結果行を戻します。次の行を取得するために、制御は *SQL-routine-body* 内の PIPE ステートメントの次のステートメントに戻ります。

REPLACE の規則**:** REPLACE によって関数を再作成する場合は、以下のようになります。

- 既存のコメントまたはラベルは破棄されます。
- 権限を持つユーザーは維持されます。オブジェクト所有者は変更される可能性があります。

- 現在のジャーナル監査は保持されます。

関数が置き換えられ、関数シグニチャーまたは結果データ・タイプが変更される場合、その関数を参照する、任意の関数、マテリアライズ照会表、プロシージャー、トリガー、およびビューからの結果は、予測不能なものになる可能性があります。参照されるオブジェクトはすべて再作成する必要があります。

関数の作成**:** SQL 関数が作成される際、データベース・マネージャーは、組み込み SQL ステートメントと一緒に C ソース・コードが収められる一時ソース・ファイルを作成します。次いで、CRTSRVPGM コマンドを使用して、*SRVPGM オブジェクトが作成されます。サービス・プログラムの作成に使用される SQL オプションは、CREATE FUNCTION ステートメントの実行時に有効なオプションです。サービス・プログラムは、ACTGRP(*CALLER) を使用して作成します。

SQL 関数が作成されると、関数の属性は、作成されたサービス・プログラム・オブジェクトに保管されます。*SRVPGM オブジェクトが保管された後、このシステムまたは別のシステムに復元されると、属性が使用されてカタログが更新されます。

ソース・ファイル・メンバーと *SRVPGM オブジェクトの判別には、特定名が使用されます。特定名が有効なシステム名ならば、その特定名がメンバーやプログラムの名前として使用されます。メンバーは、既に存在している場合、オーバーレイされます。指定されたライブラリー内にプログラムが既に存在している場合は、システム表名の生成に関する規則を使用して固有名が生成されます。特定名が有効なシステム名でない場合は、システム表名の生成に関する規則を使用して固有名が生成されます。

関数の呼び出し**:** SQL 関数が呼び出されると、その関数は呼び出し側プログラムの活動化グループ内で実行します。

インライン関数**:** SQL 表関数がインライン化された場合、照会の一部として関数を呼び出す代わりに、関数の RETURN ステートメントの全選択 を照会自体にコピーする (インラインで書き込む) こともできます。そのような関数のことをインライン関数 といいます。以下の場合に、表関数はインライン関数 です。

- SQL 関数が NO EXTERNAL ACTION として定義されています。
- *SQL-routine-body* に RETURN ステートメントだけが入っています。
- 結果表に、XML データ・タイプの列がありません。
- 関数で参照されるすべてのオブジェクトが、関数の作成時に存在しています。
- *SQL-routine-body* に、入力パラメーターを参照する共通表式が含まれません。

インライン関数 は、以下の場合にのみ、照会にコピー (インラインで記述) されます。

- 照会が SQL 照会エンジン (SQE) に適格です。
- 関数が別のサーバーの表を参照しません。
- 関数がオブジェクトを参照し、関数と照会の権限属性が、次のいずれかの条件に基づいて互換です。
  - 関数がユーザーの権限 (*USER) で実行されるよう定義されています。
  - 照会が所有者の権限 (*OWNER) で実行され、照会の所有者が関数の所有者と同じです。

– 照会がユーザーの権限 (*USER) で実行され、そのユーザーまたはユーザーのグループ・プロファイルが関数の所有者と同じです。

注 : 関数が FENCED として定義されている場合、照会は借用権限を使用してはなりません。照会が所有者の権限 (*OWNER) で実行され、関数がユーザーの権限 (*USER) で実行される場合、照会の所有者はこのユーザーまたはユーザーのグループ・プロファイルと同じでなければなりません。

関数がインライン化されると、関数の作成時に指定されたオプションの中で、以下のオプションが無視されます。

- PARALLEL または NOT PARALLEL
- MODIFIES SQL DATA
- コミットメント制御レベル
- CONCURRENT ACCESS RESOLUTION
- ALWCPYDTA
- ATOMIC または NOT ATOMIC

関数がインラインで記述されていて、特殊レジスターへの参照を含んでいる場合、その特殊レジスターの値は、照会内の同じ特殊レジスターへの他の参照と同じになります。

難読化されたステートメント**:** CREATE FUNCTION ステートメントは、難読化された形式で実行できます。難読化されたステートメントでは、 WRAPPED キーワードの前にある関数名とパラメーターのみが判読可能です。ステートメントの残りは、判読できないが、難読化されたステートメントをサポートするデータベース・サーバーによってデコード可能なように、エンコードされます。難読化されたステートメントは、WRAP スカラー関数を呼び出すことによって生成できます。難読化されたステートメントから関数が作成されるときに指定されるデバッグ・オプションはすべて無視されます。難読化されたステートメントから作成された関数を難読化がサポートされていないリリースにリストアすることはできません。

従属オブジェクト**:** SQL ルーチンは、*SQL-routine-body* で参照されるオブジェクトに従属します。従属オブジェクトの名前は、カタログ・ビュー SYSROUTINEDEP に保管されます。 *SQL-routine-body* のオブジェクト参照が完全修飾名である場合、または非修飾名が現行スキーマによって SQL 命名で修飾されている場合、SYSROUTINEDEP 内のオブジェクトのスキーマ名は、指定された名前か現行スキーマの値に設定されます。それ以外の場合は、スキーマ名は、特定のスキーマ名に設定されません。非修飾の関数名、変数名、およびタイプ名のスキーマ名は CURRENT PATH になります。名前を実際のスキーマ名に設定しないと、DROP ステートメントおよび ALTER ステートメントは、ルーチンが変更中または除去中のオブジェクトに従属しているかどうかを判別することができません。

代替構文**:** 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード VARIANT と NOT VARIANT は、NOT DETERMINISTIC と DETERMINISTIC の同義語として使用することができます。

- キーワード NULL CALL と NOT NULL CALL は、CALLED ON NULL INPUT と RETURNS NULL ON NULL INPUT の同義語として使用できます。
- DETERMINISTIC の同義語として、キーワード IS DETERMINISTIC を使用できます。

例

指定した部門番号に該当する社員を戻す表関数を定義します。

```
CREATE FUNCTION DEPTEMPLOYEES (DEPTNO CHAR(3))
  RETURNS TABLE (EMPNO CHAR(6),
                 LASTNAME VARCHAR(15),
                 FIRSTNAME VARCHAR(12))
  LANGUAGE SQL
  READS SQL DATA
  NO EXTERNAL ACTION
  DETERMINISTIC
  DISALLOW PARALLEL
  RETURN
      SELECT EMPNO,LASTNAME,FIRSTNME
        FROM EMPLOYEE
        WHERE EMPLOYEE.WORKDEPT =DEPTEMPLOYEES.DEPTNO
```

CREATE INDEX

CREATE INDEX ステートメントは、現行サーバーで表の索引を作成します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- スキーマ内に作成する特権。詳しくは、スキーマ内で作成する必要のある権限を参照してください。
- データベース管理者権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 次のシステム権限
  - 論理ファイル作成 (CRTLF) コマンドに対する *USE 権限。
  - データ・ディクショナリーに対する *CHANGE 権限。ただし、索引が作成されるライブラリーが、データ・ディクショナリーを持つ SQL スキーマの場合。
- データベース管理者権限

このステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- 参照される表の場合
  - 該当の表に対する INDEX 特権。
  - 表が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

SQL 名が指定され、該当の表が作成されるライブラリーの名前と同じ名前のユーザー・プロファイルが存在し、しかもその名前がステートメントの権限 ID と異なっている場合、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

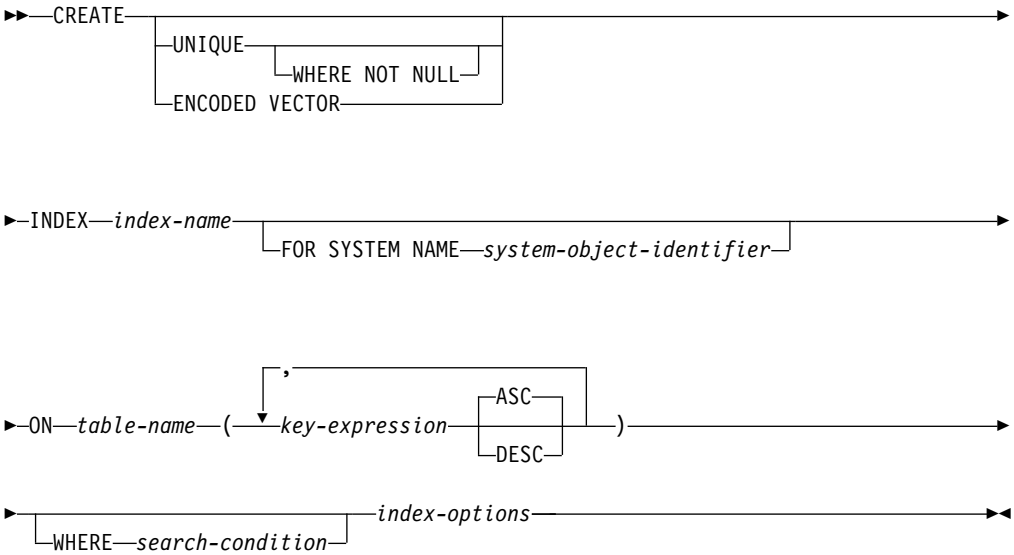
- その名前を持つユーザー・プロファイルに対する *ADD システム権限
- データベース管理者権限

特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

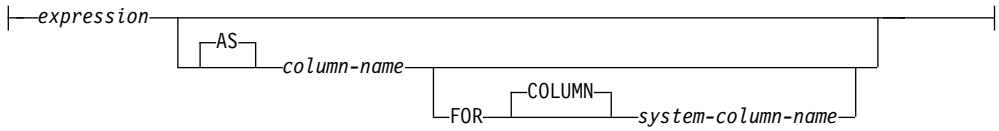
- ステートメント内で識別された、それぞれの特殊タイプごとに、
  - その特殊タイプに対する USAGE 特権、および
  - 特殊タイプが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

SQL 特権に対応するシステム権限については、『表またはビューへの権限を検査する際の対応するシステム権限』を参照してください。

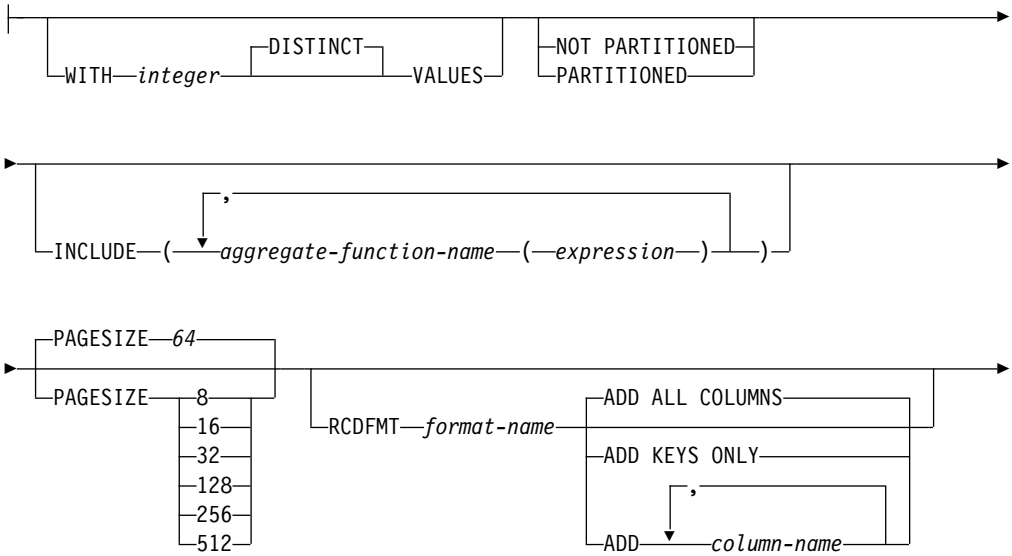
構文

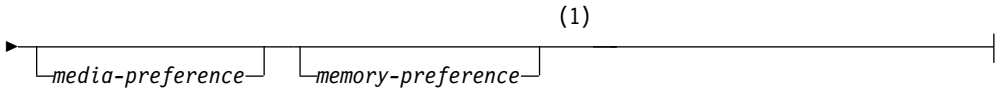


key-expression:



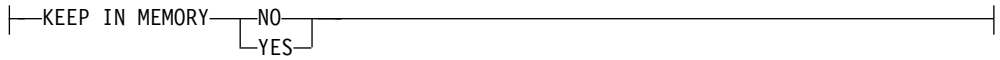
index-options:





media-preference:

memory-preference:



注:

1 索引オプション はどのような順序で指定しても構いません。

説明

UNIQUE
表に、同一の索引キーの値を持つ行が複数入るのを防止します。 UNIQUE を使用する場合、列のすべての NULL 値が等しいものと見なされます。例えば、NULL 値を入れることができる単一の列をキーにすると、その列には、NULL 値が 1 つしか入らなくなります。この制約が適用されるのは、表の行を更新するときと、新しい行を挿入するときです。

CREATE INDEX ステートメントの実行時にも、この制約が検査されます。重複するキーの値を持つ行が既に表に入っている場合、索引は作成されません。

UNIQUE WHERE NOT NULL
表に、同一の索引キーの値を持つ行が複数入るのを防止します。列が NULL 値の場合には、それらすべては等しいとは見なされません。 1 つの列に NULL 値が複数入ることが許可されます。その他の点では、UNIQUE と同じです。

ENCODED VECTOR
これを指定すると、結果の索引は、コード化ベクトル索引 (EVI) になります。

コード化ベクトル索引を使用して、行の順序を保証することはできません。これは、データベース・マネージャーが照会のパフォーマンスを向上させる場合に使用します。詳細については、「データベース パフォーマンスおよび Query 最適化」トピック集を参照してください。

index-name
索引の名前を指定します。暗黙的または明示的修飾子も含め、この名前は、現行サーバーに既に存在している索引、表、ビュー、別名、またはファイルと同じ名前にすることはできません。

SQL 名が指定されている場合、索引は、暗黙的または明示的修飾子で指定しているスキーマ内に作成されます。

システム名が指定されている場合、索引名は、修飾子で指定しているスキーマ内に作成されます。修飾されていない場合、索引名は、その索引の作成に使用した表と同じスキーマ内に作成されます。

索引名が有効なシステム名でなく、FOR SYSTEM NAME 節が使用されていない場合、Db2 for i はシステム名を生成します。名前の生成に関する規則については、 1298 ページの『表名の生成の規則』を参照してください。

FOR SYSTEM NAME *system-object-identifier*
索引のシステム・オブジェクト *ID* を示します。システム・オブジェクト *ID* は、現行サーバーに既に存在する表、ビュー、別名、または索引と同一であってはなりません。システム・オブジェクト *ID* は、非修飾システム ID でなければなりません。

システム・オブジェクト *ID* が指定される場合、索引名 は有効なシステム・オブジェクト名であってはなりません。

ON *table-name*
その索引を作成したい表を指定します。この表名 は、現行サーバーに存在している基本表 (ビューではなく) を識別するものでなくてはなりません。

表がパーティション化された表である場合、単一パーティションを識別する別名を指定できます。その場合、作成される索引は、指定されたパーティション上だけで作成されます。

key-expression
索引キーを構成する列または式を識別します。

索引に定義するキーの数は 120 を超えてはならず、それぞれのバイト長の合計は 32766-*n* を超えてはなりません。ここで、*n* は NULL が許されると指定されたキーの数です。

expression
式 に列名 のみが入る場合、この列名は、表の列を識別する非修飾名でなければなりません。式 は列参照を含んでいなければなりません。 次のような場合は、同一の列名 を複数回指定することはできません。

- WHERE 文節、INCLUDE 文節、または RCDFMT 文節が指定されている、
- 式が索引キーの一部として定義されている、または
- 列が AS 文節を使用して名前変更されている。

expression が列名ではない場合、フィールド・プロシージャーを含む列を *expression* が参照することはできません。

列名 では、LOB 列、XML 列、または DATALINK 列あるいは LOB 列、XML 列、または DATALINK 列に基づく特殊タイプを識別することはできません。*expression* が列名ではない場合、中間結果式および最終結果式のデータ・タイプを DATALINK、LOB、および XML にすることはできません。次のいずれも含めることはできません。

- 副照会
- 集約関数
- 変数
- グローバル変数
- パラメーター・マーカー
- 特殊レジスター
- シーケンス参照

- OLAP の指定
- ROW CHANGE 式
- REGEXP_LIKE 述部
- 特殊タイプの作成に伴って暗黙に生成された関数以外のユーザー定義関数
- deterministic でない関数
- 以下の組み込みスカラー関数:

ATAN2	DLURLSCHEME	JSON_TO_BSON	ROUND_TIMESTAMP
BSON_TO_JSON	DLURLSERVER	JSON_VALUE	RPAD
CARDINALITY	DLVALUE	LOCATE_IN_STRING	SCORE
CONTAINS	ENCRYPT_AES	LPAD	SOUNDEX
CURDATE	ENCRYPT_RC2	MAX_CARDINALITY	TABLE_NAME
CURTIME	ENCRYPT_TDES	MONTHNAME	TABLE_SCHEMA
DATAPARTITIONNAME	GENERATE_UNIQUE	MONTHS_BETWEEN	TIMESTAMP_FORMAT
DATAPARTITIONNUM	GETHINT	NEXT_DAY	TIMESTAMPDIFF
DAYNAME	HASH	NOW	TRUNC_TIMESTAMP
DBPARTITIONNAME	HASH_MD5	OVERLAY	VARCHAR_FORMAT
DECRYPT_BINARY	HASH_SHA1	RAISE_ERROR	VERIFY_GROUP_FOR_USER
DECRYPT_BIT	HASH_SHA256	RAND	WEEK_ISO
DECRYPT_CHAR	HASH_SHA512	REGEXP_COUNT	WRAP
DECRYPT_DB	IDENTITY_VAL_LOCAL	REGEXP_INSTR	XMLPARSE
DIFFERENCE	INSERT	REGEXP_REPLACE	XMLVALIDATE
DLURLCOMPLETE [1]	JSON_ARRAY	REGEXP_SUBSTR	XSLTRANSFORM
DLURLPATH	JSON_OBJECT	REPEAT	
DLURLPATHONLY	JSON_QUERY	REPLACE	

[1] FILE LINK CONTROL と READ PERMISSION DB の属性が指定されたデータ・リンクの場合。

column-name
索引の列の名前を指定します。索引の複数の列または索引のシステム列名に同一の名前を使用しないでください。

式 が列名でない場合および名前が指定されていない場合、名前は索引キー列に生成されます。名前は SQLIXxxxxx になります。ここで *xxxxx* は、索引に固有の列名を作成する番号です。

FOR COLUMN システム列名
列の IBM i 名を指定します。索引の複数の列または索引の列名に、同一の名前を使用しないでください。

システム列名が指定されず、また列名が有効なシステム列名でない場合には、システム列名が生成されます。名前は SQLIXxxxxx になります。ここで *xxxxx* は、索引に固有の列名を作成する番号です。

ASC
索引項目を列値の昇順で保持することを指定します。 ASC がデフォルトです。

DESC
索引項目を列値の降順で保持することを指定します。

順序付けは、 113 ページの『割り当ておよび比較』で説明した比較規則にしたがって行われます。 NULL 値は、他のどの値よりも上位に置かれます。

WHERE *search-condition*
索引に含まれる行に対して適用する条件を指定します。検索条件 に、副照会を

指定する述部を入れることはできません。キー式 の制約事項としてリストされた項目のいずれも含めることはできません。

WITH *integer* DISTINCT VALUES
特殊キー値の見積数を指定します。 この文節は、あらゆるタイプの索引に対して指定することができます。

コード化ベクトル索引の場合は、これを使用し、それぞれの特殊キー値に割り当てられるコードの初期サイズを決定します。1、2、および 4 バイトコードのみが使用されます。 INCLUDE 文節が指定されない場合、デフォルト値は 255 (1 バイトコード) です。それ以外の場合は、4 バイトコードが使用されます。索引の作成または再作成中に、特殊値の数が、コードのサイズでサポートされる最大数を超えた場合、コードのサイズが増加されます。

非コード化ベクトル索引の場合、この文節は無視されます。

PARTITIONED
表に定義された各データ・パーティションごとに、指定した列を使用して索引パーティションを作成することを指定します。表名では、パーティション化された表を識別する必要があります。索引が固有である場合、その索引の列はデータ・パーティション・キーの列と同じであるか、そのスーパーセットでなければなりません。索引が固有ではなく、表がパーティション化されている場合、PARTITIONED がデフォルトになります。

NOT PARTITIONED
表に定義されたデータ・パーティションのすべてにわたる単一の索引を作成することを指定します。表名では、パーティション化された表を識別する必要があります。索引が固有であり、表がパーティション化されている場合、NOT PARTITIONED がデフォルトになります。パーティション化されていない表の索引も、デフォルトではパーティション化されません。

エンコードされたベクトル索引が指定された場合、NOT PARTITIONED は使用できません。

PAGESIZE
索引に使用する論理ページを K バイト単位で指定します。一般に、論理ページのサイズが大きい索引の方が、照会処理におけるスキャン時の効率がよくなります。単純索引探索や個別キー検索の場合は、一般に論理ページのサイズが小さい索引の方が効率がよくなります。

PAGESIZE のデフォルト値はキーの長さで決まり、最小値は 64 です。

エンコードされたベクトル索引が指定された場合、PAGESIZE は使用できません。

INCLUDE
索引に含まれる集約関数式を指定します。こうした集約関数式を指定すると、索引で、照会の集約結果を直接戻すことができるようになります。INCLUDE を使用できるのは、エンコード化ベクトル索引に対してのみです。

aggregate-function-name (expression)
この集約関数名は、組み込み関数
AVG、COUNT、COUNT_BIG、SUM、STDDEV、STDDEV_SAMP、
VARIANCE、または VARIANCE_SAMP のいずれかでなければなりませ

ん。DISTINCT キーワードを指定してはなりません。この集約関数の *expression* 引数には、*key-expression* の制約事項としてリストされた項目のいずれも含めることはできません。

RCDFMT *format-name*
索引の IBM i レコード・フォーマット名を指定する非修飾名です。*format-name* は、システム ID です。

INCLUDE キーワードを指定する場合、 RCDFMT は使用できません。

ADD ALL COLUMNS
表名 のすべての隠されていない列が索引のフォーマットに追加されるように指定します。すべての列は、表名 のフォーマットに表示されるものと同一の順で定義され、索引キーとして定義された式の前に置かれます。

ADD KEYS ONLY
索引キー列として指定された列のみが、索引のフォーマットに追加されるように指定します。表名 からの他の列は追加されません。

ADD *column-name*
リストされた列が索引のフォーマットに追加されるように指定します。索引キー列が先頭になり、その後に追加の列が続きます。

media-preference

索引の優先ストレージ・メディアを指定します。

UNIT ANY
どのストレージ・メディアも優先しません。索引のストレージは、使用可能な任意のストレージ・メディアから割り振られます。

UNIT SSD
ソリッド・ステート・ディスク・ストレージ・メディアを優先します。索引のストレージは、使用可能な場合にはソリッド・ステート・ディスク・ストレージ・メディアから割り振ることができます。

memory-preference

KEEP IN MEMORY
索引のデータが照会で使用されるときに、データを主記憶域プールに入れるかどうかを指定します。

NO データは主記憶域プールに入れられません。

YES
データは主記憶域プールに入れられます。

注

ステートメントの影響 **:** CREATE INDEX は、索引の記述を作成します。指定した表に既にデータが入っていれば、CREATE INDEX によって、そのデータに関する索引項目が作成されます。表にまだデータが入っていない場合、索引項目は、表にデータが挿入されたときに作成されます。

照合シーケンス**:** SBCS または混合データを含む列に対して作成される索引は、このステートメントの実行時点で有効な照合シーケンスに従って作成されます。照合シ

ーケンスが *HEX 以外の場合は、SBCS データまたは混合データのキーは、該当の照合シーケンスに基づいてキーが重み付けされた値です。

索引の属性 : 索引はキー付き論理ファイルとして作成されます。索引が作成される場合、ファイル待ち時間とレコード待ち時間の属性は、論理ファイル作成 (CRTLF) コマンドの WAITFILE キーワードと WAITRCD キーワード上に指定されたデフォルト値に設定されます。

日時の結果列に使用される日時形式は ISO です。

分散表に対して作成される索引は、この表が配布されるサーバーのすべてで作成されます。分散表についての詳細は、Db2 マルチシステム (Multisystem) を参照してください。

索引の所有権: SQL 名が指定されている場合、

- 作成した索引が入れられるスキーマと同じ名前のユーザー・プロファイルが存在する場合、索引の所有者 はそのユーザー・プロファイルです。
- その他の場合は、索引の所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

システム名を指定した場合は、索引の所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

索引の権限 : SQL 名を使用する場合は、索引は、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成されます。システム名を使用する場合、索引は、スキーマの作成権限 (CRTAUT) パラメーターによって決められる *PUBLIC に対する権限を使用して作成されます。

索引の所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、その索引に対する権限が与えられます。

レコード・フォーマットの共有: キー列を式として定義する索引、あるいは RCDFMT、WHERE、INCLUDE、または AS 節を指定する索引は、表名 のフォーマットを共有しません。それ以外の場合、索引は表名 のフォーマットを共有し、そのフォーマット名 は索引のシステム・オブジェクト名 と同じになります。

例

例 *1:* PROJECT 表に UNIQUE_NAM という名前の索引を作成します。この索引の目的は、表内に、同じプロジェクト名 (PROJNAME) が重複して入ることがないようにすることです。索引項目は昇順になります。

```
CREATE UNIQUE INDEX UNIQUE_NAM
  ON PROJECT(PROJNAME)
```

例 *2:* EMPLOYEE 表に JOB_BY_DPT という名前の索引を作成します。索引項目は、各部門 (WORKDEPT) ごとにジョブ・タイトル (JOB) にしたがって昇順に並べます。

```
CREATE INDEX JOB_BY_DPT
  ON EMPLOYEE (WORKDEPT, JOB)
```

例 *3:* DEPARTMENT 表に DEPT_TYPE という名前の索引を作成します。索引項目は、部門のタイプに従って昇順に並べます。部門のタイプは、部門番号 (DEPTNO) の 2 文字目および 3 文字目で判別されます。

```
CREATE INDEX DEPT_TYPE
 ON DEPARTMENT (SUBSTR(DEPTNO,2,2))
```

CREATE MASK

CREATE MASK ステートメントは、現行サーバーで列アクセス制御の列マスクを作成します。列マスクは、指定された列について戻される値を指定します。

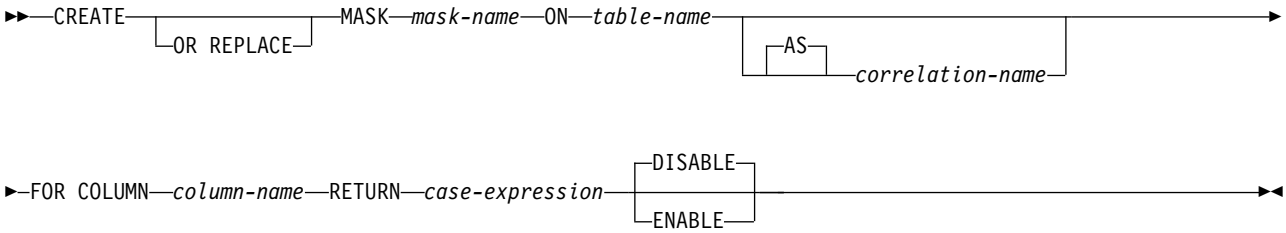
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、あるいは対話式に実行することもできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの許可 ID には、セキュリティー管理者権限 がなければなりません。 21 ページの『管理権限』を参照してください。

構文



説明

OR REPLACE
列マスクの定義が現行のサーバー上に存在している場合に、その列マスクの定義を置換するために指定します。既存の定義は、新しい定義がカタログ内で置換される前に、効率的にドロップされます。

mask-name
列アクセス制御の列マスクの名前を指定します。暗黙的または明示的修飾子も含め、この名前は、現行サーバーに既に存在している列マスクまたは行の許可と同じ名前にすることはできません。*mask-name* は QIBM で始まってはなりません。

SQL 名が指定されている場合、マスクは、暗黙的または明示的修飾子で指定しているスキーマ内に作成されます。

システム名が指定されている場合、マスクは、修飾子で指定しているスキーマ内に作成されます。修飾されておらず、デフォルトのスキーマがない場合、マスクは *table-name* と同じスキーマに作成されます。

mask-name のスキーマ名は、*table-name* のスキーマ名と同じでなければなりません。

table-name
列マスクを作成する対象の表を指定します。この名前は、現行サーバーに存在する表を示すものでなければなりません。宣言済み一時表、 QTEMP 内の表、分散表、ビュー、論理ファイル、メンバー別名、読み取りトリガーのあるファイル、またはカタログ表を示すものであってはなりません。

correlation-name
表を指定するために *case-expression* 内で使用できる相関名を指定します。

FOR COLUMN *column-name*
マスクを適用する列を指定します。表の列を示す非修飾名でなければなりません。この列にはマスクが既に存在していてはなりません。

RETURN *case-expression*
列について戻す値を決定するために評価される CASE 式を指定します。CASE 式の結果が、行の列値の位置に返されます。CASE 式の結果のデータ・タイプ、長さ、ヌル属性、および CCSID は、列のデータ・タイプと互換でなければなりません。列が NULL 値を許容しない場合、CASE 式の結果は NULL 値にはなりません。 データ・タイプの互換性に関する詳細については、 113 ページの『割り当ておよび比較』を参照してください。 *column-name* のデータ・タイプがユーザー定義のデータ・タイプである場合、 CASE 式の結果のデータ・タイプは同じユーザー定義のタイプでなければなりません。 CASE 式で参照されるオブジェクトは、現行サーバーに存在していなければなりません。CASE 式では以下を参照してはなりません。

- 列マスクが定義されている表
- 宣言済みグローバル一時表
- 変数 (ホスト変数、SQL 変数、SQL パラメーター、またはトリガー遷移変数)
- パラメーター・マーカー
- NOT SECURED として定義されているユーザー定義関数
- 非 deterministic 関数[96]または外部アクションを持つ関数
- 列マスクを定義する表を参照する RRN、RID、HASHED_VALUE、DATAPARTITIONNAME、DATAPARTITIONNUM、DBPARTITIONNAME、または DBPARTITIONNUM 関数
- OLAP 指定
- ROW CHANGE 式
- シーケンス参照
- SELECT 文節内の * または *name*.*
- QTEMP 内の表
- メンバー別名
- 分散表
- 読み取りトリガーのあるファイル
- 複数フォーマット論理ファイル
- リモート・オブジェクト
- 上記のいずれかを含んでいるビュー

ENABLE または **DISABLE**
列アクセス制御のための列マスクを使用可能または使用不可にすることを指定します。

96. STATEMENT DETERMINISTIC 関数は許可されていますが、お勧めしません。

DISABLE
列アクセス制御のための列マスクを使用不可に設定することを指定します。表の列アクセス制御が活動化されているかどうかに関係なく、列マスクは無効なままになります。これはデフォルトです。

ENABLE
列アクセス制御で列マスクを有効にするように指定します。この表の列アクセス制御が現在活動化されていない場合、列マスクは、表の列アクセス制御が活動化されると有効になります。表の列アクセス制御が現在活動化されている場合、列マスクは直ちに有効になります。

注

前提条件**:** マスクを作成するためには、IBM Advanced Data Security がインストールされている必要があります。

照会への列マスクの影響**:** 有効な列マスクを適用しても、ステートメント内の他の節 (WHERE、GROUP BY、HAVING、SELECT DISTINCT、ORDER BY など) の操作が妨げられることはありません。最終結果表に戻される行は同じですが、結果行の値が列マスクによってマスクされている可能性があります。このため、sort-key 式が含まれている ORDER BY 文節にマスクされた列も指定されている場合、順序は列の元の値に基づいており、最終結果表のマスクされた値にはこの順序が反映されていないことがあります。同様に、マスクされた値には、SELECT DISTINCT ステートメントにより適用される固有性が反映されない可能性があります。マスクされた列が式に組み込まれている場合、式評価が可能になる前に列マスクが列に適用されるので、式の結果が異なることがあります。例えば、列 SSN の列マスクにより、集約関数 COUNT(DISTINCT SSN) の結果が変更されることがあります。これは、DISTINCT 操作がマスクされた値に対して実行されるためです。ただし、照会の中の式が、列マスク定義で列値をマスクするために使用された式と同じ場合、式の結果は変化しない可能性があります。例えば、照会の中の式が 'XXX-XX-' || SUBSTR(SSN, 8, 4) であり、これと同じ式が列マスク定義にも指定されているとします。この特定の例では、照会から式を削除して、同じ式が 2 回評価されるのを回避できます。

列マスクの定義と **SQL** との矛盾**:** 列マスクは、独立したオブジェクトとして作成され、使用される可能性のあるコンテキストが作成時にすべて認識されるわけではありません。最終結果表の列の値をマスクするため、列マスクの定義が Db2 による照会にマージされます。列マスクの定義がステートメントのコンテキストに取り込まれるときに、この定義がステートメントの特定の SQL セマンティクスと矛盾することがあります。したがって、特定の状況では、ステートメントと列マスクの適用方法の組み合わせによってはエラーが戻されることがあります。この状況が発生した場合は、ステートメントを変更するか、または列マスクの削除または異なる定義での再作成を行う必要があります。

列マスクとヌル列**:** 列が NULL 可能でない場合、その列の列マスクの定義は、ほとんどの場合、列の NULL 値を考慮しません。外部結合の NULL 埋め込み表の場合、表の列アクセス制御がアクティブにされた後で、最終結果表の列値が NULL になる可能性があります。列マスクにより NULL 値をマスクできることを確実にするため、Db2 では、外部結合の NULL 埋め込み表の場合には列マスク定義に 1 番目の WHEN 文節として WHEN target-column IS NULL THEN NULL が追加

されます。これにより、NULL 値が常に NULL 値として強制的にマスクされます。NULL 可能列の場合、これにより、NULL 値を他の値としてマスクすることができなくなります。例 4 に、この追加される WHEN 文節を示します。

データ変更 **SQL** ステートメントでの列マスク値**:** INSERT、UPDATE、および MERGE では、新しい行の値を導出するときに列が参照されている場合に、その列に対して有効な列マスクがあれば、マスクされた値が新しい値を導出するために使用されます。オブジェクト表でも列アクセス制御がアクティブになっている場合、新しい値を導出するために適用される列マスクは、定数や式ではなく、列自体を戻す必要があります。列マスクを列に適用した結果が列自体にならない場合、新しい値を挿入または更新に使用できず、エラーが戻されます。新しい値を得るために列マスクを適用するときに使用されるルールは、照会の最終結果表の同じルールに従います。挿入可能性と更新可能性に影響する列マスクの使用方法については、データ変更ステートメントを参照してください。

列マスクとトリガー遷移変数**:** OLD ROW および OLD TABLE 遷移変数の値に、マスクされた値が含まれることはありません。

SET 遷移変数 割り当てステートメントは、マスクされたデータを変数に割り当てることができます。列に違反チェック制約が存在しない場合、マスクされたデータが表の列で挿入または更新され、エラーは発行されません。

列アクセス制御がアクティブになる前に作成された列マスク**:** CREATE MASK ステートメントは独立したステートメントであり、これを使用して、表の列アクセス制御がアクティブになる前に列アクセス制御マスクを作成することができます。この場合の唯一の要件は、表と列がマスクの作成前に存在していることです。1 つの表に対して複数の列マスクを作成できますが、1 つの列に設定できるマスクは 1 つのみです。マスクの定義は Db2 カタログに保管されます。マスクの作成対象の表への従属関係と、定義で参照されるその他のオブジェクトへの従属関係が記録されます。列マスクの作成時には、列アクセス制御に対してその列マスクが有効または無効であるかどうかを指定できます。有効に設定されている列マスクは、ACTIVATE COLUMN ACCESS CONTROL 文節が指定された ALTER TABLE ステートメントを使用して表の列アクセス制御がアクティブになった後で、初めて有効になります。表の列アクセス制御がアクティブになっても、無効な列マスクは引き続き無効です。ALTER MASK ステートメントを使用して、ENABLE と DISABLE を切り替えることができます。表の列アクセス制御がアクティブになった後、表がデータ操作ステートメントで参照されるとき、その表に対して作成されているすべての有効な列マスクが Db2 によって暗黙的に適用され、照会の最終結果表で参照される列について返される値がマスクされるか、データ変更ステートメントで使用する新しい値が決定されます。

行アクセス制御がアクティブになった後で作成された列マスク**:** 有効に設定された列マスクは、コミットされるとすぐに有効になります。したがって、データ操作ステートメントで表が参照されている場合、有効な列マスクはすべて、Db2 によりそのステートメントに暗黙に適用されます。表の列アクセス制御がアクティブになっても、無効な列マスクは引き続き無効です。

列アクセス制御または行アクセス制御が適用されている表が列マスク定義で参照される場合、カスケード効果はない**:** 列マスク定義が、行アクセス制御または列アクセス制御が現在適用されている表と列を参照することがあります。列マスクの作成対

象の表がデータ操作ステートメントで参照される場合、このような表と列のアクセス制御は無視されます。

例

例 *1:* 表 EMPLOYEE の列アクセス制御がアクティブになった後で、給与管理部門の Paul は、従業員番号が 123456 の従業員の社会保障番号を確認できます。管理職の Mary は、社会保障番号の最後の 4 文字のみを確認できます。Peter は給与管理部門に属しておらず、また管理者でもないため、ソーシャル・セキュリティー番号を確認できません。

```
CREATE MASK SSN_MASK ON EMPLOYEE
   FOR COLUMN SSN RETURN
      CASE
         WHEN (VERIFY_GROUP_FOR_USER(SESSION_USER,'PAYROLL') = 1)
            THEN SSN
         WHEN (VERIFY_GROUP_FOR_USER(SESSION_USER,'MGR') = 1)
            THEN 'XXX-XX-' || SUBSTR(SSN,8,4)
         ELSE NULL
      END
   ENABLE;

COMMIT;

ALTER TABLE EMPLOYEE
   ACTIVATE COLUMN ACCESS CONTROL;

COMMIT;

SELECT SSN FROM EMPLOYEE
   WHERE EMPNO = 123456;
```

例 *2:* SELECT ステートメントで、列マスク SSN_MASK に使用されている式と同じ式に列 SSN が組み込まれています。表 EMPLOYEE に対して列アクセス制御をアクティブにした後に、列マスク SSN_MASK が SELECT ステートメントの列 SSN に適用されます。この特定の式の場合、SELECT ステートメントの結果は、すべてのユーザーに対して列アクセス制御をアクティブにする前と同じ結果になります。ユーザーは SELECT ステートメントの式を列 SSN に置き換えることによって、同じ式が 2 回評価されるのを回避できます。

```
CREATE MASK SSN_MASK ON EMPLOYEE
   FOR COLUMN SSN RETURN
      CASE
         WHEN (1 = 1)
             THEN 'XXX-XX-' || SUBSTR(SSN,8,4)
         ELSE NULL
      END
   ENABLE;

COMMIT;

ALTER TABLE EMPLOYEE
   ACTIVATE COLUMN ACCESS CONTROL;

COMMIT;

SELECT 'XXX-XX-' || SUBSTR(SSN,8,4) FROM EMPLOYEE
   WHERE EMPNO = 123456;
```

例 *3* EMPNO が 123456 の従業員の 5 月のボーナスは $8000、給与は $80000 です。管理者がこの従業員の給与を取り出すときに、NULL 値ではなく給与額が取り

出されます。これは、列マスク SALARY_MASK が、列マスク BONUS_MASK が定義されている列 BONUS を参照する場合、カスケード効果が発生しないためです。

```
CREATE MASK SALARY_MASK ON EMPLOYEE
   FOR COLUMN SALARY RETURN
      CASE
         WHEN (BONUS < 10000)
            THEN SALARY
         ELSE NULL
      END
   ENABLE;

COMMIT;

CREATE MASK BONUS_MASK ON EMPLOYEE
   FOR COLUMN BONUS RETURN
      CASE
         WHEN (BONUS > 5000)
            THEN NULL
         ELSE BONUS
      END
   ENABLE;

COMMIT;

ALTER TABLE EMPLOYEE
   ACTIVATE COLUMN ACCESS CONTROL;

COMMIT;

SELECT SALARY FROM EMPLOYEE
   WHERE EMPNO = 123456;
```

例 *4* この例では、Db2 は、列マスク定義に 1 番目の WHEN 文節として「WHEN target-column IS NULL THEN NULL」を追加し、次に列マスク定義をステートメントにマージします。

```
CREATE TABLE EMPLOYEE (EMPID INT,
                       DEPTID CHAR(8),
                       SALARY DEC(9,2) NOT NULL,
                       BONUS DEC(9,2));

CREATE MASK SALARY_MASK ON EMPLOYEE
   FOR COLUMN SALARY RETURN
      CASE
         WHEN SALARY < 10000
            THEN CAST(SALARY*2 AS DEC(9,2))
         ELSE COALESCE(CAST(SALARY/2 AS DEC(9,2)), BONUS)
      END
   ENABLE;

COMMIT;

CREATE MASK BONUS_MASK ON EMPLOYEE
   FOR COLUMN BONUS RETURN
      CASE
         WHEN BONUS > 1000
            THEN BONUS
         ELSE NULL
      END
   ENABLE;

COMMIT;

ALTER TABLE EMPLOYEE
```

```
   ACTIVATE COLUMN ACCESS CONTROL;

COMMIT;

SELECT SALARY FROM DEPT
   LEFT JOIN EMPLOYEE ON DEPTNO = DEPTID;

/* When SALARY_MASK is effectively merged into the above statement,
* 'WHEN SALARY IS NULL THEN NULL' is added by Db2 as the
* first WHEN clause, as follows:
*/

SELECT CASE WHEN SALARY IS NULL THEN NULL
            WHEN SALARY < 10000 THEN CAST(SALARY*2 AS DEC(9,2))
            ELSE COALESCE(CAST(SALARY/2 AS DEC(9,2)), BONUS)
       END SALARY
  FROM DEPT
    LEFT JOIN EMPLOYEE ON DEPTNO = DEPTID;
```

CREATE PERMISSION

CREATE PERMISSION ステートメントは、現行サーバーで行アクセス制御のための行の許可を作成します。これは *search-condition* の結果に基づいて表内の使用可能な行を決定します。

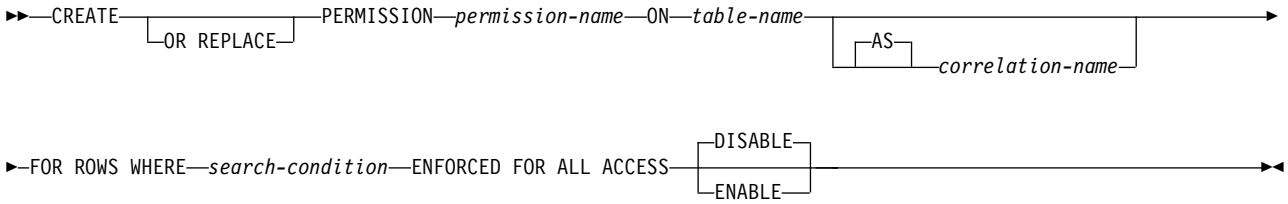
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、あるいは対話式に実行することもできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの許可 ID には、セキュリティー管理者権限 がなければなりません。 21 ページの『管理権限』を参照してください。

構文



説明

OR REPLACE
行権限の定義が現行サーバー上に存在している場合に、その定義を置換するために指定します。既存の定義は、新しい定義がカタログ内で置換される前に、効率的にドロップされます。

permission-name
行アクセス制御の行の許可の名前を指定します。暗黙的または明示的修飾子も含め、この名前は、現行サーバーに既に存在している列マスクまたは行の許可と同じ名前にすることはできません。*permission-name* は QIBM で始まってはなりません。

SQL 名が指定されている場合、許可は、暗黙的または明示的修飾子で指定しているスキーマ内に作成されます。

システム名が指定されている場合、許可は、修飾子で指定しているスキーマ内に作成されます。修飾されておらず、デフォルトのスキーマがない場合、許可は *table-name* と同じスキーマに作成されます。

permission-name のスキーマ名は、*table-name* のスキーマ名と同じでなければなりません。

table-name
列権限を作成する対象の表を指定します。この名前は、現行サーバーに存在する表を示すものでなければなりません。宣言済み一時表、 QTEMP 内の表、分散表、ビュー、論理ファイル、メンバー別名、読み取りトリガーのあるファイル、またはカタログ表を示すものであってはなりません。

correlation-name
表を指定するために *search-condition* 内で使用できる相関名を指定します。

FOR ROWS WHERE
行の許可が作成されることを指定します。行の許可は、表の行にアクセスできる検索条件を指定します。

search-condition
表の行について真、偽、または不明となる条件を指定します。

search-condition は、WHERE 文節の検索条件に使用される規則に従います。また、以下のものは、いずれも参照してはなりません。

- 行の許可が定義される表
- 宣言済みグローバル一時表
- 変数 (ホスト変数、SQL 変数、SQL パラメーター、またはトリガー遷移変数)
- パラメーター・マーカー
- NOT SECURED として定義されているユーザー定義関数
- 非 deterministic 関数[97]または外部アクションを持つ関数
- 行の許可を定義する表を参照する RRN、RID、HASHED_VALUE、DATAPARTITIONNAME、DATAPARTITIONNUM、DBPARTITIONNAME、または DBPARTITIONNUM 関数
- OLAP 指定
- ROW CHANGE 式
- シーケンス参照
- SELECT 文節内の * または *name*.*
- QTEMP 内の表
- メンバー別名
- 分散表
- 読み取りトリガーのあるファイル
- 複数フォーマット論理ファイル
- リモート・オブジェクト
- 上記のいずれかを含んでいるビュー

ENFORCED FOR ALL ACCESS
行の許可がこの表のすべての参照に適用されることを指定します。この表で行アクセス制御がアクティブになっている場合、データ操作ステートメントでこの表が参照されていると、DB2® は表のアクセスを制御するため、行の許可を暗黙に適用します。SELECT などのフェッチ操作で表が参照される場合、行の許可の適用によって、フェッチ操作を要求したユーザーが取得できる行のセットが決まります。 INSERT などのデータ変更操作で表が参照される場合、行の許可の適用によって、データ変更操作を要求したユーザーが、すべての変更対象行を挿入または更新できるかどうかが決まります。

97. STATEMENT DETERMINISTIC 関数は許可されていますが、お勧めしません。

ENABLE または DISABLE
行アクセス制御で行の許可を有効または無効に初期設定することを指定します。

DISABLE
行アクセス制御で行の許可を無効にすることを指定します。表の行アクセス制御が活動化されているかどうかに関係なく、行の許可は無効なままになります。これはデフォルトです。

ENABLE
行アクセス制御で行の許可を有効にすることを指定します。この表の行アクセス制御が現在アクティブになっていない場合、行の許可は、表の行アクセス制御がアクティブになった時点で有効になります。表の行アクセス制御が現在アクティブになっている場合、行の許可は直ちに有効になります。

注

前提条件: 許可を作成するためには、IBM Advanced Data Security がインストールされている必要があります。

行の許可の適用方法および特定のステートメントへの影響: 行アクセス制御を活動化する方法と、行の許可がどのように適用されるのかについては、 ACTIVATE ROW ACCESS CONTROL 節が指定された ALTER TABLE ステートメントの説明を参照してください。行の許可の適用によるフェッチ操作への影響については、副選択の説明を参照してください。行の許可の適用によるデータ変更操作への影響については、データ変更ステートメントを参照してください。

表の行アクセス制御がアクティブになる前に作成する行権限: CREATE PERMISSION ステートメントは独立したステートメントであり、これを使用して、表の行アクセス制御がアクティブになる前に行権限を作成できます。唯一の要件は、権限の作成前に表と列が存在していることです。 1 つの表に対して、複数の行権限を作成できます。

行権限の定義は Db2 カタログに格納されます。権限の作成対象となる表への従属関係と、当該の定義で参照されるその他のオブジェクトへの従属関係が記録されます。行アクセス制御のために、行権限を使用可能または使用不可なものとして作成できます。使用可能に設定された行権限は、ACTIVATE ROW ACCESS CONTROL 節が指定された ALTER TABLE ステートメントを使用して表の行アクセス制御をアクティブにすると、初めて有効になります。表の行アクセス制御がアクティブになっても、使用不可に設定されている行権限は引き続き無効です。ALTER PERMISSION ステートメントを使用して、ENABLE と DISABLE を切り替えることができます。

表の行アクセス制御をアクティブにした後に、データ操作ステートメントで表が参照されるときに、その表に定義されたすべての使用可能な行権限が Db2 によって暗黙的に適用されて、その表へのアクセスが制御されます。

表の行アクセス制御がアクティブになった後に作成する行権限: 使用可能に設定された行権限は、コミットされるとすぐに有効になります。したがって、データ操作ステートメントで表が参照されている場合、有効な行の許可はすべて、そのステートメントに暗黙に適用されます。表の行アクセス制御がアクティブになっても、無効な行の許可はすべて引き続き無効です。

行アクセス制御または列アクセス制御が施行されている表を行権限定義内で参照している場合、カスケード効果はない**:** 行権限定義で、行アクセス制御または列アクセス制御が現在施行されている表と列を参照することがあります。行の許可の作成対象の表がデータ操作ステートメントで参照される場合、このような表のアクセス制御は無視されます。

許可の **DECRESULT** オプション**:** 許可の DECRESULT オプションは、常に最大精度 63、最大位取り 63、および最小除算位取り 0 を使用します。

例

例 1: 行の許可 SALARY_ROW_ACCESS 内のセキュアなユーザー定義関数 ACCOUNTING_UDF は、列 SALARY の機密データを処理します。表 EMPLOYEE の行アクセス制御がアクティブになった後に、会計士の Paul が、年収 100,000 ドルの従業員 (EMPNO 123456) の給与を取り出します。ユーザー定義関数 ACCOUNTING_UDF の出力値に応じて、Paul に対してこの行が表示される場合と表示されない場合があります。

```
CREATE PERMISSION SALARY_ROW_ACCESS ON EMPLOYEE
   FOR ROWS WHERE VERIFY_GROUP_FOR_USER(SESSION_USER,'MGR','ACCOUNTING') = 1
                AND
                ACCOUNTING_UDF(SALARY) < 120000
   ENFORCED FOR ALL ACCESS
   ENABLE;

COMMIT;

ALTER TABLE EMPLOYEE
ACTIVATE ROW ACCESS CONTROL;

COMMIT;

SELECT SALARY FROM EMPLOYEE
   WHERE EMPNO = 123456;
```

例 2: 銀行の現金出納係は、所属支店の顧客のデータのみにアクセスできます。すべての現金出納係に対し、2 次許可 ID として TELLER が割り当てられています。顧客サービス担当者は、銀行の全顧客のデータにアクセスできます。すべての顧客サービス担当者に対し、2 次許可 ID として CSR が割り当てられています。セキュリティー管理者権限を持つ者が定義したこれらのアクセス規則に従って、行員のグループごとに行の許可が作成されます。表 CUSTOMER の行アクセス制御がアクティブになった後、各グループがアクセスできる行セットを制御するため、SELECT ステートメントで両方の行の許可の検索条件がこのステートメントにマージされ、論理 OR 演算子で結合されます。

```
CREATE PERMISSION TELLER_ROW_ACCESS ON CUSTOMER
   FOR ROWS WHERE VERIFY_GROUP_FOR_USER(SESSION_USER,'TELLER') = 1
                AND
                BRANCH = (SELECT HOME_BRANCH FROM INTERNAL_INFO
                           WHERE EMP_ID = SESSION_USER)
   ENFORCED FOR ALL ACCESS
   ENABLE;

COMMIT;

CREATE PERMISSION CSR_ROW_ACCESS ON CUSTOMER
   FOR ROWS WHERE VERIFY_GROUP_FOR_USER(SESSION_USER,'CSR') = 1
   ENFORCED FOR ALL ACCESS
   ENABLE;
```

```
COMMIT;

ALTER TABLE CUSTOMER
   ACTIVATE ROW ACCESS CONTROL;

COMMIT;

SELECT * FROM CUSTOMER;
```

CREATE PROCEDURE

CREATE PROCEDURE ステートメントは、現行サーバーでプロシージャーを定義します。

定義できるプロシージャーのタイプは以下のとおりです。

- 外部

  このタイプのプロシージャー・プログラムまたはサービス・プログラムは、C、COBOL、Java などのプログラミング言語で書かれます。この外部実行ファイルは、現行サーバーで定義されているプロシージャーにより、プロシージャーの各種属性に基づいて参照されます。 1189 ページの『CREATE PROCEDURE (外部)』を参照してください。

- SQL

  このタイプのプロシージャーは SQL のみで書かれます。プロシージャー本体は、プロシージャーの各種属性と一緒に現行サーバーで定義されます。 1208 ページの『CREATE PROCEDURE (SQL)』を参照してください。

注

パラメーターのデータ・タイプを選択する:Db2 for i 以外のプラットフォーム間でプロシージャーを移植可能にするために、以下のデータ・タイプは使用しないでください。これらは、異なるプラットフォームでは表記が異なる場合があります。

- FLOAT。この代わりに、DOUBLE や REAL を使用すること。
- NUMERIC。この代わりに、DECIMAL を使用すること。

パラメーターに **AS LOCATOR** を指定する: 値の代わりにロケーターを渡すことにより、プロシージャーとの間で受け渡しするバイト数を削減できることがあります。これは、パラメーターの値が非常に大きい場合に便利です。 AS LOCATOR 文節は、実際の値の代わりにパラメーターの値へのロケーターを渡すことを指定します。 AS LOCATOR は、LOB または XML データ・タイプまたは LOB または XML データ・タイプに基づく特殊タイプのパラメーターの場合に限り使用するようにしてください。

SQL プロシージャーには、AS LOCATOR は指定できません。

スキーマ内のプロシージャーの固有性を判別する: 現行サーバーでは、それぞれのプロシージャー・シグニチャーを固有のものにする必要があります。プロシージャーのシグニチャーは、修飾プロシージャー名と、パラメーターの数を組み合わせたものです (パラメーターのデータ・タイプはプロシージャーのシグニチャーの一部ではありません)。これは、2 つの異なるスキーマに、名前が同じでパラメーター数も同じであるプロシージャーが含まれていてもよいということを意味します。ただし、1 つのスキーマに、名前もパラメーター数も同じである 2 つのプロシージャーを含めることはできません。

プロシージャーの特定名: 名前もスキーマも同一の (ただしパラメーター数は異なる) 複数のプロシージャーを定義するときは、特定名も指定することをお勧めします。プロシージャーの除去、プロシージャーに対する権限の認可または取り消し、

またはプロシージャーへのコメントの付加を行うときに、特定名を使用して、そのプロシージャーを一意的に識別することができます。

特定名 を指定しなかった場合、その特定名は、プロシージャー名と同じ名前になります。この特定名の関数やプロシージャーが既に存在している場合は、固有表名の生成に使用される規則にほぼ準拠した固有名が生成されます。

プロシージャー内の特殊レジスター**:** 呼び出し元の特殊レジスターの設定は、呼び出し時にプロシージャーに継承され、戻り時に呼び出し元で復元されます。プロシージャー内で特殊レジスターが変更されることもありますが、それらの変更は呼び出し元には影響しません。

CREATE PROCEDURE (外部)

CREATE PROCEDURE (外部) ステートメントは、現行サーバーで外部プロシージャーを定義します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- SYSPROCS カタログ・ビューと SYSPARMS カタログ表の場合
  - 該当の表に対する INSERT 特権、および
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

外部プログラムやサービス・プログラムが存在している場合、このステートメントの権限 ID が保持する特権には、少なくとも次のいずれか 1 つを含める必要があります。

- SQL ステートメントで参照された外部プログラムやサービス・プログラムの場合
  - その外部プログラムやサービス・プログラムが含まれるスキーマに対する USAGE 特権
  - その外部プログラムやサービス・プログラムに対するシステム権限の *EXECUTE。
  - そのプログラムやサービス・プログラムに対するシステム権限の *CHANGE。システムには、プログラムまたはサービス・プログラム・オブジェクトを更新し、プロシージャーを別のシステムに保管/復元するために必要な情報を入れる場合にこの権限が必要となります。ユーザーにこの権限が与えられていない場合、プロシージャーは同じように作成されますが、プログラムまたはサービス・プログラム・オブジェクトは更新されません。
- データベース管理者権限

特殊タイプまたは配列タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別された、それぞれの特殊タイプまたは配列タイプごとに、
  - そのタイプに対する USAGE 特権、および
  - 特殊タイプまたは配列タイプが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

既存のプロシージャーに置き換えるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

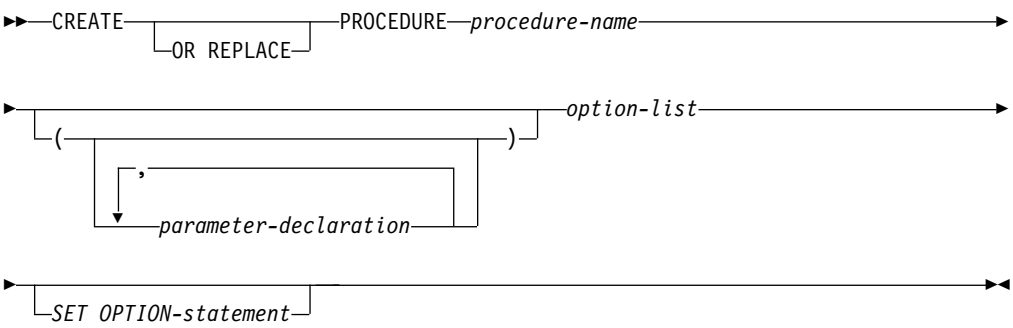
- 次のシステム権限

- このプロシージャーに関連したプログラム・オブジェクトに対する *OBJMGT システム権限
- このプロシージャーを削除するために必要な全権限
- SYSPROCS カタログ・ビューと SYSPARMS カタログ表に対する *READ システム権限

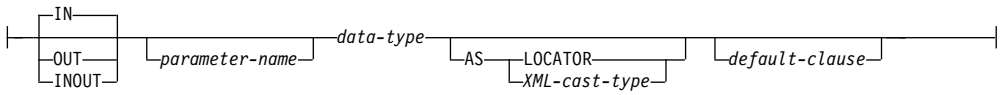
- データベース管理者権限

SQL 特権に対応するシステム権限の説明については、『関数またはプロシージャーへの権限を検査する際の対応するシステム権限』および 『特殊タイプへの権限を検査する際の対応するシステム権限』を参照してください。

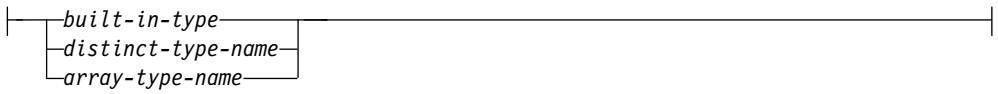
構文



parameter-declaration:

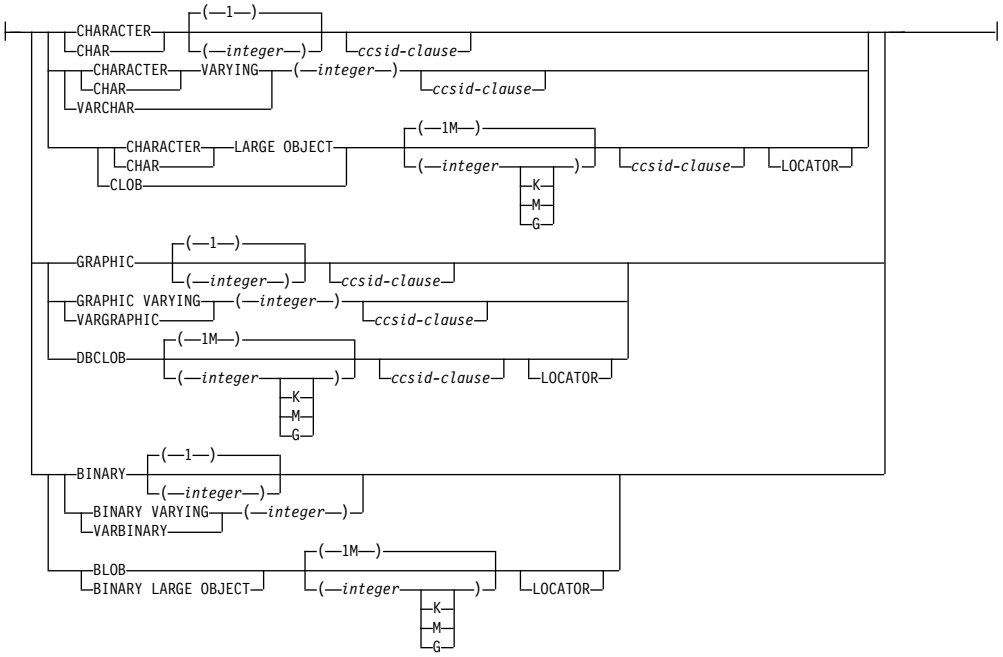


data-type:

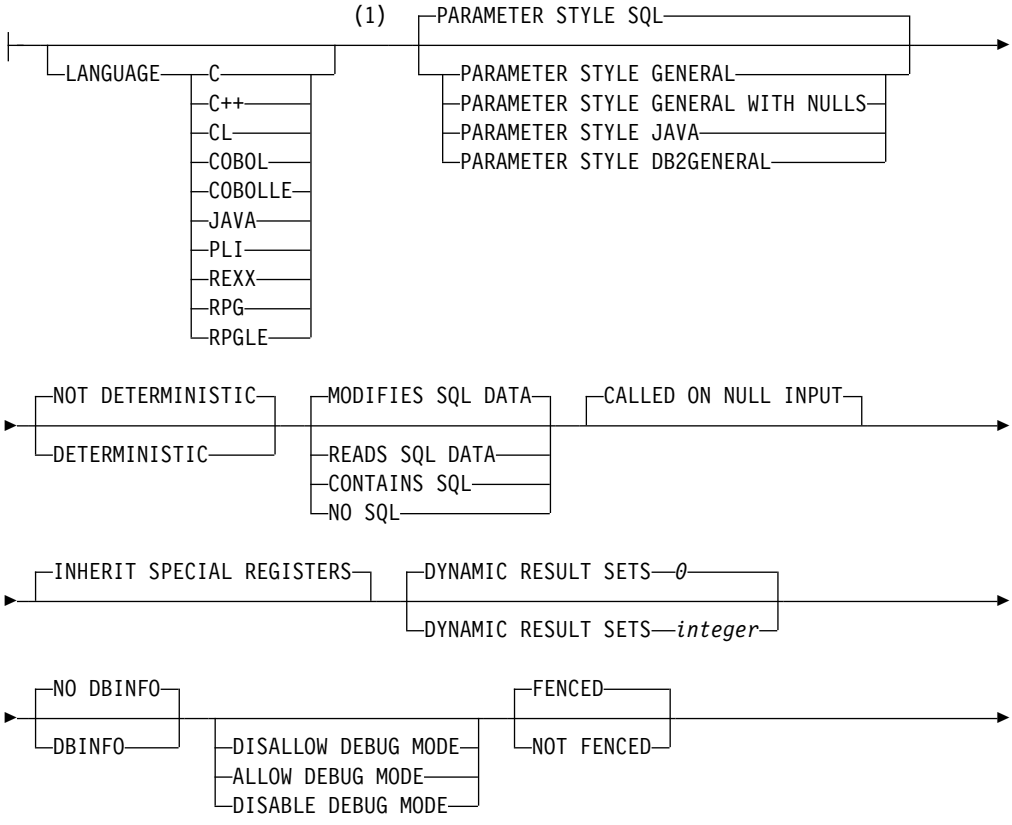


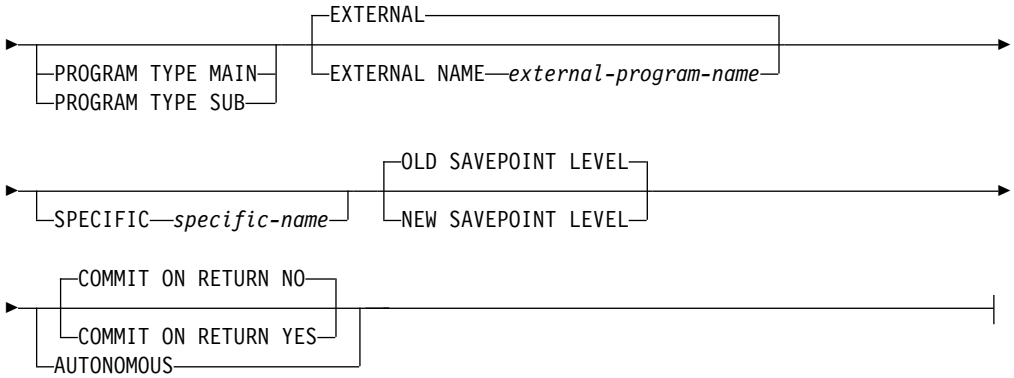
default-clause:

XML-cast-type:


CHARACTER
CHAR
VARCHAR
VARYING
LARGE OBJECT
CLOB
GRAPHIC
VARGRAPHIC
DBCLOB
BINARY
VARBINARY
BLOB
integer
ccsid-clause
LOCATOR
K
M
G

option-list:

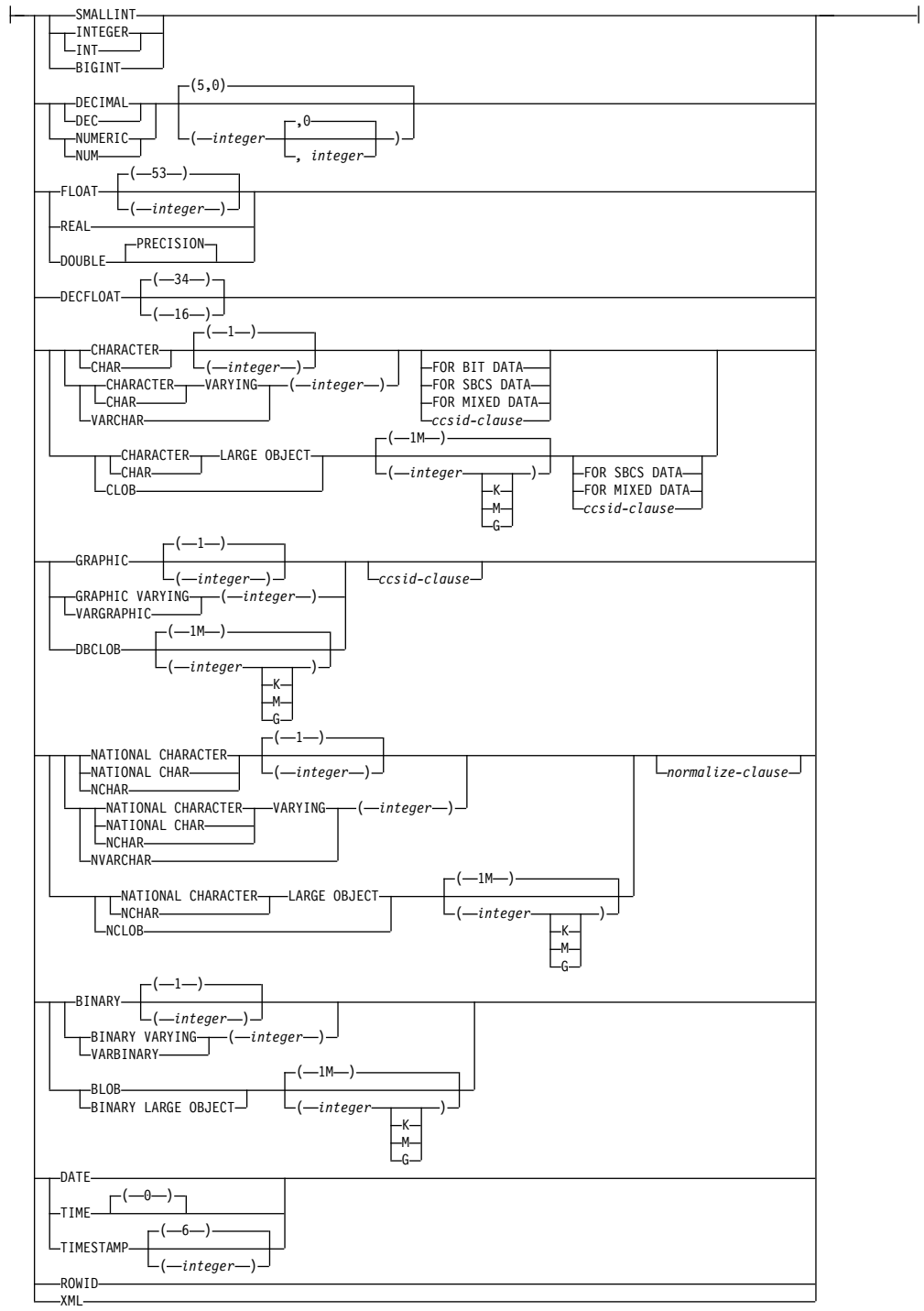

(1)
LANGUAGE
C
C++
CL
COBOL
COBOLLE
JAVA
PLI
REXX
RPG
RPGLE
PARAMETER STYLE SQL
PARAMETER STYLE GENERAL
PARAMETER STYLE GENERAL WITH NULLS
PARAMETER STYLE JAVA
PARAMETER STYLE DB2GENERAL
NOT DETERMINISTIC
DETERMINISTIC
MODIFIES SQL DATA
READS SQL DATA
CONTAINS SQL
NO SQL
CALLED ON NULL INPUT
INHERIT SPECIAL REGISTERS
DYNAMIC RESULT SETS 0
DYNAMIC RESULT SETS integer
NO DBINFO
DBINFO
DISALLOW DEBUG MODE
ALLOW DEBUG MODE
DISABLE DEBUG MODE
FENCED
NOT FENCED



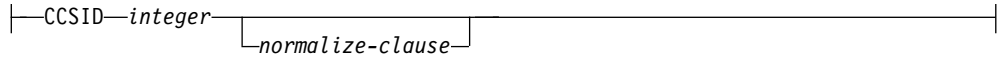
注:

1 オプション文節は、別の順序で指定することができます。

built-in-type:



ccsid-clause:



normalize-clause:

説明

OR REPLACE
現行サーバーにこのプロシージャーの定義が存在する場合に、その定義を置き換える、という動作を指定します。実際には、カタログで既存の定義を削除してから新しい定義に置き換える、という動作になりますが、例外として、このプロシージャーに対して与えられていた特権は影響を受けません。現行サーバーにこのプロシージャーの定義が存在しなければ、このオプションは無視されます。既存のプロシージャーを置き換えるには、新しい定義の *specific-name* および *procedure-name* が古い定義の *specific-name* および *procedure-name* と同じであるか、新しい定義のシグニチャーが古い定義のシグニチャーと一致している必要があります。そうでなければ、新しいプロシージャーが作成されます。

procedure-name
プロシージャーを指定します。名前、スキーマ名、パラメーターの数の組み合わせで、現行サーバーに存在しているプロシージャーを識別してはなりません。

SQL 命名の場合、プロシージャーは、暗黙または明示修飾子で指定されたスキーマ内に作成されます。

システム命名の場合、プロシージャーは、修飾子によって指定されたスキーマ内に作成されます。修飾子を指定しなかった場合:

- CURRENT SCHEMA 特殊レジスターの値が *LIBL である場合、プロシージャーは、現行ライブラリー (*CURLIB) 内に作成されます。
- そうでない場合、プロシージャーは現行スキーマ内に作成されます。

(*parameter-declaration,...***)**
プロシージャーのパラメーターの数とそれぞれのパラメーターのデータ・タイプを指定します。 プロシージャーに関するパラメーターは、入力専用、出力専用、または入出力両用に使用できます。すべてのパラメーターが NULL 可能です。それぞれのパラメーターに名前を指定することができますが、これは必須ではありません。

CREATE PROCEDURE で使用できるパラメーターの最大数は言語のタイプとパラメーター・スタイルによって異なり、以下のようになります。

- JAVA と ILE プログラムおよびサービス・プログラムの場合、最大数は 2000 です。
- OPM プログラムおよび REXX の場合は、以下のようになります。
  - PARAMETER STYLE GENERAL を指定した場合、最大数は 255 です。
  - PARAMETER STYLE GENERAL WITH NULLS を指定した場合、最大数は 254 です。
  - PARAMETER STYLE SQL を指定した場合、最大数は 254 です。

パラメーターの最大数は、その言語で許されるパラメーターの最大数によってさらに制限される可能性があります。

IN パラメーターが、プロシージャーへの入力パラメーターであることを指定し

ます。プロシージャー内でパラメーターに対する変更が行われても、制御が戻った後で、呼び出し元の SQL アプリケーションがその変更内容を使用することはできません。[98]デフォルトは IN です。

OUT
パラメーターが、プロシージャーから戻される出力パラメーターであることを示します。

データ・リンクやデータ・リンクをベースとした特殊タイプは、出力パラメーターとして指定することはできません。

INOUT
パラメーターが、このプロシージャー用の入出力両方のパラメーターであることを指定します。 INOUT パラメーターがデフォルトと共に定義されていて、プロシージャーの呼び出し時にそのデフォルトが使用される場合、パラメーターに戻される値はありません。

データ・リンクやデータ・リンクをベースとした特殊タイプは、入出力パラメーターとして指定することはできません。

parameter-name
パラメーター名を指定します。この名前は、このプロシージャー用の他のパラメーター名 と同じものであってはなりません。

data-type
パラメーターのデータ・タイプを指定します。

built-in-type
組み込みデータ・タイプを指定します。それぞれの組み込みデータ・タイプの詳細については、 1238 ページの『CREATE TABLE』を参照してください。

distinct-type-name
特殊タイプを指定します。パラメーターの長さ、精度、位取り、コード化スキームの属性は、 1328 ページの『CREATE TYPE (特殊)』 で指定する特殊タイプのソース・タイプの属性になります。

特殊タイプの名前が修飾されていない場合、データベース・マネージャーは、SQL パス上のスキーマを検索することでそのスキーマ名を解決します。

配列タイプ名
配列タイプを指定します。配列タイプは、LANGUAGE JAVA でのみサポートされています。Java ストアード・プロシージャーのパラメーターとして配列タイプを使用する場合は、パラメーター・スタイルが JAVA になっていなければなりません。

配列タイプの名前が修飾されていない場合、データベース・マネージャーは、SQL パス上のスキーマを検索することでそのスキーマ名を解決します。

98. 言語タイプが REXX の場合、パラメーターは、すべて、入力パラメーターでなければなりません。

CCSID が指定されている場合、プロシージャーに渡される前に、パラメーターはその CCSID に変換されます。CCSID が指定されていない場合は、CCSID は、プロシージャーの呼び出し時点における現行サーバーのデフォルトの CCSID によって決まります。

日付、時刻、およびタイム・スタンプの各パラメーターは、ISO 形式の文字ストリングとしてプロシージャーに渡されます。

データ・タイプによってはすべての言語ではサポートされていないものもあります。 SQL データ・タイプとホスト言語データ・タイプのマッピングについての詳細は、「組み込み SQL プログラミング」トピック集を参照してください。組み込みデータ・タイプの仕様は、プロシージャーの作成に使用する言語に対応していれば、指定することができます。

XML タイプのパラメーターでは、*XML-cast-type* 文節または AS LOCATOR 文節を指定する必要があります。

AS LOCATOR
これを指定すると、パラメーターは、実際の値ではなく、値のロケーターになります。AS LOCATOR は、パラメーターに LOB または XML データ・タイプや LOB または XML データ・タイプをベースとする特殊タイプが指定されている場合にのみ、指定することができます。AS LOCATOR を指定した場合、FOR SBCS DATA または FOR MIXED DATA を指定してはなりません。

AS *XML-cast-type*
XML タイプまたは XML タイプに基づく特殊タイプのパラメーターのデータ・タイプとしてこのプロシージャーに渡すデータ・タイプを指定します。LOCATOR を指定する場合は、パラメーターが実際の値ではなく値のロケーターになります。

CCSID 値を指定する場合に、グラフィック・データ・タイプとして指定できるのは、Unicode CCSID 値に限られます。CCSID 値を指定しない場合は、このプロシージャーの作成時に、SQL_XML_DATA_CCSID QAQQINI オプションの設定に基づいて CCSID が設定されます。デフォルトの CCSID は 1208 です。このオプションの説明については、 101 ページの『XML 値』を参照してください。

default
パラメーターのデフォルト値を指定します。デフォルト値は、定数、特殊レジスター、グローバル変数、式、またはキーワード NULL にすることができます。式は、集約関数および列名を含まない、 196 ページの『式』で定義されている任意の式です。デフォルト値が指定されていない場合、パラメーターにデフォルト値がないため、プロシージャーの呼び出し時に省略できません。式ストリングの最大長は 64K です。

デフォルトの式で SQL データを変更することはできません。式は、パラメーターのデータ・タイプに対して割り当ての互換性がなければなりません。デフォルト式の中で参照されるすべてのオブジェクトは、プロシージャーが作成されるときに存在している必要があります。プロシージャーが呼び出されると、デフォルトは起動側の権限を使用して評価されます。

デフォルト式内でリスト中の数値定数を区切る区切り記号として使用するコンマの後には、スペースが 1 つ必要です。

デフォルトを指定できません。

- OUT パラメーターに対して
- 配列タイプのパラメーターに対して

LANGUAGE

その外部プログラムまたはサービス・プログラムの作成に使用されている言語を指定します。この文節は、外部プログラムが REXX プロシージャーである場合に必要です。

LANGUAGE の指定がない場合は、プロシージャーの作成時点で、該当の外部プログラムまたはサービス・プログラムに関連する属性情報から、LANGUAGE を決定します。該当のプログラムまたはサービス・プログラムに関連する属性情報では認識可能な言語が識別されない場合、または該当のプログラムまたはサービス・プログラムが見つからない場合は、言語は C であると見なされます。

C 外部プログラムは C で作成されます。

C++
外部プログラムは C++ で作成されます。

CL 外部プログラムは CL で作成されます。

COBOL
外部プログラムは COBOL で作成されます。

COBOLLE
外部プログラムは ILE COBOL で作成されます。

JAVA
外部プログラムは JAVA で作成されます。

PLI
外部プログラムは PL/I で作成されます。

REXX
外部プログラムは REXX プロシージャーです。

RPG
外部プログラムは RPG で作成されます。

RPGLE
外部プログラムは ILE RPG で作成されます。

PARAMETER STYLE

プロシージャーにパラメーターを渡し、プロシージャーから値を戻すために使用する規則を指定します。

SQL
CALL ステートメントに指定されているパラメーターに加えて、幾つかの追加パラメーターをプロシージャーに渡すことを指定します。これらのパラメーターは、次の順序で配列されるように定義されます。

- 最初の *n* 個のパラメーターは、CREATE PROCEDURE ステートメント上に指定されるパラメーターです。
- パラメーターに対する標識変数用の *n* 個のパラメーター

- SQLSTATE の CHAR(5) 出力パラメーター。戻される SQLSTATE は、プロシージャーが成功したかどうかを示します。戻される SQLSTATE は、外部プログラムによって割り当てられたものです。

  ユーザーは、プロシージャーからエラーまたは警告を戻すために、外部プログラム内で SQLSTATE を任意の有効な値にセットすることができます。
- 完全修飾プロシージャー名の VARCHAR(517) 入力パラメーター。
- 特定の名前の VARCHAR(128) 入力パラメーター。
- メッセージ・テキストの VARCHAR(1000) 出力パラメーター。

以下の追加のパラメーターを最後のパラメーターとして渡すことができます。

- DBINFO が CREATE PROCEDURE ステートメント上に指定されている場合は、dbinfo 構造体のパラメーター。

これらのパラメーターは、指定の LANGUAGE に基づいて渡されます。例えば、言語が C または C++ であれば、VARCHAR パラメーターはヌル終了ストリングとして渡されます。 渡されるパラメーターについての詳細は、ライブラリー QSYSINC の該当するソース・ファイル内の組み込み sqludf を参照してください。例えば、C の場合、sqludf は QSYSINC/H で見つかります。

LANGUAGE JAVA を指定した場合は、PARAMETER STYLE SQL は使用できません。

DB2GENERAL
このプロシージャーに、Java メソッド用として定義されているパラメーター引き渡し規則を使用することを指定します。

PARAMETER STYLE DB2GENERAL を指定できるのは、LANGUAGE JAVA を指定した場合のみです。 Java でのパラメーター引き渡しの詳細については、「IBM Developer Kit for Java」を参照してください。

GENERAL
このプロシージャーが CALL に指定されているパラメーターを受け取るようなパラメーター引き渡しメカニズムを使用することを指定します。標識変数に対し、引数がさらに渡されることはありません。

LANGUAGE JAVA を指定した場合は、PARAMETER STYLE GENERAL は使用できません。

GENERAL WITH NULLS
CALL ステートメントで GENERAL に指定されているパラメーターに加えて、他の引数もプロシージャーに渡すことを指定します。この追加の引数には、CALL ステートメントの各パラメーターについてそれぞれ 1 つずつエレメントがある標識配列が含まれています。 C では、これは多くの場合、短精度整数の配列です。標識の処理方法について詳しくは、「SQL プログラミング」トピック集を参照してください。

LANGUAGE JAVA を指定した場合は、PARAMETER STYLE GENERAL WITH NULLS は使用できません。

JAVA
このプロシージャーで、Java 言語および ISO/IEC FCD 9075-13:2003 「*Information technology - Database languages - SQL - Part 13: Java Routines and Types (SQL/JRT)*」仕様に準拠するパラメーター引き渡し規則を使用することを指定します。 INOUT および OUT パラメーターは、値を戻しやすくするために、単一項目配列として渡されます。

PARAMETER STYLE JAVA を指定できるのは、LANGUAGE JAVA を指定した場合だけです。移植性を高めるためには、PARAMETER STYLE JAVA 規則を使用する Java プロシージャーを書く必要があります。 Java でのパラメーター引き渡しの詳細については、「IBM Developer Kit for Java」トピック集を参照してください。

パラメーターを渡す方法は、外部プロシージャーの言語によって決まります。例えば、C では、VARCHAR または CHAR パラメーターは NULL 文字で終了するストリングとして渡されます。詳しくは、「SQL プログラミング」のトピック集を参照してください。 Java ルーチンについては、「IBM Developer Kit for Java」のトピック集を参照してください。

DETERMINISTIC または NOT DETERMINISTIC
このプロシージャーが、同じ IN 引数および INOUT 引数を指定して呼び出された場合に、常に同じ結果を戻すかどうかを指定します。デフォルトは NOT DETERMINISTIC です。

NOT DETERMINISTIC
このプロシージャーは、同じ IN 引数および INOUT 引数を指定して呼び出された場合に、データベース内の参照先データが変更されていなくても、必ずしも同じ結果を戻すとは限りません。

DETERMINISTIC
このプロシージャーは、同じ IN 引数および INOUT 引数を指定して呼び出された場合に、データベース内の参照先データが変更されていない限り、常に同じ結果を戻します。

MODIFIES SQL DATA、READS SQL DATA、CONTAINS SQL、または NO SQL
このプロシージャーが実行できる SQL ステートメントおよびネストされたルーチンの種別を指定します。データベース・マネージャーは、プロシージャーと、プロシージャーからローカルで呼び出すすべてのルーチンが発行する SQL ステートメントが、この指定と整合しているかどうかを検査します。ネストされたリモート・ルーチンが呼び出された場合、検査は実行されません。各ステートメントの分類については、 1855 ページの『付録 B. SQL ステートメントの特性』を参照してください。デフォルトは、MODIFIES SQL DATA です。このオプションは、パラメーター・デフォルト式では無視されます。

MODIFIES SQL DATA
このプロシージャーで、どのプロシージャーでもサポートされないステートメントを除くすべての SQL ステートメントを実行できることを指定します。

READS SQL DATA
このプロシージャーが、データ・アクセス種別 READS SQL DATA、CONTAINS SQL、または NO SQL を指定したステートメントを実行できるように指定します。

CONTAINS SQL
このプロシージャーが、データ・アクセス種別 CONTAINS SQL または NO SQL のステートメントのみを実行できるように指定します。

NO SQL
このプロシージャーが、データ・アクセス種別 NO SQL の SQL ステートメントのみを実行できるように指定します。

CALLED ON NULL INPUT
引数値のいずれかまたは全部がヌルである場合にプロシージャーを呼び出すことを指定します。この指定は、ヌル引数値のテストを行うようにプロシージャーをコーディングする必要があります。

INHERIT SPECIAL REGISTERS
特殊レジスターの既存の値は、プロシージャーに入った後に継承されることを示します。

FENCED または NOT FENCED
このパラメーターは、他のプロダクトとの互換性を保持するために許可されており、Db2 for i で使用されることはありません。

DYNAMIC RESULT SETS *integer*
プロシージャーから戻すことのできる結果セットの最大数を指定します。*integer* の最小値はゼロ、最大値は 32767 です。

DYNAMIC RESULT SETS 文節を指定しなければ、プロシージャーの終了時にオープンしたままになっているすべてのカーソルの結果セットが返されます。

結果セットは、対応するカーソルがオープンした順序で返されます。ただし、プロシージャーで SET RESULT SETS ステートメントを実行する場合は例外です。プロシージャーの終了時に結果セットのためにオープンしたままになっているカーソルの数が、DYNAMIC RESULT SETS 文節で指定した最大数を超えている場合は、CALL ステートメントで警告が返され、DYNAMIC RESULT SETS 文節で指定した数の結果セットが返されます。

SET RESULT SETS ステートメントを発行した場合は、戻される結果の数は、このキーワードに指定した結果セットの数と、 SET RESULT SETS ステートメントに指定した結果セットの数のいずれか少ない方です。 SET RESULT SETS ステートメントに結果セットの最大数よりも大きい値が指定された場合、警告が戻されます。 RETURN TO CLIENT 属性を持つカーソルからの結果セットは、最外部プロシージャーの結果セットの数に含まれます。

結果セットを戻すのにカーソルが使用され、カーソルがスクロール可能である場合、結果セットはスクロール可能です。結果セットを戻すのにカーソルが使用された場合、結果セットはカーソル位置から始まります。つまり、5 つの FETCH NEXT 操作が実行された後、プロシージャーから戻った場合、結果セットは、結果セットの 6 行目から始まります。

カーソルの結果セットが返されるのは、外部プログラムに ACTGRP(*NEW) の属性がない場合に限られます。

結果セットの詳細については、 1724 ページの『SET RESULT SETS』を参照してください。

DISALLOW DEBUG MODE、ALLOW DEBUG MODE、または DISABLE DEBUG MODE
プロシージャーを Unified Debugger でデバッグできるように作成するかどう

かを示します。DEBUG MODE が指定されない場合、プロシージャーは CURRENT DEBUG MODE 特殊レジスターで指定されるデバッグ・モードを使用して作成されます。

DEBUG MODE を指定できるのは、LANGUAGE JAVA を指定した場合のみです。

DISALLOW DEBUG MODE
プロシージャーは Unified Debugger でデバッグできません。プロシージャーの DEBUG MODE 属性が DISALLOW の場合、後でプロシージャーを変更してデバッグ・モード属性を変えることができます。

ALLOW DEBUG MODE
プロシージャーは Unified Debugger でデバッグすることができます。プロシージャーの DEBUG MODE 属性が ALLOW の場合、後でプロシージャーを変更してデバッグ・モード属性を変えることができます。

DISABLE DEBUG MODE
プロシージャーは Unified Debugger でデバッグできません。プロシージャーの DEBUG MODE 属性が DISABLE の場合、後でプロシージャーを変更してデバッグ・モード属性を変えることはできません。

PROGRAM TYPE
他の製品との互換性を備えるために、このパラメーターが許可されています。これは、ルーチンの外部プログラムが、プログラム (*PGM) であるか、サービス・プログラム (*SRVPGM) のプロシージャーであるかを示します。

SUB
プロシージャーがサービス・プログラムのプロシージャーとして実行することを指定します。外部プログラムは、*SRVPGM オブジェクトでなければなりません。

MAIN
ルーチンがプログラムのメインエントリー・ポイントとして実行することを指定します。外部プログラムは、*PGM オブジェクトでなければなりません。

DBINFO
プロシージャーを呼び出すときに、追加の状況情報を渡すかどうかを指定します。デフォルトは NO DBINFO です。

NO DBINFO
追加情報は渡されません。

DBINFO
プロシージャーを呼び出すときに、追加の引数が渡されます。

この引数は、現行サーバーの名前、アプリケーション実行時許可 ID、およびプロシージャーを呼び出したデータベース・マネージャーのバージョンおよびリリースの ID などの情報を含む構造体です。詳細については、 1202 ページの表 82を参照してください。 DBINFO 構造体についての詳しい情報は、ライブラリー QSYSINC 内の該当するソース・ファイルの組み込み sqludf に入っています。例えば、C の場合、sqludf は QSYSINC/H で見つかります。

DBINFO は、PARAMETER STYLE SQL でのみ許可されます。

表 82. *DBINFO* フィールド

フィールド	データ・タイプ	説明
リレーショナル・データベース	VARCHAR(128)	現行サーバーの名前
権限 ID	VARCHAR(128)	実行時権限 ID
CCSID 情報	INTEGER INTEGER INTEGER INTEGER INTEGER INTEGER INTEGER INTEGER INTEGER INTEGER CHAR(8)	ジョブの CCSID 情報。3 つの CCSID が 3 セット戻されます。各セット内の 3 つの CCSID を識別する情報は、次のとおりです。 • SBCS CCSID • DBCS CCSID • 混合 CCSID 3 セットの CCSID の後に、3 セットの CCSID のうちのどのセットが該当するかを示す整数と 8 バイトの予約済みスペースが続きます。 CCSID の各設定は、異なるエンコード・スキーム (EBCDIC、ASCII、およびユニコード) のためのものです。 CREATE PROCEDURE ステートメントのパラメーターの 1 つとして明示的に CCSID を指定していない場合は、入力ストリングは、プロシージャーの実行時にジョブの CCSID でコード化されるものと見なされます。入力ストリングの CCSID がパラメーターの CCSID と同じではない場合は、この外部プロシージャーに渡される入力ストリングは、外部プログラムの呼び出しの前に変換されます。
ターゲット列	VARCHAR(128) VARCHAR(128) VARCHAR(128)	プロシージャーへの呼び出しには適用されません。
バージョンとリリース	CHAR(8)	データベース・マネージャーのバージョン、リリース、および修正レベル。
プラットフォーム	INTEGER	サーバーのプラットフォーム・タイプ。

EXTERNAL

CREATE PROCEDURE ステートメントを使用して、外部プログラミング言語によるコードに基づいた新規プロシージャーを定義するように指定します。

NAME 文節を指定しない場合は、「NAME *procedure-name*」が使用されます。その場合、*procedure-name* は 8 文字以下でなければなりません。LANGUAGE JAVA プロシージャーの場合、デフォルト名は Java プロシージャーに対して有効でないので、NAME 文節が必要です。

NAME *external-program-name*

該当のプロシージャーが CALL ステートメントによって呼び出される時点で実行されるプログラムまたはサービス・プログラムを指定します。このプログラム名は、プロシージャーの呼び出し時点で該当のアプリケーション・

サーバーに存在しているプログラムまたはサービス・プログラムを識別するものでなければなりません。命名オプションが *SYS であり、その名前が修飾されていない場合:

- プロシージャーの呼び出し時に現行パスを使用して該当のプログラムを検索します。
- COMMENT、GRANT、LABEL、REVOKE の各操作をそのプロシージャーで実行する時点で、*LIBL を使用して対象のプログラムまたはサービス・プログラムを検索します。

この名前の妥当性は、アプリケーション・サーバーで検査されます。名前の形式が正しくない場合、エラーが戻されます。

この外部プログラムまたはサービス・プログラムは、プロシージャーの作成時点に存在している必要はありませんが、プロシージャーの呼び出し時点には存在していなければなりません。

CONNECT、SET CONNECTION、RELEASE、DISCONNECT、および SET TRANSACTION ステートメントは、リモート・アプリケーション・サーバー上で実行中のプロシージャー内で使用することはできません。COMMIT および ROLLBACK ステートメントは、 ATOMIC SQL プロシージャーまたはリモート・アプリケーション・サーバーへの接続上で実行中のプロシージャー内で使用することはできません。

SPECIFIC *specific-name*
関数の固有名を指定します。特定名の詳細については、 1187 ページの『CREATE PROCEDURE』の『プロシージャーに特定の名前を指定する』を参照してください。

OLD SAVEPOINT LEVEL または **NEW SAVEPOINT LEVEL**
このプロシージャーに入ったときに、新しいセーブポイント・レベルを作成するかどうかを指定します。

OLD SAVEPOINT LEVEL
新しいセーブポイント・レベルを作成しません。このプロシージャー内で、OLD SAVEPOINT LEVEL が暗黙的または明示的に指定された SAVEPOINT ステートメントが発行された場合、SAVEPOINT ステートメントはプロシージャーの呼び出し元と同じセーブポイント・レベルで作成されます。これはデフォルトです。

NEW SAVEPOINT LEVEL
このプロシージャーに入ったときに、新しいセーブポイント・レベルが作成されます。プロシージャー内に設定されているすべてのセーブポイントは、このプロシージャーが呼び出されたレベルより深くネストされたセーブポイント・レベルで作成されます。したがって、プロシージャー内のどの新規セーブポイントも、同じ名前を持つ上位のセーブポイント・レベル (例えば呼び出し側プログラムまたはサービス・プログラムのセーブポイント・レベル) で設定されている既存のセーブポイントと競合することはありません。

COMMIT ON RETURN
データベース・マネージャーが、プロシージャーからの戻りと同時にトランザクションをコミットするかどうかを指定します。

NO データベース・マネージャーは、プロシージャーから戻ったときにコミットを行いません。 NO がデフォルトです。

YES

データベース・マネージャーは、プロシージャーから正常に戻ったときにコミットを行います。プロシージャーの戻り時にエラーがあった場合は、コミットは行われません。

コミット操作の対象には、呼び出し側アプリケーション・プロセスおよびこのプロシージャーが行う作業が含まれます。[99]

プロシージャーが結果セットを戻す場合に、結果セットに関連したカーソルをコミット後に使用できるようにするには、カーソルを WITH HOLD として定義しておく必要があります。

AUTONOMOUS

呼び出し側アプリケーションから独立した作業単位でプロシージャーが実行されることを指定します。このオプションが指定されていると、データベースは常にプロシージャーから戻された SQLSTATE に基づいて、自律型プロシージャーのトランザクション作業をコミットまたはロールバックします。SQLSTATE が無条件な成功または警告を示している場合、トランザクションはコミットされます。他のすべての SQLSTATE の場合、自律型プロシージャーの作業単位はロールバックされます。

自律型プロシージャーの内部からの、トリガー、関数、またはプロシージャーの起動は、そのトリガー、関数、またはプロシージャーが、別の活動化グループの下で実行するように明示的に作成されたものでない限り、自律型プロシージャーの作業単位の一部になります。

自律型プロシージャーを別の自律型プロシージャーから直接または間接に呼び出すことはできません。

AUTONOMOUS を指定する場合、DYNAMIC RESULT SETS 0 を指定する必要があります。

SET OPTION-statement

パラメーター・デフォルトに使用されるオプションを指定します。各オプションのデフォルト値は、作成時に有効だったオプションによって異なります。詳しくは、 1696 ページの『SET OPTION』を参照してください。

デフォルト値式を処理するときには、オプション
ALWCPYDTA、CONACC、DATFMT、DATSEP、DECFLTRND、DECMPT、DECRESULT、DFTRDBCOL、
LANGID、SQLCURRULE、SQLPATH、SRTSEQ、TGTRLS、TIMFMT、および TIMSEP が使用されます。オプション
CNULRQD、CNULIGN、COMPILEOPT、EXTIND、NAMING、SQLCA は、CREATE PROCEDURE ステートメントでは使用できません。他のオプションは、受け入れられますが、無視されます。

99. 外部プログラムまたはサービス・プログラムが ACTGRP(*NEW) を指定して作成されており、ジョブ・コミットメント定義を使用しない場合は、プロシージャーにより行われた作業は、活動化グループ終了に伴ってコミットまたはロールバックされます。

注

プロシージャー定義に関する一般考慮事項: プロシージャーの定義に関する一般情報については、 1187 ページの『CREATE PROCEDURE』を参照してください。

言語に関する考慮事項: プロシージャー用プログラムの作成に必要な情報については、「組み込み SQL プログラミング」を参照してください。

REPLACE の規則: REPLACE によって外部プロシージャーを再作成する場合は、以下のようになります。

- 既存のコメントまたはラベルは破棄されます。
- 別の外部プログラムを指定する場合、
  - 権限を持つユーザーは新しいプログラムにコピーされません。
  - ジャーナル監査は変更されません。
- 上記以外の場合、
  - 権限を持つユーザーは維持されます。オブジェクト所有者は変更されない可能性があります。
  - 現在のジャーナル監査は変更されません。

エラー処理に関する考慮事項: プロシージャーによってエラーが戻されると、プロシージャーに渡された引数の値のうち OUT パラメーターに対応するものは未定義になり、INOUT パラメーターに対応するものは変わりません。

プロシージャーの作成: ILE 外部プログラムまたはサービス・プログラムに関連した外部プロシージャーが作成されると、その関数に関連したプログラムやサービス・プログラムのオブジェクトへのプロシージャーの属性の保管が試行されます。*PGM または *SRVPGM オブジェクトが保管された後、このシステムまたは別のシステムに復元されると、属性が使用されてカタログが更新されます。

外部プロシージャーの場合は、次の制約の範囲内で属性を保管することができます。

- 外部プログラム・ライブラリーは、QSYS であってはなりません。
- 外部プログラムは、CREATE PROCEDURE ステートメントの発行時に存在していなければなりません。

  システム命名が指定され、外部プログラム名が修飾されない場合は、外部プログラムはライブラリー・リストで検出されなければなりません。
- 外部プログラムは、ILE *PGM オブジェクトか *SRVPGM オブジェクトにする必要があります。
- 外部プログラムには、32 ルーチンの属性が既に入っていてはなりません。

オブジェクトを更新できない場合でも、それにかかわらず、プロシージャーは作成されます。

プロシージャーの呼び出し: DECLARE PROCEDURE ステートメントで、作成されたプロシージャーと同じ名前のプロシージャーを定義し、そのプロシージャー名が変数によって識別されていない静的 CALL ステートメントが、同じソース・プログ

ラムから実行される場合は、CREATE PROCEDURE ステートメントの属性ではなく、DECLARE PROCEDURE ステートメントの属性が使用されます。

CREATE PROCEDURE ステートメントが適用されるのは、静的および動的 CALL ステートメント、ならびにそのプロシージャー名が変数によって識別されている CALL ステートメントです。

外部プロシージャーが呼び出されると、その関数は、外部プログラムやサービス・プログラムの作成時に指定された活動化グループであれば、どの活動化グループ内でも実行します。ただし、通常は、プロシージャーが呼び出し側プログラムと同じ活動化グループ内で実行するように ACTGRP(*CALLER) を使用する必要があります。

AUTONOMOUS 属性を持つプロシージャーの外部プログラムまたはサービス・プログラムは、活動化グループ QSQAUTOAG 内で実行するように定義される必要があります。自律型プロシージャーがジョブ内で起動される場合、QSQAUTOAG 活動化グループ内で実行するすべてのプロシージャーは同じストレージ・モデルで作成されたものでなければなりません。それらはすべて *TERASPACE であるか、すべて *SNGLVL でなければなりません。

デフォルト値の設定: プロシージャーのパラメーターがデフォルト値を指定して定義されていれば、そのプロシージャーの呼び出し時にパラメーターはデフォルト値に設定されますが、それは、対応する引数に値が与えられていない場合か、引数が DEFAULT と指定されている場合に限ります。

Java プロシージャーに関する注釈: Java プロシージャーを実行するためには、システムに IBM Developer Kit for Java をインストールしておく必要があります。インストールされていないと、SQLCODE -443 が戻され、CPDB521 メッセージがジョブ・ログに入ります。

Java プロシージャーの実行中にエラーが発生すると、SQLCODE -443 が戻されます。エラーによっては、プロシージャーが実行されていたジョブのジョブ・ログに他のメッセージが入っている場合があります。

代替構文: 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード VARIANT と NOT VARIANT は、NOT DETERMINISTIC と DETERMINISTIC の同義語として使用することができます。
- キーワード NULL CALL は、CALLED ON NULL INPUT の同義語として使用できます。
- キーワード SIMPLE CALL は、GENERAL の同義語として使用できます。
- DB2GENERAL の同義語として、値 DB2GENRL を使用できます。
- DYNAMIC RESULT SET、RESULT SETS、および RESULT SET は、DYNAMIC RESULT SETS の同義語として使用できます。
- PARAMETER STYLE 文節のキーワード PARAMETER STYLE はオプションです。
- PARAMETER STYLE SQL の同義語として、キーワード PARAMETER STYLE DB2SQL を使用できます。

例

例 *1:* Java で書かれたプロシージャーのプロシージャー定義を作成します。このプロシージャーは、部品番号を渡されて、部品の価格と現在入手可能な数量を戻します。

```
CREATE PROCEDURE PARTS_ON_HAND (IN  PARTNUM  INTEGER,
                                OUT COST     DECIMAL(7,2),
                                OUT QUANTITY INTEGER)
       LANGUAGE JAVA
       PARAMETER STYLE JAVA
       EXTERNAL NAME 'parts.onhand'
```

例 *2:* C で書かれたプロシージャーのプロシージャー定義を作成します。このプロシージャーは、アセンブリー番号を渡されて、アセンブリーを構成する部品の数、部品の合計価格、および部品番号、数量、各部品の単価をリストする結果セットを戻します。

```
CREATE PROCEDURE ASSEMBLY_PARTS (IN  ASSEMBLY_NUM  INTEGER,
                                 OUT NUM_PARTS     INTEGER,
                                 OUT COST DOUBLE)
       LANGUAGE C
       PARAMETER STYLE GENERAL
       DYNAMIC RESULT SETS 1
       FENCED
       EXTERNAL NAME ASSEMBLY
```

CREATE PROCEDURE (SQL)

CREATE PROCEDURE (SQL) ステートメントは、現行サーバーで SQL プロシージャーを作成します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- スキーマ内に作成する特権。詳しくは、スキーマ内で作成する必要のある権限を参照してください。
- データベース管理者権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- SYSPROCS カタログ・ビューと SYSPARMS カタログ表の場合
  - 該当の表に対する INSERT 特権、および
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 次のシステム権限
  - プログラム作成 (CRTPGM) コマンドに対する *USE
- データベース管理者権限

SQL 名が指定され、該当のプロシージャーが作成されるライブラリーの名前と同じ名前のユーザー・プロファイルが存在し、しかもその名前がステートメントの権限 ID と異なっている場合、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- その名前を持つユーザー・プロファイルに対する *ADD システム権限
- データベース管理者権限

特殊タイプまたは配列タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

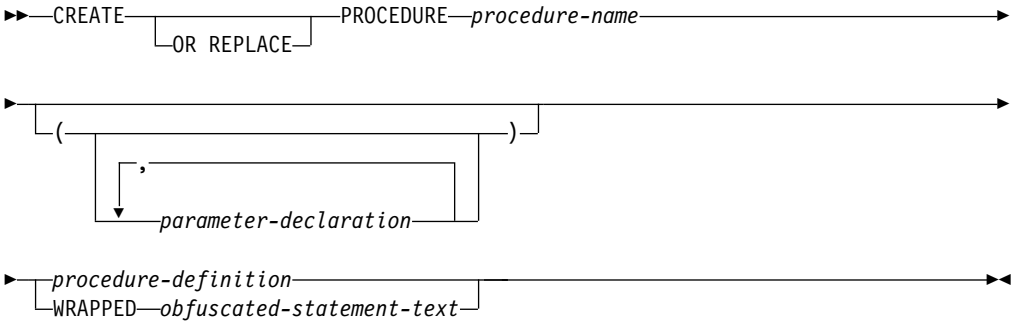
- ステートメント内で識別された、それぞれの特殊タイプまたは配列タイプごとに、
  - そのタイプに対する USAGE 特権、および
  - 特殊タイプまたは配列タイプが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

既存のプロシージャーに置き換えるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

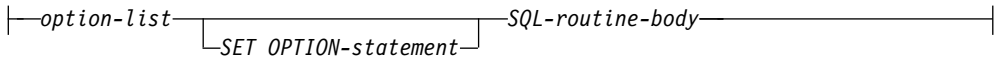
- 次のシステム権限
  - このプロシージャーに関連したプログラム・オブジェクトまたはサービス・プログラム・オブジェクトに対する *OBJMGT システム権限
  - このプロシージャーを削除するために必要な全権限
  - SYSPROCS カタログ・ビューと SYSPARMS カタログ表に対する *READ システム権限
- データベース管理者権限

SQL 特権に対応するシステム権限の説明については、『関数またはプロシージャーへの権限を検査する際の対応するシステム権限』および 『特殊タイプへの権限を検査する際の対応するシステム権限』を参照してください。

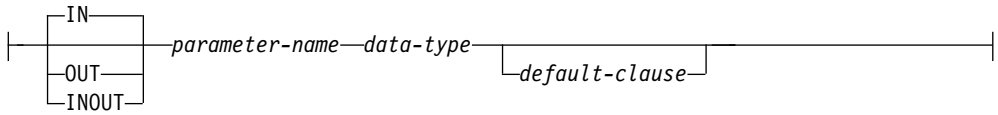
構文



procedure-definition:



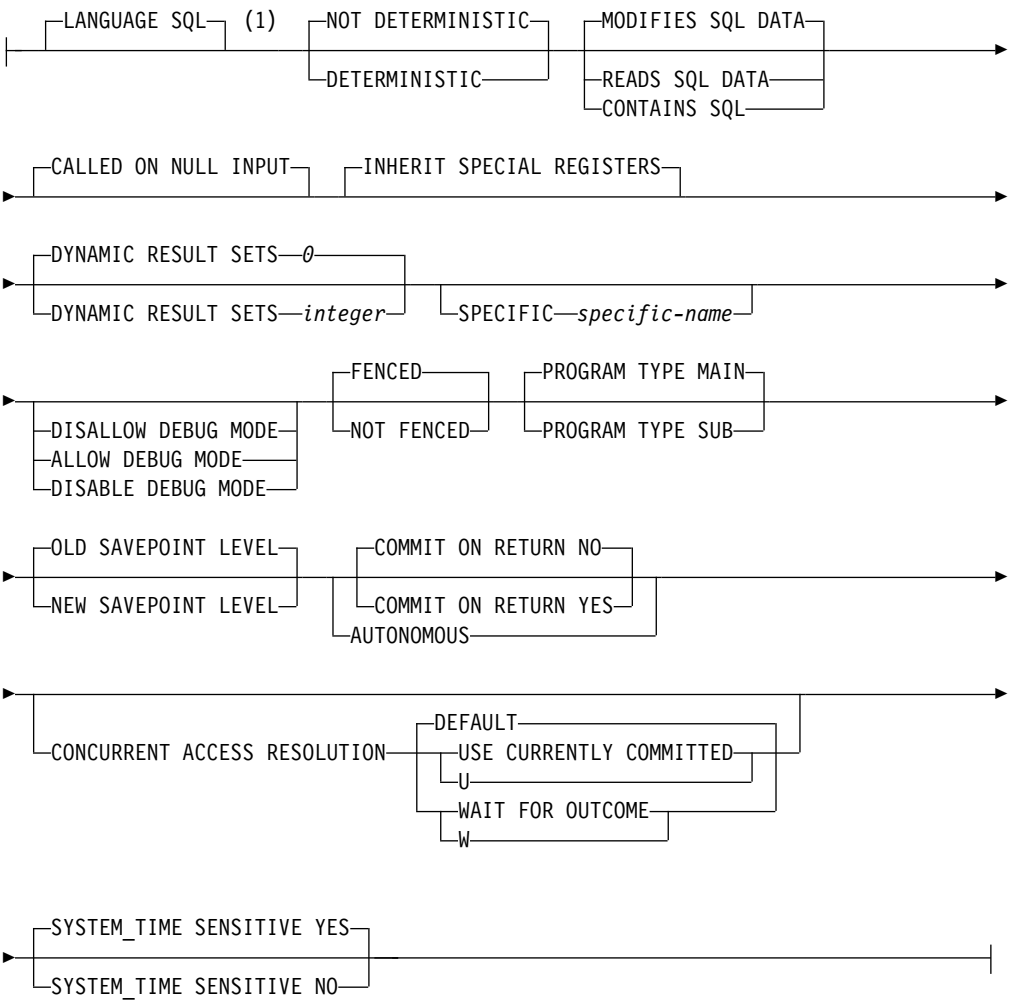
parameter-declaration:



data-type:

default-clause:

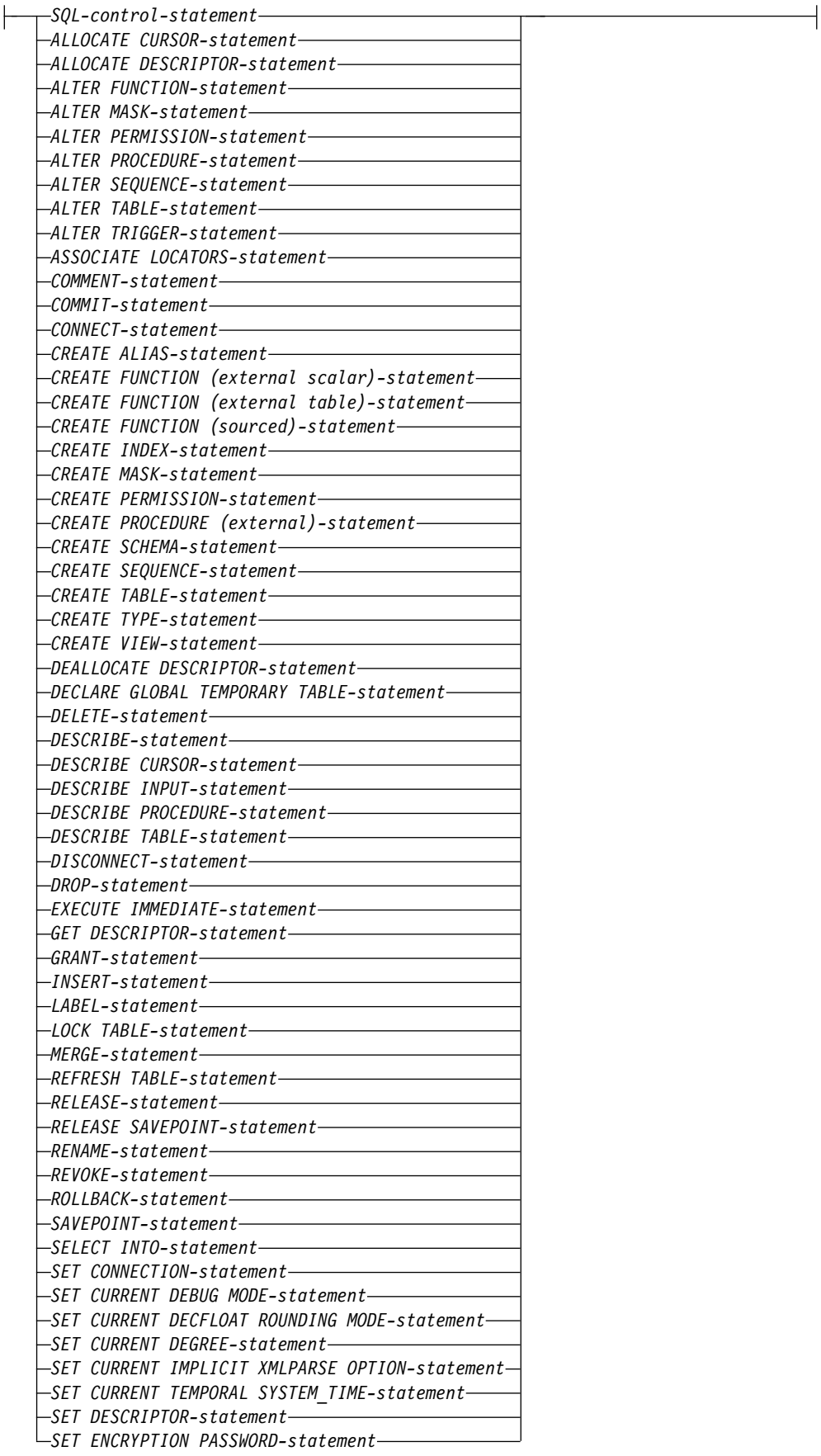
option-list:



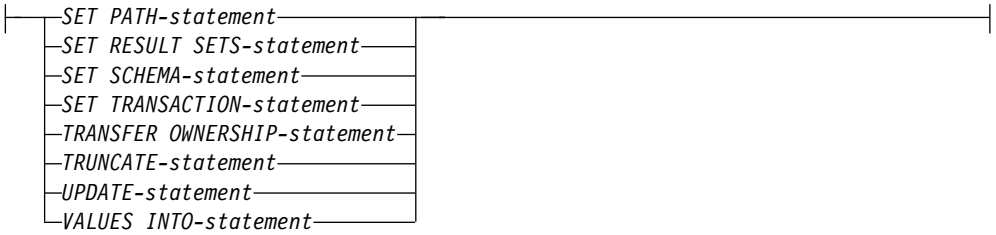
注:

1 オプション文節は、別の順序で指定することができます。

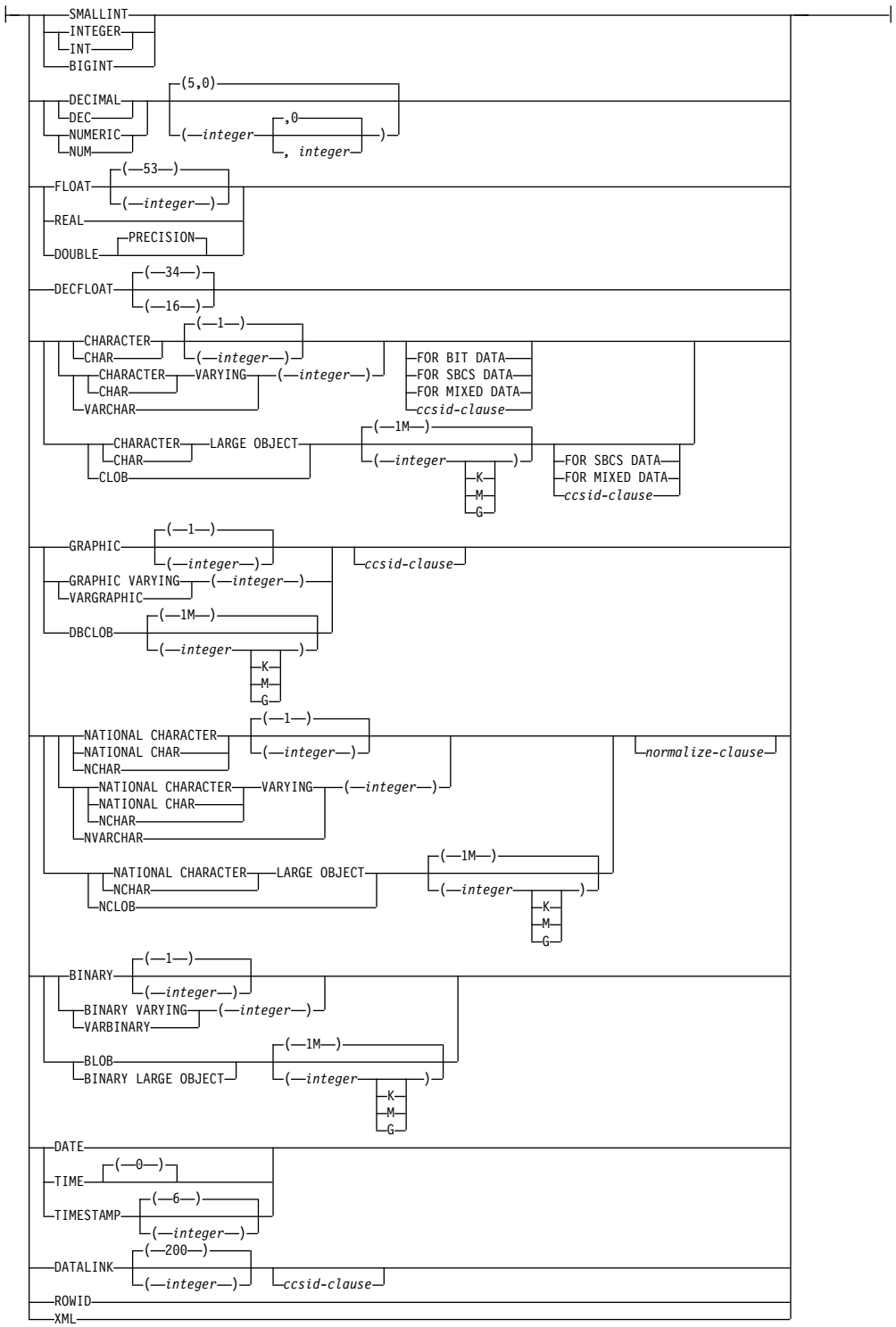
SQL-routine-body:


SQL-control-statement
ALLOCATE CURSOR-statement
ALLOCATE DESCRIPTOR-statement
ALTER FUNCTION-statement
ALTER MASK-statement
ALTER PERMISSION-statement
ALTER PROCEDURE-statement
ALTER SEQUENCE-statement
ALTER TABLE-statement
ALTER TRIGGER-statement
ASSOCIATE LOCATORS-statement
COMMENT-statement
COMMIT-statement
CONNECT-statement
CREATE ALIAS-statement
CREATE FUNCTION (external scalar)-statement
CREATE FUNCTION (external table)-statement
CREATE FUNCTION (sourced)-statement
CREATE INDEX-statement
CREATE MASK-statement
CREATE PERMISSION-statement
CREATE PROCEDURE (external)-statement
CREATE SCHEMA-statement
CREATE SEQUENCE-statement
CREATE TABLE-statement
CREATE TYPE-statement
CREATE VIEW-statement
DEALLOCATE DESCRIPTOR-statement
DECLARE GLOBAL TEMPORARY TABLE-statement
DELETE-statement
DESCRIBE-statement
DESCRIBE CURSOR-statement
DESCRIBE INPUT-statement
DESCRIBE PROCEDURE-statement
DESCRIBE TABLE-statement
DISCONNECT-statement
DROP-statement
EXECUTE IMMEDIATE-statement
GET DESCRIPTOR-statement
GRANT-statement
INSERT-statement
LABEL-statement
LOCK TABLE-statement
MERGE-statement
REFRESH TABLE-statement
RELEASE-statement
RELEASE SAVEPOINT-statement
RENAME-statement
REVOKE-statement
ROLLBACK-statement
SAVEPOINT-statement
SELECT INTO-statement
SET CONNECTION-statement
SET CURRENT DEBUG MODE-statement
SET CURRENT DECFLOAT ROUNDING MODE-statement
SET CURRENT DEGREE-statement
SET CURRENT IMPLICIT XMLPARSE OPTION-statement
SET CURRENT TEMPORAL SYSTEM_TIME-statement
SET DESCRIPTOR-statement
SET ENCRYPTION PASSWORD-statement

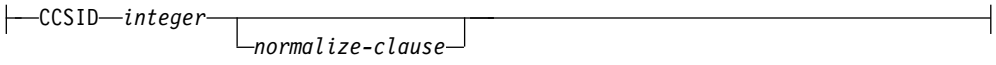
SQL-routine-body (続き):



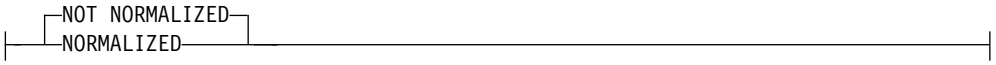
built-in-type:



ccsid-clause:



normalize-clause:



説明

OR REPLACE

現行サーバーにこのプロシージャーの定義が存在する場合に、その定義を置き換える、という動作を指定します。実際には、カタログで既存の定義を削除してから新しい定義に置き換える、という動作になりますが、例外として、このプロシージャーに対して与えられていた特権は影響を受けません。現行サーバーにこのプロシージャーの定義が存在しなければ、このオプションは無視されます。既存のプロシージャーを置き換えるには、新しい定義の *specific-name* および *procedure-name* が古い定義の *specific-name* および *procedure-name* と同じであるか、新しい定義のシグニチャーが古い定義のシグニチャーと一致している必要があります。そうでなければ、新しいプロシージャーが作成されます。

procedure-name

プロシージャーを指定します。名前、スキーマ名、パラメーターの数の組み合わせで、現行サーバーに存在しているプロシージャーを識別してはなりません。

SQL 命名の場合、プロシージャーは、暗黙または明示修飾子で指定されたスキーマ内に作成されます。

システム命名の場合、プロシージャーは、修飾子によって指定されたスキーマ内に作成されます。修飾子を指定しなかった場合:

- CURRENT SCHEMA 特殊レジスターの値が *LIBL である場合、プロシージャーは、現行ライブラリー (*CURLIB) 内に作成されます。
- そうでない場合、プロシージャーは現行スキーマ内に作成されます。

スキーマ名 は、QSYS2、QSYS、QTEMP、または SYSIBM にはできません。

(*parameter-declaration,...***)**

プロシージャーのパラメーターの数とそれぞれのパラメーターのデータ・タイプを指定します。 プロシージャーに関するパラメーターは、入力専用、出力専用、または入出力両用に使用できます。すべてのパラメーターが NULL 可能です。

SQL プロシージャーに許されるパラメーターの最大数は 2000 です。

IN パラメーターが、プロシージャーへの入力パラメーターであることを指定します。プロシージャー内でパラメーターに対する変更が行われても、制御が戻った後で、呼び出し元の SQL アプリケーションがその変更内容を使用することはできません。デフォルトは IN です。

配列タイプ、XML タイプ、または LOB タイプのパラメーターは、読み取り専用です。

OUT

パラメーターが、プロシージャーから戻される出力パラメーターであることを示します。プロシージャー内でこのパラメーターが設定されていない場合は、NULL 値が戻されます。

INOUT
パラメーターが、このプロシージャー用の入出力両方のパラメーターであることを指定します。プロシージャー内でパラメーターが設定されなければ、入力値が戻されます。INOUT パラメーターがデフォルトと共に定義されていて、プロシージャーの呼び出し時にそのデフォルトが使用される場合、パラメーターに戻される値はありません。

parameter-name
パラメーター名を指定します。この名前は、このプロシージャー用の他のパラメーター名 と同じものであってはなりません。

data-type
パラメーターのデータ・タイプを指定します。このデータ・タイプは、組み込みデータ・タイプまたは特殊データ・タイプにすることができます。

built-in-type
組み込みデータ・タイプを指定します。それぞれの組み込みデータ・タイプの詳細については、 1238 ページの『CREATE TABLE』を参照してください。

distinct-type-name
特殊タイプを指定します。パラメーターの長さ、精度、または位取り属性は、特殊タイプのソース・タイプの属性 (CREATE TYPE で指定された属性) と同じになります。特殊タイプの作成についての詳細は、1328 ページの『CREATE TYPE (特殊)』を参照してください。.

特殊タイプの名前が修飾されていない場合、データベース・マネージャーは、SQL パス上のスキーマを検索することでそのスキーマ名を解決します。

配列タイプ名
配列タイプを指定します。

配列タイプの名前が修飾されていない場合、データベース・マネージャーは、SQL パス上のスキーマを検索することでそのスキーマ名を解決します。

CCSID が指定されている場合、プロシージャーに渡される前に、パラメーターはその CCSID に変換されます。CCSID が指定されていない場合は、CCSID は、プロシージャーの呼び出し時点における現行サーバーのデフォルトの CCSID によって決まります。

default
パラメーターのデフォルト値を指定します。デフォルト値は、定数、特殊レジスター、グローバル変数、式、またはキーワード NULL にすることができます。式は、集約関数および列名を含まない、 196 ページの『式』で定義されている任意の式です。デフォルト値が指定されていない場合、パラメーターにデフォルト値がないため、プロシージャーの呼び出し時に省略できません。式ストリングの最大長は 64K です。

デフォルトの式で SQL データを変更することはできません。式は、パラメーターのデータ・タイプに対して割り当ての互換性がなければなりません。デフォルト式の中で参照されるすべてのオブジェクトは、プロシージャーが作成されるときに存在している必要があります。

デフォルト式内でリスト中の数値定数を区切る区切り記号として使用するコンマの後には、スペースが 1 つ必要です。

デフォルトを指定できません。

- OUT パラメーターに対して
- 配列タイプのパラメーターに対して

LANGUAGE SQL
このプロシージャーを SQL だけで記述する、ということを指定します。

SPECIFIC *specific-name*
プロシージャーの固有名を指定します。特定名の詳細については、 1187 ページの『CREATE PROCEDURE』の『プロシージャーに特定の名前を指定する』を参照してください。

DETERMINISTIC または **NOT DETERMINISTIC**
このプロシージャーが、同じ IN 引数および INOUT 引数を指定して呼び出された場合に、常に同じ結果を戻すかどうかを指定します。デフォルトは NOT DETERMINISTIC です。

NOT DETERMINISTIC
このプロシージャーは、同じ IN 引数および INOUT 引数を指定して呼び出された場合に、データベース内の参照先データが変更されていなくても、必ずしも同じ結果を戻すとは限りません。

DETERMINISTIC
このプロシージャーは、同じ IN 引数および INOUT 引数を指定して呼び出された場合に、データベース内の参照先データが変更されていない限り、常に同じ結果を戻します。

MODIFIES SQL DATA、**READS SQL DATA**、または **CONTAINS SQL**
このプロシージャーが実行できる SQL ステートメントおよびネストされたルーチンの種別を指定します。データベース・マネージャーは、プロシージャーと、プロシージャーからローカルで呼び出すすべてのルーチンが発行する SQL ステートメントが、この指定と整合しているかどうかを検査します。ネストされたリモート・ルーチンが呼び出された場合、検査は実行されません。各ステートメントの分類については、 1855 ページの『付録 B. SQL ステートメントの特性』を参照してください。デフォルトは、MODIFIES SQL DATA です。このオプションは、すべてのパラメーター・デフォルト式に適用されます。

MODIFIES SQL DATA
このプロシージャーで、どのプロシージャーでもサポートされないステートメントを除くすべての SQL ステートメントを実行できることを指定します。

READS SQL DATA
このプロシージャーが、データ・アクセス種別 READS SQL DATA、CONTAINS SQL、または NO SQL を指定したステートメントを実行できるように指定します。

CONTAINS SQL
このプロシージャーが、データ・アクセス種別 CONTAINS SQL または NO SQL のステートメントのみを実行できるように指定します。

CALLED ON NULL INPUT
引数値のいずれかまたは全部がヌルである場合にプロシージャーを呼び出すことを指定します。この指定は、ヌル引数値のテストを行うようにプロシージャーをコーディングする必要があります。

INHERIT SPECIAL REGISTERS
特殊レジスターの既存の値は、プロシージャーに入った後に継承されることを示します。

DYNAMIC RESULT SETS *integer*
プロシージャーから戻すことのできる結果セットの最大数を指定します。*integer* の最小値はゼロ、最大値は 32767 です。デフォルトは DYNAMIC RESULT SETS 0 です。

結果セットは、対応するカーソルがオープンした順序で返されます。ただし、プロシージャーで SET RESULT SETS ステートメントを実行する場合は例外です。プロシージャーの終了時に結果セットのためにオープンしたままになっているカーソルの数が、DYNAMIC RESULT SETS 文節で指定した最大数を超えている場合は、CALL ステートメントで警告が返され、DYNAMIC RESULT SETS 文節で指定した数の結果セットが返されます。

SET RESULT SETS ステートメントを発行した場合は、戻される結果の数は、このキーワードに指定した結果セットの数と、 SET RESULT SETS ステートメントに指定した結果セットの数のいずれか少ない方です。 SET RESULT SETS ステートメントに結果セットの最大数よりも大きい値が指定された場合、警告が戻されます。 RETURN TO CLIENT 属性を持つカーソルからの結果セットは、最外部プロシージャーの結果セットの数に含まれます。

結果セットを戻すのにカーソルが使用され、カーソルがスクロール可能である場合、結果セットはスクロール可能です。結果セットを戻すのにカーソルが使用された場合、結果セットはカーソル位置から始まります。つまり、5 つの FETCH NEXT 操作が実行された後、プロシージャーから戻った場合、結果セットは、結果セットの 6 行目から始まります。

結果セットの詳細については、 1724 ページの『SET RESULT SETS』を参照してください。

DISALLOW DEBUG MODE、ALLOW DEBUG MODE、または DISABLE DEBUG MODE
プロシージャーを Unified Debugger でデバッグできるように作成するかどうかを示します。DEBUG MODE を指定する場合は、SET OPTION ステートメント内の DBGVIEW オプションを指定してはなりません。

DISALLOW DEBUG MODE
プロシージャーは Unified Debugger でデバッグできません。プロシージャーの DEBUG MODE 属性が DISALLOW の場合、後でプロシージャーを変更してデバッグ・モード属性を変えることができます。

ALLOW DEBUG MODE
プロシージャーは Unified Debugger でデバッグすることができます。プロシージャーの DEBUG MODE 属性が ALLOW の場合、後でプロシージャーを変更してデバッグ・モード属性を変えることができます。

DISABLE DEBUG MODE
プロシージャーは Unified Debugger でデバッグできません。プロシージ

ャーの DEBUG MODE 属性が DISABLE の場合、後でプロシージャーを変更してデバッグ・モード属性を変えることはできません。

DEBUG MODE を指定せずに SET OPTION ステートメント内の DBGVIEW オプションを指定した場合、プロシージャーを Unified Debugger でデバッグすることはできませんが、システム・デバッグ機能を使用してデバッグできる場合があります。 DEBUG MODE オプションも DBGVIEW オプションも指定しない場合は、CURRENT DEBUG MODE 特殊レジスターでのデバッグ・モードが使用されます。

FENCED または NOT FENCED
このパラメーターは、他のプロダクトとの互換性を保持するために許可されており、Db2 for i で使用されることはありません。

PROGRAM TYPE MAIN または PROGRAM TYPE SUB
プロシージャーをプログラム (*PGM) として作成するか、サービス・プログラム (*SRVPGM) のプロシージャーとして作成するかを指定します。

PROGRAM TYPE MAIN
プロシージャーがプログラム (*PGM) として作成されることを指定します。

PROGRAM TYPE SUB
プロシージャーがサービス・プログラム (*SRVPGM) のプロシージャーとして作成されることを指定します。

通常、PROGRAM TYPE SUB プロシージャーのパフォーマンスは、PROGRAM TYPE MAIN プロシージャーよりも若干優れています。

OLD SAVEPOINT LEVEL または NEW SAVEPOINT LEVEL
このプロシージャーに入ったときに、新しいセーブポイント・レベルを作成するかどうかを指定します。

OLD SAVEPOINT LEVEL
新しいセーブポイント・レベルを作成しません。このプロシージャー内で、OLD SAVEPOINT LEVEL が暗黙的または明示的に指定された SAVEPOINT ステートメントが発行された場合、SAVEPOINT ステートメントはプロシージャーの呼び出し元と同じセーブポイント・レベルで作成されます。これはデフォルトです。

NEW SAVEPOINT LEVEL
このプロシージャーに入ったときに、新しいセーブポイント・レベルが作成されます。プロシージャー内に設定されているすべてのセーブポイントは、このプロシージャーが呼び出されたレベルより深くネストされたセーブポイント・レベルで作成されます。したがって、プロシージャー内のどの新規セーブポイントも、同じ名前を持つ上位のセーブポイント・レベル (例えば呼び出し側プログラムのセーブポイント・レベル) で設定されている既存のセーブポイントと競合することはありません。

COMMIT ON RETURN
データベース・マネージャーが、プロシージャーからの戻りと同時にトランザクションをコミットするかどうかを指定します。

NO データベース・マネージャーは、プロシージャーから戻ったときにコミットを行いません。 NO がデフォルトです。

YES
データベース・マネージャーは、プロシージャーから正常に戻ったときにコミットを行います。プロシージャーの戻り時にエラーがあった場合は、コミットは行われません。

コミット操作の対象には、呼び出し側のアプリケーション・プロセスとこのプロシージャーが行う作業が含まれます。

プロシージャーが結果セットを戻す場合に、結果セットに関連したカーソルをコミット後に使用できるようにするには、カーソルを WITH HOLD として定義しておく必要があります。

AUTONOMOUS
呼び出し側アプリケーションから独立した作業単位でプロシージャーが実行されることを指定します。このオプションが指定されていると、データベースは常にプロシージャーから戻された SQLSTATE に基づいて、自律型プロシージャーのトランザクション作業をコミットまたはロールバックします。SQLSTATE が無条件な成功または警告を示している場合、トランザクションはコミットされます。他のすべての SQLSTATE の場合、自律型プロシージャーの作業単位はロールバックされます。

自律型プロシージャーの内部からの、トリガー、関数、またはプロシージャーの起動は、そのトリガー、関数、またはプロシージャーが、別の活動化グループの下で実行するように明示的に作成されたものでない限り、自律型プロシージャーの作業単位の一部になります。

自律型プロシージャーを別の自律型プロシージャーから直接または間接に呼び出すことはできません。

AUTONOMOUS を指定する場合、DYNAMIC RESULT SETS 0 を指定する必要があります。

CONCURRENT ACCESS RESOLUTION
データベース・マネージャーが更新プロセスの過程にあるデータを待つかどうかを指定します。DEFAULT がデフォルトです。

DEFAULT
このプロシージャーに関する並行アクセスの解決方法を明示的に設定しないことを指定します。このプロシージャーの呼び出し時に有効だった値が使用されます。

WAIT FOR OUTCOME
データベース・マネージャーが更新プロセスの過程にあるデータのコミットまたはロールバックを待つ、という動作を指定します。

USE CURRENTLY COMMITTED
データベース・マネージャーが更新プロセスの過程にあるデータを検出したときに現時点でのコミット済みバージョンのデータを使用する、という動作を指定します。

読み取りトランザクションと削除/更新トランザクションの間でロックの競合が発生した場合に、この文節の適用対象になるのは、分離レベル CS のスキャンです (CS KEEP LOCKS のスキャンは対象になりません)。

SYSTEM_TIME SENSITIVE
静的および動的 SQL ステートメントの両方でのシステム期間テンポラル表への

参照が、CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターの値の影響を受けるかどうかを決定します。YES がデフォルトです。

YES
システム期間テンポラル表への参照は、CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターの値の影響を受けます。

NO システム期間テンポラル表の参照は、CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターの値の影響を受けません。

WRAPPED *obfuscated-statement-text*
エンコードされたプロシージャー定義を指定します。WRAP スカラー関数を使用して CREATE PROCEDURE ステートメントをエンコードできます。

SET OPTION-statement
プロシージャーを作成するときに使用するオプションを指定します。これらのオプションは、すべてのデフォルト値式にも適用されます。 例えば、デバッグ可能なプロシージャーを作成するときは、次のステートメントを含めることができます。

```
SET OPTION DBGVIEW = *SOURCE
```

各オプションのデフォルト値は、作成時に有効だったオプションによって異なります。詳しくは、 1696 ページの『SET OPTION』を参照してください。

オプション
CLOSQLCSR、CNULRQD、CNULIGN、COMPILEOPT、NAMING、SQLCA は、CREATE PROCEDURE ステートメントでは使用できません。デフォルト値式を処理するときには、オプション ALWCPYDTA、
CONACC、DATFMT、DATSEP、DECFLTRND、DECMPT、DECRESULT、DFTRDBCOL、LANGID、
SQLCURRULE、SQLPATH、SRTSEQ、TGTRLS、TIMFMT、および TIMSEP が使用されます。

SQL-routine-body
単一の *SQL-procedure-statement* (複合ステートメントを含む) を指定します。SQL プロシージャーの定義に関する詳細については、 1771 ページの『第 8 章 SQL 制御ステートメント』 を参照してください。

CONNECT、SET CONNECTION、RELEASE、DISCONNECT、および SET TRANSACTION ステートメントは、リモート・アプリケーション・サーバー上で実行中のプロシージャー内で使用することはできません。 COMMIT および ROLLBACK ステートメントは、 ATOMIC SQL プロシージャーまたはリモート・アプリケーション・サーバーへの接続上で実行中のプロシージャー内で使用することはできません。

REPLACE キーワードを指定する ALTER PROCEDURE (SQL)、ALTER FUNCTION (SQL スカラー)、および ALTER FUNCTION (SQL 表) は、*SQL-routine-body* では使用できません。

注

プロシージャー定義に関する一般考慮事項: プロシージャーの定義に関する一般情報については、 1187 ページの『CREATE PROCEDURE』を参照してください。

プロシージャーの所有権: SQL 名が指定されている場合、

- 作成したプロシージャーが入れられるスキーマと同じ名前のユーザー・プロファイルが存在する場合、プロシージャーの所有者 はそのユーザー・プロファイルです。
- その他の場合は、プロシージャーの所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

システム名を指定した場合は、プロシージャーの所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

プロシージャーの権限 : SQL 名を使用する場合は、プロシージャーは、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成されます。システム名を使用する場合、プロシージャーは、スキーマの作成権限 (CRTAUT) パラメーターによって決められる *PUBLIC に対する権限を使用して作成されます。

プロシージャーの所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、そのプロシージャーに対する権限が与えられます。

REPLACE の規則: REPLACE によってプロシージャーを再作成する場合は、以下のようになります。

- 既存のコメントまたはラベルは破棄されます。
- 権限を持つユーザーは維持されます。オブジェクト所有者は変更される可能性があります。
- 現在のジャーナル監査は保持されます。

プロシージャー内のエラー処理: プロシージャー本体のそれぞれの SQL ステートメントごとに起こりうる例外について考慮すべき点があります。例外 SQLSTATE は、複合ステートメント内のハンドラーを使用してプロシージャー内で処理されないと、プロシージャーの呼び出し元に戻されます。プロシージャーによってエラーが戻されると、プロシージャーに渡された引数の値のうち OUT パラメーターに対応するものは未定義になり、INOUT パラメーターに対応するものは変わりません。

プロシージャーの作成: SQL プロシージャーが作成される際、SQL は、組み込み SQL ステートメントと一緒に C ソース・コードが収められる一時ソース・ファイルを作成します。次いで、CRTPGM または CRTSRVPGM コマンドを使用して、プログラムまたはサービス・プログラム・オブジェクトを作成します。プログラムの作成に使用される SQL オプションは、CREATE PROCEDURE ステートメントの実行時に有効なオプションです。AUTONOMOUS が指定されている場合、プログラムまたはサービス・プログラムは ACTGRP(QSQAUTOAG) を使用して作成されます。 それ以外の場合は、プログラムは、ACTGRP(*CALLER) を使用して作成します。

SQL プロシージャーが作成されると、そのプロシージャーの属性は、作成されたプログラム・オブジェクトまたはサービス・プログラム・オブジェクト内に保管され

ます。 *PGM または *SRVPGM オブジェクトが保管された後、このシステムまたは別のシステムに復元されると、属性が使用されてカタログが更新されます。

特定のプロシージャー名は、それが有効なシステム名である場合は、ソース・ファイルのメンバーの名前、ならびに、プログラム・オブジェクトの名前として使用されます。プロシージャー名が有効なシステム名でない場合は、固有名が生成されます。同じ名前のソース・ファイル・メンバーが既に存在している場合は、そのメンバーがオーバーレイされます。同じ名前のモジュールやプログラムが既に存在している場合、オブジェクトはオーバーレイされずに、固有名が生成されます。これらの固有名は、システム表名の生成に関する規則に従って生成されます。

プロシージャーの呼び出し: DECLARE PROCEDURE ステートメントで、作成されたプロシージャーと同じ名前のプロシージャーを定義し、そのプロシージャー名が変数によって識別されていない静的 CALL ステートメントが、同じソース・プログラムから実行される場合は、CREATE PROCEDURE ステートメントの属性ではなく、 DECLARE PROCEDURE ステートメントの属性が使用されます。

CREATE PROCEDURE ステートメントが適用されるのは、静的および動的 CALL ステートメント、ならびにそのプロシージャー名が変数によって識別されている CALL ステートメントです。

SQL プロシージャーは、SQL CALL ステートメントを使用して呼び出す必要があります。 SQL プロシージャーは、呼び出されると、呼び出し側プログラムの活動化グループ内で実行します。

SQL プロシージャーは *TERASPACE ストレージ・モデルで作成されます。自律型 SQL プロシージャーがジョブ内で起動される場合、QSQAUTOAG 活動化グループ内で実行するすべてのプロシージャーは *TERASPACE ストレージ・モデルを使用しなければなりません。

難読化されたステートメント: CREATE PROCEDURE ステートメントは、難読化された形式で実行できます。難読化されたステートメントでは、 WRAPPED キーワードの前にあるプロシージャー名とパラメーターのみが判読可能です。ステートメントの残りは、判読できないが、難読化されたステートメントをサポートするデータベース・サーバーによってデコード可能なように、エンコードされます。難読化されたステートメントは、WRAP スカラー関数を呼び出すことによって生成できます。難読化されたステートメントからプロシージャーが作成されるときに指定されるデバッグ・オプションはすべて無視されます。難読化されたステートメントから作成されたプロシージャーを難読化がサポートされていないリリースにリストアすることはできません。

デフォルト値の設定: プロシージャーのパラメーターがデフォルト値を指定して定義されていれば、そのプロシージャーの呼び出し時にパラメーターはデフォルト値に設定されますが、それは、対応する引数に値が与えられていない場合か、引数が DEFAULT と指定されている場合に限ります。

従属オブジェクト: SQL ルーチンは、*SQL-routine-body* で参照されるオブジェクトに従属します。従属オブジェクトの名前は、カタログ・ビュー SYSROUTINEDEP に保管されます。 *SQL-routine-body* のオブジェクト参照が完全修飾名である場合、または非修飾名が現行スキーマによって SQL 命名で修飾されている場合、

SYSROUTINEDEP 内のオブジェクトのスキーマ名は、指定された名前か現行スキーマの値に設定されます。それ以外の場合は、スキーマ名は、特定のスキーマ名に設定されません。非修飾の関数名、変数名、およびタイプ名のスキーマ名は CURRENT PATH になります。名前を実際のスキーマ名に設定しないと、DROP ステートメントおよび ALTER ステートメントは、ルーチンが変更中または除去中のオブジェクトに従属しているかどうかを判別することができません。

代替構文: 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード VARIANT と NOT VARIANT は、NOT DETERMINISTIC と DETERMINISTIC の同義語として使用することができます。
- キーワード NULL CALL は、CALLED ON NULL INPUT の同義語として使用できます。
- DYNAMIC RESULT SET、RESULT SETS、および RESULT SET は、DYNAMIC RESULT SETS の同義語として使用できます。

例

社員の給与の中央値を戻す SQL プロシージャーを作成します。給与の中央値を超える給与を得ている全社員の氏名、肩書き、および給与の入った結果セットを戻します。

```
CREATE PROCEDURE MEDIAN_RESULT_SET (OUT medianSalary DECIMAL(7,2))
     LANGUAGE SQL
     DYNAMIC RESULT SETS 1
BEGIN
     DECLARE v_numRecords INTEGER DEFAULT 1;
     DECLARE v_counter INTEGER DEFAULT 0;
     DECLARE c1 CURSOR FOR
        SELECT salary
           FROM staff
           ORDER BY salary;
     DECLARE c2 CURSOR WITH RETURN FOR
        SELECT name, job, salary
           FROM staff
           WHERE salary > medianSalary
           ORDER BY salary;
     DECLARE EXIT HANDLER FOR NOT FOUND
        SET medianSalary = 6666;
     SET medianSalary = 0;
     SELECT COUNT(*) INTO v_numRecords FROM STAFF;
     OPEN c1;
     WHILE v_counter < (v_numRecords / 2 + 1)
        DO FETCH c1 INTO medianSalary;
        SET v_counter = v_counter + 1;
     END WHILE;
     CLOSE c1;
     OPEN c2;
END
```

CREATE SCHEMA

CREATE SCHEMA ステートメントは、現行サーバーにスキーマを定義し、オプションとして、表、ビュー、別名、索引、シーケンス、および特殊タイプを作成します。 これらのオブジェクトのカタログ記述内にコメントおよびラベルを加えることができ、ユーザーに特権を付与することができます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 次の CL コマンドに対する *USE システム権限
  - ライブラリー作成 (CRTLIB)
  - WITH DATA DICTIONARY を指定する場合は、データ・ディクショナリー作成 (CRTDTADCT)
- データベース管理者権限

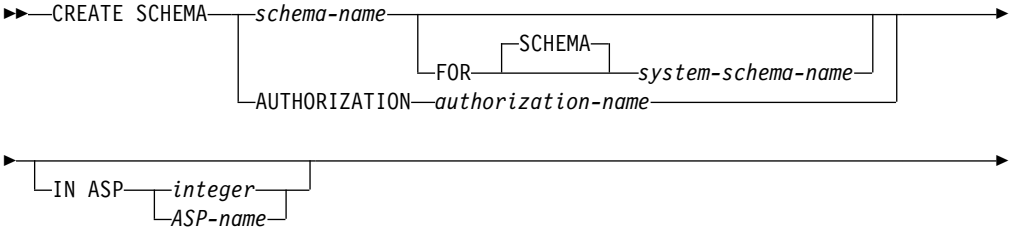
このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

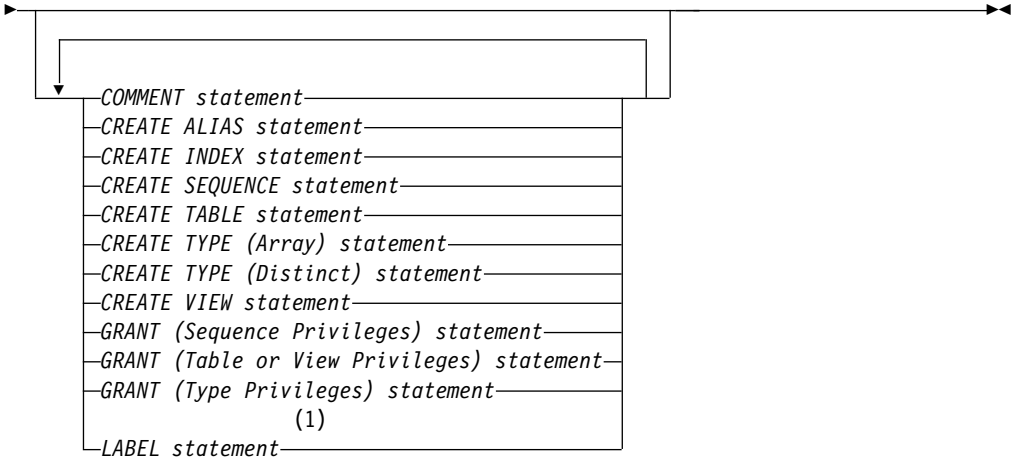
- CREATE SCHEMA ステートメントに含まれている各 SQL ステートメントについて定義されている特権
- データベース管理者権限

AUTHORIZATION 文節の指定がある場合、そのステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- 権限名によって識別されるユーザー・プロファイルに対する *ADD システム権限
- データベース管理者権限

構文





注:

1 CREATE SCHEMA ステートメント内では、スキーマ内に作成されるオブジェクト・タイプに対するラベルおよびコメントのみがサポートされます。

説明

schema-name

スキーマの名前を指定します。スキーマは、この名前で作成されます。スキーマ名を指定すると、このステートメントの権限 ID は、実行時権限 ID になります。この名前は、現行サーバーにある既存のスキーマの名前と同じであってはなりません。この名前を「SYS」または「Q」で始めることはできません。これらのスキーマ名は、スキーマがシステム・スキーマ であることを示します。

schema-name が有効なシステム名ではなく、*system-schema-name* を指定しないと、SQL はシステム名を生成します。名前の生成に関する規則については、1228 ページの『スキーマ名の生成規則』を参照してください。

FOR SCHEMA *system-schema-name*

スキーマのシステム名を示します。*system-schema-name* は、現行サーバーの既存のスキーマと同じにすることはできません。 *system-schema-name* は、有効なシステム名である非修飾システム ID でなければなりません。

schema-name および *system-schema-name* をどちらも指定すると、両方を有効なシステム名にすることはできません。

authorization-name

ステートメントの権限 ID を示します。この権限名は、スキーマ名でもあります。この名前は、現行サーバーにある既存のスキーマの名前と同じであってはなりません。

IN ASP

スキーマを作成したい補助記憶域プール (ASP) を指定します。

integer

1 から 32 までの整数を指定します。 1 を指定した場合、スキーマはシステム ASP に作成されます。

ASP-name
この名前は、現行サーバーに存在する補助記憶域プールを示すものでなければなりません。

この文節を省略すると、

- 呼び出しているスレッドの名前空間が独立補助記憶域プール (IASP) に設定されると、スキーマは IASP 名前空間に作成されます。
- それ以外の場合、1 の ASP が想定されます。

COMMENT ステートメント
スキーマ内に作成されるオブジェクトのカタログ記述内のコメントを追加または置換します。スキーマ内に作成されるオブジェクトに対するコメントのみが許可されます。 1027 ページの『COMMENT』の COMMENT ステートメントの節を参照してください。

CREATE ALIAS ステートメント
別名を作成してスキーマ内に入れます。 1065 ページの『CREATE ALIAS』の CREATE ALIAS ステートメントの節を参照してください。

CREATE INDEX ステートメント
索引を作成してスキーマ内に入れます。 1166 ページの『CREATE INDEX』の CREATE INDEX ステートメントの節を参照してください。

CREATE SEQUENCE ステートメント
シーケンスを作成してスキーマ内に入れます。 1230 ページの『CREATE SEQUENCE』の CREATE SEQUENCE ステートメントの節を参照してください。

CREATE TABLE ステートメント
表を作成してスキーマ内に入れます。 1238 ページの『CREATE TABLE』の CREATE TABLE ステートメントの節を参照してください。

CREATE TYPE (配列) ステートメント
配列タイプを作成してスキーマ内に入れます。 1323 ページの『CREATE TYPE (配列)』の CREATE TYPE (配列) ステートメントの節を参照してください。

CREATE TYPE (特殊) ステートメント
ユーザー定義の特殊タイプを作成してスキーマ内に入れます。 1328 ページの『CREATE TYPE (特殊)』の CREATE TYPE (特殊) ステートメントの節を参照してください。

CREATE VIEW ステートメント
ビューを作成してスキーマ内に入れます。 1342 ページの『CREATE VIEW』の CREATE VIEW ステートメントの節を参照してください。

GRANT (シーケンス特権) ステートメント
スキーマ内のシーケンスに対する特権を与えます。 1532 ページの『GRANT (シーケンス特権)』の GRANT ステートメントの節を参照してください。

GRANT (表またはビュー特権) ステートメント
このスキーマの表およびビューに対する特権を与えます。 1535 ページの『GRANT (表またはビューの特権)』の GRANT ステートメントの節を参照してください。

GRANT (タイプ特権) ステートメント
スキーマ内のタイプに対する特権を与えます。 1542 ページの『GRANT (タイプ特権)』の GRANT ステートメントの節を参照してください。

LABEL ステートメント
スキーマ内に作成されるオブジェクトのカタログ記述内のラベルを追加または置換します。スキーマ内に作成されるオブジェクトに対するラベルのみが許可されます。 1570 ページの『LABEL』の LABEL ステートメントの項を参照してください。

注

スキーマの属性 : スキーマは以下のものとして作成されます。

- ライブラリー: ライブラリーは、関連オブジェクトをグループ化し、名前でオブジェクトを見つけることができるようにします。
- カタログ: カタログには、そのスキーマ内の表、ビュー、索引、および他のオブジェクトの記述が含まれます。カタログは、一連のビューで構成されます。詳しくは、「SQL プログラミング」を参照してください。
- ジャーナルおよびジャーナル・レシーバー: ジャーナル QSQJRN とジャーナル・レシーバー QSQJRN0001 がスキーマ内に作成され、以後にスキーマ内に作成されるすべての表への変更を記録するのに使用されます。詳しくは、「ジャーナル管理」を参照してください。

分散表に対して作成される索引は、この表が配布されるサーバーのすべてで作成されます。分散表についての詳細は、Db2 マルチシステム (Multisystem) を参照してください。

オブジェクトの所有権 : スキーマおよび作成したオブジェクトの所有者は以下のように決定されます。

- AUTHORIZATION 文節が指定されている場合は、指定された権限 ID が、このステートメントによって作成されたスキーマおよびすべてのオブジェクトを所有します。
- これら以外の場合は、このステートメントによって作成されるスキーマおよびすべてのオブジェクトの所有者 は、このステートメントを実行するスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

オブジェクトの権限 : SQL 名を使用する場合は、スキーマおよびその他のオブジェクトは、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成され、作成権限パラメーター CRTAUT(*EXCLUDE) を指定してライブラリーが作成されます。スキーマに関する権限を持つユーザーは、スキーマの所有者だけです。他のユーザーがそのスキーマに対する権限を必要とする場合、スキーマの所有者は、CL コマンドのオブジェクト権限付与 (GRTOBJAUT) を使用して、作成したオブジェクトに対する権限を与えることができます。

システム名を使用する場合は、スキーマおよびその他のオブジェクトの作成時に *PUBLIC に与えられるシステム権限は、システム値 QCRTAUT によって決まり、ライブラリーは CRTAUT(*SYSVAL) を指定して作成されます。システム・セキュリティーについて詳しくは、機密保護解説書、および SQL プログラミングを参照してください。

スキーマの所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、そのスキーマに対する権限が与えられます。

オブジェクト名**:** CREATE TABLE、CREATE INDEX、CREATE ALIAS、CREATE TYPE、CREATE SEQUENCE、または CREATE VIEW ステートメントに、作成中の表、索引、別名、特殊タイプ、シーケンス、またはビューの修飾名を指定する場合は、その修飾名の中のスキーマ名には、作成中のスキーマと同一の名前を指定する必要があります。スキーマ定義で参照されるその他のオブジェクト名は、どのようなスキーマ名で修飾されていても構いません。どの SQL ステートメントにおいても、非修飾の表、索引、別名、特殊タイプ、シーケンス、またはビューの名前は、作成されるスキーマの名前で暗黙的に修飾されます。

SQL ステートメント相互間には、区切り記号を使用しません。

SQL ステートメント長**:** RUNSQLSTM コマンドによって CREATE SCHEMA ステートメントが実行される場合、CREATE SCHEMA ステートメント内の個々の CREATE TABLE、CREATE INDEX、CREATE TYPE、CREATE ALIAS、CREATE SEQUENCE、CREATE VIEW、COMMENT、LABEL、または GRANT ステートメントの最大長は、どれも 2 097 152 です。それ以外の場合は、CREATE SCHEMA ステートメント全体として 2 097 152 に制限されます。

ネーム解決のパフォーマンス**:** スキーマの名前は、そのスキーマ内のオブジェクトを参照するステートメントのパフォーマンスに影響を及ぼす可能性があります。スキーマ名の長さが 30 を超えると、スキーマ内のオブジェクト検索のパフォーマンスは、スキーマ名の長さが 30 以内の場合に比べて劣るようになります。パフォーマンス上の影響を最小限にとどめるには、システム名とスキーマ名の最初の 5 文字を同じにしてください。

代替構文 **:** 旧リリースとの互換性を維持するために、 SCHEMA の同義語として COLLECTION キーワードを使用できます。これは標準キーワードではないので、使用しないようにしてください。

推奨されない機能**:** WITH DATA DICTIONARY 文節を使用すると、スキーマ内に IDDU データ・ディクショナリーが作成されます。この文節は現在もサポートされており、CREATE SCHEMA ステートメントの終わりで指定することはできますが、推奨されません。

データ・ディクショナリーを指定して作成されたスキーマには、LOB 列、XML 列、または DATALINK 列を含む表を入れることはできません。この文節は、カタログ・ビューの作成には効果がありません。

スキーマ名の生成規則

10 文字を超える名前のスキーマが作成されると、システム名が生成されます。

SQL 名、または対応するシステム名はいずれも、SQL ステートメントで使用して、作成済みの該当スキーマのアクセスに用いることができます。ただし、SQL 名を識別するのは、Db2 for i だけであり、他の環境では、システム名を使用する必要があります。

schema-name が 10 文字を超える通常 ID の場合、10 文字からなる *system-schema-name* が以下のように生成されます。

- その名前の最初の 5 文字
- 5 桁の固有の番号

例えば、次のようになります。

LONGSCHEMANAME の system-schema-name は LONGS00001 となります。

schema-name が 10 文字を超える区切り文字付き ID の場合、10 文字からなる *system-schema-name* が以下のように生成されます。

- 区切り文字の最初の 4 文字が *system-schema-name* の最初の 4 文字として使用されます。
- 最初の 4 文字がすべて大文字、数字、または下線の場合、下線と 5 桁の数値が付加されます。
- その他の場合、4 桁の固有な数値が付加されます。

例えば、次のようになります。

「longschemaname」のシステム名は「long0001」となります。
「LONGSchemaName」のシステム名は「LONG_00001」となります。

例

例 *1:* 在庫部品表と部品番号に関する索引を持つスキーマを作成します。ユーザー・プロファイル JONES に、スキーマに対する権限を与えています。

```
CREATE SCHEMA INVENTORY

CREATE TABLE PART (PARTNO SMALLINT NOT NULL,
                   DESCR  VARCHAR(24),
                   QUANTITY INT)

CREATE INDEX PARTIND ON PART (PARTNO)

GRANT ALL ON PART TO JONES
```

例 *2:* SMITH の権限 ID を使用してスキーマを作成します。学生番号列に、コメントを付け、学生表を作成します。

```
CREATE SCHEMA AUTHORIZATION SMITH

CREATE TABLE SMITH.STUDENT (STUDNBR SMALLINT NOT NULL UNIQUE,
                            LASTNAME CHAR(20),
                            FIRSTNAME CHAR(20),
                            ADDRESS CHAR(50))

COMMENT ON STUDENT (STUDNBR IS 'THIS IS A UNIQUE ID#')
```

CREATE SEQUENCE

CREATE SEQUENCE ステートメントはアプリケーション・サーバーでシーケンスを作成します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- スキーマ内に作成する特権。詳しくは、スキーマ内で作成する必要のある権限を参照してください。
- データベース管理者権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 次のシステム権限
  - データ域作成 (CRTDTAARA) コマンドに対する *USE 権限
- データベース管理者権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- SYSSEQOBJECTS カタログ表の場合
  - 該当の表に対する INSERT 特権、および
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

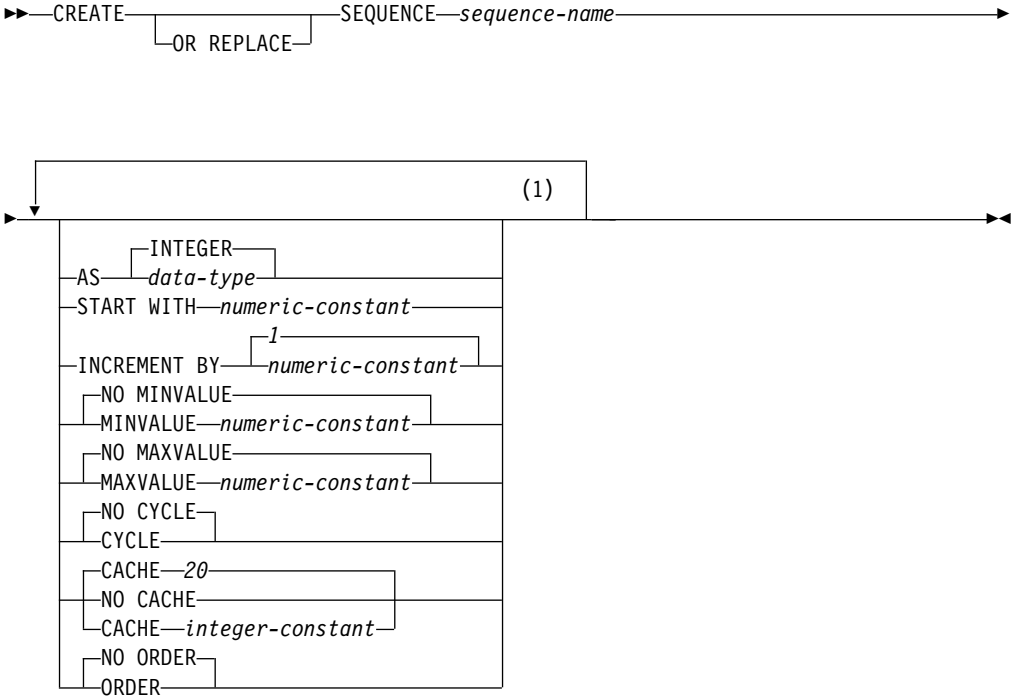
- ステートメント内で識別される特殊タイプに対しては次のもの。
  - その特殊タイプに対する USAGE 特権、および
  - 特殊タイプが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

既存のシーケンスに置き換えるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

- 次のシステム権限
  - シーケンスに関連したデータ域に対する *OBJMGT システム権限
  - このシーケンスを削除するために必要な全権限
  - SYSSEQOBJECTS カタログ表に対する *READ システム権限
- データベース管理者権限

SQL 特権に対応するシステム権限の説明については、『シーケンスへの権限を検査する際の対応するシステム権限』および 『特殊タイプへの権限を検査する際の対応するシステム権限』を参照してください。

構文

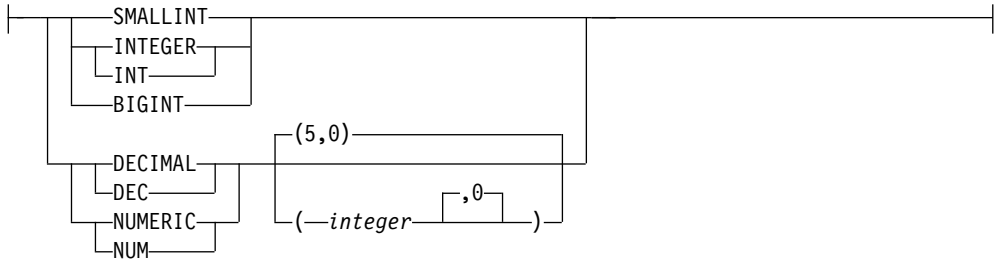


注:

1 同じ文節を複数回指定することはできません。

data-type:

built-in-type:



説明

OR REPLACE
現行サーバーにこのシーケンスの定義が存在する場合に、その定義を置き換え

る、という動作を指定します。実際には、カタログで既存の定義を削除してから新しい定義に置き換える、という動作になりますが、例外として、このシーケンスに対して与えられていた特権は影響を受けません。現行サーバーにこのシーケンスの定義が存在しなければ、このオプションは無視されます。

sequence-name
シーケンスを指定します。 暗黙的または明示的修飾子も含め、この名前で、現行サーバーに既に存在しているシーケンスまたはデータ域を識別することはできません。修飾されたシーケンス名を指定する場合、スキーマ名 は、QSYS2、QSYS、または SYSIBM にはできません。

SQL 名が指定されている場合、シーケンスは、暗黙的または明示的修飾子で指定しているスキーマ内に作成されます。

システム名が指定されている場合、シーケンスは、修飾子で指定しているスキーマ内に作成されます。修飾されない場合:

- CURRENT SCHEMA 特殊レジスターの値が *LIBL である場合、シーケンスは、現行ライブラリー (*CURLIB) 内に作成されます。
- そうでない場合、シーケンスは現行スキーマ内に作成されます。

AS*data-type*
シーケンス値に使用するデータ・タイプを指定します。データ・タイプは、厳密に位取りがゼロの数値タイプ (SMALLINT、INTEGER、BIGINT、DECIMAL、または NUMERIC)、またはソース・タイプが厳密に位取りがゼロの数値タイプであるユーザー定義の特殊タイプにすることができます。デフォルトは INTEGER です。

built-in-type
シーケンスの内部表示のベースとして使用される組み込みデータ・タイプを指定します。データ・タイプが DECIMAL または NUMERIC である場合、精度は 63 以下、位取りは 0 でなければなりません。各組み込みデータ・タイプについての詳細は、 1238 ページの『CREATE TABLE』を参照してください。

プラットフォーム間でのアプリケーションの移植性を保つには、NUMERIC データ・タイプの代わりに DECIMAL を使用します。

distinct-type-name
シーケンスのデータ・タイプが、特殊タイプ (ユーザー定義のデータ・タイプ) であることを指定します。ソース・タイプが DECIMAL または NUMERIC である場合、シーケンスの精度は該当する特殊タイプのソース・タイプの精度になります。ソース・タイプの精度は 63 以下、また位取りは 0 でなければなりません。スキーマ名なしの特殊タイプを指定すると、その特殊タイプ名は、SQL パス上のスキーマを検索することで解決されます。

START WITH *numeric-constant*
シーケンスについて生成される最初の値を指定します。小数点の右側にゼロ以外の数字がないことを条件として、シーケンスに関連したデータ・タイプの列に割り当てることのできる任意の正または負の値を指定できます。

シーケンスを定義するときに値を明示的に指定していない場合、デフォルト値は、昇順の場合は MINVALUE で降順の場合は MAXVALUE です。

この値は、シーケンスが最大値または最小値に達した後で、シーケンスの循環により到達する値になるとは限りません。 START WITH 文節を使用することに

より、この循環に使用される値の範囲外の値からシーケンスを開始することができます。循環に使用する範囲は、MINVALUE および MAXVALUE で定義します。

INCREMENT BY*numeric-constant*
シーケンスの連続した値の間隔を指定します。小数点の右側にゼロ以外の数字がないことと、長精度整数定数の値を超えないことを条件として、シーケンスに関連したデータ・タイプの列に割り当てることのできる任意の正または負の値を指定できます。

この値が 0 または正の場合は、昇順になります。この値が負の場合は、降順になります。デフォルトは、1 です。

NO MINVALUE または MINVALUE
降順シーケンスが値の生成を循環または停止する最小値、あるいは最大値に達した後、昇順シーケンスが循環する最小値を指定します。デフォルトは NO MINVALUE です。

NO MINVALUE
昇順シーケンスの場合、値は START WITH 値であり、 START WITH が指定されていない場合は 1 です。降順シーケンスの場合、シーケンスに関連するデータ・タイプ (および精度 (DECIMAL または NUMERIC の場合)) の最小値です。

MINVALUE*numeric-constant*
最小値を示す数値定数を指定します。小数点の右側にゼロ以外の数字がないことを条件として、シーケンスに関連したデータ・タイプの列に割り当てることのできる任意の正または負の値を指定できます。値は、最大値またはそれより小さい値でなければなりません。

NO MAXVALUE または MAXVALUE
昇順シーケンスが値の生成を循環または停止する最大値、あるいは最小値に達した後、降順シーケンスが循環する最大値を指定します。デフォルトは NO MAXVALUE です。

NO MAXVALUE
昇順シーケンスの場合、シーケンスに関連するデータ・タイプ (および精度 (DECIMAL または NUMERIC の場合)) の最大値です。降順シーケンスの場合、値は START WITH 値であり、START WITH が指定されていない場合は -1 です。

MAXVALUE *numeric-constant*
最大値を示す数値定数を指定します。小数点の右側にゼロ以外の数字がないことを条件として、シーケンスに関連したデータ・タイプの列に割り当てることのできる任意の正または負の値を指定できます。値は、最小値またはそれより大きい値でなければなりません。

NO CYCLE または CYCLE
シーケンスの最大値または最小値に達した後も、このシーケンスで値を生成し続けるかどうかを指定します。デフォルトは NO CYCLE です。

NO CYCLE
シーケンスの最大値または最小値に達した後は、このシーケンスについて値を生成しないことを指定します。

CYCLE
最大値または最小値に達した後も、このシーケンスの値を生成し続けることを指定します。このオプションを使用した場合は、昇順シーケンスがシーケンスの最大値に達した後では、最小値が生成されます。降順シーケンスがシーケンスの最小値に達した後は、最大値が生成されます。列の最大値と最小値によって、循環に使用される範囲が決まります。

CYCLE が有効になっていると、シーケンスで重複値が生成される可能性があります。

CACHE または NO CACHE
事前割り振りの値をメモリー内に保持するかどうかを指定します。値を事前に割り振ってキャッシュに保管しておくと、NEXT VALUE シーケンス式のパフォーマンスが向上します。デフォルトは CACHE 20 です。

CACHE *integer-constant*
事前割り振りされてメモリーに保持されるシーケンス値の最大数を指定します。値を事前割り振りしてキャッシュに保管することにより、パフォーマンスが向上します。

システム障害のような特定の状態になると、キャッシュに保管されていてコミット済みステートメントでまだ使用されていないシーケンス値はすべて失われるため、その後使用されることはありません。 CACHE オプションに指定する値は、こうした状態で失われる可能性のあるシーケンス値の最大数です。

指定できる最小の値は 2 です。

NO CACHE
シーケンスの値を事前割り振りしないことを指定します。 NO CACHE を指定すると、 CACHE を指定した場合よりも NEXT VALUE シーケンス式のパフォーマンスが低下します。

ORDER または NO ORDER
シーケンス値を要求された順序で生成するかどうかを指定します。デフォルトは NO ORDER です。

NO ORDER
シーケンス番号を要求の順序どおりに生成する必要がないことを指定します。

ORDER
シーケンス番号を要求の順序どおりに生成するように指定します。ORDER を指定すると、 NO ORDER を指定した場合よりも NEXT VALUE シーケンス式のパフォーマンスが低下します。

注

シーケンス属性: シーケンスが *DTAARA オブジェクトとして作成されます。SQL を通して SQL シーケンスを使用した場合に予期しない失敗または予期しない結果が生じる可能性があるため、 Change Data Area (*CHGDTAARA) または他の同様のインターフェースを使用して *DTAARA オブジェクトを変更しないでください。

シーケンス所有権**:** シーケンスの所有者 は、ステートメントを実行するスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

シーケンスの権限 **:** SQL 名を使用する場合は、シーケンスは、 *PUBLIC に対するシステム権限 *EXCLUDE を使用して作成されます。システム名を使用する場合、シーケンスは、スキーマの作成権限 (CRTAUT) パラメーターによって決められる *PUBLIC に対する権限を使用して作成されます。

シーケンスの所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、そのシーケンスに対する権限が与えられます。

REPLACE の規則**:** REPLACE によってシーケンスを再作成する場合は、以下のようになります。

- 既存のコメントまたはラベルは破棄されます。
- 権限を持つユーザーは維持されます。オブジェクト所有者は変更される可能性があります。
- 現在のジャーナル監査は保持されます。

MINVALUE と **MAXVALUE** の関係**:** 通常、MINVALUE は MAXVALUE より小さくなりますが、これは必須ではありません。 MINVALUE が MAXVALUE と等しくなる場合もあります。 START WITH が MINVALUE および MAXVALUE と同じ値になっていて、CYCLE が暗黙的または明示的に指定された場合、これは定数シーケンスになります。この場合、シーケンスによって生成されるすべての値は実際は同じであるため、次の値を要求することは何の効果もありません。

MINVALUE は MAXVALUE 以下でなければなりません。

定数シーケンスの定義**:** 常に定数値を返すシーケンスを定義することも可能です。これは、INCREMENT 値にゼロを指定して START WITH 値には MAXVALUE を超えない値を指定するか、あるいは START WITH、MINVALUE、および MAXVALUE に同じ値を指定することによって実行できます。定数シーケンスの場合には、シーケンスに関する NEXT VALUE が呼び出されるたびに、同じ値が戻ります。定数シーケンスは、数値グローバル変数として使用できます。ALTER SEQUENCE を使用すると、定数シーケンスのために生成される値を調整することができます。

循環するシーケンスの定義**:** ALTER SEQUENCE ステートメントを使用して、シーケンスを手動で循環させることができます。NO CYCLE が暗黙的または明示的に指定されている場合、 ALTER SEQUENCE ステートメントでシーケンスを再始動または拡張し、そのシーケンスの最大または最小値に達した後でも値の生成を続行できます。

CYCLE キーワードを指定して、シーケンスが循環するように明示的に指定できます。シーケンスを定義する際に CYCLE オプションを使用して、生成された値が境界に達するたびに循環するよう指示します。シーケンスが自動的に循環するように定義されると (つまり CYCLE が明示的に指定された場合)、増分値が 1 または -1 以外の場合には、シーケンスに対して生成される最大または最小値は、実際に指定された MAXVALUE または MINVALUE ではない可能性があります。例えば、

START WITH=1, INCREMENT=2, MAXVALUE=10 と定義されたシーケンスは、最大値 9 を生成し、値 10 は生成しないはずです。

シーケンスに CYCLE を定義する際、 (アプリケーションを他のベンダー・プラットフォームから Db2 に変換するための) アプリケーション変換ツールは、MINVALUE、MAXVALUE および START WITH も明示的に指定する必要があります。

シーケンス番号のキャッシュ**:** シーケンス番号の範囲を高速アクセスのためにメモリーに保管できます。アプリケーションが、次のシーケンス番号をキャッシュから割り振ることができるシーケンスにアクセスすると、シーケンス番号の割り振りは素早く行われます。ただし、次のシーケンス番号をキャッシュから割り振ることができないシーケンスにアクセスする場合、シーケンス番号の割り振りは、*DTAARA オブジェクトの更新を必要とします。

CACHE に高い値を選択することによって、連続したシーケンス番号へのより高速なアクセスが許可されます。ただし、失敗した場合、キャッシュ内のすべてのシーケンス値が失われます。 NO CACHE オプションを使用する場合、シーケンスの値はシーケンス・キャッシュに保管されません。この場合、シーケンスにアクセスするたびに *DTAARA オブジェクトの更新が必要になります。 CACHE の値を選択する場合、パフォーマンス要件とアプリケーション要件との間のトレードオフを考慮に入れる必要があります。

最後に生成されたシーケンス値の持続性**:** データベース・マネージャーは、 SQL セッション内で最後に生成されたシーケンスの値を覚えていて、この値を PREVIOUS VALUE 式に対して戻し、シーケンス名を指定します。この値は、シーケンス用に次の値が生成されるまで、またはシーケンスが除去、変更、あるいは置き換えられるまで、またはアプリケーション・セッションの終わりまで持続します。この値は COMMIT ステートメントおよび ROLLBACK ステートメントの影響を受けません。

PREVIOUS VALUE を定義して、アプリケーション・セッション内でリニアな有効範囲を持つようにします。このため、ネストされたアプリケーションでは、

- ネストされた関数、プロシージャー、またはトリガーへの入り口で、ネストされたアプリケーションは、シーケンスに対して最後に生成された値を継承します。つまり、ネストされたアプリケーションで PREVIOUS VALUE 式の呼び出しを指定した場合、ネストされたアプリケーションに入る前に、呼び出されるアプリケーション、ルーチン、またはトリガーで実行されるシーケンス活動が反映されます。ネストされたアプリケーションで PREVIOUS VALUE 式を呼び出した場合、指定されたシーケンスの NEXT VALUE 式が、呼び出されるアプリケーション、ルーチン、またはトリガーで実行されていないと、エラーが生じます。
- 関数、プロシージャー、またはトリガーから戻る際に、関数内のシーケンス・アクティビティーは、呼び出されるアプリケーション、ルーチン、またはトリガーに影響します。つまり、ネストされたアプリケーションから戻された後に、呼び出されるアプリケーション、ルーチン、またはトリガーで PREVIOUS VALUE 式の呼び出しを指定した場合、低いレベルのアプリケーションで発生したシーケンス・アクティビティーが反映されます。

シーケンスのジャーナリング**:** シーケンスの作成時に、ジャーナリングを自動的に開始することができます。

- QDFTJRN と呼ばれるデータ域がシーケンスが作成されたのと同一のスキーマに存在し、ユーザーがそのデータ域に対する権限を持っていると、以下のすべてが当てはまる場合、ジャーナリングがデータ域で指定されたジャーナルに対して開始されます。
  - 表の識別されたスキーマは、QSYS、QSYS2、QRECOVERY、QSPL、QRCL、QRPLOBJ、QGPL、QTEMP、SYSIBM、またはこれらのライブラリーと同等の IASP であってはなりません。
  - データ域で指定されたジャーナルが存在していなければならず、ユーザーはジャーナルに対してジャーナリングを開始する権限を持っていなければなりません。
  - データ域の最初の 10 バイトには、ジャーナルを検索するスキーマの名前が含まれている必要があります。
  - 次の 10 バイトにはジャーナルの名前が含まれている必要があります。
  - 残りのバイトには、暗黙的にジャーナルに記録されるオブジェクト・タイプと、いつ暗黙的ジャーナリングを行うかに関係のあるオプションが含まれます。オブジェクト・タイプには値 *DTAARA または *ALL を含める必要があります。値 *NONE を使用して、ジャーナリングが開始しないようにすることができます。

  詳細については、「ジャーナル管理」のトピック集を参照してください。
- シーケンスを指定した (STRJRNLIB コマンドを使用して) スキーマ内に作成すると、ジャーナリングが暗黙的に開始されます。

代替構文**:** 以下のキーワードは、他の Db2 製品の旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード NOMINVALUE、NOMAXVALUE、NOCYCLE、NOCACHE、および NOORDER を、 NO MINVALUE、NO MAXVALUE、NO CYCLE、NO CACHE、および NO ORDER の同義語として使用することができます。
- コンマは、複数のシーケンス・オプションを分離するのに使用できます。

例

1 で始まり、1 つずつ増分し、循環しない、同時に 24 の値をキャッシュに入れる ORG_SEQ というシーケンスを作成します。

```
CREATE SEQUENCE ORG_SEQ
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO CYCLE
  CACHE 24
```

START WITH 1、INCREMENT 1、NO MAXVALUE、NO CYCLE の各オプションは、それぞれ明示的に指定しない場合に使用される値です。

CREATE TABLE

CREATE TABLE ステートメントは、現行サーバーで表を定義します。この定義には、その表の名前、およびその表の列の名前と属性を含める必要があります。この定義には、基本キーなど、表の他の属性も含めることができます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- スキーマ内に作成する特権。詳しくは、スキーマ内で作成する必要のある権限を参照してください。
- データベース管理者権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 次のシステム権限
  - 物理ファイル作成 (CRTPF) コマンドに対する *USE 権限
  - データ・ディクショナリー に対する *CHANGE 権限。ただし、表が作成されるライブラリーが、データ・ディクショナリーを持つ SQL スキーマの場合。
- データベース管理者権限

SQL 名が指定され、該当の表が作成されるライブラリーの名前と同じ名前のユーザー・プロファイルが存在し、しかもその名前がステートメントの権限 ID と異なっている場合、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- その名前を持つユーザー・プロファイルに対する *ADD システム権限
- データベース管理者権限

外部キーを定義する場合、ステートメントの権限 ID が保持する特権には、親表に関して少なくとも次の 1 つが含まれていなければなりません。

- 該当の表に対する REFERENCES 特権またはオブジェクト管理権限。
- 指定された親キーの各列に対する REFERENCES 特権。
- その表の所有権
- データベース管理者権限

フィールド・プロシージャーを定義する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- 次のシステム権限
  - プログラムに対する *EXECUTE システム権限、および

  - プログラムが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

LIKE 文節または選択ステートメント を指定する場合は、このステートメントの権限 ID が保持する特権には、これらの文節で指定する表またはビューに対する次の特権の少なくとも 1 つが含まれていなければなりません。

- 表またはビューに対する SELECT 特権
- 表またはビューの所有権
- データベース管理者権限

特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別された、それぞれの特殊タイプごとに、
  - その特殊タイプに対する USAGE 特権、および
  - 特殊タイプが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

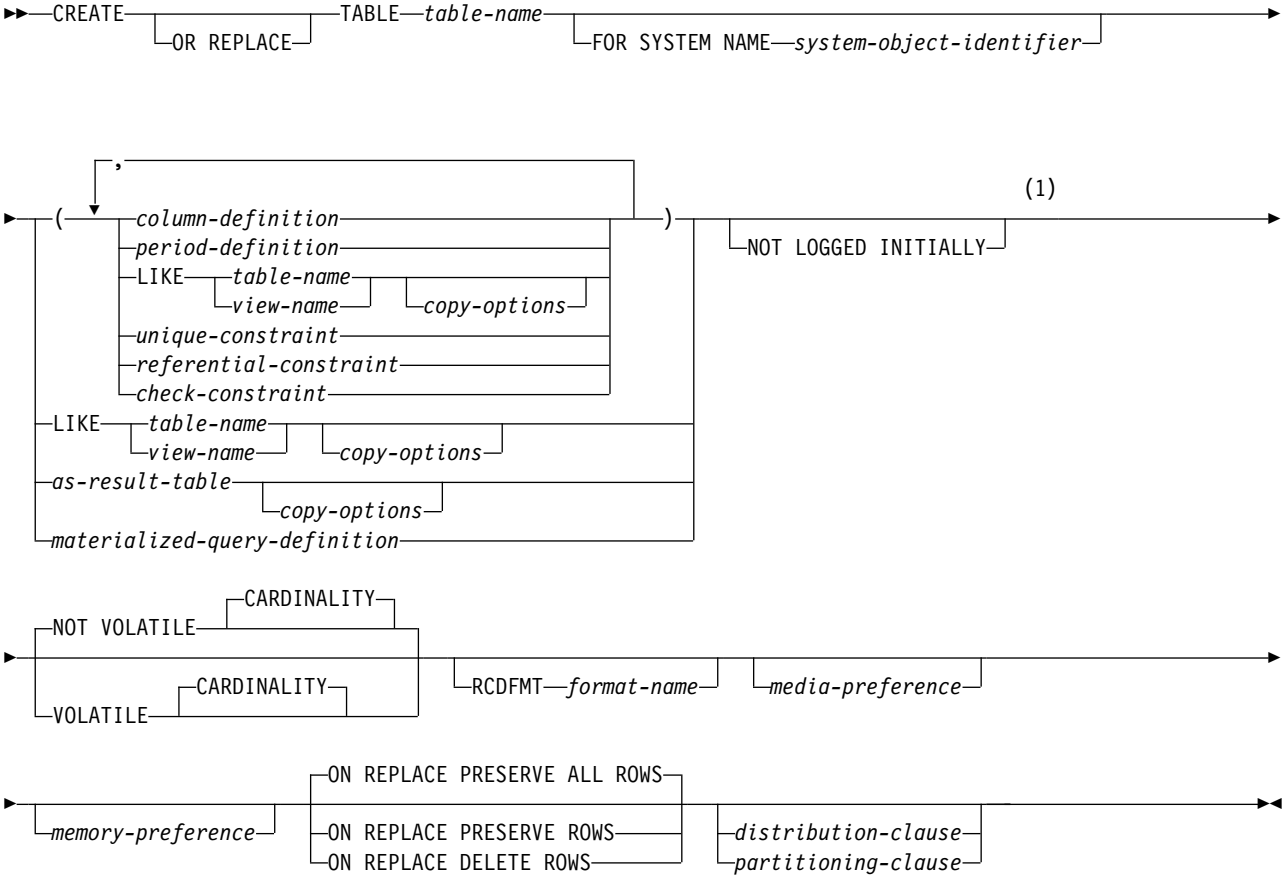
既存の表に置き換えるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

- 次のシステム権限
  - 表に対する *OBJMGT のシステム権限
  - 表を削除するために必要な全権限
- データベース管理者権限

SQL 特権に対応するシステム権限については、『表またはビューへの権限を検査する際の対応するシステム権限』および『特殊タイプへの権限を検査する際の対応するシステム権限』を参照してください。

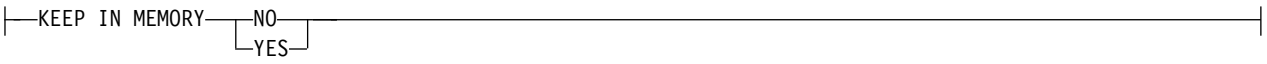
CREATE TABLE

構文



media-preference:

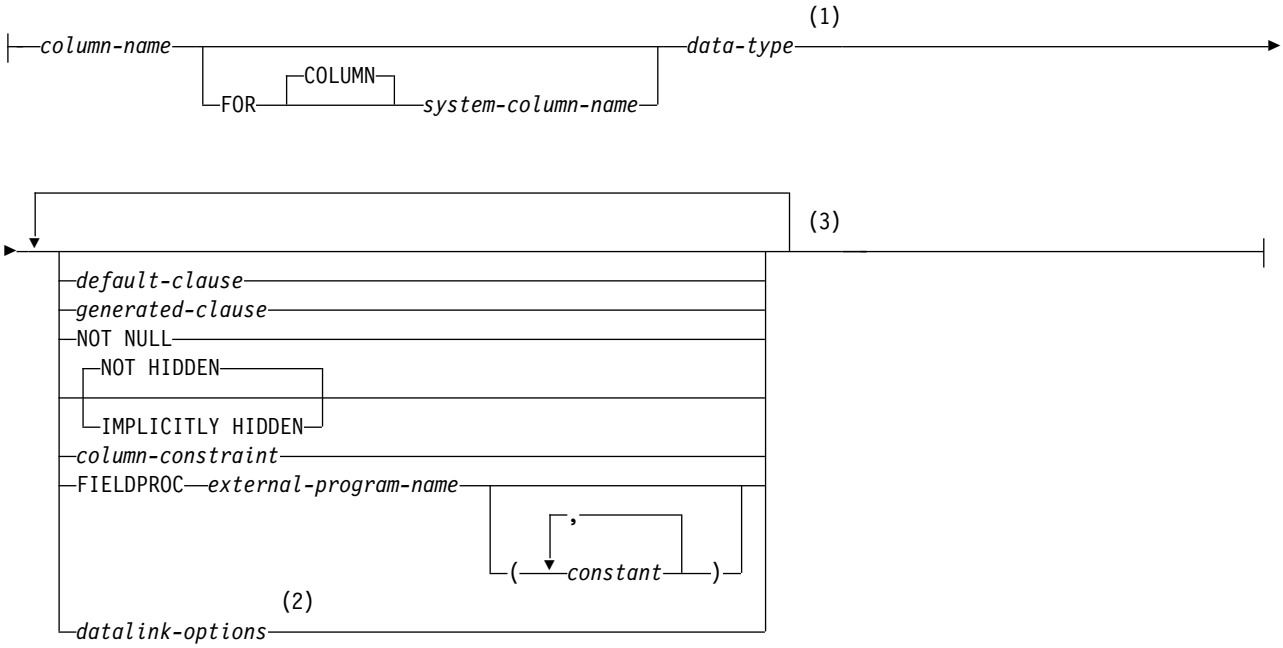
memory-preference:



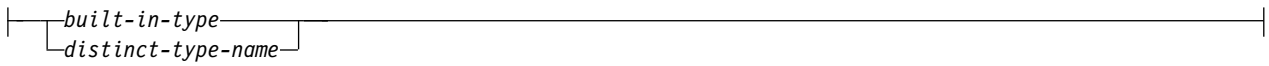
注:

1 オプション文節は、どのような順序で指定しても構いません。

column-definition:



data-type:



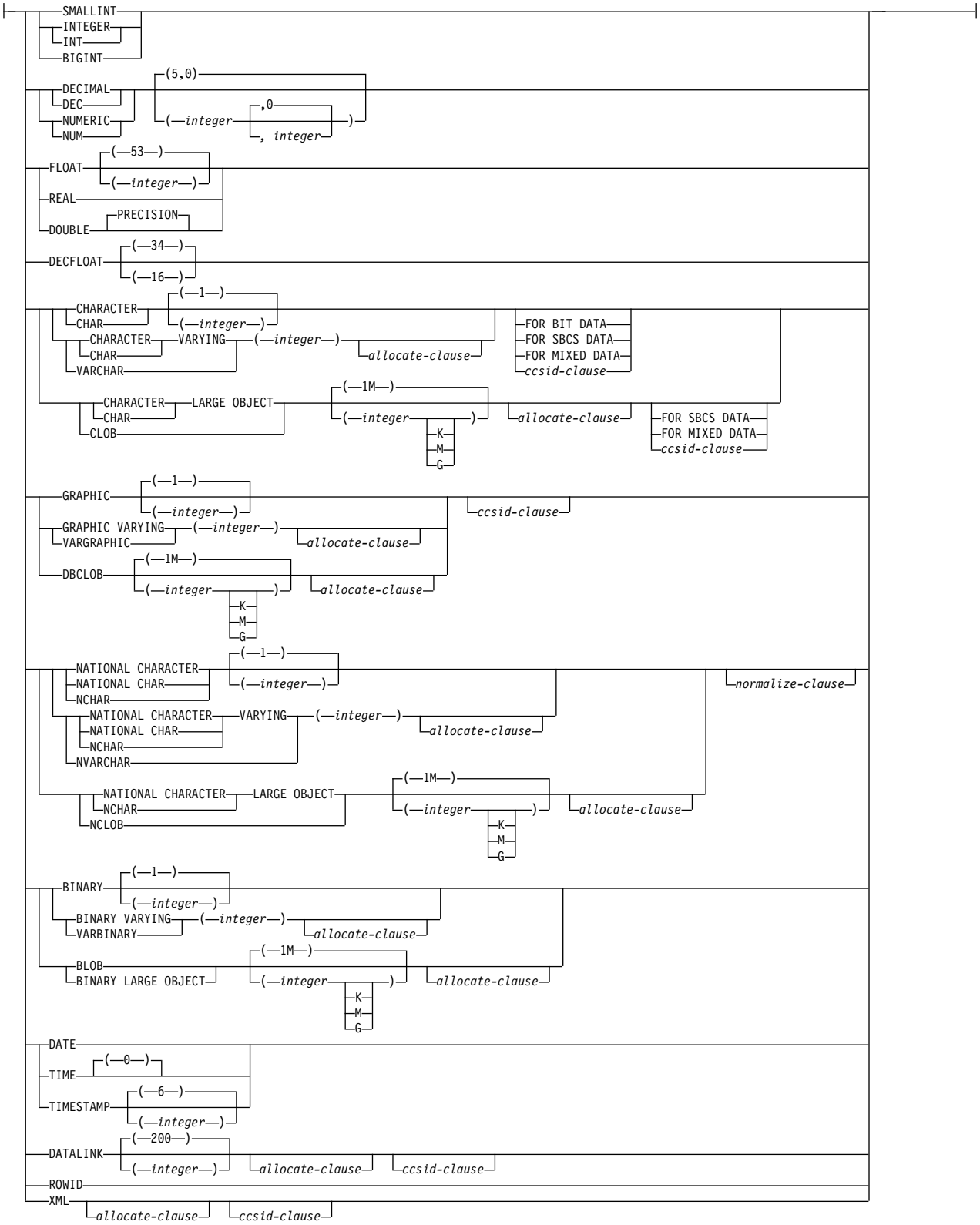
注:

1 *data-type* は、行変更タイム・スタンプ列、行開始および行終了タイム・スタンプ列、およびトランザクション開始 ID タイム・スタンプ列にはオプションです。

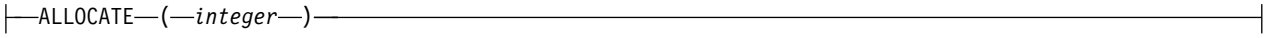
2 データ・リンク・オプション は、DATALINK、および DATALINK をソースとする特殊タイプに対してのみ指定することができます。

3 同じ文節を複数回指定することはできません。

built-in-type:



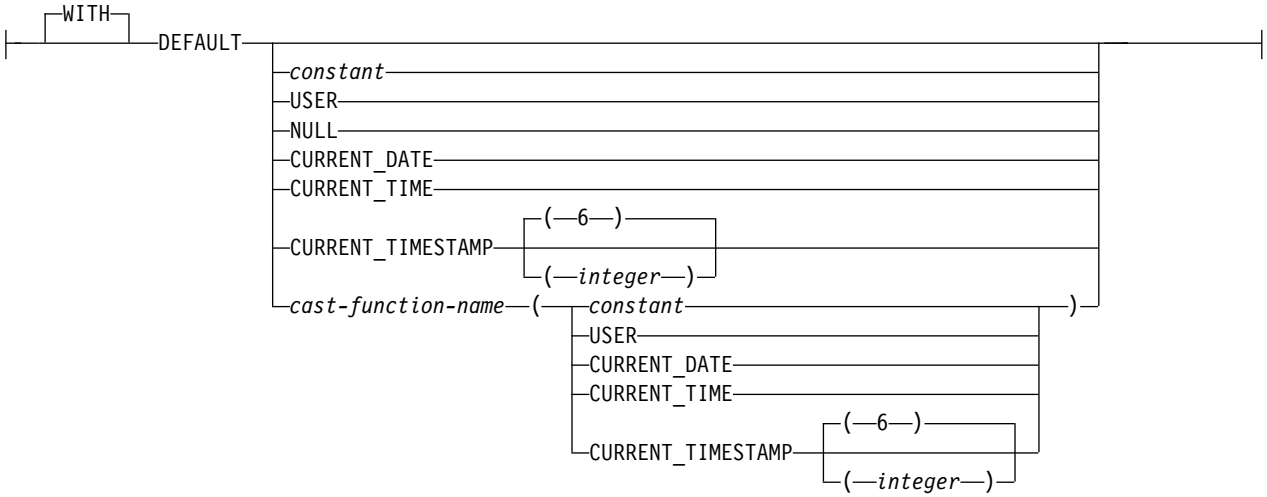
allocate-clause:



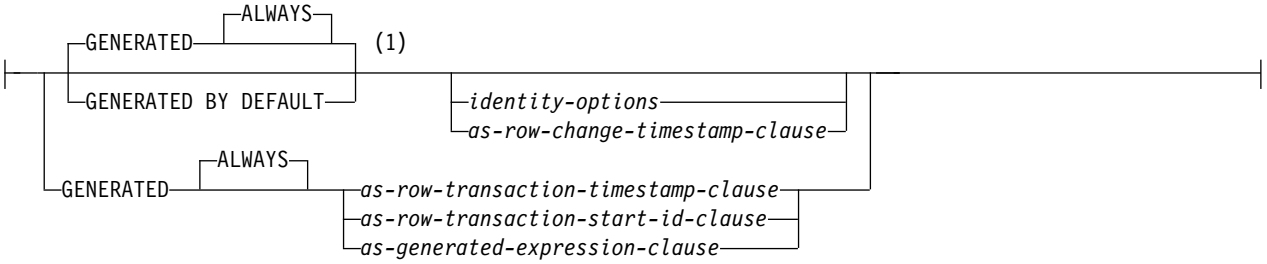
ccsid-clause:

normalize-clause:

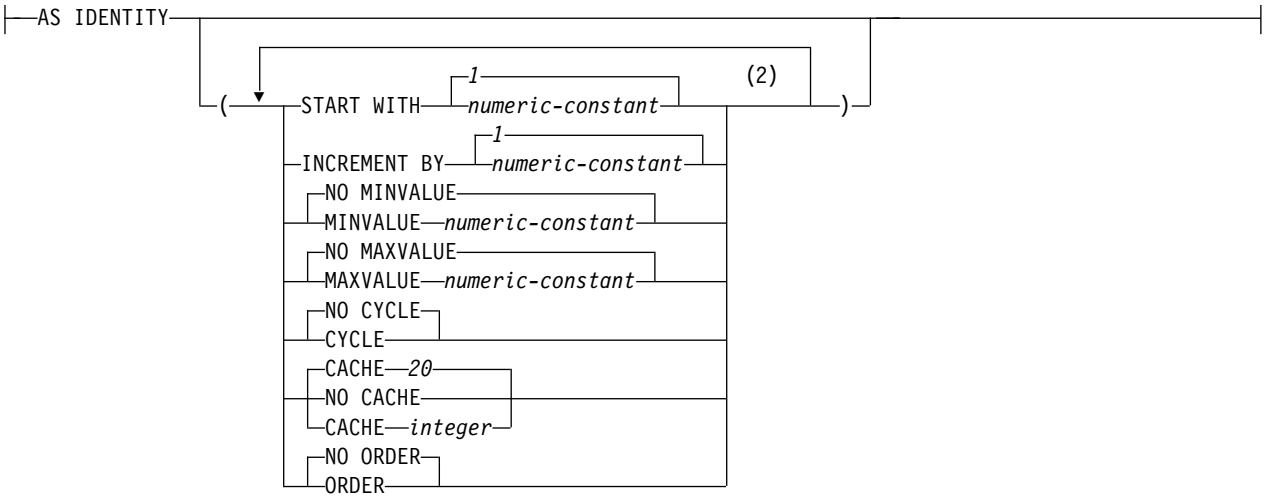
default-clause:



generated-clause:



identity-options:



as-row-change-timestamp-clause:

as-row-transaction-timestamp-clause:

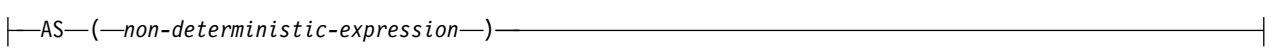
as-row-transaction-start-id-clause:

注:

1 GENERATED を指定できるのは、列のデータ・タイプが ROWID (または ROWID データ・タイプに基づく特殊タイプ) であるか、列が ID 列であるか、または列が行変更タイム・スタンプである場合のみです。

2 同じ文節を複数回指定することはできません。

as-generated-expression-clause:

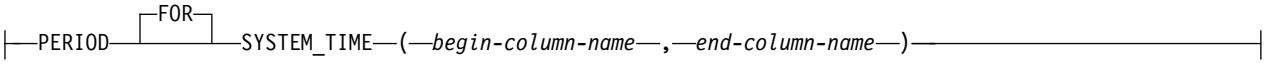


non-deterministic-expression:

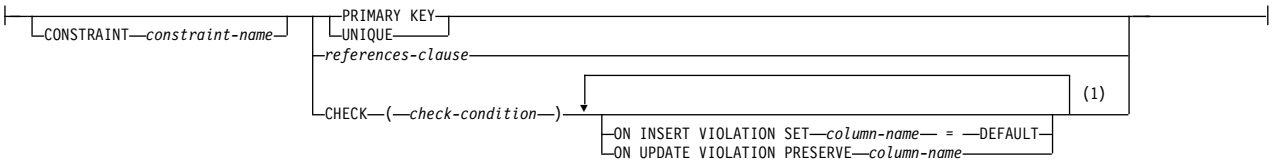
special-register:

built-in-global-variable:

period-definition:



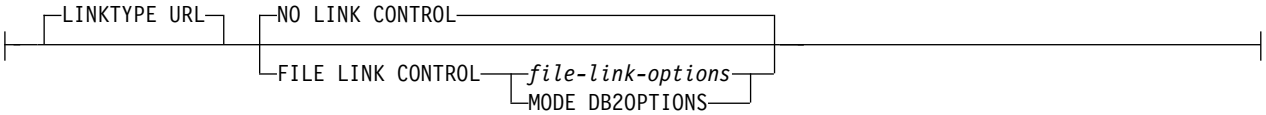
column-constraint:



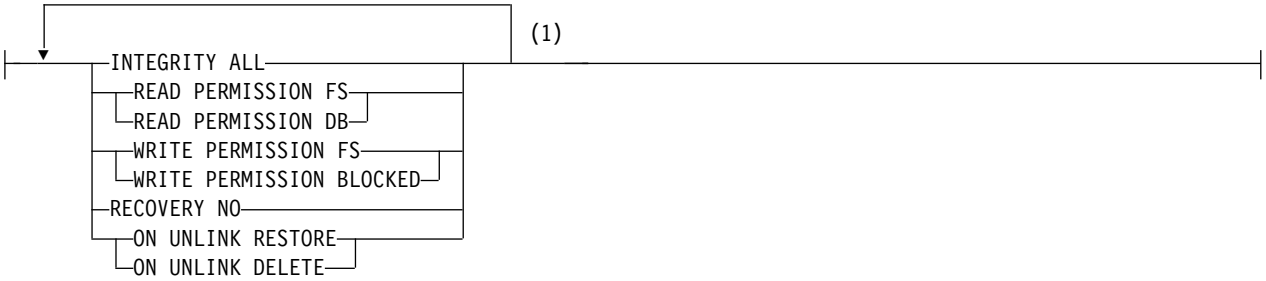
注:

1 同じ文節を複数回指定することはできません。

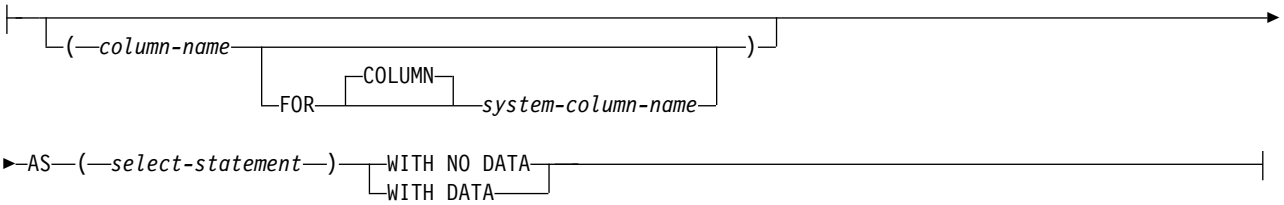
datalink-options:



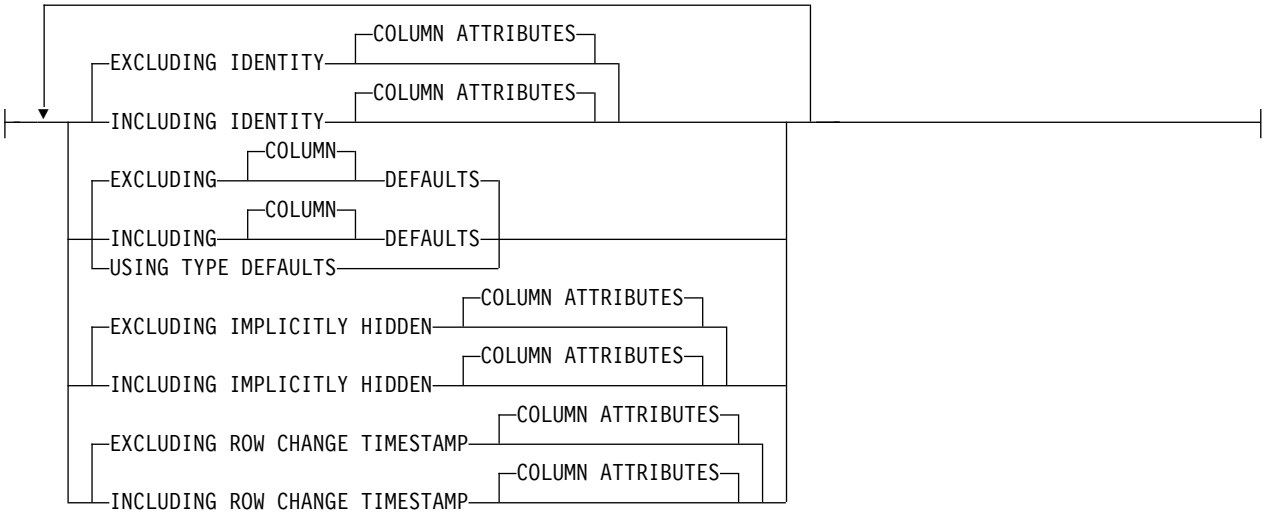
file-link-options:



as-result-table:



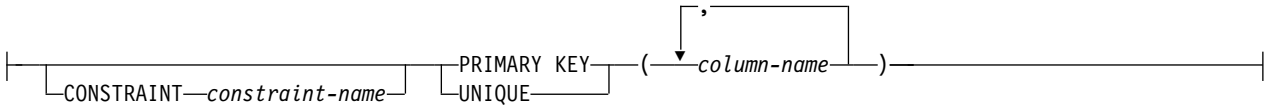
copy-options:



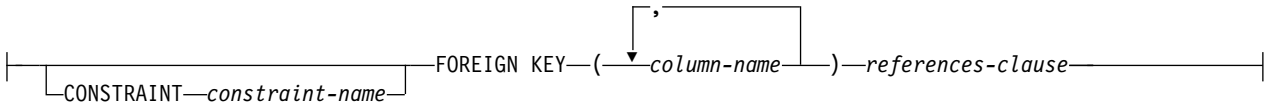
注:

1 5 つのファイル・リンク・オプション をすべて指定する必要がありますが、指定する順序は任意です。

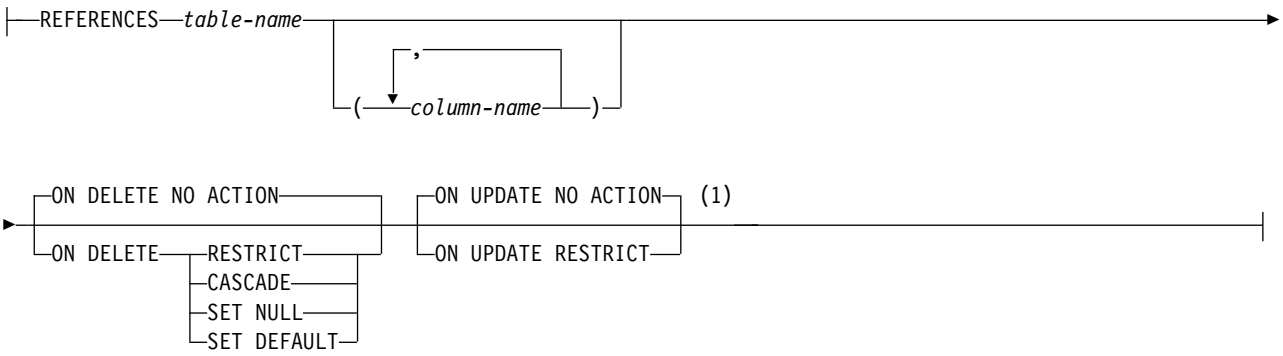
unique-constraint:



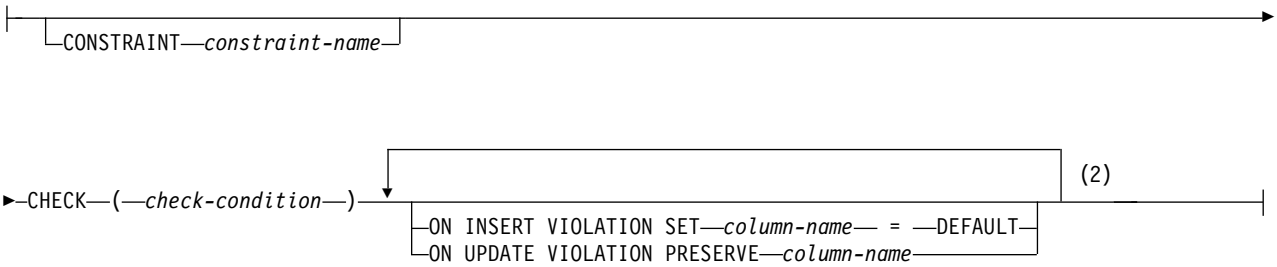
referential-constraint:



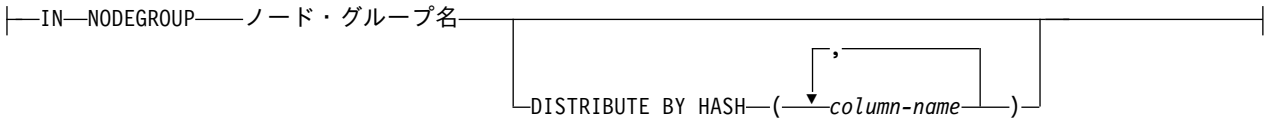
references-clause:



check-constraint:



distribution-clause:

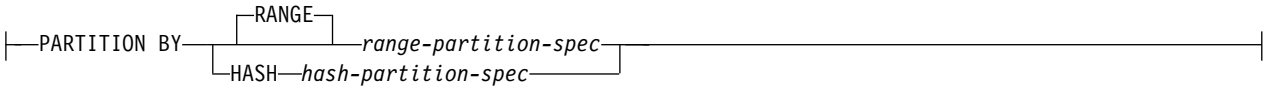


注:

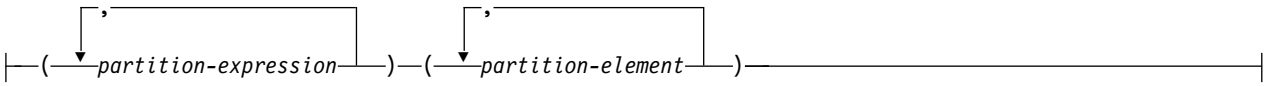
1 ON DELETE と ON UPDATE 文節は、どの順序で指定しても構いません。

2 同じ文節を複数回指定することはできません。

partitioning-clause:

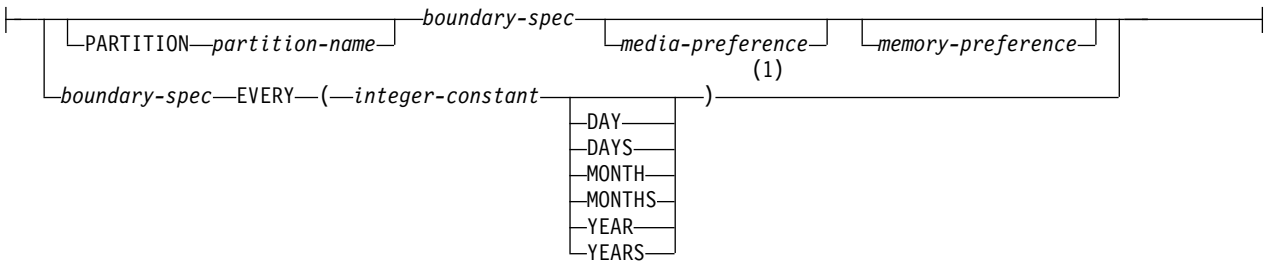


range-partition-spec:



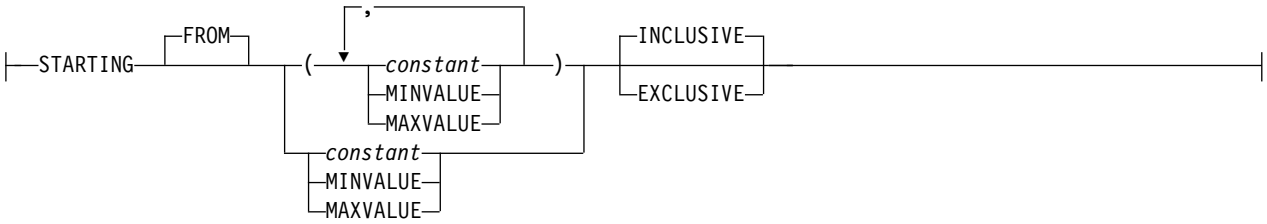
partition-expression:

partition-element:



boundary-spec:

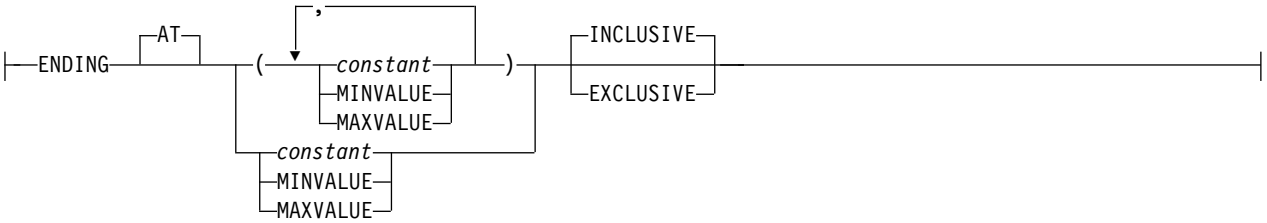
starting-clause:



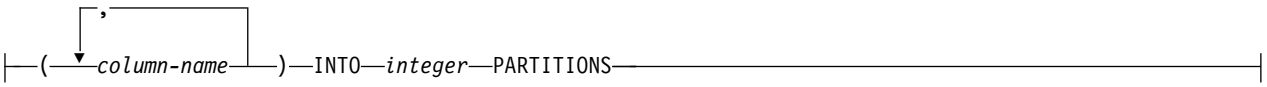
注:

1 *partition-element* のこの構文が有効なのは、数値データ・タイプまたは日時データ・タイプの *partition-expression* が 1 つだけ存在する場合です。

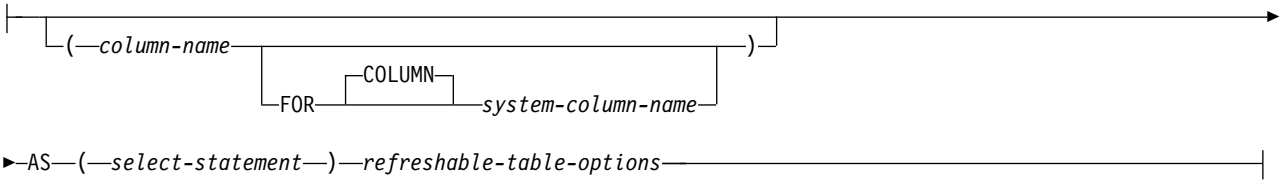
ending-clause:



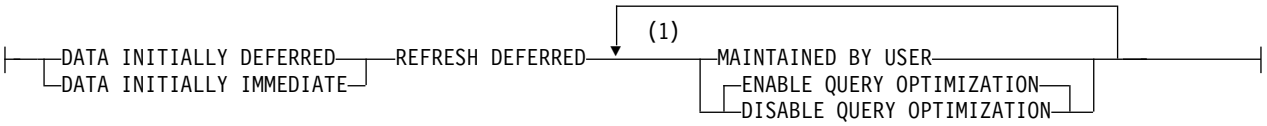
hash-partition-spec:



materialized-query-definition:



refreshable-table-options:



注:

1 同じ文節を複数回指定することはできません。MAINTAINED BY USER を指定しなければなりません。

説明

OR REPLACE

表が現行サーバーに存在する場合に、その変数の定義を置き換えるように指定します。カタログ内で新しい定義が置き換えられる前に、既存の定義が実際に変更されます。

表の定義は、以下の場合に存在します。

- FOR SYSTEM NAME が指定されていて、*system-object-identifier* が既存の表の *system-object-identifier* と一致する。
- FOR SYSTEM NAME が指定されておらず、*table-name* が既存の表の *system-object-identifier* と一致するシステム・オブジェクト名である。

表の定義が存在していて、*table-name* がシステム・オブジェクト名でない場合は、*table-name* を変更して表に新しい名前を付けることができます。

このオプションは、表の定義が現行サーバーに存在しない場合には無視されます。

table-name
表の名前を指定します。 暗黙的または明示的修飾子も含め、この名前で、現行サーバーに既に存在している別名、ファイル、索引、表、またはビューを識別することはできません。

SQL 名が指定されている場合、表は、暗黙的または明示的修飾子で指定しているスキーマ内に作成されます。

システム名が指定されている場合、表名は、修飾子で指定しているスキーマ内に作成されます。修飾されない場合:

- CURRENT SCHEMA 特殊レジスターの値が *LIBL である場合、表は、現行ライブラリー (*CURLIB) 内に作成されます。
- そうでない場合、表は現行スキーマ内に作成されます。

FOR SYSTEM NAME *system-object-identifier*
表のシステム・オブジェクト *ID* を示します。システム・オブジェクト *ID* は、現行サーバーに既に存在する表、ビュー、別名、または索引と同一であってはなりません。システム・オブジェクト *ID* は、非修飾システム ID でなければなりません。

システム・オブジェクト *ID* が指定される場合、表名 は有効なシステム・オブジェクト名であってはなりません。

column-definition

列の属性を定義します。少なくとも 1 つ以上で、8000 を超えない列の定義がなければなりません。

列の行バッファー・バイト・カウントの合計は、32766 (VARCHAR 列、VARGRAPHIC 列、または VARBINARY 列が指定されている場合は 32740) 以下でなければなりません。さらに、LOB 列または XML 列を指定してある場合は、挿入または更新の時点で、すべての列の行データ・バイト・カウントの合計が 3 758 096 383 を超えていてはなりません。データ・タイプごとの列のバイト・カウントについては、 1295 ページの『最大行サイズ 』を参照してください。

column-name
表を構成する列の名前を指定します。列名 は修飾できません。表の複数の列や表のシステム列名に同じ名前を使用することもできません。

FOR COLUMN システム列名
列の IBM i 名を指定します。表の複数の列やシステム列名に対して、同じ名前を使用してはなりません。

システム列名が指定されず、また列名が有効なシステム列名でない場合には、システム列名が生成されます。システム列名の生成方法に関する詳細については、 1297 ページの『列名の生成の規則』を参照してください。

data-type
列のデータ・タイプを指定します。

built-in-type
組み込みタイプ には以下のいずれかを使用します。

SMALLINT
短整数を示します。

INTEGER または **INT**
長整数を示します。

BIGINT
64 ビット整数を示します。

DECIMAL(整数,整数) または **DEC(整数,整数)**
DECIMAL(整数) または **DEC(整数)**
DECIMAL または **DEC**
パック 10 進数を示します。最初の整数は数値の精度、つまり総桁数であり、1 から 63 までの範囲で指定できます。 2 番目の整数は、数値の位取り (小数点の右側に置く桁数) です。位取りは、0 からその数値の精度までの範囲で指定できます。

DECIMAL(*p*,0) は、DECIMAL(*p*) と指定でき、また DECIMAL(5,0) は、DECIMAL と指定できます。

NUMERIC(*integer***,***integer***)** または **NUM(***integer***,***integer***)**
NUMERIC(*integer***)** または **NUM(***integer***)**
NUMERIC または **NUM**
ゾーン 10 進数を示します。最初の整数は数値の精度、つまり総桁数であり、1 から 63 までの範囲で指定できます。 2 番目の整数は、数値の位取り (小数点の右側に置く桁数) です。位取りは、0 からその数値の精度までの範囲で指定できます。

NUMERIC(*p*) を NUMERIC(*p*,0) の代わりに、NUMERIC を NUMERIC(5,0) の代わりに使用しても構いません。

FLOAT
倍精度の浮動小数点数を示します。

FLOAT(*integer***)**
指定する整数の値によって、単精度、または倍精度の浮動小数点数を示します。整数の値は、1 から 53 までの範囲になければなりません。 1 から 24 までの値は単精度、25 から 53 までの値は倍精度を示します。デフォルト値は 53 です。

REAL
単精度の浮動小数点数を示します。

DOUBLE PRECISION または **DOUBLE**
倍精度の浮動小数点数を示します。

DECFLOAT(*integer***)**
DECFLOAT
IEEE 10 進浮動小数点数を示します。整数 の値は、16 か 34 でなければならず、これは、保管できる有効数字の数を表します。整数 を省略すると、DECFLOAT 列は、34 の有効数字を提示可能になります。

CHARACTER(*integer***)** または **CHAR(***integer***)**
CHARACTER または **CHAR**
整数 (バイト数) で指定した長さの固定長文字ストリングを示します。整数として指定できる値の範囲は、1 から 32766 (NULL 可能な場合には

32765) までです。 FOR MIXED DATA または混合データの CCSID を指定する場合は、4 から 32766 (NULL 可能な場合は 32765) までが範囲になります。長さ指定を省略すると、長さとして 1 が使用されます。

CHARACTER VARYING (*integer*) または **CHAR VARYING (*integer*)** または **VARCHAR (*integer*)**
最大長が整数 (バイト数) の可変長文字ストリングを示します。指定できる範囲は 1 から 32740 (ヌル可能な場合は 32739) までです。FOR MIXED DATA または混合データの CCSID を指定する場合は、4 から 32740 (NULL 可能な場合は 32739) までが範囲になります。

CHARACTER LARGE OBJECT (*integer*[K|M|G]) または **CHAR LARGE OBJECT (*integer*[K|M|G])** または **CLOB (*integer*[K|M|G])**
CHARACTER LARGE OBJECT または **CHAR LARGE OBJECT** または **CLOB**
指定された最大長 (バイト数) の文字ラージ・オブジェクト・ストリングを示します。最大長は、1 から 2 147 483 647 の範囲の値でなければなりません。 FOR MIXED DATA や混合データ CCSID を指定する場合は、4 から 2 147 483 647 の範囲の値を指定します。長さ指定を省略すると、1 メガバイトの長さが想定されます。 CLOB は、分散表内で使用することはできません。

integer
整数の最大値は 2 147 483 647 です。整数 は、ストリングの最大長を表します。

整数 **K**
整数の最大値は 2 097 152 です。 ストリングの最大長は、整数 の 1024 倍です。

整数 **M**
整数の最大値は 2 048 です。 ストリングの最大長は、整数 の 1 048 576 倍です。

整数 **G**
整数の最大値は 2 です。ストリングの最大長は、整数 の 1 073 741 824 倍です。

GRAPHIC(*integer*)
GRAPHIC
整数 で指定された長さを持つ固定長グラフィック・ストリングを示します。この整数は、1 から 16383 (NULL が使用可能な場合は 16382) までの範囲です。長さ指定を省略すると、長さとして 1 が使用されます。

VARGRAPHIC(*integer*) または **GRAPHIC VARYING(*integer*)**
最大長が整数 の可変長グラフィック・ストリングを示します。指定できる範囲は 1 から 16370 (NULL 可能な場合は 16369) までです。

DBCLOB(*integer*[K|M|G])
DBCLOB
指定された最大長の 2 バイト文字ラージ・オブジェクト・ストリングを示します。

最大長は、1 から 1 073 741 823 の範囲の値でなければなりません。長さ指定を省略すると、1 メガバイトの長さが想定されます。 DBCLOB は、分散表内で使用することはできません。

integer
整数の最大値は 1 073 741 823 です。 整数 は、ストリングの最大長を表します。

整数 K
整数の最大値は 1 028 576 です。 ストリングの最大長は、整数 の 1024 倍です。

整数 M
整数の最大値は 1 024 です。 ストリングの最大長は、整数 の 1 048 576 倍です。

整数 G
整数の最大値は 1 です。ストリングの最大長は、整数 の 1 073 741 824 倍です。

NATIONAL CHARACTER (*integer***)** または **NATIONAL CHAR (***integer***)** または **NCHAR (***integer***)**
NATIONAL CHARACTER または **NATIONAL CHAR** または **NCHAR**
整数 で指定された長さを持つ固定長 Unicode グラフィック・ストリングを示します。この整数は、1 から 16383 (NULL が使用可能な場合は 16382) までの範囲です。長さ指定を省略すると、長さとして 1 が使用されます。CCSID は 1200 です。

NATIONAL CHARACTER VARYING (*integer***)** または **NATIONAL CHAR VARYING (***integer***)** または **NCHAR VARYING (***integer***)** または **NVARCHAR (***integer***)**
最大長が整数 の可変長 Unicode グラフィック・ストリングを示します。指定できる範囲は 1 から 16370 (NULL 可能な場合は 16369) までです。CCSID は 1200 です。

NATIONAL CHARACTER LARGE OBJECT (*integer***[K|M|G])** または **NCHAR LARGE OBJECT (***integer***[K|M|G])** または **NCLOB(***integer***[K|M|G])**
NATIONAL CHARACTER LARGE OBJECT または **NCHAR LARGE OBJECT** または **NCLOB**
指定された最大長の Unicode 2 バイト文字ラージ・オブジェクト・ストリングを示します。

最大長は、1 から 1 073 741 823 の範囲の値でなければなりません。長さ指定を省略すると、1 メガバイトの長さが想定されます。 CCSID は 1200 です。 NCLOB は、分散表内で使用することはできません。

integer
整数の最大値は 1 073 741 823 です。 整数 は、ストリングの最大長を表します。

整数 K
整数の最大値は 1 028 576 です。 ストリングの最大長は、整数 の 1024 倍です。

整数 M
整数の最大値は 1 024 です。 ストリングの最大長は、整数 の 1 048 576 倍です。

整数 G
整数の最大値は 1 です。ストリングの最大長は、整数 の 1 073 741 824 倍です。

BINARY(*integer*)
BINARY
整数 で指定した長さの固定長バイナリー・ストリングを示します。整数として指定できる値の範囲は、1 から 32766 (NULL 可能な場合には 32765) までです。長さ指定を省略すると、長さとして 1 が使用されます。

BINARY VARYING (*integer*) または **VARBINARY(*integer*)**
最大長が整数 の可変長バイナリー・ストリングを示します。指定できる範囲は 1 から 32740 (NULL 可能な場合は 32739) までです。

BLOB(*integer*[K|M|G]) または **BINARY LARGE OBJECT(*integer*[K|M|G])**
BLOB または **BINARY LARGE OBJECT**
指定された最大長の 2 進ラージ・オブジェクト・ストリングを示します。最大長は、1 から 2 147 483 647 の範囲の値でなければなりません。長さ指定を省略すると、1 メガバイトの長さが想定されます。 BLOB は、分散表内で使用することはできません。

integer
整数の最大値は 2 147 483 647 です。整数 は、ストリングの最大長を表します。

整数 K
整数の最大値は 2 097 152 です。 ストリングの最大長は、整数 の 1024 倍です。

整数 M
整数の最大値は 2 048 です。 ストリングの最大長は、整数 の 1 048 576 倍です。

整数 G
整数の最大値は 2 です。ストリングの最大長は、整数 の 1 073 741 824 倍です。

DATE
日付を示します。

TIME
時刻を示します。

TIMESTAMP(*integer*) または **TIMESTAMP**
タイム・スタンプを示します。integer は 0 から 12 までの整数で、秒未満の精度を 0 (秒) から 12 (ピコ秒) で指定します。デフォルトは 6 (マイクロ秒) です。

DATALINK(*integer*) または **DATALINK**
指定された最大長のデータ・リンクを示します。最大長は、1 から 32717 の範囲の値でなければなりません。 FOR MIXED DATA や混合データ CCSID を指定する場合は、4 から 32717 の範囲の値を指定します。予期される最大の URL と、何らかのデータ・リンク・コメントが収まるだけの充

分な長さを指定する必要があります。長さ指定を省略すると、200 という長さが想定されます。DATALINK は、分散表内で使用することはできません。

DATALINK 値は、1 組の組み込みスカラー関数でカプセル化される値です。DLVALUE 関数で DATALINK 値を作成します。次の関数を使用すれば、DATALINK 値から属性を抽出することができます。

- DLCOMMENT
- DLLINKTYPE
- DLURLCOMPLETE
- DLURLPATH
- DLURLPATHONLY
- DLURLSCHEME
- DLURLSERVER

データ・リンクは、どの索引にもその一部として含めることはできません。したがって、基本キー、外部キー、または固有制約の 1 つの列としてデータ・リンクを組み込むことは不可能です。

ROWID

行 ID を示します。 ROWID 列は、1 つの表に 1 つだけ設けることができます。 ROWID はパーティション化された表で使用することはできません。

XML

XML 文書の場合。XML 列に挿入できるのは整形式の文書のみです。列の CCSID を 65535 とすることはできません。列の最大長は、常に 2 147 483 647 バイトになります。

XML 列には、以下の制限があります。

- 列は、どの索引にもその一部として含めることはできません。
- 列は、主キー、ユニーク・キー、または外部キーにもその一部として含めることはできません。
- チェック制約でその列を使用することはできません。
- その列でデフォルト値 (WITH DEFAULT) を指定することはできません。その列がヌル可能である場合は、列のデフォルト値は NULL 値です。
- 分散表の分散文節でその列を指定することはできません。
- パーティション表のパーティション化文節でその列を指定することはできません。

distinct-type-name

列のデータ・タイプが、特殊タイプ (ユーザー定義のデータ・タイプ) であることを指定します。この列の長さ、精度、および位取りは、それぞれ、特殊タイプのソースとなっているタイプの長さ、精度、および位取りと同じになります。スキーマ名なしの特殊タイプを指定すると、その特殊タイプ名は、SQL パス上のスキーマを検索することで解決されます。

ALLOCATE(*integer***)**

VARCHAR、VARGRAPHIC、VARBINARY、XML、および LOB タイプ

に関して、それぞれの行の該当列に対して予約するスペースを指定します。長さが割り振られた値以下の列の値は、行の固定長部分に保管されます。長さが割り振られた値より長い列の値は、行の可変長部分に保管され、これを検索するためには、余分の入出力操作が必要になります。割り振ることのできる値の範囲は、1 からストリングの最大長までですが、最大行バッファー・サイズにより制限されます。最大行バッファー・サイズについては、1295 ページの『最大行サイズ 』を参照してください。FOR MIXED DATA または混合データの CCSID を指定する場合は、4 からストリングの最大長までが範囲になります。割り振られる長さを指定しなかった場合は、割り振られる長さに 0 を指定したものと見なされます。
VARGRAPHIC の場合、整数は、DBCS または Unicode GRAPHIC 文字の数になります。デフォルト値として定数が指定され、ALLOCATE の長さがそのデフォルト値の長さより小さい場合は、ALLOCATE の長さはそのデフォルト値の長さであると見なされます。

FOR BIT DATA
列の値が、コード化文字セットと関連付けられていないこと、および変換されないことを指定します。 FOR BIT DATA は、CHARACTER または VARCHAR の列にのみ有効です。FOR BIT DATA を指定した列の CCSID は、65535 です。 FOR BIT DATA は、CLOB 列には使用できません。

FOR SBCS DATA
列の値に、SBCS (1 バイト文字セット) データが入ることを指定します。FOR SBCS DATA が、CHAR、VARCHAR、および CLOB 列のデフォルト値になるのは、表の作成時における現行サーバーのデフォルトの CCSID が DBCS 対応でない場合、あるいは、それらの列の長さが 4 よりも小さい場合です。FOR SBCS DATA は、CHARACTER、VARCHAR、または CLOB 列のみに有効です。 FOR SBCS DATA の CCSID は、表作成時点における現行サーバーのデフォルトの CCSID によって決まります。

FOR MIXED DATA
列の値に、SBCS データと DBCS データの両方が入ることを指定します。FOR MIXED DATA が、CHAR、VARCHAR、および CLOB 列のデフォルト値になるのは、表の作成時における現行サーバーのデフォルトの CCSID が DBCS 対応であり、しかも、それらの列の長さが 3 よりも大きい場合です。どの FOR MIXED DATA 列も DBCS 混用アプリケーション・サーバー・フィールドです。FOR MIXED DATA は、CHARACTER、VARCHAR、または CLOB 列のみに有効です。 FOR MIXED DATA の CCSID は、表作成時点における現行サーバーのデフォルトの CCSID によって決まります。

CCSID 整数
整数で指定した CCSID を持つデータが、列の値に入ることを指定します。その整数が SBCS CCSID である場合、その列のデータは SBCS データです。整数が混合データ CCSID である場合、その列は混合データとなり、その列の長さには、3 よりも大きい値を指定する必要があります。文字列の場合、CCSID は SBCS CCSID や 混合データ CCSID にする必要があります。グラフィックの列の場合は、CCSID は DBCS、UTF-16、または UCS-2 CCSID でなければなりません。 CCSID がグラフィックの列に指定されていない場合は、CCSID は、表の作成時点における現行サーバーのデ

フォルトの CCSID によって決まります。 XML 列の場合、CCSID は 65535 であってはなりません。XML 列の CCSID を指定しない場合は、CREATE TABLE の実行時に、SQL_XML_DATA_CCSID QAQQINI オプションの設定に基づいて CCSID が設定されます。デフォルトの CCSID は 1208 です。このオプションの説明については、 101 ページの『XML 値』を参照してください。 有効な CCSID のリストについては、 1895 ページの『付録 E. CCSID の値』の項を参照してください。

CCSID 1208 (UTF-8) または 1200 (UTF-16) データには結合文字を含めることができます。合成文字サポートにより、1 つ以上の文字を組み合わせて 1 文字にすることが可能です。データ・ストリングとして、1 文字目の後に、最大 300 の異なる非スペーシング・アクセント文字 (ウムラウト、アクサンなど) を続けることができます。結果文字が文字セットで既に定義済みの文字である場合、その文字には複数の表記があります。正規化 は、合成文字のストリングを定義済みの 16 進値で置き換えます。これにより、同じ文字が単一の一貫した方法で表記されるようになります。正規化が実行されない場合、同一に見える 2 つのストリングは等しく比較されません。

NOT NORMALIZED
データはアプリケーションからの受け渡し時に正規化されません。

NORMALIZED
データはアプリケーションからの受け渡し時に正規化されます。

DEFAULT
列のデフォルト値を指定します。この文節は、1 つの列定義 の中で複数回指定することはできません。次のタイプの列に対しては、Db2 がデフォルト値を生成するため、DEFAULT を指定できません。

- ROWID 列
- ID 列 (AS IDENTITY として定義される列)
- 行変更タイム・スタンプ列
- 行開始列
- 行終了列
- トランザクション開始 ID 列
- 生成式列

XML 列のデフォルトは、NULL です。ただし、NOT NULL を指定した場合、デフォルトはありません。

DEFAULT キーワードの後に値が指定されていない場合は、次のようになります。

- 列がNULL 可能の場合、デフォルト値は NULL 値になります。
- 列がNULL 可能でない場合、デフォルト値は列のデータ・タイプによって決まります。

データ・タイプ	デフォルト値
数値	0
固定長文字またはグラフィック・ストリング	ブランク
固定長バイナリー・ストリング	16 進ゼロ

データ・タイプ	デフォルト値
可変長ストリング	0 のストリング長
日付	INSERT の時点の当日の日付
時刻	INSERT の時点の当日の時刻
タイム・スタンプ	INSERT の時点の当日のタイム・スタンプ
データ・リンク	DLVALUE('','URL','') に対応する値
特殊タイプ	特殊タイプの対応するソース・タイプのデフォルト値

NOT NULL および DEFAULT を列の定義 から省いた場合、DEFAULT NULL の暗黙の指定が取られます。

constant
その列のデフォルト値としての定数を指定します。これは、 113 ページの『割り当ておよび比較』で説明している割り当て規則に従って、その列に割り当てることができる値を表す定数にする必要があります。浮動小数点定数または 10 進浮動小数点定数は、SMALLINT、INTEGER、DECIMAL、または NUMERIC 列に使用してはなりません。10 進定数には、小数点より右方に、その列に指定された位取りより多くの桁を含めてはなりません。

USER
INSERT または UPDATE の時点での USER 特殊レジスターの値を、その列のデフォルト値として指定します。列のデータ・タイプは、USER 特殊レジスターの長さ属性と同じかそれより大きい長さ属性を持つ CHAR または VARCHAR でなければなりません。

NULL
その列のデフォルト値として NULL を指定します。 NOT NULL を指定する場合は、同じ列の定義 内で DEFAULT NULL を指定してはなりません。

データ・リンク列に使用できるデフォルト値は NULL のみです。

CURRENT_DATE
現在の日付を列のデフォルト値として指定します。 CURRENT_DATE を指定する場合は、列のデータ・タイプは DATE または DATE に基づく特殊タイプでなければなりません。

CURRENT_TIME
現在の時刻を列のデフォルト値として指定します。 CURRENT_TIME を指定する場合は、列のデータ・タイプは TIME または TIME に基づく特殊タイプでなければなりません。

CURRENT_TIMESTAMP または **CURRENT_TIMESTAMP(***integer***)**
現在のタイム・スタンプを列のデフォルト値として指定します。 CURRENT_TIMESTAMP を指定する場合は、列のデータ・タイプは TIMESTAMP または TIMESTAMP に基づく特殊タイプでなければなりません。デフォルトとして使用される CURRENT_TIMESTAMP 特殊レジスターのタイム・スタンプ精度は、この特殊レジスターに指定された精度に関係なく、常に列のタイム・スタンプ精度と一致します。

cast-function-name
この形式のデフォルト値は、特殊タイプやデータ・タイプ、BINARY、

VARBINARY、BLOB、 CLOB、DBCLOB、DATE、TIME または TIMESTAMP として定義された列でのみ使用することができます。次の表は、これらのキャスト関数 の許可されている使用法を示します。

データ・タイプ	キャスト関数名
BINARY、VARBINARY、BLOB、CLOB、または DBCLOB に基づく特殊タイプ N	BINARY、VARBINARY、BLOB、CLOB、または DBCLOB *
DATE、TIME、または TIMESTAMP に基づく特殊タイプ N	N (N の作成時に生成されたユーザー定義のキャスト関数) ** あるいは DATE、TIME、または TIMESTAMP *
他のデータ・タイプに基づく特殊タイプ	N (N の作成時に生成されたユーザー定義のキャスト関数) **
BINARY、VARBINARY、BLOB、CLOB、または DBCLOB	BINARY、VARBINARY、BLOB、CLOB、または DBCLOB *
DATE、TIME、または TIMESTAMP	DATE、TIME、または TIMESTAMP *
注: * 関数には、QSYS2 の暗黙的または明示的スキーマ名のデータ・タイプ (または、特殊タイプのソース・タイプ) の名前と一致する名前を指定する必要があります。 ** 関数には、列の特殊タイプの名前と一致する名前を指定する必要があります。スキーマ名で修飾する場合は、特殊タイプのスキーマ名と同じ名前を指定する必要があります。修飾しない場合は、関数の解析から得られるスキーマ名は、特殊タイプのスキーマ名と同じ名前にする必要があります。	

constant
定数を引数として指定します。この定数は、特殊タイプのソース・タイプの定数、あるいは、特殊タイプでない場合は、データ・タイプの定数の規則に準拠する必要があります。BINARY、VARBINARY、 BLOB、CLOB、DBCLOB、DATE、TIME、および TIMESTAMP 関数の場合は、この定数をストリング定数にする必要があります。

USER
INSERT または UPDATE の時点での USER 特殊レジスターの値をその列のデフォルト値として指定します。この列の特殊タイプのソース・タイプのデータ・タイプは、USER 特殊レジスターの長さ属性と同じかそれより大きい長さ属性を持つ CHAR または VARCHAR でなければなりません。

CURRENT_DATE
現在の日付を列のデフォルト値として指定します。 CURRENT_DATE を指定する場合、列の特殊タイプのソース・タイプのデータ・タイプは、DATE にする必要があります。

CURRENT_TIME
現在の時刻を列のデフォルト値として指定します。 CURRENT_TIME を指定する場合、列の特殊タイプのソース・タイプのデータ・タイプは、TIME にする必要があります。

CURRENT_TIMESTAMP または **CURRENT_TIMESTAMP(***integer***)**
現在のタイム・スタンプを列のデフォルト値として指定します。CURRENT_TIMESTAMP を指定する場合、列の特殊タイプのソース・タイプのデータ・タイプは、TIMESTAMP にする必要があります。デフォルトとして使用される CURRENT_TIMESTAMP 特殊レジスターのタイム・スタンプ精度は、この特殊レジスターに指定された精度に関係なく、常に列のタイム・スタンプ精度と一致します。

指定した値が無効である場合、エラーが戻されます。

GENERATED
データベース・マネージャーが列の値を生成することを示します。列が次のいずれかのタイプの列であると見なされる場合には、GENERATED を指定できます。

- ID 列
- 行変更タイム・スタンプ列

列を次のいずれかのタイプの列に統合する場合には、GENERATED を指定する必要があります。

- 行開始列
- 行終了列
- トランザクション開始 ID 列
- 生成式列

また、列のデータ・タイプが ROWID (または ROWID に基づく特殊タイプ) である場合も、GENERATED を指定できます。その他の場合は、GENERATED を指定してはなりません。GENERATED は、列定義内に *default-clause* とともに指定してはなりません。in

ALWAYS
行の挿入時または更新時にデフォルト値を生成しなければならない場合に、データベース・マネージャーが常に列の値を生成することを指定します。ALWAYS は推奨値です。

BY DEFAULT
行の挿入時または更新時にデフォルト値を生成しなければならない場合に、明示的な値が指定されていない限り、データベース・マネージャーが列の値を生成することを指定します。

ROWID 列の場合は、データベース・マネージャーは指定された値を使用しますが、その値は、既にデータベース・マネージャーか Db2 for i によって生成されている有効な固有の行 ID の値でなければなりません。

ID 列または行変更タイム・スタンプ列の場合は、データベース・マネージャーは指定された値を挿入または更新しますが、その ID 列または行変更タイム・スタンプ列がユニーク制約を持っているか、その ID 列または行変更タイム・スタンプ列を単独で指定するユニーク索引を持っている場合を除き、その値がその列の固有な値であるかどうかの検査は行いません。

AS IDENTITY
列が表の識別列であることを指定します。 1 つの表は識別列を 1 つだけ持つことができます。識別列は、分散表内で使用することはできません。AS

IDENTITY を指定できるのは、列のデータ・タイプが、厳密に位取りがゼロの数値タイプ (SMALLINT、INTEGER、 BIGINT、DECIMAL、または位取りがゼロの NUMERIC、またはこれらのデータ・タイプに基づく特殊タイプ) である場合だけです。 DECIMAL または NUMERIC データ・タイプが指定された場合、精度は 31 以下でなければなりません。

識別列は、暗黙的に NOT NULL になります。

START WITH数値定数
識別列について生成される最初の値を指定します。小数点の右側にゼロ以外の数字がないことを条件として、この列に割り当てることのできる任意の正または負の値を指定できます。

識別列を定義するときに値を明示的に指定していない場合のデフォルト値は、昇順の場合は MINVALUE で、降順の場合は MAXVALUE です。この値は、シーケンスが最大値または最小値に達した後で、シーケンスの循環により到達する値になるとは限りません。 START WITH 文節を使用することにより、この循環に使用される値の範囲外の値からシーケンスを開始することができます。循環に使用する範囲は、MINVALUE および MAXVALUE で定義します。

INCREMENT BY*numeric-constant*
識別列の連続した値の間隔を指定します。この値は長整数定数の値を超過せず、かつ小数点の右側にゼロ以外の数字があってはなりません。値は列に割り当て可能でなければなりません。デフォルトは、1 です。

この値が 0 または正である場合は、識別列の値の順序は昇順になります。この値が負の場合は、値の順序は降順になります。

MAXVALUE または MINVALUE
昇順の ID 列が循環するか値の生成を停止する最大値、または降順の ID 列が最小値に達した後で循環する先の最大値を指定します。

MAXVALUE *numeric-constant*
この ID 列用として生成される最大値を示す数値定数を指定します。この値には、この列に割り当てることのできる任意の正または負の値を指定できますが、最小値より大きい値でなければなりません。

識別列を定義するときにこの値を明示的に指定していない場合は、この値は、昇順シーケンスの場合は該当データ・タイプの最大値になります。降順シーケンスの場合は、この値は START WITH の値ですが、START WITH を指定していなければ -1 です。

MINVALUE*numeric-constant*
この識別列用として生成される最小値を示す数値定数を指定します。この値には、この列に割り当てることのできる任意の正または負の値を指定できますが、最大値より小さい値でなければなりません。

識別列を定義するときにこの値を明示的に指定していない場合は、この値は、昇順シーケンスの場合は START WITH の値ですが、START WITH を指定していなければ 1 です。降順シーケンスの場合は、この値は、該当データ・タイプ (および、DECIMAL の場合は精度) の最小値です。

CACHE または **NO CACHE**
事前割り振りの値をメモリー内に保持するかどうかを指定します。値を事前に割り振ってキャッシュに保管しておくと、表に行を挿入するときのパフォーマンスが向上します。

CACHE 整数
データベース・マネージャーが事前割り振りしてメモリー内に保持する、識別列シーケンスの値の数を指定します。指定できる最小の値は 2 で、最大の値は、1 つの整数で表せる最大の値です。デフォルト値は 20 です。

システム障害のような特定の状態になると、キャッシュに保管されていてコミット済みステートメントでまだ使用されていない ID 列値はすべて失われるため、その後使用されることはありません。 CACHE オプションに指定する値は、こうした状態になった場合に失われる ID 列値の最大数です。

NO CACHE
識別列の値を事前割り振りしないことを指定します。

CYCLE または **NO CYCLE**
シーケンスの最大値または最小値に達した後も、この識別列について値を生成し続けるかどうかを指定します。

CYCLE
最大値または最小値に達した後も、この列の値を生成し続けることを指定します。このオプションを使用した場合は、昇順シーケンスがシーケンスの最大値に達した後では、最小値が生成されます。降順シーケンスがシーケンスの最小値に達した後は、最大値が生成されます。列の最大値と最小値によって、循環に使用される範囲が決まります。

CYCLE が有効であるとき、データベース・マネージャーは ID 列に対して重複する値を生成する可能性があります。対象の識別列について固有制約または固有索引が存在する場合は、固有でない値が生成されるとエラーが起こります。

NO CYCLE
シーケンスの最大値または最小値に達した後は、この識別列について値を生成しないことを指定します。これはデフォルトです。

ORDER または **NO ORDER**
識別値を要求された順序で生成するかどうかを指定します。

ORDER
要求された順序で値を生成することを指定します。

NO ORDER
値を要求された順序で生成する必要がないことを指定します。これはデフォルトです。

FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
列がタイム・スタンプであり、値はデータベース・マネージャーによって生成されることを指定します。データベース・マネージャーは、行が挿入されるたびに各行の列に値を生成し、また列が更新されるたびに各行に値を生成します。行変更タイム・スタンプ列に生成される値は、その行の挿入または更新の時刻に対応

するタイム・スタンプです。1 つの SQL ステートメントを指定して複数の行が挿入される場合、行変更タイム・スタンプ列の値は、各行が挿入された時点を反映するために行ごとに異なる可能性があります。生成された値が固有である保証はありません。

表には、行変更タイム・スタンプ列を 1 つだけ指定できます。データ・タイプを指定する場合は、精度 6 の TIMESTAMP であるか、または、精度 6 の TIMESTAMP に基づく特殊タイプでなければなりません。行変更タイム・スタンプ列の場合、DEFAULT 節を指定することはできず、NOT NULL でなければなりません。

AS ROW BEGIN

列がタイム・スタンプ・データを含み、値がデータベース・マネージャーによって生成されることを指定します。データベース・マネージャーは、行が挿入されるたびに各行の列に値を生成し、また列が更新されるたびに各行に値を生成します。生成される値は、最新のトランザクションに関連付けられている開始時刻に対応するタイム・スタンプです。単一 SQL ステートメントで複数の行が挿入される場合、トランザクション開始タイム・スタンプ列の値は各行で同じになります。

システム期間テンポラル表の場合、行開始列の値は、トランザクション全体にわたり固有になるようにデータベース・マネージャーによって生成されます。関連した履歴表に挿入される行の終了タイム・スタンプ値が開始タイム・スタンプ値より大きくなるように、タイム・スタンプ値が調整される可能性があります。これは、競合するトランザクションがシステム期間テンポラル表の同じ行を更新しているときに行われる場合があります。このタイム・スタンプ値の調整を行うには、SYSTIME_PERIOD_ADJ QAQQINI オプションを *ADJUST に設定する必要があります。単一の SQL トランザクション内で複数の行が挿入または更新され、調整が必要ではない場合、行開始列の値はすべての行において同じになり、別のトランザクションでその列のために生成された値とは異なる固有の値になります。

行開始列は、システム期間テンポラル表で使用するためのもので、SYSTEM_TIME 期間の最初の列として必要です。 1 つの表は 1 つの行開始列しか持てません。*data-type* が指定されない場合、列は TIMESTAMP(12) として定義されます。*data-type* を指定する場合は、TIMESTAMP(12) でなければなりません。この列には DEFAULT 節を指定できないため、NOT NULL として定義する必要があります。行開始列は更新できません。

AS ROW END

これを指定すると、行が挿入されるとき、または行内のいずれかの列が更新されるときには常に、データベース・マネージャーによって列のデータ・タイプの値が割り当てられます。割り当てられる値は TIMESTAMP '9999-12-30-0.00.00.000000000000' です。システム期間テンポラル表では、行が削除されると、履歴行の行終了列の値に、行がいつ削除されたかが反映されます。単一の SQL ステートメントで複数の行が削除される場合、履歴行の列の値は同じになります。

行終了列は、システム期間テンポラル表で使用するためのもので、SYSTEM_TIME 期間の 2 番目の列として必要です。 表には 1 つの行終了列のみを含めることができます。*data-type* が指定されない場合、列は TIMESTAMP(12) として定義されます。*data-type* を指定する場合は、

TIMESTAMP(12) でなければなりません。この列には DEFAULT 節を指定できないため、NOT NULL として定義する必要があります。行終了列は更新できません。

AS TRANSACTION START ID
これを指定すると、行が表に挿入されるとき、または行のいずれかの列が更新されるときには常に、データベース・マネージャーによって値が割り当てられます。データベース・マネージャーは、トランザクションごとに固有のタイム・スタンプ値、または NULL 値を割り当てます。トランザクション開始 ID 列が NULL 可能で、値を調整する必要がない行開始列が表にある場合には、その列に NULL 値が割り当てられます。それ以外の場合、この値は、次のいずれかの場合に時刻機構を読み取ることによって生成されます。(1) トランザクションの中で、表に含まれる行開始列またはトランザクション開始 ID 列に値を割り当てる必要があるようなデータ変更ステートメントを最初に実行するとき。(2) システム期間テンポラル表に含まれる行を削除するとき。単一の SQL トランザクション内で複数の行が挿入または更新される場合、トランザクション開始 ID 列の値はすべての行において同じになり、別のトランザクションでその列のために生成された値とは異なる固有の値になります。

トランザクション開始 ID 列は、システム期間テンポラル表で使用するためのもので、システム期間テンポラル表に必要です。 1 つの表は 1 つのトランザクション開始 ID 列しか持てません。*data-type* が指定されない場合、列は TIMESTAMP(12) として定義されます。*data-type* を指定する場合は、TIMESTAMP(12) でなければなりません。トランザクション開始 ID 列には DEFAULT 節を指定できません。トランザクション開始 ID 列は更新できません。

DATA CHANGE OPERATION
挿入された各行、列が更新されたすべての行、および履歴表が ON DELETE ADD EXTRA ROW で定義されている場合にシステム期間テンポラル表から削除されたすべての行に、データベース・マネージャーが値を生成することを指定します。列には、以下のいずれかの値が含まれます。

I 挿入操作

U 更新操作

D 削除操作

data-type が指定されない場合、列は CHAR(1) として定義されます。*data-type* を指定する場合は、CHAR(1) でなければなりません。この列には、DEFAULT 文節もフィールド・プロシージャーも指定できません。

special-register
挿入された各行、列が更新されたすべての行、および履歴表が ON DELETE ADD EXTRA ROW で定義されている場合にシステム期間テンポラル表から削除されたすべての行に、データベース・マネージャーによって特殊レジスターの値が割り当てられることを指定します。データ変更ステートメントの時点での特殊レジスターの値が使用されます。単一の SQL ステートメントを使用して複数の行が変更される場合、列の値は、すべての行で同じになります。

data-type は、以下の表に従って定義する必要があります。

特殊レジスター	列のデータ・タイプ
CURRENT CLIENT_ACCTNG	VARCHAR(255)
CURRENT CLIENT_APPLNAME	VARCHAR(255)
CURRENT CLIENT_PROGRAMID	VARCHAR(255)
CURRENT CLIENT_USERID	VARCHAR(255)
CURRENT CLIENT_WRKSTNNAME	VARCHAR(255)
CURRENT SERVER	VARCHAR(18)
SESSION_USER	VARCHAR(128)
USER	VARCHAR(18)

この列には、DEFAULT 文節もフィールド・プロシージャーも指定できません。

built-in-global-variable
挿入された各行、列が更新されたすべての行、および履歴表が ON DELETE ADD EXTRA ROW で定義されている場合にシステム期間テンポラル表から削除されたすべての行に、データベース・マネージャーによって組み込みグローバル変数の値が割り当てられることを指定します。データ変更ステートメントの時点での組み込みグローバル変数の値が使用されます。単一の SQL ステートメントを使用して複数の行が変更される場合、列の値は、すべての行で同じになります。

data-type は、以下の表に従って定義する必要があります。

組み込みグローバル変数	列のデータ・タイプ
QSYS2.JOB_NAME	VARCHAR(28)
QSYS2.SERVER_MODE_JOB_NAME	VARCHAR(28)
SYSIBM.CLIENT_HOST	VARCHAR(255)
SYSIBM.CLIENT_IPADDR	VARCHAR(128)
SYSIBM.CLIENT_PORT	INTEGER
SYSIBM.PACKAGE_NAME	VARCHAR(128)
SYSIBM.PACKAGE_SCHEMA	VARCHAR(128)
SYSIBM.PACKAGE_VERSION	VARCHAR(64)
SYSIBM.ROUTINE_SCHEMA	VARCHAR(128)
SYSIBM.ROUTINE_SPECIFIC_NAME	VARCHAR(128)
SYSIBM.ROUTINE_TYPE	CHAR(1)

この列には、DEFAULT 文節もフィールド・プロシージャーも指定できません。

NOT NULL
列に NULL 値が入るのを防止します。NOT NULL を指定しないことは、その列が NULL であってもよいことを意味します。行変更タイム・スタンプ列、行開始列、および行終了列には、NOT NULL が必要です。

NOT HIDDEN
列が SQL ステートメントの表の暗黙的参照に組み込まれることを示します。これはデフォルトです。

IMPLICITLY HIDDEN
名前で明示的に参照されない限り、列は SQL ステートメントから不可視であることを示します。例えば、SELECT * は、結果に隠し列を含みません。表には、少なくとも 1 つの IMPLICITLY HIDDEN でない列が含まれていなければなりません。

period-definition

PERIOD FOR
表の期間を定義します。

SYSTEM_TIME (*begin-column-name***, *end-column-name*)**
システム期間を SYSTEM_TIME という名前で定義します。表内に名前 SYSTEM_TIME の列があってはなりません。表に指定できる SYSTEM_TIME 期間は 1 つのみです。

begin-column-name
行が有効である期間の開始を記録する列を指定します。この名前は、表内に存在する列を示すものでなければなりません。*begin-column-name* は、*end-column-name* と同じであってはなりません。*begin-column-name* は AS ROW BEGIN として定義する必要があります。

end-column-name
行が有効である期間の終了を記録する列を指定します。システム期間テンポラル表に関連付けられた履歴表で、システム期間テンポラル表の *end-column-name* に対応する履歴表の列はその行の削除を反映するように設定されます。この名前は、表内に存在する列を示すものでなければなりません。*end-column-name* は AS ROW END として定義する必要があります。

column-constraint

CONSTRAINT *constraint-name*
制約の名前を指定します。制約名 は、既に CREATE TABLE ステートメントで指定され、かつ既に現行サーバーに存在している制約を示すものであってはなりません。

この文節の指定がない場合、固有制約の名前がデータベース・マネージャーによって生成されます。

PRIMARY KEY
これは、1 つの列からなる基本キーを定義する簡便な手段です。列 C の定義に PRIMARY KEY を指定した場合、その効果は、別個の文節として PRIMARY KEY(C) 文節を指定したのと同一です。

この文節は複数の列の定義に指定してはなりません。また列の定義に UNIQUE 文節の指定がある場合には、この文節を指定してはなりません。この列は、LOB 列、DATALINK 列、または XML 列であってはなりません。ソート・シーケンスを指定する場合、列にフィールド・プロシージャーを含めることはできません。

基本キーを追加すると、CHECK 制約が暗黙的に追加され、その基本キーを構成する列で NULL を使用することはできないという規則が適用されます。

UNIQUE
これは、1 つの列からなるユニーク制約を定義する簡便な手段です。列 C

の定義に UNIQUE の指定がある場合、その効果は、別個の文節として UNIQUE(C) 文節が指定された場合と同一です。

この文節は、1 つの列定義で複数回指定することはできません。また、列定義で PRIMARY KEY が指定されている場合は、この文節を指定してはなりません。この列は、LOB 列、DATALINK 列、または XML 列であってはなりません。ソート・シーケンスを指定する場合、列にフィールド・プロシージャーを含めることはできません。

references-clause
列定義 の *REFERENCES* 文節 は、1 つの列からなる外部キーを定義する簡便な手段です。列 C の定義に参照文節の指定がある場合、その効果は、C が識別された唯一の列である FOREIGN KEY 文節の一環としてその参照文節が指定されている場合と同一です。表が宣言済みグローバル一時表、分散表、または履歴表である場合は、参照文節 を使用することはできません。この列は、行変更タイム・スタンプ列であってはなりません。

CHECK(*check-condition***)**
列定義 の CHECK(検査条件) は、単一の列のみを参照する検査条件 を持つ検査制約を定義するための簡便な手段です。したがって、列 C の列定義で CHECK を指定した場合、検査制約の検査条件 では、C 以外の列を参照することができなくなります。結果は、検査制約を別個の文節として指定した場合と同じです。

ON INSERT VIOLATION または ON UPDATE VIOLATION が指定される場合、これらの節の中で列 C が参照される必要があります。

CHECK 制約の中で、FILE LINK CONTROL 列を持つ ROWID、XML、および DATALINK を参照することはできません。その他の制限事項については、 1278 ページの『check-constraint』を参照してください。

FIELDPROC
列のフィールド・プロシージャー出口ルーチンとして、外部プログラム名 を指定します。SQL が含まれていない ILE プログラムを指定する必要があります。サービス・プログラムを指定することはできません。

このフィールド・プロシージャーは列の値のエンコードとデコードを行います。列に値が挿入されるときは、フィールド・プロシージャーに渡されてエンコードされてから挿入されます。列に入っている値が使用されるときは、フィールド・プロシージャーに渡されてデコードされてから使用されます。

フィールド・プロシージャーは、CREATE TABLE ステートメントの処理中にも呼び出されます。この呼び出しの場合、プロシージャーは Db2 に列のフィールド記述を提供します。フィールド記述は、エンコードされた値のデータ特性を定義します。一方、CREATE TABLE ステートメントで列について指定する情報では、デコードされた値のデータ特性を定義します。

constant
フィールド・プロシージャーの呼び出し時にフィールド・プロシージャーに渡すパラメーターを指定します。パラメーター・リストはオプションです。

ROWID または DATALINK の列、あるいは ROWID または DATALINK に基づく特殊タイプの列でフィールド・プロシージャーを定義することはできません。この列は、ID 列、行変更タイム・スタンプ列、行開始列、行終了列、トラ

ンザクション開始 ID 列、および生成式列であってはなりません。この列に、CURRENT DATE、CURRENT TIME、CURRENT TIMESTAMP、USER のいずれかのデフォルト値が入っていてはなりません。フィールドのエンコード形式とデコード形式では、NULL 可能性属性が一致している必要があります。チェック条件でこの列を参照することはできません。ただし、NULL 述部で参照する場合は例外です。この列が外部キーの一部になっている場合は、対応する親キー列でも同じフィールド・プロシージャーを使用する必要があります。フィールド・プロシージャーを作成する方法の詳細については、『SQL プログラミング』を参照してください。

datalink-options
DATALINK データ・タイプに関連したオプションを指定します。

LINKTYPE URL
リンクのタイプを URL として定義します。

NO LINK CONTROL
これを指定すると、リンク済みファイルが存在するか否かを判別するための検査は行われなくなります。URL の構文だけが検査されます。リンク済みファイルに関してデータベース・マネージャー制御は行われません。

FILE LINK CONTROL
これを指定すると、リンク済みファイルの存在を確かめるための検査を行う必要が生じます。追加のオプションを使用して、データベース・マネージャーにリンク済みファイルに対するより強力な制御権を与えることが可能です。

FILE LINK CONTROL が指定されると、各ファイルは一度だけリンクすることができます。つまり、その URL を指定できるのは単一の表内の単一の FILE LINK CONTROL 列内だけです。

ファイル・リンク・オプション
リンク済みファイルのデータベース・マネージャー制御のレベルを定義する追加のオプションです。

INTEGRITY
DATALINK 値と実ファイルとの間のリンクの整合性レベルを指定します。

ALL
これを指定すると、DATALINK 値として指定されたどのファイルもデータベース・マネージャーに制御されるようになります。また、標準ファイル・システムのプログラミング・インターフェースを使用してそれらのファイルの削除または名前変更は行うことができなくなります。

READ PERMISSION
DATALINK 値に指定されたファイルの読み取り許可を決定する方法を指定します。

FS これを指定すると、読み取りアクセス許可は、ファイル・システム許可によって決定されるようになります。このようなファイルには、列からファイル名を検索しなくてもアクセスできます。

DB これを指定すると、読み取りアクセス許可は、データベースによっ

て決定されるようになります。このファイルへのアクセスは、オープン操作において、表から DATALINK 値の検索時に戻される有効なファイル・アクセス・トークンを渡すことでしか許可されません。READ PERMISSION DB を指定する場合は、WRITE PERMISSION BLOCKED も指定する必要があります。

WRITE PERMISSION
DATALINK 値に指定されたファイルの書き込み許可を決定する方法を指定します。

FS これを指定すると、書き込みアクセス許可は、ファイル・システム許可によって決定されるようになります。このようなファイルには、列からファイル名を検索しなくてもアクセスできます。

BLOCKED
これを指定すると、書き込みアクセスがブロックされます。ファイルは、どのインターフェースを介しても直接更新することはできません。情報に対して更新を実行するには、代替メカニズムを使用する必要があります。例えば、ファイルをコピーし、そのコピーを更新してから、その DATALINK 値を更新することで、そのファイルの新しいコピーを指し示します。

RECOVERY
この列の値で参照されるファイルの特定時点リカバリーをデータベース・マネージャーがサポートするか否かを指定します。

NO これを指定すると、特定時点リカバリーはサポートされません。

ON UNLINK
DATALINK 値の変更または削除 (リンク解除) 時にファイルに対して講じるアクションを指定します。これは、WRITE PERMISSION FS を使用している場合には適用できないことに注意してください。

RESTORE
これを指定すると、ファイルのリンクが解除されたら、データ・リンク・ファイル・マネージャーは、そのファイルを、それがリンクされた時点で存在していた許可と一緒に所有者に戻そうとします。その所有者が既にファイル・サーバーへの登録を解除されている場合、この結果は、それらのファイルが収められているファイル・システムによって異なります。それらのファイルが AIX® ファイル・システムにある場合の所有者は「dfmunknown」です。 IFS にある場合の所有者は QDLFM です。これは、INTEGRITY ALL と WRITE PERMISSION BLOCKED も指定されている場合にのみ指定することができます。

DELETE
これを指定すると、ファイルは、リンク解除の時点で削除されます。これは、READ PERMISSION DB と WRITE PERMISSION BLOCKED も指定されている場合にのみ指定することができます。

MODE DB2OPTIONS
このモードは、1 組のデフォルト・ファイル・リンク・オプションを定義します。 DB2OPTIONS によって定義されるデフォルト値は、次のとおりです。

- INTEGRITY ALL
- READ PERMISSION FS
- WRITE PERMISSION FS
- RECOVERY NO

LIKE

table-name または *view-name*

表の列の名前と記述が、指定された表 (表名) またはビュー (ビュー名) の列とまったく同じであることを指定します。この名前は、現行サーバーにある表またはビューを識別するものでなければなりません。

LIKE を使用すると、*n* 列を暗黙的に定義したことになります。ここで、*n* は、指定した表またはビュー内の列の数です。この暗黙の定義には、*n* 列の以下の属性が含まれます (そのデータ・タイプに該当する場合):

- 列名 (および、システム列名)
- データ・タイプ、長さ、精度、および位取り
- CCSID
- FIELDPROC (*table-name* のためにコピーされます)

表名 の直後に LIKE 文節を指定し、括弧で囲まなかった場合は、以下の列属性も含まれます。その他の場合は、これらの属性は含まれません (デフォルト値、識別、行変更タイム・スタンプ、および隠し属性は、コピー・オプションを使用して制御することもできます)。

- デフォルト値 (表名 が指定され、ビュー名 は指定されていない場合)
- NULL 可能性
- 隠し属性
- 列の見出しとテキスト (1570 ページの『LABEL』を参照)

列の REFFLD 情報は、新しい列定義にコピーされます。

table-name に行変更タイム・スタンプ列、行開始列、行終了列、トランザクション開始 ID 列、または生成式列が含まれている場合、新しい表の対応する列はソース列のデータ・タイプのみを継承します。この新しい列は、生成された列とは見なされません。

暗黙の定義には、識別された表またはビューのその他のオプション属性は含まれません。例えば、新しい表に基本キー、外部キー、トリガー、または期間が自動的に組み込まれることはありません。新規の表にこうしたオプション属性が組み込まれるのは、オプション文節を明示的に指定した場合に限られます。

指定された表またはビューが非 SQL 作成の物理ファイルまたは論理ファイルの場合、非 SQL 属性は除去されます。例えば、日付と時刻の形式は ISO 形式に変更されます。

copy-options

INCLUDING IDENTITY COLUMN ATTRIBUTES または **EXCLUDING IDENTITY COLUMN ATTRIBUTES**

ID 列の属性を継承するかどうかを指定します。

INCLUDING IDENTITY COLUMN ATTRIBUTES
この表が、選択ステートメント、表名、またはビュー名 の結果として生じる列の識別属性 (もしあれば) を継承することを指定します。一般に、識別属性がコピーされるのは、表、ビュー、または選択ステートメント の中の対応する列のエレメントが、識別属性を持つ基本表列の名前に直接または間接にマップされる表列またはビュー列の名前である場合です。
INCLUDING IDENTITY COLUMN ATTRIBUTES 文節と AS 選択ステートメント 文節を指定してあるときは、以下の場合には新規の表の列は識別属性を継承しません。

- 選択ステートメント の選択リストに、ID 列名の複数のインスタンスが含まれている (つまり同じ列を複数回選択している) 場合。
- 選択ステートメント の選択リストに、複数の ID 列が含まれている (つまり、結合が複数の ID 列を戻した) 場合。
- 選択リスト内の式のいずれかに ID 列が含まれている場合。
- 選択ステートメント に一組の演算 (UNION または INTERSECT) が含まれている場合。

INCLUDING IDENTITY を指定しなかった場合は、表には ID 列は含まれません。

EXCLUDING IDENTITY COLUMN ATTRIBUTES
この表が、全選択、表名、またはビュー名 の結果として生じる列の識別属性を継承しないことを指定します。

EXCLUDING COLUMN DEFAULTS または **INCLUDING COLUMN DEFAULTS** または **USING TYPE DEFAULTS**
列のデフォルトを継承するかどうかを指定します。

EXCLUDING COLUMN DEFAULTS
ソース表の定義から列のデフォルトを継承しないことを指定します。新しい表の列のデフォルト値は NULL になるか、またはデフォルト値がなくなります。列を NULL にできる場合、デフォルトは NULL 値になります。列を NULL にできない場合はデフォルト値がなくなるため、新しい表に対する INSERT で列の値が指定されない場合はエラーが発生します。

INCLUDING COLUMN DEFAULTS
この表が、選択ステートメント、表名、またはビュー名 から生じる列のデフォルト値を継承することを指定します。一般に、デフォルト値がコピーされるのは、表、ビュー、または選択ステートメント の中の対応する列のエレメントが、デフォルト値を持つ基本表列の名前に直接または間接にマップされる表列またはビュー列の名前である場合です。デフォルト値は、INSERT で値が指定されていない場合に、列に割り当てられる値です。

USING TYPE DEFAULTS を指定する場合は、INCLUDING COLUMN DEFAULTS を指定しないでください。

INCLUDING COLUMN DEFAULTS を指定しなかった場合は、デフォルト値は継承されません。

USING TYPE DEFAULTS
この表のデフォルト値が、選択ステートメント、表名、またはビュー名 から生じる列のデータ・タイプに応じて決まることを指定します。その列が

NULL 可能である場合は、デフォルト値は NULL 値です。その他の場合は、デフォルト値は以下のようになります。

データ・タイプ	デフォルト値
数値	0
固定長文字またはグラフィック・ストリング	ブランク
固定長バイナリー・ストリング	16 進ゼロ
可変長ストリング	0 のストリング長
日付	INSERT の時点の当日の日付
時刻	INSERT の時点の当日の時刻
タイム・スタンプ	INSERT の時点の当日のタイム・スタンプ
データ・リンク	DLVALUE('','URL','') に対応する値
XML	デフォルト値はありません
特殊タイプ	特殊タイプの対応するソース・タイプのデフォルト値

INCLUDING COLUMN DEFAULTS を指定する場合は、USING TYPE DEFAULTS は指定しないでください。

INCLUDING IMPLICITLY HIDDEN COLUMN ATTRIBUTES または EXCLUDING IMPLICITLY HIDDEN COLUMN ATTRIBUTES

暗黙的な隠し列を継承するかどうかを指定します。

INCLUDING IMPLICITLY HIDDEN COLUMN ATTRIBUTES

この表が、選択ステートメント、表名、または ビュー名から暗黙的な隠し列を継承することを示します。また、これらの列は、新規の表の暗黙的隠し属性を使用して定義されます。

INCLUDING IMPLICITLY HIDDEN COLUMN ATTRIBUTES を指定しない場合、表は、暗黙的な隠し列を持ちません。

EXCLUDING IMPLICITLY HIDDEN COLUMN ATTRIBUTES

表が、全選択、表名、または ビュー名 から、暗黙的な隠し列を継承しないことを指定します。

INCLUDING ROW CHANGE TIMESTAMP COLUMN ATTRIBUTES または EXCLUDING ROW CHANGE TIMESTAMP COLUMN ATTRIBUTES

行変更タイム・スタンプ属性が継承されるかどうかを指定します。

INCLUDING ROW CHANGE TIMESTAMP COLUMN ATTRIBUTES

この表が、選択ステートメント、表名、またはビュー名 の結果として生じる列の行変更タイム・スタンプ属性 (もしあれば) を継承することを指定します。一般に、行変更タイム・スタンプ属性がコピーされるのは、表、ビュー、または選択ステートメント の中の対応する列のエレメントが、行変更タイム・スタンプ属性を指定する基本表列の名前に直接または間接にマップされる表列またはビュー列の名前である場合です。 INCLUDING ROW COLUMN ATTRIBUTES 文節と AS 選択ステートメント 文節を指定してあるときは、以下の場合には新規の表の列は、行変更タイム・スタンプ属性を継承しません。

- 選択ステートメント の選択リストに、行変更タイム・スタンプ列名の複数インスタンスが含まれている (つまり同じ列を複数回選択している) 場合。
- 選択ステートメント の選択リストに、複数の行変更タイム・スタンプ列が含まれている (つまり、結合が複数の行変更タイム・スタンプ列を戻した) 場合。
- 選択リスト内の式に行変更タイム・スタンプ列が含まれている場合。
- 選択ステートメント に一組の演算 (UNION または INTERSECT) が含まれている場合。

INCLUDING ROW CHANGE TIMESTAMP COLUMN ATTRIBUTES を指定しない場合、表は、行変更タイム・スタンプ列を持ちません。

EXCLUDING ROW CHANGE TIMESTAMP COLUMN ATTRIBUTES
この表が、全選択、表名、またはビュー名 の結果として生じる列の行変更タイム・スタンプ属性 (もしあれば) を継承しないことを指定します。

as-result-table

column-name
表の列の名前を指定します。列名のリストを指定する場合は、そのリストは、選択ステートメント の結果表にある列の数と同じ数の列名で構成されている必要があります。 各 *column-name* (列名) は、ユニークで、しかも非修飾でなければなりません。列名のリストを指定しなかった場合、表の列は選択ステートメント の結果表の列の名前を継承します。

選択ステートメント の結果表に重複する列名または無名列がある場合は、列名のリストを指定する必要があります。無名列は、定数、関数、式、またはセット演算 (UNION または INTERSECT) から生じる名前のない列で、この列には選択リストの AS 文節が使用されます。

FOR COLUMN システム列名
列の IBM i 名を指定します。表の複数の列やシステム列名に対して、同じ名前を使用してはなりません。

システム列名 が指定されず、また列名が有効なシステム列名 でない場合には、システム列名が生成されます。システム列名の生成方法に関する詳細については、 1298 ページの『表名の生成の規則』を参照してください。

選択ステートメント
表の列の名前および記述を、選択ステートメント を実行した場合に選択ステートメント の派生結果表に現れる列と同じにすることを指定します。 AS (選択ステートメント) を使用すると、この表について *n* 個の列を暗黙的に定義したことになります。*n* は、選択ステートメント の結果として発生する列の数です。

この暗黙の定義には、*n* 列の以下の属性が含まれます (そのデータ・タイプに該当する場合):

- 列名 (および、システム列名)
- データ・タイプ、長さ、精度、および位取り
- CCSID
- NULL 可能性

- FIELDPROC
- 列の見出しとテキスト (1570 ページの『LABEL』を参照)

以下の属性は組み込まれません (一部の属性は、*copy-options* を使用して組み込むことができます)。

- デフォルト値
- 非表示属性
- 識別属性
- 行変更タイム・スタンプ属性
- 行開始、行終了、およびトランザクション開始 ID
- 生成式

暗黙の定義には、識別された表またはビューのその他のオプション属性は含まれません。例えば、新規の表には、表からの基本キーや外部キーは自動的には組み込まれません。新規の表にこうしたオプション属性が組み込まれるのは、オプション文節を明示的に指定した場合に限られます。

選択文節 に含まれる列で、別の表またはビュー内の列への直接参照であるか、結果属性を変更するための CAST のみを使用している列には、ファイル・オブジェクト内の定義のために生成される REFFLD 情報が入ります。

select-statement は変数を参照してはなりませんが、グローバル変数を参照することはできます。

選択ステートメント には、PREVIOUS VALUE 式または NEXT VALUE 式を含めてはなりません。UPDATE、SKIP LOCKED DATA、USE AND KEEP EXCLUSIVE LOCKS の各文節を指定することはできません。

select-statement が *isolation-clause* を含む場合、*isolation-clause* で指定された分離レベルが SQL ステートメント全体に適用されます。

WITH DATA
選択ステートメント を実行することを指定します。表の作成後に、選択ステートメント の結果表の行が自動的に表に挿入されます。

WITH NO DATA
選択ステートメント を新しい表の属性を定義する目的のみに使用することを指定します。選択ステートメント の結果を使用した表へのデータの挿入は行われません。

unique-constraint

CONSTRAINT *constraint-name*
制約の名前を指定します。制約名 は、既に CREATE TABLE ステートメントで指定され、かつ既に現行サーバーに存在している制約を示すものであってはなりません。

この文節の指定がない場合、固有制約の名前がデータベース・マネージャーによって生成されます。

PRIMARY KEY (*column-name***,...)**
指定した列で構成される基本キーを定義します。表は基本キーを 1 つだけ持つことができます。したがって、この文節は複数回指定することはできず、またこの簡便な手法が表の基本キーを定義するのに使用されていた場合には、指定する

ことはできません。指定する列は、その CREATE TABLE ステートメントで前に指定した他の UNIQUE 制約で指定されている列と同じであってはなりません。例えば、UNIQUE(B,A) が既に指定されている場合、PRIMARY KEY(A,B) の指定は許されません。

それぞれの *column-name* は、該当の表の列を識別する非修飾の名前でなければなりません。同じ列を複数回指定することはできません。この列は、LOB 列、DATALINK 列、または XML 列であってはなりません。ソート・シーケンスを指定する場合、列にフィールド・プロシージャーを含めることはできません。指定できる列の数は 120 を超えてはならず、それぞれのバイト・カウントの合計は 32766-*n* を超えてはなりません。ここで、*n* は NULL が許されると指定された列の数です。バイト数については、 1295 ページの表 83 を参照してください。

固有索引は、別個のシステム論理ファイルとしてではなく、システム物理ファイルの一部として作成されます。基本キーを追加すると、CHECK 制約が暗黙的に追加され、その基本キーを構成するどの列でも NULL を使用することはできないという規則が適用されます。

UNIQUE (*column-name***,...)**

識別された列で構成されるユニーク制約を定義します。 UNIQUE 文節は複数回指定しても構いません。指定する列は、その CREATE TABLE ステートメントで前に指定した他の UNIQUE 制約や PRIMARY KEY で指定されている列と同じであってはなりません。ある固有制約が他の制約の指定と同一であるか否かを判別するには、その列のリストを対比します。例えば、UNIQUE(A,B) は UNIQUE(B,A) と同一です。

それぞれの column-name は、該当の表の列を識別する非修飾の名前でなければなりません。同じ列を複数回指定することはできません。この列は、LOB 列、DATALINK 列、または XML 列であってはなりません。ソート・シーケンスを指定する場合、列にフィールド・プロシージャーを含めることはできません。指定できる列の数は 120 を超えてはならず、それぞれのバイト・カウントの合計は 32766-*n* を超えてはなりません。ここで、*n* は NULL が許されると指定された列の数です。バイト数については、 1295 ページの表 83 を参照してください。

指定された列に基づく固有索引が、その CREATE TABLE ステートメントの実行過程で作成されます。固有索引は、別個のシステム論理ファイルとしてではなく、システム物理ファイルの一部として作成されます。

referential-constraint

CONSTRAINT *constraint-name*

制約の名前を指定します。制約名 は、既に CREATE TABLE ステートメントで指定され、かつ既に現行サーバーに存在している制約を示すものであってはなりません。

この文節の指定がない場合、固有制約の名前がデータベース・マネージャーによって生成されます。

FOREIGN KEY

FOREIGN KEY 文節を指定するたびに、参照制約が定義されます。表が宣言済みグローバル一時表または分散表である場合、FOREIGN KEY を使用することはできません。

(*column-name***,...)**

参照制約の外部キーは、指定した列で構成されます。それぞれの *column-name* は、該当の表の列を識別する非修飾の名前でなければなりません。同じ列を複数回指定することはできません。この列は、LOB 列、DATALINK 列、または XML 列であってはならず、また行変更タイム・スタンプ列であってもなりません。ソート・シーケンスを指定する場合、列にフィールド・プロシージャーを含めることはできません。指定できる列の数は 120 を超えてはならず、その長さの合計は 32766-*n* を超えてはなりません。ここで、*n* は NULL が許されると指定された列の数です。

REFERENCES *table-name*

REFERENCES 文節で指定する表名 は、アプリケーション・サーバー上に作成される表か既に存在している基本表を示すものでなければなりませんが、カタログ表、宣言済み一時表、分散表、または履歴表を示すものであってはなりません。親がパーティション表の場合は、親のユニーク制約を適用するユニーク索引がパーティション化されていてはなりません。

参照制約の外部キー、親キー、および親表が、前に指定した参照制約の外部キー、親キー、および親表と同一である場合は、その参照制約は重複 します。重複する参照制約は許されますが、お勧めできません。

以下の説明で、T1 は作成される表を指し、T2 は識別された親表を表しています。

指定した外部キーは、T2 の親キーと同じ数の列を持たなければなりません。外部キーの *n* 番目の列の記述とその親キーの *n* 番目の列の記述は、同一のデータ・タイプ、長さ、CCSID、および FIELDPROC を持たなければなりません。

(*column-name***,...)**

参照制約の親キーは、ここで指定する列によって構成されます。各 *column-name* は、T2 の列を指定する非修飾名でなければなりません。同じ列を複数回指定することはできません。この列は、LOB 列、DATALINK 列、または XML 列であってはならず、また行変更タイム・スタンプ列であってもなりません。ソート・シーケンスを指定する場合、列にフィールド・プロシージャーを含めることはできません。指定できる列の数は 120 を超えてはならず、それぞれのバイト・カウントの合計は 32766-*n* を超えてはなりません。ここで、*n* は NULL が許されると指定された列の数です。バイト数については、 1295 ページの表 83 を参照してください。

この列名のリストは、T2 の基本キーまたは T2 に存在する UNIQUE 制約の列名のリストと同一でなければなりません。 名前は、基本キーと同じ順序で指定する必要はありません。しかし、外部キー 文節の列のリストに対応する順序で指定する必要があります。列名のリストの指定がない場合、T2 は基本キーを持たなければなりません。列名のリストの省略は、基本キーの列の暗黙の指定を意味しています。

FOREIGN KEY 文節によって指定される参照制約は、T2 が親表で、T1 がその従属表である関係を定義します。

ON DELETE
親表の行が削除される時点で、従属表について行うアクションを指定します。可能なアクションには以下の 5 つがあります。

- NO ACTION (デフォルト)
- RESTRICT
- CASCADE
- SET NULL
- SET DEFAULT

外部キーの列に NULL が許される列がある場合を除いて、SET NULL を指定してはなりません。

FILE LINK CONTROL を指定した DATALINK 列が T1 に含まれる場合には、CASCADE を指定してはなりません。

削除規則は、T2 の行が DELETE または波及削除操作の対象で、しかもその行が T1 に従属する行を持っている場合に適用されます。 *p* は、そのような T2 の行を表すと想定します。

- RESTRICT または NO ACTION を指定した場合、エラーが生じ、行の削除は行われません。
- CASCADE を指定すると、 T1 の *p* の従属行に削除操作が伝搬します。
- SET NULL が指定された場合、T1 の *p* のそれぞれの従属行の外部キーの NULL 可能な列が NULL 値に設定されます。従属表がパーティション化された表である場合や外部キーの列がパーティション・キーの場合も、SET NULL を使用することはできません。
- SET DEFAULT を指定した場合、T1 の *p* の各従属行の外部キーの各列は、そのデフォルト値に設定されます。SET DEFAULT は、従属表がパーティション化された表であり、外部キー列もパーティション・キーである場合は、デフォルトで同じパーティションに行が保持される場合を除いて、使用することはできません。

ON UPDATE
親表の行が更新される時点で、従属表で行うアクションを指定します。

更新規則は、T2 の行が UPDATE または波及更新操作の対象で、しかもその行が T1 に従属行を持つ場合に適用されます。 *p* は、そのような T2 の行を表すと想定します。

- RESTRICT または NO ACTION を指定した場合、エラーが生じ、行の更新は行われません。

check-constraint

CONSTRAINT *constraint-name*
検査制約の名前を指定します。制約名 は、既に CREATE TABLE ステートメントで指定され、かつ既に現行サーバーに存在している制約を示すものであってはなりません。

この文節の指定がない場合、固有制約の名前がデータベース・マネージャーによって生成されます。

CHECK (*check-condition*)
検査制約を定義します。どのような場合も、検査条件 は、表の行ごとに真か不明にする必要があります。

検査条件 は、検索条件 です。ただし、次の条件は除きます。

- 表の列だけを参照することができます。
- *check-condition* で指定する式の結果を、ROWID、XML の各データ・タイプ、または FILE LINK CONTROL を伴う DATALINK データ・タイプにすることはできません。
- 次のいずれも含めることはできません。
  - 副照会
  - 集約関数
  - 変数
  - グローバル変数
  - パラメーター・マーカー
  - シーケンス参照
  - LOB を含む複合式 (連結など)
  - OLAP の指定
  - ROW CHANGE 式
  - IS JSON、JSON_EXISTS、または REGEXP_LIKE の述部
  - 特殊レジスター
  - 特殊タイプの作成に伴って暗黙に生成された関数以外のユーザー定義関数
  - 以下の組み込みスカラー関数:

ATAN2	DLURLSCHEME	JSON_TO_BSON	ROUND_TIMESTAMP
BSON_TO_JSON	DLURLSERVER	JSON_VALUE	RPAD
CARDINALITY	DLVALUE	LOCATE_IN_STRING	SCORE
CONTAINS	ENCRYPT_AES	LPAD	SOUNDEX
CURDATE	ENCRYPT_RC2	MAX_CARDINALITY	TABLE_NAME
CURTIME	ENCRYPT_TDES	MONTHNAME	TABLE_SCHEMA
DATAPARTITIONNAME	GENERATE_UNIQUE	MONTHS_BETWEEN	TIMESTAMP_FORMAT
DATAPARTITIONNUM	GETHINT	NEXT_DAY	TIMESTAMPDIFF
DAYNAME	HASH	NOW	TRUNC_TIMESTAMP
DBPARTITIONNAME	HASH_MD5	OVERLAY	VARCHAR_FORMAT
DECRYPT_BINARY	HASH_SHA1	RAISE_ERROR	VERIFY_GROUP_FOR_USER
DECRYPT_BIT	HASH_SHA256	RAND	WEEK_ISO
DECRYPT_CHAR	HASH_SHA512	REGEXP_COUNT	WRAP
DECRYPT_DB	IDENTITY_VAL_LOCAL	REGEXP_INSTR	XMLPARSE
DIFFERENCE	INSERT	REGEXP_REPLACE	XMLVALIDATE
DLURLCOMPLETE [1]	JSON_ARRAY	REGEXP_SUBSTR	XSLTRANSFORM
DLURLPATH	JSON_OBJECT	REPEAT	
DLURLPATHONLY	JSON_QUERY	REPLACE	

[1] FILE LINK CONTROL と READ PERMISSION DB の属性が指定されたデータ・リンクの場合。

ON INSERT VIOLATION
挿入される行の *check-condition* が false の場合の処置を指定します。この節が指定されていないと、挿入の *check-condition* が false の場合はエラーが発生します。

SET *column-name* **= DEFAULT**
挿入操作で提供される値の代わりに、*column-name* のデフォルト値が表に挿入されます。

column-name は *check-condition* 内で参照されている必要があります。

ON UPDATE VIOLATION
更新される行に対する *check-condition* が false の場合の処置を指定します。この節が指定されていないと、更新の *check-condition* が false の場合はエラーが発生します。

PRESERVE *column-name*
column-name の現行値は、更新操作によって提供される値で置き換えられるのではなく、表に残ります。

column-name は *check-condition* 内で参照されている必要があります。

検索条件の詳細については、275 ページの『検索条件』を参照してください。
LOB データ・タイプおよび式が含まれる検査制約について詳しくは、「データベース・プログラミング」トピック集を参照してください。

NOT LOGGED INITIALLY

このステートメントによって作成された表に対して同一作業単位内の INSERT、DELETE、または UPDATE ステートメントによって行われた変更は、ログ (ジャーナル) に記録されません。

現行作業単位の完了時に NOT LOGGED INITIALLY 属性が非活動化され、後続の作業単位で表に対して行われるすべての操作はログ (ジャーナル) に記録されます。

この NOT LOGGED INITIALLY オプションは、代替ソース (別の表またはファイル) のデータを使用して大きい結果セットを作成する必要があり、かつ表のリカバリーが不要である場合に役立ちます。このオプションを使用すると、データのロギング (ジャーナリング) のオーバーヘッドを節約できます。

表に FILE LINK CONTROL が指定された DATALINK 列 がある場合、ACTIVATE NOT LOGGED INITIALLY は無視されます。

VOLATILE または NOT VOLATILE

表表名 のカーディナリティーを実行時に大きく変えることができるかをオプティマイザーに示します。揮発性は表の行数に適用され、表そのものに適用されるわけではありません。デフォルトは NOT VOLATILE です。

VOLATILE
実行時に表名 のカーディナリティーを空から大規模に大きく変えることができることを指定します。オプティマイザーは表にアクセスするとき、可能であれば通常は索引を使用します。

NOT VOLATILE
table-name のカーディナリティーが揮発性でないことを指定します。この表を参照するアクセス・プランは、アクセス・プランが構築された時点の表のカーディナリティーに基づいたものになります。 NOT VOLATILE がデフォルトです。

RCDFMT

表のレコード・フォーマット名を示します。

RCDFMT *format-name*
表の IBM i レコード・フォーマット名を指定する非修飾名です。 *format-name* は、システム ID です。

レコード・フォーマット名が指定されない場合、*format-name* は、表の *system-object-name* と同一のものになります。

media-preference

表またはパーティションの優先ストレージ・メディアを指定します。

UNIT ANY
どのストレージ・メディアも優先しません。使用可能なストレージ・メディアであればどのストレージ・メディアからでも表またはパーティションのストレージが割り振られます。表で UNIT ANY を指定すると、パーティションで指定する *media-preference* が使用されます。

UNIT SSD
ソリッド・ステート・ディスク・ストレージ・メディアを優先します。ソリッド・ステート・ディスクのストレージ・メディアが使用可能になっていれば、そのソリッド・ステート・ディスクのストレージ・メディアから表またはパーティションのストレージが割り振られます。表で UNIT SSD を指定すると、パーティションで指定する *media-preference* は無視されます。

memory-preference

KEEP IN MEMORY
表のデータが照会で使用されるときに、データを主記憶域プールに入れるかどうかを指定します。

NO データは主記憶域プールに入れられません。

YES
データは主記憶域プールに入れられます。

ON REPLACE

現行サーバーに存在する表を置き換える場合の処置を指定します。既存の表が置換されない場合、このオプションは無視されます。

PRESERVE ALL ROWS
指定された表の現在の行が保持されます。WITH DATA が *result-table-as* で指定されている場合、PRESERVE ALL ROWS は使用できません。

パーティション化された表内のすべてのパーティションのすべての行が保持されます。新しい表定義が範囲パーティション化された表の場合、定義された範囲は既存のパーティションのすべての行を含むことができなければなりません。

列が除去された場合、その列値は保持されません。列が変更された場合、その列値を変更することができます。

表がパーティション化された表ではない場合、またはハッシュ・パーティション化された表である場合、PRESERVE ALL ROWS と PRESERVE ROWS は等価です。

PRESERVE ROWS
指定された表の現在の行が保持されます。WITH DATA が *result-table-as* で指定されている場合、PRESERVE ROWS は使用できません。

範囲パーティション化された表のパーティションが除去された場合、そのパーティションの行は、削除トリガーを処理せずに削除されます。範囲パーティション化された表のパーティションが除去されるかどうかを判別するために、新しい表定義内のパーティションの範囲定義とパーティション名 (もしあれば) が既存の表定義のパーティションと比較されます。パーティションの指定された範囲または名前のいずれかが一致すると、それが保持されます。パーティションに *partition-name* がない場合、その *boundary-spec* は、既存のパーティションと一致しなければなりません。

ハッシュ・パーティション表のパーティションが除去された場合、そのパーティションの行は保持されます。

列が除去された場合、その列値は保持されません。列が変更された場合、その列値を変更することができます。

DELETE ROWS
指定された表の現在の行が削除されます。既存の DELETE トリガーは起動されません。

表がシステム期間テンポラル表または履歴表の場合、DELETE ROWS は使用できません。

distribution-clause

IN NODEGROUP *nodegroup-name*
この表のデータが分散されるノード・グループを指定します。この名前は、現行サーバーに存在するノード・グループを示すものでなければなりません。この文節を指定すると、表は、そのノード・グループのシステムすべてにわたる分散表として作成されます。

LOB 列、DATALINK 列、XML 列、または IDENTITY 列は、分散表内で使用することはできません。

分散表を作成するには、Db2 Multisystem・プロダクトをインストールする必要があります。分散表の詳細については、「Db2 UDB for iSeries マルチ・システム」トピック集を参照してください。

DISTRIBUTE BY HASH (*column-name***,...)**
パーティション・キーを指定します。パーティション・キーは、ノード・グループのどのノードに行を置くかを判別するために使用します。それぞれの *column-name* は、該当の表の列を識別する非修飾の名前でなければなりません。同じ列を複数回指定することはできません。 DISTRIBUTE BY 文節の指定がない場合、基本キーの最初の列が、パーティション・キーとして使用されます。基本キーがない場合は、表の最初の列で、浮動小数点、日付、時刻、あるいはタイム・スタンプではない列が、パーティション・キーとして使用されます。

パーティション・キーを構成する列は、その表に対して固有の制約を構成する列のサブセットでなければなりません。パーティション・キーでは、浮動小数点、日付、時刻、タイム・スタンプ、LOB、XML、DataLink、 ROWID、および、フィールド・プロシージャーがある列を使用することはできません。

partitioning-clause

PARTITION BY RANGE
表に行を挿入する場合、ターゲット・データのパーティションの判別に列値の範囲を使用することを指定します。パーティションの数は、256 以下でなければなりません。

partition-expression
データのターゲット・データ・パーティションを決定するために範囲を定義する対象のキー・データを指定します。

column-name
データ・パーティション・キーの列を識別します。パーティション・キーは、表内のどのパーティションに行を置くかを判別するために使用します。 *column-name* は、該当の表の列を識別する非修飾の名前でなければなりません。同じ列を複数回指定することはできません。

パーティション・キーでは、LOB、XML、DataLink、ROWID、行変更タイム・スタンプ列、ID 列、および、フィールド・プロシージャーのある列を使用することはできません。

指定する列数は 120 を超えてはなりません。列の長さ属性の合計が 2000 を超えてはなりません。

NULLS LAST
NULL 値の比較順位を上位に設定することを指定します。

NULLS FIRST
NULL 値の比較順位を下位に設定することを指定します。

partition-element
データ・パーティション・キーの範囲を指定します。

PARTITION *partition-name*
データ・パーティションに名前を付けます。表の他のデータ・パーティションと同じ名前を指定することはできません。

この文節の指定がない場合、固有のパーティション名がデータベース・マネージャーによって生成されます。

boundary-spec
範囲パーティションの境界を指定します。複数のパーティション・キーが指定された場合、境界は昇順で指定されなければなりません。範囲はオーバーラップしてはなりません。

starting-clause
データ・パーティションの範囲の下限を指定します。指定された開始値の数値は、パーティション化キーの列の値と同じでなければなりません。最初の *boundary-spec* で *starting-clause* を指定しない場合は、パーティション・キーの各列で MINVALUE INCLUSIVE がデフォルトとして使用されます。後続の *boundary-spec* で

starting-clause を指定しない場合は、直前の隣接する *boundary-spec* に *ending-clause* を組み込む必要があります。デフォルトは、その *ending-clause* と同じです (ただし、INCLUSIVE 属性または EXCLUSIVE 属性が逆になります)。

STARTING FROM
この後に、*starting-clause* を指定します。

constant
パーティション・キーの対応する列のデータ・タイプの定数の規則に準拠する定数を指定します。この定数は、パーティション・キーの対応する列が特殊タイプでない場合は、特殊タイプのソース・タイプの規則に準拠する必要があります。この値は、表の他の境界スペック の範囲内であってはなりません。

MINVALUE
対応する *column-name* のデータ・タイプで有効な最小値より小さい値を指定します。MINVALUE を指定した場合、開始文節 内の後続のすべての値も MINVALUE でなければなりません。

MAXVALUE
対応する *column-name* のデータ・タイプで有効な最大値より大きい値を指定します。MAXVALUE を指定した場合、終了文節 内の後続のすべての値も MAXVALUE でなければなりません。

INCLUSIVE
指定した範囲値をデータ・パーティションに含めることを示します。

EXCLUSIVE
指定した範囲値をデータ・パーティションから除外することを示します。MINVALUE または MAXVALUE が指定されている場合、この指定は無視されます。

ending-clause
データ・パーティションの範囲の上限を指定します。指定された終了値の数値は、データ・パーティション化キーの列の値と同じでなければなりません。最後の *boundary-spec* では、*ending-clause* を指定する必要があります。前の *boundary-spec* で *ending-clause* を指定しない場合は、後続の隣接する *boundary-spec* に *starting-clause* を組み込む必要があります。デフォルトは、その *starting-clause* と同じです (ただし、INCLUSIVE 属性または EXCLUSIVE 属性が逆になります)。

ENDING AT
この後に、*ending-clause* を指定します。

constant
パーティション・キーの対応する列のデータ・タイプの定数の規則に準拠する定数を指定します。この定数は、パーティション・キーの対応する列が特殊タイプでない場合は、特殊

タイプのソース・タイプの規則に準拠する必要があります。この値は、表の他の境界スペック の範囲内であってはなりません。

MINVALUE
対応する *column-name* のデータ・タイプで有効な最小値より小さい値を指定します。MINVALUE を指定した場合、開始文節 内の後続のすべての値も MINVALUE でなければなりません。

MAXVALUE
対応する *column-name* のデータ・タイプで有効な最大値より大きい値を指定します。MAXVALUE を指定した場合、終了文節 内の後続のすべての値も MAXVALUE でなければなりません。

INCLUSIVE
指定した範囲値をデータ・パーティションに含めることを示します。

EXCLUSIVE
指定した範囲値をデータ・パーティションから除外することを示します。MINVALUE または MAXVALUE が指定されている場合、この指定は無視されます。

EVERY 整数定数
複数のデータ・パーティションが、整数定数 が各データ・パーティションの範囲の幅を指定する場所に追加されることを指定します。
EVERY が指定された場合、パーティション・キーに指定できるのは単一の SMALLINT、INTEGER、
BIGINT、DECIMAL、NUMERIC、DATE、または TIMESTAMP 列のみです。

最初のデータ・パーティションの開始値が、指定された STARTING 値になります。以前のパーティション + 整数定数 の開始値が、後続の各パーティションの開始値になります。開始文節 が EXCLUSIVE を指定した場合、各パーティションの開始値は EXCLUSIVE になります。そうでない場合、各パーティションの開始値は INCLUSIVE になります。

範囲の各パーティションの終了値は (パーティションの開始値 + 整数定数) になります。 *ending-clause* が EXCLUSIVE を指定した場合、最後のパーティションの終了値は EXCLUSIVE になります。そうでない場合、最後のパーティションの終了値は INCLUSIVE になります。開始値が INCLUSIVE の場合、その他のパーティションの終了値は EXCLUSIVE になります。そうでない場合、その他のパーティションの終了値は INCLUSIVE になります。

追加するパーティションの数は、 ENDING 値に達するまで整数定数を繰り返し STARTING 値に追加することによって判別されます。例えば次のようなものがあります。

```
CREATE TABLE FOO
  (A  INT)
  PARTITION BY RANGE(A)
    (STARTING(1) ENDING(10) EVERY(2))
```

上記は、次の CREATE TABLE ステートメントと等価です。

```
CREATE TABLE FOO
  (A  INT)
  PARTITION BY RANGE(A)
    (STARTING(1) ENDING(2),
     STARTING(3) ENDING(4),
     STARTING(5) ENDING(6),
     STARTING(7) ENDING(8),
     STARTING(9) ENDING(10))
```

日付およびタイム・スタンプの場合、EVERY 値をラベル付き期間にする必要があります。例えば、次のようになります。

```
CREATE TABLE FOO
  (A  DATE)
  PARTITION BY RANGE(A)
    (STARTING('2001-01-01') ENDING('2010-01-01') EVERY(3 MONTHS))
```

PARTITION BY HASH
表に行を挿入する場合、ターゲット・データのパーティションの判別にハッシュ関数を使用することを指定します。

(*column-name***,...)**
パーティション・キーを指定します。パーティション・キーは、表内のどのパーティションに行を置くかを判別するために使用します。それぞれの *column-name* は、該当の表の列を識別する非修飾の名前でなければなりません。同じ列を複数回指定することはできません。

パーティション・キーでは、浮動小数点、LOB、XML、日付、時刻、タイム・スタンプ、DataLink、ROWID、ID 列、および、フィールド・プロシージャーのある列を使用することはできません。

INTO 整数 PARTITIONS
パーティションの数を指定します。パーティションの数は、256 以下でなければなりません。

materialized-query-definition

column-name
表の列の名前を指定します。列名のリストを指定する場合は、そのリストは、選択ステートメント の結果表にある列の数と同じ数の列名で構成されている必要があります。 各 *column-name* (列名) は、ユニークで、しかも非修飾でなければなりません。列名のリストを指定しなかった場合、表の列は選択ステートメント の結果表の列の名前を継承します。

選択ステートメント の結果表に重複する列名または無名列がある場合は、列名のリストを指定する必要があります。無名列は、定数、関数、式、またはセット演算 (UNION または INTERSECT) から生じる名前のない列で、この列には選択リストの AS 文節が使用されます。

FOR COLUMN システム列名
列の IBM i 名を指定します。表の複数の列やシステム列名に対して、同じ名前を使用してはなりません。

システム列名 が指定されず、また列名が有効なシステム列名 でない場合には、システム列名が生成されます。システム列名の生成方法に関する詳細については、 1298 ページの『表名の生成の規則』を参照してください。

選択ステートメント
表の列の名前および記述を、選択ステートメント を実行した場合に選択ステートメント の派生結果表に現れる列と同じにすることを指定します。 AS (選択ステートメント) を使用すると、この表について *n* 個の列を暗黙的に定義したことになります。*n* は、選択ステートメント の結果として発生する列の数です。

この暗黙の定義には、*n* 列の以下の属性が含まれます (そのデータ・タイプに該当する場合):

- 列名 (および、システム列名)
- データ・タイプ、長さ、精度、および位取り
- CCSID
- NULL 可能性
- FIELDPROC
- 列の見出しとテキスト (1570 ページの『LABEL』を参照)

以下の属性は組み込まれません。

- デフォルト値
- 非表示属性
- 識別属性
- 行変更タイム・スタンプ属性
- 行開始、行終了、およびトランザクション開始 ID
- 生成式

暗黙の定義には、識別された表またはビューのその他のオプション属性は含まれません。例えば、新規の表には、表からの基本キーや外部キーは自動的には組み込まれません。新規の表にこうしたオプション属性が組み込まれるのは、オプション文節を明示的に指定した場合に限られます。

選択ステートメント は、変数、グローバル変数、または組み込みパラメーター・マーカーを参照するものであってはなりません。*select-statement* の SELECT 文節の式が列名でない場合は、フィールド・プロシージャーがある列をその式で参照することはできません。

選択ステートメント には、PREVIOUS VALUE 式または NEXT VALUE 式を含めてはなりません。UPDATE、SKIP LOCKED DATA、USE AND KEEP EXCLUSIVE LOCKS の各文節を指定することはできません。

refreshable-table-options
表がマテリアライズ照会表 であり、REFRESH TABLE ステートメントを使用して選択ステートメント の結果を表に移植することを指定します。

選択ステートメント が GROUP BY 文節を含むマテリアライズ照会表は、選択ステートメント で参照された表からのデータを要約します。このようなマテリアライズ照会表は、サマリー表と呼ばれます。要約表は、特殊なタイプのマテリアライズ照会表です。

マテリアライズ照会表が定義されると、以下の選択ステートメント 制限が適用されます。

- 選択ステートメント には、別のマテリアライズ照会表の参照またはマテリアライズ照会表を参照するビューの参照を含めることはできません。
- 選択ステートメント には、宣言済み一時表、QTEMP 内の表、プログラム記述ファイル、または FROM 文節の非 SQL 論理ファイルへの参照を含めることはできません。
- 選択ステートメントには、データ変更ファイル参照 は含められません。
- 選択ステートメント に、別のマテリアライズ照会表または宣言済み一時表を参照するビューの参照を含めることはできません。ENABLE QUERY OPTIMIZATION を指定してマテリアライズ照会表が定義されると、選択ステートメント には、次の段落で示す制約事項のうちのいずれかを含むビューの参照を含めることはできません。
- 選択ステートメント には、 DataLink または DataLink が FILE LINK CONTROL である DataLink に基づく特殊タイプがある式を含めることはできません。
- 選択ステートメント には、精度、DBCS-ONLY、または DBCS-EITHER を持つバイナリーといった、 SQL データ・タイプではない結果列を含めることはできません。

ENABLE QUERY OPTIMIZATION でマテリアライズ照会表が定義されると、以下の付加的な選択ステートメント 制限が適用されます。

- 特殊レジスターを含めてはなりません。
- 他の非 deterministic 関数を含めてはなりません。
- ORDER BY 文節を使用できますが、REFRESH によってのみ使用されます。これによって、マテリアライズ照会表のデータの参照の局所性が改善される場合があります。
- 副選択がビューを参照する場合、ビュー定義内の選択ステートメント は、これらの制約事項を満たさなければなりません。

DATA INITIALLY DEFERRED
データの作成時に、そのデータをマテリアライズ照会表に挿入しないことを指定します。 REFRESH TABLE ステートメントを使用してマテリアライズ照会表を移植するか、 INSERT ステートメントを使用してデータをマテリアライズ照会表に挿入します。

DATA INITIALLY IMMEDIATE
データの作成時に、そのデータをマテリアライズ照会表に挿入することを指定します。

REFRESH DEFERRED
表内のデータを REFRESH TABLE ステートメントを使用していつでもリフレッシュできるように指定します。表内のデータは、REFRESH TABLE ステートメントの処理時または最後に更新された時のスナップショットとしての照会の結果のみを反映します。

MAINTAINED BY USER
マテリアライズ照会表がユーザーによって保守されるように指定します。ユ

ーザーは表に対して INSERT、DELETE、UPDATE、または REFRESH TABLE ステートメントを使用できます。

ENABLE QUERY OPTIMIZATION または **DISABLE QUERY OPTIMIZATION**
このマテリアライズ照会表を最適化に使用できるかどうかを指定します。デフォルトは ENABLE QUERY OPTIMIZATION です。

ENABLE QUERY OPTIMIZATION
マテリアライズ照会表を照会の最適化に使用できるように指定します。指定した選択ステートメント が照会の最適化のための制約事項を満たしていない場合は、エラーが戻されます。

DISABLE QUERY OPTIMIZATION
マテリアライズ照会表を照会の最適化に使用できないように指定します。それでもその表を直接照会することはできます。

select-statement 内で直接または間接に参照されているいずれかの表で行レベルまたは列レベルのアクセス制御がアクティブになっている場合、作成される表に対して行アクセス制御が暗黙的にアクティブ化されます。これにより、マテリアライズ照会表の内容への直接アクセスが制限されます。表を明示的に参照する照会では、表内にデータがないことを示す警告が戻されます。マテリアライズ照会表にアクセスするには、適切な行権限を作成します。またはふさわしいようであれば、マテリアライズ照会表に対する ALTER TABLE DEACTIVATE ROW ACCESS CONTROL を発行して、行レベルの保護を除去します。

注

表の属性 **:** 表は物理ファイルとして作成されます。表が作成される場合、ファイル待ち時間とレコード待ち時間の属性は、物理ファイル作成 (CRTPF) コマンドの WAITFILE キーワードと WAITRCD キーワード上に指定されたデフォルト値に設定されます。

SQL 表は、削除済みの行で使用していたスペースがその後の挿入要求で再利用されるように作成されます。この属性は、コマンドの *CHGPF*、および *REUSEDLT(*NO)* パラメーターの指定によって変更することができます。 *CHGPF* コマンドについて詳しくは、「CL 解説書」を参照してください。

分散表は、その表が配布されるすべてのサーバーで作成されます。 分散表についての詳細は、Db2 マルチシステム (Multisystem) を参照してください。

表のジャーナリング**:** 表の作成時に、ジャーナリングを自動的に開始することができます。

- QDFTJRN と呼ばれるデータ域が表が作成されたのと同じスキーマに存在し、ユーザーがそのデータ域に対する権限を持っていると、以下のいずれかが当てはまる場合、ジャーナリングがデータ域で指定されたジャーナルに対して開始されます。
  - 表の識別されたスキーマは、QSYS、QSYS2、QRECOVERY、QSPL、QRCL、QRPLOBJ、QGPL、QTEMP、SYSIBM、またはこれらのライブラリーと同等の iASP であってはなりません。

  - データ域で指定されたジャーナルが存在していなければならず、ユーザーはジャーナルに対してジャーナリングを開始する権限を持っていなければなりません。
  - データ域の最初の 10 バイトには、ジャーナルを検索するスキーマの名前が含まれている必要があります。
  - 次の 10 バイトにはジャーナルの名前が含まれている必要があります。
  - 残りのバイトには、暗黙的にジャーナルに記録されるオブジェクト・タイプと、いつ暗黙的ジャーナリングを行うかに関係のあるオプションが含まれます。オブジェクト・タイプには値 *FILE または *ALL を含める必要があります。値 *NONE を使用して、ジャーナリングが開始しないようにすることができます。

  詳しくは、ジャーナル管理を参照してください。
- 指定した (STRJRNLIB コマンドを使用して) スキーマ内に表を作成すると、ジャーナリングが暗黙的に開始されます。
- QDFTJRN と呼ばれるデータ域が表の作成時と同一のスキーマに存在しないか、またはユーザーがそのデータ域に対する権限を持っておらず、ジャーナリングを開始することがスキーマに指定されていない場合、表の作成時と同一のスキーマに QSQJRN というジャーナルが存在していると、そのジャーナルに対してジャーナリングが開始されます。

表の所有権**:** SQL 名が指定されている場合、
- 作成した表が入れられるスキーマと同じ名前のユーザー・プロファイルが存在する場合、表の所有者 はそのユーザー・プロファイルです。
- その他の場合は、表の所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

システム名を指定した場合は、表の所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

表の権限 **:** SQL 名を使用する場合は、表は、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成されます。システム名を使用する場合、表は、スキーマの作成権限 (CRTAUT) パラメーターによって決められる *PUBLIC に対する権限を使用して作成されます。

表の所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、その表に対する権限が与えられます。

REPLACE の規則**:** 表が REPLACE によって PRESERVE ROWS を使用して再作成される際には、表の新しい定義が古い定義と比較され、論理的に、二者間の各相違点について、対応する ALTER 操作が実行されます。 DELETE ROWS オプションを使用すると、表は論理的に除去され、再作成されます。表に従属するオブジェクトが有効である限り、変更は可能です。詳しくは、ALTER TABLE の 996 ページの表 78 を参照してください。列、制約、およびパーティションについては、それらの名前と属性に基づいて比較が実行されます。

列名またはシステム列名のいずれかを修正し、もう一方の名前は変えないままにして、列名とシステム列名を変更することができます。どちらの名前も既存の列に一致しない場合、新しい列が作成されます。列名に依存する別のオブジェクトが存在すると、名前変更をできない可能性があります。

新規の定義と既存の定義	同等の **ALTER TABLE** アクション
列	
列が両方に存在し、属性は同じである	変更なし
列が両方に存在し、属性は異なる	ALTER COLUMN
列が新規の表定義にのみ存在する	ADD COLUMN
列が既存の表定義にのみ存在する	DROP COLUMN RESTRICT
制約	
制約が両方に存在し、同じである	変更なし
制約が両方に存在し、異なる	DROP *constraint* RESTRICT および ADD *constraint*
制約が新規の表定義にのみ存在する	ADD *constraint*
制約が既存の表定義にのみ存在する	DROP *constraint* RESTRICT
materialized-query-definition	
materialized-query-definition が両方に存在し、同じである	変更なし
materialized-query-definition が両方に存在し、異なる	ALTER MATERIALIZED QUERY
materialized-query-definition が新規の表定義にのみ存在する	ADD MATERIALIZED QUERY
materialized-query-definition が既存の表定義にのみ存在する	DROP MATERIALIZED QUERY
partitioning-clause	
partitioning-clause が両方に存在し、同じである	変更なし
partitioning-clause が両方に存在し、異なる	ADD PARTITION、DROP PARTITION、および ALTER PARTITION
partitioning-clause が新規の表定義にのみ存在する	ADD *partitioning-clause*
partitioning-clause が既存の表定義にのみ存在する	DROP PARTITIONING
PERIOD	
PERIOD が両方に存在し、同じである	変更なし
PERIOD が新規の表定義にのみ存在する	ALTER ADD PERIOD
PERIOD が既存の表定義にのみ存在する	ALTER DROP PERIOD 表がシステム期間テンポラル表である場合、期間は除去できません
NOT LOGGED INITIALLY	
NOT LOGGED INITIALLY が両方に存在する	変更なし

新規の定義と既存の定義	同等の **ALTER TABLE** アクション
NOT LOGGED INITIALLY が新規の表定義にのみ存在する	NOT LOGGED INITIALLY
NOT LOGGED INITIALLY が既存の表定義にのみ存在する	最初にログに記録
VOLATILE	
VOLATILE 属性が両方に存在し、同じである	変更なし
VOLATILE 属性が新規の表定義にのみ存在する	VOLATILE
VOLATILE 属性が既存の表定義にのみ存在する	NOT VOLATILE
media-preference	
media-preference が両方に存在し、同じである	変更なし
media-preference が新規の表定義にのみ存在する	ALTER *media-preference*
media-preference が既存の表定義にのみ存在する	UNIT ANY
memory-preference	
memory-preference が両方に存在し、同じである	変更なし
memory-preference が新規の表定義にのみ存在する	ALTER *memory-preference*
memory-preference が既存の表定義にのみ存在する	KEEP IN MEMORY NO

CREATE TABLE ステートメントで指定できない属性はすべて保持されます。

- 権限を持つユーザーは維持されます。オブジェクト所有者は変更される可能性があります。
- 現在のジャーナル監査は保持されます。ただし、他のオブジェクトと異なり、表の REPLACE では、ZC (オブジェクト変更) ジャーナル監査項目が生成されます。
- 現行のデータ・ジャーナル処理は保持されます。
- コメントとラベルは保持されます。
- トリガーは、可能であれば保持されます。トリガーを保持できない場合は、エラーが戻されます。
- マスクと許可は、可能であれば保持されます。マスクや許可を保持できない場合は、エラーが戻されます。
- テンポラル表 VERSIONING と履歴表は保持されます。
- 表に従属するビュー、マテリアライズ照会表、および索引は、可能であれば保持されるか、再作成されます。従属するビュー、マテリアライズ照会表、または索引を保持できない場合は、エラーが戻されます。

ID 列の使用**:** 表に ID 列がある場合は、データベース・マネージャーは、表に行が挿入されたときに、その列の順次数値を自動的に生成することができます。したがって、ID 列は基本キーとして最適です。

ID 列と ROWID 列は、どちらにもデータベース・マネージャーが生成する値が含まれるという点で同じです。 ROWID 列は、直接行アクセスに使用すると便利です。 ROWID 列には、ROWID データ・タイプの値が入ります。これは、規則的に昇順または降順にはならない 40 バイトの VARCHAR 値を戻します。したがって、ROWID データ値は、社員番号や製品番号の生成など、多くのアプリケーション用途にはあまり適していません。直接行アクセスを必要としないデータの場合は、一般に ID 列を使用する方が効果的です。なぜなら、ID 列には既存の数値データ・タイプが含まれており、ROWID 値には不向きなさまざまな用途に利用できるからです。

表が特定時点の状態に回復されたときに (RMVJRNCHG を使用)、ID 列について生成される値のシーケンスに大きなギャップが生じることがあります。例えば、増分値が 1 で、時刻 T1 に最後に生成された値が 100 で、データベース・マネージャーがその後 1000 まで順次、値を生成する ID 列がある表を想定します。さらに、この表が時点 T1 にさかのぼって回復されたとします。この場合、リカバリーの完了後最初に挿入される行の ID 列の値は 1001 になり、ID 列の値に 100 から 1001 というギャップが生じます。

CYCLE を指定した場合は、列に対して固有制約または固有索引が定義されていない限り、その列が GENERATED ALWAYS であっても、その列について重複値が生成されることがあります。

マテリアライズ照会表の作成**:** マテリアライズ照会表が照会に使用される前にデータを持つようにするには、以下のようにします。

- DATA INITIALLY IMMEDIATE を使用してマテリアライズ照会表を作成する必要があります。または、
- 照会最適化を使用不可にした状態でマテリアライズ照会表を作成して、表のリフレッシュ後に表の照会最適化を使用可能にする必要があります。

CREATE TABLE ステートメントが実行されたときの分離レベルが、マテリアライズ照会表の分離レベルになります。 *ISOLATION* 文節 を使用して、分離レベルを明示的に指定できます。

暗黙的な非表示列に関する考慮事項**:** 暗黙的に非表示と定義されている列は、SELECT リスト内で * を指定する照会の結果表の一部にはなりません。ただし、暗黙的な非表示列を照会で明示的に参照することはできます。例えば、暗黙的な非表示列を照会の SELECT リストや述部で参照できます。さらに、暗黙的な非表示列は、COMMENT ステートメント、CREATE INDEX ステートメント、ALTER TABLE ステートメント、INSERT ステートメント、MERGE ステートメント、または UPDATE ステートメントで明示的に参照できます。暗黙的な非表示列は、参照制約で参照できます。列リストを含まない REFERENCES 文節は、親表の主キーを暗黙的に参照します。親表の主キーに、暗黙的に非表示と定義された列が組み込まれる場合があります。そのような参照制約を使用可能です。

マテリアライズ照会定義の全選択の SELECT リストで、暗黙的な非表示列を明示的に参照している場合、その列はマテリアライズ照会表の一部になります。

ビュー定義 (CREATE VIEW ステートメント) の全選択の SELECT リストで、暗黙的な非表示列を明示的に参照している場合、その列は当該ビューの一部となりますが、そのビューの列は「非表示」とは見なされません。

トランザクション開始 **ID** 列に関する考慮事項**:** トランザクション開始 ID 列で NULL 値が許可され、行開始列が存在し、その行開始列の値が他のトランザクションで生成された行開始列の値とは異なる固有の値になっている場合、トランザクション開始 ID 列には NULL 値が含まれます。列に NULL 値が含まれる可能性があるため、その列から値を取り出すときには、以下のいずれかの方式を使用することをお勧めします。

```
COALESCE (transaction_start_id_col, row_begin_col)
```

```
CASE WHEN transaction_start_id_col IS NOT NULL
          THEN transaction_start_id_col
     ELSE row_begin_col END
```

システム期間テンポラル表の定義**:** システム期間テンポラル表の定義には、以下が含まれます。

- SYSTEM_TIME という名前のシステム期間。これは、行開始列と行終了列を使用して定義されます。 AS ROW BEGIN、AS ROW END、および period-definition の説明を参照してください。
- トランザクション開始 ID 列。AS TRANSACTION START ID の説明を参照してください。
- 後続の ALTER TABLE ステートメントで指定される、ADD VERSIONING アクションを指定するシステム期間データ・バージョン管理定義。これには関連履歴表の名前が含まれます。 982 ページの『ADD VERSIONING USE HISTORY TABLE history-table-name』を参照してください。

パーティション化された表のパフォーマンス**:** パーティション化された表のパーティション数が大きくなると、 SQL データ変更および SQL データ・ステートメントのオーバーヘッドも大きくなります。このオーバーヘッドを最小化するのに必要な最小数のパーティションを持つパーティション化された表を作成する必要があります。パーティション化された表にアクセスする場合、並列処理の度合いを 1 より大きくすることを考慮するようお勧めします。

リモート選択ステートメントを使用した表作成**:** *as-result-table* の *select-statement* は、表を作成するサーバーとは別のサーバー上の表を参照できます。これを行うには、3 部構成のオブジェクト名を使用するか、または、表またはビューの 3 部構成の名前を参照するよう定義された別名を使用します。*select-statement* はマテリアライズ照会表に対するものにはできず、その結果はフィールド・プロシージャーが定義された列を含むことはできません。リモート・サーバーが Db2 for LUW または Db2 for z/OS である場合、*copy-options* は許可されません。リモート・サーバーが Db2 for LUW の場合、AS キーワードの前に列リストを明示的に指定する必要があります。

代替構文**:** 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- INLINE LENGTH は ALLOCATE の同義語です。

- 参照制約 内の FOREIGN KEY キーワードの後に制約名 (CONSTRAINT キーワードなし) を指定することもできます。
- DEFINITION ONLY は WITH NO DATA の同義語です。
- PARTITIONING KEY は DISTRIBUTE BY HASH の同義語です。
- PART は PARTITION の同義語です。
- PARTITION パーティション名 の代わりに、 PARTITION パーティション番号 を指定できます。 *partition-number* では、CREATE TABLE ステートメントで既に指定されているパーティションを指定することはできません。

  partition-number を指定しない場合は、データベース・マネージャーによって固有のパーティション番号が生成されます。
- VALUES は ENDING AT の同義語です。
- マテリアライズ照会表の作成時の CREATE と TABLE の間の SUMMARY です。

最大行サイズ

最大行サイズについては、列定義 の記述で参照される制約事項が 2 つあります。

- 最大行バッファー・サイズは 32766、あるいは VARCHAR、VARGRAPHIC、VARBINARY、LOB、または XML 列が指定されている場合は 32740 です。
- LOB または XML 列が指定されている場合の最大行データ・サイズは、3 758 096 383 です。このサイズは、行が挿入または更新されるときに決定されます。LOB または XML 列が指定されていない場合の最大行データ・サイズは 32766 で、VARCHAR 列、VARGRAPHIC 列、または VARBINARY 列が指定されている場合の最大行データ・サイズは 32740 です。

行バッファーおよび行データ、またはその両方の長さを決定するには、そのデータ・タイプのバイト数に基づいて、その行のそれぞれの列に対応する長さを加算します。

次の表は、NULL 値が使用できない列に関して、データ・タイプごとの列のバイト・カウントを示します。 NULL 値が許される列であればどのような列であっても、8 つの列ごとに 1 バイトが必要になります。フィールド・プロシージャーがある列の場合は、そのフィールド・プロシージャーの結果に基づいてカウントが変わる可能性もあります。

表 *83.* データ・タイプ別の列のバイト・カウント

データ・タイプ	行バッファー・バイト・カウント	行データ・バイト・カウント
SMALLINT	2	2
INTEGER	4	4
BIGINT	8	8
DECIMAL(*p*, *s*)	(*p*/2) + 1 の整数部分	(*p*/2) + 1 の整数部分
NUMERIC(*p*, *s*)	*p*	*p*
FLOAT (単精度)	4	4
FLOAT (倍精度)	8	8

表 83. データ・タイプ別の列のバイト・カウント (続き)

データ・タイプ	行バッファー・バイト・カウント	行データ・バイト・カウント
DECFLOAT(16)	8	8
DECFLOAT(34)	16	16
CHAR(*n*)	*n*	*n*
VARCHAR(*n*)	*n*+2	*n*+2
CLOB (*n*)	29+ 埋め込み	*n*+29
GRAPHIC(*n*)	*n**2	*n**2
VARGRAPHIC (n)	*n**2+2	*n**2+2
DBCLOB(*n*)	29+ 埋め込み	*n**2+29
BINARY(*n*)	*n*	*n*
VARBINARY(*n*)	*n*+2	*n*+2
BLOB(*n*)	29+ 埋め込み	*n*+29
DATE	10	4
TIME	8	3
TIMESTAMP(*p*)	*p* が 0 の場合は 19、それ以外の場合は 20+*p*	((*p*+1)/2) の整数部分 + 7
DATALINK(*n*)	*n*+24	*n*+24
ROWID	42	28
XML	29+ 埋め込み	2 147 483 647
特殊タイプ	ソース・タイプのバイト・カウント	ソース・タイプのバイト・カウント
注: 埋め込み は、境界合わせに必要な 1 から 15 の値です。		

データベースに記述される精度

- 浮動小数点フィールドは、ビット精度ではなく、10 進精度で Db2 for iデーターベース内に定義されます。ビット数による精度を 10 進精度に変換するには、*10 進精度 = CEILING(n/3.31)* (*n* は、変換するビット数) という算式を使用します。 10 進精度は、対話式 SQL を使用した場合に、表示される数値の桁数を決めるのに使用されます。
- SMALLINT (短整数) フィールドは、10 進精度 (4,0) で保管されます。
- INTEGER (整数) フィールドは、10 進精度 (9,0) で保管されます。
- BIGINT フィールドは、10 進精度 (19,0) で保管されます。

LONG VARCHAR、LONG VARGRAPHIC、および LONG VARBINARY

非標準構文である LONG VARCHAR、LONG VARGRAPHIC、および LONG VARBINARY がサポートされていますが、これは使用しないようにしてください。標準構文である VARCHAR(整数)、VARGRAPHIC(整数)、および VARBINARY(整数) の方が優先されます。したがって、VARCHAR(整数)、VARGRAPHIC(整数)、および VARBINARY(整数) を使用することをお勧めします。CREATE TABLE ス

テートメントの処理後、データベース・マネージャーは、LONG VARCHAR 列を VARCHAR、LONG VARGRAPHIC 列を VARGRAPHIC、そして LONG VARBINARY 列を VARBINARY と見なして処理を進めます。最大長は、移植不能な製品固有の方式で計算されます。

LONG VARCHAR [100]
行内で使用可能なスペースの量によって最大長が決まる可変長文字ストリングを示します。

LONG VARGRAPHIC [100]
行内で使用可能なスペースの量によって最大長が決まる可変長グラフィック・ストリングを示します。

LONG VARBINARY [100]
行内で使用可能なスペースの量によって最大長が決まる可変長バイナリー・ストリングを示します。

LONG 列の最大長は、次のようにして決まります。ただし、

- `i` は、表のすべての列 (ただし、LONG VARCHAR、LONG VARGRAPHIC、または LONG VARBINARY でもない) の行バッファー・バイト数の合計とする。
- `j` は、表の LONG VARCHAR、LONG VARGRAPHIC、および LONG VARBINARY の列の数とする。
- `k` は、該当の行で NULL が使用可能な列の数とする。

LONG VARCHAR および LONG VARBINARY 列それぞれの長さは、INTEGER((32716 - i-((k+7)/8))/j) になります。

それぞれの LONG VARGRAPHIC 列の長さは、LONG VARCHAR 列に関して計算した長さを 2 で割って決定します。この結果の整数部が長さになります。

システム名の生成規則

システムがシステム表、ビュー、索引、または列名を生成する場合は、特定の方法があります。以下の各項では、これらの方法およびシステム名生成規則について説明します。

列名の生成の規則

システム列名は、表またはビューの作成時点でそのシステム列名の指定がなく、しかも、列名が有効なシステム列名でない場合に生成されます。

列名に特殊文字が入っておらず、しかもその長さが 10 桁を超える場合には、10 桁のシステム列名が次のように生成されます。

- その名前の最初の 5 文字
- 5 桁の固有の番号

例えば、次のようになります。

100. 他のプロダクトとの互換性を備えるために、このオプションが用意されています。ただし、代わりに VARCHAR(整数)、VARGRAPHIC(整数)、または VARBINARY(整数) の指定をお勧めします。

```
LONGCOLUMNNAME のシステム列名は LONGC00001
```

列名が区切られている場合には、

- 区切り文字と区切り文字の範囲内にある文字から、最初の 5 文字がシステム列名の最初の 5 文字として使用されます。その範囲内の文字の数が、5 文字以下の場合には、その名前の右側は、下線 (_) で埋められます。小文字は、大文字に変換されます。システム列名に使用できる有効な文字は、A から Z、0 から 9、@、 #、¥、および _ だけです。これら以外の文字は、いずれも下線 (_) 文字に変えられます。この結果、最初の文字が下線になる場合、最初の文字は文字 Q に変えられます。
- 上記の 5 桁の文字に 5 桁の固有の番号が付加されます。

例えば、次のようになります。

```
"abc" のシステム列名は ABC__00001
"COL2.NAME" のシステム列名は COL2_00001
"C 3" のシステム列名は C_3__00001
"??" のシステム列名は Q____00001

"*column1" のシステム列名は QCOLU00001
```

表名の生成の規則

表、ビュー、別名、または索引が FOR SYSTEM NAME 節を使用せずに作成され、以下のいずれかの名前である場合、システム名が作成されます。

- 長さが 10 桁を超える名前
- システム名での使用が有効でない文字を含む名前

SQL 名、または対応するシステム名はいずれも、SQL ステートメントで使用して、作成済みの該当ファイルのアクセスに用いることができます。ただし、SQL 名を識別するのは、Db2 for i だけであり、他の環境では、システム名を使用する必要があります。

システム名を生成するには 2 つの方法があります。

- QGENOBJNAM と呼ばれるデータ域が表が作成されたのと同一のスキーマに存在する場合、ユーザーは生成される名前を変更することができます。

  データ域には、次のような制限があります。

  – ユーザーには、データ域を読み取る権限がなければならない。
  – データ域は、CHAR(10) の属性を持っていなければならない。
  – データ域値の先頭の 5 文字は、「?????」でなければならない。.
  – データ域値の次の 5 文字は、5 桁の数値でなければならない。

  上記の条件のいずれかが満たされない場合、またはデータ域の開始値にアクセスしている最中に何らかのエラーが発生した場合は、データ域が存在しなかった場合と同様に、デフォルト名生成規則が使用されます。

  データ域が上記のすべての制約事項を満たした場合、生成される名前は、下記のデフォルト名生成規則と同様になります。ただし、名前の先頭の 5 (または 4) 文字の後の固有番号には、(「00001」または「0001」の代わりに) 最初にデータ域で指定された 5 桁が入ります。

例えば、データ域の値が「?????00999」だった場合。

```
「??」のシステム名は「__00999」
"longtablename" のシステム名は "lon00999"
"LONGTableName" のシステム名は LONG00999
"A b   " のシステム名は "A_b00999"
```

- それ以外の場合は、デフォルト名生成規則が使用されます。

  名前に特殊文字が含まれておらず、その長さが 10 桁を超えている場合には、次のように 10 桁のシステム名が生成されます。

  – その名前の最初の 5 文字
  – 5 桁の固有の番号

  例えば、次のようになります。

  ```
  LONGTABLENAME のシステム名は LONGT00001
  ```

  その SQL 名に特殊文字が入っている場合、システム名の生成は次のようになります。

  – その名前の最初の 4 文字
  – 4 桁の固有の番号

  さらに、

  – 特殊文字は、すべて下線 (_) に置き換えられます。
  – 末尾ブランクは、すべてその名前から除去されます。
  – その名前を有効なシステム名にする上で区切り文字が必要になる場合には、その名前は二重の引用符 (") によって区切られます。

  例えば、次のようになります。

  ```
  "??" のシステム名は "__0001"
  "longtablename" のシステム名は "long0001"
  "LONGTableName" のシステム名は LONG0001
  "A b   " のシステム名は "A_b0001"
  ```

  SQL は相互参照ファイルを検索して、システム名が固有であることを保証します。ある名前が既に相互参照ファイルに存在している場合、その名前が固有の名前になるまで、その番号を増やします。

  上記の規則を使用しても固有名を決められない場合は、名前の番号の桁数を 1 桁追加して、固有名になるまで、または範囲の限界に達するまで、番号を増分します。例えば、"longtablename" を作成しているときに、"long0001" から "long9999" が既に存在する場合、名前は "lon00001" になります。

例

例 1: データベース管理者権限を持っているものとして、'ROSSITER.INVENTORY' という名前の表を作成します。この表は、次のような列から構成されます。

部品番号
　　短整数、NULL 不可

説明　 0 から 24 の文字、NULL 可

在庫数量、
整数、NULL 可

```
CREATE TABLE ROSSITER.INVENTORY
  (PARTNO         SMALLINT     NOT NULL,
   DESCR          VARCHAR(24),
   QONHAND        INT)
```

例 *2:* DEPARTMENT という名前の表を作成します。この表は、次のような列から構成されます。

部門番号
3 文字長で、NULL は使用できない。

部門名
0 から 36 文字長で、NULL は使用できない。

管理者番号
6 文字長

管理部門
3 文字長で、NULL は使用できない。

ロケーション名
16 文字長で NULL を使用できる。

基本キーは、列 DEPTNO です。

```
CREATE TABLE DEPARTMENT
  (DEPTNO    CHAR(3)     NOT NULL,
   DEPTNAME  VARCHAR(36) NOT NULL,
   MGRNO     CHAR(6),
   ADMRDEPT  CHAR(3)     NOT NULL,
   LOCATION  CHAR(16),
   PRIMARY KEY(DEPTNO))
```

例 *3:* ビュー PRJ_LEADER の列と同じ列定義に従って、REORG_PROJECTS という名前の表を作成します。

```
CREATE TABLE REORG_PROJECTS
  LIKE PRJ_LEADER
```

例 *4:* EMP_NO という ID 列を持つ EMPLOYEE2 表を作成します。 Db2 for i が常に列の値を生成するように、ID 列を定義します。割り当てる最初の値および後に生成される連続番号の間の増分差に対して、デフォルト値である 1 を使用します。

```
CREATE TABLE EMPLOYEE2
  ( EMPNO INTEGER GENERATED ALWAYS AS IDENTITY,
    ID SMALLINT,
    NAME CHAR(30),
    SALARY DECIMAL(5,2),
    DEPTNO SMALLINT)
```

例 *5:* TRANS という名前の非常に大きいトランザクション表に、会社に処理されるトランザクション処理ごとに 1 つの行が含まれると想定します。表は、多くの列で定義されます。日付およびトランザクションの量に関する毎日のサマリー・データが含まれる、マテリアライズ照会表を作成します。

```
CREATE TABLE STRANS
  AS (SELECT  YEAR AS SYEAR, MONTH AS SMONTH, DAY AS SDAY, SUM(AMOUNT) AS SSUM
      FROM TRANS
```

```
        GROUP BY YEAR, MONTH, DAY )
    DATA INITIALLY DEFERRED
    REFRESH DEFERRED
    MAINTAINED BY USER
```

例 *6:* SYSTEM_TIME 期間を使用する表 *policy_info* を作成し、履歴表 *hist_policy_info* を作成します。次に、*policy_info* 表を *hist_policy_info table* 表に関連付ける ALTER TABLE ステートメントを発行します。

```
CREATE TABLE policy_info
    (policy_id CHAR(10) NOT NULL,
     coverage INT NOT NULL,
     sys_start TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN,
     sys_end TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END,
     create_id TIMESTAMP(12) GENERATED ALWAYS AS TRANSACTION START ID,
     PERIOD SYSTEM_TIME(sys_start,sys_end));

CREATE TABLE hist_policy_info
    (policy_id CHAR(10) NOT NULL,
     coverage INT NOT NULL,
     sys_start TIMESTAMP(12) NOT NULL,
     sys_end TIMESTAMP(12) NOT NULL,
     create_id TIMESTAMP(12));

ALTER TABLE policy_info
 ADD VERSIONING USE HISTORY TABLE hist_policy_info;
```

LIKE について生成属性はコピーされないため、履歴表は以下の方法でも作成できます。

```
CREATE TABLE hist_policy_info
      LIKE policy_info;
```

CREATE TRIGGER

CREATE TRIGGER ステートメントは、現行サーバーでトリガーを定義します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- スキーマ内に作成する特権。詳しくは、スキーマ内で作成する必要のある権限を参照してください。
- データベース管理者権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 以下のそれぞれが必要です。
  - トリガーが定義される表またはビューの ALTER 特権、
  - トリガーが定義される表またはビューの SELECT 特権、
  - トリガー・アクション での検索条件 で参照された表またはビューの SELECT 特権、
  - BEFORE UPDATE トリガーに NEW 相関変数を変更する SET ステートメントが含まれている場合、トリガーが定義される表の UPDATE 特権、
  - 各トリガー *SQL* ステートメント の実行に必要な特権、
  - トリガーを定義する表またはビューが入っているスキーマに対する USAGE 特権
- データベース管理者権限

固有に更新できないビューに INSTEAD OF トリガーが追加される場合、*OBJMGT システム権限もビュー上に必要になります。

さらに、このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 次のシステム権限
  - 物理ファイル・トリガー追加 (ADDPFTRG) コマンドに対する *USE
  - プログラム作成 (CRTPGM) コマンドに対する *USE
- データベース管理者権限

SECURED 属性が指定されるか、または、トリガーがセキュアであり OR REPLACE が指定される場合は、次のとおりです。

- このステートメントの許可 ID には、セキュリティー管理者権限 がなければなりません。 21 ページの『管理権限』を参照してください。

SQL 名が指定され、該当のトリガーが作成されるライブラリーの名前と同じ名前のユーザー・プロファイルが存在し、しかもその名前がステートメントの権限 ID と異なっている場合、ステートメントの権限 ID が保持する特権には、少なくとも次の 1 つが含まれていなければなりません。

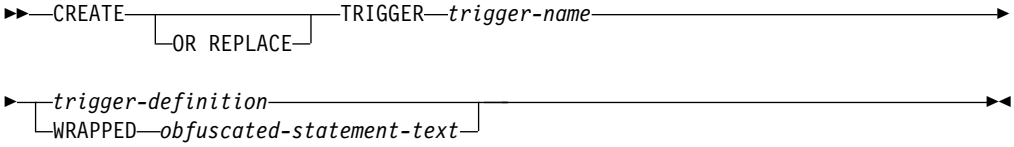
- *ALLOBJ および *SECADM 特殊権限
- データベース管理者権限

既存のトリガーに置き換えるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

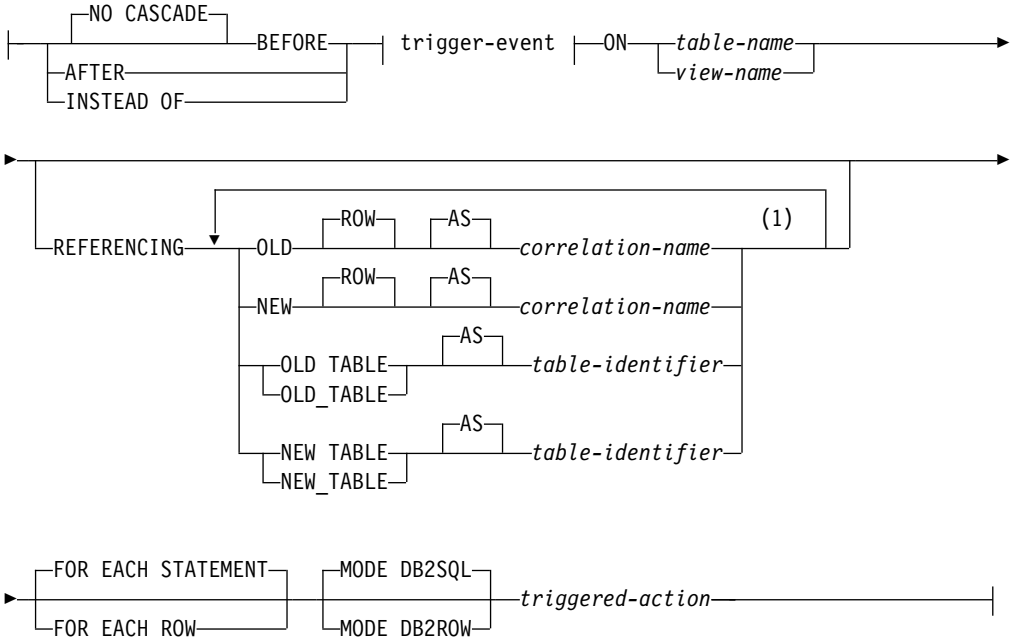
- 次のシステム権限
  - トリガー・プログラム・オブジェクトに対する *OBJMGT システム権限
  - このトリガーを削除するために必要な全権限
- データベース管理者権限

SQL 特権に対応するシステム権限については、『表またはビューへの権限を検査する際の対応するシステム権限』を参照してください。

構文



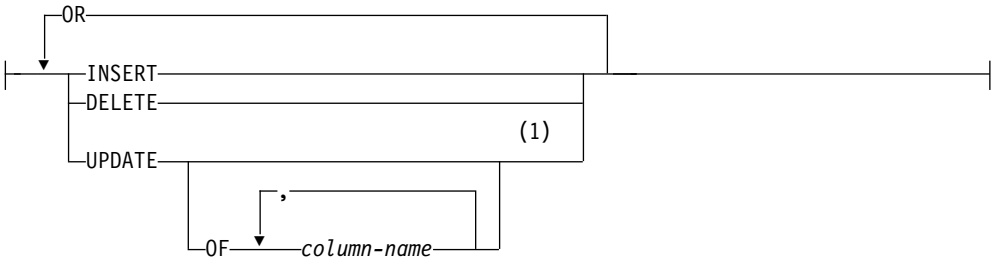
trigger-definition:



注:

1 同じ文節を複数回指定することはできません。

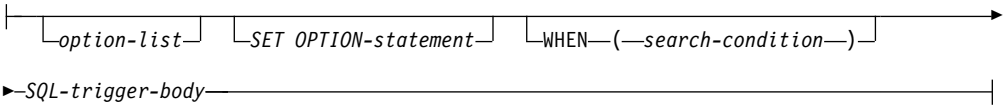
trigger-event:



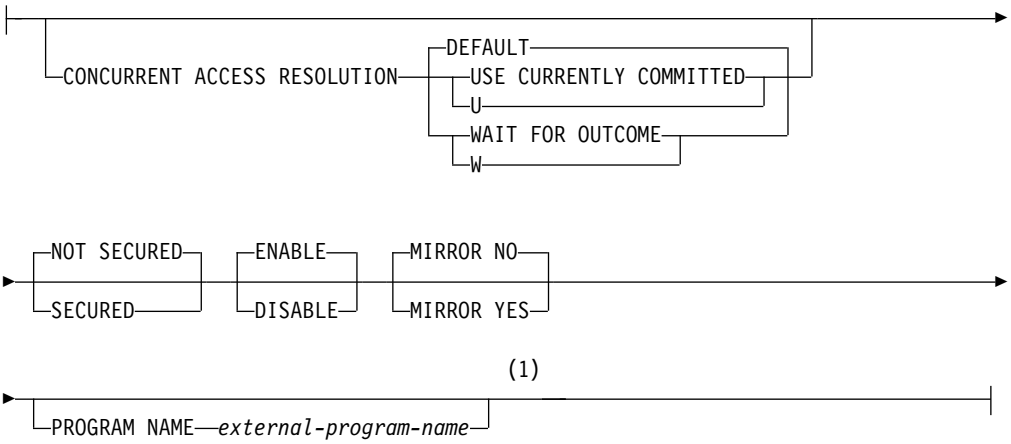
注:

1 各 *trigger-event* オプションは 1 回だけ指定できます。

triggered-action:



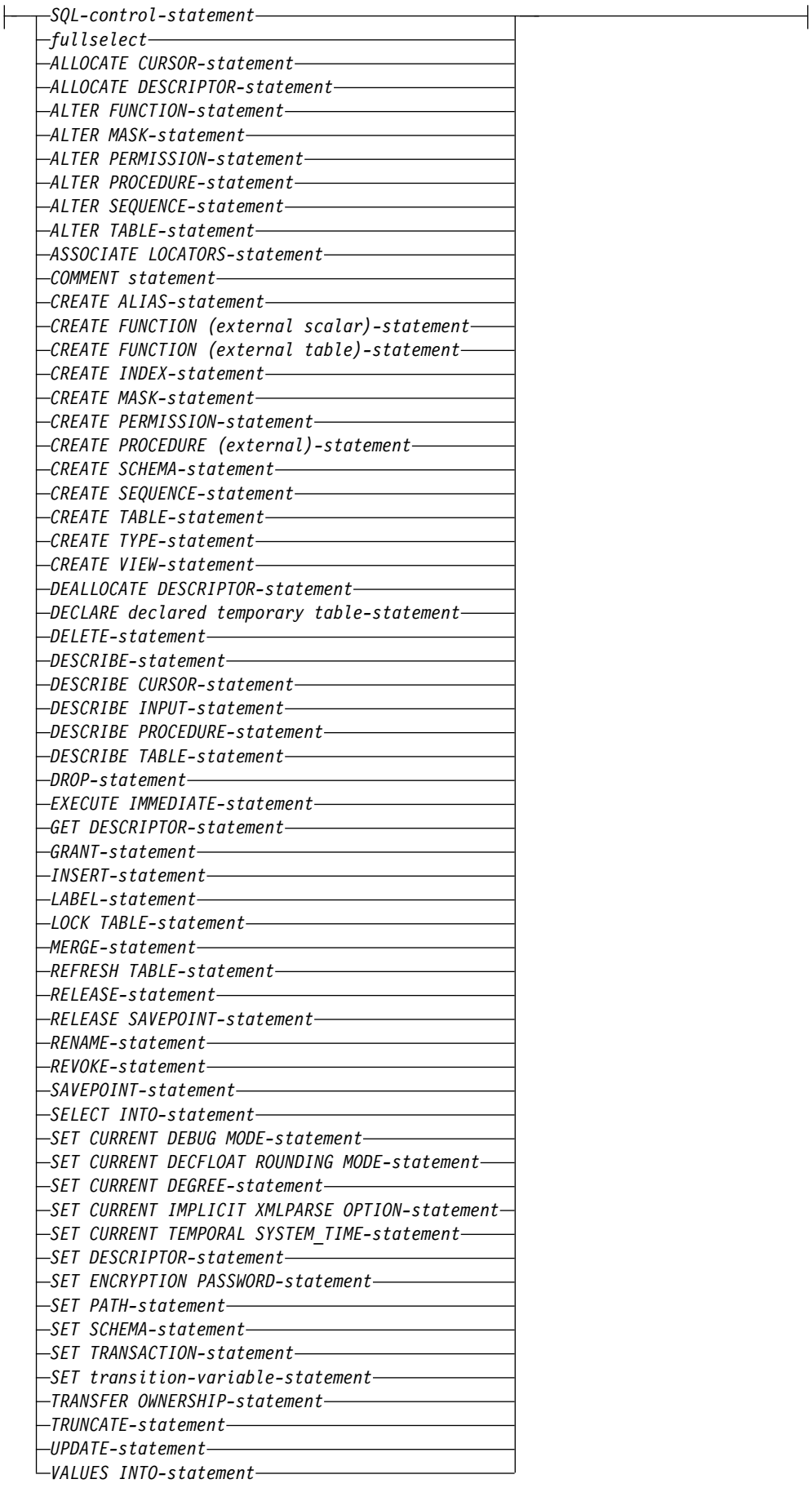
option-list:



注:

1 オプションは任意の順序で指定できます。同じ文節を複数回指定することはできません。

SQL-trigger-body:


SQL-control-statement
fullselect
ALLOCATE CURSOR-statement
ALLOCATE DESCRIPTOR-statement
ALTER FUNCTION-statement
ALTER MASK-statement
ALTER PERMISSION-statement
ALTER PROCEDURE-statement
ALTER SEQUENCE-statement
ALTER TABLE-statement
ASSOCIATE LOCATORS-statement
COMMENT statement
CREATE ALIAS-statement
CREATE FUNCTION (external scalar)-statement
CREATE FUNCTION (external table)-statement
CREATE INDEX-statement
CREATE MASK-statement
CREATE PERMISSION-statement
CREATE PROCEDURE (external)-statement
CREATE SCHEMA-statement
CREATE SEQUENCE-statement
CREATE TABLE-statement
CREATE TYPE-statement
CREATE VIEW-statement
DEALLOCATE DESCRIPTOR-statement
DECLARE declared temporary table-statement
DELETE-statement
DESCRIBE-statement
DESCRIBE CURSOR-statement
DESCRIBE INPUT-statement
DESCRIBE PROCEDURE-statement
DESCRIBE TABLE-statement
DROP-statement
EXECUTE IMMEDIATE-statement
GET DESCRIPTOR-statement
GRANT-statement
INSERT-statement
LABEL-statement
LOCK TABLE-statement
MERGE-statement
REFRESH TABLE-statement
RELEASE-statement
RELEASE SAVEPOINT-statement
RENAME-statement
REVOKE-statement
SAVEPOINT-statement
SELECT INTO-statement
SET CURRENT DEBUG MODE-statement
SET CURRENT DECFLOAT ROUNDING MODE-statement
SET CURRENT DEGREE-statement
SET CURRENT IMPLICIT XMLPARSE OPTION-statement
SET CURRENT TEMPORAL SYSTEM_TIME-statement
SET DESCRIPTOR-statement
SET ENCRYPTION PASSWORD-statement
SET PATH-statement
SET SCHEMA-statement
SET TRANSACTION-statement
SET transition-variable-statement
TRANSFER OWNERSHIP-statement
TRUNCATE-statement
UPDATE-statement
VALUES INTO-statement

説明

OR REPLACE
現行サーバーにこのトリガーの定義が存在する場合に、その定義を置き換える、という動作を指定します。実際には、カタログで既存の定義を削除してから新しい定義に置き換える、という動作になりますが、例外として、このトリガーに対して与えられていた特権は影響を受けません。現行サーバーにこのトリガーの定義が存在しなければ、このオプションは無視されます。

trigger-name

トリガーの名前を指定します。暗黙的または明示的修飾子も含め、この名前は、現行サーバーに既に存在しているトリガーと同じ名前にすることはできません。QTEMP は、トリガー名 スキーマ修飾子として使用することはできません。

SQL 名が指定されている場合、トリガーは、暗黙的または明示的修飾子で指定しているスキーマ内に作成されます。

システム名が指定されている場合、トリガーは、修飾子で指定しているスキーマ内に作成されます。修飾されていない場合、トリガーは、対象表と同じスキーマ内に作成されます。

トリガー名が有効なシステム名でない場合、または同じ名前のプログラムが既に存在する場合、データベース・マネージャーはシステム名を生成します。名前の生成に関する規則については、 1298 ページの『表名の生成の規則』を参照してください。

NO CASCADE
NO CASCADE は、他のプロダクトとの互換性を保持するために許可されており、Db2 for i では無視されます。

BEFORE
トリガーが前 トリガーであることを指定します。データベース・マネージャーは、対象表に対する挿入、削除、または更新操作による変更を適用する前に、トリガー・アクション を実行します。前トリガーのトリガー・アクション には更新を含めることができないので、このトリガー・アクション は別のトリガーを起動しないことも指定します。

view-name を指定する場合は、BEFORE を指定しないでください。FOR EACH STATEMENT は、BEFORE トリガーには指定してはいけません。

AFTER
トリガーが後 トリガーであることを指定します。データベース・マネージャーは、対象表に対する挿入、削除、または更新操作による変更を適用後に、トリガー・アクション を実行します。*view-name* を指定する場合は、AFTER を指定しないでください。

INSTEAD OF
トリガーが代用トリガーであることを指定します。対象ビューに対するアクションは、関連付けられたトリガー・アクションに置き換えられます。対象ビューに対する操作の種類ごとに、INSTEAD OF トリガーを 1 つのみ使用できます。データベース・マネージャーは、対象ビューに対する挿入、削除、更新の操作の代わりに、トリガー・アクション を実行します。

table-name を指定する場合は、INSTEAD OF を指定しないでください。
INSTEAD OF トリガーの場合は、WHEN 文節を指定してはいけません。
FOR EACH STATEMENT は、INSTEAD OF トリガーには指定してはいけません。

trigger-event
これを指定すると、サブジェクト表またはビューにいずれかのイベントが適用される場合には必ず、このトリガーに関連するトリガー・アクションが実行されます。任意の組み合わせのイベントを指定できますが、各イベント (INSERT、DELETE、UPDATE) は 1 回しか指定できません。

INSERT
サブジェクト表で挿入操作があった場合には必ずこのトリガーに関連付けられたトリガー・アクション を実行することを指定します。

BEFORE INSERT トリガーは、履歴表に追加できません。

DELETE
サブジェクト表で削除操作があった場合には必ずこのトリガーに関連付けられたトリガー・アクション を実行することを指定します。

DELETE トリガーは、ON DELETE CASCADE 参照制約を含む表には追加できません。

UPDATE
サブジェクト表で更新操作があった場合には必ずこのトリガーに関連付けられたトリガー・アクション を実行することを指定します。

UPDATE トリガー・イベントは、ON DELETE SET NULL または ON DELETE SET DEFAULT 参照制約を含む表には追加できません。

BEFORE UPDATE トリガーは、履歴表に追加できません。

明示的な列名 リストが指定されていない場合、後で ALTER TABLE ステートメントによって追加される列も含めて、対象表の列に対する更新操作はすべてトリガー・アクション を起動します。

OF *column-name*,...
指定する各列名 は、対象表の列でなければならず、リストには一度しか表示できません。リストされた列に対する更新操作はすべてトリガー・アクション を起動します。この文節は INSTEAD OF トリガーには指定できません。

ON 表名
BEFORE または AFTER トリガー定義の対象表を識別します。この名前は、現行サーバー上に存在する基本表を示すものでなければならず、カタログ表、QTEMP 内の表、または宣言済み一時表を示すものであってはなりません。

ON ビュー名
INSTEAD OF トリガー定義の対象ビューを識別します。名前は、現行サーバー上に存在するビューを示すものでなければならず、カタログ・ビューまたは QTEMP 内のビューを示すものであってはなりません。名前は、WITH CHECK OPTION を使用して定義されたビュー、または WITH CHECK OPTION ビューとして定義されているビューを、直接または間接的に示すものであってはなりません。

REFERENCING

遷移表の相関名と遷移表の表名を指定します。相関名 は、トリガー SQL 操作の影響を受ける行セット内の特定行を識別します。表 *ID* は、影響を受ける行セット全体を識別します。

次のように相関名 を指定して列を修飾することにより、トリガー SQL 操作の影響を受ける各行が、トリガー・アクション に対して使用可能になります。

OLD ROW AS 相関名

トリガー SQL 操作の前の行の値を識別する相関名を指定します。トリガー・イベントが挿入の場合、OLD ROW の各列の値は NULL 値です。

NEW ROW AS 相関名

トリガー SQL 操作と既に実行された前トリガー内の SET ステートメントによって変更された行の値を識別する相関名を指定します。トリガー・イベントが削除の場合、NEW ROW の各列の値は NULL 値です。

次のように一時表名を指定することにより、トリガー SQL 操作によって影響を受ける行セット全体が、トリガー・アクション に対して使用可能になります。

OLD TABLE AS 表 *ID*

トリガー SQL 操作の前の、影響を受ける行セット全体の値を識別する一時表の名前を指定します。現行トリガーが、あるトリガーの *SQL* トリガー本体 内のステートメントによって起動された場合、OLD TABLE には、そのトリガーによって影響を受けた行も含まれます。トリガー・イベントが挿入の場合、この一時表は空です。

NEW TABLE AS 表 *ID*

トリガー SQL 操作とすでに実行された前トリガー内の SET ステートメントによって変更された、影響を受ける行セット全体の状態を識別する一時表の名前を指定します。トリガー・イベントが削除の場合、この一時表は空です。

1 つのトリガーには、相関名 として、 1 つの OLD と 1 つの NEW だけしか指定できません。 1 つのトリガーには、表 *ID* として 1 つの OLD_TABLE と 1 つの NEW_TABLE しか指定できません。相関名 および表 *ID* のすべては相互に固有でなければなりません。

OLD 相関名 および OLD_TABLE 表 *ID* は、トリガー・イベントが削除操作または更新操作の場合のみ設定されます。削除操作の場合、OLD *correlation-name* は削除される行の列の値を取り込み、OLD_TABLE *table-identifier* は削除される行のセットにある値を取り込みます。UPDATE 操作の場合、OLD 相関名 は、その UPDATE 操作の前の時点での行の列の値を取り込み、 OLD_TABLE 表 *ID* は、更新された行のセットの値を取り込みます。

NEW ROW 相関名 および NEW TABLE 表 *ID* は、トリガー・イベントが INSERT 操作または UPDATE 操作の場合のみ設定されます。どちらの操作の場合も、NEW ROW 相関名 は、挿入または更新された行内の列の値を取り込み、NEW TABLE 表 *ID* は、挿入または更新された行セット内の値を取り込みます。前トリガーの場合、更新された行の値には、前トリガーのトリガー・アクション 内の SET ステートメントからの変更が含まれます。

OLD ROW および NEW ROW 相関名 変数は、AFTER トリガーまたは INSTEAD OF トリガー内では変更できません。

下表は、相関変数と遷移表の可能な組み合わせを要約しています。

細分性：**FOR EACH ROW**

MODE	起動時	トリガー操作	許される相関変数	許される遷移表
DB2ROW	BEFORE	DELETE	OLD	NONE
		INSERT	NEW	
		UPDATE	OLD, NEW	
	AFTER または INSTEAD OF	DELETE	OLD	
		INSERT	NEW	
		UPDATE	OLD, NEW	
DB2SQL	BEFORE	DELETE	OLD	
		INSERT	NEW	
		UPDATE	OLD, NEW	
	AFTER または INSTEAD OF	DELETE	OLD	OLD TABLE
		INSERT	NEW	NEW TABLE
		UPDATE	OLD, NEW	OLD TABLE, NEW TABLE

細分性：**FOR EACH STATEMENT**

MODE	起動時	トリガー操作	許される相関変数	許される遷移表
DB2SQL	AFTER または INSTEAD OF	DELETE	NONE	OLD TABLE
		INSERT		NEW TABLE
		UPDATE		OLD TABLE, NEW TABLE

文字データ・タイプの遷移変数は、対象表の列の CCSID を継承します。トリガー・アクション の実行時に、遷移変数は変数のように扱われます。したがって、文字変換が行われる可能性があります。

一時遷移表は、読み取り専用です。これは変更できません。

それぞれの *correlation-name*、およびそれぞれの *table-identifier* の有効範囲は、トリガー定義全体です。

FOR EACH ROW
データベース・マネージャーは、トリガー操作が変更する対象表の各行ごとにトリガー・アクション を実行することを指定します。そのトリガー操作がどの行も変更しない場合には、トリガー・アクション は実行されません。

FOR EACH STATEMENT
データベース・マネージャーは、トリガー操作につき一度だけ、トリガー・アクション を実行することを指定します。そのトリガー操作がどの行も変更または削除しない場合でも、トリガー・アクションは一度実行します。

FOR EACH STATEMENT は、BEFORE トリガーに対しては指定できません。

FOR EACH STATEMENT は、MODE DB2ROW トリガーに対しては指定できません。

MODE DB2SQL
MODE DB2SQL は AFTER トリガーに対して有効です。MODE DB2SQL AFTER トリガーは、すべての行操作が完了した後で起動されます。

REFERENCING 節が指定されておらず、トリガー表が *SQL-trigger-body* 内で参照されていない場合のみ、MODE DB2SQL は BEFORE トリガーに対して有効です。MODE DB2SQL BEFORE トリガーは、各行の操作時に起動されます。

MODE DB2ROW
MODE DB2ROW トリガーは、各行の操作時に起動されます。

MODE DB2ROW は、BEFORE と AFTER 起動時の両方に有効です。

CONCURRENT ACCESS RESOLUTION
データベース・マネージャーが更新プロセスの過程にあるデータを待つかどうかを指定します。DEFAULT がデフォルトです。

DEFAULT
このトリガーに関する並行アクセスの解決方法を明示的に設定しないことを指定します。このトリガー・プログラムの呼び出し時に有効だった値が使用されます。

WAIT FOR OUTCOME
データベース・マネージャーが更新プロセスの過程にあるデータのコミットまたはロールバックを待つ、という動作を指定します。

USE CURRENTLY COMMITTED
データベース・マネージャーが更新プロセスの過程にあるデータを検出したときに現時点でのコミット済みバージョンのデータを使用する、という動作を指定します。

読み取りトランザクションと削除/更新トランザクションの間でロックの競合が発生した場合に、この文節の適用対象になるのは、分離レベル CS のスキャンです (CS KEEP LOCKS のスキャンは対象になりません)。

SECURED または NOT SECURED
トリガーが行アクセス制御と列アクセス制御においてセキュアであると見なされるかどうかを指定します。NOT SECURED がデフォルトです。

SECURED
トリガーが行アクセス制御と列アクセス制御においてセキュアであると見なされることを指定します。

トリガーのサブジェクト表が行アクセス制御または列アクセス制御を使用している場合、そのトリガーには SECURED を指定する必要があります。また、トリガーがビューに対して作成され、そのビュー定義内の 1 つ以上の基礎表が行アクセス制御または列アクセス制御を使用している場合も、そのトリガーに SECURED を指定する必要があります。

NOT SECURED
トリガーが行アクセス制御と列アクセス制御においてセキュアでないと見なされることを指定します。

トリガーのサブジェクト表が行アクセス制御または列アクセス制御を使用している場合、そのトリガーに対しては NOT SECURED を明示的にも暗黙的にも指定してはなりません。また、NOT SECURED は、ビュー定義にある 1 つ以上の基礎表が行アクセス制御または列アクセス制御を使用しているビューに対して作成されたトリガーにも指定してはなりません。

ENABLE または **DISABLE**
トリガーの状態を指定します。ENABLE がデフォルトです。

ENABLE
トリガーは該当するデータ変更操作中に起動されます。

DISABLE
トリガーは該当するデータ変更操作中に起動されません。

MIRROR NO または **MIRROR YES**
ミラー保護環境でトリガーを起動する場所を指定します。ミラーリングがアクティブではない場合、このオプションは無視されます。 MIRROR NO がデフォルトです。

MIRROR NO
ミラー保護環境で、トリガーはソース・ノードでのみ起動されます。

MIRROR YES
ミラー保護環境で、トリガーは、ミラー保護されたペアの両ノードで起動されます。

MIRROR YES は、INSTEAD OF トリガーや、MODE DB2SQL として定義されたトリガーには使用できません。

WRAPPED *obfuscated-statement-text*
エンコードされたトリガー定義を指定します。WRAP スカラー関数を使用して CREATE TRIGGER ステートメントをエンコードできます。

PROGRAM NAME *external-program-name*
トリガー用に作成するプログラムの非修飾名を指定します。
external-program-name は、プログラム名の形式でなければなりません。サービス・プログラム名であってはなりません。

triggered-action
トリガーの起動時に実行するアクションを指定します。トリガー・アクションは、1 つ以上の SQL ステートメントと、ステートメントを実行するかどうかを制御するオプション条件から構成されます。

SET OPTION-statement
トリガーを作成するときに使用するオプションを指定します。例えば、デバッグ可能トリガーを作成する場合は、次のステートメントを含めることができます。

```
SET OPTION DBGVIEW = *SOURCE
```

各オプションのデフォルト値は、作成時に有効だったオプションによって異なります。詳しくは、 1696 ページの『SET OPTION』を参照してください。

オプション CNULIGN、CNULRQD、COMPILEOPT、NAMING、SQLCA は、CREATE TRIGGER ステートメントでは使用できません。COMMIT オプションを指定することはできますが、指定しても無視されます。

オプション DATFMT、DATSEP、TIMFMT、および TIMSEP は、OLD ROW または NEW ROW が指定されている場合は使用できません。

WHEN (*search-condition***)**
真、偽、または不明として評価される条件を指定します。起動された SQL ステートメントは、検索条件 が真と評価された場合にのみ実行されます。WHEN 文節を省略した場合は、関連の SQL ステートメントは常に実行されます。

INSTEAD OF トリガーでは WHEN 文節を指定してはなりません。

SQL トリガー本体
単一の *SQL-procedure-statement* (複合ステートメントを含む) を指定します。SQL トリガーの定義についての詳細は、 1771 ページの『第 8 章 SQL 制御ステートメント』を参照してください。

CONNECT、 SET CONNECTION、 RELEASE、 DISCONNECT、 COMMIT、 ROLLBACK、 SET TRANSACTION、および SET RESULT SETS ステートメントを実行するプロシージャーへの呼び出しは、トリガーのトリガー・アクション 内では使用できません。

UNDO ハンドラーは、トリガーでは使用できません。

トリガー・アクション 内で参照される表、ビュー、別名、特殊タイプ、グローバル変数、ユーザー定義関数、およびプロシージャーはすべて、そのトリガーが作成された時点での現行サーバーに存在していることが必要です。別名が参照している表やビューも、トリガーの作成時に存在していなければなりません。これには、ライブラリー QTEMP 内のオブジェクトも含まれます。QTEMP 内のオブジェクトはトリガー・アクション で参照できますが、QTEMP 内のこれらのオブジェクトを除去しても、トリガーは除去されません。

すべての遷移変数の名前は、サブジェクト表の列名です。サブジェクト表のシステム列名を遷移変数の名前として使用することはできません。

XML タイプの列に対する BEFORE トリガーのトリガー・アクションでは、SET ステートメントで XMLVALIDATE 関数を呼び出したり、XML タイプの値を変更しないでそのまま残したり、SET ステートメントで値を NULL に設定したりする操作を実行できます。

トリガー・アクション 内の動的ステートメントは、 3 パートの名前を使用してリモート・サーバーにアクセスできます。トリガー・プログラムのライブラリーと同じ名前のライブラリーが、リモート・サーバーに存在しなければなりません。トリガー・アクション 内の静的ステートメントは、3 パートの名前を使用できません。

トリガー・アクション 内の静的および動的ステートメントは、プロシージャーまたはユーザー定義関数が異なる活動化グループ内で実行される場合、現行サーバー以外のサーバーにアクセスできるプロシージャーまたはユーザー定義関数を呼び出すことができます。

注

トリガーの所有権**:** SQL 名が指定されている場合、

- 作成したトリガーが入れられるスキーマと同じ名前のユーザー・プロファイルが存在する場合、トリガーの所有者 はそのユーザー・プロファイルです。
- その他の場合は、トリガーの所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

システム名を指定した場合は、トリガーの所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

トリガー権限**:** トリガー・プログラム・オブジェクトの権限は、次のとおりです。

- SQL 命名が有効の場合、トリガー・プログラムは *EXCLUDE 共通権限によって作成され、その名前のユーザー・プロファイルが存在する場合は、トリガー名のスキーマ修飾子から権限を借用します。スキーマ修飾子のユーザー・プロファイルが存在しない場合には、トリガー・プログラムの所有者は、スキーマ修飾子のユーザー・プロファイルになります。スキーマ修飾子と同じ名前のユーザー・プロファイルが存在し、その名前がステートメントの権限 ID と異なっている場合、スキーマ修飾子ライブラリー内にトリガー・プログラム・オブジェクトを作成するには、特殊権限の *ALLOBJ と *SECADM が必要です。スキーマ修飾子のユーザー・プロファイルが存在しない場合は、トリガー・プログラムの所有者は、SQL CREATE TRIGGER ステートメントを実行するスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルになります。グループ・ユーザー・プロファイルがトリガー・プログラム・オブジェクトの所有者になるのは、ステートメントを実行するユーザーのプロファイルで OWNER(*GRPPRF) が指定された場合に限られます。トリガー・プログラムの所有者がグループ・プロファイルのメンバーであり、ユーザーのプロファイルで OWNER(*GRPPRF) が指定された場合、プログラムは、グループ・プロファイルの借用権限を使用して実行されます。
- システム命名が有効の場合、トリガー・プログラムは *EXCLUDE 共通権限によって作成され、SQL CREATE TRIGGER ステートメントを実行するスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルから権限を借用します。

実行権限**:** トリガー SQL 操作を実行するユーザーに、静的なトリガー *SQL* ステートメント を実行する権限は必要ありません。トリガー *SQL* ステートメント は、トリガーの所有者 の権限で実行できます。

REPLACE の規則**:** REPLACE によってトリガーを再作成する場合は、以下のようになります。

- 既存のコメントまたはラベルは破棄されます。
- 権限を持つユーザーは維持されます。オブジェクト所有者は変更される可能性があります。
- 現在のジャーナル監査は保持されます。
- トリガーの起動順序は維持されません。

トリガーのアクティブ化**:** 挿入操作、削除操作、または更新操作のみが、トリガーをアクティブにできます。参照制約の結果として生じる削除操作は、トリガーを起動しません。したがって、次のようになります。

- DELETE トリガー・イベントを含むトリガーは、ON DELETE CASCADE 参照制約を含む表には追加できません。
- UPDATE トリガー・イベントを含むトリガーは、ON DELETE SET NULL または ON DELETE SET DEFAULT 参照制約を含む表には追加できません。

トリガーの起動によって、トリガー・カスケード が生じることがあります。これは、あるトリガーの起動によって、SQL ステートメントが実行され、その実行によって別のトリガーが起動されたり、同じトリガーが再度起動されたりする結果起きるものです。トリガー・アクションでは、最初の変更の結果として更新が行われ、その結果としてさらにトリガーが起動されるといったことも起こります。トリガー・カスケードを使用すると、有効なトリガー・チェーンを起動することが可能で、単一の削除、挿入、または更新ステートメントによって、データベースに対する多数の変更を行うことができます。カスケードのレベル数は、200 またはジョブやプロセスに許容される最大記憶量のいずれか先に達する値に制限されます。

トリガーを追加して制約を実行する**:** 既に行が含まれている表に対してトリガーを追加しても、トリガー・アクションは実行されません。したがって、表内のデータに対して制約を適用するためのトリガーを設計した場合、既存の行内のデータは、それらの制約を満たしていない可能性があります。

暗黙的な非表示列に関する考慮事項**:** トリガーの本体内で、暗黙的な非表示列に対応するトリガー遷移変数を参照することができます。暗黙的な非表示列がある表に対応するトリガー遷移表には、この列が遷移表の一部として格納されます。

同様に、トリガー遷移変数が、暗黙的に非表示と定義される列に対して存在します。暗黙的な非表示列に対応するトリガー遷移変数は、トリガーの本体内で参照できます。

読み取り専用ビュー**:** ビューに対して INSTEAD OF トリガーを追加すると、そのビューの読み取り専用特性に影響します。読み取り専用ビューが INSTEAD OF トリガーと従属関係にあると、その INSTEAD OF トリガーに対して定義される操作のタイプによって、このビューが削除可能、挿入可能、または更新可能であるかが定義されます。

遷移変数と **INSTEAD OF** トリガー**:** INSTEAD OF トリガー内のすべてのトリガー遷移変数は NULL 可能です。

INSTEAD OF INSERT トリガーで可視の新しい遷移変数の初期値、つまり遷移表の新しい列の初期値は、以下のように設定されます。

- 列の値が INSERT ステートメントで明示的に指定された場合、対応する新しい遷移変数つまり遷移表の新しい列は、明示的に指定されたその値になります。
- 列の値が INSERT ステートメントで明示的に指定されないか、または DEFAULT キーワードが指定された場合、対応する新しい遷移変数つまり遷移表の新しい列は、次のようになります。

  - ビュー列が (INSTEAD OF トリガーなしで) 更新可能であり、生成された列または ROWID に基づいていない場合は、基本表の列のデフォルト値になります。
  - その他の場合は、NULL 値になります。

INSTEAD OF UPDATE トリガーで可視の新しい遷移変数の初期値、つまり遷移表の新しい列の初期値は、以下のように設定されます。

- 列の値が UPDATE ステートメントで明示的に指定された場合、対応する新しい遷移変数つまり遷移表の新しい列は、明示的に指定されたその値になります。
- UPDATE ステートメントで列に DEFAULT キーワードが明示的に指定された場合、対応する新しい遷移変数つまり遷移表の新しい列は、次のようになります。
  - ビュー列が (INSTEAD OF トリガーなしで) 更新可能であり、生成された列または ROWID に基づいていない場合は、基本表の列のデフォルト値になります。
  - その他の場合は、NULL 値になります。
- これら以外の場合、対応する新しい遷移変数つまり遷移表の新しい列は、行内の列の既存の値になります。

難読化されたステートメント**:** CREATE TRIGGER ステートメントは、難読化された形式で実行できます。難読化されたステートメントでは、トリガー名のみが判読可能で、その後に WRAPPED キーワードが続きます。ステートメントの残りは、判読できないが、難読化されたステートメントをサポートするデータベース・サーバーによってデコード可能なように、エンコードされます。難読化されたステートメントは、WRAP スカラー関数を呼び出すことによって生成できます。難読化されたステートメントからトリガーが作成されるときに指定されるデバッグ・オプションはすべて無視されます。

難読化されたステートメントから作成されたトリガーを難読化がサポートされていないリリースにリストアすることはできません。したがって、難読化されたステートメントから作成されたトリガーを伴う表を、難読化がサポートされていないリリースにリストアすることはできません。

SECURED オプションを使用したトリガー作成**:** トリガーは、 CREATE TRIGGER ステートメントが実行された後はセキュアであると見なされます。Db2 は SECURED 属性を、トリガー本体のすべてのアクティビティーに対する監査手順をユーザーが確立したことを宣言するアサーションとして扱います。セキュア・トリガーがユーザー定義関数を参照する場合、Db2 では妥当性検査を実行せずに、このようなユーザー定義関数がセキュアであると想定します。このような関数が機密データにアクセスする可能性がある場合、セキュリティー管理者権限を持つユーザーは、それらの関数がそのデータにアクセスすることを許可されていること、それらの関数のための監査手順が整っていること、および、後続のすべての ALTER FUNCTION ステートメントがこの監査手順によってレビューされることを確認する必要があります。

トリガーのサブジェクト表が行アクセス制御または列アクセス制御を使用している場合、そのトリガーはセキュアでなければなりません。また、トリガーがビューに

対して作成され、ビュー定義内にある基礎表の 1 つ以上がアクティブな行アクセス制御または列アクセス制御を使用している場合、そういったトリガーにも SECURED を指定する必要があります。

NOT SECURED オプションを使用したトリガー作成**:** CREATE TRIGGER ステートメントは、トリガーのサブジェクト表が行アクセス制御または列アクセス制御を使用している場合、または、サブジェクト表がビューであり、ビュー内の基礎表の 1 つ以上が行アクセス制御または列アクセス制御を使用している場合、エラーを戻します。

遷移変数値および行アクセス制御と列アクセス制御**:** 遷移変数および遷移表には行アクセス制御と列アクセス制御は適用されません。トリガー表に行アクセス制御または列アクセス制御が適用されている場合、遷移変数および遷移表の初期値には行の許可および列マスクは適用されません。トリガー表に適用されている行アクセス制御および列アクセス制御は、トリガー本体において参照されている遷移変数および遷移表に対して、またはトリガー本体内で呼び出されるユーザー定義関数に引数として渡される遷移変数および遷移表に対しては無視されます。SQL ステートメントが遷移変数または遷移表の機密データにアクセスすることに関してセキュリティー上の問題がないことを確実にするため、SECURED オプションを使用してトリガーを作成してください。トリガーがセキュアでない場合、行アクセス制御および列アクセス制御をトリガー表に対して適用できず、 CREATE TRIGGER ステートメントはエラーを戻します。

複数のトリガー**:** 1 つの表に対してトリガー SQL 操作と起動時が同一の複数のトリガーを定義できます。トリガーは、モードおよび作成された順序に基づいて活動化されます。

- 最初に MODE DB2ROW トリガー (および ADDPFTRG CL コマンドによって作成された固有トリガー) が、作成された順序で起動されます。
- 次に MODE DB2SQL トリガーが、作成された順序で起動されます。

最初に作成された MODE DB2ROW トリガーが最初に実行され、2 番目に作成された MODE DB2ROW トリガーが 2 番目に実行されるというようになります。

ソース表には、最大 300 のトリガーを追加できます。

REPLACE でトリガーを再作成するときに、活動化順序内の位置は維持されません。トリガーをいったん削除して作成し直した場合と同じ動作になります。

対照表またはトリガー・アクション内で参照されている表への列の追加**:** トリガーを定義した後で対照表に列を追加する場合は、次の規則が適用されます。

- 列リストが明示的に定義されていない UPDATE トリガーの場合、新規の列の更新時にトリガーが起動されます。
- トリガー・アクション 内の SQL ステートメントがトリガー表を参照している場合、トリガーを再作成するまでは、新規の列は SQL ステートメントに関連付けられません。
- OLD_TABLE および NEW_TABLE 遷移表には新規の列は含まれますが、トリガーを再作成しない限り、その列を参照することはできません。

トリガー・アクション内の SQL ステートメントによって参照されている表に列を追加した場合、トリガーを再作成するまでは、新規の列は SQL ステートメントに関連付けられません。

トリガー・アクションの中で参照されている表に対する特権の除去または取り消し: トリガー・アクション の中で参照されている表、ビュー、別名などのオブジェクトを除去すると、トリガーの起動時に、そのオブジェクトを参照しているステートメントのアクセス・プランが再作成されます。その時点でそのオブジェクトが存在していない場合は、対象表についての対応する INSERT、UPDATE、または DELETE 操作は失敗します。

トリガーの作成者がトリガー実行のために必要としている特権が取り消された場合は、トリガーの起動時に、そのオブジェクトを参照しているステートメントのアクセス・プランが再作成されます。その時点でその特権が存在していない場合は、対象表についての対応する INSERT、UPDATE、または DELETE 操作は失敗します。

トリガーの実行時のエラー: *SQL-trigger-body* で SIGNAL ステートメントを実行すると、SQLCODE -438 と、SIGNAL ステートメントで指定した SQLSTATE が返されます。

SQL-trigger-body のステートメントの実行時に発生する他のエラーは、SQLSTATE 09000 と SQLCODE -723 で返されます。

トリガーにおける特殊レジスター: トリガーが活動化される前に特殊レジスターの値は保管され、トリガーからの戻りにおいてリストアされます。特殊レジスターの値は、トリガーする SQL 操作から継承されます。

トランザクション分離: すべてのトリガーは活動化されると、トリガーを呼び出すアプリケーション・プログラムの分離レベルがトリガー・プログラムのデフォルトの分離レベルと同じである場合を除いて、 SET TRANSACTION ステートメントを実行します。トリガーによる操作をすべてそのトリガーを起動させたアプリケーション・プログラムと同じ分離レベルで実行するために、これが必要になります。ただし、ユーザーは独自の SET TRANSACTION ステートメントを、トリガーの *SQL* トリガー本体 内の *SQL* 制御ステートメント に組み込むことができます。ユーザーが SET TRANSACTION ステートメントをトリガーの *SQL* トリガー本体 に組み込んだ場合、トリガーは、そのトリガーを起動させたアプリケーション・プログラムの分離レベルではなく、 SET TRANSACTION ステートメントで指定された分離レベルで実行されます。

トリガーを起動させたアプリケーション・プログラムが No Commit (COMMIT(*NONE) または COMMIT(*NC)) 以外の分離レベルで実行されている場合、トリガー内部の操作はコミットメント制御下で実行され、アプリケーションが現行作業単位をコミットするまでは、コミットやロールバックは行われません。トリガーの *SQL* トリガー本体 で ATOMIC が指定され、ATOMIC トリガーを起動させたアプリケーション・プログラムが分離レベル No Commit (COMMIT(*NONE) または COMMIT(*NC)) で実行されている場合、トリガー内部の操作はコミットメント制御下では実行されません。トリガーを起動させたアプリ

ケーションが分離レベル No Commit (COMMIT(*NONE) または COMMIT(*NC)) で実行されている場合、トリガーの操作は即時にデータベースに書き込まれ、ロールバックすることはできません。

ある表に対して、物理ファイルトリガー追加 (ADDPFTRG) CL コマンドによって定義されたシステム・トリガーと、CREATE TRIGGER ステートメントによって定義された SQL トリガーの両方が定義されている場合、システム・トリガーが SET TRANSACTION ステートメントを実行して、トリガーを起動した元のアプリケーションと同じ分離レベルで実行されるようにすることをお勧めします。また、システム・トリガーは、呼び出し元アプリケーションの活動化グループ内で実行することもお勧めします。システム・トリガーを別の活動化グループ (ACTGRP(*NEW)) で実行すると、それらのシステム・トリガーは、呼び出し元アプリケーションの作業単位に参加せず、SQL トリガーの作業単位にも参加しません。別の活動化グループで実行されるシステム・トリガーは、コミットメント制御下で実行したデータベース操作をコミットまたはロールバックする責任があります。 CREATE TRIGGER ステートメントによって定義された SQL トリガーは、常に呼び出し元の活動化グループ内で実行されます。

トリガー・アプリケーションがコミットメント制御を使用して実行されている場合、SQL トリガーの操作およびカスケード SQL トリガーは、副作業単位に取り込まれます。トリガーの操作およびカスケード・トリガーが成功した場合、副作業単位に取り込まれた操作は、トリガー・アプリケーションが現行の作業単位をコミットまたはロールバックする時点で、コミットまたはロールバックされます。呼び出し元と同じ活動化グループ内で実行され、呼び出し元の分離レベルで SET TRANSACTION を実行するシステム・トリガーも、副作業単位に参加します。トリガー・アプリケーションがコミットメント制御を使用せずに実行されている場合、SQL トリガーの操作もコミットメント制御を使用せずに実行されます。

トリガーを起動させたアプリケーションが分離レベル No Commit (COMMIT(*NONE) または COMMIT(*NC)) で実行されており、INSERT ステートメント、 UPDATE ステートメント、または DELETE ステートメントを実行して、ステートメントの実行中にエラーが発生した場合、その操作のエラーの後は、他のシステム・トリガーや SQL トリガーは起動されません。ただし、いくつかの変更は既に行われています。トリガー・アプリケーションがコミットメント制御を使用して実行されている場合、副作業単位に取り込まれたトリガーの操作は、最初のエラーが検出された時点でロールバックされ、現行の INSERT、UPDATE、または DELETE ステートメントの追加のトリガーは起動されません。

パフォーマンスの考慮**:** トリガーを起動させたアプリケーション・プログラムによって最も頻繁に使用される分離レベルの下にトリガーを作成します。 SET OPTION ステートメントを使用して、明示的に分離レベルを選択することができます。

ROW トリガー (特に、MODE DB2ROW トリガー) は、 TABLE レベル・トリガーよりもパフォーマンスの面でより優れています。

暗黙的に隠される列に関する考慮事項**:** 暗黙的に隠されると定義した列に対しては、遷移変数が存在します。暗黙的な隠し列に対応する遷移変数は、トリガーの本体で参照できます。

カタログのトリガー・アクション: トリガーの作成時に、トリガー・アクションは、CREATE TRIGGER ステートメントの結果として、次のように変更されます。

- 命名方式が SQL 命名に切り替えられます。
- 未修飾のオブジェクト参照子は、すべて明示的に修飾されます。
- すべての暗黙の列リスト (例えば、SELECT *、列リストのない INSERT、UPDATE SET ROW) は、実際の列名リストに展開されます。

変更されたトリガー・アクション は、カタログに保管されます。

トリガー・アクション内で参照されている表の名前変更または移動: トリガー・アクション 内で参照されている表はすべて (対象表を含む) 移動や名前変更が可能です。ただし、トリガー・アクション は、引き続き古い名前やスキーマを参照します。トリガー・アクション の実行時に参照された表が見つからないと、エラーになります。そのため、トリガーをいったん除去した後、トリガーを再作成して、名前変更または移動した表を参照するようにする必要があります。

日時に関する考慮事項: OLD ROW または NEW ROW が指定されている場合、日付または時刻定数、あるいはトリガー・アクション 内の SQL ステートメントで使用されている変数内の日付と時刻のストリング表現は、ISO、EUR、JIS、USA 形式であるか、または DDS および CRTPF CL コマンドを使用して表を作成した場合は、その表の作成時に指定された日時形式に一致していなければなりません。 DDS 指定に複数の異なる日時形式が含まれている場合、トリガーは作成できません。

トリガーを無効にする操作: 作動不能トリガー とは、もう起動して利用できなくなったトリガーをいいます。トリガーが無効になると、対象表または対象ビューに対する INSERT、UPDATE、または DELETE 操作は行えません。トリガーが無効になるのは、次のような場合です。

- トリガー・アクション 内の SQL ステートメントが対象表または対象ビューを参照しており、トリガーが自己参照トリガーであるときに、その表またはビューをシステム CRTDUPOBJ CL コマンドを使用して複写した場合。
- トリガー・アクション 内の SQL ステートメントが from ライブラリー内の表またはビューを参照しており、システム CRTDUPOBJ CL コマンドを使用して表またはビューを複写したときに、オブジェクトが新規ライブラリー内で見つからない場合。
- システム RSTOBJ または RSTLIB CL コマンドを使用して表またはビューを新規ライブラリーに復元したときに、トリガー・アクション が対象表または対象ビューを参照しており、トリガーが自己参照トリガーである場合。

無効トリガーは、最初にそれを除去してから、CREATE TRIGGER ステートメントを使用して再作成しなければなりません。 対象表に対してトリガー操作および起動時が同一の複数のトリガーが定義されている場合、トリガーの除去と再作成は、トリガーの起動順序に影響を与えるので注意が必要です。

トリガー・プログラム・オブジェクト: トリガーが作成されるときに、SQL は、組み込み SQL ステートメントを含む C ソース・コードが入った一時ソース・ファイルを作成します。次いで、CRTPGM コマンドを使用して、プログラム・オブジェクトを作成します。プログラムの作成に使用される SQL オプションは、CREATE

TRIGGER ステートメントの実行時に有効なオプションです。プログラムは、ACTGRP(*CALLER) を使用して作成します。

プログラムは、STGMDL(*SNGLVL) を使用して作成します。STGMDL (*TERASPACE) とさらにコミットメント制御も使用するアプリケーションのためのトリガーを実行する場合は、そのアプリケーション全体を、ジョブを有効範囲とするコミットメント定義 (STRCMTCTL CMTSCOPE(*JOB)) のもとで実行する必要があります。

トリガーは、トリガーの所有者 の借用権限を使用して実行されます。

例

例 *1:* 会社が管理する従業員の数を追跡する 2 つのトリガーを作成します。トリガー表は EMPLOYEE 表で、トリガーは COMPANY_STATS 表の総従業員数の列を増分または減分します。COMPANY_STATS 表のプロパティーは、次のとおりです。

```
CREATE TABLE COMPANY_STATS
  (NBEMP INTEGER,
   NBPRODUCT INTEGER,
   REVENUE DECIMAL(15,0))
```

この例は、行トリガーを使用して、要約データを別表に保守します。

最初のトリガー NEW_HIRE を作成して、新しい従業員が雇用されるたびに従業員数を増分するようにします。つまり、新しい行が EMPLOYEE 表に挿入されるたびに、表 COMPANY_STATS の列 NBEMP を 1 だけ増分します。

```
CREATE TRIGGER NEW_HIRE
  AFTER INSERT ON EMPLOYEE
  FOR EACH ROW MODE DB2SQL
    UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1
```

2 番目のトリガー FORM_EMP を作成して、従業員が会社を辞めるたびに従業員数を減分するようにします。つまり、表 EMPLOYEE から行が削除されるたびに、表 COMPANY_STATS の列 NBEMP を 1 だけ減分します。

```
CREATE TRIGGER FORM_EMP
  AFTER DELETE ON EMPLOYEE
  FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
    UPDATE COMPANY_STATS SET NBEMP = NBEMP - 1;
  END
```

例 *2:* トリガー REORDER を作成します。これは、部品レコードが更新され、該当部品の手持ちの数量が最大在庫量の 10% を下回るたびに出荷要求を出すために、ユーザー定義関数 ISSUE_SHIP_REQUEST を呼び出します。ユーザー定義関数 ISSUE_SHIP_REQUEST は、その部品の最大在庫量から手持ちの数量を差し引いた数量の部品を発注します。この関数は、同じ PARTNO をオーダーする重複する要求を除去し、固有のオーダーを該当する製造業者に送信します。

この例は、行トリガーではなく、ステートメント・トリガーとしてトリガーを定義する方法も示しています。WHERE 文節で真と評価された遷移表内の各行について、その部品の出荷要求が出されます。

```
CREATE TRIGGER REORDER
  AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
  REFERENCING NEW_TABLE AS NTABLE
  FOR EACH STATEMENT MODE DB2SQL
  BEGIN ATOMIC
    DECLARE REQUEST_VAR INT;
    SELECT ISSUE_SHIP_REQUEST(MAX_STOCKED - ON_HAND, PARTNO)
    INTO REQUEST_VAR
    FROM NTABLE
    WHERE ON_HAND < 0.10 * MAX_STOCKED;
  END
```

例 *3:* 表 EMPLOYEE に列 SALARY が含まれると想定します。従業員の給料を 20% を超えて更新できないようにし、エラーのシグナルを送るトリガー SAL_ADJ を作成します。 75001 の SQLSTATE で戻されるエラーおよび記述を持ちます。この例では、SIGNAL SQLSTATE ステートメントが、ビジネス規則に違反する変更を制限するのに役立つことを示しています。

```
CREATE TRIGGER SAL_ADJ
  AFTER UPDATE OF SALARY ON EMPLOYEE
  REFERENCING OLD AS OLD_EMP
              NEW AS NEW_EMP
  FOR EACH ROW MODE DB2SQL
  WHEN (NEW_EMP.SALARY > (OLD_EMP.SALARY *1.20))
  BEGIN ATOMIC
    SIGNAL SQLSTATE '75001'('Invalid Salary Increase - Exceeds 20%');
  END
```

CREATE TYPE

CREATE TYPE ステートメントは、ユーザー定義のデータ・タイプを現在のサーバーで定義します。

以下のタイプのユーザー定義データ・タイプを定義できます。

- 配列

  通常配列であるユーザー定義データ・タイプ。配列タイプのエレメントは、組み込みデータ・タイプのいずれか 1 つに基づいています。 1323 ページの『CREATE TYPE (配列)』を参照してください。

- 特殊

  組み込みデータ・タイプの 1 つと共通表現を共有するユーザー定義データ・タイプ。ユーザー定義特殊タイプの作成時に、ユーザー定義特殊タイプとソース組み込みデータ・タイプの間でキャストする関数が生成されます。オプションで、ユーザー定義特殊タイプの作成時に、ユーザー定義特殊タイプと共に使用する比較演算のサポートを生成することもできます。 1328 ページの『CREATE TYPE (特殊)』を参照してください。

CREATE TYPE (配列)

CREATE TYPE (配列) ステートメントは、現行サーバーで配列タイプを定義します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、あるいは対話式に実行することもできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- スキーマ内に作成する特権。詳しくは、スキーマ内で作成する必要のある権限を参照してください。
- データベース管理者権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

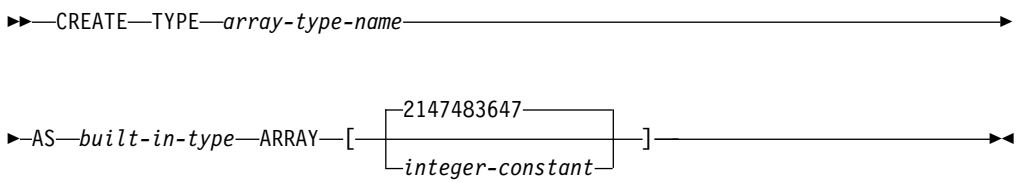
- SYSTYPES カタログ表の場合
  - 該当の表に対する INSERT 特権、および
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

SQL 名が指定され、配列タイプが作成されるライブラリーと同じ名前のユーザー・プロファイルが存在し、しかも、その名前がステートメントの権限 ID と異なっている場合、ステートメントの権限 ID が保持している特権には、少なくとも次のいずれか 1 つを含める必要があります。

- その名前を持つユーザー・プロファイルに対する *ADD システム権限
- データベース管理者権限

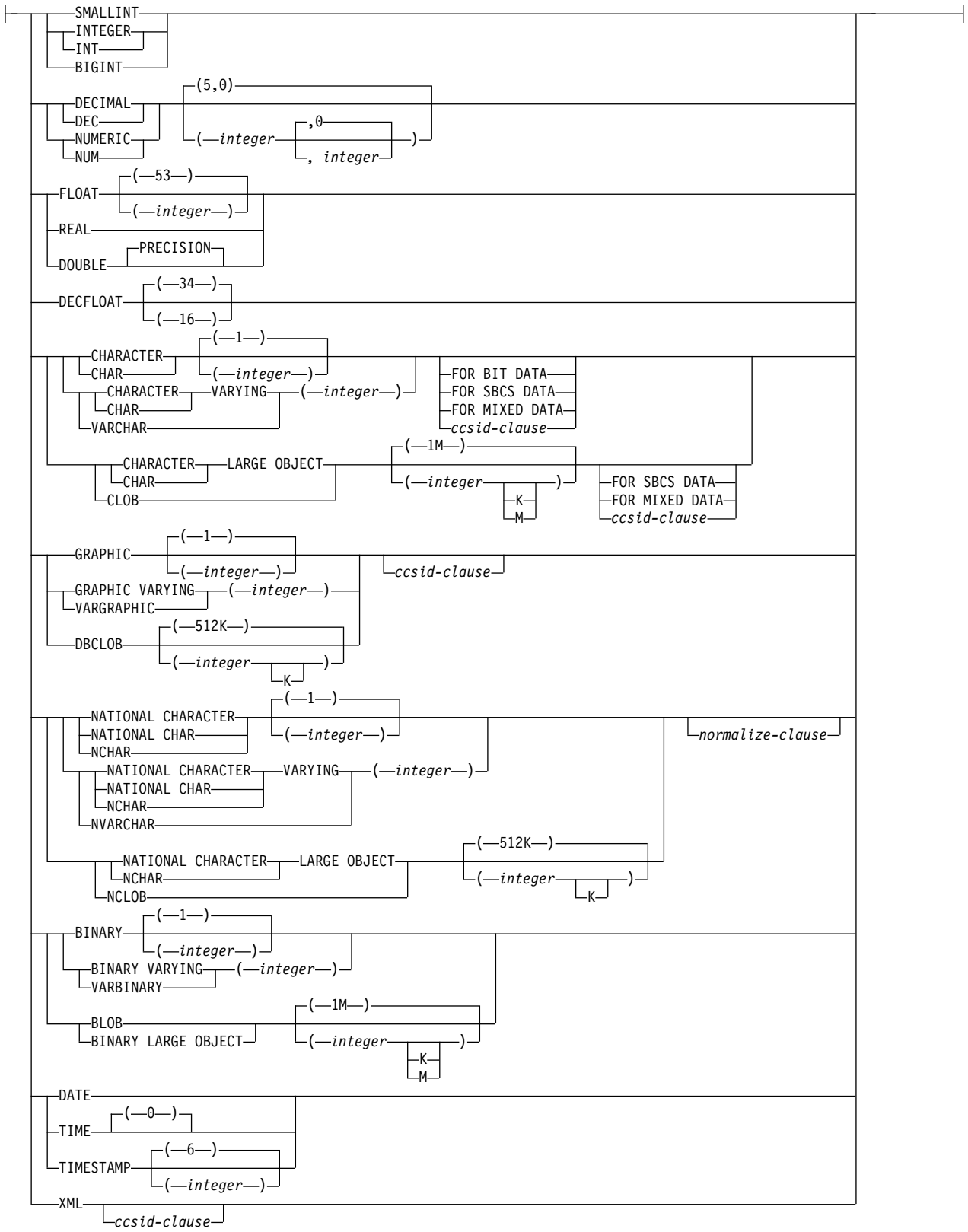
SQL 特権に対応するシステム権限については、『表またはビューへの権限を検査する際の対応するシステム権限』を参照してください。

構文

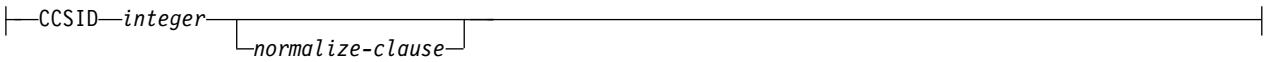


built-in-type:

CREATE TYPE (配列)



ccsid-clause:



normalize-clause:

説明

array-type-name

配列の名前を指定します。暗黙的または明示的修飾子も含め、この名前は、現行サーバーに既に存在している特殊タイプまたは配列タイプと同じ名前にすることはできません。

SQL 名が指定されている場合、配列タイプは、暗黙的または明示的修飾子で指定しているスキーマ内に作成されます。

システム名が指定されている場合、配列タイプは、修飾子で指定しているスキーマ内に作成されます。修飾されない場合:

- CURRENT SCHEMA 特殊レジスターの値が *LIBL である場合、配列タイプは、現行ライブラリー (*CURLIB) 内に作成されます。
- そうでない場合、配列タイプは現行スキーマ内に作成されます。

配列タイプが有効なシステム名でない場合、Db2 for i は、システム名を生成します。名前の生成に関する規則については、 1298 ページの『表名の生成の規則』を参照してください。

配列タイプ名 は、組み込みデータ・タイプの名前にすることはできません。また、下記のシステム予約のどのキーワードにすることもできません。これは、それらのキーワードを区切り文字付き ID として指定している場合にも該当します。

=	<	>	>=
<=	<>	¬=	¬<
¬<	!=	!<	!>
ALL	FALSE	POSITION	XMLAGG
AND	FOR	RID	XMLATTRIBUTES
ANY	FROM	RRN	XMLCOMMENT
ARRAY_AGG	HASHED_VALUE	SELECT	XMLCONCAT
BETWEEN	IN	SIMILAR	XMLDOCUMENT
BOOLEAN	INTERVAL	SOME	XMLELEMENT
CASE	IS	STRIP	XMLFOREST
CAST	LIKE	SUBSTRING	XMLGROUP
CHECK	MATCH	TABLE	XMLNAMESPACES
DATAPARTITIONNAME	NODENAME	THEN	XMLPARSE
DATAPARTITIONNUM	NODENUMBER	TRIM	XMLPI
DBPARTITIONNAME	NOT	TRUE	XMLROW
DBPARTITIONNUM	NULL	TYPE	XMLSERIALIZE
DISTINCT	ONLY	UNIQUE	XMLTEXT
EXCEPT	OR	UNKNOWN	XMLVALIDATE
EXISTS	OVERLAPS	WHEN	XSLTRANSFORM
EXTRACT	PARTITION		

修飾付きの配列タイプ名 を指定した場合は、スキーマ名は QSYS、QSYS2、QTEMP、または SYSIBM であってはなりません。

built-in-type
配列のすべてのエレメントのデータ・タイプとして使用する組み込みデータ・タイプを指定します。それぞれの組み込みデータの詳細については、 1238 ページの『CREATE TABLE』を参照してください。

長さ、精度、位取りの属性があるデータ・タイプで特定の値が指定されていない場合は、構文図で示されているデフォルト属性がそのデータ・タイプで暗黙指定されます。

配列タイプがストリング・データ・タイプに対応している場合は、配列タイプの作成時に、CCSID がその配列タイプに関連付けられます。データ・タイプの詳細については、 1238 ページの『CREATE TABLE』 を参照してください。

ARRAY [*integer-constant***]**
配列の最大カーディナリティーとして *integer-constant* を指定します。その値は、0 より大きい正数でなければなりません。値を指定しない場合は、最大整数値 2147483647 が使用されます。配列内に割り当てることのできる配列エレメントの最大数は、4 ギガバイトに収まるエレメント数です。可変長、LOB、および XML の配列エレメントは、それぞれの最大長で割り振られます。

注

追加の生成関数: 配列タイプの変換のための関数は作成されますが、サービス・プログラムは作成されません。したがって、それらの関数に関する特権を付与したり取り消したりすることはできません。

生成されるキャスト関数の名前:キャスト関数のうちの 1 つの非修飾名は ARRAY です。配列タイプに変換するキャスト関数の名前は、その配列タイプの名前です。そのキャスト関数の入力パラメーターのデータ・タイプは、ARRAY のデータ・タイプと同じです。

例えば、T_SHOESIZES という名前の配列タイプが、次のステートメントを使用して作成されると仮定します。

```
CREATE TYPE CLAIRE.T_SHOESIZES AS INT ARRAY[]
```

このステートメントが実行されると、データベース・マネージャーは、次のキャスト関数も生成します。 つまり、配列タイプから配列に変換する ARRAY と、配列から配列タイプに変換する T_SHOESIZES です。

生成されたキャスト関数を明示的に除去することはできません。配列タイプ用に生成される cast 関数は、配列タイプが DROP ステートメントでドロップされると、暗黙的にドロップされます。

配列タイプの属性 : 配列タイプは *SQLUDT オブジェクトとして作成されます。

配列タイプの所有権: SQL 名が指定されている場合、

- 作成した配列タイプが入れられるスキーマと同じ名前のユーザー・プロファイルが存在する場合、配列タイプの所有者 はそのユーザー・プロファイルです。

- その他の場合は、配列タイプの所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

システム名を指定した場合は、配列タイプの所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

配列タイプの権限 : SQL 名を使用する場合は、配列タイプは、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成されます。システム名を使用する場合、配列タイプは、スキーマの作成権限 (CRTAUT) パラメーターによって決められる *PUBLIC に対する権限を使用して作成されます。

配列タイプの所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、その配列タイプに対する権限が与えられます。

例

例 *1:* PHONENUMBERS という名前の配列タイプを作成します。そのエレメントのデータ・タイプは DECIMAL(10,0)、エレメントの最大数は 5 個です。

```
CREATE TYPE PHONENUMBERS AS DECIMAL(10,0) ARRAY[5]
```

例 *2:* スキーマ GENERIC で NUMBERS という名前の配列タイプを作成します。エレメントの最大数は不明です。

```
CREATE TYPE GENERIC.NUMBERS
          AS BIGINT ARRAY[]
```

CREATE TYPE (特殊)

CREATE TYPE (特殊) ステートメントは、現行サーバー上に特殊タイプを定義します。特殊タイプは、常に組み込みデータ・タイプの 1 つをソースとして作成されます。

ステートメントの実行が正常に完了すると、以下のものも生成されます:

- 特殊タイプからそのソース・タイプにキャストする 1 つの関数
- ソース・タイプからその特殊タイプにキャストする 1 つの関数
- 該当する場合、特殊タイプを持つ比較演算子の使用をサポートします。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- スキーマ内に作成する特権。詳しくは、スキーマ内で作成する必要のある権限を参照してください。
- データベース管理者権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

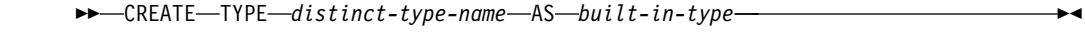
- SYSTYPES カタログ表の場合
  - 該当の表に対する INSERT 特権、および
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

SQL 名が指定され、特殊タイプが作成されるライブラリーと同じ名前のユーザー・プロファイルが存在し、しかも、その名前がステートメントの権限 ID と異なっている場合、ステートメントの権限 ID が保持している特権には、少なくとも次のいずれか 1 つを含める必要があります。

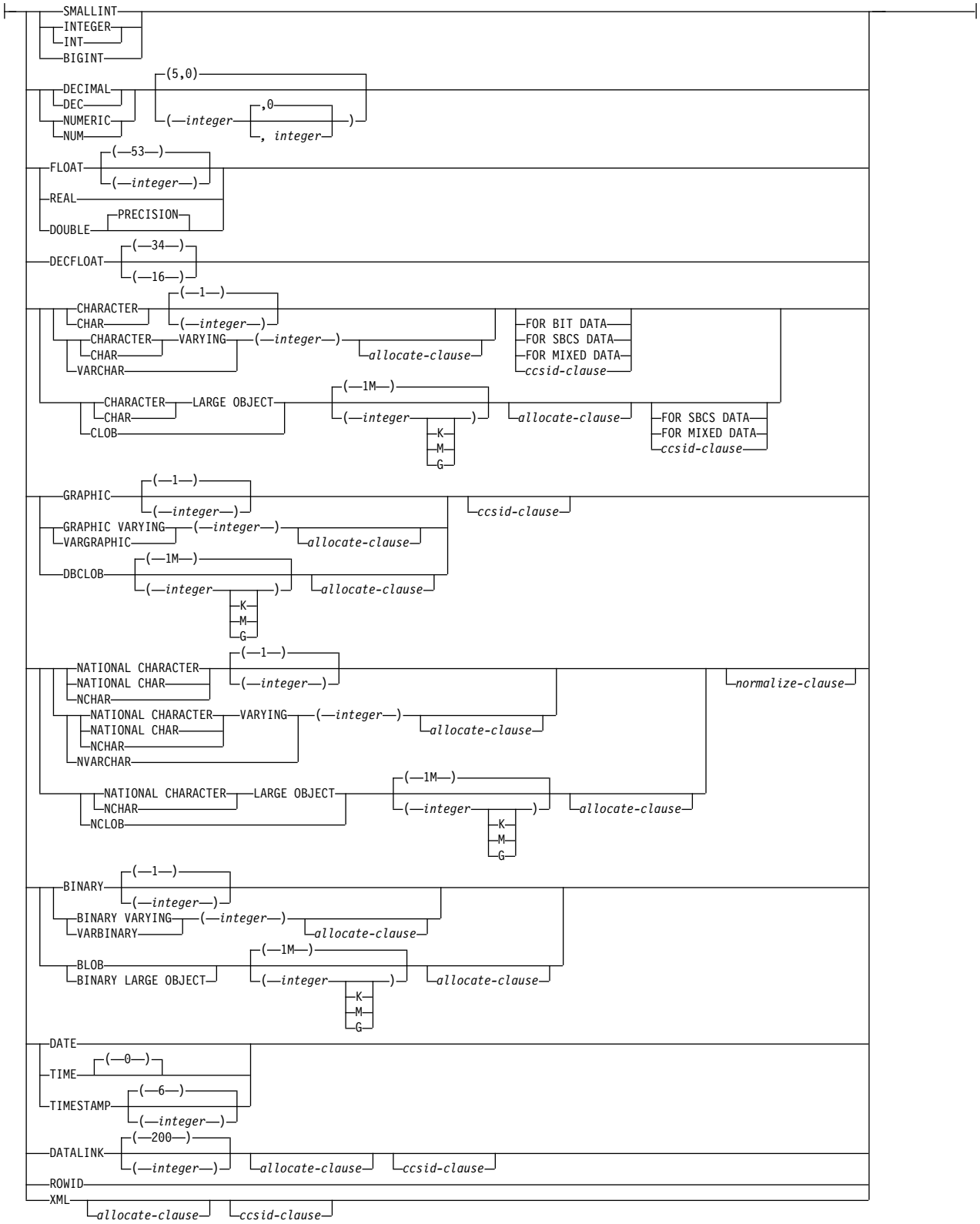
- その名前を持つユーザー・プロファイルに対する *ADD システム権限
- データベース管理者権限

SQL 特権に対応するシステム権限については『表またはビューへの権限を検査する際の対応するシステム権限』を参照してください。

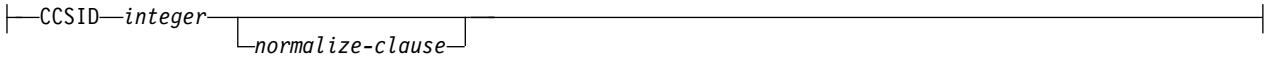
構文



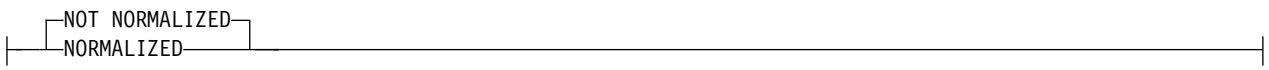
built-in-type:



ccsid-clause:



normalize-clause:



説明

distinct-type-name

特殊タイプを指定します。暗黙的または明示的修飾子も含め、この名前は、現行サーバーに既に存在している特殊タイプまたは配列タイプと同じ名前にすることはできません。

SQL 名が指定されている場合、特殊タイプは、暗黙的または明示的修飾子で指定しているスキーマ内に作成されます。

システム名が指定されている場合、特殊タイプは、修飾子で指定しているスキーマ内に作成されます。修飾されない場合:

- CURRENT SCHEMA 特殊レジスターの値が *LIBL である場合、特殊タイプは、現行ライブラリー (*CURLIB) 内に作成されます。
- そうでない場合、特殊タイプは現行スキーマ内に作成されます。

特殊タイプが有効なシステム名でない場合、Db2 for i は、システム名を生成します。名前の生成に関する規則については、 1298 ページの『表名の生成の規則』を参照してください。

特殊タイプ名 は、組み込みデータ・タイプの名前にすることはできません。また、下記のシステム予約のどのキーワードにすることもできません。これは、それらのキーワードを区切り文字付き ID として指定している場合にも該当します。

=	<	>	>=
<=	<>	¬=	¬<
¬<	!=	!<	!>
ALL	FALSE	POSITION	XMLAGG
AND	FOR	RID	XMLATTRIBUTES
ANY	FROM	RRN	XMLCOMMENT
ARRAY_AGG	HASHED_VALUE	SELECT	XMLCONCAT
BETWEEN	IN	SIMILAR	XMLDOCUMENT
BOOLEAN	INTERVAL	SOME	XMLELEMENT
CASE	IS	STRIP	XMLFOREST
CAST	LIKE	SUBSTRING	XMLGROUP
CHECK	MATCH	TABLE	XMLNAMESPACES
DATAPARTITIONNAME	NODENAME	THEN	XMLPARSE
DATAPARTITIONNUM	NODENUMBER	TRIM	XMLPI
DBPARTITIONNAME	NOT	TRUE	XMLROW
DBPARTITIONNUM	NULL	TYPE	XMLSERIALIZE
DISTINCT	ONLY	UNIQUE	XMLTEXT
EXCEPT	OR	UNKNOWN	XMLVALIDATE
EXISTS	OVERLAPS	WHEN	XSLTRANSFORM

EXTRACT	PARTITION		

修飾付きの特殊タイプ名 を指定した場合は、スキーマ名は QSYS、QSYS2、QTEMP、または SYSIBM であってはなりません。

built-in-type
特殊タイプの内部表示のベースとして使用される組み込みデータ・タイプを指定します。それぞれの組み込みデータの詳細については、 1238 ページの『CREATE TABLE』を参照してください。

プラットフォーム間でのアプリケーションの移植性を保つには、推奨される次のデータ・タイプ名を使用します。

- FLOAT の代わりに DOUBLE または REAL。
- NUMERIC の代わりに DECIMAL。

長さ、精度、位取りの属性があるデータ・タイプで特定の値が指定されていない場合は、構文図で示されているデフォルト属性がそのデータ・タイプで暗黙指定されます。

特殊タイプのソースがストリング・データ・タイプの場合、CCSID は、その特殊タイプの作成時の特殊データ・タイプに関連しています。データ・タイプの詳細については、 1238 ページの『CREATE TABLE』 を参照してください。

注

追加の生成関数: 上記のシステム生成比較演算子のほかに、ソース・タイプの変換のために以下の関数を使用できるようになります。

- 特殊タイプからソース・タイプへ
- ソース・タイプから特殊タイプへ
- INTEGER から特殊タイプへ (ソース・タイプが SMALLINT の場合)
- DOUBLE から特殊タイプへ (ソース・タイプが REAL の場合)
- VARCHAR から特殊タイプへ (ソース・タイプが CHAR の場合)
- VARGRAPHIC から特殊タイプへ (ソース・タイプが GRAPHIC の場合)

これらの関数は、次のステートメントを実行した場合と同様に作成されます (ただし、サービス・プログラムは作成されないので、これらの関数に対する特権を認可したり取り消したりすることはできません)。

```
CREATE FUNCTION source-type-name (distinct-type-name)
  RETURNS source-type-name

CREATE FUNCTION distinct-type-name (source-type-name)
  RETURNS distinct-type-name
```

生成されたキャスト関数の名前: 1332 ページの表 84 には、生成されたキャスト関数に関する詳細が記載されています。特殊タイプからソース・タイプに変換するキャスト関数の非修飾名は、そのソース・データ・タイプの名前です。

CREATE TYPE ステートメントのソース・データ・タイプに長さ、精度、または位取りを指定した場合、特殊タイプからソース・タイプに変換するキャスト関数の非修飾名は、単に、そのソース・データ・タイプの名前です。キャスト関数が戻す値

のデータ・タイプには、CREATE TYPE ステートメントのソース・データ・タイプに指定した長さ、精度、または位取りの値がすべて組み込まれます。

ソース・タイプから特殊タイプに変換するキャスト関数の名前は、その特殊タイプの名前です。キャスト関数の入力パラメーターには、長さ、精度、および位取りも含め、ソース・データ・タイプと同じデータ・タイプが指定されます。

生成されるキャスト関数は、特殊タイプと同じスキーマで作成されます。 同じ名前と同じ関数シグニチャーを持つ関数が、現行サーバー内に既に存在していてはなりません。

例えば、T_SHOESIZE という名前の特殊タイプが、次のステートメントを使用して作成されると仮定します。

```
CREATE TYPE CLAIRE.T_SHOESIZE AS VARCHAR(2) WITH COMPARISONS
```

このステートメントが実行されると、データベース・マネージャーは、次のキャスト関数も生成します。 VARCHAR は、特殊タイプからソース・タイプに変換し、T_SHOESIZE は、ソース・タイプから特殊タイプに変換します。

```
FUNCTION CLAIRE.VARCHAR (CLAIRE.T_SHOESIZE) RETURNS VARCHAR(2)

FUNCTION CLAIRE.T_SHOESIZE (VARCHAR(2) RETURNS CLAIRE.T_SHOESIZE
```

関数 VARCHAR は、データ・タイプ VARCHAR(2) と一緒に値を戻し、関数 T_SHOESIZE には、データ・タイプ VARCHAR(2) と一緒に入力パラメーターが指定されます。

生成されたキャスト関数を明示的に除去することはできません。特殊タイプに対して生成されるキャスト関数は、その特殊タイプが DROP ステートメントで除去される時点で暗黙に除去されます。

次の表は、特殊タイプに対してソース・データ・タイプにすることができる組み込みデータ・タイプごとに、生成されたキャスト関数の名前、入力パラメーターのデータ・タイプ、およびそれらの関数が戻す値のデータ・タイプを示します。

表 *84.* 特殊タイプに対するキャスト関数

ソース・タイプ名	関数名	パラメーター・タイプ	戻りタイプ
SMALLINT	*distinct-type-name*	SMALLINT	*distinct-type-name*
	distinct-type-name	INTEGER	*distinct-type-name*
	SMALLINT	*distinct-type-name*	SMALLINT
INTEGER	*distinct-type-name*	INTEGER	*distinct-type-name*
	INTEGER	*distinct-type-name*	INTEGER
BIGINT	*distinct-type-name*	BIGINT	*distinct-type-name*
	BIGINT	*distinct-type-name*	BIGINT
DECIMAL	*distinct-type-name*	DECIMAL(p,s)	*distinct-type-name*
	DECIMAL	*distinct-type-name*	DECIMAL(p,s)
NUMERIC	*distinct-type-name*	NUMERIC(p,s)	*distinct-type-name*
	NUMERIC	*distinct-type-name*	NUMERIC(p,s)

表 84. 特殊タイプに対するキャスト関数 (続き)

ソース・タイプ名	関数名	パラメーター・タイプ	戻りタイプ
REAL または FLOAT(n) (ここで、n <= 24)	*distinct-type-name*	REAL	*distinct-type-name*
	distinct-type-name	DOUBLE	*distinct-type-name*
	REAL	*distinct-type-name*	REAL
DOUBLE または DOUBLE PRECISION または FLOAT(n) ここで、n > 24	*distinct-type-name*	DOUBLE	*distinct-type-name*
	DOUBLE	*distinct-type-name*	DOUBLE
DECFLOAT	*distinct-type-name*	DECFLOAT(n)	*distinct-type-name*
	DECFLOAT	*distinct-type-name*	DECFLOAT(n)
CHAR	*distinct-type-name*	CHAR(n)	*distinct-type-name*
	CHAR	*distinct-type-name*	CHAR(n)
	distinct-type-name	VARCHAR(n)	*distinct-type-name*
VARCHAR	*distinct-type-name*	VARCHAR(n)	*distinct-type-name*
	VARCHAR	*distinct-type-name*	VARCHAR(n)
CLOB	*distinct-type-name*	CLOB(n)	*distinct-type-name*
	CLOB	*distinct-type-name*	CLOB(n)
GRAPHIC	*distinct-type-name*	GRAPHIC (n)	*distinct-type-name*
	GRAPHIC	*distinct-type-name*	GRAPHIC (n)
	distinct-type-name	VARGRAPHIC (n)	*distinct-type-name*
VARGRAPHIC	*distinct-type-name*	VARGRAPHIC (n)	*distinct-type-name*
	VARGRAPHIC	*distinct-type-name*	VARGRAPHIC (n)
DBCLOB	*distinct-type-name*	DBCLOB(n)	*distinct-type-name*
	DBCLOB	*distinct-type-name*	DBCLOB(n)
BINARY	*distinct-type-name*	BINARY(n)	*distinct-type-name*
	BINARY	*distinct-type-name*	BINARY(n)
	distinct-type-name	VARBINARY(n)	*distinct-type-name*
VARBINARY	*distinct-type-name*	VARBINARY(n)	*distinct-type-name*
	VARBINARY	*distinct-type-name*	VARBINARY(n)
BLOB	*distinct-type-name*	BLOB(n)	*distinct-type-name*
	BLOB	*distinct-type-name*	BLOB(n)
DATE	*distinct-type-name*	DATE	*distinct-type-name*
	DATE	*distinct-type-name*	DATE
TIME	*distinct-type-name*	TIME	*distinct-type-name*
	TIME	*distinct-type-name*	TIME
TIMESTAMP	*distinct-type-name*	TIMESTAMP(p)	*distinct-type-name*
	TIMESTAMP	*distinct-type-name*	TIMESTAMP(p)

表 84. 特殊タイプに対するキャスト関数 (続き)

ソース・タイプ名	関数名	パラメーター・タイプ	戻りタイプ
DATALINK	*distinct-type-name*	DATALINK	*distinct-type-name*
	DATALINK	*distinct-type-name*	DATALINK
ROWID	*distinct-type-name*	ROWID	*distinct-type-name*
	ROWID	*distinct-type-name*	ROWID

可搬性のあるアプリケーションに対して特殊タイプを作成する場合、NUMERIC と FLOAT はお勧めできません。それらの代わりに、DECIMAL と DOUBLE を使用するようにしてください。

組み込み関数: 上の表で説明されている関数は、特殊タイプの定義時に自動的に生成される関数のみです。このため、特殊タイプで自動的にサポートされる組み込み関数 (AVG、MAX、LENGTH など) はありません。組み込み関数を特殊タイプで使用できるのは、組み込み関数に基づくソース化されたユーザー定義関数が特殊タイプに対して作成された後だけです。 1073 ページの『組み込み関数の拡張またはオーバーライド: 』を参照してください。

これらの演算子やキャスト関数を SQL ステートメントで正しく使用するには、特殊タイプに特殊タイプのスキーマ名が組み込まれている必要があります。

特殊タイプの属性 : 特殊タイプは *SQLUDT オブジェクトとして作成されます。

特殊タイプの所有権: SQL 名が指定されている場合、

- 作成した特殊タイプが入れられるスキーマと同じ名前のユーザー・プロファイルが存在する場合、特殊タイプの所有者 はそのユーザー・プロファイルです。
- その他の場合は、特殊タイプの所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

システム名を指定した場合は、特殊タイプの所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

特殊タイプの権限 : SQL 名を使用する場合は、特殊タイプは、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成されます。システム名を使用する場合、特殊タイプは、スキーマの作成権限 (CRTAUT) パラメーターによって決められる *PUBLIC に対する権限を使用して作成されます。

特殊タイプの所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、その特殊タイプに対する権限が与えられます。

代替構文: WITH COMPARISONS 文節は、特殊タイプの 2 つのインスタンスを比較するためにシステム生成の比較演算子を作成するように指定します。この文節は、ステートメントの最後の文節として指定できます。WITH COMPARISONS は、Db2 ファミリーの他製品との互換性を保つために必要な場合に限って使用してください。

旧バージョンの Db2 との互換性

- CREATE TYPE の代わりに CREATE DISTINCT TYPE を指定できます。

例

例 *1:* 組み込み INTEGER データ・タイプをソースとする SHOESIZE という名前の特殊タイプを作成します。

```
CREATE TYPE SHOESIZE AS INTEGER WITH COMPARISONS
```

このステートメントの実行が正常に完了すると、2 つのキャスト関数も生成されます。関数 INTEGER(SHOESIZE) は、データ・タイプ INTEGER が指定された値を戻し、関数 SHOESIZE(INTEGER) は、特殊タイプ SHOESIZE が指定された値を戻します。

例 *2:* 組み込み DOUBLE データ・タイプをソースとする MILES という名前の特殊タイプを作成します。

```
CREATE TYPE MILES
        AS DOUBLE WITH COMPARISONS
```

このステートメントの実行が正常に完了すると、2 つのキャスト関数も生成されます。関数 DOUBLE(MILES) は、データ・タイプ DOUBLE が指定された値を戻し、関数 MILES(DOUBLE) は、特殊タイプ MILES が指定された値を戻します。

例 *3:* 組み込み CHAR データ・タイプをソースとする特殊タイプ T_DEPARTMENT を作成します。

```
CREATE TYPE CLAIRE.T_DEPARTMENT AS CHAR(3)
   WITH COMPARISONS
```

このステートメントの実行が正常に完了すると、次の 3 つのキャスト関数も生成されます:

- 関数 CLAIRE.CHAR は T_DEPARTMENT を入力として取り、データ・タイプ CHAR(3) を持つ値を戻します。
- 関数 CLAIRE.T_DEPARTMENT は CHAR(3) を入力として取り、特殊タイプ T_DEPARTMENT を持つ値を戻します。
- 関数 CLAIRE.T_DEPARTMENT は VARCHAR(3) を入力として取り、特殊タイプ T_DEPARTMENT を持つ値を戻します。

CREATE VARIABLE

CREATE VARIABLE ステートメントは、アプリケーション・サーバーでグローバル変数を定義します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、あるいは対話式に実行することもできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- スキーマ内に作成する特権。詳しくは、スキーマ内で作成する必要のある権限を参照してください。
- データベース管理者権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- SYSVARIABLES カタログ表の場合
  - 該当の表に対する INSERT 特権、および
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

特殊タイプまたはシーケンスを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別される特殊タイプまたはシーケンスに対しては次のもの。
  - その特殊タイプまたはシーケンスに対する USAGE 特権、および
  - 特殊タイプまたはシーケンスが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

関数を参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別される関数に対しては次のもの。
  - その関数に対する EXECUTE 特権、および
  - 関数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

グローバル変数を参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別されるグローバル変数に対しては次のもの。
  - そのグローバル変数に対する READ 特権
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

表またはビューを直接または間接的に参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

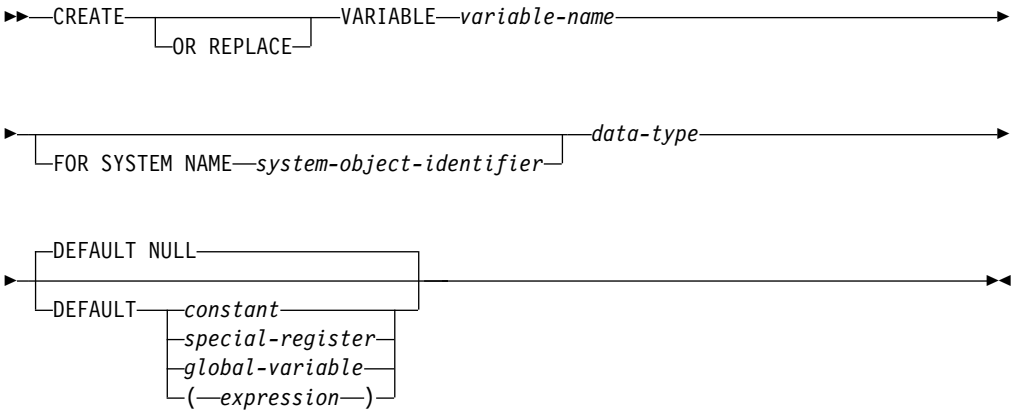
- 直接的または間接的に参照されている表/ビューごとに、以下の権限が必要です。
  - 表やビューに対する SELECT 特権、および
  - 表またはビューが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

既存の変数に置き換えるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

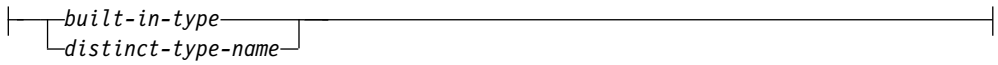
- 次のシステム権限
  - この変数のサービス・プログラムに対する *OBJMGT システム権限
  - この変数を削除するために必要な全権限
  - SYSVARIABLES カタログ表に対する *READ システム権限
- データベース管理者権限

SQL 特権に対応するシステム権限の説明については、『シーケンスへの権限を検査する際の対応するシステム権限』および 『特殊タイプへの権限を検査する際の対応するシステム権限』を参照してください。

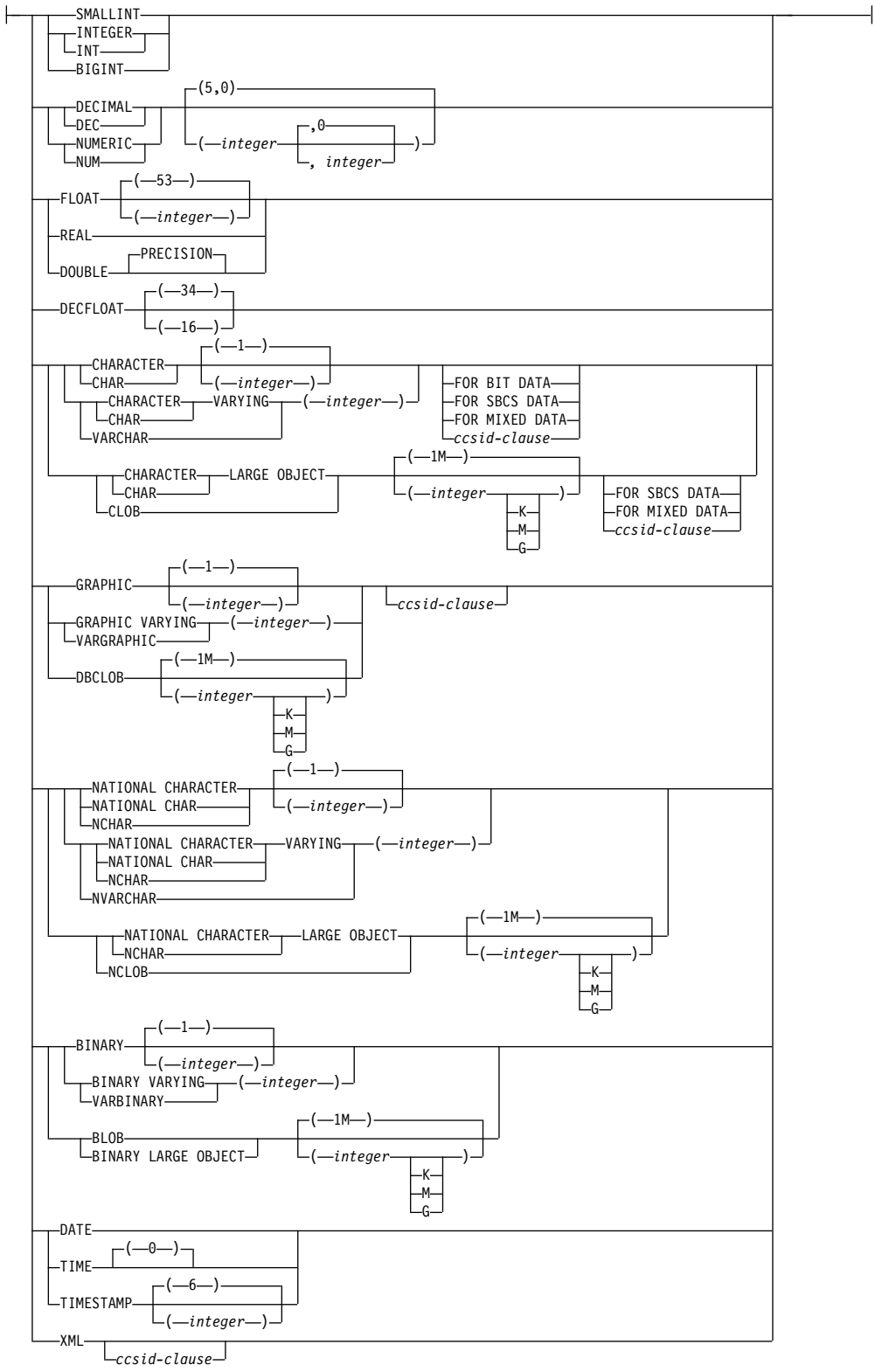
構文



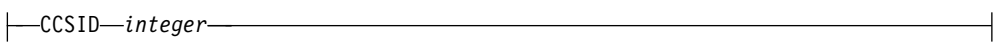
data-type:



built-in-type:



ccsid-clause:



説明

OR REPLACE
現行サーバーにこの変数の定義が存在する場合に、その定義を置き換える、という動作を指定します。実際には、カタログで既存の定義を削除してから新しい定義に置き換える、という動作になりますが、例外として、この変数に対して与えられていた特権は影響を受けません。現行サーバーにこの変数の定義が存在しなければ、このオプションは無視されます。

variable-name
グローバル変数の名前を指定します。 暗黙的または明示的修飾子も含め、この名前で、現行サーバーに既に存在しているグローバル変数を識別することはできません。修飾された変数名を指定する場合、スキーマ名 は、QSYS2、QSYS、QTEMP、または SYSIBM にはできません。

SQL 名が指定されている場合、変数は、暗黙的または明示的修飾子で指定しているスキーマ内に作成されます。

システム名が指定されている場合、変数は、修飾子で指定しているスキーマ内に作成されます。修飾されない場合:

- CURRENT SCHEMA 特殊レジスターの値が *LIBL である場合、変数は、現行ライブラリー (*CURLIB) 内に作成されます。
- そうでない場合、変数は現行スキーマ内に作成されます。

FOR SYSTEM NAME *system-object-identifier*
グローバル変数のシステム・オブジェクト *ID (system-object-identifier)* を示します。 *system-object-identifier* は、現行サーバーに既に存在しているグローバル変数と同じものであってはなりません。システム・オブジェクト *ID* は、非修飾システム ID でなければなりません。

system-object-identifier が指定される場合、*variable-name* は有効なシステム・オブジェクト名であってはなりません。

data-type
データ・タイプまたはグローバル変数を指定します。

built-in-type
組み込みデータ・タイプを指定します。それぞれの組み込みデータの詳細については、 1238 ページの『CREATE TABLE』を参照してください。

distinct-type-name
特殊タイプを指定します。グローバル変数の長さ、精度、位取りは、それぞれ特殊タイプの長さ、精度、位取りです。スキーマ名なしの特殊タイプを指定すると、その特殊タイプ名は、SQL パス上のスキーマを検索することで解決されます。組み込みタイプに当てはまる制限は、特殊タイプのソース・タイプにも当てはまります。

DEFAULT
グローバル変数のデフォルト値を指定します。定数、特殊レジスター、グローバル変数、式、NULL キーワードを値として使用できます。明示的に値が指定されない場合、デフォルト値はその最初の参照時に決定されます。 デフォルト値が指定されていないと、変数は NULL 値に初期化されます。

デフォルトの式で SQL データを変更したり外部アクションを実行したりすることはできません。式を指定する場合は、変数のデータ・タイプとの間に割り当ての互換性がある式でなければなりません。

デフォルトの式で参照する表、ビュー、別名、特殊タイプ、シーケンス、グローバル変数、ユーザー定義関数はすべて、グローバル変数の作成時に現行サーバーに存在していなければなりません。別名が参照している表やビューも、変数の作成時に存在していなければなりません。これには、ライブラリー QTEMP 内のオブジェクトも含まれます。QTEMP 内のオブジェクトはデフォルトの式で参照できますが、QTEMP 内のこれらのオブジェクトを除去しても、グローバル変数は除去されません。

注

セッション有効範囲: グローバル変数の有効範囲はセッションです。つまり、グローバル変数は、データベースでアクティブになっているすべてのセッションで使用できますが、その値は各セッションごとの専用の値になります。

グローバル変数の値の変更: グローバル変数の値の変更は、トランザクション制御の対象ではありません。COMMIT ステートメントの場合でも ROLLBACK ステートメントの場合でも、グローバル変数の値は、トランザクションの終了時に保存されます。

グローバル変数を使用するための特権: このステートメントで作成するグローバル変数の読み取りや書き込みの操作を実行しようとする権限 ID には、そのグローバル変数に対する適切な特権が必要です。変数の定義者には暗黙的にその変数へのすべての特権が付与されます。

デフォルト値の設定: 作成したグローバル変数は、有効範囲内で初めて参照されたときに、デフォルト値でインスタンス化されます。ステートメントでグローバル変数を参照する場合は、そのステートメントの制御フローとは無関係にグローバル変数がインスタンス化されます。

新しく作成したセッション・グローバル変数の使用: 1 つのセッションでグローバル変数を作成した場合は、作業単位がコミットされるまで、他のセッションでそのグローバル変数を使用することはできません。ただし、作業単位のコミット前でも、その新しいグローバル変数を作成したセッションでは、その変数を使用できます。

1 つのセッションでグローバル変数をインスタンス化した後に、別のセッション (DROP や GRANT など) でそのグローバル変数が変更されても、既にインスタンス化した変数は影響を受けない場合があります。

グローバル変数の作成: グローバル変数は、*SRVPGM オブジェクトとして作成します。変数名が有効なシステム名でも、同じ名前の *SRVPGM が既に存在していれば、エラーが生成されます。変数名が有効なシステム名でない場合は、システム表名の生成に関する規則を使用して固有名が生成されます。名前の生成に関する規則については、 1298 ページの『表名の生成の規則』を参照してください。

グローバル変数の定義は、関連したサービス・プログラム・オブジェクトに保管されます。*SRVPGM オブジェクトが保管された後、このシステムや別のシステム上に復元すると、カタログは定義を使用して自動的に更新されます。

グローバル変数の復元時には、次のような動作が生じます。

- 同じシステム名の *SRVPGM オブジェクトが存在する場合は、その *SRVPGM が置き換えられます。

グローバル変数と SQL ルーチンの名前が同じ場合は、グローバル変数を最初に作成することによって、命名の競合を回避できます。

変数の所有権**:** 変数の所有者 は、そのステートメントを実行するスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

変数の権限**:** SQL 名を使用する場合は、変数は、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成されます。システム名を使用する場合、変数は、スキーマの作成権限 (CRTAUT) パラメーターによって決められる *PUBLIC に対する権限を使用して作成されます。

変数の所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、その変数に対する権限が与えられます。

変数のインスタンス化に関する権限**:** グローバル変数がインスタンス化されるときには、そのグローバル変数の所有者の権限に基づいて DEFAULT 文節が評価されます。

REPLACE の規則**:** REPLACE によって変数を再作成する場合は、以下のようになります。

- 既存のコメントまたはラベルは破棄されます。
- 権限を持つユーザーは維持されます。オブジェクト所有者は変更される可能性があります。
- 現在のジャーナル監査は保持されます。

例

例 1: セッションで使用するプリンターを指定するためのグローバル変数を作成します。

```
CREATE VARIABLE MYSCHEMA.MYJOB_PRINTER VARCHAR(30)
      DEFAULT 'Default printer'
```

例 2: 従業員が所属する部署を指定するためのグローバル変数を作成します。

```
CREATE VARIABLE SCHEMA1.GV_DEPTNO INTEGER
      DEFAULT ((SELECT DEPTNO FROM HR.EMPLOYEES
                WHERE EMPUSER = SESSION_USER))
```

例 3: 現行ユーザーのセキュリティー・レベルを指定するためのグローバル変数を作成します。

```
CREATE VARIABLE SCHEMA2.GV_SECURITY_LEVEL INTEGER
      DEFAULT ( GET_SECURITY_LEVEL ( SESSION_USER))
```

CREATE VIEW

CREATE VIEW ステートメントは、現行サーバーに 1 つ以上の表またはビューに関するビューを作成します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- スキーマ内に作成する特権。詳しくは、スキーマ内で作成する必要のある権限を参照してください。
- データベース管理者権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 次のシステム権限
  - 論理ファイル作成 (CRTLF) CL コマンドに対する *USE 権限
  - データ・ディクショナリーに対する *CHANGE 権限。ただし、ビューが作成されるライブラリーが、データ・ディクショナリーを持つ SQL スキーマの場合。
- データベース管理者権限

このステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

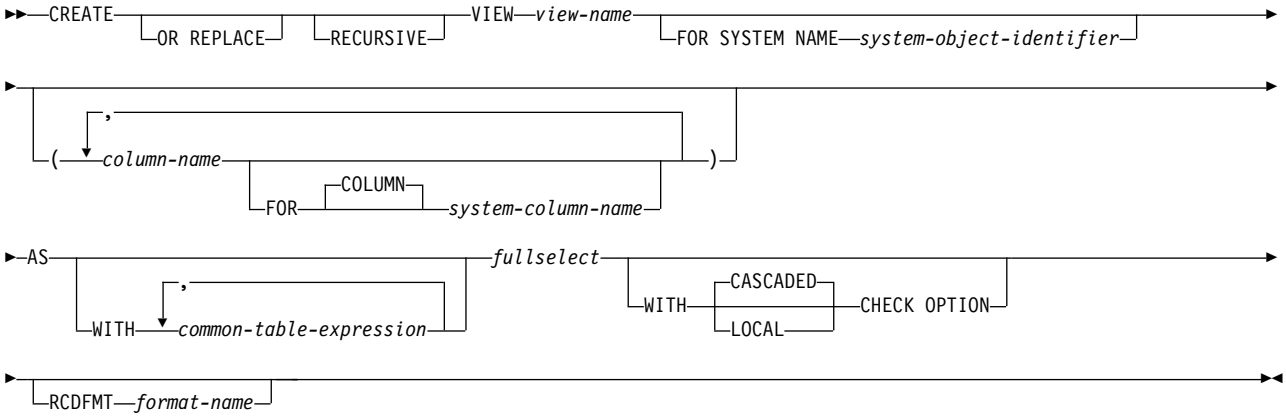
- 全選択を介して直接的に参照されたり、あるいは、全選択で参照されるビューを介して間接的に参照されるそれぞれの表とビューに対する次の特権。
  - 表やビューに対する SELECT 特権、および
  - 表またはビューが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

既存のビューに置き換えるには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

- 次のシステム権限
  - ビューに対する *OBJMGT のシステム権限
  - このビューを削除するために必要な全権限
- データベース管理者権限

SQL 特権に対応するシステム権限については、『表またはビューへの権限を検査する際の対応するシステム権限』および『特殊タイプへの権限を検査する際の対応するシステム権限』を参照してください。

構文



説明

OR REPLACE
現行サーバーにこのビューの定義が存在する場合に、その定義を置き換える、という動作を指定します。実際には、カタログで既存の定義を削除してから新しい定義に置き換える、という動作になりますが、例外として、このビューに対して与えられていた特権は影響を受けません。既存のオブジェクトは論理ファイルであってはなりません。現行サーバーにこのビューの定義が存在しなければ、このオプションは無視されます。

RECURSIVE
ビューが潜在的に反復的であることを示します。

FROM 文節内で、ビューの全選択 がビュー自身に対する参照を含んでいる場合は、そのビューは反復ビュー です。反復を使用するビューは、部品表 (BOM)、予約システム、ネットワーク計画のようなアプリケーションをサポートする場合に役立ちます。

反復ビュー に適用される以下の制約事項は、反復共通表式での制約事項と似ています。

- 列名 のリストは、全選択の結果列に既に名前が付けられていない限り、ビュー名 の後に指定しなければなりません。
- UNION ALL セット演算子を指定しなければなりません。
- 最初の和集合の最初の全選択 (初期化全選択) の FROM 文節には、ビュー自身の参照を含めてはなりません。
- 反復サイクルの一部である各全選択には、集約関数、GROUP BY 文節、または HAVING 文節を含めてはなりません。
- 各全選択の FROM 文節に、反復サイクルの一部であるビューの参照を 1 つまで組み込むことができます。
- 共通表式 で定義されている表を、共通表式 を定義する全選択の副照会で参照することはできません。
- 共通表式が右オペランドの場合、LEFT OUTER JOIN および FULL OUTER JOIN は使用できません。共通表式が左オペランドの場合、RIGHT OUTER JOIN および FULL OUTER JOIN は使用できません。

ビューの列名が反復全選択で参照される場合、結果列の属性は、結果列に関する規則を使用して決定します。詳しくは、 133 ページの『結果のデータ・タイプに関する規則』を参照してください。

照会が以下のいずれかを指定する場合、反復ビューは許可されません。

- 分散表
- 読み取りトリガーを指定する表
- 全選択で直接または間接的に参照される表は、DDS 作成の論理ファイルであってはなりません。
- 複数の物理ファイル・メンバー上に構築された論理ファイル

view-name

ビューの名前を指定します。暗黙的または明示的修飾子も含め、この名前で、現行サーバーに既に存在している別名、ファイル、索引、表、またはビューと同じ名前にすることはできません。

SQL 名が指定されている場合、ビューは、暗黙的または明示的修飾子で指定しているスキーマ内に作成されます。

システム名が指定されている場合、ビューは、修飾子で指定しているスキーマ内に作成されます。修飾されていない場合、およびデフォルトのスキーマが無い場合、ビュー名は、最初の FROM 文節 (任意の共通表式またはネストされた表式の FROM 文節を含む) 上に指定されている、最初の表と同一のスキーマ内に作成されます。表が全選択で参照されない場合、ビューは最初のユーザー定義表関数と同一のスキーマ内に作成されます。表またはユーザー定義表関数が全選択で参照されない場合は、現行ライブラリー (*CURLIB) が使用されます。

ビュー名が有効なシステム名でなく、FOR SYSTEM NAME 節が使用されていない場合、Db2 for i がシステム名を生成します。名前の生成に関する規則については、 1298 ページの『表名の生成の規則』を参照してください。

FOR SYSTEM NAME *system-object-identifier*

ビューのシステム・オブジェクト *ID* を示します。システム・オブジェクト *ID* は、現行サーバーに既に存在する表、ビュー、別名、または索引と同一であってはなりません。システム・オブジェクト *ID* は、非修飾システム ID でなければなりません。

システム・オブジェクト *ID* が指定される場合、ビュー名 は有効なシステム・オブジェクト名であってはなりません。

(column-name, ...)

このビューの列の名前を指定します。列名のリストを指定する場合は、そのリストは、全選択の結果表にある列の数と同じ数の列名で構成されている必要があります。それぞれの列名 およびシステム列名 は固有でなければならず、修飾は付けられません。列名のリストを指定しなかった場合は、ビューの列は、全選択の結果表の列名および列のシステム名を継承します。

副選択の結果表に、重複する列名、重複するシステム列名、または名前なしの列がある場合は、列名 (およびシステム列名) のリストを指定する必要があります。名前が指定されない列についての詳細は、 792 ページの『結果列の名前』を参照してください。

FOR COLUMN システム列名
列の IBM i 名を指定します。ビューの複数の列またはビューの列名に、同じ名前を使用してはなりません。

システム列名が指定されず、また列名が有効なシステム列名でない場合には、システム列名が生成されます。システム列名の生成方法に関する詳細については、1297 ページの『列名の生成の規則』を参照してください。

AS ビューを定義します。

WITH *common-table-expression*
後に続く全選択指定で使用するための共通表式を定義します。共通表式については、848 ページの『共通表式』を参照してください。

fullselect
ビューを定義します。ビューは、常に、全選択が実行された場合に結果として生じる行から構成されます。

全選択 は変数を参照してはなりませんが、グローバル変数を参照することはできます。

ビューで許可される列の最大数は 8000 です。この数は、列名の長さ、および WHERE 文節の長さによっても減少します。ビューで許可される基本表の最大数は 256 です。

全選択 の説明については、841 ページの『全選択』を参照してください。

共通表式 は、後に続く全選択 で使用するための共通表式を定義します。詳しくは、848 ページの『共通表式』を参照してください。

スキーマ QTEMP でビューを作成する場合を除き、*fullselect* で宣言済み一時表を参照することはできません。

ORDER BY 節、 FETCH 節、および OFFSET 節は、ビューの外側の全選択で指定できません。

WITH CASCADED CHECK OPTION または **WITH LOCAL CHECK OPTION**
このビューを介して挿入または更新される行は、すべてがこのビューの定義に適合しなければならないことを指定します。このビューの定義に適合しない行は、このビューを使用して検索することができない行です。

以下の場合は、CHECK OPTION は指定できません。

- ビューが読み取り専用である。
- ビューの定義に、ビューの外部選択リストのスカラー全選択以外の副照会が含まれている。
- ビューの定義に、ビューの外部選択リスト以外の非決定的、MODIFIES SQL DATA、または EXTERNAL ACTION 関数が含まれている。
- ビューの定義に、ビューの外部選択リスト以外の特殊レジスターが含まれている。
- ビューが別のビューを参照し、そのビューは INSTEAD OF トリガーを持つ。
- ビューが反復的である。

挿入を許さない更新可能ビューに関して CHECK OPTION を指定した場合は、検査オプションは更新のみに適用されます。

CHECK OPTION を指定しない場合は、ビューの定義は、そのビューを使用するいずれの挿入または更新操作のチェックにも使用されません。ただし、そのビューが CHECK OPTION を伴う他のビューに直接または間接に従属している場合には、挿入または更新の操作の過程でなおチェックが行われます。そのビューの定義は使用されないので、そのビューの定義に適合しない行がそのビューを介して挿入、または更新されることがあります。

CHECK OPTION 文節の 2 つの形式 (CASCADED と LOCAL) の違いが意味を持つのは、ビューが別のビューに依存している場合に限られます。デフォルトは CASCADED です。あるビューの定義が直接的または間接的に別のビューに基づいている場合、その基礎になっているビューのことを基本ビュー といいます。

CASCADED

ビュー V に関する WITH CASCADED CHECK OPTION は、V に直接または間接的に従属している更新可能などのビューにも継承されます。したがって、更新可能なビューが V 上に定義されている場合は、そのビューに対して WITH CHECK OPTION が指定されていなくても、V に関する検査オプションはそのビューにも適用されます。例えば、次のような更新可能なビューを想定します。

```
CREATE VIEW V1 AS SELECT COL1 FROM T1 WHERE COL1 > 10

CREATE VIEW V2 AS SELECT COL1 FROM V1 WITH CHECK OPTION

CREATE VIEW V3 AS SELECT COL1 FROM V2 WHERE COL1 < 100
```

SQL ステートメント	結果の記述
INSERT INTO V1 VALUES(5)	V1 には CHECK OPTION 文節の指定がなく、また CHECK OPTION 文節の指定を持つ他のビューに従属していないので、正しく実行されます。
INSERT INTO V2 VALUES(5)	挿入された行が、暗黙的に V2 の定義の一部である V1 の検索条件に適合していないため、エラーが生じます。
INSERT INTO V3 VALUES(5)	V3 が CHECK OPTION 文節に従属しており、挿入された行が V2 の定義に適合していないため、エラーが生じます。
INSERT INTO V3 VALUES(200)	V3 の定義には適合していなくても (V3 にはビュー CHECK OPTION 文節の指定がない)、 V2 の定義には適合しているので (ビュー CHECK OPTION 文節の指定がある)、正常に実行されます。

LOCAL

WITH LOCAL CHECK OPTION は、次の点を除いて、WITH CASCADED CHECK OPTION と同等です。すなわち、WITH LOCAL CHECK OPTION を指定して定義されたビューでは、行の更新によってその行がそのビューの定義に適合しなくなる場合でも、なおその行の更新が可能である点が異なります。これは、そのようなビューが、その定義に WITH CASCADED CHECK OPTION または WITH LOCAL CHECK OPTION のどちらの文節も指定されていないビューに直接、または間接に従属している場合にのみ起こります。

WITH LOCAL CHECK OPTION は、行の挿入または更新の時点で、以下の基本的なビューの検索条件がチェックされることを指定します。

- WITH LOCAL CHECK OPTION を指定するビュー
- WITH CASCADED CHECK OPTION を指定するビュー
- WITH CASCADED CHECK OPTION を指定するビューの基礎となるすべてのビュー

これに対して、WITH CASCADED CHECK OPTION は、行の挿入または更新の時点で、すべての基礎となるビューの検索条件がチェックされることを指定します。

CASCADED と LOCAL の相違点を例によって示します。次のような更新可能なビューについて考えます。この場合の x と y は、LOCAL か CASCADED のどちらかを表します。

```
V1 は表 T0 で定義されている。
V2 は V1 WITH x CHECK OPTION を指定して定義されている。
V3 は V2 で定義されている。
V4 は V3 WITH y CHECK OPTION を指定して定義されている。
V5 は V4 で定義されている。
```

次の表は、INSERT または UPDATE の操作中に、どのビューの検索条件がチェックされるかを示しています。

表 *85. INSERT* および *UPDATE* の過程でその検索条件がチェックされるビュー

INSERT または **UPDATE** で使用されるビュー	**x = LOCAL** **y = LOCAL**	**x = CASCADED** **y = CASCADED**	**x = LOCAL** **y = CASCADED**	**x = CASCADED** **y = LOCAL**
V1	なし	なし	なし	なし
V2	V2	V2 V1	V2	V2 V1
V3	V2	V2 V1	V2	V2 V1
V4	V4 V2	V4 V3 V2 V1	V4 V3 V2 V1	V4 V2 V1
V5	V4 V2	V4 V3 V2 V1	V4 V3 V2 V1	V4 V2 V1

RCDFMT *format-name*

ビューの IBM i レコード・フォーマット名を指定する非修飾名です。*format-name* は、システム ID です。

レコード・フォーマット名が指定されない場合、*format-name* は、ビューの *system-object-name* と同一のものになります。

注

ビューの所有権**:** SQL 名が指定されている場合、

- 作成したビューが入れられるスキーマと同じ名前のユーザー・プロファイルが存在する場合、ビューの所有者 はそのユーザー・プロファイルです。
- その他の場合は、ビューの所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

システム名を指定した場合は、ビューの所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルまたはグループ・ユーザー・プロファイルです。

ビューの権限 : SQL 名を使用する場合は、ビューは、*PUBLIC に対するシステム権限 *EXCLUDE を使用して作成されます。システム名を使用する場合、ビューは、スキーマの作成権限 (CRTAUT) パラメーターによって決められる *PUBLIC に対する権限を使用して作成されます。

ビューの所有者がグループ・プロファイルのメンバー (GRPPRF キーワード) であり、グループ権限が指定されている (GRPAUT キーワード) 場合は、そのグループ・プロファイルにも、そのビューに対する権限が与えられます。

所有者には常に、そのビューに関する WITH GRANT OPTION 付きの SELECT 特権とそのビューを削除する権限が与えられます。

また、所有者はそのビューについての INSERT、UPDATE、および DELETE 特権も入手することがあります。ビューが読み取り専用でない場合、全選択の最初の FROM 文節で識別された表やビューに対して所有者が持つ特権と同じ特権を新たなビューについても獲得することになります。そのような特権が認可できるのは、それらの元となっている特権も認可できる場合だけに限られます。

REPLACE の規則**:** REPLACE によってビューを再作成する場合は、以下のようになります。

- 既存のコメントまたはラベルは破棄されます。
- 権限を持つユーザーは維持されます。オブジェクト所有者は変更される可能性があります。
- 現在のジャーナル監査は保持されます。ただし、他のオブジェクトと異なり、ビューの REPLACE では、ZC (オブジェクト変更) ジャーナル監査項目が生成されます。
- ビューに定義された INSTEAD OF トリガーは除去されます (ビューを参照するトリガーは除去されません)。
- そのビューに依存するビューとマテリアライズ照会表は再作成されます (ただし、可能な場合に限られます)。従属するビューまたはマテリアライズ照会表を再作成できない場合は、エラーが戻されます。

削除可能ビュー**:** 削除操作の INSTEAD OF トリガーがビューに定義されている場合、または次のいずれかが当てはまる場合、ビューは削除可能 です。

- 外側の全選択が指定しているのは、カタログ表またはカタログ・ビューではない 1 つの基本表または削除可能ビューのみである。それは、ネストされた表の式、表関数、および FOR SYSTEM_TIME 文節を使用した表であってはならない。
- 外側の全選択に VALUES 文節が含まれていない。
- 外側の全選択に、GROUP BY 文節または HAVING 文節が含まれていない。
- 外側の全選択の選択リストに集約関数が含まれていない。
- 外側の全選択に UNION 演算子、UNION ALL 演算子、EXCEPT 演算子、または INTERSECT 演算子が含まれていない。
- 外側の全選択に DISTINCT 文節が含まれていない。

更新可能ビュー: 更新操作の INSTEAD OF トリガーがビューに定義されている場合、または次のいずれかが当てはまる場合、ビューは更新可能 です。

- 削除可能なビューである (削除操作の INSTEAD OF トリガーとは無関係)。
- ビューの少なくとも 1 つが更新可能である。

ビューに更新操作用 INSTEAD OF トリガーが定義されている場合、または副選択の対応する結果列が表の列、または別のビューの更新可能な列からのみ取り出される場合、ビューの列は 更新可能 です (すなわち、結果の列は、演算子、スカラー関数、定数、またはそれ自体がそのような式から取り出された列を含む式から取り出される列であってはなりません)。

挿入可能ビュー: ビューに INSTEAD OF トリガーが定義されている場合、またはビューの少なくとも 1 つの列が更新可能である場合、ビューは挿入可能 です。

読み取り専用のビュー: 削除可能でないビューは読み取り専用 です。

読み取り専用ビューに対して INSERT、UPDATE、または DELETE ステートメントを実行することはできません。

テンポラル・サポート用の特殊レジスター: 特殊レジスター CURRENT TEMPORAL SYSTEM_TIME の値は、ビューを定義する照会式には、そのビューが定義されている限り影響を及ぼしません。 SQL ステートメントでビューを使用する場合、SYSTIME オプションの値が YES に設定されていれば、その SQL ステートメントを処理しているセッションの CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターの値が、ビューに適用されます。

非修飾表名: CREATE VIEW ステートメントが非修飾表名を参照する場合、実際に参照する表を決定するために、次の規則が適用されます。

- 非修飾名が全選択 で指定された 1 つ以上の共通表式表 *ID* に対応する場合、その名前は有効範囲が最も内側の共通表式を示します。
- それ以外の場合、名前は永続表、一時表、またはデフォルトのスキーマに存在するビューを識別します。

暗黙的な隠し列に関する考慮事項: 暗黙的な隠し列として定義された基本表の列が、全選択の結果表に含まれることがあります。この状態が発生するのは、暗黙的な非表示列がビュー定義の全選択で明示的に参照される場合です。ただし、ビューの対応する列は、暗黙的な隠し列としての属性を継承しません。 ビューの列を非表示として定義することはできません。

照合順序: ビューは、CREATE VIEW ステートメントの実行時に有効な照合順序に従って作成されます。ビューの照合順序は、そのビューの全選択における SBCS データおよび混合データに関連するすべての比較に適用されます。照会にビューが含まれる場合は、そのビューの全選択から中間結果表が作成されます。照会を実行するときに有効な照合順序が、その照会で指定される選択すべてに適用されます。

ビューの属性 : ビューは、キーのない論理ファイルとして作成されます。ビューが作成される場合、ファイル待ち時間とレコード待ち時間の属性は、論理ファイル作成 (CRTLF) コマンドの WAITFILE キーワードと WAITRCD キーワード上に指定されたデフォルト値に設定されます。

日時の結果列に使用される日時形式は ISO です。

分散表を介して作成されるビューは、その表が配布されるすべてのシステム上で作成されます。ビューが複数の分散表に作成され、その表が同一のノード・グループを使用して配布されない場合は、そのビューは、CREATE VIEW ステートメントを実行するシステムにのみ作成されます。分散表の詳細については、「Db2 UDB for iSeries マルチ・システム」トピック集を参照してください。

ID 列および行変更タイム・スタンプ列: ビューの列が ID 列または行変更タイム・スタンプ列であると見なされるのは、ビュー定義の全選択の中の対応する列のエレメントが、表の ID 列または行変更タイム・スタンプ列の名前であるか、基本表の ID 列または行変更タイム・スタンプ列の名前に直接的または間接的にマップするビューの列の名前である場合です。その他の場合は、ビューの列は ID プロパティーまたは行変更タイム・スタンプ・プロパティーを取得しません。例えば、次のようになります。

- ビュー定義の選択リストに、ID 列の名前の複数のインスタンスが含まれている (つまり同じ列を複数回選択している) 場合。
- ビュー定義に結合が含まれている場合。
- ビュー定義の中の列のいずれかに、ID 列を参照する式が含まれている場合。
- ビュー定義に UNION または INTERSECT が含まれている場合。

例

例 1: C 表 PROJECT をもとに、MA_PROJ という名前のビューを作成します。この表には PROJNO (プロジェクト番号) が「MA」という文字で始まっている行だけを入れます。

```
CREATE VIEW MA_PROJ
  AS SELECT * FROM PROJECT
       WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

例 2: 例 1 と同じようにビューを作成します。ただし、このビューでは、PROJNO (プロジェクト番号)、PROJNAME (プロジェクト名)、および RESPEMP (プロジェクトに関与している従業員) の各列だけを選択します。

```
CREATE VIEW MA_PROJ
  AS SELECT PROJNO, PROJNAME, RESPEMP FROM PROJECT
       WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

例 3: 例 2 と同様のビューを作成します。ただし、ビューの中でプロジェクトの責任者の列を IN_CHARGE と呼びます。

```
CREATE VIEW MA_PROJ (PROJNO, PROJNAME, IN_CHARGE)
  AS SELECT PROJNO, PROJNAME, RESPEMP FROM PROJECT
       WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

注: 列名を 1 つだけ変更する場合でも、MA_PROJ の後の括弧内に、ビューを構成する 3 つの列の名前をすべて指定しなければなりません。

例 4: PRJ_LEADER という名前のビューを作成します。このビューには、PROJECT 表の最初の 4 つの列 (PROJNO, PROJNAME, DEPTNO, RESPEMP) と、そのプロジェクトの責任者 (RESPEMP) の名字 (LASTNAME) を合わせて入れます。名前は、表 EMPLOYEE 内の EMPNO と表 PROJECT 内の RESEMP を突き合わせることによって、表 EMPLOYEE から取得しています。

```
CREATE VIEW PRJ_LEADER
  AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME
       FROM PROJECT, EMPLOYEE
       WHERE RESPEMP = EMPNO
```

例 *5:* 例 4 と同じようにビューを作成します。ただし、このビューには、PROJNO、PROJNAME、 DEPTNO、RESEMP、および LASTNAME の各列に加えて、責任者の給与総額 (SALARY + BONUS +COMM) を入れます。さらに、このビューでは、PRSTAFF (平均人員数) が 1 より大きいプロジェクトだけを選択しています。

```
CREATE VIEW PRJ_LEADER (PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME, TOTAL_PAY)
  AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME, SALARY+BONUS+COMM
       FROM PROJECT, EMPLOYEE
       WHERE RESPEMP = EMPNO AND PRSTAFF > 1
```

例 *6:* 共通表式と同様の結果を戻す反復ビューを作成します (852 ページの『例 1: 単一レベルの部品展開』 を参照)。

```
CREATE RECURSIVE VIEW RPL (PART, SUBPART, QUANTITY) AS
        SELECT ROOT.PART, ROOT.SUBPART, ROOT.QUANTITY
           FROM PARTLIST ROOT
           WHERE ROOT.PART = '01'
       UNION ALL
        SELECT CHILD.PART, CHILD.SUBPART, CHILD.QUANTITY
           FROM RPL PARENT, PARTLIST CHILD
           WHERE PARENT.SUBPART = CHILD.PART

SELECT DISTINCT *
FROMRPL
ORDER BY PART, SUBPART, QUANTITY
```

DEALLOCATE DESCRIPTOR

DEALLOCATE DESCRIPTOR ステートメントは、SQL 記述子を割り振り解除します。

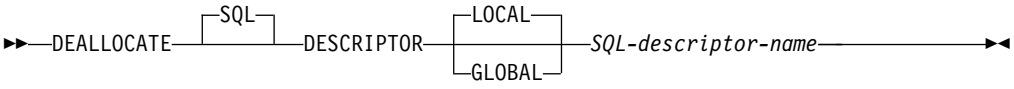
呼び出し

このステートメントは、アプリケーション・プログラム、SQL 関数、 SQL プロシージャー、またはトリガー内にのみ組み込むことができます。これを対話式に発行することはできません。これは実行可能ステートメントですが、動的に準備することはできません。REXX で指定してはなりません。

権限

権限は不要です。

構文



説明

LOCAL
記述子の名前の有効範囲はプログラム呼び出しのローカルであることを指定します。このローカル有効範囲で既知の記述子が割り振り解除されます。

GLOBAL
記述子の名前の有効範囲は SQL セッション全体であることを指定します。同じデータベース接続を使用して実行するどのプログラムにも既知の記述子が割り振り解除されます。

SQL-descriptor-name
割り振り解除する記述子の名前を指定します。名前は、指定の有効範囲を持つ既存の記述子を識別するものでなければなりません。

注

記述子の永続性**:** ローカル記述子とグローバル記述子も暗黙的に割り振り解除されます。詳細については、『記述子持続性』を参照してください。

例

'NEWDA' という記述子を割り振り解除します。

```
EXEC SQL DEALLOCATE DESCRIPTOR 'NEWDA'
```

DECLARE CURSOR

DECLARE CURSOR ステートメントは、カーソルを定義します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、実行可能ステートメントではありません。 Java では指定できません。

権限

このステートメントを使用するための権限は不要です。ただし、カーソルに関して OPEN または FETCH を使用するには、ステートメントの権限 ID によって保持される特権に、少なくとも次の 1 つが含まれていなければなりません。

- 該当のカーソルの SELECT ステートメントで識別される各表またはビューに対して、
  - 表やビューに対する SELECT 特権、および
  - 表またはビューが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

カーソルの SELECT ステートメントは、次のいずれかです。

- ステートメント名 によって識別される準備済み選択ステートメント。
- 指定した選択ステートメント。

statement-name を指定した場合:

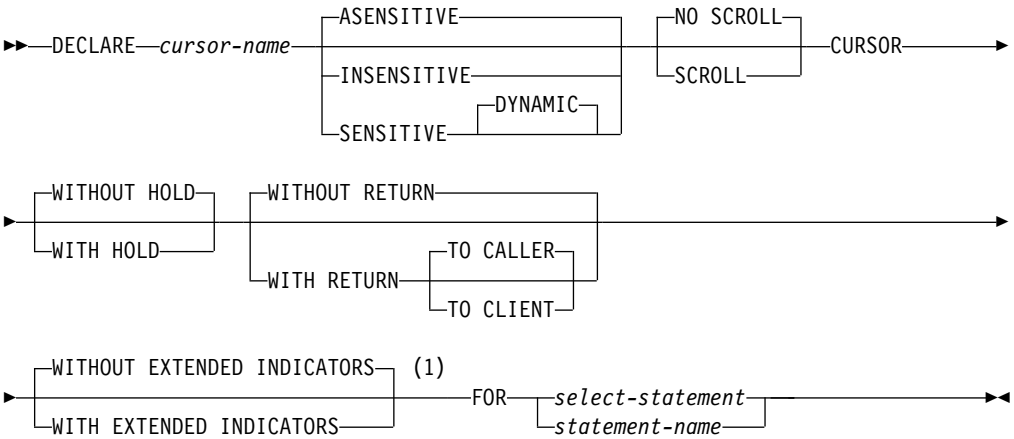
- ステートメントの権限 ID は、プログラムが作成されたときに CRTSQLxxx コマンドに USRPRF(*OWNER) および DYNUSRPRF(*OWNER) が指定された場合を除いて、実行時の権限 ID です。詳しくは、 78 ページの『権限 ID と権限名』を参照してください。
- CRTSQLxxx コマンドで DLYPRP(*YES) が指定されていない場合には、選択ステートメント を準備するときに権限検査が行われます。
- DLYPRP (*YES) パラメーターを使用してコンパイルされているプログラムについては、カーソルをオープンするときに権限検査が行われます。

選択ステートメント を指定した場合は、

- SQL 命名を指定した USRPRF(*OWNER) または USRPRF(*NAMING) が、CRTSQLxxx コマンドで指定された場合は、ステートメントの権限 ID は、その SQL プログラムまたはパッケージの所有者です。
- システム命名を指定した USRPRF(*USER) または USRPRF(*NAMING) が、CRTSQLxxx コマンドで指定された場合は、ステートメントの権限 ID は、実行時権限 ID です。
- REXX では、ステートメントの権限 ID は、実行時の権限 ID です。
- カーソルがオープンされるときには、権限検査が実行されます。

SQL 特権に対応するシステム権限については、『表またはビューへの権限を検査する際の対応するシステム権限』を参照してください。

構文



注:

1 HOLD、RETURN、および EXTENDED INDICATORS 文節は、どのような順序でも指定できます。

説明

cursor-name
カーソルの名前を指定します。ソース・プログラムで宣言されている、他のカーソルと同じ名前を指定してはなりません。

ASENSITIVE、SENSITIVE、または INSENSITIVE
カーソルが変更に対して反応を決めない、反応する、または反応しないことを指定します。 *statement-name* を指定した場合、デフォルトはこのステートメントの対応する準備属性になります。それ以外の場合は、ASENSITIVE がデフォルトです。

ASENSITIVE
カーソルは選択ステートメント の最適化の内容に応じて SENSITIVE または INSENSITIVE として動作できることを指定します。

SENSITIVE
カーソルがオープンになった後にデータベースに加えられた変更が結果表で可視になることを指定します。カーソルには、カーソルがオープンになった後に結果表の基礎となる行に加えられた更新または削除に対して、幾つかの感度レベルがあります。カーソルは、同じカーソルを使用した定位置の更新または削除に対して常にセンシティブです。さらに、カーソルはこのカーソルの外側でなされた変更に対する感度を持つことができます。データベース・マネージャーが変更をカーソルに対して可視にすることができない場合は、エラーが戻されます。カーソルが暗黙的に読み取り専用になった場合、データベース・マネージャーは変更をカーソルに対して可視にすることができません。 (カーソルの結果表を参照してください。) SENSITIVE を指定する場合は、SELECT ステートメントにデータ変更表参照 を入れられません。

INSENSITIVE
これを指定するとカーソルは、そのオープン後は、この活動化グループや他

の活動化グループで実行する挿入、更新、または削除を検知しなくなります。 INSENSITIVE を指定すると、カーソルは読み取り専用になり、カーソルのオープン時に一時結果が作成されます。さらに、SELECT ステートメントに UPDATE 文節を含めることができなくなり、また、アプリケーションでは、データ (ALWCPYDTA(*OPTIMIZE) または ALWCPYDTA(*YES)) のコピーを許可する必要が生じます。

NO SCROLL または SCROLL

カーソルがスクロール可能かどうかを指定します。*statement-name* を指定した場合、デフォルトはこのステートメントの対応する準備属性になります。それ以外の場合は、NO SCROLL がデフォルトです。

NO SCROLL

カーソルがスクロール可能でないことを指定します。

SCROLL

カーソルがスクロール可能であることを指定します。カーソルは、他の活動化グループによって行われる挿入、更新、および削除をただちに検知する場合とそうでない場合があります。

WITHOUT HOLD または WITH HOLD

コミット操作の結果として、カーソルがクローズされるのを防止するかどうかを指定します。 *statement-name* を指定した場合、デフォルトはこのステートメントの対応する準備属性になります。それ以外の場合は、WITHOUT HOLD がデフォルトです。

WITHOUT HOLD

コミット操作の結果としてカーソルをクローズすることを回避しません。

WITH HOLD

コミット操作の結果として、カーソルがクローズされるのを防止します。WITH HOLD 文節を使用して宣言されたカーソルがコミット時点で暗黙にクローズするのは、そのカーソルに関連する接続がコミット操作中に終了する場合だけです。

WITH HOLD の指定がある場合、コミット操作はその時点の作業単位における変更をすべてコミットし、そのカーソル一を維持する上で必要なロック以外のすべてのロックを解放します。その後、位置指定 UPDATE または DELETE ステートメントを実行できるようにするために FETCH ステートメントが必要になります。

カーソルはすべて、CONNECT (タイプ 1) またはロールバック操作によって暗黙にクローズされます。ある接続に関連するカーソルはすべて、その接続の切り離しによって暗黙にクローズされます。カーソルは、WITH HOLD が指定されていない場合、または、そのカーソルに関連した接続が解除保留状態にある場合にも、コミット操作によって暗黙にクローズされます。

カーソルがコミット操作の前にクローズされた場合、その効果は、そのカーソルが WITH HOLD オプションを指定せずに宣言されたのと同一です。

WITHOUT RETURN または WITH RETURN

カーソルの結果表をプロシージャーの結果セットとして使用することを指定しま

す。*statement-name* を指定した場合、デフォルトはこのステートメントの対応する準備属性になります。それ以外の場合は、WITHOUT RETURN がデフォルトです。

WITHOUT RETURN
カーソルの結果表をプロシージャーの結果セットとして使用しないことを指定します。

WITH RETURN
カーソルの結果表をプロシージャーの結果セットとして使用することを指定します。プロシージャーのソース・コードに DECLARE CURSOR ステートメントが含まれていなければ、この文節は無視されます。

SQL プロシージャーの場合に結果セットが返されるのは、プロシージャー定義の DYNAMIC RESULT SETS 文節で結果セットの最大数としてゼロ以外の値を指定した場合に限られます。

- WITH RETURN 文節を使用して定義したカーソルがプロシージャーの終了時にオープンしたままになっていれば、それらのカーソルによってプロシージャーの結果セットが定義されます。プロシージャーの作成時に CLOSQLCSR(*ENDACTGRP) を指定していない限り、他のすべてのオープン・カーソルは、プロシージャーの終了時にクローズします。
- WITH RETURN 文節または WITHOUT RETURN 文節を使用して定義したカーソルがストアード・プロシージャーに存在しない場合は、ストアード・プロシージャーの終了時にオープンしているカーソルが結果セット・カーソルになる可能性があります。
- プロシージャーの結果セットがどのように決定されるかにかかわる他の注意点については、 1208 ページの『CREATE PROCEDURE (SQL)』にある DYNAMIC RESULT SETS 文節の説明を参照してください。

外部プロシージャーの場合は、以下のようになります。

- プロシージャーの作成時に CLOSQLCSR(*ENDACTGRP) を指定していない限り、WITH RETURN 文節を使用して定義したカーソル (または SET RESULT SETS ステートメントで結果セット・カーソルとして指定したカーソル) がプロシージャーの終了時にオープンしていれば、それらのカーソルによってプロシージャーの結果セットが定義される可能性があります。他のすべてのオープン・カーソルは、オープンしたままの状態で残ります。
- WITH RETURN 文節または WITHOUT RETURN 文節を使用して定義したカーソルがストアード・プロシージャーに存在せず、SET RESULT SETS ステートメントで結果セット・カーソルとして指定したカーソルも存在しない場合は、ストアード・プロシージャーの終了時にオープンしているカーソルが結果セット・カーソルになる可能性があります。
- プロシージャーの結果セットがどのように決定されるかにかかわる他の注意点については、 1189 ページの『CREATE PROCEDURE (外部)』にある DYNAMIC RESULT SETS 文節の説明を参照してください。

スクロール可能ではないカーソルの場合、結果セットには、現行カーソル位置から結果表の最後までのすべての行が含まれます。スクロール可能なカーソルの場合、結果セットには、結果表のすべての行が含まれます。

TO CALLER
カーソルがプロシージャーの呼び出し側に結果セットを戻せることを指定します。例えば、呼び出し側がクライアント・アプリケーションである場合、結果セットはそのクライアント・アプリケーションに戻されます。

TO CLIENT
カーソルがクライアント・アプリケーションに結果セットを戻せることを指定します。このカーソルは、中間にネストされたプロシージャーからは見えません。関数またはトリガーが直接または間接的にプロシージャーを呼び出すと、結果セットはクライアントに返されず、プロシージャーの終了後にカーソルがクローズされます。

複数のモジュールを持つ ILE プログラムから結果セットが戻される場合は、TO CLIENT が必要な場合があります。

WITHOUT EXTENDED INDICATORS または WITH EXTENDED INDICATORS
拡張標識を使用可能にするかどうかを指定します。*statement-name* を指定した場合、デフォルトはこのステートメントの対応する準備属性になります。指定していない場合、デフォルトは収容側のプログラムまたはサービス・プログラムで指定された属性になります。

WITHOUT EXTENDED INDICATORS
拡張標識変数を使用不可にすること、および暗黙または明示的な UPDATE 文節または *select-statement* の中では更新可能な列のみを使用できることを指定します。

WITH EXTENDED INDICATORS
拡張標識変数を使用可能にすること、および選択ステートメント の暗黙または明示的な UPDATE 文節内では非更新可能な列を使用できることを指定します。

選択ステートメント
カーソルの SELECT ステートメントを指定します。詳しくは、 847 ページの『選択ステートメント』を参照してください。

select-statement にはパラメーター・マーカーを含めることはできません (REXX の場合を除く) が、変数への参照は含めることができます。 REXX 以外のホスト言語では、ソース・プログラム内でホスト変数の宣言の前に DECLARE CURSOR ステートメントを配置する必要があります。 REXX の場合は、変数の代わりにパラメーター・マーカーを使用してこのステートメントを作成する必要があります。

statement-name
カーソルの SELECT ステートメントは、そのカーソルのオープンの時点で ステートメント名 によって識別される準備済み選択ステートメント です。このステートメント名 は、ソース・プログラムの他の DECLARE CURSOR ステートメントで指定されているステートメント名 と同じであってはなりません。準備済みステートメントについての詳細は、 1603 ページの『PREPARE』を参照してください。

注

DECLARE CURSOR の配置**:** DECLARE CURSOR ステートメントは、 C および PL/I を除いて、該当するカーソルを明示的に参照するどのステートメントよりも前に置かなければなりません。

カーソルの結果表**:** オープン状態にあるカーソルは、結果表 と、その結果表の行に対する相対的な位置を指定します。指示される表は、該当するカーソルの SELECT ステートメントで指定されている結果表です。

以下の条件がすべて満たされている場合は、カーソルは削除可能 です。

- 外部の全選択で指定されているのが、1 つの基本表または削除可能ビューだけであり、その基本表または削除可能ビューがカタログ表またはカタログ・ビューではなく、ネストされた表の式に含まれているわけでもない。
- 外側の全選択に VALUES 文節が含まれていない。
- 外側の全選択に、GROUP BY 文節または HAVING 文節が含まれていない。
- 外側の全選択の選択リストに集約関数が含まれていない。
- 外側の全選択に UNION 演算子、UNION ALL 演算子、EXCEPT 演算子、または INTERSECT 演算子が含まれていない。
- 外側の全選択の *SELECT* 文節 に DISTINCT 文節が含まれていない。
- 外部全選択に FOR SYSTEM_TIME 期間指定が含まれない
- 外側の全選択に FROM 文節のデータ変更表参照 が含まれていない。
- *select-statement* に ORDER BY 文節と UPDATE 文節が含まれておらず、SENSITIVE が DECLARE CURSOR ステートメントに指定されていない。
- 選択ステートメント に FOR READ ONLY 文節が含まれていない。
- 外側の全選択の結果に一時表が使用されていない。
- *select-statement* に SCROLL キーワードが含まれていないか、または SENSITIVE キーワードか UPDATE 文節も指定されている。
- UPDATE 文節が指定されていない場合に、選択リストに DATALINK 列が含まれていない。

以下のすべての条件が満たされている場合は、カーソルに関連した外側の全選択の選択リスト内の結果列は更新可能 です。

- カーソルが削除可能。
- 結果列が、表の 1 つの列またはビューの更新可能な列からのみ派生したものである。すなわち、結果の列は、演算子、スカラー関数、定数、またはそれ自体がそのような式から取り出された列を含む式から取り出される列であってはなりません。

カーソルが読み取り専用 であるのは、削除可能でない場合です。

UPDATE 文節を省略する場合、副選択の SELECT 文節の中の列のうち、更新可能なものだけを変更できます。

UPDATE が列名のリストを使用せずに指定された場合、このカーソルを識別する後続の位置指定 UPDATE ステートメントの割り当て文節のターゲットとして指定できる列のリストは、次のように決定されます。

- WITH EXTENDED INDICATORS を指定した場合は、全選択の最初の FROM 文節で識別される表またはビューのすべての列。
- それ以外の場合は、全選択の最初の FROM 文節で識別される表またはビューの更新可能な列のみ。

列名のリストを使用して UPDATE が指定された場合、列名のリストに指定された列のみが、このカーソルを識別する後続の位置指定 UPDATE ステートメントの割り当て文節のターゲットとして指定できます。

カーソルの有効範囲**:** *cursor-name* の有効範囲は、これが定義されたソース・プログラム、つまりプリコンパイラーに実行依頼されたプログラムです。したがって、カーソルを参照できるステートメントは、そのカーソル宣言と一緒にプリコンパイルされたステートメントだけです。例えば、別個にコンパイルされた他のプログラムから呼び出されるプログラムでは、その呼び出し側プログラムでオープンされているカーソルを使用することはできません。

また、カーソル名の有効範囲は、カーソルが収められているプログラムの実行場所であるスレッドに限定されます。例えば、同一ジョブ内の 2 つの別個のスレッドで同じプログラムが実行しているとした場合、2 番目のスレッドは、最初のスレッドでオープンされたカーソルを使用することはできません。

CRTSQLxxx コマンドに CLOSQLCSR(*ENDJOB)、CLOSQLCSR(*ENDSQL)、または CLOSQLCSR(*ENDACTGRP) が指定されている場合を除いて、カーソルを参照できるのは、プログラム・スタック内の該当するプログラムの同じインスタンスに限られます。

- CLOSQLCSR(*ENDJOB) が指定されている場合は、プログラム・スタックにある該当するプログラムのどのインスタンスでもカーソルを参照することができます。
- CLOSQLCSR(*ENDSQL) が指定されている場合は、プログラム・スタック上の最後の SQL プログラムが終了するまでは、そのプログラム・スタックにある該当するプログラムのどのインスタンスでもカーソルを参照することができます。
- CLOSQLCSR(*ENDACTGRP) が指定されている場合は、活動化グループが終了するまでは、その活動化グループ内のモジュールのすべてのインスタンスでカーソルを参照することができます。

カーソルの有効範囲は、そのカーソルが宣言されているプログラムですが、そのプログラムから作成された各パッケージは、そのカーソルの別個のインスタンスを含み、実行時に複数のカーソルが存在することがあります。例えば、CONNECT (タイプ 2) ステートメントを使用して、次の順序でロケーション X とロケーション Y に接続するプログラムを想定します。

```
EXEC SQL DECLARE C CURSOR FOR...
EXEC SQL CONNECT TO X;
EXEC SQL OPEN C;
EXEC SQL FETCH C INTO...
EXEC SQL CONNECT TO Y;
EXEC SQL OPEN C;
EXEC SQL FETCH C INTO...
```

2 番目の OPEN C ステートメントは、カーソル C の別個のインスタンスを参照しているので、エラーにはなりません。

SELECT ステートメントは、カーソルがオープンされる時点で評価されます。同一のカーソルをいったんオープンし、クローズした後で、再びオープンした場合には、結果が異なる可能性があります。カーソルの SELECT ステートメントに CURRENT DATE、CURRENT TIME、または CURRENT TIMESTAMP が含まれる場合、これらの特殊レジスターへのすべての参照で、FETCH ごとにそれぞれ同じ日時値が取り出されます。この値は、カーソルがオープンされたときに決まります。同一の SELECT ステートメントを使用する、複数のカーソルを同時にオープンすることができます。これらのカーソルはそれぞれ、独立したアクティビティーと見なされます。

シーケンス式の使用**:** カーソルでの NEXT VALUE および PREVIOUS VALUE 式の使用方法については、カーソルを使ったシーケンス式の使用を参照してください。

データのブロック化**:** データの処理効率を高めるために、データベース・マネージャーは読み取り専用カーソル用のデータをブロック化することができます。カーソルを位置指定 UPDATE または DELETE ステートメントの中で使用することを予定していない場合は、カーソルを FOR READ ONLY として宣言してください。

REXX での使用法**:** REXX プロシージャー内の DECLARE CURSOR ステートメントで変数を使用する場合、その DECLARE CURSOR は PREPARE および EXECUTE のオブジェクトでなければなりません。

一時的な結果**:** 特定の *SELECT* ステートメント を一時結果表としてインプリメントすることができます。

- 一時的結果表は、下記の場合に作成されます。
  - INSENSITIVE が指定された場合。
  - ORDER BY 文節と GROUP BY 文節の指定する列が異なる場合、または列の指定の順序が異なる場合。
  - ORDER BY 文節と GROUP BY 文節に、ユーザー定義の関数、あるいは、スカラー関数である DLVALUE、DLURLPATH、DLURLPATHONLY、DLURLSERVER、DLURLSCHEME、または、属性が FILE LINK CONTROL と READ PERMISSION DB のデータ・リンクの場合は DLURLCOMPLETE のいずれかが含まれている場合。
  - UNION 文節、EXCEPT 文節、INTERSECT 文節、または DISTINCT 文節が指定されている場合。
  - ORDER BY 文節または GROUP BY 文節で指定されている列が、すべて同じ表のものでない場合。
  - JOINDFT データ記述仕様 (DDS) キーワードによって定義された論理ファイルが、別のファイルに結合されている場合。
  - 複数のデータベース・ファイルのメンバーに基づく論理ファイルが指定されている場合。
  - DECLARE CURSOR の選択ステートメントが GROUP BY 文節を使用しているときに、FETCH ステートメントで CURRENT または RELATIVE スクロール・オプションが指定されている場合。
  - FETCH FIRST n ROWS ONLY 文節が指定されていない場合。
- 次のような副照会が組み込まれている照会。

- 最外部の照会が内部の副選択に相関値を提供しない場合。
- 最外部の照会で、IN、= ANY、= SOME、または <> ALL 副照会を参照しない場合。

カーソルの感応性 **:** DYNAMIC SCROLL カーソルの場合は、ALWCPYDTA プリコンパイル・オプションは無視されます。挿入、更新、および削除に対する感応性を維持しなければならない場合には、照会の実行に一時的結果が必要でない限り、データの一時コピーは作成されません。

OFFSET 文節または FETCH FIRST 文節、あるいはその両方を持つスクロール可能なセンシティブ・カーソルは、以下の場合、指定されたオフセットの前、または指定された行数の後に、行を戻す可能性があります。

- カーソルを通じてデータをフェッチ中にカーソルの基礎表に対して更新操作、削除操作、または挿入操作が行われる。かつ
- カーソルに対するフェッチ操作が、カーソルによって以前にフェッチされた同じ行を戻そうとする。

代替構文**:** 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- DYNAMIC SCROLL は SENSITIVE DYNAMIC SCROLL の同義語です。

例

例 *1:* 表 DEPARTMENT からデータを検索するための照会のカーソルとして、C1 を宣言します。 DECLARE CURSOR ステートメントにこの照会自体が含まれています。

```
EXEC SQL DECLARE C1 CURSOR FOR
            SELECT DEPTNO, DEPTNAME, MGRNO
              FROM DEPARTMENT
              WHERE ADMRDEPT = 'A00';
```

例 *2:* 表 DEPARTMENT からデータを検索するための照会のカーソルとして、C1 を宣言します。検索した更新データによるデータの更新は後で行われ、データは照会が実行されるときはロックされるものとします。 DECLARE CURSOR ステートメントにこの照会自体が含まれています。

```
EXEC SQL DECLARE C1 CURSOR FOR
            SELECT DEPTNO, DEPTNAME, MGRNO
              FROM DEPARTMENT
              WHERE ADMRDEPT = 'A00'
              FOR READ ONLY WITH RS USE AND KEEP EXCLUSIVE LOCKS;
```

例 *3:* STMT2 という名前のステートメント用のカーソルとして、C2 を宣言します。

```
EXEC SQL DECLARE C2 CURSOR FOR STMT2;
```

例 *4:* 表 EMPLOYEE の位置指定更新に使用する照会用のカーソルとして、C3 を宣言します。更新が完了するたびに、カーソルをクローズせずに更新がコミットされるようにします。

```
  EXEC SQL DECLARE C3 CURSOR WITH HOLD FOR
             SELECT *
               FROM EMPLOYEE
               FOR UPDATE OF WORKDEPT, PHONENO, JOB, EDLEVEL, SALARY;
```

更新する列を明示的に指定する代わりに、列を指定せずに UPDATE 文節を使用することもできます。この方法で、表の更新可能な列をすべて更新することができます。このカーソルは更新可能なので、このカーソルを使用して表から行を削除することもできます。

例 *5:* C プログラムにおいて、カーソル C1 を使用して、指定したプロジェクト (PROJNO) に関する値を、表 EMPPROJACT の最初の 4 列から一度に 1 行ずつ取り出して、それらの値を EMP (CHAR(6))、 PRJ (CHAR(6))、ACT (SMALLINT)、および TIM (DECIMAL(5,2)) というホスト変数に入れています。検索するプロジェクトの値は、ホスト変数 SEARCH_PRJ (CHAR(6)) から入手します。動的に 選択ステートメント を準備して、プログラムの実行時に検索するプロジェクトを指定できるようにします。

```
void main ()
  {
    EXEC SQL  BEGIN DECLARE SECTION;
    char           EMP[7];
    char           PRJ[7];
    char           SEARCH_PRJ[7];
    short          ACT;
    double         TIM;
    char           SELECT_STMT[201];
    EXEC SQL  END DECLARE SECTION;
    EXEC SQL  INCLUDE SQLCA;

    strcpy(SELECT_STMT, "SELECT EMPNO, PROJNO, ACTNO, EMPTIME \
                   FROM EMPPROJACT \
                   WHERE PROJNO = ?");
    .
    .
    .
    EXEC SQL PREPARE SELECT_PRJ FROM :SELECT_STMT;

    EXEC SQL DECLARE C1 CURSOR FOR SELECT_PRJ;

/* Obtain the value for SEARCH_PRJ from the user.     */
    .
    .
    .
    EXEC SQL OPEN C1 USING :SEARCH_PRJ;

    EXEC SQL  FETCH C1 INTO :EMP, :PRJ, :ACT, :TIM;

    if (strcmp(SQLSTATE, "02000", 5) )
      {
        data_not_found();
      }
    else
      {
        while (strcmp(SQLSTATE, "00", 2) || strcmp(SQLSTATE, "01", 2) )
          {
            EXEC SQL  FETCH C1 INTO :EMP, :PRJ, :ACT, :TIM;
          }
      }

    EXEC SQL  CLOSE C1;
```

```
    .
    .
    .
}
```

例 *6:* DECLARE CURSOR ステートメントによって、カーソル名 C1 を SELECT の結果に関連付けます。 C1 は、更新可能、スクロール可能なカーソルです。

```
EXEC SQL DECLARE C1 SENSITIVE SCROLL CURSOR FOR
    SELECT DEPTNO, DEPTNAME, MGRNO
    FROM TDEPT
    WHERE ADMRDEPT = 'A00';
```

例 *7:* 4 つの列から値を取り出し、逐次化可能 (RR) 分離レベルを使用して、それらの値を変数に割り当てるために、カーソルを宣言します。

```
DECLARE CURSOR1 CURSOR FOR
  SELECT COL1, COL2, COL3, COL4
  FROM TBLNAME WHERE COL1 = :varname
  WITH RR
```

例 *8:* EMPLOYEE 表が、生成された列 WEEKLYPAY (年間の給与に基づいて週ごとの支払いを計算する) を追加するように調整されていると想定します。カーソルを宣言して、挿入される行からシステムが生成した列の値を取り出します。

```
DECLARE C2 CURSOR FOR
  SELECT E.WEEKLYPAY
  FROM FINAL TABLE
    (INSERT INTO EMPLOYEE
        (EMPNO, FIRSTNME, MIDINIT, LASTNAME, EDLEVEL, SALARY)
        VALUES('000420', 'Peter', 'U', 'Bender', 16, 31842)) AS E;
```

DECLARE GLOBAL TEMPORARY TABLE

DECLARE GLOBAL TEMPORARY TABLE ステートメントは、現行アプリケーション・プロセス用の宣言済み一時表を定義します。宣言済み一時表記述は、システム・カタログには含まれません。この記述は持続性のあるものではなく、複数のアプリケーション・プロセス間で共用することはできません。同名の宣言済み一時表を定義するアプリケーション・プロセスは、それぞれ固有の一時表の記述を使用します。アプリケーション・プロセスが終了すると、一時表は除去されます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

LIKE または AS 選択ステートメント 文節を指定する場合は、このステートメントの権限 ID が保持する特権には、その LIKE 文節または *as-result-table* 文節で指定するすべての表またはビューに対して、少なくとも次の 1 つが含まれていなければなりません。

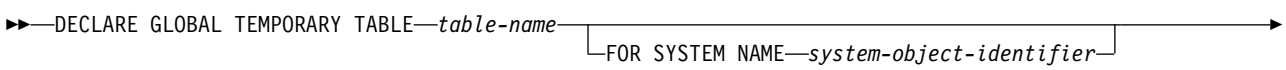
- 表またはビューに対する SELECT 特権
- 表またはビューの所有権
- データベース管理者権限

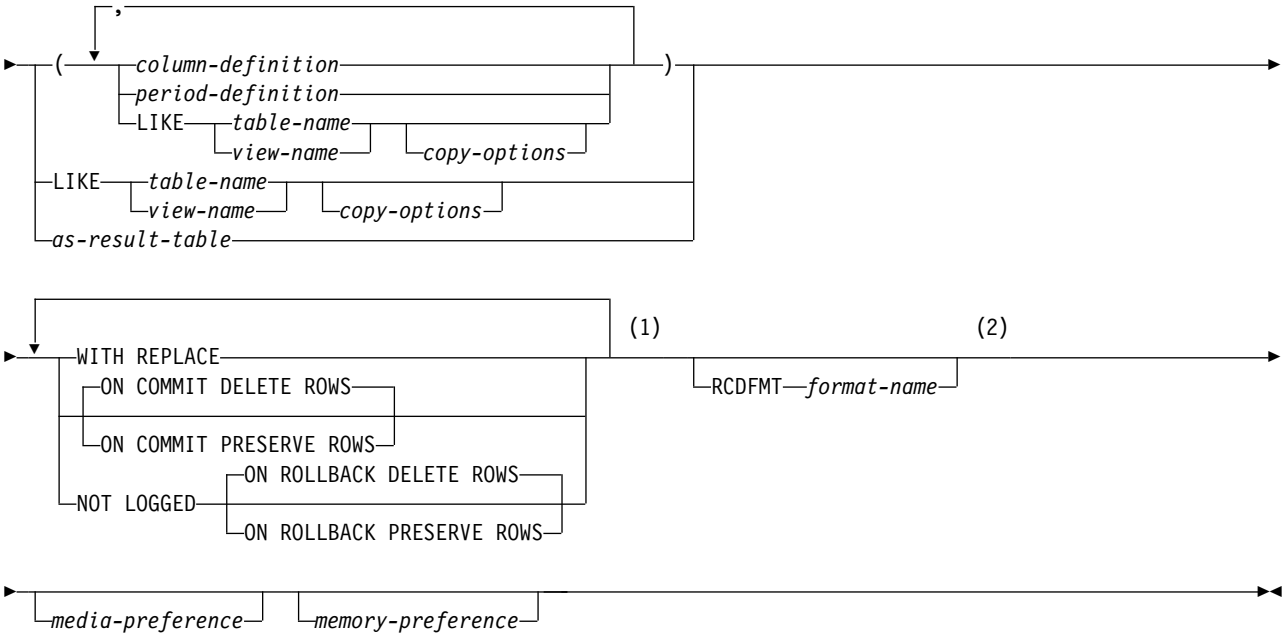
特殊タイプを参照する場合は、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別された、それぞれの特殊タイプごとに、
  - その特殊タイプに対する USAGE 特権、および
  - 特殊タイプが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

SQL 特権に対応するシステム権限については、『表またはビューへの権限を検査する際の対応するシステム権限』および 『特殊タイプへの権限を検査する際の対応するシステム権限』を参照してください。

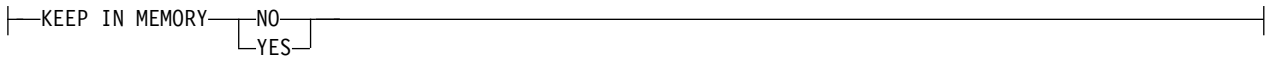
構文



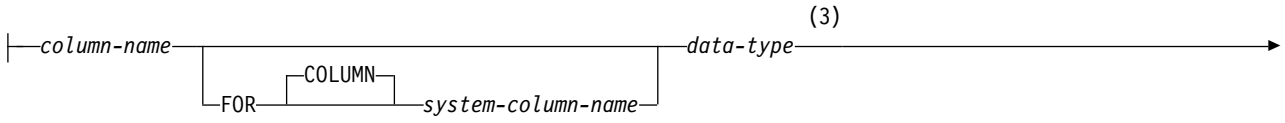


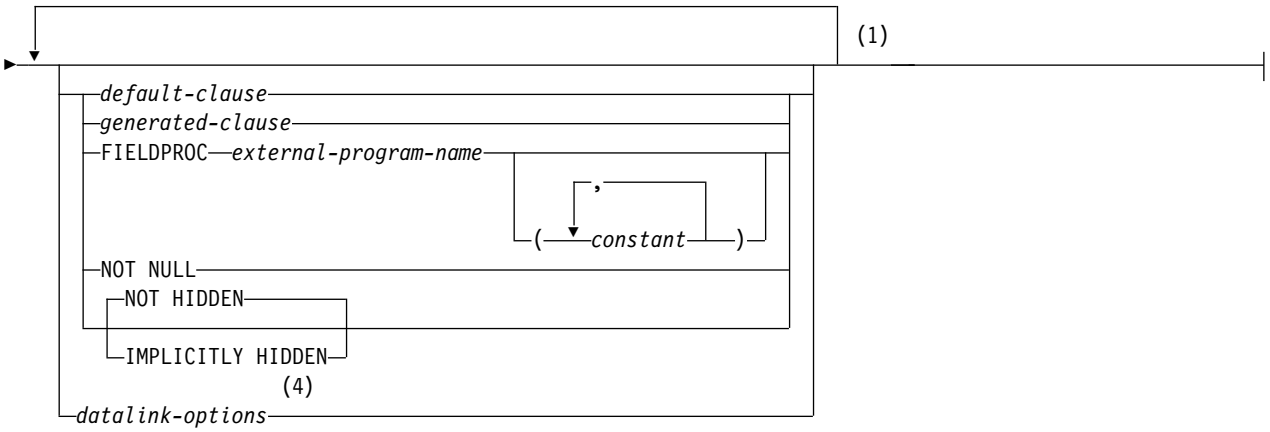
media-preference:

memory-preference:

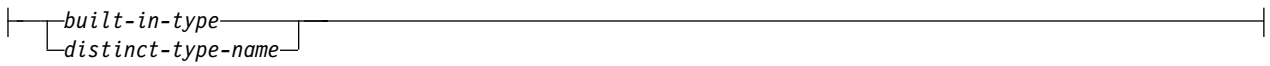


column-definition:





data-type:



注**:**

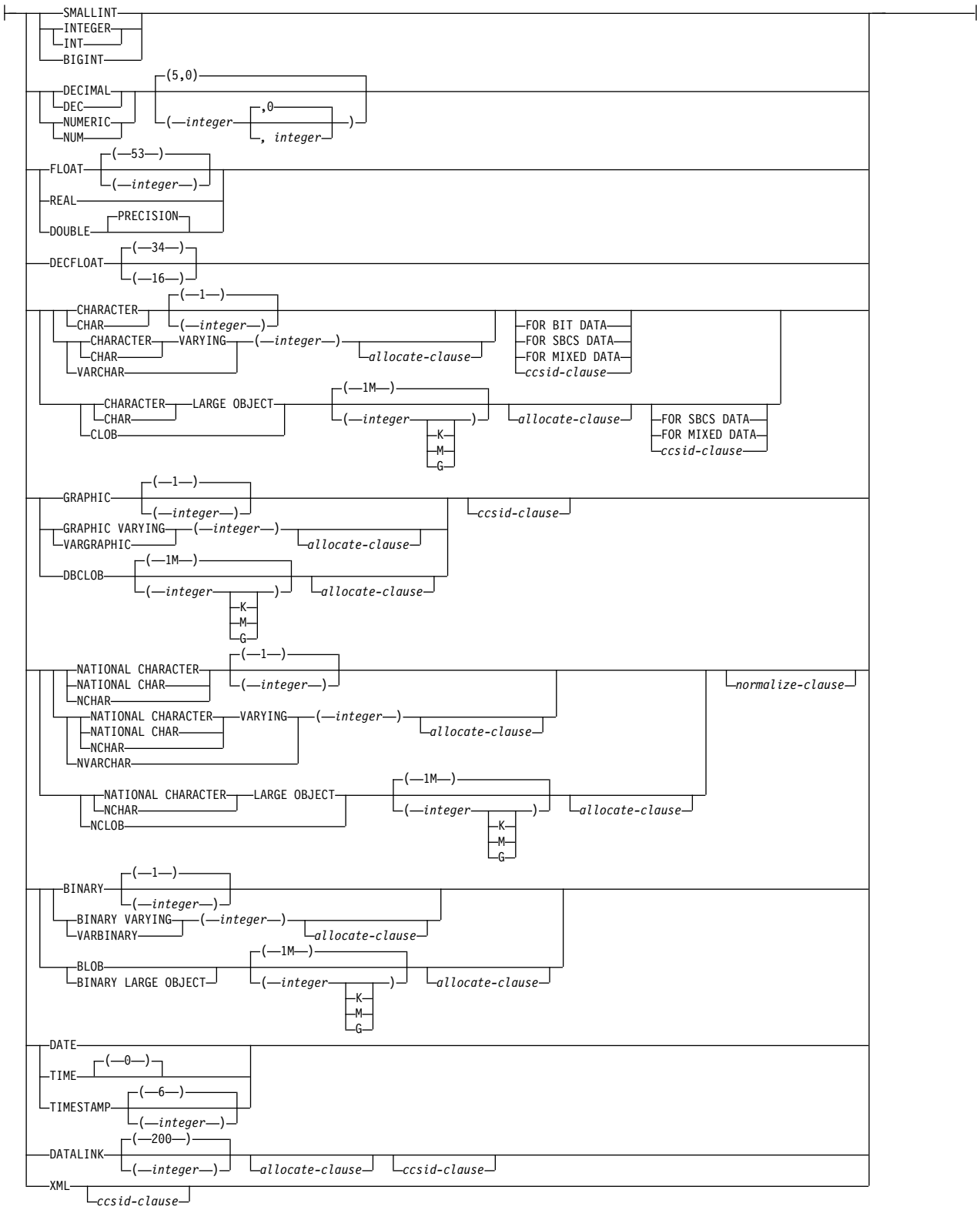
1 同じ文節を複数回指定することはできません。

2 オプション文節は、どのような順序で指定しても構いません。

3 *data-type* は、行変更タイム・スタンプ列、行開始列、行終了列、およびトランザクション開始 ID 列にはオプションです。

4 データ・リンク・オプションは、DATALINK、および DATALINK をソースとする特殊タイプに対してのみ指定することができます。

built-in-type:

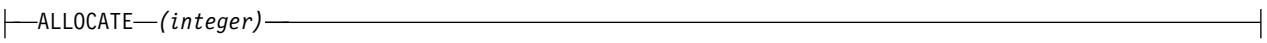


ccsid-clause:

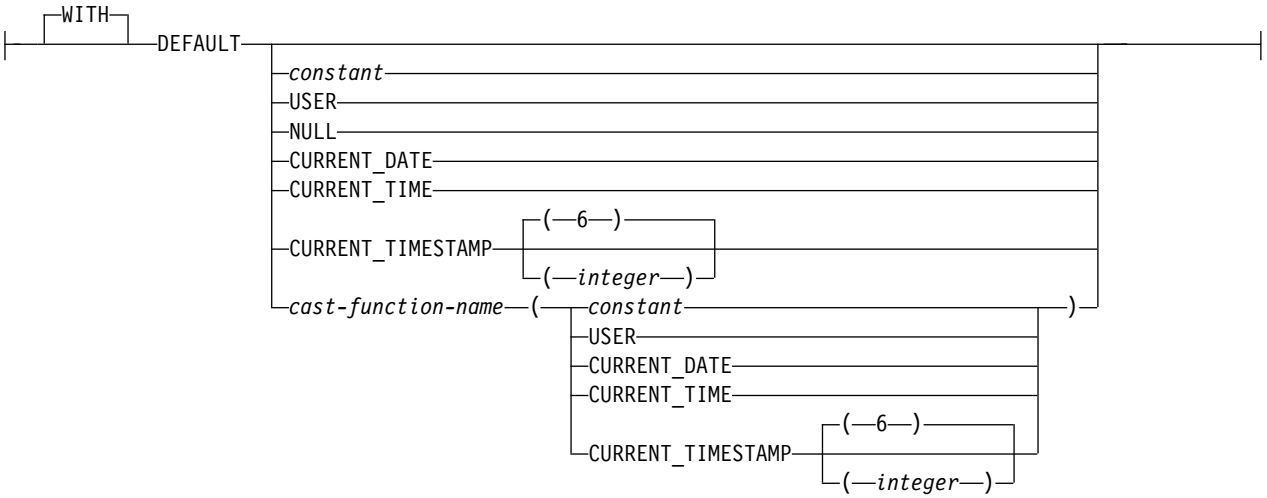
DECLARE GLOBAL TEMPORARY TABLE

normalize-clause:

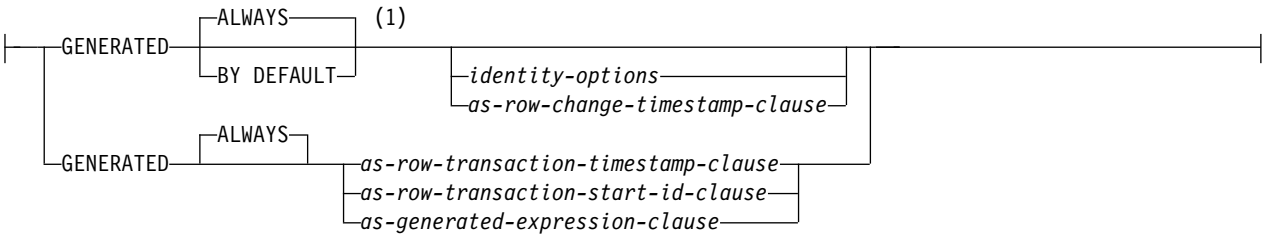
allocate-clause:



default-clause:



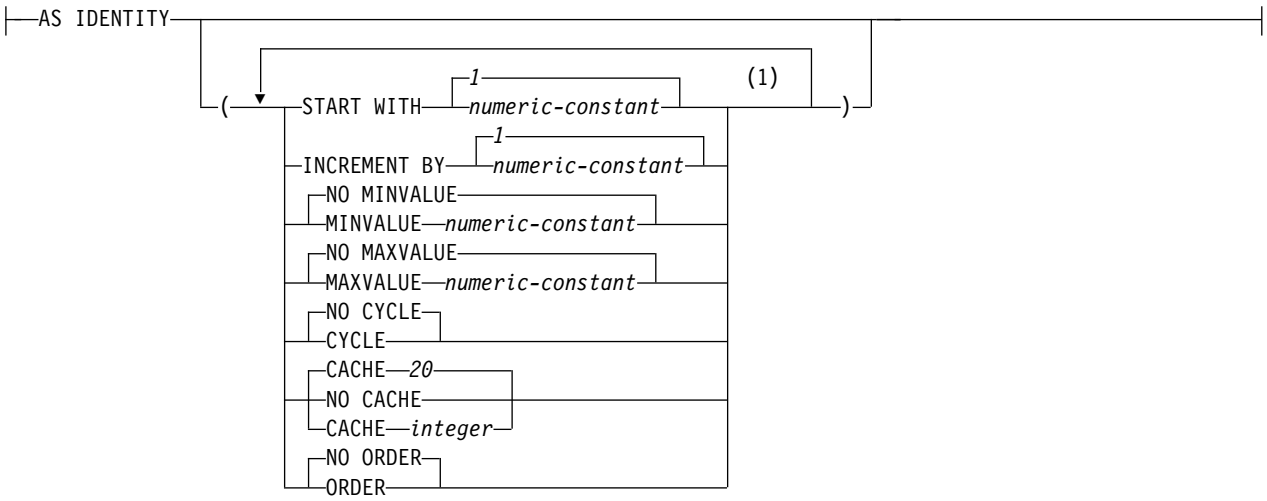
generated-clause:



注:

1 GENERATED を指定できるのは、列のデータ・タイプが ROWID (または ROWID データ・タイプに基づく特殊タイプ) であるか、列が ID 列であるか、または列が行変更タイム・スタンプである場合のみです。

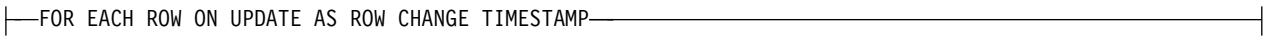
identity-options:



注:

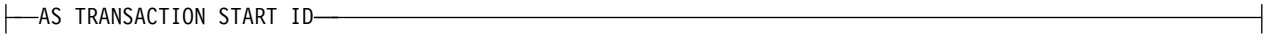
1 同じ文節を複数回指定することはできません。

as-row-change-timestamp-clause:



as-row-transaction-timestamp-clause:

as-row-transaction-start-id-clause:



as-generated-expression-clause:

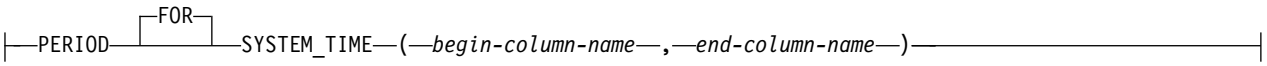
non-deterministic-expression:

special-register:

DECLARE GLOBAL TEMPORARY TABLE

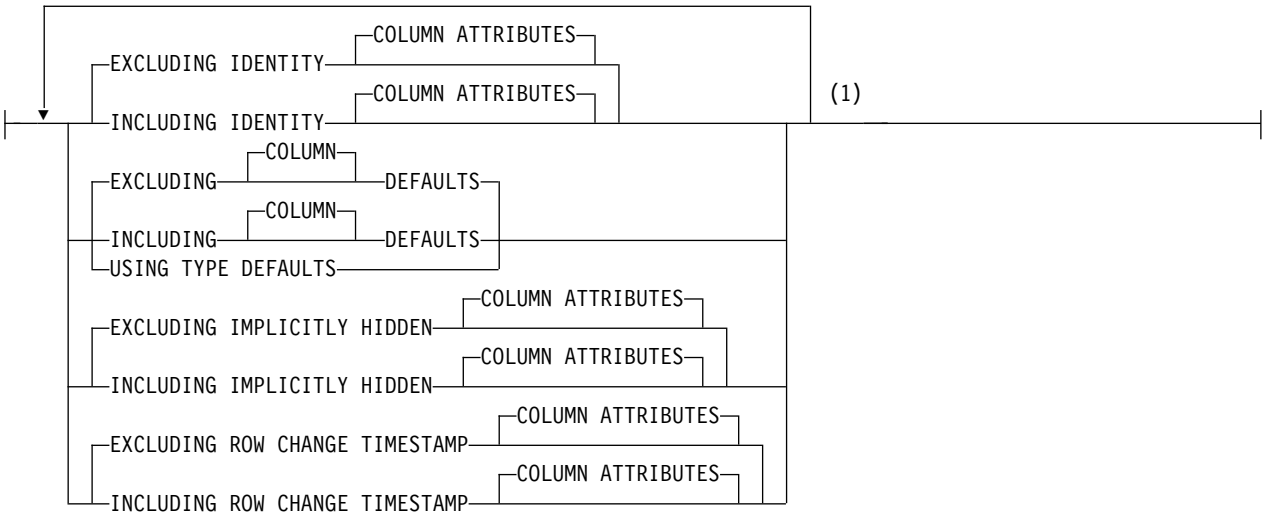
built-in-global-variable:

period-definition:

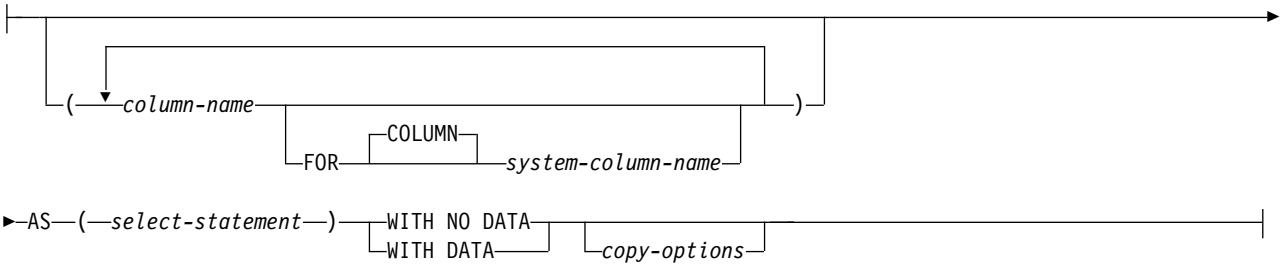


datalink-options:

copy-options:



as-result-table:



注:

1 文節は、どのような順序で指定しても構いません。

説明

table-name
一時表の名前を指定します。 修飾子を明示指定する場合は、SESSION でなければなりません。そうでない場合、エラーが戻されます。指定しなかった場合は、修飾子は暗黙的に SESSION に設定されます。宣言した一時表、または宣言した一時表に従属した索引またはビューが同じ名前で既に存在する場合、エラーが戻されます。

同名およびスキーマ名 SESSION を持つ表、ビュー、索引、または別名が既に存在している場合は、以下のアクションがとられます。

- 宣言した一時表は、SESSION.*table-name* の名前で定義されます。宣言済み一時表の解決には永続ライブラリーは含まれないので、エラーは起こりません。
- SESSION.*table-name* に対する参照は、SESSION.*table-name* の名前を持つ永続的な表、ビュー、索引、または別名ではなく、この宣言済み一時表に解決されます。

この表は QTEMP ライブラリー内に作成されます。

FOR SYSTEM NAME *system-object-identifier*
表のシステム・オブジェクト *ID* を示します。システム・オブジェクト *ID* は、現行サーバーに既に存在する表、ビュー、別名、または索引と同一であってはなりません。システム・オブジェクト *ID* は、非修飾システム ID でなければなりません。

システム・オブジェクト *ID* が指定される場合、表名 は有効なシステム・オブジェクト名であってはなりません。

column-definition

列の属性を定義します。少なくとも 1 つ以上で、8000 を超えない列の定義がなければなりません。

列の行バッファー・バイト・カウントの合計は、32766 (VARCHAR 列、VARGRAPHIC 列、または VARBINARY 列が指定されている場合は 32740) 以下でなければなりません。さらに、LOB 列または XML 列を指定してある場合は、す

べての列の行データ・バイト・カウントの合計が 3.5 GB を超えていてはなりません。データ・タイプごとの列のバイト・カウントについては、 1295 ページの『最大行サイズ 』を参照してください。

column-name
表を構成する列の名前を指定します。列名 は修飾できません。表の複数の列や表のシステム列名に同じ名前を使用することもできません。

FOR COLUMN *system-column-name*
列の IBM i 名を指定します。表の複数の列やシステム列名に対して、同じ名前を使用してはなりません。

システム列名が指定されず、また列名が有効なシステム列名でない場合には、システム列名が生成されます。システム列名の生成方法に関する詳細については、 1297 ページの『列名の生成の規則』を参照してください。

data-type
列のデータ・タイプを指定します。

built-in-type
組み込みデータ・タイプを指定します。組み込みタイプ の説明については、 1238 ページの『CREATE TABLE』を参照してください。

宣言済み一時表については、ROWID 列、または FILE LINK CONTROL を伴う DATALINK 列は指定できません。

distinct-type-name
列のデータ・タイプが、特殊タイプ (ユーザー定義のデータ・タイプ) であることを指定します。この列の長さ、精度、および位取りは、それぞれ、特殊タイプのソースとなっているタイプの長さ、精度、および位取りと同じになります。スキーマ名なしの特殊タイプを指定すると、その特殊タイプ名は、SQL パス上のスキーマを検索することで解決されます。

DEFAULT
列のデフォルト値を指定します。この文節は、1 つの列定義 の中で複数回指定することはできません。次のタイプの列に対しては、Db2 がデフォルト値を生成するため、DEFAULT を指定できません。

- ID 列 (AS IDENTITY として定義される列)
- 行変更タイム・スタンプ列
- 行開始列
- 行終了列
- トランザクション開始 ID 列
- 生成式列

XML 列のデフォルトは、NULL です。ただし、NOT NULL を指定した場合、デフォルトはありません。

DEFAULT キーワードの後に値が指定されていない場合は、次のようになります。

- 列がNULL 可能の場合、デフォルト値は NULL 値になります。
- 列がNULL 可能でない場合、デフォルト値は列のデータ・タイプによって決まります。

データ・タイプ	デフォルト値
数値	0
固定長文字またはグラフィック・ストリング	ブランク
固定長バイナリー・ストリング	16 進ゼロ
可変長ストリング	0 のストリング長
日付	INSERT の時点の当日の日付
時刻	INSERT の時点の当日の時刻
タイム・スタンプ	INSERT の時点の当日のタイム・スタンプ
データ・リンク	DLVALUE('','URL','') に対応する値
特殊タイプ	特殊タイプの対応するソース・タイプのデフォルト値

NOT NULL および DEFAULT を列の定義 から省いた場合、DEFAULT NULL の暗黙の指定が取られます。

constant
その列のデフォルト値としての定数を指定します。これは、 113 ページの『割り当ておよび比較』で説明している割り当て規則に従って、その列に割り当てることができる値を表す定数にする必要があります。浮動小数点定数または 10 進浮動小数点定数は、SMALLINT、INTEGER、DECIMAL、または NUMERIC 列に使用してはなりません。10 進定数には、小数点より右方に、その列に指定された位取りより多くの桁を含めてはなりません。

USER
INSERT または UPDATE の時点での USER 特殊レジスターの値を、その列のデフォルト値として指定します。この列のデータ・タイプは、USER 特殊レジスターの長さ属性と同じかそれより大きい長さ属性を持つ CHAR または VARCHAR でなければなりません。

NULL
その列のデフォルト値として NULL を指定します。 NOT NULL を指定する場合は、同じ列の定義 内で DEFAULT NULL を指定してはなりません。

CURRENT_DATE
現在の日付を列のデフォルト値として指定します。 CURRENT_DATE を指定する場合は、列のデータ・タイプは DATE または DATE に基づく特殊タイプでなければなりません。

CURRENT_TIME
現在の時刻を列のデフォルト値として指定します。 CURRENT_TIME を指定する場合は、列のデータ・タイプは TIME または TIME に基づく特殊タイプでなければなりません。

CURRENT_TIMESTAMP または **CURRENT_TIMESTAMP(***integer***)**
現在のタイム・スタンプを列のデフォルト値として指定します。 CURRENT_TIMESTAMP を指定する場合は、列のデータ・タイプは TIMESTAMP または TIMESTAMP に基づく特殊タイプでなければなりません。デフォルトとして使用される CURRENT_TIMESTAMP 特殊レジス

ターのタイム・スタンプ精度は、この特殊レジスターに指定された精度に関係なく、常に列のタイム・スタンプ精度と一致します。

cast-function-name
この形式のデフォルト値は、特殊タイプやデータ・タイプ、BINARY、VARBINARY、BLOB、CLOB、DBCLOB、DATE、TIME または TIMESTAMP として定義された列でのみ使用することができます。次の表は、これらのキャスト関数 の許可されている使用法を示します。

データ・タイプ	キャスト関数名
BINARY、VARBINARY、BLOB、CLOB、または DBCLOB に基づく特殊タイプ N	BINARY、VARBINARY、BLOB、CLOB、または DBCLOB *
DATE、TIME、または TIMESTAMP に基づく特殊タイプ N	N (N の作成時に生成されたユーザー定義のキャスト関数) ** あるいは DATE、TIME、または TIMESTAMP *
他のデータ・タイプに基づく特殊タイプ	N (N の作成時に生成されたユーザー定義のキャスト関数) **
BINARY、VARBINARY、BLOB、CLOB、または DBCLOB	BINARY、VARBINARY、BLOB、CLOB、または DBCLOB *
DATE、TIME、または TIMESTAMP	DATE、TIME、または TIMESTAMP *

注:

* 関数には、QSYS2 の暗黙的または明示的スキーマ名のデータ・タイプ (または、特殊タイプのソース・タイプ) の名前と一致する名前を指定する必要があります。

** 関数には、列の特殊タイプの名前と一致する名前を指定する必要があります。スキーマ名で修飾する場合は、特殊タイプのスキーマ名と同じ名前を指定する必要があります。修飾しない場合は、関数の解析から得られるスキーマ名は、特殊タイプのスキーマ名と同じ名前にする必要があります。

constant
定数を引数として指定します。この定数は、特殊タイプのソース・タイプの定数、あるいは、特殊タイプでない場合は、データ・タイプの定数の規則に準拠する必要があります。BINARY、VARBINARY、 BLOB、CLOB、DBCLOB、DATE、TIME、および TIMESTAMP 関数の場合は、この定数をストリング定数にする必要があります。

USER
INSERT または UPDATE の時点での USER 特殊レジスターの値をその列のデフォルト値として指定します。この列の特殊タイプのソース・タイプのデータ・タイプは、USER 特殊レジスターの長さ属性と同じかそれより大きい長さ属性を持つ CHAR または VARCHAR でなければなりません。

CURRENT_DATE
現在の日付を列のデフォルト値として指定します。 CURRENT_DATE を指定する場合、列の特殊タイプのソース・タイプのデータ・タイプは、DATE にする必要があります。

CURRENT_TIME
現在の時刻を列のデフォルト値として指定します。 CURRENT_TIME を指定する場合、列の特殊タイプのソース・タイプのデータ・タイプは、TIME にする必要があります。

CURRENT_TIMESTAMP または **CURRENT_TIMESTAMP(***integer***)**
現在のタイム・スタンプを列のデフォルト値として指定します。 CURRENT_TIMESTAMP を指定する場合、列の特殊タイプのソース・タイプのデータ・タイプは、TIMESTAMP にする必要があります。デフォルトとして使用される CURRENT_TIMESTAMP 特殊レジスターのタイム・スタンプ精度は、この特殊レジスターに指定された精度に関係なく、常に列のタイム・スタンプ精度と一致します。

指定した値が無効である場合、エラーが戻されます。

GENERATED
データベース・マネージャーが列の値を生成することを示します。列が次のいずれかのタイプの列であると見なされる場合には、GENERATED を指定できます。

- ID 列
- 行変更タイム・スタンプ列

列を次のいずれかのタイプの列に統合する場合には、GENERATED を指定する必要があります。

- 行開始列
- 行終了列
- トランザクション開始 ID 列
- 生成式列

また、列のデータ・タイプが ROWID (または ROWID に基づく特殊タイプ) である場合も、GENERATED を指定できます。その他の場合は、GENERATED を指定してはなりません。GENERATED は、列定義内に *default-clause* とともに指定してはなりません。in

ALWAYS
行の挿入時または更新時にデフォルト値を生成しなければならない場合に、データベース・マネージャーが常に列の値を生成することを指定します。ALWAYS は推奨値です。

BY DEFAULT
行の挿入時または更新時にデフォルト値を生成しなければならない場合に、明示的な値が指定されていない限り、データベース・マネージャーが列の値を生成することを指定します。

ID 列または行変更タイム・スタンプ列の場合は、データベース・マネージャーは指定された値を挿入または更新しますが、その ID 列または行変更タイム・スタンプ列がユニーク制約を持っているか、その ID 列または行変更タイム・スタンプ列を単独で指定するユニーク索引を持っている場合を除き、その値がその列の固有な値であるかどうかの検査は行いません。

AS IDENTITY
列が表の ID 列であることを指定します。表には 1 つの ID 列しか入りませ

ん。AS IDENTITY を指定できるのは、列のデータ・タイプが、厳密に位取りがゼロの数値タイプ (SMALLINT、INTEGER、BIGINT、 DECIMAL、または位取りがゼロの NUMERIC、またはこれらのデータ・タイプに基づく特殊タイプ) である場合だけです。 DECIMAL または NUMERIC データ・タイプが指定された場合、精度は 31 以下でなければなりません。

ID 列は暗黙的に NOT NULL になります。ID 列に DEFAULT 文節を指定することはできません。識別属性の説明については、 1238 ページの『CREATE TABLE』内の AS IDENTITY 文節を参照してください。

FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP

列がタイム・スタンプであり、値はデータベース・マネージャーによって生成されることを指定します。データベース・マネージャーは、行が挿入されるたびに各行の列に値を生成し、また列が更新されるたびに各行に値を生成します。行変更タイム・スタンプ列に生成される値は、その行の挿入または更新の時刻に対応するタイム・スタンプです。1 つの SQL ステートメントを指定して複数の行が挿入される場合、行変更タイム・スタンプ列の値は、各行が挿入された時点を反映するために行ごとに異なる可能性があります。生成された値が固有である保証はありません。

表には、行変更タイム・スタンプ列を 1 つだけ指定できます。データ・タイプを指定する場合は、精度 6 の TIMESTAMP であるか、または、精度 6 の TIMESTAMP に基づく特殊タイプでなければなりません。行変更タイム・スタンプ列の場合、DEFAULT 節を指定することはできず、NOT NULL でなければなりません。

AS ROW BEGIN

列がタイム・スタンプ・データを含み、値がデータベース・マネージャーによって生成されることを指定します。データベース・マネージャーは、行が挿入されるたびに各行の列に値を生成し、また列が更新されるたびに各行に値を生成します。生成される値は、最新のトランザクションに関連付けられている開始時刻に対応するタイム・スタンプです。単一 SQL ステートメントで複数の行が挿入される場合、トランザクション開始タイム・スタンプ列の値は各行で同じになります。

システム期間テンポラル表の場合、行開始列の値は、トランザクション全体にわたり固有になるようにデータベース・マネージャーによって生成されます。関連した履歴表に挿入される行の終了タイム・スタンプ値が開始タイム・スタンプ値より大きくなるように、タイム・スタンプ値が調整される可能性があります。これは、競合するトランザクションがシステム期間テンポラル表の同じ行を更新しているときに行われる場合があります。このタイム・スタンプ値の調整を行うには、SYSTIME_PERIOD_ADJ QAQQINI オプションを *ADJUST に設定する必要があります。単一の SQL トランザクション内で複数の行が挿入または更新され、調整が必要ではない場合、行開始列の値はすべての行において同じになり、別のトランザクションでその列のために生成された値とは異なる固有の値になります。行開始列は、SYSTEM_TIME 期間の開始列として必要です。

1 つの表は 1 つの行開始列しか持てません。*data-type* が指定されない場合、列は TIMESTAMP(12) として定義されます。*data-type* を指定する場合は、TIMESTAMP(12) でなければなりません。この列には DEFAULT 節を指定できないため、NOT NULL として定義する必要があります。行開始列は更新できません。

AS ROW END
これを指定すると、行が挿入されるとき、または行内のいずれかの列が更新されるときには常に、データベース・マネージャーによって列のデータ・タイプの値が割り当てられます。割り当てられる値は TIMESTAMP '9999-12-30-00.00.00.000000000000'です。

行終了列は、SYSTEM_TIME 期間の 2 番目の列として必要です。

表には 1 つの行終了列のみを含めることができます。*data-type* が指定されない場合、列は TIMESTAMP(12) として定義されます。*data-type* を指定する場合は、TIMESTAMP(12) でなければなりません。この列には DEFAULT 節を指定できないため、NOT NULL として定義する必要があります。行終了列は更新できません。

AS TRANSACTION START ID
これを指定すると、行が表に挿入されるとき、または行のいずれかの列が更新されるときには常に、データベース・マネージャーによって値が割り当てられます。データベース・マネージャーは、トランザクションごとに固有のタイム・スタンプ値、または NULL 値を割り当てます。トランザクション開始 ID 列が NULL 可能で、値を調整する必要がない行開始列が表にある場合には、その列に NULL 値が割り当てられます。それ以外の場合、この値は、次のいずれかの場合に時刻機構を読み取ることによって生成されます。(1) トランザクションの中で、表に含まれる行開始列またはトランザクション開始 ID 列に値を割り当てる必要があるようなデータ変更ステートメントを最初に実行するとき。(2) システム期間テンポラル表に含まれる行を削除するとき。単一の SQL トランザクション内で複数の行が挿入または更新される場合、トランザクション開始 ID 列の値はすべての行において同じになり、別のトランザクションでその列のために生成された値とは異なる固有の値になります。

システム期間テンポラル表にはトランザクション開始 ID 列が必要です。

1 つの表は 1 つのトランザクション開始 ID 列しか持てません。*data-type* が指定されない場合、列は TIMESTAMP(12) として定義されます。*data-type* を指定する場合は、TIMESTAMP(12) でなければなりません。トランザクション開始 ID 列には DEFAULT 節を指定できません。トランザクション開始 ID 列は更新できません。

DATA CHANGE OPERATION
挿入された各行、列が更新されたすべての行、および履歴表が ON DELETE ADD EXTRA ROW で定義されている場合にシステム期間テンポラル表から削除されたすべての行に、データベース・マネージャーが値を生成することを指定します。列には、以下のいずれかの値が含まれます。

I 挿入操作

U 更新操作

D 削除操作

data-type が指定されない場合、列は CHAR(1) として定義されます。*data-type* を指定する場合は、CHAR(1) でなければなりません。この列には、DEFAULT 文節もフィールド・プロシージャーも指定できません。

special-register
挿入された各行、列が更新されたすべての行、および履歴表が ON DELETE

ADD EXTRA ROW で定義されている場合にシステム期間テンポラル表から削除されたすべての行に、データベース・マネージャーによって特殊レジスターの値が割り当てられることを指定します。データ変更ステートメントの時点での特殊レジスターの値が使用されます。単一の SQL ステートメントを使用して複数の行が変更される場合、列の値は、すべての行で同じになります。

data-type は、以下の表に従って定義する必要があります。

特殊レジスター	列のデータ・タイプ
CURRENT CLIENT_ACCTNG	VARCHAR(255)
CURRENT CLIENT_APPLNAME	VARCHAR(255)
CURRENT CLIENT_PROGRAMID	VARCHAR(255)
CURRENT CLIENT_USERID	VARCHAR(255)
CURRENT CLIENT_WRKSTNNAME	VARCHAR(255)
CURRENT SERVER	VARCHAR(18)
SESSION_USER	VARCHAR(128)
USER	VARCHAR(18)

この列には、DEFAULT 文節もフィールド・プロシージャーも指定できません。

built-in-global-variable

挿入された各行、列が更新されたすべての行、および履歴表が ON DELETE ADD EXTRA ROW で定義されている場合にシステム期間テンポラル表から削除されたすべての行に、データベース・マネージャーによって組み込みグローバル変数の値が割り当てられることを指定します。データ変更ステートメントの時点での組み込みグローバル変数の値が使用されます。単一の SQL ステートメントを使用して複数の行が変更される場合、列の値は、すべての行で同じになります。

data-type は、以下の表に従って定義する必要があります。

組み込みグローバル変数	列のデータ・タイプ
QSYS2.JOB_NAME	VARCHAR(28)
QSYS2.SERVER_MODE_JOB_NAME	VARCHAR(28)
SYSIBM.CLIENT_HOST	VARCHAR(255)
SYSIBM.CLIENT_IPADDR	VARCHAR(128)
SYSIBM.CLIENT_PORT	INTEGER
SYSIBM.PACKAGE_NAME	VARCHAR(128)
SYSIBM.PACKAGE_SCHEMA	VARCHAR(128)
SYSIBM.PACKAGE_VERSION	VARCHAR(64)
SYSIBM.ROUTINE_SCHEMA	VARCHAR(128)
SYSIBM.ROUTINE_SPECIFIC_NAME	VARCHAR(128)
SYSIBM.ROUTINE_TYPE	CHAR(1)

この列には、DEFAULT 文節もフィールド・プロシージャーも指定できません。

FIELDPROC
列のフィールド・プロシージャー出口ルーチンとして、外部プログラム名 を指定します。SQL が含まれていない ILE プログラムを指定する必要があります。サービス・プログラムを指定することはできません。

このフィールド・プロシージャーは列の値のエンコードとデコードを行います。列に値が挿入されるときは、フィールド・プロシージャーに渡されてエンコードされてから挿入されます。列に入っている値が使用されるときは、フィールド・プロシージャーに渡されてデコードされてから使用されます。

フィールド・プロシージャーは、CREATE TABLE ステートメントの処理中にも呼び出されます。この呼び出しの場合、プロシージャーは Db2 に列のフィールド記述を提供します。フィールド記述は、エンコードされた値のデータ特性を定義します。一方、CREATE TABLE ステートメントで列について指定する情報では、デコードされた値のデータ特性を定義します。

constant
フィールド・プロシージャーの呼び出し時にフィールド・プロシージャーに渡すパラメーターを指定します。パラメーター・リストはオプションです。

ROWID または DATALINK の列、あるいは ROWID または DATALINK に基づく特殊タイプの列でフィールド・プロシージャーを定義することはできません。この列は、ID 列、行変更タイム・スタンプ列、行開始列、行終了列、トランザクション開始 ID 列、および生成式列であってはなりません。この列に、CURRENT DATE、CURRENT TIME、CURRENT TIMESTAMP、USER のいずれかのデフォルト値が入っていてはなりません。チェック条件でこの列を参照することはできません。この列が外部キーの一部になっている場合は、対応する親キー列でも同じフィールド・プロシージャーを使用する必要があります。フィールド・プロシージャーの例については、『組み込み SQL プログラミング』トピック集を参照してください。

NOT NULL
列に NULL 値が入るのを防止します。NOT NULL を指定しないことは、その列が NULL であってもよいことを意味します。行変更タイム・スタンプ列、行開始列、および行終了列には、NOT NULL が必要です。

NOT HIDDEN
列が SQL ステートメントの表の暗黙的参照に組み込まれることを示します。これはデフォルトです。

IMPLICITLY HIDDEN
名前で明示的に参照されない限り、列は SQL ステートメントから不可視であることを示します。例えば、SELECT * は、結果に隠し列を含みません。表には、少なくとも 1 つの IMPLICITLY HIDDEN でない列が含まれていなければなりません。

datalink-options
DATALINK データ・タイプに関連したオプションを指定します。

LINKTYPE URL
リンクのタイプを URL として定義します。

NO LINK CONTROL
これを指定すると、リンク済みファイルが存在するか否かを判別するための

検査は行われなくなります。URL の構文だけが検査されます。リンク済みファイルに関してデータベース・マネージャー制御は行われません。

period-definition

PERIOD FOR
表の期間を定義します。

SYSTEM_TIME (*begin-column-name***, *end-column-name*)**
システム期間を SYSTEM_TIME という名前で定義します。表内に名前 SYSTEM_TIME の列があってはなりません。表に指定できる SYSTEM_TIME 期間は 1 つのみです。

begin-column-name
行が有効である期間の開始を記録する列を指定します。この名前は、表内に存在する列を示すものでなければなりません。*begin-column-name* は、*end-column-name* と同じであってはなりません。*begin-column-name* は AS ROW BEGIN として定義する必要があります。

end-column-name
行が有効である期間の終了を記録する列を指定します。システム期間テンポラル表に関連付けられた履歴表で、システム期間テンポラル表の *end-column-name* に対応する履歴表の列はその行の削除を反映するように設定されます。この名前は、表内に存在する列を示すものでなければなりません。*end-column-name* は AS ROW END として定義する必要があります。

LIKE

table-name または *view-name*
指定の表またはビューに定義されている列をこの表に含めることを指定します。LIKE 文節で指定する *table-name* または *view-name* は、アプリケーション・サーバーに既に存在している表またはビューを識別していなければなりません。

LIKE の使用は、n 列 (n は、識別された表またはビュー内の列数) を暗黙的に定義したことになります。この暗黙の定義には、n 列の以下の属性が含まれます (そのデータ・タイプに該当する場合)

- 列名 (および、システム列名)
- データ・タイプ、長さ、精度、および位取り
- CCSID

表名 の直後に LIKE 文節を指定し、括弧で囲まなかった場合は、以下の列属性も含まれます。その他の場合は、これらの属性は含まれません (デフォルト値、識別、行変更タイム・スタンプ、および隠し属性は、コピー・オプション を使用して制御することもできます)。

- デフォルト値 (表名 が指定され、ビュー名 は指定されていない場合)
- 識別属性
- NULL 可能性
- 隠し属性
- 行変更タイム・スタンプ属性
- 列の見出しとテキスト (1570 ページの『LABEL』を参照)

table-name に行変更タイム・スタンプ列、行開始列、行終了列、トランザクション開始 ID 列、または生成式列が含まれている場合、新しい表の対応する列はソース列のデータ・タイプのみを継承します。この新しい列は、生成された列とは見なされません。

指定された表またはビューが非 SQL 作成の物理ファイルまたは論理ファイルの場合、非 SQL 属性は除去されます。例えば、日付と時刻の形式は ISO 形式に変更されます。

暗黙の定義には、識別された表またはビューのその他のオプション属性は含まれません。例えば、新規の表には、表からの基本キーや外部キーは自動的には組み込まれません。新規の表にこうしたオプション属性が組み込まれるのは、オプション文節を明示的に指定した場合に限られます。

as-result-table

column-name
表を構成する列の名前を指定します。列名 は修飾できません。表の複数の列や表のシステム列名に同じ名前を使用することもできません。

FOR COLUMN *system-column-name*
列の IBM i 名を指定します。表の複数の列やシステム列名に対して、同じ名前を使用してはなりません。

システム列名が指定されず、また列名が有効なシステム列名でない場合には、システム列名が生成されます。システム列名の生成方法に関する詳細については、1297 ページの『列名の生成の規則』を参照してください。

select-statement
表の列の名前および記述が、選択ステートメント を実行した場合に選択ステートメント の派生結果表に現れる列と同じになるようにすることを指定します。AS 選択ステートメント を使用すると、この表について *n* 個の列を暗黙的に定義したことになります。*n* は、選択ステートメント の結果として発生する列の数です。この暗黙の定義には、*n* 列の以下の属性が含まれます (そのデータ・タイプに該当する場合):

- 列名 (および、システム列名)
- データ・タイプ、長さ、精度、および位取り
- CCSID
- NULL 可能性
- 列の見出しとテキスト (1570 ページの『LABEL』を参照)

以下の属性は組み込まれません (一部の属性は、*copy-options* を使用して組み込むことができます)。

- デフォルト値
- 非表示属性
- 識別属性
- 行変更タイム・スタンプ属性
- 行開始、行終了、およびトランザクション開始 ID 属性
- 生成式属性

暗黙の定義には、選択ステートメント で参照された表またはビューのその他のオプション属性は含まれません。

暗黙的に定義された列は、選択ステートメント の結果表の列の名前を継承します。したがって、すべての結果列について、選択ステートメント または列名リストの中で列名を指定する必要があります。式、定数、および関数から派生する結果列については、選択ステートメント で結果列の直後に AS 列名文節を指定するか、選択ステートメント の前の列リスト内に名前を指定する必要があります。

選択ステートメント は、変数または組み込みパラメーター・マーカー (疑問符) を参照するものであってはなりません。選択ステートメント には、PREVIOUS VALUE 式または NEXT VALUE 式を含めてはなりません。UPDATE、SKIP LOCKED DATA、USE AND KEEP EXCLUSIVE LOCKS の各文節を指定することはできません。

select-statement が *isolation-clause* を含む場合、*isolation-clause* で指定された分離レベルが SQL ステートメント全体に適用されます。

WITH DATA
選択ステートメント を実行することを指定します。表の作成後に、選択ステートメント の結果表の行が自動的に表に挿入されます。

WITH NO DATA
選択ステートメント を実行しないことを指定します。したがって、自動的に表に挿入される行のセットを持つ結果表はありません。

copy-options

INCLUDING IDENTITY COLUMN ATTRIBUTES または **EXCLUDING IDENTITY COLUMN ATTRIBUTES**
ID 列の属性を継承するかどうかを指定します。

INCLUDING IDENTITY COLUMN ATTRIBUTES
この表が、選択ステートメント、表名、またはビュー名 の結果として生じる列の識別属性 (もしあれば) を継承することを指定します。一般に、識別属性がコピーされるのは、表、ビュー、または選択ステートメント の中の対応する列のエレメントが、識別属性を持つ基本表列の名前に直接または間接にマップされる表列またはビュー列の名前である場合です。
INCLUDING IDENTITY COLUMN ATTRIBUTES 文節と AS 選択ステートメント 文節を指定してあるときは、以下の場合には新規の表の列は識別属性を継承しません。

- 選択ステートメント の選択リストに、ID 列名の複数のインスタンスが含まれている (つまり同じ列を複数回選択している) 場合。
- 選択ステートメント の選択リストに、複数の ID 列が含まれている (つまり、結合が複数の ID 列を戻した) 場合。
- 選択リスト内の式のいずれかに ID 列が含まれている場合。
- 選択ステートメント に一組の演算 (UNION または INTERSECT) が含まれている場合。

INCLUDING IDENTITY を指定しなかった場合は、表には ID 列は含まれません。

EXCLUDING IDENTITY COLUMN ATTRIBUTES
この表が、全選択、表名、またはビュー名 の結果として生じる列の識別属性を継承しないことを指定します。

EXCLUDING COLUMN DEFAULTS または **INCLUDING COLUMN DEFAULTS** または **USING TYPE DEFAULTS**
列のデフォルトを継承するかどうかを指定します。

EXCLUDING COLUMN DEFAULTS
ソース表の定義から列のデフォルトを継承しないことを指定します。新しい表の列のデフォルト値は NULL になるか、またはデフォルト値がなくなります。列を NULL にできる場合、デフォルトは NULL 値になります。列を NULL にできない場合はデフォルト値がなくなるため、新しい表に対する INSERT で列の値が指定されない場合はエラーが発生します。

INCLUDING COLUMN DEFAULTS
この表が、選択ステートメント、表名、またはビュー名 から生じる列のデフォルト値を継承することを指定します。デフォルト値は、INSERT で値が指定されていない場合に、列に割り当てられる値です。

USING TYPE DEFAULTS を指定する場合は、INCLUDING COLUMN DEFAULTS を指定しないでください。

INCLUDING COLUMN DEFAULTS を指定しなかった場合は、デフォルト値は継承されません。

USING TYPE DEFAULTS
この表のデフォルト値が、選択ステートメント、表名、またはビュー名 から生じる列のデータ・タイプに応じて決まることを指定します。その列が NULL 可能である場合は、デフォルト値は NULL 値です。その他の場合は、デフォルト値は以下のようになります。

データ・タイプ	デフォルト値
数値	0
固定長文字またはグラフィック・ストリング	ブランク
固定長バイナリー・ストリング	16 進ゼロ
可変長ストリング	0 のストリング長
日付	INSERT の時点の当日の日付
時刻	INSERT の時点の当日の時刻
タイム・スタンプ	INSERT の時点の当日のタイム・スタンプ
データ・リンク	DLVALUE('','URL','') に対応する値
特殊タイプ	特殊タイプの対応するソース・タイプのデフォルト値

INCLUDING COLUMN DEFAULTS を指定する場合は、USING TYPE DEFAULTS は指定しないでください。

INCLUDING IMPLICITLY HIDDEN COLUMN ATTRIBUTES または **EXCLUDING IMPLICITLY HIDDEN COLUMN ATTRIBUTES**
暗黙的な隠し列を継承するかどうかを指定します。

INCLUDING IMPLICITLY HIDDEN COLUMN ATTRIBUTES
この表が、選択ステートメント、表名、またはビュー名 から暗黙的な隠し列を継承することを示します。また、これらの列は、新規の表の暗黙的隠し属性を使用して定義されます。

INCLUDING IMPLICITLY HIDDEN COLUMN ATTRIBUTES を指定しない場合、表は、暗黙的な隠し列を持ちません。

EXCLUDING IMPLICITLY HIDDEN COLUMN ATTRIBUTES
表が、全選択、表名、または ビュー名 から、暗黙的な隠し列を継承しないことを指定します。

INCLUDING ROW CHANGE TIMESTAMP COLUMN ATTRIBUTES または **EXCLUDING ROW CHANGE TIMESTAMP COLUMN ATTRIBUTES**
行変更タイム・スタンプ属性が継承されるかどうかを指定します。

INCLUDING ROW CHANGE TIMESTAMP COLUMN ATTRIBUTES
この表が、選択ステートメント、表名、またはビュー名 の結果として生じる列の行変更タイム・スタンプ属性 (もしあれば) を継承することを指定します。一般に、行変更タイム・スタンプ属性がコピーされるのは、表、ビュー、または選択ステートメント の中の対応する列のエレメントが、行変更タイム・スタンプ属性を指定する基本表列の名前に直接または間接にマップされる表列またはビュー列の名前である場合です。 INCLUDING ROW COLUMN ATTRIBUTES 文節と AS 選択ステートメント 文節を指定してあるときは、以下の場合には新規の表の列は、行変更タイム・スタンプ属性を継承しません。

- 選択ステートメント の選択リストに、行変更タイム・スタンプ列名の複数インスタンスが含まれている (つまり同じ列を複数回選択している) 場合。
- 選択ステートメント の選択リストに、複数の行変更タイム・スタンプ列が含まれている (つまり、結合が複数の行変更タイム・スタンプ列を戻した) 場合。
- 選択リスト内の式に行変更タイム・スタンプ列が含まれている場合。
- 選択ステートメント に一組の演算 (UNION または INTERSECT) が含まれている場合。

INCLUDING ROW CHANGE TIMESTAMP COLUMN ATTRIBUTES を指定しない場合、表は、行変更タイム・スタンプ列を持ちません。

EXCLUDING ROW CHANGE TIMESTAMP COLUMN ATTRIBUTES
この表が、全選択、表名、またはビュー名 の結果として生じる列の行変更タイム・スタンプ属性 (もしあれば) を継承しないことを指定します。

WITH REPLACE
指定した名前の宣言済み一時表が既に存在している場合に、その既存の表を、このステートメントで定義した一時表で置き換える (そして既存の表のすべての行を削除する) ことを指定します。

WITH REPLACE を指定しない場合は、現行セッション内に既に存在しているどの宣言済み一時表とも異なる名前を指定する必要があります。

ON COMMIT
COMMIT 操作が行われるときにこの宣言済み一時表に対して行うアクションを指定します。デフォルトは DELETE ROWS です。

この宣言済み一時表が No Commit (NC) 分離レベルでオープンされた場合、または COMMIT HOLD 操作が行われる場合は、ON COMMIT 文節は適用されません。

DELETE ROWS
表について WITH HOLD カーソルがオープンされていない場合は、表のすべての行が削除されます。

PRESERVE ROWS
表の行は保存されます。

NOT LOGGED
表に対する変更 (表の作成も含む) をログに記録しないことを指定します。ROLLBACK (または ROLLBACK TO SAVEPOINT) 操作を行うときに、この作業単位 (またはセーブポイント) で表が変更されていても、その変更はロールバックされません。この作業単位 (またはセーブポイント) の中で表を作成した場合、その表は除去されます。この作業単位 (またはセーブポイント) の中で表を除去した場合は、その表は復元されますが、行は含まれていません。

ON ROLLBACK
ROLLBACK 操作が行われるときにこの宣言済み一時表に対して行うアクションを指定します。

この宣言済み一時表が No Commit (NC) 分離レベルでオープンされた場合、または ROLLBACK HOLD 操作が行われる場合は、ON ROLLBACK 文節は適用されません。

DELETE ROWS
表のすべての行が削除されます。これはデフォルトです。

PRESERVE ROWS
表の行は保存されます。

RCDFMT *format-name*
表の IBM i レコード・フォーマット名を指定する非修飾名です。 *format-name* は、システム ID です。

レコード・フォーマット名が指定されない場合、*format-name* は、表の *system-object-name* と同一のものになります。

media-preference
表の優先ストレージ・メディアを指定します。

UNIT ANY
どのストレージ・メディアも優先しません。表のストレージは、使用可能な任意のストレージ・メディアから割り振られます。

UNIT SSD
ソリッド・ステート・ディスク・ストレージ・メディアを優先します。表のストレージは、使用可能な場合にはソリッド・ステート・ディスク・ストレージ・メディアから割り振ることができます。

KEEP IN MEMORY
表のデータが照会で使用されるときに、データを主記憶域プールに入れるかどうかを指定します。

NO データは主記憶域プールに入れられません。

YES
データは主記憶域プールに入れられます。

注

インスタンス化、有効範囲、および終了 **:** P はアプリケーション・プロセスを表し、T は、P の中のアプリケーション・プログラム内の宣言済み一時表であるものとします。

- P の中のプログラムが DECLARE GLOBAL TEMPORARY TABLE ステートメントを発行すると、T の空のインスタンスが作成されます。
- P の中のどのプログラムも T を参照でき、それらの参照はどれも T の同じインスタンスに対する参照です。(SQL 関数、SQL プロシージャー、またはトリガーの複合ステートメント内で DECLARE GLOBAL TEMPORARY ステートメントを指定した場合は、宣言済み一時表の有効範囲は、その複合ステートメントではなくアプリケーション・プロセスです。)

  T がリモート・サーバーで宣言されたものである場合は、T に対する参照では、T を宣言するときに使用したものと同じ接続を使用する必要があり、その接続は T の宣言後に終了していてはなりません。 T が宣言されたデータベース・サーバーへの接続が終了すると、T は除去されます。
- T の定義に ON COMMIT DELETE ROWS 文節が含まれている場合は、コミット操作により P の中の 1 つの作業単位が終了した時点で、P の中のどのプログラムについても T に依存する WITH HOLD カーソルがオープン状態にないときは、すべての行が削除されます。
- T の定義に ON ROLLBACK DELETE ROWS 文節が含まれている場合は、ロールバック操作により P の中の 1 つの作業単位が終了した時点で、すべての行が削除されます。
- T を宣言したアプリケーション・プロセスが終了すると、T は除去されます。

一時表の所有権 **:** この表の所有者 は、このステートメントを実行しているスレッドのユーザー・プロファイルです。

一時表の権限 **:** 宣言済み一時表を定義するときに、その表に関するすべての表特権、およびその表を除去する権限が、PUBLIC に対して暗黙に認可されます。

他の **SQL** ステートメント内での宣言済み一時表の参照 **:** 多くの SQL ステートメントが宣言済み一時表をサポートしています。 DECLARE GLOBAL TEMPORARY TABLE 以外の SQL ステートメントの中で一時表を参照するには、その表を暗黙的または明示的に SESSION で修飾する必要があります。

表名の修飾子として SESSION を使用しても、アプリケーション・プロセスに、その表名についての DECLARE GLOBAL TEMPORARY TABLE ステートメントが

含まれていない場合は、データベース・マネージャーは、宣言済み一時表が参照されているのではないと判断します。そして、データベース・マネージャーは、このような表参照を永続表に解決します。

宣言済み一時表の使用に関する制限 :

- ALTER TABLE、COMMENT、CREATE ALIAS、GRANT、LABEL、LOCK、RENAME、または REVOKE ステートメントでは、宣言済み一時表は指定できません。
- 宣言済み一時表は、参照制約の中で親表として指定することはできません。
- CREATE INDEX または CREATE VIEW ステートメントで宣言済み一時表を参照する場合は、該当の索引またはビューは、SESSION (または QTEMP ライブラリー) の中に作成する必要があります。

リモート選択ステートメントを使用した表作成**:** *as-result-table* の *select-statement* は、表を作成するサーバーとは別のサーバー上の表を参照できます。これを行うには、3 部構成のオブジェクト名を使用するか、または、表またはビューの 3 部構成の名前を参照するよう定義された別名を使用します。*select-statement* の結果は、フィールド・プロシージャーが定義された列を含むことはできません。リモート・サーバーが Db2 for LUW または Db2 for z/OS である場合、*copy-options* は許可されません。リモート・サーバーが Db2 for LUW の場合、AS キーワードの前に列リストを明示的に指定する必要があります。

代替構文**:** 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- INLINE LENGTH は ALLOCATE の同義語です。
- DEFINITION ONLY は WITH NO DATA の同義語です。

例

例 *1:* 社員番号、給与、歩合給、および賞与に関する列定義のある宣言済み一時表を定義します。

```
DECLARE GLOBAL TEMPORARY TABLE SESSION.TEMP_EMP
  (EMPNO    CHAR(6)     NOT NULL,
   SALARY    DECIMAL(9, 2),
   BONUS     DECIMAL(9, 2),
   COMM      DECIMAL(9, 2))
  ON COMMIT PRESERVE ROWS
```

例 *2:* USER1.EMPTAB という基本表に 3 つの列があり、その 1 つが ID 列であるとします。この基本表を同じ列名および属性 (識別属性も含む) を持つ一時表を宣言します。

```
DECLARE GLOBAL TEMPORARY TABLE TEMPTAB1
  LIKE USER1.EMPTAB
  INCLUDING IDENTITY
  ON COMMIT PRESERVE ROWS
```

上記の例では、データベース・マネージャーは TEMPTAB1 の暗黙修飾子として SESSION を使用します。

DECLARE PROCEDURE

DECLARE PROCEDURE ステートメントは、外部プロシージャーを定義します。

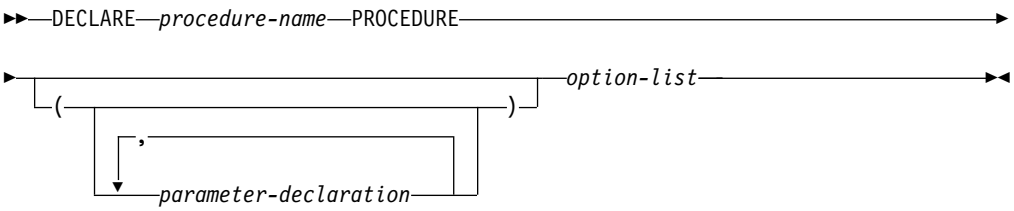
呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、実行可能ステートメントではありません。REXX で指定してはなりません。

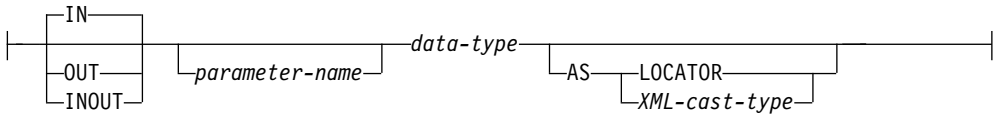
権限

不要です。

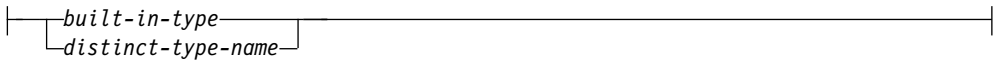
構文



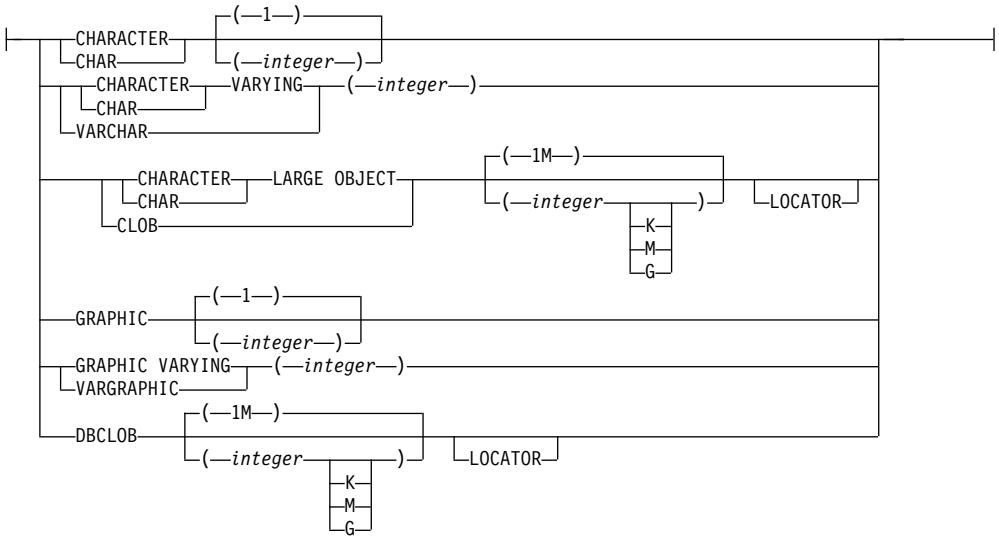
parameter-declaration:



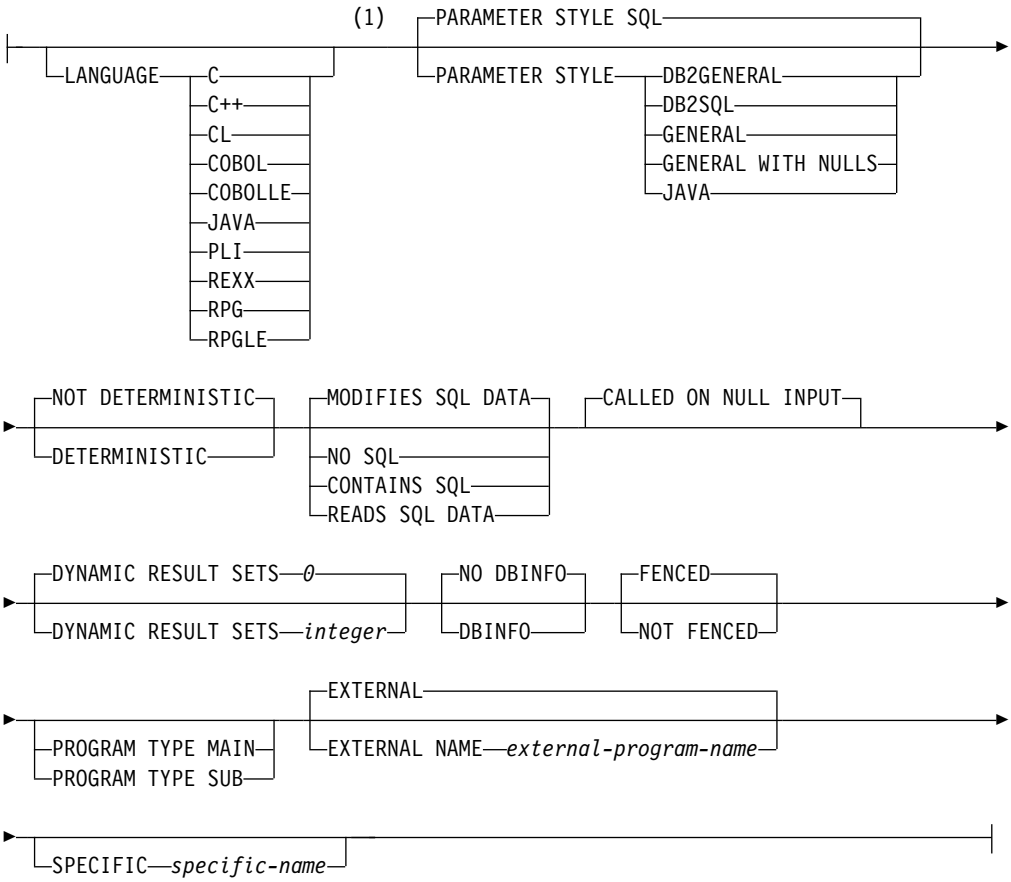
data-type:



XML-cast-type:



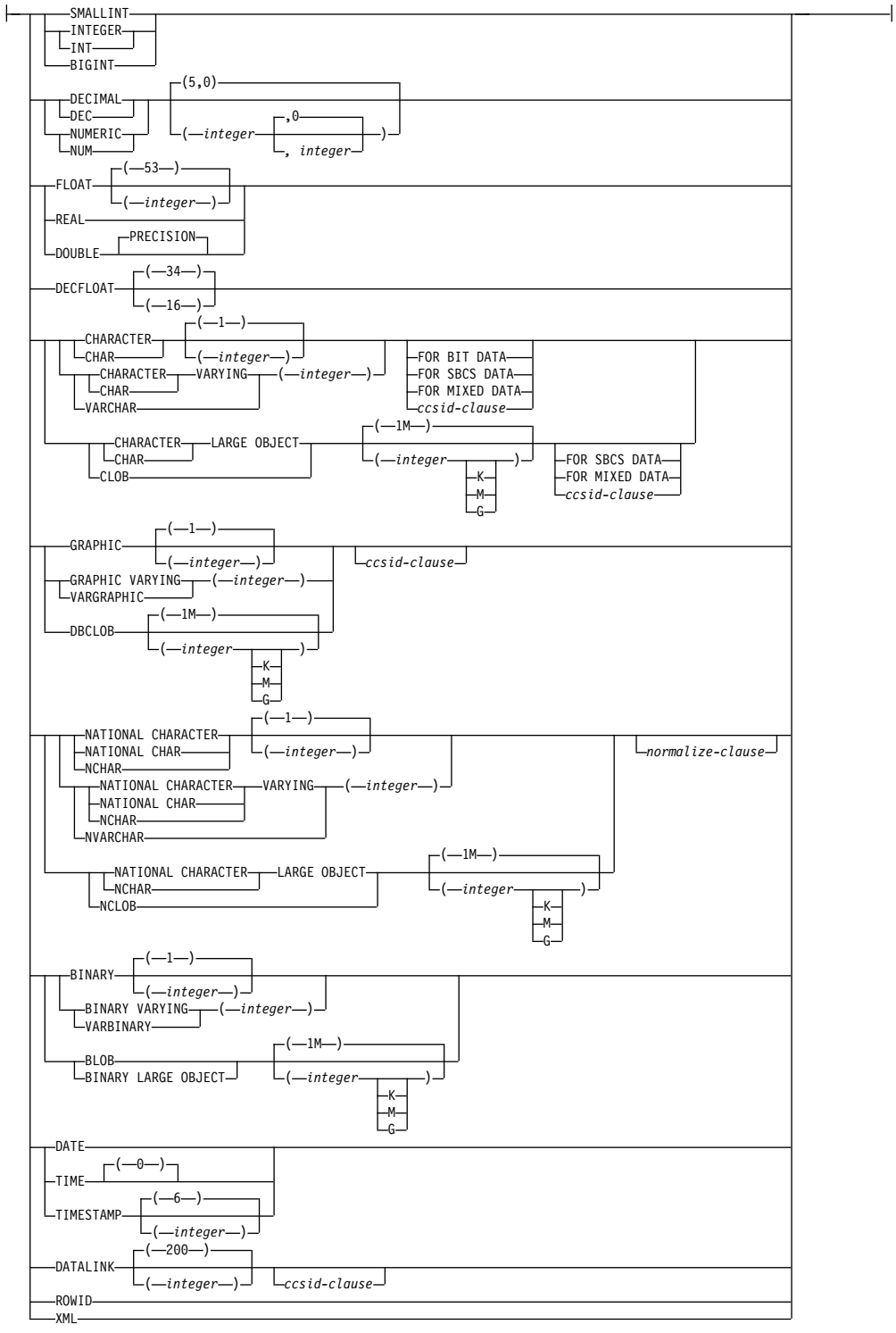
option-list:



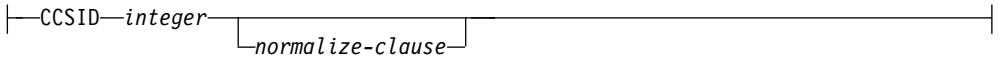
注:

1 オプション文節は、別の順序で指定することができます。

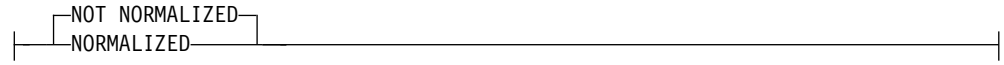
built-in-type:



ccsid-clause:



normalize-clause:



説明

procedure-name
プロシージャーを指定します。この名前は、そのソース・プログラムで宣言されている他のプロシージャーの名前と同じであってはなりません。

(*parameter-declaration,...***)**
プロシージャーのパラメーターの数とそれぞれのパラメーターのデータ・タイプを指定します。 プロシージャーに関するパラメーターは、入力専用、出力専用、または入出力両用に使用できます。それぞれのパラメーターに名前を指定することができますが、これは必須ではありません。

CREATE PROCEDURE で使用できるパラメーターの最大数は言語のタイプとパラメーター・スタイルによって異なり、以下のようになります。

- JAVA と ILE プログラムおよびサービス・プログラムの場合、最大数は 2000 です。
- OPM プログラムおよび REXX の場合は、以下のようになります。
  - PARAMETER STYLE GENERAL を指定した場合、最大数は 255 です。
  - PARAMETER STYLE GENERAL WITH NULLS を指定した場合、最大数は 254 です。
  - PARAMETER STYLE SQL を指定した場合、最大数は 254 です。

パラメーターの最大数は、その言語で許されるパラメーターの最大数によってさらに制限される可能性があります。

IN パラメーターが、プロシージャーへの入力パラメーターであることを指定します。プロシージャー内でパラメーターに対する変更が行われても、制御が戻った時点で、呼び出し元の SQL アプリケーションがその変更内容を使用することはできません。[101]

OUT
パラメーターが、プロシージャーから戻される出力パラメーターであることを示します。

データ・リンクやデータ・リンクをベースとした特殊タイプは、出力パラメーターとして指定することはできません。

INOUT
パラメーターが、このプロシージャー用の入出力両方のパラメーターであることを指定します。

データ・リンクやデータ・リンクをベースとした特殊タイプは、入出力パラメーターとして指定することはできません。

101. 言語タイプが REXX の場合は、すべてのパラメーターが入力パラメーターでなければなりません。

parameter-name
パラメーター名を指定します。この名前は、このプロシージャー用の他のパラメーター名 と同じものであってはなりません。

data-type
パラメーターのデータ・タイプを指定します。

指定するデータ・タイプは LANGUAGE 文節で指定する言語にとって有効なものでなければなりません。すべてのデータ・タイプが SQL プロシージャーに有効です。データ・リンクは、外部プロシージャーには無効です。データ・タイプについて詳しくは、 1238 ページの『CREATE TABLE』、および「SQL プログラミング」のトピック集を参照してください。

CCSID が指定されている場合、プロシージャーに渡される前に、パラメーターはその CCSID に変換されます。CCSID が指定されていない場合は、CCSID は、プロシージャーの呼び出し時点における現行サーバーのデフォルトの CCSID によって決まります。

XML タイプのパラメーターでは、*XML-cast-type* 文節を指定する必要があります。

AS LOCATOR
これを指定すると、パラメーターは、実際の値ではなく、値のロケーターになります。AS LOCATOR は、パラメーターに LOB または XML データ・タイプや LOB または XML データ・タイプをベースとする特殊タイプが指定されている場合にのみ、指定することができます。AS LOCATOR を指定した場合、FOR SBCS DATA または FOR MIXED DATA を指定してはなりません。

AS *XML-cast-type*
XML タイプまたは XML タイプに基づく特殊タイプのパラメーターのデータ・タイプとしてこのプロシージャーに渡すデータ・タイプを指定します。LOCATOR を指定する場合は、パラメーターが実際の値ではなく値のロケーターになります。

CCSID 値を指定する場合に、グラフィック・データ・タイプとして指定できるのは、Unicode CCSID 値に限られます。CCSID 値を指定しない場合は、このプロシージャーが含まれているプログラムやモジュールやサービス・プログラムの作成時に、SQL_XML_DATA_CCSID QAQQINI オプションの設定に基づいて CCSID が設定されます。デフォルトの CCSID は 1208 です。このオプションの説明については、 101 ページの『XML 値』を参照してください。

DYNAMIC RESULT SETS *integer*
プロシージャーから戻すことのできる結果セットの最大数を指定します。整数は、ゼロ以上で 32768 より小さくなければなりません。ゼロを指定すると、結果セットは戻されません。 SET RESULT SETS ステートメントを発行した場合は、戻される結果の数は、このキーワードに指定した結果セットの数と、 SET RESULT SETS ステートメントに指定した結果セットの数のいずれか少ない方です。

結果セットの詳細については、 1724 ページの『SET RESULT SETS』を参照してください。

LANGUAGE
その外部プログラムの作成に使用されている言語を指定します。この文節は、外部プログラムが REXX プロシージャーである場合に必要です。

LANGUAGE の指定がない場合は、その LANGUAGE は該当の外部プログラムに関連するプログラム属性情報によって決定されます。該当のプログラムに関連するプログラム属性情報によっては認識可能な言語を特定できない場合には、言語は C であると見なされます。

C 外部プログラムは C で作成されます。

C++
外部プログラムは C++ で作成されます。

CL 外部プログラムは CL で作成されます。

COBOL
外部プログラムは COBOL で作成されます。

COBOLLE
外部プログラムは ILE COBOL で作成されます。

JAVA
外部プログラムは JAVA で作成されます。

PLI
外部プログラムは PL/I で作成されます。

REXX
外部プログラムは REXX プロシージャーです。

RPG
外部プログラムは RPG で作成されます。

RPGLE
外部プログラムは ILE RPG で作成されます。

PARAMETER STYLE
プロシージャーにパラメーターを渡し、プロシージャーから値を戻すために使用する規則を指定します。

SQL
CALL ステートメントに指定されているパラメーターに加えて、幾つかの追加パラメーターをプロシージャーに渡すことを指定します。これらのパラメーターは、次の順序で配列されるように定義されます。

- 最初の N 個のパラメーターは、DECLARE PROCEDURE ステートメント上に指定されるパラメーターです。
- パラメーターの標識変数を表す N 個のパラメーター。
- SQLSTATE の CHAR(5) 出力パラメーター。戻される SQLSTATE は、プロシージャーが成功したかどうかを示します。戻される SQLSTATE は、外部プログラムによって割り当てられたものです。

  ユーザーは、関数からエラーまたは警告を戻すために、外部プログラム内で SQLSTATE を任意の有効な値にセットすることができます。
- 完全修飾プロシージャー名の VARCHAR(517) 入力パラメーター。
- 特定の名前の VARCHAR(128) 入力パラメーター。

- メッセージ・テキストの VARCHAR(1000) 出力パラメーター。

以下の追加のパラメーターを最後のパラメーターとして渡すことができます。

- DBINFO が CREATE PROCEDURE ステートメント上に指定されている場合は、dbinfo 構造体のパラメーター。

これらのパラメーターは、指定の LANGUAGE に基づいて渡されます。例えば、言語が C または C++ であれば、VARCHAR パラメーターはヌル終了ストリングとして渡されます。 渡されるパラメーターについての詳細は、ライブラリー QSYSINC の該当するソース・ファイル内の組み込み sqludf を参照してください。例えば、C の場合、sqludf は QSYSINC/H で見つかります。

LANGUAGE JAVA を指定した場合は、PARAMETER STYLE SQL は使用できません。

DB2GENERAL
このプロシージャーに、Java メソッド用として定義されているパラメーター引き渡し規則を使用することを指定します。

PARAMETER STYLE DB2GENERAL を指定できるのは、LANGUAGE JAVA を指定した場合のみです。 Java でのパラメーター引き渡しの詳細については、「IBM Developer Kit for Java」トピック集を参照してください。

GENERAL
このプロシージャーが CALL に指定されているパラメーターを受け取るようなパラメーター引き渡しメカニズムを使用することを指定します。標識変数に対し、引数がさらに渡されることはありません。

LANGUAGE JAVA を指定した場合は、PARAMETER STYLE GENERAL は使用できません。

GENERAL WITH NULLS
CALL ステートメントで GENERAL に指定されているパラメーターに加えて、他の引数もプロシージャーに渡すことを指定します。この追加の引数には、CALL ステートメントの各パラメーターについてそれぞれ 1 つずつエレメントがある標識配列が含まれています。 C では、これは多くの場合、短精度整数の配列です。標識の処理方法について詳しくは、「SQL プログラミング」トピック集を参照してください。

LANGUAGE JAVA を指定した場合は、PARAMETER STYLE GENERAL WITH NULLS は使用できません。

JAVA
このプロシージャーで、Java 言語および SQLJ ルーチンの仕様に準拠するパラメーター引き渡し規則を使用することを指定します。 INOUT および OUT パラメーターは、値を戻しやすくするために、単一項目配列として渡されます。移植性を高めるためには、PARAMETER STYLE JAVA 規則を使用する Java プロシージャーを書く必要があります。

PARAMETER STYLE JAVA を指定できるのは、LANGUAGE JAVA を指定した場合だけです。 Java でのパラメーター引き渡しの詳細については、「IBM Developer Kit for Java」トピック集を参照してください。

パラメーターを渡す方法は、外部関数の言語によって決まります。例えば、C では、VARCHAR または CHAR パラメーターは NULL 文字で終了するストリングとして渡されます。詳しくは、「SQL プログラミング」を参照してください。Java ルーチンについては、「IBM Developer Kit for Java」を参照してください。

SPECIFIC *specific-name*
プロシージャーを一意に識別する修飾または非修飾名を指定します。暗黙的または明示的修飾子も含め、この特定名 には、プロシージャー名 と同じ名前を指定する必要があります。

修飾子を指定しないと、プロシージャー名 の暗黙的または明示的修飾子が使用されます。修飾子を指定する場合、その修飾子は、プロシージャー名 の明示的または暗黙的修飾子と同じものにする必要があります。

特定名 を指定しなかった場合、その特定名は、プロシージャー名と同じ名前になります。

DETERMINISTIC または **NOT DETERMINISTIC**
このプロシージャーが、同じ IN 引数および INOUT 引数を指定して呼び出された場合に、常に同じ結果を戻すかどうかを指定します。

NOT DETERMINISTIC
このプロシージャーは、同じ IN 引数および INOUT 引数を指定して呼び出された場合に、データベース内の参照先データが変更されていなくても、必ずしも同じ結果を戻すとは限りません。

DETERMINISTIC
このプロシージャーは、同じ IN 引数および INOUT 引数を指定して呼び出された場合に、データベース内の参照先データが変更されていない限り、常に同じ結果を戻します。

MODIFIES SQL DATA、**READS SQL DATA**、**CONTAINS SQL**、または **NO SQL**
SQL ステートメントがある場合に、このプロシージャーまたはこのプロシージャーから呼び出されたルーチンの中で、どの SQL ステートメントを実行できるかを指定します。デフォルトは、MODIFIES SQL DATA です。各データ・アクセス指示の下で実行できる SQL ステートメントの詳細なリストについては、1855 ページの『付録 B. SQL ステートメントの特性』を参照してください。

MODIFIES SQL DATA
このプロシージャーで、どのプロシージャーでもサポートされないステートメントを除くすべての SQL ステートメントを実行できることを指定します。

READS SQL DATA
このプロシージャーに、SQL データを変更しない SQL ステートメントを組み込めることを指定します。

CONTAINS SQL
このプロシージャーで、SQL データの読み取りも変更も行わない SQL ステートメントを実行できることを指定します。

NO SQL
このプロシージャーではどの SQL ステートメントも実行できないことを指定します。

CALLED ON NULL INPUT
引数値のいずれかまたは全部が NULL である場合、関数を呼び出して、その関数に NULL 引数値のテストを行わせることを指定します。関数は NULL または非 NULL 値を戻すことができます。

FENCED または NOT FENCED
このパラメーターは、他のプロダクトとの互換性を保持するために許可されており、Db2 for i で使用されることはありません。

PROGRAM TYPE MAIN または PROGRAM TYPE SUB
他の製品との互換性を備えるために、このパラメーターが許可されています。これは、ルーチンの外部プログラムが、プログラム (*PGM) であるか、サービス・プログラム (*SRVPGM) のプロシージャーであるかを示します。

PROGRAM TYPE MAIN
ルーチンがプログラムのメインエントリー・ポイントとして実行することを指定します。外部プログラムは、*PGM オブジェクトでなければなりません。

PROGRAM TYPE SUB
プロシージャーがサービス・プログラムのプロシージャーとして実行することを指定します。外部プログラムは、*SRVPGM オブジェクトでなければなりません。

DBINFO
データベース・マネージャーは、状況情報が入っている構造体をプロシージャーに渡す必要があることを指定します。 表 86 は、DBINFO 構造体の説明を示しています。 DBINFO 構造体についての詳しい情報は、ライブラリー QSYSINC 内の該当するソース・ファイルの組み込み sqludf に入っています。例えば、C の場合、sqludf は QSYSINC/H で見つかります。

DBINFO は、PARAMETER STYLE DB2SQL でのみ許可されます。

表 *86. DBINFO* フィールド

フィールド	データ・タイプ	説明
リレーショナル・データベース	VARCHAR(128)	現行サーバーの名前
権限 ID	VARCHAR(128)	実行時権限 ID

表 86. DBINFO フィールド (続き)

フィールド	データ・タイプ	説明
CCSID 情報	INTEGER INTEGER INTEGER INTEGER INTEGER INTEGER INTEGER INTEGER INTEGER INTEGER CHAR(8)	ジョブの CCSID 情報。3 つの CCSID が 3 セット戻されます。各セット内の 3 つの CCSID を識別する情報は、次のとおりです。 • SBCS CCSID • DBCS CCSID • 混合 CCSID 3 セットの CCSID の後に、3 セットの CCSID のうちのどのセットが該当するかを示す整数と 8 バイトの予約済みスペースが続きます。 CCSID の各設定は、異なるエンコード・スキーム (EBCDIC、ASCII、およびユニコード) のためのものです。 CREATE PROCEDURE ステートメントのパラメーターの 1 つとして明示的に CCSID を指定していない場合は、入力ストリングは、プロシージャーの実行時にジョブの CCSID でコード化されるものと見なされます。入力ストリングの CCSID がパラメーターの CCSID と同じではない場合は、この外部プロシージャーに渡される入力ストリングは、外部プログラムの呼び出しの前に変換されます。
ターゲット列	VARCHAR(128) VARCHAR(128) VARCHAR(128)	プロシージャーへの呼び出しには適用されません。
バージョンとリリース	CHAR(8)	データベース・マネージャーのバージョン、リリース、および修正レベル。
プラットフォーム	INTEGER	サーバーのプラットフォーム・タイプ。

EXTERNAL NAME 外部プログラム名

該当のプロシージャーが CALL ステートメントによって呼び出される時点で実行されるプログラムを指定します。このプログラム名は、アプリケーション・サーバーに存在するプログラムを識別していなければなりません。このプログラムは、ILE サービス・プログラムであってはなりません。

この名前の妥当性は、アプリケーション・サーバーで検査されます。名前の形式が正しくない場合、エラーが戻されます。

外部プログラム名の指定がない場合、外部プログラム名は該当のプロシージャー名と同じであると見なされます。

注

DECLARE PROCEDURE の有効範囲**:** プロシージャー名 の有効範囲は、それが定義されているソース・プログラムです。つまり、プリコンパイラーに実行依頼されるプログラムです。したがって、別個にコンパイルされた他のプログラムやモジュールから呼び出されるプログラムは、呼び出し側プログラムの DECLARE

PROCEDURE ステートメントからの属性を使用しません。

DECLARE PROCEDURE 規則: DECLARE PROCEDURE ステートメントは、そのプロシージャーを参照するすべての CALL ステートメントよりも前に入れなければなりません。

DECLARE PROCEDURE ステートメントが適用されるのは、静的 CALL ステートメントだけです。動的に準備された CALL ステートメント、またはプロシージャー名が変数によって識別されている CALL ステートメントには適用されません。

代替構文: 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード VARIANT と NOT VARIANT は、NOT DETERMINISTIC と DETERMINISTIC の同義語として使用することができます。
- キーワード NULL CALL は、CALLED ON NULL INPUT の同義語として使用できます。
- キーワード SIMPLE CALL は、GENERAL の同義語として使用できます。
- DB2GENERAL の同義語として、値 DB2GENRL を使用できます。
- PARAMETER STYLE 文節のキーワード PARAMETER STYLE はオプションです。
- PARAMETER STYLE SQL の同義語として、キーワード PARAMETER STYLE DB2SQL を使用できます。

例

外部プロシージャー PROC1 を、C プログラムの中で宣言します。 CALL ステートメントを使用してこのプロシージャーを呼び出すと、LIB1 ライブラリーの中の PGM1 という名前の COBOL プログラムが呼び出されます。

```
EXEC SQL
DECLARE PROC1 PROCEDURE
        (CHAR(10), CHAR(10))
        EXTERNAL NAME LIB1.PGM1
        LANGUAGE COBOL GENERAL;

EXEC SQL
CALL PROC1 ('FIRSTNAME ','LASTNAME  ');
```

DECLARE STATEMENT

DECLARE STATEMENT ステートメントは、プログラムの文書化の目的に使用します。このステートメントは、準備される SQL ステートメントを識別するのに使用する名前を宣言します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、実行可能ステートメントではありません。このステートメントは、Java または REXX では使用できません。

権限

権限は不要です。

構文

説明

statement-name
準備される SQL ステートメントを識別するために、プログラムで使用される 1 つ以上の名前をリストします。

例

この例は、C プログラムにおける DECLARE STATEMENT ステートメントの使用法を示しています。

```
EXEC SQL INCLUDE SQLDA;
void main ()
  {
   EXEC SQL BEGIN DECLARE SECTION ;
   char src_stmt[32000];
   char sqlda[32000]
   EXEC SQL END DECLARE SECTION ;
   EXEC SQL INCLUDE SQLCA ;

   strcpy(src_stmt,"SELECT DEPTNO, DEPTNAME, MGRNO \
     FROM DEPARTMENT \
     WHERE ADMRDEPT = 'A00'");

   EXEC SQL DECLARE OBJ_STMT STATEMENT;

   (Allocate storage from SQLDA)

   EXEC SQL DECLARE C1 CURSOR FOR OBJ_STMT;

   EXEC SQL PREPARE OBJ_STMT FROM :src_stmt;
   EXEC SQL DESCRIBE OBJ_STMT INTO :sqlda;

   (Examine SQLDA)  (Set SQLDATA pointer addresses)

   EXEC SQL OPEN C1;
```

```
while (strncmp(SQLSTATE, "00000", 5) )
  {
  EXEC SQL FETCH C1 USING DESCRIPTOR :sqlda;

  (Print results)

  }

EXEC SQL CLOSE C1;
return;
}
```

DECLARE VARIABLE

DECLARE VARIABLE ステートメントは、ホスト変数に対して、デフォルト値以外のサブタイプまたは CCSID を割り当てるのに使用します。

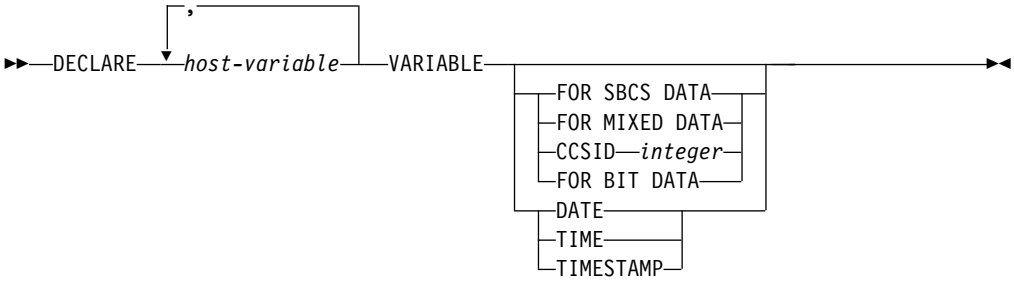
呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、実行可能ステートメントではありません。Java および REXX では指定できません。

権限

権限は不要です。

構文



説明

host-variable
該当のプログラムで定義されている文字、グラフィック、または XML ストリングのホスト変数の名前を指定します。このホスト変数には、標識変数は指定できません。そのホスト変数の定義は、その変数を参照する DECLARE VARIABLE ステートメントの前でも後でも構いません。

FOR BIT DATA
ホスト変数の値が、コード化文字セットに関連付けられていないことを指定します。したがって、変換は行われません。 FOR BIT DATA が指定されたホスト変数の CCSID は 65535 です。 FOR BIT DATA は、グラフィックまたは XML のホスト変数には指定できません。

FOR SBCS DATA
ホスト変数の値に、SBCS (1 バイト文字セット) データが入ることを指定します。アプリケーション・リクエスターにおけるジョブの CCSID 属性が DBCS 対応でない場合や、ホスト変数の長さが 4 より小さい場合は、FOR SBCS DATA がデフォルト値となります。FOR SBCS DATA の CCSID は、アプリケーション・リクエスターにおけるジョブの CCSID 属性によって決まります。 FOR SBCS DATA は、グラフィックまたは XML のホスト変数には指定できません。

FOR MIXED DATA
ホスト変数の値に、SBCS データと DBCS データが両方とも入るように指示します。アプリケーション・リクエスターにおけるジョブの CCSID 属性が

DBCS 対応であり、ホスト変数の長さが 3 より大きい場合は、FOR MIXED DATA がデフォルト値となります。FOR DBCS DATA の CCSID は、アプリケーション・リクエスターにおけるジョブの CCSID 属性によって決まります。FOR MIXED DATA は、グラフィックまたは XML のホスト変数には指定できません。

CCSID 整数
ホスト変数の値に、CCSID 整数のデータが入ることを指定します。この整数が SBCS の CCSID である場合は、ホスト変数は SBCS データです。この整数が混合データの CCSID である場合は、ホスト変数は混合データです。文字ホスト変数の場合は、指定する CCSID は SBCS または混合 CCSID でなければなりません。

この変数がグラフィック・ストリング・データ・タイプのものである場合は、指定する CCSID は、DBCS、UTF-16、または UCS-2 の CCSID でなければなりません。有効な CCSID のリストについては、 1895 ページの『付録 E. CCSID の値』の項を参照してください。 UTF-16 または UCS-2 データを表すには、CCSID 1200 または 13488 を指定することを考慮してください。 CCSID が指定されていない場合は、グラフィック・ストリング変数の CCSID は、そのジョブに関連付けられている DBCS CCSID です。

この変数が XML データ・タイプのものである場合は、指定する CCSID は、SBCS、混合 CCSID、または Unicode CCSID でなければなりません。その CCSID は、XML AS データ・タイプとの互換性がある値でなければなりません。CCSID を指定しない場合は、SQL_XML_DATA_CCSID QAQQINI 設定で指定されている CCSID 値が使用されます。詳しくは、 101 ページの『XML 値』を参照してください。

ファイル参照変数の場合、CCSID はファイル内のデータではなく CCSID のパスおよびファイル名を指定します。

DATE
このホスト変数の値に、日付を示すデータが入ることを指定します。

TIME
このホスト変数の値に、時刻を示すデータが入ることを指定します。

TIMESTAMP
このホスト変数の値に、タイム・スタンプを示すデータが入ることを指定します。

注

配置制限: DECLARE VARIABLE ステートメントは、アプリケーションのうち、SQL ステートメントが有効であればどの位置にでも指定することができます。ただし、次のような例外があります。

- ホスト言語が COBOL または RPG の場合は、DECLARE VARIABLE ステートメントは、その DECLARE VARIABLE ステートメントで指定されるホスト変数を参照する SQL ステートメントより前でなければなりません。
- DATE、TIME、または TIMESTAMP を NULL 文字で終了する文字ストリングに対して C で指定すると、その C 宣言の長さは 1 だけ減らされます。

プリコンパイラー規則: 以下の場合は、プリコンパイル時にエラー・メッセージが出されます。

- 存在しないホスト変数を参照している。
- 数値変数が参照されている。
- 既に参照されている変数を参照している。
- 固有でないホスト変数が参照されている。
- UCS-2 として定義される ILE RPG 変数が参照されている。
- NATIONAL として定義される ILE COBOL 変数が参照されている。
- SQL ステートメントと DECLARE VARIABLE ステートメントが同一のホスト変数を参照しているときに、DECLARE VARIABLE ステートメントの方がその SQL ステートメントより後に置かれている。
- グラフィックまたは XML のホスト変数に関して、FOR BIT DATA、FOR SBCS DATA、または FOR MIXED DATA 文節が指定されている。
- グラフィックのホスト変数に関して、SBCS または混合の CCSID が指定されている。
- 文字ホスト変数に関して、DBCS、UTF-16、または UCS-2 の CCSID が指定されている。
- XML ホスト変数で DBCS CCSID が指定されている。
- DATE、TIME、または TIMESTAMP が文字でないホスト変数に対して指定されている。
- DATE、TIME、または TIMESTAMP に使用されるホスト変数の長さが不足していて、最小の日付、時刻、またはタイム・スタンプの値を指定することができない。

例

この例では、C プログラムの変数 *fred* および *pete* を混合データとして、また *jean* および *dave* を CCSID が 37 の SBCS データとして宣言しています。

```
void main ()
  {
     EXEC SQL BEGIN DECLARE SECTION;
     char fred[10];
     EXEC SQL DECLARE :fred VARIABLE  FOR MIXED DATA;

     decimal(6,0) mary;
     char pete[4];
     EXEC SQL DECLARE :pete VARIABLE FOR MIXED DATA;

     char jean[30];
     char dave[9];
     EXEC SQL DECLARE :jean, :dave VARIABLE  CCSID 37;
     EXEC SQL END DECLARE SECTION;
     EXEC SQL INCLUDE SQLCA;
   ...

  }
```

DELETE

DELETE ステートメントは、表またはビューから行を削除します。 INSTEAD OF DELETE トリガーが定義されていないビューから行を削除すると、そのビューの元になっている表から行が削除されます。そのようなトリガーが定義されていれば、代わりにこのトリガーが活動化されます。

このステートメントには、以下の 2 つの形式があります。

- 検索 DELETE 形式。この形式は、1 つ以上の行を削除する場合に使用します。必要に応じて、削除される行を検索条件によって限定することができます。
- 位置指定 DELETE 形式。この形式は、1 行だけ削除する場合に使用します。削除される行は、カーソルの現在位置によって決まります。

呼び出し

検索 DELETE ステートメントは、アプリケーション・プログラムに組み込めるほか、対話式に呼び出すこともできます。位置指定 DELETE ステートメントは、アプリケーション・プログラムに組み込んで使用しなければなりません。どちらの形式も、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメントに指定された表またはビューに対して、
  - その表またはビューに対する DELETE 特権
  - 表またはビューが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

検索 DELETE の検索条件 に、その表またはビューの列の参照が含まれている場合、そのステートメントの権限 ID によって保持される特権には、以下の 1 つも含まれていなければなりません。

- その表またはビューについての SELECT 特権
- データベース管理者権限

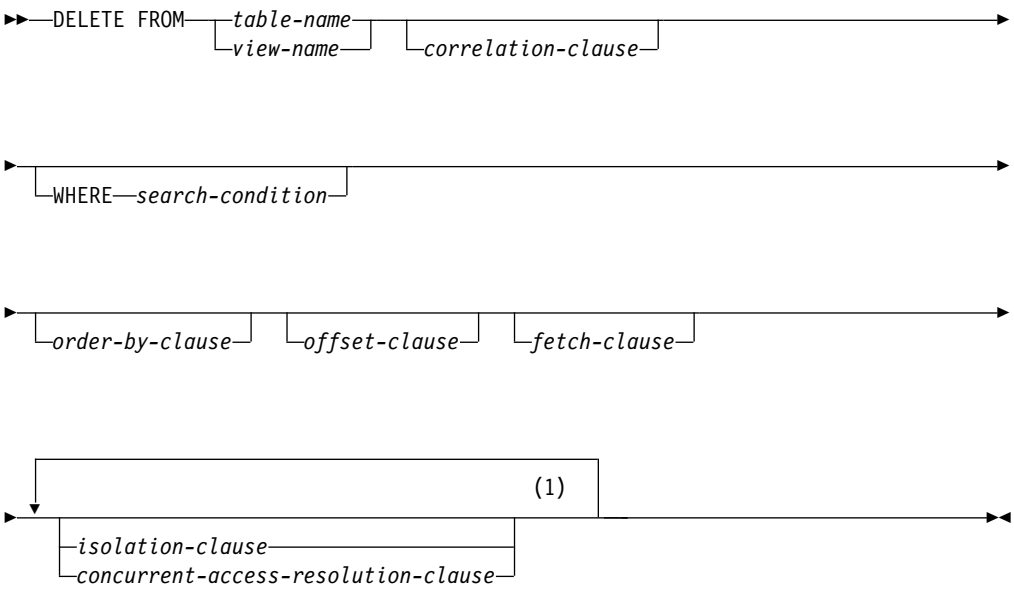
検索条件 に副照会が含まれている場合、ステートメントの権限 ID によって保持される特権には、少なくとも以下の 1 つも含まれていなければなりません。

- その副照会で識別されている各表またはビューについて、
  - 表やビューに対する SELECT 特権、および
  - 表またはビューが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

SQL 特権に対応するシステム権限については『表またはビューへの権限を検査する際の対応するシステム権限』を参照してください。

構文

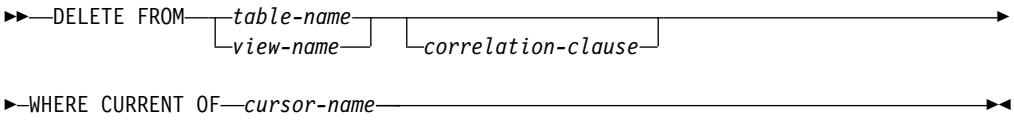
検索 **DELETE:**



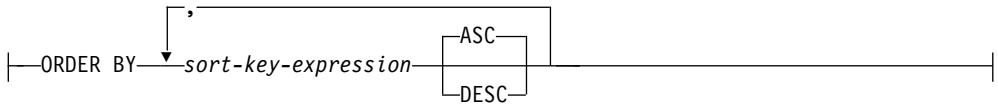
注:

1 同じ文節を複数回指定することはできません。

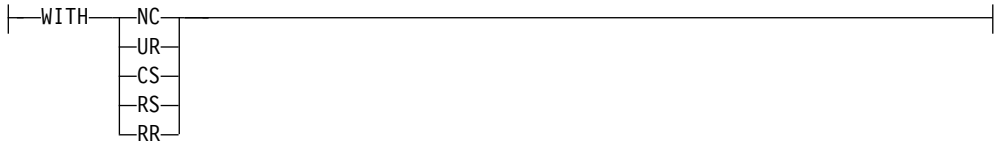
位置指定 **DELETE:**



ORDER BY 文節:



ISOLATION 文節:



説明

FROM *table-name* または *view-name*
行を削除する表またはビューを識別します。この名前は、現行サーバーに存在し

ている表またはビューを識別していなければなりませんが、カタログ表、カタログ表のビュー、または削除不可のビューを識別するものであってはなりません。削除可能なビューの説明については、 1342 ページの『CREATE VIEW』を参照してください。

correlation-clause
search-condition の中で表またはビューを指定するために使用できる代替名を指定します。相関文節 の説明については、 787 ページの『第 6 章 照会』を参照してください。相関名 の説明については、 163 ページの『相関名』を参照してください。

WHERE
削除する行を指定します。文節を省略するか、あるいは検索条件 またはカーソル名 を指定できます。この文節を指定しなかった場合は、指定した表またはビューのすべての行が削除されます。

search-condition
275 ページの『検索条件』で述べた検索条件です。副照会以外の検索条件で指定する列名 は、該当する表またはビューの列を識別するものでなければなりません。

検索条件 は、表またはビューの各行に適用されます。削除される行は、検索条件 の結果が真になった行です。

検索条件 に副照会が含まれている場合は、行に検索条件 が適用されるたびにその副照会が実行され、検索条件 を適用する際に結果が使用されるたびに、その副照会が実行されると考えることができます。実際に、相関参照のない副照会は一度しか実行されないことがありますが、相関参照を伴う副照会は各行ごとに一度実行する必要がある場合があります。

副照会が DELETE ステートメントのオブジェクト・テーブル、あるいは、CASCADE、SET NULL、または SET DEFAULT の削除規則を使用して従属表を参照する場合、その副照会は、行が削除される前に、完全に評価されます。

CURRENT OF *cursor-name*
削除操作で使用するカーソルを識別します。この *cursor-name* は、宣言されているカーソルを識別しなければなりません。カーソルの宣言については、DECLARE CURSOR ステートメントの Notes の項を参照してください。

識別された表またはビューは、カーソルの選択ステートメント の FROM 文節でも指定されなければならず、カーソルは削除可能でなければなりません。削除可能なカーソルの説明については、 1353 ページの『DECLARE CURSOR』を参照してください。

DECLARE CURSOR ステートメントには、DELETE ステートメントで使用される表またはビューの period-specification があってはなりません。

DELETE ステートメントが実行される時点では、カーソルはある 1 行 (削除される行) に位置付けられていなければなりません。行が削除されると、カーソルは、結果表にあるその次の行の前に位置付けられます。次の行が存在しない場合は、カーソルは最後の行の後に位置付けられます。

order-by-clause
offset-clause と *fetch-clause* を適用する行の順序を指定します。*offset-clause* と

fetch-clause に基づいて削除される行セットを決定するために予測どおりの順序になるように、*order-by-clause* を指定する必要があります。

sort-key-expression
削除操作の対象となる行の順序付けに使用される値を指定する式。 1 つの *sort-key-expression* を指定すると、行はその *sort-key-expression* の値の順に並べられます。複数の *sort-key-expression* を指定すると、行は最初の *sort-key-expression* の値の順に並べられ、次に 2 番目の *sort-key-expression* の値の順に、というように順に並べられます。

sort-key-expression の結果は、DATALINK または XML であってはなりません。

ASC
sort-key-expression の値を昇順に使用します。これはデフォルトです。

DESC
sort-key-expression の値を降順に使用します。

順序付けは、 57 ページの『第 2 章 言語エレメント』で説明した比較規則にしたがって行われます。 NULL 値は、他のすべての値より高位となります。ユーザーの順序付けの指定によって完全な順序が判別できない場合は、最後に指定された *sort-key-expression* の値が重複する行は、順序が不定になります。ORDER BY を指定しなければ、削除される行は任意の順序で並べられます。

offset-clause
限定した行のサブセットをスキップして、削除の対象範囲を制限します。*offset-clause* について詳しくは、 836 ページの『offset-clause』 を参照してください。

fetch-clause
限定した行のサブセットだけに削除の対象範囲を制限します。*fetch-clause* について詳しくは、 837 ページの『fetch-clause』 を参照してください。

isolation-clause
このステートメントに関して使用する分離レベルを指定します。

WITH

分離レベルを指定します。次のいずれかになります。
- *RR* 反復可能読み取り
- *RS* 読み取り固定
- *CS* カーソル固定
- *UR* 非コミット読み取り
- *NC* コミットなし

ISOLATION 文節 を指定しなかった場合は、デフォルトの分離レベルが使用されます。デフォルトの分離レベルについての詳細は、 859 ページの『ISOLATION 文節』を参照してください。

concurrent-access-resolution-clause
SELECT ステートメントで使用する並行アクセスの解決方法を指定します。詳しくは、 861 ページの『concurrent-access-resolution-clause』を参照してください。

DELETE の規則

トリガー:識別された表または識別されたビューの基本表が削除トリガーを持つ場合、トリガーが起動します。トリガーが起動された結果、他のステートメントが実行されたり、削除された値に基づいてエラー条件が返されたりすることがあります。

参照保全: 識別された表、または識別された表の基本表が親表である場合、選択された行は、RESTRICT または NO ACTION の削除規則との関係において従属関係を持ってはなりません。また、その DELETE は、RESTRICT または NO ACTION の削除規則との関係において従属関係を持つ下層行にカスケードしてはなりません。

削除操作が、RESTRICT または NO ACTION 削除規則によって防止されない場合には、選択された行は削除されます。選択された行に従属する行は、いずれも影響を受けます。

- SET NULL の削除規則に関連する行の従属行の外部キーのNULL 可能な列は、NULL 値に設定されます。
- CASCADE の削除規則に関連する行の従属行はいずれも削除され、それらの行についても上記の規則が適用されます。
- SET DEFAULT の削除規則に関連する行の従属行の外部キーの列は、対応するデフォルト値に設定されます。

参照制約 (RESTRICT 削除規則を伴う参照制約以外の) は、ステートメントの終わりで実際上チェックされます。複数行削除の場合、これは、すべての行が削除され、何らかの関連トリガーが起動された後で起こります。

検査制約: 検査制約を指定することにより、親表と従属表の関係について SET NULL または SET DEFAULT の削除規則が設定されている場合に、親表の行が削除されないようにすることができます。親表の行のどれかを削除したときに、従属表の中の列が NULL またはデフォルト値に設定されることになり、その NULL またはデフォルト値が原因で検査制約の検索条件の評価結果が偽になる場合は、その行は削除されません。

注

削除操作エラー: なんらかの削除操作を実行しているときにエラーが起きた場合は、このステートメントからの変更、参照制約、およびトリガーされた SQL ステートメントはロールバックされます (ただし、このステートメントまたは他のトリガーされた SQL ステートメントの分離レベルが NC である場合を除きます)。

ロック: 適切なロックが既に存在している場合を除き、正常な DELETE ステートメントの実行の過程で、1 つ以上の排他ロックが獲得されます。コミット操作またはロールバック操作によりロックが解除されるまでは、DELETE 操作の効果を認識できるのは以下のものだけです。

- 削除を行ったアプリケーション・プロセス
- 分離列 UR または NC を使用している他のアプリケーション・プロセス

ロックは、他のアプリケーション・プロセスがその表の操作を行うのを防止します。ロックの詳細については、COMMIT、ROLLBACK、および LOCK TABLE の各ステートメントの説明、および 32 ページの『分離レベル』を参照してください。

アプリケーション・プロセスが、その非更新可能なカーソルのいずれかが位置づけられている行を削除する場合、それらのカーソルはそれらの結果表の次の行の前に位置付けられます。次の行 R の前に位置付けられるカーソルが C であると想定します (OPEN、C を介した DELETE、他のカーソルを介した DELETE、または検索 DELETE の結果として)。 R の派生元である基本表に作用する INSERT、UPDATE、および DELETE 操作が発生すると、C を参照する次の FETCH 操作では、必ずしも、R 上に C を置くとは限りません。例えば、この操作で C を R' 上に置く可能性があります。この場合の R' とは、その時点で結果表の次の行になる新しい行です。

COMMIT(*RR)、COMMIT(*ALL)、COMMIT(*CS)、または COMMIT(*CHG) を指定している場合、1 つの DELETE ステートメントで削除または変更できる行の最大数は 4000000 です。変更された行の数には、トリガー、CASCADE、SET NULL、または SET DEFAULT 参照保全削除規則の結果として、同じコミットメント定義のもとで挿入、更新、または削除された行はいずれも含まれます。

カーソルの位置**:** アプリケーション・プロセスが、カーソルのいずれかが位置づけられている行を削除する場合、それらのカーソルはそれらの結果表の次の行の前に位置付けられます。行 R の前に位置付けられるカーソルが C であると想定します (OPEN、C を介した DELETE、他のカーソルを介した DELETE、または検索 DELETE の結果として)。R の派生元である基本表に作用する INSERT、UPDATE、および DELETE 操作が発生すると、C を参照する次の FETCH 操作では、必ずしも、R 上に C を置くとは限りません。例えば、この操作で C を R' 上に置く可能性があります。この場合の R' とは、その時点で結果表の次の行になる新しい行です。

削除された行の数**:** DELETE ステートメントが完了すると、削除された行の数が、SQL 診断域 (または SQLCA の SQLERRD(3)) の ROW_COUNT 条件領域項目に戻されます。 ROW_COUNT 項目の値には、CASCADE 削除規則またはトリガーの結果として削除された行の数は含まれません。

SQLCA についての説明は、 1867 ページの『付録 C. SQLCA (SQL 連絡域)』を参照してください。

DELETE パフォーマンス**:** WHERE 節、*offset-clause*、および *fetch-clause* を含まない SQL DELETE ステートメントは、表のすべての行を削除します。この場合、消去操作 (コミットメント制御下で実行していない場合) またはファイル変更操作 (コミットメント制御下で実行している場合) を使用して行を削除することができます。コミットメント制御下で実行している場合、削除をそのままコミットするかロールバックすることができます。このインプリメンテーションは、個別に各行を削除するよりもより高速ですが、各行の個別のジャーナル項目はジャーナルに記録されません。この技法を使用するのは、以下のすべてが当てはまる場合のみです。

- ターゲット表はビューではありません。
- ターゲット表はシステム期間テンポラル表ではありません。

- かなりの行数を削除中です。
- DELETE ステートメントを発行するジョブには、ファイルにオープン・カーソルがありません (疑似クローズされた SQL カーソルを含まない)。
- 他のジョブで表に対してロックをかけていません。
- 表は、アクティブな削除トリガーを持っていません。
- 表は、CASCADE、SET NULL、または SET DEFAULT 削除規則を使用した参照制約内の親ではありません。
- DELETE ステートメントを発行するユーザーは、 DELETE 特権に加えて表の *OBJMGT または *OBJALTER システム権限を持っています。
- SQL_FAST_DELETE_ROW_COUNT QAQQINI オプションは高速削除を許可します。

この方法が正常に完了すると、増分数 (CHGPF CL コマンドの SIZE キーワードを参照) がゼロに設定されます。

表からすべての行を削除するには、TRUNCATE ステートメントを使用できます。

参照保全に関する考慮事項**:** SQL 診断域 (または SQLCA の SQLERRD(5)) の DB2_ROW_COUNT_SECONDARY 条件情報項目は、参照制約によって影響を受けた行の数を示します。この値には、CASCADE 削除規則の結果として削除された行の数、および SET NULL または SET DEFAULT 削除規則の結果として、その外部キーが NULL またはデフォルト値に設定された行の数が含まれます。

DB2_ROW_COUNT_SECONDARY についての説明は、 1489 ページの『GET DIAGNOSTICS』を参照してください。SQLCA についての説明は、 1867 ページの『付録 C. SQLCA (SQL 連絡域)』を参照してください。

行アクセス制御が適用されている表の行の削除**:** 行アクセス制御が適用されている表に対して DELETE ステートメントが発行されると、有効な行の許可に指定されている規則によって、行が削除可能かどうかが判定されます。通常、これらの規則は、ステートメントの許可 ID に基づきます。以下に、有効になっている行の許可および列マスクが DELETE 時にどのように使用されるかについて説明します。

- 行の許可を使用して、削除する行のセットが特定されます。

  1 つの表に対して、有効な行の許可が複数定義されている場合は、有効になっている各許可の検索条件に論理 OR 演算子を適用することにより、行アクセス制御検索条件が導出されます。この行アクセス制御検索条件が表に適用されて、DELETE ステートメントの許可 ID でアクセス可能な行が決定されます。DELETE ステートメントに WHERE 節が指定されている場合、アクセス可能な行にユーザー指定の述部が適用されて、削除する行が決定されます。WHERE 節がない場合、削除する行は、すべてのアクセス可能な行です。

システム期間テンポラル表に関する考慮事項**:** DELETE ステートメントに履歴表を参照する (履歴表の名前の明示的な参照、または FROM 節での期間指定の使用による暗黙的な参照) 相関副照会が含まれる検索条件がある場合、履歴行として格納された削除行がそれらの行の削除操作で参照され、その後ステートメントで処理される可能性があります。

CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターが NULL 以外の値に設定され、SYSTIME オプションの値が YES の場合は、DELETE ステートメントの基礎ターゲットを、システム期間テンポラル表にしてはなりません。この制限は、システム期間テンポラル表への参照が直接か間接かにかかわらず適用されます。

履歴表に関する考慮事項**:** システム期間テンポラル表の 1 つの行が削除されるとき、その行の履歴コピーが対応する履歴表に挿入され、履歴行の終了タイム・スタンプが、データ変更操作の時刻に対応するシステム判別値の形式でキャプチャーされます。データベース・マネージャーが割り当てるこの値は、次のいずれかの場合に時刻機構を読み取ることによって生成されます。(1) トランザクションの中で、表に含まれる行開始列またはトランザクション開始 ID 列に値を割り当てる必要があるようなデータ変更ステートメントを最初に実行するとき。(2) システム期間テンポラル表に含まれる行を削除するとき。終了列の値は、トランザクション全体の履歴表で固有になるようにデータベース・マネージャーによって生成されます。履歴表に挿入される行の終了タイム・スタンプ値が開始タイム・スタンプ値より大きくなるように、タイム・スタンプ値が調整される可能性があります。これは、競合するトランザクションがシステム期間テンポラル表の同じ行を更新しているときに行われる場合があります。このタイム・スタンプ値の調整を行うには、SYSTIME_PERIOD_ADJ QAQQINI オプションを *ADJUST に設定する必要があります。存在しない場合は、エラーが戻されます。

削除操作では、この調整は、関連付けられたシステム期間テンポラル表の行終了列に対応する、履歴表の終了列の値にのみ影響があります。この表への以降の参照では、関連付けられたシステム期間テンポラル表の期間 SYSTEM_TIME の行開始列および行終了列においてトランザクション開始時刻が検索される場合に、上記の調整を考慮に入れてください。

履歴表が ON DELETE ADD EXTRA ROW で定義された場合、削除が実行される前の行のバージョンが履歴表に追加されます。この行が追加される際に、行開始列、行終了列、および生成式列の値が生成されます。この情報は、行の削除時点を表します。

REXX: 変数は、REXX プロシージャー内の DELETE ステートメントでは使用できません。その代わりとして、DELETE は、パラメーター・マーカーを使用して PREPARE と EXECUTE のオブジェクトにする必要があります。

代替構文**:** 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード NONE を NC の同義語として使用することができます。
- キーワード CHG を UR の同義語として使用することができます。
- キーワード ALL を RS の同義語として使用することができます。
- FROM キーワードはオプションです。

例

例 1: 表 DEPARTMENT から、部門 (DEPTNO) 'D11' を削除します。

```
DELETE FROM DEPARTMENT
  WHERE DEPTNO = 'D11'
```

例 *2:* 表 DEPARTMENT から、すべての部門を削除します (つまり、表を空にします)。

```
DELETE FROM DEPARTMENT
```

例 *3:* Java プログラム・ステートメントを使用して、接続コンテキスト「ctx」上の PROJECT 表から、ホスト変数 HOSTDEPT の部門 (java.lang.String) と等しい部門 (DEPTNO) のすべてのサブプロジェクト (MAJPROJ が NULL のもの) を削除します。

```
#sql [ctx] { DELETE FROM PROJECT
               WHERE DEPTNO = :HOSTDEPT
               AND MAJPROJ IS NULL };
```

例 *4:* 以下の例に示す Java プログラム (一部分) は、退職した社員 (JOB) を表示し、要求があれば、接続コンテキスト 'ctx' 上にある EMPLOYEE 表から特定の社員を削除します。

```
#sql iterator empIterator implements sqlj.runtime.ForUpdate
     ( ... );
empIterator C1;

#sql [ctx] C1 = { SELECT * FROM EMPLOYEE
                    WHERE JOB = 'RETIRED' };

#sql {  FETCH C1 INTO ...     };
while ( !C1.endFetch() )  {
   System.out.println( ... );
               ...
   if ( condition for deleting row ) {
       #sql [ctx] { DELETE FROM EMPLOYEE
                      WHERE CURRENT OF C1  };
   }
   #sql { FETCH C1 INTO ...     };
}
C1.close();
```

DESCRIBE

DESCRIBE ステートメントは、準備済みステートメントに関する情報の入手に使用します。

準備済みステートメントの説明については、 1603 ページの『PREPARE』を参照してください。

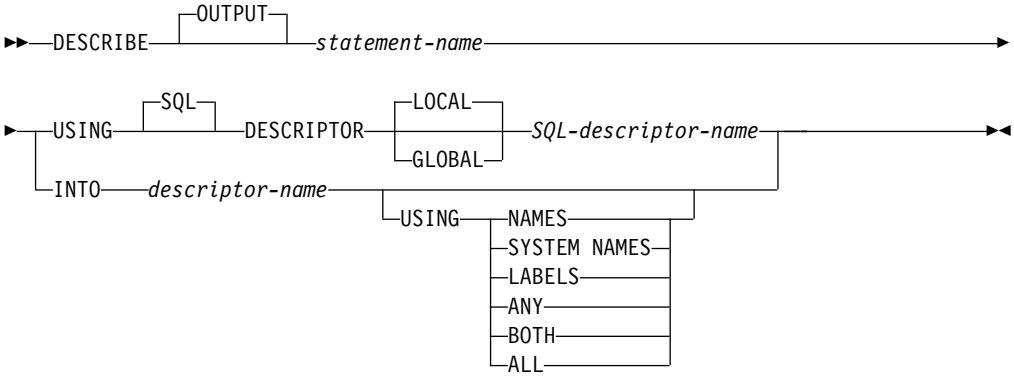
呼び出し

このステートメントは、アプリケーション・プログラム、SQL 関数、 SQL プロシージャー、またはトリガー内にのみ組み込むことができます。これは実行可能ステートメントですが、動的に準備することはできません。Java では指定できません。

権限

権限は不要です。準備済みステートメントを作成するために必要な権限については、 1603 ページの『PREPARE』を参照してください。

構文



説明

statement-name
準備済みステートメントを識別します。DESCRIBE ステートメントが実行される時点で、この名前はアプリケーション・サーバー側で準備済みステートメントを識別していなければなりません。

準備済みステートメントが全選択または VALUES INTO ステートメントの場合は、結果表の列の記述が情報として返されます。準備済みステートメントが CALL ステートメントの場合は、プロシージャーの OUT および INOUT パラメーターの記述が情報として戻されます。

USING
SQL 記述子を識別します。

LOCAL
記述子の名前の有効範囲はプログラム呼び出しのローカルであることを指定します。

GLOBAL
記述子の名前の有効範囲は SQL セッション全体であることを指定します。

SQL-descriptor-name
SQL 記述子の名前を指定します。名前は、指定の有効範囲を持つ既存の記述子を識別するものでなければなりません。

SQL 記述子に入る情報の説明については、 1476 ページの『GET DESCRIPTOR』を参照してください。

INTO *descriptor-name*
SQL 記述子域 (SQLDA) を指定します。これについては、 1877 ページの『付録 D. SQLDA (SQL 記述子域)』で説明しています。DESCRIBE ステートメントを実行する前に、SQLDA の以下の変数をセットしておく必要があります。

SQLN
SQLDA で用意される SQLVAR 項目の数を指定します。 DESCRIBE ステートメントを実行する前に、SQLN にゼロ以上の値をセットしておく必要があります。必要なオカレンスの数を決定する手法については、 1880 ページの『必要な SQLVAR オカレンスの数の決定』を参照してください。

REXX の場合は、規則が異なります。詳しくは、「組み込み SQL プログラミング」のトピック集を参照してください。

DESCRIBE ステートメントが実行されると、データベース・マネージャーでは、SQLDA の各変数に次のような値を割り当てます。

SQLDAID
最初の 6 バイトは 'SQLDA ' (つまり、5 文字の後にスペース文字) に設定されます。

7 番目のバイトは、記述された結果列に基づいて設定されます。

- SQLDA の各項目 (または、結果表の列) に 2 つ、3 つ、または 4 つの SQLVAR 項目が入っている場合、この 7 番目のバイトはそれぞれ '2'、'3'、または '4' に設定されます。この技法は、LOB または特殊タイプ結果列、ラベル、およびシステム名に対応するために使用されています。
- それ以外の場合、7 番目のバイトはスペース文字に設定されます。

SQLDA 内にすべての結果列の記述を含める余地がない場合、7 番目のバイトはスペース文字に設定されます。

8 番目のバイトはスペース文字に設定されます。

SQLDABC
SQLDA の長さ (バイト)。

SQLD
準備済みステートメントが SELECT の場合、SQLD は、その結果表の列の数に拡張 SQLVAR 項目の数を足したものに設定されます。拡張 SQLVAR 項目については、 1882 ページの『SQLVAR のオカレンスのフィールドの説明』を参照してください。準備済みステートメントが CALL ステートメントの場合、SQLD はプロシージャーの OUT および INOUT パラメーターの数に設定されます。準備済みステートメントが VALUES INTO の場

合、SQLD は VALUES 節にある式の数に拡張 SQLVAR 項目の数を加えたものに設定されます。 これら以外の場合、SQLD は 0 に設定されます。

SQLVAR

SQLD の値が 0 の場合、または SQLD の値が SQLN の値より大きい場合は、SQLVAR のオカレンスには値が割り当てられません。

SQLD の値が *n* (*n* は 0 より大きいが、SQLN の値以下) の場合、SQLVAR の最初から *n* 番目までのオカレンスには、 SQLVAR の最初のオカレンスに結果表の最初の列 (または、パラメーターまたは VALUES 節内の式) の記述が入り、SQLVAR の 2 番目のオカレンスに結果表の 2 番目の列 (または、パラメーターまたは VALUES 節内の式) の記述が入り、以下同様に、値が割り当てられます。SQLVAR オカレンスに割り当てられる値については、 1882 ページの『SQLVAR のオカレンスのフィールドの説明』を参照してください。

USING

SQLDA のそれぞれの SQLNAME 変数に、どのような値を割り当てるかを指定します。要求した値が存在しない場合、または名前の長さが 30 より大きい場合は、SQLNAME の長さは 0 にセットされます。

NAMES

列 (またはパラメーター) の名前を割り当てます。これはデフォルトです。選択リストに名前が明示的にリストされている準備済みステートメントについての DESCRIBE の場合、指定されたその名前が戻されます。戻される列名は大/小文字の区別があり、区切り文字はありません。

SYSTEM NAMES

列のシステム列名を割り当てます。

LABELS

列のラベルを割り当てます。(列のラベルは、LABEL ステートメントによって定義されます。) ラベルの最初の 20 バイトだけが戻されます。

ANY

列のラベルを割り当てます。列にラベルがない場合は、代わりに列の名前が割り当てられます。

BOTH

列のラベルと名前の両方を割り当てます。この場合、追加情報に応じるために、1 つの列ごとに SQLVAR の 2 つから 3 つのオカレンスが必要になりますが、その数は、結果セットに特殊タイプが入っているか否かによって決まります。この拡張の SQLVAR 配列を指定するには、SQLN を 2*n* か 3*n* (この場合の *n* は、表やビュー内の列数) に設定します。 SQLVAR の最初の *n* 個のオカレンスには、列の名前が入り、 2 番目または 3 番目の *n* オカレンスには、列のラベルが含まれます。 特殊タイプがない場合、SQLVAR 項目の 2 番目のセットにそのラベルが戻されます。それ以外の場合、ラベルは、SQLVAR 項目の 3 番目のセット内に戻されます。

ALL

ラベル、列名、およびシステム列名を割り当てます。この場合、追加情報に応じるために、 1 つの列ごとに SQLVAR の 3 つから 4 つのオカレンスが必要になりますが、その数は、結果セットに特殊タイプが入っているか否かによって決まります。この拡張の SQLVAR 配列を指定するには、SQLN

を 3*n* か 4*n* (この場合の *n* は、結果表内の列数) に設定します。SQLVAR の最初の *n* オカレンスには、システム列名が入ります。 2 番目または 3 番目の *n* オカレンスには、列のラベルが含まれます。列名がシステム列名とは異なる場合、列名は 3 番目または 4 番目の *n* オカレンスに含まれます。その他の場合、SQLNAME フィールドは長さゼロに設定されます。特殊タイプが指定されていない場合、ラベルは、SQLVAR 記入項目の 2 番目のセット内に戻され、列名は、SQLVAR 記入項目の 3 番目のセット内に戻されます。それ以外の場合、ラベルは、SQLVAR 記入項目の 3 番目のセット内に戻され、列名は、 SQLVAR 記入項目の 4 番目のセット内に戻されます。

注

PREPARE INTO: 準備済みステートメントに関する情報は、PREPARE ステートメントの INTO 文節を使用しても入手することができます。

SQL 記述子の割り振り**:** DESCRIBE ステートメントを実行する前に、ALLOCATE DESCRIPTOR ステートメントを使用して SQL 記述子を割り振らなければなりません。割り振られた記述子項目の数が結果列の数よりも小さい場合、警告 (SQLSTATE 01005) が戻されます。

SQLDA の割り振り**:** C、COBOL、PL/I、および RPG では、 DESCRIBE または PREPARE INTO ステートメントが実行される前に、十分なストレージをいくつかの SQLVAR オカレンスに割り当てる必要があります。SQLN は、割り当てられた SQLVAR オカレンスの数に設定されなければなりません。準備済み SELECT ステートメントの結果表にある列の記述を入手する場合は、 SQLVAR 項目のオカレンスの数を、結果表にある列の数以上にしなければなりません。さらに、列に LOB や特殊タイプを指定している場合は、SQLVAR 項目のオカレンス数として、列数の 2 倍の数値を指定する必要があります。詳しくは、 1880 ページの『必要な SQLVAR オカレンスの数の決定』を参照してください。 SQLDA を割り振るための可能な方法としては、以下の 3 通りがあります。

最初の技法
: アプリケーションで処理する必要があるどの選択リストにも対応できるように、SQLVAR 項目のオカレンス数を十分にとって SQLDA を割り振ります。極端なことをいえば、SQLVAR の数を、結果表で許容されている列の最大数の 2 倍にすることも可能です。いったん割り振りが終了すれば、アプリケーションでは、この SQLDA を繰り返し使用することができます。

  この方法は、大量の記憶域を使用します。また、ある特定の選択リストで、その記憶域のごく一部しか使用しないとしても、記憶域の割り振りが解除されることはありません。

2 番目の技法
: 選択リストを処理するたびに、以下の 3 つのステップを繰り返し実行します。

  1. SQLVAR 項目のオカレンスがない SQLDA、つまり SQLN がゼロの SQLDA を使用して、 DESCRIBE ステートメントを実行します。

SQLD に戻される値は、SQLVAR 項目のオカレンスの必要数か、結果列の数です。 SQLVAR 項目がないため、警告が出されます。[102]

2. SQLDAID フィールドの第 7 バイトがブランクでない場合は、SQLDA を (SQLDAID の第 7 バイトの値) * SQLD オカレンス数で割り振り、新規の SQLDA の SQLN を (SQLDAID の第 7 バイトの値) * SQLD に設定します。そうでない場合、SQLDA を SQLD オカレンスで割り振り、新規の SQLDA の SQLN を SQLD の値に設定します。
3. 次に、この新しい SQLDA を使用して、再び DESCRIBE ステートメントを実行します。

この方法をとると、方法 1 より記憶域管理が向上しますが、DESCRIBE ステートメントの数が 2 倍になります。

3 番目の技法

選択リストの大半 (ほとんど全部) を扱うのに十分な範囲で、なるべく小さな SQLDA を割り振ります。 SQLDA が小さ過ぎることが原因で、DESCRIBE ステートメントの実行に失敗した場合は、より大きな SQLDA を割り振って、再び DESCRIBE ステートメントを実行します。新しい SQLDA では、最初の DESCRIBE の実行で SQLD に戻された値 (または SQLD の 2 倍の値) を使用して、 SQLVAR 項目のオカレンス数を指定してください。

この方法は、方法 1 と方法 2 を組み合わせたものです。方法 3 の効果は、最初の SQLDA のサイズを適切に選定できるかどうかによって左右されます。

暗黙的な非表示列に関する考慮事項**:** DESCRIBE OUTPUT ステートメントから暗黙的な非表示列に関する情報が返されるのは、照会を記述するときに、照会の最終的な結果表の SELECT リストの一部として、暗黙的な非表示列と定義されている基本表の列を組み込むことを明示的に指定した場合に限られます。暗黙的な非表示列が照会の結果表の一部でない場合、その照会に関する情報を戻す DESCRIBE OUTPUT ステートメントにはどの暗黙的な非表示列に関する情報も含まれていません。

例

C プログラムの中で、SQLVAR 項目のオカレンスなしの SQLDA を使用して DESCRIBE ステートメントを実行します。 SQLD がゼロより大きければ、その値を使用して、SQLVAR 項目に必要なオカレンス数を指定した SQLDA を割り振ります。その後で、新しい SQLDA を使用して DESCRIBE ステートメントを実行します。

```
EXEC SQL  BEGIN DECLARE SECTION;
  char stmt1_str [200];
EXEC SQL  END DECLARE SECTION;
EXEC SQL  INCLUDE SQLDA;
struct sqlda initialsqlda;
struct sqlda *sqldaPtr;

EXEC SQL  DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;
```

102. LOB または UDT が結果セットに入っていない場合、この警告が戻されるのは、標準オプションが指定されている場合のみです。標準オプションについては、 xi ページの『標準への準拠』を参照してください。

```
... /* code to prompt user for a query, then to generate    */
    /* a select-statement in the stmt1_str                  */
EXEC SQL  PREPARE STMT1_NAME FROM :stmt1_str;

... /* code to set SQLN to zero and to allocate the SQLDA   */
EXEC SQL  DESCRIBE STMT1_NAME INTO :initialsqlda;

if (initialsqlda.sqld == 0);  /* statement is a select-statement */
  {
   ... /* Code to allocate correct size SQLDA (sets sqldaPtr) */

   if (strcmp(SQLSTATE,"01005") == 0)
     {
      sqldaPtr->sqln = 2*initialsqlda.sqld;
      SETSQLDOUBLED(sqldaPtr, SQLDOUBLED);
     }
   else
     {
      sqldaPtr->sqln = initialsqlda.sqld;
      SETSQLDOUBLED(sqldaPtr, SQLSINGLED);
     }
   EXEC SQL  DESCRIBE STMT1_NAME INTO :*sqldaPtr;

   ... /* code to prepare for the use of the SQLDA             */
   EXEC SQL  OPEN DYN_CURSOR;

   ... /* loop to fetch rows from result table                 */
   EXEC SQL  FETCH DYN_CURSOR USING DESCRIPTOR :*sqldaPtr;

   ...
  }
...
```

DESCRIBE CURSOR

DESCRIBE CURSOR ステートメントは、カーソルに関する情報を取得します。列情報などの情報は、記述子に入れられます。

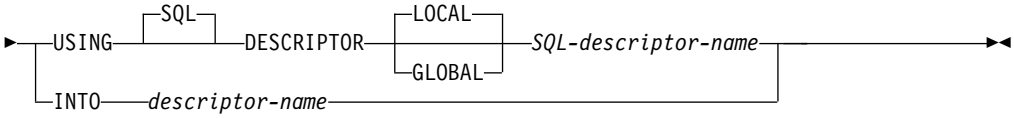
呼び出し

このステートメントは、アプリケーション・プログラム、SQL 関数、 SQL プロシージャー、またはトリガー内にのみ組み込むことができます。これは実行可能ステートメントですが、動的に準備することはできません。Java および REXX では指定できません。

権限

権限は不要です。

構文



説明

cursor-name **or** *variable*

ソース・プログラムで既にオープンしているカーソルまたは割り振られているカーソルを指定します。

変数 を指定する場合、

- その変数は、文字ストリング変数またはユニコード・グラフィック・ストリングでなければなりません。この変数は、グローバル変数にすることはできません。
- 標識変数を伴っていてはなりません。
- 変数内に含まれるカーソル名は左寄せでなければならず、その長さが変数の長さより短い場合は、右側にブランクを埋め込まなければなりません。
- カーソルの名前は、区切り文字付きの名前でない限り、大文字でなければなりません。

USING

SQL 記述子を識別します。

LOCAL

記述子の名前の有効範囲はプログラム呼び出しのローカルであることを指定します。

GLOBAL

記述子の名前の有効範囲は SQL セッション全体であることを指定します。

SQL-descriptor-name
SQL 記述子の名前を指定します。名前は、指定の有効範囲を持つ既存の記述子を識別するものでなければなりません。

記述子域に戻される情報は、指定されたカーソルと関連する結果セット内の列を説明します。DESCRIBE CURSOR の実行後の記述子域の内容は、SELECT の DESCRIBE の実行後の内容と同じです。ただし、以下の内容が追加されます。

- GET DESCRIPTOR ステートメントから、プロシージャーでカーソルが WITH HOLD で宣言されているかどうかを示す標識として DB2_CURSOR_HOLD を返すことができます。

SQL 記述子に入る情報の説明については、 1476 ページの『GET DESCRIPTOR』を参照してください。

INTO *descriptor-name*
SQL 記述子域 (SQLDA) を指定します。これについては、 1877 ページの『付録 D. SQLDA (SQL 記述子域)』で説明しています。DESCRIBE CURSOR ステートメントを実行する前に、SQLDA の以下の変数をセットしておく必要があります。

SQLN
SQLDA で用意される SQLVAR オカレンスの数を指定します。DESCRIBE CURSOR ステートメントを実行する前に、SQLN にゼロ以上の値をセットしておく必要があります。必要なオカレンスの数を決定する手法については、 1880 ページの『必要な SQLVAR オカレンスの数の決定』を参照してください。

DESCRIBE ステートメントが実行されると、データベース・マネージャーでは、SQLDA の各変数に次のような値を割り当てます。

SQLDAID
最初の 5 バイトは「SQLRS」に設定される。6 から 8 バイトは予約されている。プロシージャー内でカーソルが WITH HOLD に宣言されると、8 バイト目の高位ビットは 1 に設定される。

SQLDABC
SQLDA の長さ (バイト)。

SQLD
結果表内の列数に拡張 SQLVAR 項目数を加えたものです。拡張 SQLVAR 項目については、 1882 ページの『SQLVAR のオカレンスのフィールドの説明』を参照してください。

SQLVAR
SQLD の値が 0 の場合、または SQLD の値が SQLN の値より大きい場合は、SQLVAR のオカレンスには値が割り当てられません。

SQLD の値が *n* (ただし、*n* は 0 より大きいが、SQLN の値より小さいか、または等しい値) の場合は、値が SQLVAR の最初の *n* 個のオカレンスに割り当てられ、その結果、SQLVAR の最初のオカレンスには結果表の最初の列の記述が入り、SQLVAR の 2 番目のオカレンスには結果表の 2 番目の列の記述が入ります。以下同様です。SQLVAR オ

カレンスに割り当てられる値については、 1882 ページの『SQLVAR のオカレンスのフィールドの説明』を参照してください。

注

SQL 記述子の割り振り**:** DESCRIBE CURSOR ステートメントを実行する前に、ALLOCATE DESCRIPTOR ステートメントを使用して SQL 記述子を割り振らなければなりません。割り振られた記述子項目の数がカーソル結果セットの列の数より小さい場合は、警告 (SQLSTATE 01005) が戻されます。

SQLDA の割り振り**:** DESCRIBE CURSOR ステートメントを実行する前に、SQLN にゼロ以上の値をセットして、SQLDA に用意する SQLVAR のオカレンスの数を指示するとともに、SQLN の各オカレンスを収容するのに十分な記憶域を割り振っておく必要があります。カーソル結果セットの列の記述を取得するには、SQLVAR のオカレンスの数が列の数以上でなければなりません。

SQLDA の割り振りのために使用する方法については、 1877 ページの『付録 D. SQLDA (SQL 記述子域)』を参照してください。

例

カーソル C1 に関連する結果セットについての情報を記述子に入れます。

```
EXEC SQL ALLOCATE DESCRIPTOR 'DESCR1';
EXEC SQL DESCRIBE CURSOR C1 USING SQL DESCRIPTOR 'DESCR1';
```

DESCRIBE INPUT

DESCRIBE INPUT ステートメントは、準備済みステートメントの IN および INOUT パラメーター・マーカーに関する情報を取得します。

準備済みステートメントの説明については、 1603 ページの『PREPARE』を参照してください。

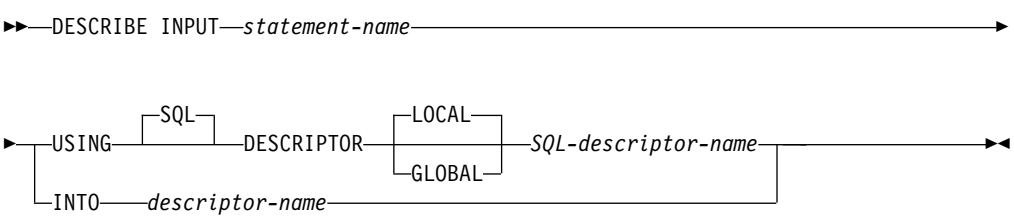
呼び出し

このステートメントは、アプリケーション・プログラム、SQL 関数、 SQL プロシージャー、またはトリガー内にのみ組み込むことができます。これは実行可能ステートメントですが、動的に準備することはできません。Java および REXX では指定できません。

権限

権限は不要です。準備済みステートメントを作成するために必要な権限については、 1603 ページの『PREPARE』を参照してください。

構文



説明

statement-name
準備済みステートメントを識別します。DESCRIBE INPUT ステートメントが実行される時点で、この名前は、現行サーバー側で準備済みステートメントを識別していなければなりません。

USING
SQL 記述子を識別します。

LOCAL
記述子の名前の有効範囲はプログラム呼び出しのローカルであることを指定します。

GLOBAL
記述子の名前の有効範囲は SQL セッション全体であることを指定します。

SQL-descriptor-name
SQL 記述子の名前を指定します。名前は、指定の有効範囲を持つ既存の記述子を識別するものでなければなりません。

SQL 記述子に入る情報の説明については、 1476 ページの『GET DESCRIPTOR』を参照してください。

INTO *descriptor-name*

SQL 記述子域 (SQLDA) を指定します。これについては、 1877 ページの『付録 D. SQLDA (SQL 記述子域)』で説明しています。DESCRIBE INPUT ステートメントを実行する前に、SQLDA の以下の変数をセットしておく必要があります。

SQLN

SQLDA で用意される SQLVAR オカレンスの数を指定します。DESCRIBE INPUT ステートメントを実行する前に、SQLN をゼロ以上の値にセットしておく必要があります。必要なオカレンスの数を決定する手法については、 1880 ページの『必要な SQLVAR オカレンスの数の決定』を参照してください。

DESCRIBE INPUT ステートメントが実行されると、データベース・マネージャーでは、SQLDA の各変数に次のような値を割り当てます。

SQLDAID

最初の 6 バイトは 'SQLDA ' (つまり、5 文字の後にスペース文字) に設定されます。

7 番目のバイトは、記述されたパラメーター・マーカーに基づいて次のように設定されます。

- SQLDA に入力パラメーター・マーカーごとに 2 つの SQLVAR 項目が含まれる場合、7 番目のバイトは '2' に設定されます。この技法は、LOB 入力パラメーターに対応するために使用されています。
- それ以外の場合、7 番目のバイトはスペース文字に設定されます。

SQLDA 内にすべての入力パラメーター・マーカーの記述を含める余地がない場合、7 番目のバイトはスペース文字に設定されます。

8 番目のバイトはスペース文字に設定されます。

SQLDABC

SQLDA の長さ (バイト)。

SQLD

準備済みステートメント内の入力パラメーター・マーカーの数。

SQLVAR

SQLD の値が 0 の場合、または SQLD の値が SQLN の値より大きい場合は、SQLVAR のオカレンスには値が割り当てられません。

SQLD の値が *n* (ただし、*n* は 0 より大きいが、SQLN の値より小さいか、または等しい値) の場合は、値が SQLVAR の最初の *n* 個のオカレンスに割り当てられ、その結果、SQLVAR の最初のオカレンスには最初の入力パラメーター・マーカーの記述が入り、SQLVAR の 2 番目のオカレンスには 2 番目の入力パラメーター・マーカーの記述が入ります。以下同様です。SQLVAR オカレンスに割り当てられる値については、 1882 ページの『SQLVAR のオカレンスのフィールドの説明』を参照してください。

注

SQL 記述子の割り振り: DESCRIBE INPUT ステートメントを実行する前に、ALLOCATE DESCRIPTOR ステートメントを使用して SQL 記述子を割り振らなけ

ればなりません。割り振る記述子項目の数は、入力パラメーター・マーカーの数以上でなければなりません。そうしないとエラーが戻されます。

SQLDA の割り振り**:** DESCRIBE INPUT ステートメントを実行する前に、SQLVAR の出現回数を想定して十分な量のストレージを割り振っておく必要があります。SQLN は、割り当てられた SQLVAR オカレンスの数に設定されなければなりません。準備済みステートメント内の入力パラメーター・マーカーの記述を取得するためには、SQLVAR のオカレンスの数が入力パラメーター・マーカーの数以上でなければなりません。さらに、入力パラメーター・マーカーに LOB または特殊タイプが含まれる場合は、SQLVAR のオカレンス数を入力パラメーター・マーカーの数の 2 倍にする必要があります。詳しくは、 1880 ページの『必要な SQLVAR オカレンスの数の決定』を参照してください。

提供されるオカレンスが不足していて、オカレンスのすべてのセットを戻せるとは限らない場合、SQLN は、すべての情報を戻すのに必要なオカレンスの合計数に設定されます。それ以外の場合、SQLN は、入力パラメーター・マーカーの数に設定されます。

SQLDA を割り振るための可能な方法としては、以下の 3 通りがあります。

最初の技法
: アプリケーションで処理する必要があるどの入力パラメーター・マーカーの数にも対応できるように、SQLVAR 項目のオカレンス数を十分にとって SQLDA を割り振ります。極端なことをいえば、SQLVAR の数を、準備済みステートメントで許容されているパラメーター・マーカーの最大数の 2 倍にすることも可能です。いったん割り振りが終了すれば、アプリケーションでは、この SQLDA を繰り返し使用することができます。

  この方法は、大量の記憶域を使用します。また、ある特定の準備済みステートメントで、その記憶域のごく一部しか使用しないとしても、記憶域の割り振りが解除されることはありません。

2 番目の技法
: 準備済みステートメントを処理するたびに、以下の 3 つのステップを繰り返し実行します。

  1. SQLVAR 項目のオカレンスがない SQLDA、つまり SQLN がゼロの SQLDA を使用して、DESCRIBE INPUT ステートメントを実行します。SQLD で返される値は、準備済みステートメントの入力パラメーター・マーカーの数です。その値は、SQLVAR 項目の必要出現回数か、その回数の半分になります。SQLVAR 項目がないので、警告が出されます。[103]
  2. その警告を伴う SQLSTATE が 01005 と等しい場合は、SQLDA に 2 * SQLD の出現回数を割り振り、その新しい SQLDA の SQLN を 2 * SQLD に設定します。そうでない場合、SQLDA を SQLD オカレンスで割り振り、新規の SQLDA の SQLN を SQLD の値に設定します。
  3. 次に、この新しい SQLDA を使用して、再び DESCRIBE INPUT ステートメントを実行します。

---

103. LOB または UDT が結果セットに入っていない場合、この警告が戻されるのは、標準オプションが指定されている場合のみです。標準オプションについては、 xi ページの『標準への準拠』を参照してください。

この方法をとると、方法 1 より記憶域管理が向上しますが、DESCRIBE INPUT ステートメントの数が 2 倍になります。

3 番目の技法

準備済みステートメント内のパラメーター・マーカーの大半 (ほとんど全部) を扱うのに十分な範囲で、なるべく小さな SQLDA を割り振ります。SQLDA が小さ過ぎることが原因で、DESCRIBE INPUT ステートメントの実行に失敗した場合は、より大きな SQLDA を割り振って、再び DESCRIBE INPUT ステートメントを実行します。新しい SQLDA では、最初の DESCRIBE INPUT の実行で SQLD に戻された値 (または SQLD の 2 倍の値) を使用して、SQLVAR 項目のオカレンス数を指定してください。

この方法は、方法 1 と方法 2 を組み合わせたものです。方法 3 の効果は、最初の SQLDA のサイズを適切に選定できるかどうかによって左右されます。

例

例 *1:* C プログラムで DESCRIBE INPUT ステートメントを実行します。準備済みステートメントの入力パラメーター・マーカーの数を想定し、その数の入力パラメーター・マーカーを記述するための十分な SQLDA を割り振ります。多くても 5 つのパラメーター・マーカーを記述すればよく、入力データに LOB が含まれないものとします。

```
EXEC SQL  BEGIN DECLARE SECTION;
  char stmt1_str [200];
EXEC SQL  END DECLARE SECTION;
EXEC SQL  INCLUDE SQLDA;
struct sqlda initialsqlda;
struct sqlda *sqldaPtr;

... /* stmt1_str contains INSERT statement with VALUES     */
    /* clause                                              */
EXEC SQL  PREPARE STMT1_NAME   FROM :stmt1_str;

... /* code to set SQLN to five and to allocate the SQLDA */
EXEC SQL  DESCRIBE INPUT  STMT1_NAME INTO  :SQLDA;

 ...
```

例 *2:* 20 項目の記述子域を保持できる大きさの 'NEWDA' という記述子を割り振り、それを DESCRIBE INPUT で使用します。

```
EXEC SQL ALLOCATE DESCRIPTOR 'NEWDA'
  WITH MAX 20;

EXEC SQL DESCRIBE INPUT STMT1
  USING SQL DESCRIPTOR 'NEWDA';
```

DESCRIBE PROCEDURE

DESCRIBE PROCEDURE ステートメントは、プロシージャーが戻した結果セットについての情報を入手します。結果セットなどの情報は、記述子に入れられます。

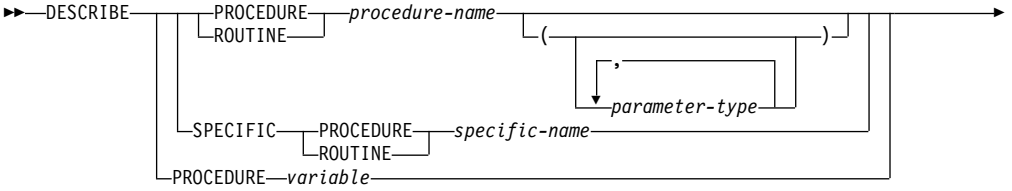
呼び出し

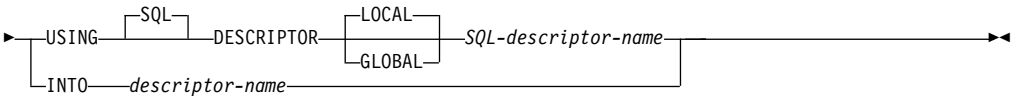
このステートメントは、アプリケーション・プログラム、SQL 関数、 SQL プロシージャー、またはトリガー内にのみ組み込むことができます。これは実行可能ステートメントですが、動的に準備することはできません。Java および REXX では指定できません。

権限

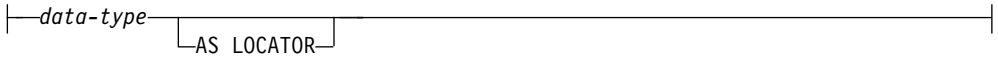
権限は不要です。

構文

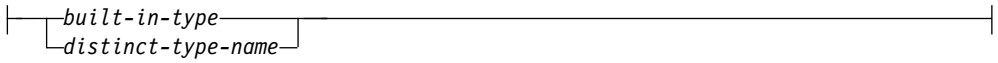




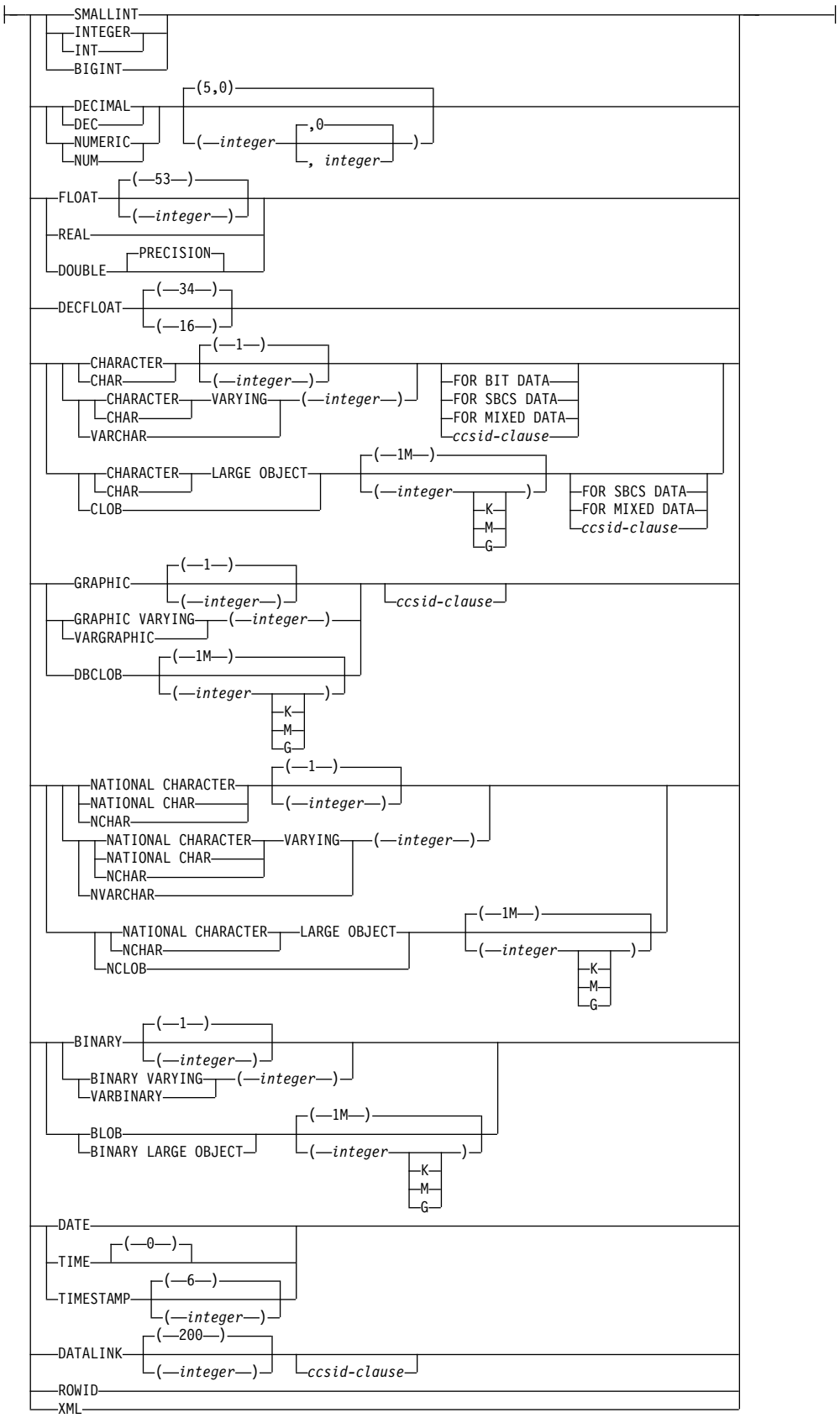
parameter-type:



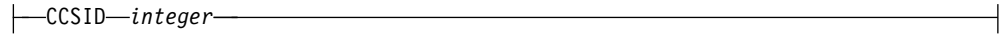
data-type:



built-in-type:


SMALLINT
INTEGER
INT
BIGINT
DECIMAL
DEC
NUMERIC
NUM
(5,0)
(integer)
,0
, integer
FLOAT
(53)
(integer)
REAL
DOUBLE
PRECISION
DECFLOAT
(34)
(16)
CHARACTER
CHAR
(1)
(integer)
CHARACTER VARYING
CHAR VARYING
VARCHAR
FOR BIT DATA
FOR SBCS DATA
FOR MIXED DATA
ccsid-clause
CHARACTER LARGE OBJECT
CHAR LARGE OBJECT
CLOB
(1M)
K
M
G
GRAPHIC
VARGRAPHIC
GRAPHIC VARYING
DBCLOB
NATIONAL CHARACTER
NATIONAL CHAR
NCHAR
NATIONAL CHARACTER VARYING
NATIONAL CHAR VARYING
NVARCHAR
NATIONAL CHARACTER LARGE OBJECT
NCLOB
BINARY
BINARY VARYING
VARBINARY
BLOB
BINARY LARGE OBJECT
DATE
TIME
(0)
TIMESTAMP
(6)
DATALINK
(200)
ROWID
XML

ccsid-clause:



説明

procedure-name または *specific-name* または *variable*
1 つ以上の結果セットを戻したプロシージャーを識別します。DESCRIBE PROCEDURE ステートメントを実行する際のプロシージャー名は、リクエスターがすでに SQL CALL ステートメントを使用して呼び出したプロシージャーを示している必要があります。

PROCEDURE または **SPECIFIC PROCEDURE**
記述するプロシージャーを指定します。このプロシージャー名 は、現行サーバーに存在しているプロシージャーを識別していなければなりません。

PROCEDURE *procedure-name*
プロシージャーを名前によって識別します。プロシージャー名 は、ただ 1 つのプロシージャーを識別していなければなりません。このプロシージャーには、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前のプロシージャーが複数ある場合、エラーが戻されます。

PROCEDURE *procedure-name (parameter-type, ...)*
プロシージャーを一意的に識別するプロシージャー・シグニチャーによって、プロシージャーを識別します。 *procedure-name (parameter-type,...)* では、指定されたプロシージャー・シグニチャーを持つプロシージャーを識別する必要があります。指定されたパラメーターは、プロシージャーの作成時に指定された、対応する位置にあるデータ・タイプと一致していなければなりません。ラベルを付ける対象のプロシージャー・インスタンスを識別するには、データ・タイプの数とデータ・タイプの論理連結を使用します。データ・タイプの同義語は、一致として扱われます。デフォルトがあるパラメーターは、このシグニチャーに含まれていなければなりません。

プロシージャー名 *()* を指定する場合、識別されるプロシージャーにパラメーターを使用することはできません。

procedure-name
プロシージャーの名前を識別します。

(parameter-type, ...)
プロシージャーのパラメーターを識別します。

非修飾の特殊タイプ名または配列タイプ名を指定する場合、データベース・マネージャーはその特殊タイプまたは配列タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、以下のいずれかを使用します。

- 空の括弧は、データベース・マネージャーがデータ・タイプの一致の判別に際して属性を無視することを示します。例えば、DEC() は、DEC(7,2) のデータ・タイプで定義されたプロシージ

ャーのパラメーターに一致するものとみなされます。ただし、FLOAT に空の括弧を指定することはできません。これは、そのパラメーター値が特定のデータ・タイプ (REAL または DOUBLE) を示しているからです。

- 長さ属性、精度属性、あるいは位取り属性に特定の値を指定する場合、その値は、 CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプが FLOAT の場合、突き合わせはデータ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度は指定された値に厳密に一致している必要はありません。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。暗黙の長さは、CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

FOR DATA 文節または CCSID 文節の指定はオプショナルです。いずれの文節も指定しないと、データ・タイプが一致するかどうかを判定する場合に、データベース・マネージャーが属性を無視することを示します。どちらか一方の文節を指定する場合は、CREATE PROCEDURE ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

AS LOCATOR
プロシージャーが、このパラメーターのロケーターを受け取るように定義されることを示します。 AS LOCATOR を指定する場合は、データ・タイプは LOB または XML、あるいは LOB または XML に基づく特殊タイプでなければなりません。 AS LOCATOR を指定した場合、FOR SBCS DATA または FOR MIXED DATA を指定してはなりません。

SPECIFIC PROCEDURE *specific-name*
プロシージャーを特定名によって識別します。特定名 は、現行サーバーに存在している特定のプロシージャーを識別していなければなりません。

variable
プロシージャー名または特定名を含んでいる変数を指定します。*variable* を指定した場合:

- その変数は、文字ストリング変数またはユニコード・グラフィック・ストリング変数でなければなりません。この変数は、グローバル変数にすることはできません。
- 標識変数を伴っていてはなりません。
- 変数内に含まれる名前は左寄せでなければならず、その長さが変数の長さより短い場合は、右側にブランクを埋め込まなければなりません。
- 名前は、区切り文字付きの名前でない限り、大文字でなければなりません。

この名前の 1 つのプロシージャーのみが CALL ステートメントを使用して呼び出された場合、変数はプロシージャー名として使用されます。この名前の複数のプロシージャーが呼び出された場合、変数は特定名として使用されます。

USING
SQL 記述子を識別します。

LOCAL
記述子の名前の有効範囲はプログラム呼び出しのローカルであることを指定します。

GLOBAL
記述子の名前の有効範囲は SQL セッション全体であることを指定します。

SQL-descriptor-name
SQL 記述子の名前を指定します。名前は、指定の有効範囲を持つ既存の記述子を識別するものでなければなりません。

DESCRIBE PROCEDURE の実行後に、GET DESCRIPTOR ステートメントで以下の値を取得できます。

- DB2_RESULT_SETS_COUNT。結果セットの総数が入ります。 0 の値は結果セットがないことを示します。

各結果セットについて 1 つの記述子域がある。

- DB2_RESULT_SET_LOCATOR。結果セットに関連した結果セット・ロケーターの値が入ります。
- DB2_CURSOR_NAME。プロシージャーが結果セットを返すために使用したカーソルの名前が入ります。
- DB2_RESULT_SET_ROWS。結果セットの行の概数。数が不明な場合は、-1 が設定されます。

SQL 記述子に入る情報の説明については、 1476 ページの『GET DESCRIPTOR』を参照してください。

INTO *descriptor-name*
SQL 記述子域 (SQLDA) を指定します。これについては、 1877 ページの『付録 D. SQLDA (SQL 記述子域)』で説明しています。DESCRIBE PROCEDURE ステートメントを実行する前に、SQLDA の以下の変数をセットしておく必要があります。

SQLN
SQLDA で用意される SQLVAR オカレンスの数を指定します。DESCRIBE PROCEDURE ステートメントを実行する前に、SQLN にゼロ以上の値をセットしておく必要があります。

DESCRIBE ステートメントが実行されると、データベース・マネージャーでは、SQLDA の各変数に次のような値を割り当てます。

SQLDAID
最初の 5 バイトは「SQLPR」に設定される。6 から 8 バイトは予約されている。

SQLDABC
SQLDA の長さ (バイト)。

SQLD
結果セットの総数。0 の値は結果セットがないことを示します。

SQLVAR
SQLD の値が 0 の場合、または SQLD の値が SQLN の値より大きい場合は、SQLVAR のオカレンスには値が割り当てられません。

SQLD の値が *n* (ただし、*n* は 0 より大きいが、SQLN の値より小さいか、または等しい値) の場合は、値が SQLVAR の最初の *n* 個のオカレンスに割り当てられ、その結果、SQLVAR の最初のオカレンスには最初の結果セットの記述が入り、SQLVAR の 2 番目のオカレンスには 2 番目の結果セットの記述が入ります。以下同様です。それぞれの SQLVAR 項目ごとに、以下のようになります。

- SQLDATA フィールドは結果セットと関連する結果セット・ロケーター値に設定される。
- SQLIND フィールドは結果セット内の行の予測数に設定される。数が不明な場合は、-1 が設定されます。
- SQLNAME フィールドは、結果セットを戻すためにストアード・プロシージャーが使用するカーソル名に設定される。カーソル名の長さが 30 文字を超えている場合は、カーソル名が 30 文字に切り捨てられます。

注

DESCRIBE PROCEDURE はプロシージャーが必要とするパラメーターについての情報を戻しません。

DESCRIBE PROCEDURE ステートメントの前に、対象のプロシージャーに対する CALL が必要です。

SQL 記述子の割り振り**:** DESCRIBE PROCEDURE ステートメントを実行する前に、ALLOCATE DESCRIPTOR ステートメントを使用して SQL 記述子を割り振らなければなりません。割り振られた記述子項目の数が結果セットの数よりも小さい場合、警告 (SQLSTATE 01005) が戻されます。

SQLDA の割り振り**:** DESCRIBE PROCEDURE ステートメントを実行する前に、SQLN にゼロ以上の値をセットして、SQLDA に用意する SQLVAR のオカレンスの数を指示するとともに、SQLN の各オカレンスを収容するのに十分な記憶域を割り振っておく必要があります。プロシージャーの結果セットの記述を取得するには、SQLVAR のオカレンスの数が結果セットの数以上でなければなりません。

提供されるオカレンスが不足していて、オカレンスのすべてのセットを戻せるとは限らない場合、SQLN は、すべての情報を戻すのに必要なオカレンスの合計数に設定されます。それ以外の場合、SQLN は、結果セットの数に設定されます。

ロケーター値の割り当て**:** SET RESULT SETS ステートメントがプロシージャーで実行される場合は、その SET RESULT SETS ステートメントが結果セットを識別します。ロケーター値は、SET RESULT SETS ステートメントで指定した順序で、記述子域の項目または SQLDA の SQLVAR 項目に割り当てられます。プロシージャーで SET RESULT SETS ステートメントを実行しなかった場合は、関連するカーソルが実行時にオープンした順序で、記述子域の項目または SQLDA の

SQLVAR 項目にロケーター値が割り当てられます。ロケーター値は、呼び出し側のアプリケーションに制御が戻されるとクローズされるカーソルには提供されません。カーソルがクローズされ、呼び出し側のアプリケーションに戻る前に再度オープンされた場合は、プロシージャー結果セットにロケーター値が戻される順序を決めるために、カーソルに対して最新に実行された OPEN CURSOR ステートメントが使用されます。例えば、プロシージャー P1 が 3 つのカーソル A、B、および、C をオープンして、カーソル B をクローズし、呼び出し側のアプリケーションに戻る前にカーソル B に対して別の OPEN CURSOR ステートメントを実行したとします。ロケーター値は、A、C、B の順序で割り当てられます。

また、ASSOCIATE LOCATORS ステートメントを使用して、ロケーター値を結果セットのロケーター変数にコピーすることもできます。

例

プロシージャー P1 から返される結果セットに関する情報を SQL 記述子に配置します。

```
EXEC SQL CALL P1;
EXEC SQL ALLOCATE DESCRIPTOR 'DESC1';
EXEC SQL DESCRIBE PROCEDURE P1 USING SQL DESCRIPTOR 'DESC1';
```

DESCRIBE TABLE

DESCRIBE TABLE ステートメントは、表またはビューに関する情報を入手します。

呼び出し

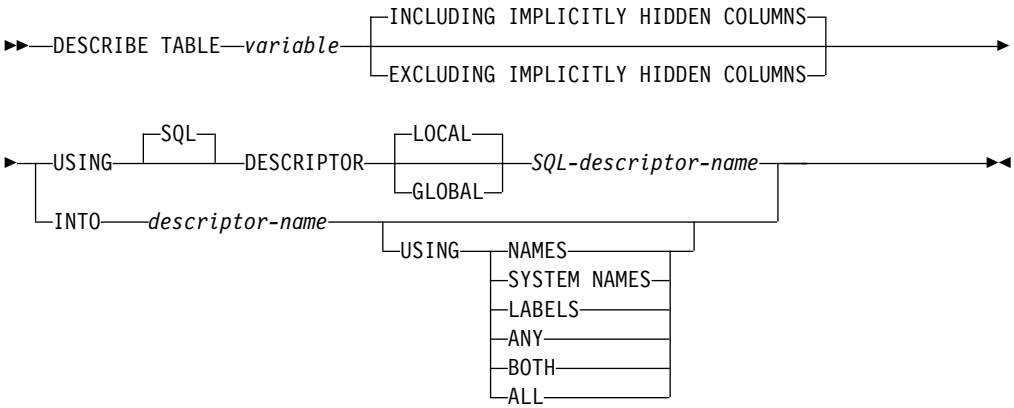
このステートメントは、アプリケーション・プログラム、SQL 関数、 SQL プロシージャー、またはトリガー内にのみ組み込むことができます。これは実行可能ステートメントですが、動的に準備することはできません。Java では指定できません。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメントに指定された表またはビューに対して、
  - その表またはビューに対する *OBJOPR システム権限
  - 表またはビューが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

構文



説明

variable

説明する表またはビューを指定します。 DESCRIBE TABLE ステートメントを実行する時点で、

- この名前は、アプリケーション・サーバーにある表またはビューを識別するものでなければなりません。
- 変数 は文字ストリング変数または Unicode グラフィック・ストリング変数でなければならず、標識変数を含んでいてはなりません。この変数は、グローバル変数にすることはできません。
- 変数 内に含まれる表名は左寄せでなければならず、その長さが変数 の長さより短い場合は、右側にブランクを埋め込まなければなりません。
- 表の名前は、区切り文字付の名前でない限り、大文字にしなければなりません。

INCLUDING IMPLICITLY HIDDEN COLUMNS または **EXCLUDING IMPLICITLY HIDDEN COLUMNS**
表の中の暗黙的隠し列に関する情報を戻すべきかどうかを指定します。

INCLUDING IMPLICITLY HIDDEN COLUMNS
暗黙的隠し列として定義された列に関する情報を戻すことを指定します。これはデフォルトです。

EXCLUDING IMPLICITLY HIDDEN COLUMNS
暗黙的隠し列として定義された列に関する情報を戻さないことを指定します。

DESCRIBE TABLE ステートメントを実行すると、データベース・マネージャーによって、SQL 記述子または SQLDA の各変数には次のような値が割り当てられます。

USING
SQL 記述子を識別します。

LOCAL
記述子の名前の有効範囲はプログラム呼び出しのローカルであることを指定します。

GLOBAL
記述子の名前の有効範囲は SQL セッション全体であることを指定します。

SQL-descriptor-name
SQL 記述子の名前を指定します。名前は、指定の有効範囲を持つ既存の記述子を識別するものでなければなりません。

SQL 記述子に入る情報の説明については、 1476 ページの『GET DESCRIPTOR』を参照してください。

INTO *descriptor-name*
SQL 記述子域 (SQLDA) を指定します。これについては、 1877 ページの『付録 D. SQLDA (SQL 記述子域)』で説明しています。DESCRIBE TABLE ステートメントを実行する前に、SQLDA の以下の変数をセットしておく必要があります。

SQLN
SQLDA で用意される SQLVAR オカレンスの数を指定します。DESCRIBE TABLE ステートメントを実行する前に、SQLN をゼロ以上の値にセットしておく必要があります。必要なオカレンスの数を決定する手法については、 1880 ページの『必要な SQLVAR オカレンスの数の決定』を参照してください。

REXX の場合は、規則が異なります。詳しくは、「組み込み SQL プログラミング」のトピック集を参照してください。

DESCRIBE ステートメントが実行されると、データベース・マネージャーでは、SQLDA の各変数に次のような値を割り当てます。

SQLDAID
最初の 6 バイトは 'SQLDA ' (つまり、5 文字の後にスペース文字) に設定されます。

7 番目のバイトは、記述された列に基づいて設定されます。

- SQLDA に表の列ごとに 2 つ、3 つ、または 4 つの SQLVAR 項目が含まれる場合、この 7 番目のバイトはそれぞれ '2'、'3'、または '4' に設定されます。この技法は、LOB または特殊タイプ結果列、ラベル、およびシステム名に対応するために使用されています。
- それ以外の場合、7 番目のバイトはスペース文字に設定されます。

SQLDA 内にすべての列の記述を含める余地がない場合、7 番目のバイトはスペース文字に設定されます。

8 番目のバイトはスペース文字に設定されます。

SQLDABC
SQLDA の長さ (バイト)。

SQLD
表内の列数に拡張 SQLVAR 項目数を加えたものです。拡張 SQLVAR 項目については、 1882 ページの『SQLVAR のオカレンスのフィールドの説明』を参照してください。

SQLVAR
SQLD の値が 0 の場合、または SQLD の値が SQLN の値より大きい場合は、SQLVAR のオカレンスには値が割り当てられません。

SQLD の値が *n* (ただし、*n* は 0 より大きいが、SQLN の値より小さいか、または等しい値) の場合は、値が SQLVAR の最初の *n* 個のオカレンスに割り当てられ、その結果、SQLVAR の最初のオカレンスには表の最初の列の記述が入り、SQLVAR の 2 番目のオカレンスには表の 2 番目の列の記述が入ります。以下同様です。SQLVAR オカレンスに割り当てられる値については、 1882 ページの『SQLVAR のオカレンスのフィールドの説明』を参照してください。

USING
SQLDA のそれぞれの SQLNAME 変数に、どのような値を割り当てるかを指定します。要求した値が存在しない場合、または名前の長さが 30 より大きい場合は、SQLNAME の長さは 0 にセットされます。

NAMES
列の名前を割り当てます。戻される列名は大/小文字の区別があり、区切り文字はありません。これはデフォルトです。

SYSTEM NAMES
列のシステム列名を割り当てます。

LABELS
列のラベルを割り当てます。(列のラベルは、LABEL ステートメントによって定義されます。) ラベルの最初の 20 バイトだけが戻されます。

ANY
列のラベルを割り当てます。列にラベルがない場合は、代わりに列の名前が割り当てられます。

BOTH
列のラベルと名前の両方を割り当てます。この場合、追加情報に応じるために、1 つの列ごとに SQLVAR の 2 つから 3 つのオカレンスが必

要になりますが、その数は、表に特殊タイプが入っているか否かによって決まります。この拡張の SQLVAR 配列を指定するには、SQLN を 2*n* か 3*n* (この場合の *n* は、表やビュー内の列数) に設定します。列名がシステム列名とは異なる場合、列名は SQLVAR の最初の *n* オカレンスに含まれます。 2 番目または 3 番目の *n* オカレンスには、列のラベルが含まれます。 特殊タイプがない場合、SQLVAR 項目の 2 番目のセットにそのラベルが戻されます。それ以外の場合、ラベルは、SQLVAR 項目の 3 番目のセット内に戻されます。

ALL
ラベル、列名、およびシステム列名を割り当てます。この場合、追加情報に応じるために、 1 つの列ごとに SQLVAR の 3 つから 4 つのオカレンスが必要になりますが、その数は、表に特殊タイプが入っているか否かによって決まります。この拡張の SQLVAR 配列を指定するには、SQLN を 3*n* か 4*n* (この場合の *n* は、表内の列数) に設定します。 SQLVAR の最初の *n* オカレンスには、システム列名が入ります。 2 番目または 3 番目の *n* オカレンスには、列のラベルが含まれます。 3 番目または 4 番目の *n* オカレンスには、列名が含まれます。特殊タイプが指定されていない場合、ラベルは、SQLVAR 記入項目の 2 番目のセット内に戻され、列名は、SQLVAR 記入項目の 3 番目のセット内に戻されます。それ以外の場合、ラベルは、SQLVAR 記入項目の 3 番目のセット内に戻され、列名は、 SQLVAR 記入項目の 4 番目のセット内に戻されます。

注

SQL 記述子の割り振り**:** DESCRIBE TABLE ステートメントを実行する前に、ALLOCATE DESCRIPTOR ステートメントを使用して SQL 記述子を割り振らなければなりません。割り振られた記述子項目の数が表またはビューの列の数より小さい場合は、警告 (SQLSTATE 01005) が戻されます。

SQLDA の割り振り**:** DESCRIBE TABLE ステートメントを実行する前に、SQLN にゼロ以上の値をセットして、SQLDA に用意する SQLVAR のオカレンスの数を指示するとともに、SQLN の各オカレンスを収容するのに十分な記憶域を割り振っておく必要があります。表またはビューの列の記述を取得するには、SQLVAR のオカレンスの数が列の数以上でなければなりません。さらに、USING BOTH や USING ALL を指定している場合、あるいは、列に LOB や特殊タイプを指定している場合は、SQLVAR のオカレンス数として、列数の 2、3、または 4 倍の数値を指定する必要があります。詳しくは、 1880 ページの『必要な SQLVAR オカレンスの数の決定』を参照してください。

提供されるオカレンスが不足していて、オカレンスのすべてのセットを戻せるとは限らない場合、SQLN は、すべての情報を戻すのに必要なオカレンスの合計数に設定されます。それ以外の場合、SQLN は、列数に設定されます。

SQLDA の割り振りのために使用する方法については、 1877 ページの『付録 D. SQLDA (SQL 記述子域)』を参照してください。

例

C プログラムの中で、SQLVAR のオカレンスなしの SQLDA を使用して DESCRIBE ステートメントを実行します。SQLD がゼロより大きければ、その値を使用して、SQLVAR に必要なオカレンス数を指定した SQLDA を割り振ります。その後で、新しい SQLDA を使用して DESCRIBE ステートメントを実行します。

```
EXEC SQL  BEGIN DECLARE SECTION;
  char table_name[201];
EXEC SQL  END DECLARE SECTION;
EXEC SQL  INCLUDE SQLDA;
EXEC SQL  DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

.../*code to prompt user for a table or view */
.../*code to set SQLN to zero and to allocate the SQLDA */
EXEC SQL  DESCRIBE TABLE :table_name INTO :sqlda;

... /* code to check that SQLD is greater than zero, to set */
    /* SQLN to SQLD, then to re-allocate the SQLDA          */
EXEC SQL  DESCRIBE TABLE :table_name INTO :sqlda;

.
.
.
```

DISCONNECT

DISCONNECT ステートメントは、無保護会話の 1 つ以上の接続を終了させます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、または対話式に呼び出すことができます。これは実行可能ステートメントですが、動的に準備することはできません。Java および REXX では指定できません。

DISCONNECT は、トリガーおよび関数で使用できません。

権限

ステートメントでグローバル変数を参照する場合は、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別されるグローバル変数に対して、
  - そのグローバル変数に対する READ 特権
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

構文

説明

server-name または *variable*

指定したサーバー名、または指定した変数に入っているサーバー名によってアプリケーション・サーバーを識別します。スキーマ名で修飾すれば、グローバル変数を使用することもできます。変数を指定する場合

- その変数は、文字ストリング変数でなければなりません。
- 標識変数を伴っていてはなりません。
- サーバー名は、その変数内で左寄せし、通常 ID の形成の規則に従っていなければなりません。
- サーバー名の長さが、変数の長さよりも短い場合、右側をブランクで埋めなければなりません。

DISCONNECT ステートメントが実行される時点で、指定したサーバー名または変数に入っているサーバー名は、その活動化グループの既存の休止接続、または現行接続を識別していなければなりません。識別された接続は、保護会話を使用できません。

CURRENT

活動化グループの現行接続を識別します。活動化グループは接続状態でなければなりません。その現行接続は、保護会話を使用してはなりません。

ALL または **ALL SQL**
活動化グループの既存のすべての接続 (ローカルおよびリモートの接続の両方) を識別します。このステートメントの実行時に接続が存在しない場合、エラーや警告は起こりません。接続は、いずれも保護会話を使用することはできません。

注

DISCONNECT と **CONNECT (**タイプ **1):** CONNECT (タイプ 1) の使用は、DISCONNECT の使用を妨げることはありません。

接続制限**:** 識別される接続は、現行作業単位の過程で SQL ステートメントを実行するのに使用された接続であってはならず、また保護会話の接続であってはなりません。保護会話の接続を終了するには、RELEASE ステートメントを使用します。ローカル接続は、保護会話と見なされることはありません。

DISCONNECT ステートメントは、コミット操作の直後に実行しなければなりません。 DISCONNECT が現行接続の終了に使用される場合、次に実行される SQL ステートメントは、 CONNECT または SET CONNECTION でなければなりません。

ROLLBACK は、DISCONNECT によって終了した接続を再接続することはありません。

切断成功**:** DISCONNECT ステートメントが正常に実行された場合は、識別された接続はそれぞれ終了します。現行接続が遮断されると、その活動化グループは未接続状態になります。

DISCONNECT は、カーソルをクローズし、リソースを解放し、その接続のそれ以上の使用を防止します。

DISCONNECT ALL は、ローカル・アプリケーション・サーバーとの接続を終了させます。接続に、WITH HOLD 文節を指定して定義されたオープン・カーソルがある場合でも、接続は終了します。

接続失敗**:** DISCONNECT ステートメントが正常に実行されなかった場合は、その活動化グループの接続状態、およびその接続の状態は変わりません。

リモート接続のリソースに関する考慮事項**:** リモート接続を作成し維持するためには、リソースが必要です。したがって、再使用の予定がないリモート接続は、できるだけ早く終了するようにする必要があり、再使用の予定があるリモート接続は、遮断されないようにする必要があります。

例

例 *1* : TOROLAB1 との接続は不要になりました。次のステートメントは、コミット操作の後で実行されます。

```
EXEC SQL DISCONNECT TOROLAB1;
```

例 *2* : 現行接続は不要になりました。次のステートメントは、コミット操作の後で実行されます。

```
EXEC SQL DISCONNECT CURRENT;
```

例 *3* : 既存の接続は不要になりました。次のステートメントは、コミット操作の後で実行されます。

```
EXEC SQL DISCONNECT ALL;
```

DROP

DROP ステートメントは、オブジェクトを除去します。削除されるオブジェクトに直接または間接的に依存しているオブジェクトも除去できます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

表、ビュー、索引、別名またはパッケージを除去するには、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- 次のシステム権限
  - 除去したいオブジェクトについての *OBJOPR および *OBJEXIST システム権限。
  - 該当のオブジェクトが表またはビューである場合は、その表またはビューに従属しているビュー、索引、および論理ファイルに対する *OBJOPR および *OBJEXIST システム権限。
  - 当該オブジェクトがシステム期間テンポラル表である場合は、関連する履歴表に対する *OBJOPR および *OBJEXIST システム権限。
  - 除去したいオブジェクトが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

スキーマを除去するには、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- 次のシステム権限
  - 除去したいライブラリーに対する *OBJEXIST、*OBJOPR、*EXECUTE、および *READ システム権限。
  - そのスキーマのすべてのオブジェクトについての *OBJOPR および *OBJEXIST システム権限、およびそのスキーマの中の表やビューに従属するビュー、索引、および論理ファイルについての *OBJOPR および *OBJEXIST システム権限。
  - そのスキーマの中のその他のオブジェクト・タイプの削除に必要な追加の権限。スキーマにデータ・ディクショナリーが入っている場合における、そのデータ・ディクショナリーに対する *OBJMGT、およびジャーナル・レシーバーに対する一部のシステム・データ権限がその例です。詳しくは、「機密保護解説書」を参照してください。
- データベース管理者権限

ユーザー定義タイプを除去するためには、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- 次のシステム権限
  - 除去したいタイプについての *OBJOPR および *OBJEXIST システム権限。

  - 除去したいタイプが含まれるスキーマに対する USAGE 特権
  - SYSTYPES、SYSPARMS、および SYSROUTINES カタログ表に対する DELETE 特権
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

グローバル変数を除去するためには、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- 次のシステム権限
  - 削除するグローバル変数の *SRVPGM オブジェクトに対する *OBJEXIST システム権限
  - 除去したいグローバル変数が含まれるスキーマに対する USAGE 特権
  - SYSVARIABLES カタログ表に対する DELETE 特権、および
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

XSR オブジェクトを除去するためには、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- 次のシステム権限
  - 削除する XSR オブジェクトの *SQLXSR オブジェクトに対する *OBJOPR システム権限と *OBJEXIST システム権限
  - 除去したい XSR オブジェクトが含まれるスキーマに対する USAGE 特権
  - XSROBJECTS、XSROBJECTCOMPONENTS、および XSRANNOTATIONINFO カタログ表に対する DELETE 特権
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

関数を除去するためには、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- 次のシステム権限
  - SQL 関数の場合、その関数に関連したサービス・プログラム・オブジェクトに対する *OBJEXIST システム権限
  - SYSFUNCS、SYSPARMS、および SYSROUTINEDEP カタログ表に対する DELETE 特権
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

プロシージャーを除去するためには、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- 次のシステム権限
  - SQL プロシージャーの場合、そのプロシージャーに関連したプログラム・オブジェクトに対する *OBJEXIST システム権限
  - SYSPROCS、SYSPARMS、および SYSROUTINEDEP カタログ表に対する DELETE 特権

  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

シーケンスを除去するためには、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- 次のシステム権限
  - シーケンスに関連したデータ域に対する *OBJEXIST システム権限
  - 除去したいシーケンスが含まれるスキーマに対する USAGE 特権
  - SYSSEQOBJECTS カタログ表に対する DELETE 特権、および
  - スキーマ QSYS2 に対する USAGE 特権、および
  - データ域削除 (DLTDTAARA) コマンドに対する *USE 権限。
- データベース管理者権限

トリガーを除去するには、ステートメントの権限 ID が保持する特権に、次のうち少なくともいずれか 1 つを含める必要があります。

- 次の権限
  - 物理ファイル削除トリガー (RMVPFTRG) コマンドに対する *USE システム権限
  - トリガーの対象表またはビューに対する権限
    - 対象表またはビューに対する ALTER 特権
    - 対象表またはビューが含まれるスキーマに対する USAGE 特権
  - 削除されるトリガーが SQL トリガーの場合
    - トリガー・プログラム・オブジェクトに対する *OBJEXIST システム権限
    - トリガーが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

マスクまたは許可を除去する場合は、次のとおりです。

- このステートメントの許可 ID には、セキュリティー管理者権限 がなければなりません。 21 ページの『管理権限』を参照してください。

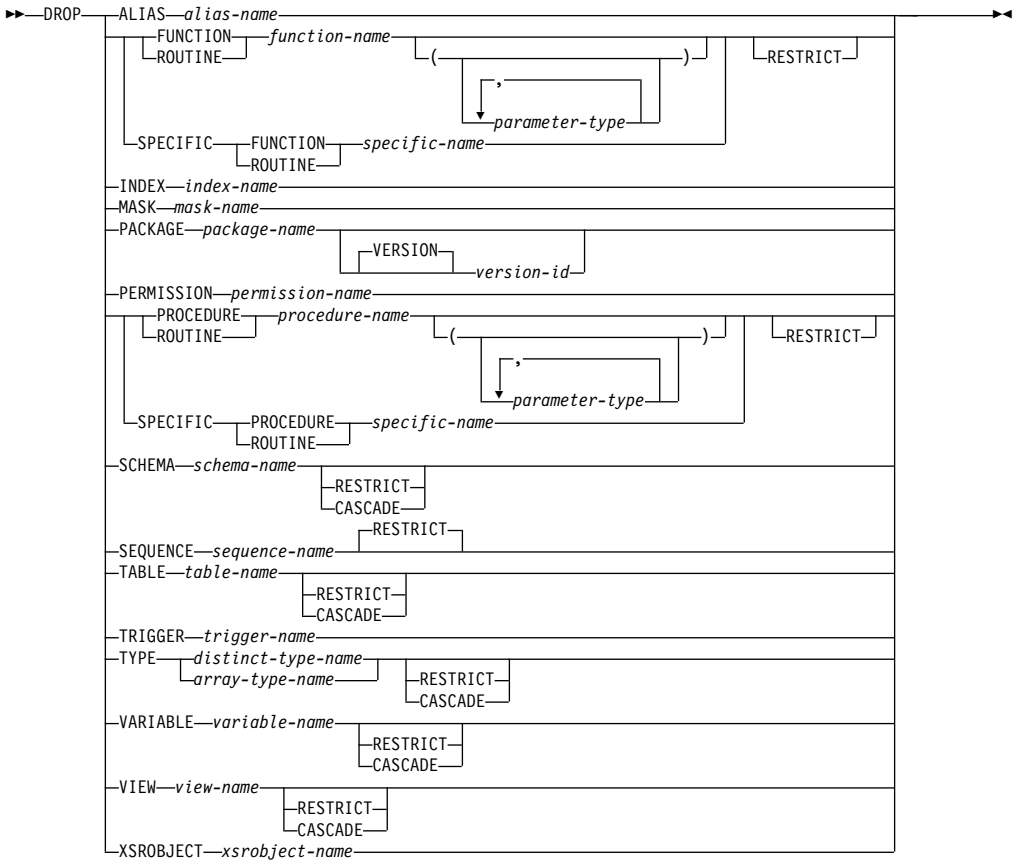
マスクまたは許可によって使用されるオブジェクトを除去する場合は、次のとおりです。

- このステートメントの許可 ID には、セキュリティー管理者権限 がなければなりません。 21 ページの『管理権限』を参照してください。

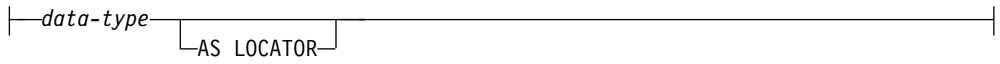
SQL 特権に対応するシステム権限については、『表またはビューへの権限を検査する際の対応するシステム権限』を参照してください。

構文

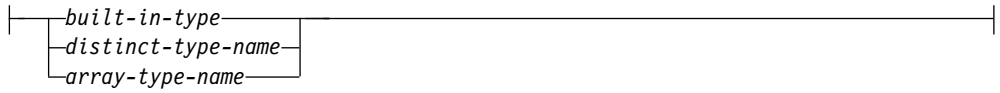
DROP



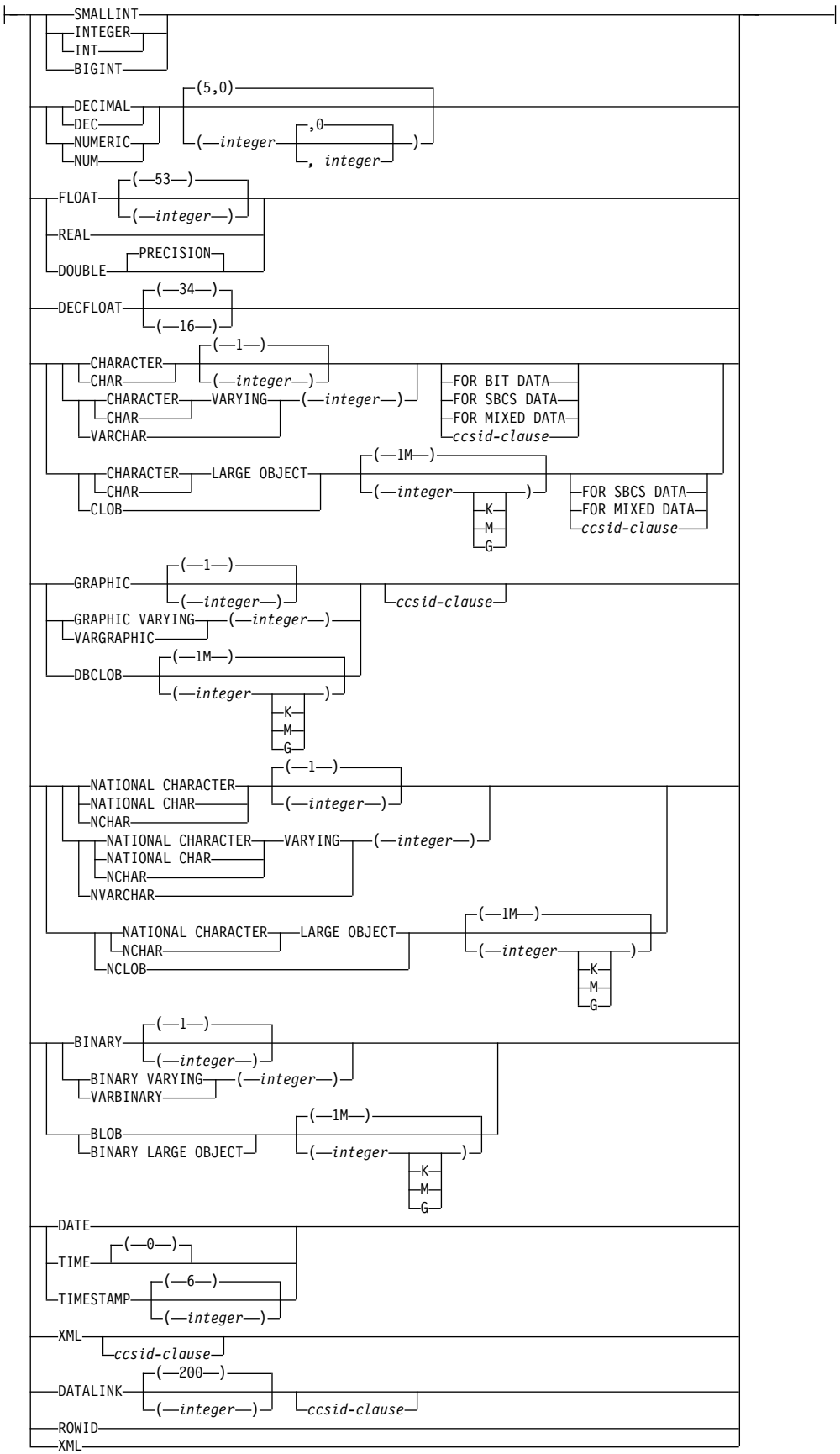
parameter-type:



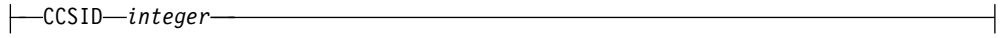
data-type:



built-in-type:


SMALLINT
INTEGER
INT
BIGINT
DECIMAL
DEC
NUMERIC
NUM
(5,0)
(integer
,0
, integer
)
FLOAT
(53)
(integer)
REAL
DOUBLE
PRECISION
DECFLOAT
(34)
(16)
CHARACTER
CHAR
(1)
(integer)
CHARACTER VARYING
CHAR VARYING
VARCHAR
FOR BIT DATA
FOR SBCS DATA
FOR MIXED DATA
ccsid-clause
CHARACTER LARGE OBJECT
CHAR LARGE OBJECT
CLOB
(1M)
K
M
G
GRAPHIC
VARGRAPHIC
GRAPHIC VARYING
DBCLOB
NATIONAL CHARACTER
NATIONAL CHAR
NCHAR
NATIONAL CHARACTER VARYING
NATIONAL CHAR VARYING
NVARCHAR
NATIONAL CHARACTER LARGE OBJECT
NCLOB
BINARY
BINARY VARYING
VARBINARY
BLOB
BINARY LARGE OBJECT
DATE
TIME
TIMESTAMP
(0)
(6)
XML
DATALINK
(200)
ROWID

ccsid-clause:



説明

ALIAS *alias-name*

除去したい別名を識別します。この別名 は、現行サーバーに存在している別名を示すものでなければなりません。

指定した別名は、スキーマから削除されます。別名を除去しても、その別名を使用して定義された制約、ビュー、またはマテリアライズ照会には影響を与えません。別名は、関数、パッケージ、プロシージャー、プログラム、トリガー、または変数で参照されているかに関係なく、除去できます。

FUNCTION または **SPECIFIC FUNCTION**

除去したい関数を識別します。その関数は現行サーバーに存在していて、CREATE FUNCTION ステートメントによって定義された関数であることが必要です。特定の関数は、それぞれその名前、関数シグニチャー、あるいは特定名によって識別することができます。

CREATE TYPE ステートメントによって暗黙的に生成された関数は、 DROP ステートメントによって除去できません。特殊タイプが除去されると、それらは暗黙的に除去されます。

関数は、別の関数がそれに従属している場合は、除去できません。関数が別の関数に従属するのは、CREATE FUNCTION ステートメントの SOURCE 文節でそれが識別されている場合です。RESTRICT が指定されなければ、関数は、関数、パッケージ、プロシージャー、プログラム、トリガー、変数、またはビューで参照されているかどうかに関係なく、除去できます。マスクまたは許可で参照されている関数は、RESTRICT が指定されていない場合でも除去できません。

指定した関数は、スキーマから除去されます。ユーザー定義関数に対する特権も、すべて除去されます。これが、SQL 関数の場合、またはソース化関数の場合、その関数に関連したサービス・プログラム (*SRVPGM) も削除されます。これが外部関数の場合、CREATE FUNCTION ステートメントに指定されているプログラムまたはサービス・プログラム内に保管されている情報も、そのオブジェクトから削除されます。

FUNCTION *function-name*

関数を名前によって識別します。 *function-name* は厳密に 1 つの関数を示す必要があります。この関数には、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前の関数が複数ある場合、エラーが戻されます。

FUNCTION *function-name (parameter-type, ...)*

関数を一意的に識別する関数シグニチャーによって、関数を識別します。 *function-name (parameter-type,...)* は、指定された関数シグニチャーを持つ関数を識別しなければなりません。指定されたパラメーターは、関数の作成時に指定された、対応する位置にあるデータ・タイプと一致していなければなりません。除去する関数インスタンスを識別する場合、データ・タイプの数

とデータ・タイプの論理連結が使用されます。データ・タイプの同義語は、一致として扱われます。デフォルトがあるパラメーターは、このシグニチャーに含まれていなければなりません。

function-name () を指定する場合、識別される関数にパラメーターを使用することはできません。

function-name
関数の名前を識別します。

(parameter-type, ...)
関数のパラメーターを識別します。

非修飾の特殊タイプ名を指定する場合、データベース・マネージャーはその特殊タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、以下のいずれかを使用します。

- 空の括弧は、データベース・マネージャーがデータ・タイプの一致の判別に際して属性を無視することを示します。例えば、DEC() は、DEC(7,2) のデータ・タイプで定義された関数のパラメーターに一致するものとみなされます。ただし、FLOAT に空の括弧を指定することはできません。これは、そのパラメーター値が特定のデータ・タイプ (REAL または DOUBLE) を示しているからです。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を指定した場合、その値は、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプが FLOAT の場合、突き合わせはデータ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度は指定された値に厳密に一致している必要はありません。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。暗黙の長さは、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

FOR DATA 文節または CCSID 文節の指定はオプショナルです。いずれの文節も指定しないと、データ・タイプが一致するかどうかを判定する場合に、データベース・マネージャーが属性を無視することを示します。どちらか一方の文節を指定する場合は、CREATE FUNCTION ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

AS LOCATOR
関数が、このパラメーターのロケーターを受け取るように定義されることを示します。 AS LOCATOR を指定する場合は、データ・タイプは LOB または XML、あるいは LOB または XML に基づく特殊タイプでなければなりません。

SPECIFIC FUNCTION *specific-name*
関数を特定名によって識別します。 *specific-name* は、現行サーバーに存在する特定の関数を示している必要があります。

RESTRICT
関数が SQL 関数、SQL プロシージャー、表、マスク、権限、SQL トリガー、変数、またはビューで参照される場合は、除去できないことを指定します。

RESTRICT を指定しない場合
関数がマスクまたは許可で参照されている場合、ステートメントの権限 ID にセキュリティー管理者権限がないと除去は失敗します。

INDEX *index-name*
除去したい索引を識別します。この索引名 は、現行サーバーに存在している索引を示すものでなければなりません。

指定した索引は、スキーマから除去されます。索引は、関数、パッケージ、プロシージャー、プログラム、またはトリガーで参照されるかどうかに関わらず、除去できます。

MASK *mask-name*
除去したいマスクを識別します。マスク名 は、現行サーバーに存在するマスクを示していなければなりません。

指定したマスクは、スキーマから除去されます。マスクは、関数、パッケージ、プロシージャー、プログラム、またはトリガーで参照されているかに関係なく、削除できます。

PACKAGE *package-name*
除去したいパッケージを識別します。このパッケージ名 は、現行サーバーに存在しているパッケージを識別していなければなりません。

指定したパッケージは、スキーマから除去されます。パッケージに対する特権も、すべて削除されます。

パッケージは、関数、パッケージ、プロシージャー、プログラム、またはトリガーで参照されているかに関係なく、削除できます。

VERSION *version-id*
バージョン *ID* は、作成時にパッケージに割り当てられたバージョン ID です。バージョン *ID* を指定しない場合、バージョン ID として NULL ストリングが使用されます。

PERMISSION *permission-name*
除去したい権限を識別します。許可名 は、現行サーバーに存在する許可を示していなければなりません。

指定した権限は、スキーマから除去されます。権限は、関数、パッケージ、プロシージャー、プログラム、またはトリガーで参照されているかに関係なく、削除できます。

PROCEDURE または **SPECIFIC PROCEDURE**
除去したいプロシージャーを識別します。このプロシージャー名 は、現行サーバーに存在しているプロシージャーを識別していなければなりません。

指定したプロシージャーは、スキーマから除去されます。そのプロシージャーに対する特権も、すべて削除されます。これが SQL プロシージャーの場合、そのプロシージャーに関連したプログラム (*PGM) またはサービス・プログラム (*SRVPGM) も削除されます。これが外部プロシージャーの場合、CREATE

PROCEDURE ステートメントに指定されているプログラム内に保管されている情報も、そのオブジェクトから削除されます。

プロシージャーは、関数、パッケージ、プロシージャー、プログラム、またはトリガーで参照されているかに関係なく、削除できます。

PROCEDURE *procedure-name*
プロシージャーを名前によって識別します。プロシージャー名 は、ただ 1 つのプロシージャーを識別していなければなりません。このプロシージャーには、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前のプロシージャーが複数ある場合、エラーが戻されます。

PROCEDURE *procedure-name (parameter-type, ...)*
プロシージャーを一意的に識別するプロシージャー・シグニチャーによって、プロシージャーを識別します。 *procedure-name (parameter-type,...)* では、指定されたプロシージャー・シグニチャーを持つプロシージャーを識別する必要があります。指定されたパラメーターは、プロシージャーの作成時に指定された、対応する位置にあるデータ・タイプと一致していなければなりません。除去するプロシージャー・インスタンスを識別する場合、データ・タイプの数とデータ・タイプの論理連結が使用されます。データ・タイプの同義語は、一致として扱われます。デフォルトがあるパラメーターは、このシグニチャーに含まれていなければなりません。

プロシージャー名 *()* を指定する場合、識別されるプロシージャーにパラメーターを使用することはできません。

procedure-name
プロシージャーの名前を識別します。

(parameter-type, ...)
プロシージャーのパラメーターを識別します。

非修飾の特殊タイプ名または配列タイプ名を指定する場合、データベース・マネージャーはその特殊タイプまたは配列タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、以下のいずれかを使用します。

- 空の括弧は、データベース・マネージャーがデータ・タイプの一致の判別に際して属性を無視することを示します。例えば、DEC() は、DEC(7,2) のデータ・タイプで定義されたプロシージャーのパラメーターに一致するものとみなされます。ただし、FLOAT に空の括弧を指定することはできません。これは、そのパラメーター値が特定のデータ・タイプ (REAL または DOUBLE) を示しているからです。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を指定する場合、その値は、 CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプが FLOAT の場合、突き合わせはデータ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度は指定された値に厳密に一致している必要はありません。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフ

ォルト属性が暗黙指定されます。暗黙の長さは、CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

FOR DATA 文節または CCSID 文節の指定はオプショナルです。いずれの文節も指定しないと、データ・タイプが一致するかどうかを判定する場合に、データベース・マネージャーが属性を無視することを示します。どちらか一方の文節を指定する場合は、CREATE PROCEDURE ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

AS LOCATOR
プロシージャーが、このパラメーターのロケーターを受け取るように定義されることを示します。 AS LOCATOR を指定する場合は、データ・タイプは LOB または XML、あるいは LOB または XML に基づく特殊タイプでなければなりません。

SPECIFIC PROCEDURE *specific-name*
プロシージャーを特定名によって識別します。特定名 は、現行サーバーに存在している特定のプロシージャーを識別していなければなりません。

RESTRICT
プロシージャーが、SQL 関数、SQL プロシージャー、または SQL トリガーで参照される場合には、除去できないことを指定します。

SCHEMA *schema-name*
除去したいスキーマを識別します。スキーマ名 は、現行サーバーに存在しているスキーマを識別していなければなりません。

除去したスキーマは削除されます。スキーマ内の各オブジェクトは、指定の削除オプション (CASCADE、RESTRICT、または、どちらもなし) を使用して該当する DROP ステートメント実行した場合と同様に除去されます。これらのオブジェクトに従属するオブジェクトの扱いについては、これらのオブジェクト・タイプの DROP の説明を参照してください。

DROP SCHEMA は、コミット・レベルが *NONE の場合にのみ有効です。

CASCADE も RESTRICT も指定しない場合
スキーマが別のスキーマ内の関数、パッケージ、プロシージャー、プログラム、表、マスク、権限、トリガー、または変数で参照されている場合も、スキーマは除去されることを指定します。

CASCADE
このスキーマ内のオブジェクトとこのスキーマを参照しているトリガーを、すべて除去することを指定します。ステートメントの権限 ID にセキュリティー管理者権限がある場合は、このスキーマを参照する、別のスキーマ内のすべてのマスクおよび許可が除去されます。

RESTRICT
スキーマが別のスキーマ内の SQL トリガー、マスク、または権限で参照されている場合、またはカタログ・ビュー、ジャーナル、ジャーナル・レシーバー以外の SQL オブジェクトがスキーマに含まれる場合は、そのスキーマを除去できないことを指定します。

SEQUENCE *sequence-name*
除去したいシーケンスを識別します。シーケンス名 は、現行サーバーに存在するシーケンスを示すものでなければなりません。

RESTRICT
シーケンスが SQL トリガー、関数、プロシージャー、または変数で参照されている場合、そのシーケンスは除去できないことを指定します。

TABLE *table-name*
除去したい表を識別します。 *table-name* は現行サーバーに存在する基本表を識別するものでなければならず、カタログ表や、システム期間テンポラル表の履歴表を識別するものであってはなりません。

指定した表は、スキーマから除去されます。その表に関する特権、制約、索引、マスク、権限、およびトリガーもすべて除去されます。

指定された表を参照している別名は、除去されません。

CASCADE も RESTRICT も指定しない場合
表が制約、索引、トリガー、マスク、権限、ビュー、またはマテリアライズ照会表で参照されていても、表は除去されることを指定します。その表を参照する索引、ビュー、およびマテリアライズ照会表は、ステートメントの権限 ID が明示的にそれらのオブジェクトに対する特権を持っていなくても、すべて除去されます。表がシステム期間テンポラル表である場合、履歴表も除去されます。その表がマスクまたは許可で参照されている場合、ステートメントの権限 ID にセキュリティー管理者権限がないと除去は失敗します。

CASCADE
表が制約、索引、トリガー、変数、マスク、権限、ビュー、XSR オブジェクト、またはマテリアライズ照会表で参照されていても、表は除去されることを指定します。その表を参照する制約、索引、トリガー、変数、ビュー、XSR オブジェクト、およびマテリアライズ照会表は、ステートメントの権限 ID が明示的にそれらのオブジェクトに対する特権を持っていなくても、すべて除去されます。表がシステム期間テンポラル表である場合、履歴表も除去されます。ステートメントの権限 ID にセキュリティー管理者権限がある場合は、その表を参照するすべてのマスクおよび許可が除去されます。

RESTRICT
表が制約、索引、マスク、権限、トリガー、変数、ビュー、XSR オブジェクト、またはマテリアライズ照会表で参照されている場合や、表がシステム期間テンポラル表である場合は表を除去できないことを指定します。

TRIGGER *trigger-name*
除去したいトリガーを識別します。トリガー名 は、現行サーバーに存在しているトリガーを識別していなければなりません。

指定したトリガーは、スキーマから除去されます。トリガーが SQL トリガーの場合、そのトリガーに関連したプログラム・オブジェクトもスキーマから削除されます。

トリガー名 がビューに対して INSTEAD OF トリガーを指定する場合、そのビューに対する更新を行うことにより、他のトリガーはそのトリガーに従属できます。

TYPE *distinct-type-name* または *array-type-name*
除去したいタイプを識別します。特殊タイプ名 または配列タイプ名 は、現行サーバーに存在する特殊タイプまたは配列タイプを識別する必要があります。指定したタイプは、スキーマから削除されます。

CASCADE も **RESTRICT** も指定しない場合
制約、索引、マスク、権限、シーケンス、表、変数、およびビューがタイプを参照している場合、そのタイプは除去できないことを指定します。

次の DROP ステートメントは、除去されるタイプのパラメーターや戻り値を持つすべてのプロシージャーまたは関数 R で、有効に実行されます。

```
DROP ROUTINE R
```

次の DROP ステートメントは、除去されるタイプを参照するすべてのトリガー T で、有効に実行されます。

```
DROP TRIGGER T
```

このステートメントをカスケードにして、従属する関数やプロシージャーを除去することも可能です。これらの関数やプロシージャーのすべてが、タイプと従属関係にあるために除去されるリストに入っている場合、タイプの除去は正常に行われます。

CASCADE
タイプが制約、関数、索引、プロシージャー、シーケンス、表、トリガー、変数、またはビューで参照されていても、タイプは除去されることを指定します。そのタイプを参照する制約、関数、索引、プロシージャー、シーケンス、表、トリガー、変数、およびビューは、すべて除去されます。カスケード処理が行われるのは、そのタイプが、ルーチン・パラメーター・タイプ、シーケンス・タイプ、変数タイプ、および列タイプを定義するために使用されている場合に限られます。ステートメントの権限 ID にセキュリティー管理者権限がある場合は、そのタイプを参照するすべてのマスクおよび許可が除去されます。

RESTRICT
タイプが制約、関数 (そのタイプの作成時に作成された関数以外の)、索引、プロシージャー、シーケンス、表、マスク、権限、トリガー、変数、またはビューで参照されている場合、そのタイプは除去できないことを指定します。制限チェックが行われるのは、タイプが、ルーチン・パラメーター・タイプ、シーケンス・タイプ、変数タイプ、および列タイプを定義するために使用されている場合に限られます。

VARIABLE *variable-name*
除去したい変数を識別します。この変数名 は、現行サーバーに存在している変数を識別していなければなりません。指定した変数は、スキーマから除去されます。

CASCADE も **RESTRICT** も指定しない場合
トリガー、プロシージャー、関数、マスク、権限、ビュー、または別の変数で参照されていても、変数は除去されることを指定します。その変数を参照するすべての表およびビューも削除されます。その変数がマスクまたは許可で参照されている場合、ステートメントの権限 ID にセキュリティー管理者権限がないと除去は失敗します。

CASCADE
表、トリガー、プロシージャー、関数、ビュー、または別の変数で参照されていても、変数は除去されることを指定します。その変数を参照するすべての表、トリガー、プロシージャー、関数、ビュー、変数も削除されます。ステートメントの権限 ID にセキュリティー管理者権限がある場合は、その変数を参照するすべてのマスクおよび許可が除去されます。

RESTRICT
変数が、表、トリガー、プロシージャー、関数、マスク、許可、ビュー、または別の変数で参照されている場合、その変数は除去できないことを指定します。

VIEW *view-name*
除去したいビューを識別します。このビュー名 は、現行サーバーに存在しているビューを識別していなければなりませんが、カタログ・ビューを識別するものであってはなりません。

指定したビューは、スキーマから除去されます。ビューが削除されると、そのビューに関する特権およびトリガーはすべて削除されます。

CASCADE も RESTRICT も指定しない場合
トリガー、マテリアライズ照会表、マスク、権限、または別のビューで参照されていても、ビューは除去されることを指定します。そのビューを参照するビューおよびマテリアライズ照会表は、すべて除去されます。そのビューがマスクまたは許可で参照されている場合、ステートメントの権限 ID にセキュリティー管理者権限がないと除去は失敗します。

CASCADE
トリガー、変数、マテリアライズ照会表、または別のビューで参照されていても、ビューは除去されることを指定します。そのビューを参照するトリガー、変数、マテリアライズ照会表、およびビューはすべて除去されます。ステートメントの権限 ID にセキュリティー管理者権限がある場合は、そのビューを参照するすべてのマスクおよび許可が除去されます。

RESTRICT
トリガー、変数、マスク、権限、マテリアライズ照会表、または別のビューで参照されている場合、ビューは除去できないことを指定します。

XSROBJECT *xsrobject-name*
除去したい XSR オブジェクトを識別します。この*XSR* オブジェクト名 は、現行サーバーに存在しているXSR オブジェクトを示すものでなければなりません。指定した XSR オブジェクトは削除されます。

注

除去の影響**:** オブジェクトを除去すると、そのオブジェクトの記述も必ずカタログから除去されます。オブジェクトが関数、パッケージ、プロシージャー、プログラム、トリガー、または変数で参照されている場合、そのオブジェクトを参照するアクセス・プランは、アクセス・プランが次回に使用されるときに、暗黙的に再準備されます。そのときにそのオブジェクトが存在しないと、エラーが戻されます。

従属関係**:** 直接または間接的にオブジェクトが除去される場合、除去されるオブジェクトに従属する他のオブジェクトも除去されることがあります。下記のセマンティ

クスによって、従属オブジェクトが従属しているオブジェクト (すなわち、基礎オブジェクト) がドロップされたときに、従属オブジェクトに何が起こるかが決定されます。

このリストには、以下の 3 つの異なるタイプの従属関係が示されています。

D 従属オブジェクトはドロップされます。

A 自動再評価が必要です。データベース・マネージャーは、オブジェクトが参照されたときにオブジェクトの再評価を試みます。

R DROP ステートメントは失敗します。

表 *87.* 従属関係のあるオブジェクトをドロップしたときの影響

DROP ステートメント	**ALIAS**	**CONSTRAINT**	**FUNCTION**	**INDEX**	**MASK**	**PERMISSION**	**PROCEDURE**	**SCHEMA**	**SEQUENCE**	**TABLE**	**TRIGGER**	**TYPE**	**VARIABLE**	**VIEW**
DROP ALIAS														
DROP FUNCTION			R[1]		A	A	A			A	A		A	A
DROP FUNCTION RESTRICT			R		R	R	R			R	R		R	R
DROP INDEX														
DROP MASK														
DROP PACKAGE														
DROP PERMISSION														
DROP PROCEDURE			A				A				A			
DROP PROCEDURE RESTRICT			R				R				R			
DROP SCHEMA[2]			A		A	A	A			A	A		A	A
DROP SCHEMA CASCADE[3]			A		D	D	A			A	D		A	A
DROP SCHEMA RESTRICT[4]			A		R	R	A			A	R		A	A
DROP SEQUENCE			A				A				A		A	
DROP SEQUENCE RESTRICT			R				R				R		R	
DROP TABLE		D	A	D	A	A	A			D	A		A	D
DROP TABLE CASCADE		D	A	D	D	D	A			D	D		D	D
DROP TABLE RESTRICT		R	A	R	R	R	A			R	R		R	R
DROP TRIGGER														
DROP TYPE		R	D[5]	R	R	R	D		R	R	D		R	R
DROP TYPE CASCADE		D	D	D	D	D	D		D	D	D		D	D
DROP TYPE RESTRICT		R	R	R	R	R	R		R	R	R		R	R
DROP VARIABLE			A		A	A	A			D	A		A	D
DROP VARIABLE CASCADE			D		D	D	D			D	D		D	D
DROP VARIABLE RESTRICT			R		R	R	R			R	R		R	R
DROP VIEW			A		A	A				D	A		A	D
DROP VIEW CASCADE			A		D	D	A			D	D		D	D
DROP VIEW RESTRICT			A		R	R	A			R	R		R	R
DROP XSROBJ			A		A	A	A				A		A	A

表 *87.* 従属関係のあるオブジェクトをドロップしたときの影響 (続き)

DROP ステートメント	ALIAS	CONSTRAINT	FUNCTION	INDEX	MASK	PERMISSION	PROCEDURE	SCHEMA	SEQUENCE	TABLE	TRIGGER	TYPE	VARIABLE	VIEW

注:

1. 除去する関数を他のユーザー定義関数がソースとしている場合、この関数を除去することはできません。
2. DROP SCHEMA は、スキーマ内のすべてのオブジェクトを除去します。このスキーマを参照する、他のスキーマ内にあるオブジェクトは、変更されることはなく、自動再評価を必要とします。
3. DROP SCHEMA CASCADE は、スキーマ内のすべてのオブジェクトを除去します。このスキーマを参照する、他のスキーマ内にあるトリガーは除去されます。このスキーマを参照する、他のスキーマ内にあるオブジェクトは、変更されることはなく、自動再評価を必要とします。
4. DROP SCHEMA RESTRICT は、スキーマ内に、SQL カタログ・ビュー、ジャーナル、およびジャーナル・レシーバー以外のオブジェクトが存在している場合は失敗します。このスキーマを参照する、他のスキーマ内にあるトリガー以外のオブジェクトは、変更されることはなく、自動再評価を必要とします。
5. このタイプとして定義されたパラメーターまたは戻り値がある関数またはプロシージャーは除去されます。

代替構文**:** 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード SYNONYM は、ALIAS の同義語として使用することができます。
- キーワード DATA TYPE または DISTINCT TYPE を TYPE の同義語として使用することができます。
- PACKAGE の同義語として、キーワード PROGRAM を使用することができます。
- キーワード COLLECTION を SCHEMA の同義語として使用できます。

例

例 *1:*MY_IN_TRAY という名前の表を削除します。この表に関して作成されているビューまたは索引がある場合は、この除去はできません。

```
DROP TABLE MY_IN_TRAY RESTRICT
```

例 *2:* MA_PROJ という名前のビューを削除します。

```
DROP VIEW MA_PROJ
```

例 *3:* PERS.PACKA という名前のパッケージを削除します。

```
DROP PACKAGE PERS.PACKA
```

例 *4:* 特殊タイプ DOCUMENT が現在使用されていなければ、それを除去します。

```
DROP DISTINCT TYPE DOCUMENT RESTRICT
```

例 *5:* 自分が SMITH であり、ATOMIC_WEIGHT が、スキーマ CHEM 内でその名前を持つ唯一の関数であると想定します。 ATOMIC_WEIGHT を除去します。

```
DROP FUNCTION CHEM.ATOMIC_WEIGHT RESTRICT
```

例 *6:* 除去する関数インスタンスを識別するために関数シグニチャーを使用して、CENTER という名前を除去します。

```
DROP FUNCTION CENTER (INTEGER, FLOAT) RESTRICT
```

例 *7:* 除去する関数インスタンスを識別するために特定名を使用して、CENTER を除去します。

```
DROP SPECIFIC FUNCTION JOHNSON.FOCUS97
```

例 *8:* プロシージャー OSMOSIS がスキーマ BIOLOGY 内にあると仮定します。OSMOSIS を除去します。

```
DROP PROCEDURE BIOLOGY.OSMOSIS
```

例 *9:* トリガー BONUS がデフォルトのスキーマに存在するとします。BONUS を除去します。

```
DROP TRIGGER BONUS
```

END DECLARE SECTION

END DECLARE SECTION ステートメントは、SQL 宣言セクションの終わりを示します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、実行可能ステートメントではありません。RPG、Java、または REXX では指定しないでください。

権限

権限は不要です。

構文

説明

END DECLARE SECTION ステートメントは、ホスト言語の規則に従って宣言を置ける場所であれば、アプリケーション・プログラム内のどこにでもコーディングできます。このステートメントは、SQL 宣言セクションの終わりを示すのに使用されます。 SQL 宣言セクションは、BEGIN DECLARE SECTION ステートメントで開始します。 BEGIN DECLARE SECTION ステートメントの詳細については、1013 ページの『BEGIN DECLARE SECTION』を参照してください。

BEGIN DECLARE SECTION と END DECLARE SECTION ステートメントは、対にして使用しなければなりません。また、これらのステートメントをネストすることはできません。

例

END DECLARE SECTION ステートメントの使用例については、 1013 ページの『BEGIN DECLARE SECTION』を参照してください。

EXECUTE

EXECUTE ステートメントは、準備済み SQL ステートメントを実行します。

呼び出し

このステートメントは、アプリケーション・プログラム、SQL 関数、 SQL プロシージャー、またはトリガー内にのみ組み込むことができます。これは実行可能ステートメントですが、動的に準備することはできません。Java では指定できません。

権限

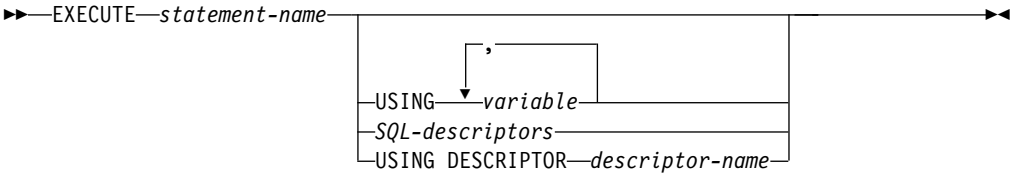
ステートメントでグローバル変数を参照する場合は、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別されるグローバル変数に対して、
  - そのグローバル変数に対する READ 特権
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

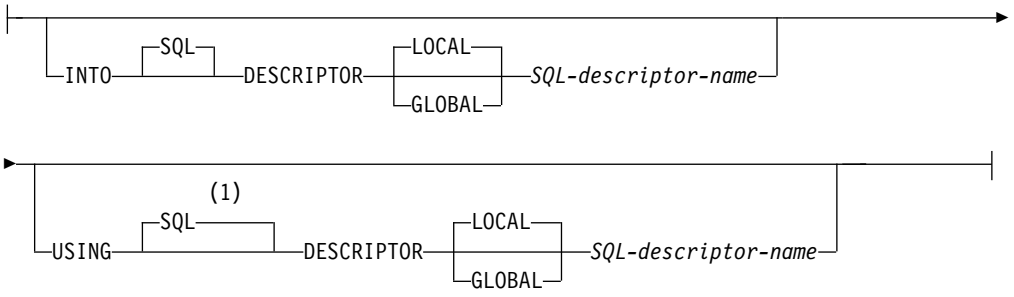
権限の規則は、EXECUTE によって指定される SQL ステートメントに対して定義されている規則が適用されます。例えば、EXECUTE を使用して INSERT ステートメントを実行する場合に適用される権限規則については、 INSERT についての説明を参照してください。

プログラムの作成時点の CRTSQLxxx コマンドに DYNUSRPRF(*OWNER) が指定されていた場合を除き、ステートメントの権限 ID は、実行時の権限 ID です。詳しくは、 78 ページの『権限 ID と権限名』を参照してください。

構文



SQL-descriptors:



注:

1 SQL 記述子が USING 文節で指定されて INTO 文節が指定されない場合、USING DESCRIPTOR は許可されないため USING SQL DESCRIPTOR を指定しなければなりません。

説明

statement-name

実行する準備済みステートメントを識別します。EXECUTE ステートメントを実行するときには、現行サーバーにある準備済みステートメントの名前を指定する必要があります。準備済みステートメントには、選択ステートメントを指定してはなりません。

USING

この次に、変数のリストを指定することを示します。準備済みステートメント内のパラメーター・マーカー (疑問符) は、このキーワードの次に指定した変数の値に置き換えられます。パラメーター・マーカーの説明については、 1603 ページの『PREPARE』を参照してください。準備済みステートメントにパラメーター・マーカーが含まれている場合は、必ず USING 文節を使用してください。ステートメントにパラメーター・マーカーが入っていなければ、USING は無視されます。

変数,...

1 つ以上のホスト構造体、あるいは変数を指定します。それらの構造体や変数は、ホスト構造体および変数の宣言の規則に従ってプログラムで宣言されていなければなりません。ホスト構造体に対する参照は、その個々の変数に対する参照に置き換えられます。リストされた変数の数は、準備済みステートメントのパラメーター・マーカーの数と同じでなければなりません。 *n* 番目の変数は、準備済みステートメントの *n* 番目のパラメーター・マーカーに対応します。

現在の接続がローカル接続である (DRDA 接続ではない) 場合のみ、グローバル変数が使用できます。

SQL-descriptors

INTO

EXECUTE ステートメントで使用される出力変数の有効な記述子を含む SQL 記述子を識別します。この文節は、CALL または VALUES INTO ステートメントでのみ有効です。 EXECUTE ステートメントを実行する前に、ALLOCATE DESCRIPTOR ステートメントを使用して記述子を割り振らなければなりません。

LOCAL

記述子の名前の有効範囲はプログラム呼び出しのローカルであることを指定します。

GLOBAL

記述子の名前の有効範囲は SQL セッション全体であることを指定します。

SQL-descriptor-name

SQL 記述子の名前を指定します。名前は、指定の有効範囲を持つ既存の記述子を識別するものでなければなりません。

SQL 記述子に入る情報の説明については、 1476 ページの『GET DESCRIPTOR』を参照してください。

USING
EXECUTE ステートメントで使用される入力変数の有効な記述子を含む SQL 記述子を識別します。 EXECUTE ステートメントを実行する前に、ALLOCATE DESCRIPTOR ステートメントを使用して記述子を割り振らなければなりません。

LOCAL
記述子の名前の有効範囲はプログラム呼び出しのローカルであることを指定します。情報は、このローカル有効範囲で既知の記述子から戻されます。

GLOBAL
記述子の名前の有効範囲は SQL セッション全体であることを指定します。情報は、同じデータベース接続を使用して実行するどのプログラムにも既知の記述子から戻されます。

SQL-descriptor-name
SQL 記述子の名前を指定します。名前は、指定の有効範囲を持つ既存の記述子を識別するものでなければなりません。

SQL 記述子内の情報の説明については、 1688 ページの『SET DESCRIPTOR』を参照してください。

DESCRIPTOR *descriptor-name*
SQLDA を識別します。この SQLDA には、変数の有効な記述が入っていなければなりません。

EXECUTE ステートメントの処理に先立って、ユーザーは SQLDA の以下のフィールドをセットしておく必要があります。 (REXX の場合は、規則が異なります。詳しくは、「組み込み SQL プログラミング」トピック集を参照してください。)

- SQLN (SQLDA に用意する SQLVAR のオカレンスの数を示します。)
- SQLDABC (SQLDA 用に割り振る記憶域のバイト数を示します。)
- SQLD (ステートメントを処理するときに SQLDA で使用する変数の個数を指示します。)
- SQLVAR の各オカレンス (変数の属性を指示します。)

SQLDA の記憶域は、SQLVAR のオカレンスをすべて収容するのに十分な大きさで割り振らなければなりません。 LOB または特殊タイプが結果の中に存在する場合、各パラメーター・マーカーごとに追加の SQLVAR 項目が必要です。 SQLVAR の説明、SQLVAR オカレンスの回数を判別する方法など、SQLDA について詳しくは、 1880 ページの『必要な SQLVAR オカレンスの数の決定』を参照してください。

SQLD には、ゼロ以上で SQLN 以下の値をセットしなければなりません。この値は、準備済みステートメント内のパラメーター・マーカーの個数と同じでなければなりません。 SQLDA によって *n* 番目に記述される変数は、準備済みステートメントの *n* 番目のパラメーター・マーカーに対応します。

RPG/400 には、ポインターを設定する機能が用意されていないことに注意してください。 SQLDA はポインターを使用して適切な変数を見つけるので、ユーザーは、RPG/400 アプリケーションの外側でそのようなポインターを設定する必要があります。

注

パラメーター・マーカーの置換**:** 準備済みステートメントのパラメーター・マーカーは、実際には、そのステートメントを実行する前に対応する変数によって置き換えられます。パラメーター・マーカーの置き換えは、変数の値をソースとし、データベース・マネージャー内部の変数をターゲットとする割り当て演算によって処理されます。タイプ付きパラメーター・マーカーの場合、ターゲット変数の属性は、CAST によって指定されたものになります。タイプ無しパラメーター・マーカーの場合、ターゲット変数の属性は、パラメーター・マーカーのコンテキストによって決まります。パラメーター・マーカーに影響を及ぼす規則については、 1612 ページの表 113を参照してください。

ここで、パラメーター・マーカーを P、そのパラメーター・マーカーに対応する変数を V としましょう。V の値は、ストレージ割り当て規則に基づいて P のターゲット変数に割り当てられます (113 ページの『割り当ておよび比較』を参照してください)。したがって、次のことがいえます。

- V は、ターゲットと互換性のあるものでなければなりません。
- V が数値ならば、V の整数部の絶対値は、ターゲットの整数部の絶対値の最大を超えてはなりません。
- V の属性がターゲットの属性と一致しない場合は、ターゲットの属性に合わせて値が変換されます。
- ターゲットに NULL を入れることができない場合は、V の値は NULL であってはなりません。

ただし、ストレージ割り当て規則とは異なる以下のような動作もあります。

- V がストリングで、その長さがターゲットの長さ属性より大きければ、V の値は途中で切り捨てられます (エラーは出されません)。

準備されたステートメントを実行するときに P の代わりに使用される値は、 P のターゲット変数の値です。例えば、V が CHAR(6) で、ターゲットが CHAR(8) の場合、P の代わりに使用される値は、 V の値に 2 つのブランクが埋め込まれた値になります。

例

この例は、COBOL プログラムの一部です。パラメーター・マーカーが指定されている INSERT ステートメントが、どのように準備され、実行されるかを示しています。

```
EXEC SQL  BEGIN DECLARE SECTION  END-EXEC.
  77 EMP              PIC X(6).
  77 PRJ              PIC X(6).
  77 ACT              PIC S9(4) COMP-4.
  77 TIM              PIC S9(3)V9(2).
  01 HOLDER.
     49  HOLDER-LENGTH    PIC S9(4) COMP-4.
     49  HOLDER-VALUE     PIC X(80).
```

```
  EXEC SQL  END DECLARE SECTION  END-EXEC.
  .
  .
  .
  MOVE 70 TO HOLDER-LENGTH.
  MOVE "INSERT INTO EMPPROJACT (EMPNO, PROJNO, ACTNO, EMPTIME)
-              "VALUES (?, ?, ?, ?)" TO HOLDER-VALUE.
  EXEC SQL  PREPARE MYINSERT FROM :HOLDER  END-EXEC.

  IF SQLCODE = 0
    PERFORM DO-INSERT THRU END-DO-INSERT
  ELSE
    PERFORM ERROR-CONDITION.

  DO-INSERT.
    MOVE "000010" TO EMP.
    MOVE "AD3100" TO PRJ.
    MOVE 160      TO ACT.
    MOVE .50      TO TIM.
    EXEC SQL  EXECUTE MYINSERT USING :EMP, :PRJ, :ACT, :TIM  END-EXEC.
  END-DO-INSERT.
  .
  .
  .
```

EXECUTE IMMEDIATE

EXECUTE IMMEDIATE ステートメントは、PREPARE ステートメントと EXECUTE ステートメントの基本機能を結合したものです。このステートメントは、変数もパラメーター・マーカーも含まない SQL ステートメントを準備し、実行するのに使用することができます。

EXECUTE IMMEDIATE ステートメントは、以下の処理を行います。

- 文字ストリング形式の SQL ステートメントをもとにして、そのステートメントの実行可能形式を準備する
- その SQL ステートメントを実行する

呼び出し

このステートメントは、アプリケーション・プログラム、SQL 関数、 SQL プロシージャー、またはトリガー内にのみ組み込むことができます。これは実行可能ステートメントですが、動的に準備することはできません。Java では指定できません。

権限

ステートメントでグローバル変数を参照する場合は、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別されるグローバル変数に対して、
  - そのグローバル変数に対する READ 特権
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

権限の規則は、EXECUTE IMMEDIATE により指定される SQL ステートメントに対する規則が適用されます。例えば、EXECUTE IMMEDIATE を使用して INSERT ステートメントを実行する場合に適用される権限規則については、 1556 ページの『INSERT』を参照してください。

プログラムの作成時点の CRTSQLxxx コマンドに DYNUSRPRF(*OWNER) が指定されていた場合を除き、ステートメントの権限 ID は、実行時の権限 ID です。詳しくは、 78 ページの『権限 ID と権限名』を参照してください。

構文

説明

variable
変数を指定します。この変数は、文字ストリングまたは Unicode グラフィック変数を宣言する規則に従って宣言されていなければなりません。この変数に、標識変数を指定してはなりません。

expression
集約関数または列名を含まない、 196 ページの『式』で説明されているタイプ

の *expression*。これは、文字ストリングまたは Unicode グラフィック・ストリングの値を戻す必要があります。*expression* の中に変数が指定される場合、その変数の CCSID は 65535 であってはなりません。[104]

指定された変数または式の値は、ステートメント・ストリング と呼ばれます。

ステートメント・ストリングは、以下の SQL ステートメントのいずれかである必要があります。[106]

ALTER	LOCK TABLE	SET CURRENT IMPLICIT XMLPARSE OPTION
CALL	MERGE	SET CURRENT TEMPORAL SYSTEM_TIME
COMMENT	REFRESH TABLE	SET ENCRYPTION PASSWORD
COMMIT	RELEASE SAVEPOINT	SET PATH
コンパウンド (動的)	RENAME	SET SCHEMA
CREATE	REVOKE	SET SESSION AUTHORIZATION
DECLARE GLOBAL TEMPORARY TABLE	ROLLBACK	SET TRANSACTION
DELETE	SAVEPOINT	SET 変数[105]
DROP	SET CURRENT DEBUG MODE	TRANSFER OWNERSHIP
GRANT	SET CURRENT DECFLOAT ROUNDING MODE	TRUNCATE
INSERT	SET CURRENT DEGREE	UPDATE
LABEL		

ステートメント・ストリングは、次のようなストリングであってはなりません。

- EXEC SQL で始める。
- END-EXEC またはセミコロンで終了する。
- 変数への参照を含むストリング。グローバル変数を使用できます。
- パラメーター・マーカーを含むストリング。

EXECUTE IMMEDIATE ステートメントを実行すると、指定したステートメント・ストリングが解析され、エラーの有無が検査されます。 SQL ステートメントとして正しくない場合は、そのステートメントは実行されず、実行を妨げているエラー条件が独立の SQLSTATE および SQLCODE で報告されます。 SQL ステートメントとして正しくても、ステートメントの実行時にエラーが発生すると、そのエラー条件が独立の SQLSTATE および SQLCODE に報告されます。エラーに関する追加情報は、SQL 診断領域 (または SQLCA) から検索できます。

注記

パフォーマンスの考慮**:** 同じ SQL ステートメントを複数回実行する場合には、EXECUTE IMMEDIATE ステートメントより、PREPARE ステートメントと EXECUTE ステートメントを使った方が効率的です。

---

104. PL/I プログラム内では、PL/I ストリング式も指定できます。

105. SET 変数ステートメントのターゲットは、グローバル変数である必要があります。

106. 選択ステートメント は許可されません。選択ステートメントを動的に処理するには、PREPARE、DECLARE CURSOR、および OPEN ステートメントを使用します。

例

C を使用して、変数 Qstring 内の SQL ステートメントを実行します。

```
void main ()
  {

    EXEC SQL  BEGIN DECLARE SECTION  END-EXEC.

    char Qstring[100] = "INSERT INTO WORK_TABLE SELECT * FROM EMPPROJACT
     WHERE ACTNO >= 100";

    EXEC SQL  END DECLARE SECTION  END-EXEC.
    EXEC SQL  INCLUDE SQLCA;
  .
  .
  .
    EXEC SQL EXECUTE IMMEDIATE :Qstring;

    return;
  }
```

FETCH

FETCH ステートメントは、カーソルを結果表の行に位置付けます。FETCH ステートメントは、ゼロ、1 つ、または複数の行を戻すこともでき、戻された行の値を変数に割り当てます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。これは実行可能ステートメントですが、動的に準備することはできません。REXX プロシージャーでは、複数行の取り出しはできません。

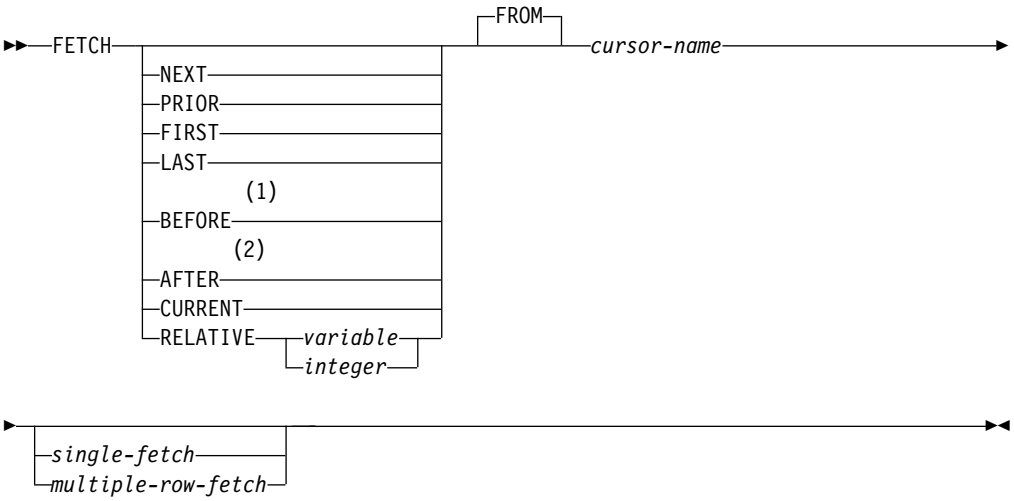
権限

カーソルの使用に必要な許可については、 1353 ページの『DECLARE CURSOR』を参照してください。

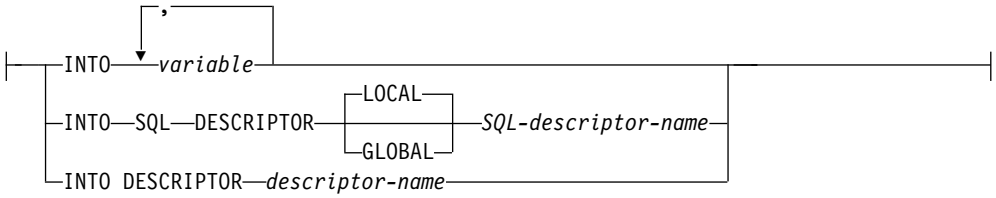
INTO 変数リスト内にグローバル変数が指定されている場合、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つが含まれていなければなりません。

- グローバル変数に対する WRITE 特権
- データベース管理者権限

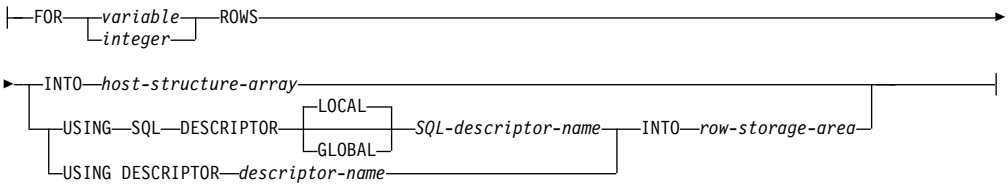
構文



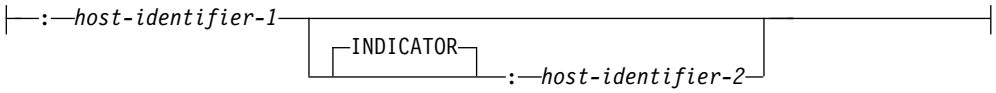
single-fetch:



multiple-row-fetch:



row-storage-area:



注:

1 BEFORE が指定されている場合には、単一取り出し、または複数行取り出しを指定することはできません。

2 AFTER が指定されている場合には、単一取り出し、または複数行取り出し を指定することはできません。

説明

NEXT、PRIOR、FIRST、LAST、BEFORE、AFTER、CURRENT、および RELATIVE の各キーワードは、カーソルの新しい位置を指定します。これらのキーワードのうち、SCROLL が宣言されていないカーソルに使用できるのは、 NEXT のみです。

NEXT
現行カーソル位置から見て、結果表の次の行にカーソルを位置付けます。他のカーソルの方向付けが指定されていない場合には、NEXT がデフォルト値になります。

PRIOR
現行カーソル位置から見て、結果表の前の行にカーソルを位置付けます。

FIRST
結果表の先頭の行にカーソルを位置付けます。

LAST
結果表の最終行にカーソルを位置付けます。

BEFORE
結果表の先頭行の前にカーソルを位置付けます。

AFTER
結果表の最終行の後にカーソルを位置付けます。

CURRENT
カーソルの位置は変えずに、現行カーソル位置を維持します。カーソルが DYNAMIC SCROLL として宣言されている場合、現在行が変更された結果、結果表のソート順序の中での位置が変化していると、エラーが戻されます。

RELATIVE
変数 または整数 が、整数値 *k* に割り当てられます。RELATIVE によって決められるカーソルの結果表内の行の位置は、*k*>0 の場合は現在行の *k* 行後、*k*<0 の場合は現在行の *k* 行前になります。変数 を指定する場合には、位取りがゼロの数値変数を指定しなければならず、また標識変数を入れることはできません。

表 *88.* 同義のスクロール指定

指定	代替
RELATIVE +1	NEXT
RELATIVE -1	PRIOR
RELATIVE 0	CURRENT

FROM
このキーワードは、文脈を分かりやすくするために使用するものです。スクロール位置オプションを指定する場合には、このキーワードが必要です。スクロール・オプションを指定しない場合には、FROM キーワードはオプショナルです。

cursor-name
取り出し操作で使用するカーソルを識別します。 *cursor-name* では、宣言済みカーソルを指定する必要があります (DECLARE CURSOR ステートメントの 1354 ページの『説明』を参照してください)。Java で使用する場合は、SQLJ イテレーターのインスタンスを指定しなければなりません。FETCH ステートメントの実行時点で、指定したカーソルはオープン状態でなければなりません。

単一行取り出し 文節または複数行取り出し 文節が指定されていない場合、データはユーザーに戻されません。ただし、カーソルは位置付けられ、行ロックが掛けられます。ロックの詳細については、 32 ページの『分離レベル』を参照してください。

single-fetch

INTO *variable*,...
1 つ以上のホスト構造体または変数を識別しますが、それらはホスト構造体および変数の宣言に関する規則に従って宣言されているものでなければなりません。INTO の操作形式では、ホスト構造体は、その個々の変数に対する参照に置き換えられます。結果の行の最初の値がリストの最初の変数に割り当てられ、2 番目の値が 2 番目の変数に割り当てられます。以下同様です。

現在の接続がローカル接続である (DRDA 接続ではない) 場合のみ、グローバル変数が使用できます。

INTO SQL DESCRIPTOR *SQL-descriptor-name*

FETCH ステートメントで使用される出力変数の有効な記述子を含む SQL 記述子を識別します。 FETCH ステートメントを実行する前に、ALLOCATE DESCRIPTOR ステートメントを使用して記述子を割り振らなければなりません。

LOCAL

記述子の名前の有効範囲はプログラム呼び出しのローカルであることを指定します。

GLOBAL

記述子の名前の有効範囲は SQL セッション全体であることを指定します。

SQL-descriptor-name

SQL 記述子の名前を指定します。名前は、指定の有効範囲を持つ既存の記述子を識別するものでなければなりません。

SQL 記述子内の情報の説明については、 1688 ページの『SET DESCRIPTOR』を参照してください。

INTO DESCRIPTOR *descriptor-name*

ゼロ個または 1 つ以上の変数の有効な記述が入っている SQLDA を識別します。

ユーザーは、FETCH ステートメントを処理する前に、SQLDA の以下のフィールドをセットしておく必要があります。(REXX の場合は、規則が異なります。詳しくは、「組み込み SQL プログラミング」のトピック集を参照してください。)

- SQLN (SQLDA に用意する SQLVAR のオカレンスの数を示します。)
- SQLDABC (SQLDA 用に割り振る記憶域のバイト数を示します。)
- SQLD (ステートメントを処理するときに、SQLDA で使用する変数の個数を指示します。)
- SQLVAR の各オカレンス (変数の属性を指示します。)

SQLDA の記憶域は、SQLVAR のオカレンスをすべて収容するのに十分な大きさで割り振らなければなりません。したがって、SQLDABC の値は、16 + SQLN*(80) よりも大きいか、または等しくなければなりません。ここで、80 は SQLVAR の 1 つのオカレンスの長さです。LOB が指定された場合には、各パラメーター・マーカーごとに 2 つの SQLVAR 項目が必要であり、SQLN はパラメーター・マーカー数の 2 倍をセットしなければなりません。

SQLD には、ゼロ以上で SQLN 以下の値をセットしなければなりません。詳しくは、 1877 ページの『付録 D. SQLDA (SQL 記述子域)』を参照してください。

Java プログラムでは、FETCH ステートメントで USING DESCRIPTOR 文節を使用できません。

multiple-row-fetch

FOR *variable* または *integer* **ROWS**

取り出す行の数を表す整数値に対して変数 または整数 を評価します。変数 を指定する場合には、位取りがゼロの数値変数を指定しなければならず、また標識変数を入れることはできません。変数をグローバル変数にすることはできませ

ん。行の合計サイズ (LOB を除く) は 16 M 未満でなければなりません。カーソルは、方向付けキーワード (例えば NEXT) の指定する行に置かれ、その行が取り出されます。それから、次の行が取り出され (表内で順方向に移動)、指定された行数が取り出されるかカーソルの終わりに達するまでこれが繰り返されます。取り出し操作の後、カーソルは最後に取り出された行に置かれます。

例えば、FETCH PRIOR FROM C1 FOR 3 ROWS を実行すると、前の行、現在行、および次の行が、この順序で戻されます。カーソルは次の行に置かれます。FETCH RELATIVE -1 FROM C1 FOR 3 ROWS でも、同じ結果が戻されます。これに対して FETCH FIRST FROM C1 FOR :x ROWS では、先頭の *x* 行が戻され、カーソルは番号 *x* の行に置かれたままとなります。

複数行取り出し が正しく実行されると、次の 3 つのステートメント情報項目が SQL 診断領域 (または SQLCA) で使用可能になります。

- ROW_COUNT (または SQLCA の SQLERRD(3)) には、取り出された行の数を示す値が設定されます。
- DB2_ROW_LENGTH (または SQLCA の SQLERRD(4)) には、取り出された行の長さが入ります。
- DB2_LAST_ROW (または SQLCA の SQLERRD(5)) には、取り出された行が最後の行である場合、+100 が設定されます。[107]

INTO *host-structure-array*
ホスト構造体配列 は、ホスト構造体の宣言に関する規則に従って定義されているホスト構造体の配列を識別します。

すなわち、配列の最初の構造は最初の行に対応し、配列の 2 番目の構造は 2 行目に対応しています。以下も同じです。さらに、行の最初の値が構造体内の最初の項目に対応し、行の 2 番目の値が構造体の 2 番目の項目に対応するというように、これも順番に対応しています。取り出す行数は、ホスト構造体配列の次元以下でなければなりません。

USING SQL DESCRIPTOR *SQL-descriptor-name*
SQL 記述子を識別します。

LOCAL
記述子の名前の有効範囲はプログラム呼び出しのローカルであることを指定します。

GLOBAL
記述子の名前の有効範囲は SQL セッション全体であることを指定します。

SQL-descriptor-name
SQL 記述子の名前を指定します。名前は、指定の有効範囲を持つ既存の記述子を識別するものでなければなりません。

記述子ヘッダーの COUNT フィールドは、結果セットの列の数を反映するように設定しなければなりません。結果セットのそれぞれの列ごとに TYPE および DATETIME_INTERVAL_CODE (適用可能な場合) を設定しなければなりません。

107. 戻された行の数が要求した行の数に等しい場合は、データ終わりの警告は発生しないことがあり、DB2_LAST_ROW (または SQLCA の SQLERRD(5)) は +100 に設定されないことがあります。

USING DESCRIPTOR *descriptor-name*

SQLDA を識別しますが、行記憶域 の中の行の形式を記述するゼロまたはそれ以上の変数の有効な記述が入っているものでなければなりません。

ユーザーは、FETCH ステートメントを処理する前に、SQLDA の以下のフィールドをセットしておく必要があります。

- SQLN (SQLDA に用意する SQLVAR のオカレンスの数を示します。)
- SQLDABC (SQLDA 用に割り振る記憶域のバイト数を示します。)
- SQLD (ステートメントを処理するときに SQLDA で使用する変数の個数を指示します。)
- SQLVAR の各オカレンス (変数の属性を指示します。)

SQLDA の他のフィールド (SQLNAME など) の値は、FETCH ステートメントを実行した後に定義することはできず、また、使用するべきではありません。

SQLDA の記憶域は、SQLVAR のオカレンスをすべて収容するのに十分な大きさで割り振らなければなりません。したがって、SQLDABC の値は、16 + SQLN*(80) よりも大きいか、または等しくなければなりません。ここで、80 は SQLVAR の 1 つのオカレンスの長さです。LOB または特殊タイプが指定された場合には、各パラメーター・マーカーごとに 2 つの SQLVAR 項目が必要であり、SQLN はパラメーター・マーカー数の 2 倍にセットしなければなりません。

SQLD には、ゼロ以上で SQLN 以下の値をセットしなければなりません。詳しくは、 1877 ページの『付録 D. SQLDA (SQL 記述子域)』を参照してください。

FETCH が完了すると、最初の SQLVAR 項目の SQLDATA ポインターは、最初の行の割り振り済み記憶域内の最初の列に対して戻された値をアドレス指定し、 2 番目の SQLVAR 項目の SQLDATA ポインターは、最初の行の割り振り済み記憶域内の 2 番目の列に対して戻された値を、というように順にアドレス指定します。NULL 可能な最初の SQLVAR 項目の SQLIND ポインターは最初の標識値をアドレス指定し、 2 番目のNULL 可能な SQLVAR 項目の SQLIND ポインターは 2 番目の標識値を、というように順にアドレス指定します。 SQLDA は、16 バイト境界上に割り振らなければなりません。

INTO *row-storage-area*

変数を使用して指定されたホスト *ID -1* は、行を戻す記憶域の割り振りを指定します。行は、SQLDA または SQL 記述子によって記述された形式でその記憶域に戻されます。ホスト *ID -1* は、要求された行をすべて保持できるだけの十分な大きさが必要です。

ホスト *ID -2* は、オプショナルの標識区域を識別します。戻されるデータ・タイプのいずれかがNULL 可能な場合にこれを指定する必要があります。この変数は、標識を戻す記憶域の割り振りを指定します。この標識は、短整数として戻されます。ホスト *ID -2* は、戻される各行についてNULL 可能な各値ごとに標識を入れられるだけの十分な大きさが必要です。

GET DIAGNOSTICS ステートメントは、行記憶域 に戻される各行の長さを示す DB2_ROW_LENGTH を戻すために使用できます。

INTO 文節によって識別されているか、または SQLDA で記述されている *n* 番目の変数は、カーソルの結果表の *n* 番目 の列に対応します。各変数のデータ・タイプは、それぞれに対応する列と互換性がなければなりません。

変数への割り当てはそれぞれ、 118 ページの『検索割り当て』で説明されている検索割り当て規則に従って行われます。[108]変数の数が行の中の値の数より少ない場合、SQLSTATE は「01503」に設定 (または SQLCA の SQLWARN3 フィールドに「W」が設定) されます。結果の列の数よりも変数の数が多い場合には、警告は出されない点に注意してください。値が NULL の場合は、標識変数が用意されている必要があります。割り当てでエラーが起こった場合、その値は変数に割り当てられず、それ以後の変数への値の割り当ては行われません。変数にすでに割り当てられている値はすべて、割り当てられたままとなります。

外側の SELECT ステートメントの SELECT リストの算術式の結果としてエラーが起こった場合 (ゼロによる除算、オーバーフロー、その他など)、または文字変換エラーが起こった場合には、結果は NULL 値になります。他の NULL 値の場合と同様に、標識変数を用意しなければなりません。該当の変数の値は、未定義になります。しかし、この場合、標識変数が -2 に設定されます。ステートメントの処理は、エラーが発生しなかった場合と同様に継続されます。 (ただし、警告が戻されます。) 標識変数を用意していない場合は、エラーが戻されます。エラーが生じた時点で、既にいくつかの値が変数に割り当てられていることがあり、それらの値は割り当てられたままになります。

結果列に LOB が含まれている場合、または現行接続がリモート・サーバーへの接続の場合、複数行取り出し は許されません。

注

カーソル位置**:** オープン状態のカーソルの位置として、次の 3 つの位置が考えられます。

- 行の前
- 行
- 最終行の後

カーソルがある行に位置付けられている場合、その行をカーソルの現在行と呼びます。 UPDATE または DELETE ステートメントで参照するカーソルは、行に位置付けられていなければなりません。カーソルは、FETCH ステートメントの結果としてのみ、行に位置付けることができます。

エラーの発生によって、カーソルの状態が予期できないものになることがあります。

変数の割り当て**:** INTO 文節によって識別されているか、または SQLDA で記述されている *n* 番目の変数は、カーソルの結果表の *n* 番目の列に対応します。各変数のデータ・タイプは、それぞれに対応する列と互換性がなければなりません。

108. SQL 変数または SQL パラメーターへの割り当て、および標準オプションが指定された場合は、記憶域割り当て規則が適用されます。標準オプションについては、 xi ページの『標準への準拠』を参照してください。

変数への割り当てはそれぞれ、 113 ページの『割り当ておよび比較』で説明されている検索割り当て規則に従って行われます。変数の数が行の中の値の数より少ない場合、SQLCA の SQLWARN3 フィールドに 'W' が設定されます。結果の列の数よりも変数の数が多い場合には、警告は出されない点に注意してください。値が NULL の場合は、標識変数が用意されている必要があります。割り当てエラーが発生すると、変数の値は予測不能になります。

ストリング変数を指定した場合に、その変数が結果を収容できるほどの大きさでなければ、警告 (SQLSTATE 01004) が返されます (さらに、SQLCA で SQLWARN1 に「W」が割り当てられます)。標識変数が用意されている場合、結果の実際の長さは、その変数 に関連する標識変数に戻されます。

変数として C の NUL で終了する変数を指定し、その変数が、結果および NUL 終了文字を入れられるだけの十分な大きさを持っていない場合は、以下のようになります。

- CRTSQLCI コマンドまたは CRTSQLCPPI コマンドに *CNULRQD オプションを指定した場合 (または SET OPTION ステートメントに CNULRQD(*YES) を指定した場合)、以下のようになります。
  - 結果が切り捨てられます。
  - 最後の文字は NUL 終了文字になります。
  - 警告 (SQLSTATE 01004) が戻されます (そして、SQLCA の SQLWARN1 に「W」が割り当てられます)。
- CRTSQLCI コマンドまたは CRTSQLCPPI コマンドに *NOCNULRQD オプションを指定した場合 (または SET OPTION ステートメントに CNULRQD(*NO) を指定した場合)、以下のようになります。
  - NUL 終了文字は戻されません。
  - 警告 (SQLSTATE 01004) が戻されます (そして、SQLCA の SQLWARN1 に「N」が割り当てられます)。

代替構文**:** 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- USING DESCRIPTOR は、単一取り出し文節で INTO DESCRIPTOR の同義語として使用することができます。

例

例 *1:* この C の例では、FETCH ステートメントが SELECT ステートメントの結果を取り出してプログラム変数 dnum、dname、および mnum に入れます。取り出す行がなくなったとき、不検出条件が戻されます。

```
EXEC SQL DECLARE C1 CURSOR FOR
  SELECT DEPTNO, DEPTNAME, MGRNO FROM TDEPT
  WHERE ADMRDEPT = 'A00';
EXEC SQL OPEN C1;
while (SQLCODE==0) {
  EXEC SQL FETCH C1 INTO :dnum, :dname, :mnum;
}
EXEC SQL CLOSE C1;
```

例 *2:* この FETCH ステートメントは、SQLDA を使用します。

```
FETCH CURS USING DESCRIPTOR :sqlda3
```

例 *3:* この ILE RPG 例では、行記憶域を使用してデータを取り出します。

```
DCL-S ONE_ROW_PTR POINTER;
DCL-DS ONE_ROW BASED(ONE_ROW_PTR);
  DEPTNO   CHAR(3);
  DEPTNAME VARCHAR(36);
  MGRNO    CHAR(6);
END-DS;
DCL-S ONE_ROW_IND_PTR POINTER;
DCL-DS ONE_ROW_IND BASED(ONE_ROW_IND_PTR);
  DEPTNOIND   INT(5);
  DEPTNAMEIND INT(5);
  MGRNOIND    INT(5);
END-DS;
DCL-S ROWAREA CHAR(450);  // 10 records * %SIZE(ONE_ROW)
DCL-S INDAREA CHAR(60);   // 10 records * %SIZE(ONE_ROW_IND)
DCL-S ROWS_RETURNED INT(5);
DCL-S I INT(5);

EXEC SQL DECLARE C1 CURSOR FOR
  SELECT DEPTNO, DEPTNAME, MGRNO FROM CORPDATA.DEPARTMENT;

// Set up the descriptor
EXEC SQL ALLOCATE DESCRIPTOR 'FETCH_ROWS' WITH MAX 10;
EXEC SQL SET DESCRIPTOR 'FETCH_ROWS'
     COUNT = 3;  // Descriptor contains 3 items
EXEC SQL SET DESCRIPTOR 'FETCH_ROWS'
     VALUE 1  TYPE = 1, LENGTH = 3; // First is CHAR(3)
EXEC SQL SET DESCRIPTOR 'FETCH_ROWS'
     VALUE 2  TYPE = 12, LENGTH = 36; // Second is VARCHAR(36)
EXEC SQL SET DESCRIPTOR 'FETCH_ROWS'
     VALUE 3  TYPE = 1, LENGTH = 6; // Third is CHAR(6)

// Fetch the data
EXEC SQL OPEN C1;
EXEC SQL FETCH C1 FOR 10 ROWS
         USING SQL DESCRIPTOR 'FETCH_ROWS'
         INTO :ROWAREA:INDAREA;
EXEC SQL GET DIAGNOSTICS :ROWS_RETURNED = ROW_COUNT;
EXEC SQL CLOSE C1;

ONE_ROW_PTR = %ADDR(ROWAREA); // Get first row
ONE_ROW_IND_PTR = %ADDR(INDAREA); // Indicators for first row
FOR I = 1 TO ROWS_RETURNED;
  IF MGRNOIND >= 0; // Not a null value
    // Do something with MGRNO
  ENDIF;
  // Handle other values for first row
  ONE_ROW_PTR = ONE_ROW_PTR + %SIZE(ONE_ROW); // Advance to next row
  ONE_ROW_IND_PTR = ONE_ROW_IND_PTR + %SIZE(ONE_ROW_IND);
ENDFOR;
```

FREE LOCATOR

FREE LOCATOR ステートメントは、ロケーター変数とその値の間の関連を除去します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。これを対話式に発行することはできません。これは、動的に準備できる実行可能ステートメントです。ただし、準備済みステートメントを実行するには、USING 文節を指定した EXECUTE ステートメントを使用しなければなりません。 FREE LOCATOR は、EXECUTE IMMEDIATE ステートメントと併用することはできません。Java および REXX では指定できません。

権限

権限は不要です。

構文

説明

変数,...

1 つ以上のロケーター変数を指定します。これらの変数は、ロケーター変数の宣言の規則に従って宣言する必要があります。ロケーター変数のタイプは、バイナリー・ラージ・オブジェクト・ロケーター、文字ラージ・オブジェクト・ロケーター、2 バイト文字ラージ・オブジェクト・ロケーター、XML ロケーターのいずれかでなければなりません。

この変数 には、現在ロケーターが割り当てられている必要があります。つまり、この作業単位中に (CALL、FETCH、SELECT INTO、割り当てステートメント、SET 変数、または VALUES INTO ステートメントによって) ロケーターが割り当てられていなければならず、それ以降そのロケーターが (FREE LOCATOR ステートメントによって) 解放されていてはならない、ということです。そうでない場合には、エラーが戻されます。

複数のロケーター変数が指定されていて、ロケーターの 1 つでエラーが発生した場合、どのロケーターも解放されることはありません。

例

従業員表に列 RESUME、HISTORY、および PICTURE が含まれていて、それらの列値を表すためにロケーターが確立されていると想定します。COBOL プログラムでは、CLOB ロケーター変数 LOCRES と LOCHIST、および BLOB ロケーター変数 LOCPIC を解放します。

```
EXEC SQL
FREE LOCATOR :LOCRES, :LOCHIST, :LOCPIC
END-EXEC.
```

GET DESCRIPTOR

GET DESCRIPTOR ステートメントは、SQL 記述子から情報を取得します。

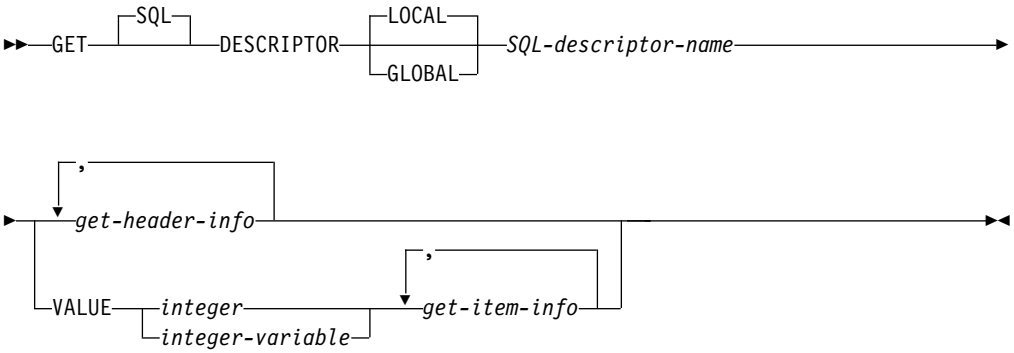
呼び出し

このステートメントは、アプリケーション・プログラム、SQL 関数、 SQL プロシージャー、またはトリガー内にのみ組み込むことができます。これを対話式に発行することはできません。これは実行可能ステートメントですが、動的に準備することはできません。REXX で指定してはなりません。

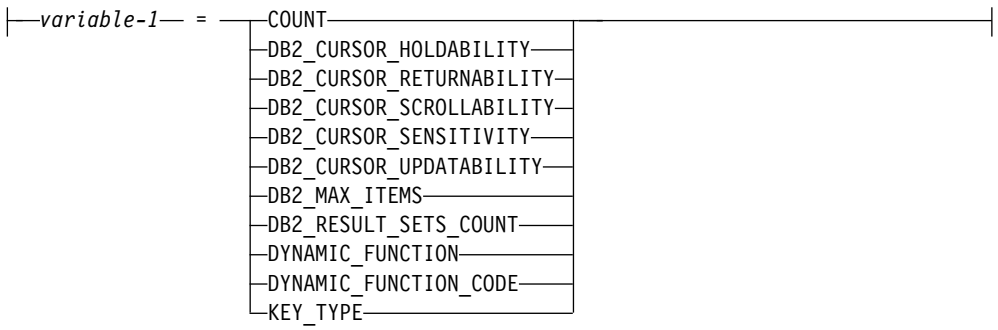
権限

権限は不要です。

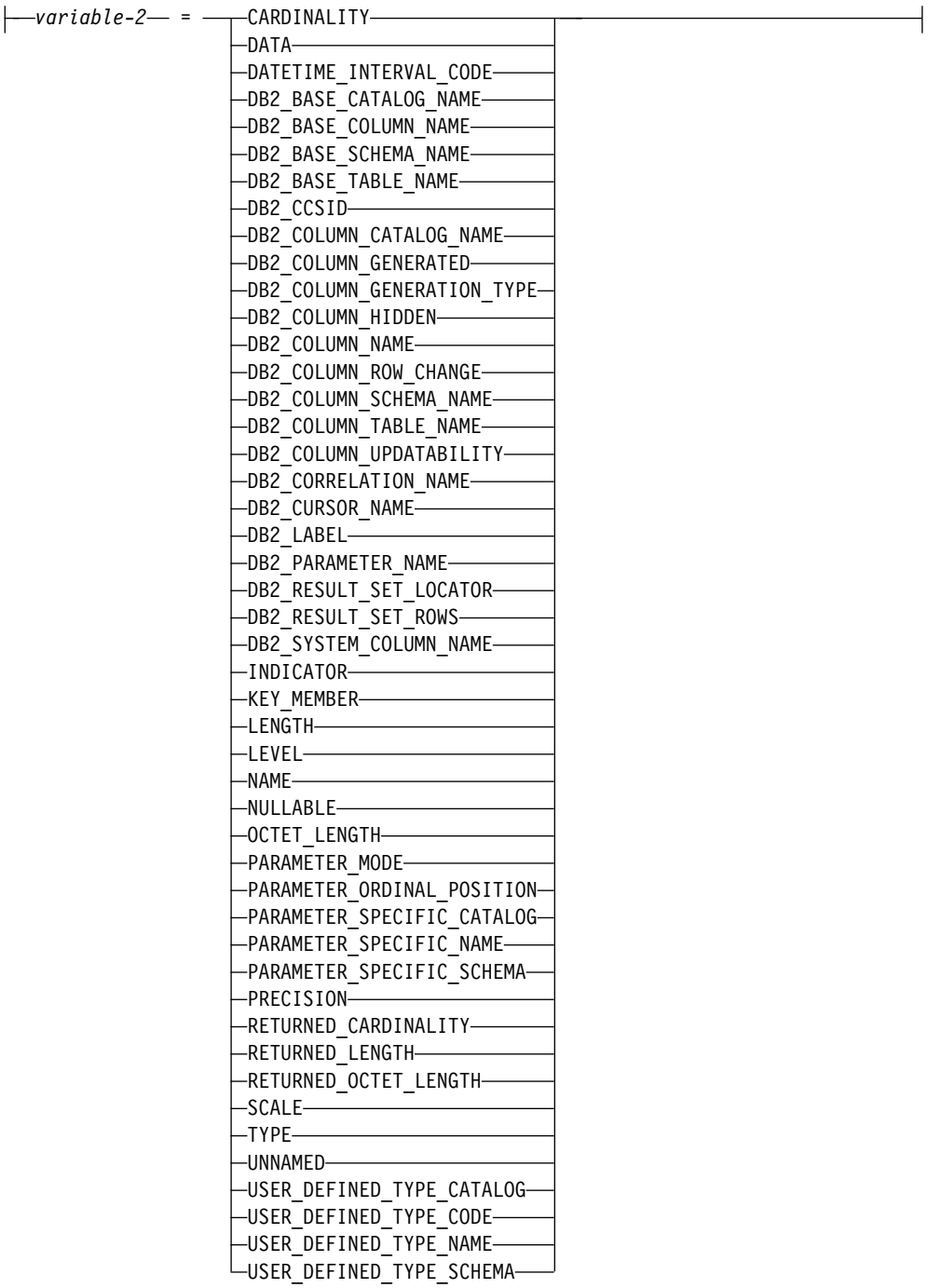
構文



get-header-info:



get-item-info:



説明

LOCAL
記述子の名前の有効範囲はプログラム呼び出しのローカルであることを指定します。情報は、このローカル有効範囲で既知の記述子から戻されます。

GLOBAL
記述子の名前の有効範囲は SQL セッション全体であることを指定します。情報は、同じデータベース接続を使用して実行するどのプログラムにも既知の記述子から戻されます。

SQL-descriptor-name
SQL 記述子の名前を指定します。名前は、指定の有効範囲を持つ既存の記述子を識別するものでなければなりません。

get-header-info
準備済み SQL ステートメントおよび SQL 記述子に関する情報を戻します。

VALUE
指定された情報が取り出される項目数を示します。この値が (ヘッダー情報からの) COUNT の値より大きい場合、結果は戻されません。記述子に割り振られる項目の最大数よりも項目数が大きいか、または項目数が 1 より小さい場合、エラーが戻されます。

integer
1 から SQL 記述子の項目数の範囲の整数定数。

integer-variable
変数を宣言する規則に従ってプログラム内で宣言された、変数を指定します。変数をグローバル変数にすることはできません。変数のデータ・タイプは、SMALLINT、INTEGER、BIGINT、または DECIMAL、あるいは位取りがゼロの NUMERIC でなければなりません。整変数 の値は、1 から SQL 記述子の項目の最大数の範囲でなければなりません。

get-item-info
SQL 記述子の特定の項目に関する情報を戻します。

ヘッダー情報の取得

variable-1
変数を宣言する規則に従ってプログラム内で宣言された、変数を指定します。ただし、これはファイル参照変数またはグローバル変数であってはなりません。変数のデータ・タイプは、 1485 ページの表 89で指定されている記述子情報項目と互換性がなければなりません。変数は、(記憶域割り当て規則を使用して) 対応する記述子項目に割り当てられます。割り当て規則の詳細については、 113 ページの『割り当ておよび比較』を参照してください。

COUNT
記述子の項目数。

DB2_CURSOR_HOLDABILITY
カーソルの保留状況。指定可能な値は次のとおりです。
0 記述子がカーソルを記述していないか、カーソルが WITH HOLD で宣言されていません。
1 WITH HOLD と宣言されたカーソル。

DB2_CURSOR_RETURNABILITY
カーソルの結果セットの戻り状況。指定可能な値は次のとおりです。
0 記述子がカーソルを記述していないか、カーソルの結果セットを返せません。
1 カーソルの結果セットをプロシージャーの呼び出し元に返せます。
2 カーソルの結果セットをクライアントに返せます。

DB2_CURSOR_SCROLLABILITY
カーソルのスクロール状況。指定可能な値は次のとおりです。

0 記述子がカーソルを記述していないか、カーソルが SCROLL で宣言されていません。
1 SCROLL と宣言されたカーソル。

DB2_CURSOR_SENSITIVITY
カーソルのセンシティビティー。指定可能な値は次のとおりです。
0 記述子がカーソルを記述していません。
1 カーソルは SENSITIVE DYNAMIC です。
3 カーソルは INSENSITIVE です。
4 カーソルは ASENSITIVE です。

DB2_CURSOR_UPDATABILITY
WHERE CURRENT OF *cursor-name* を組み込んだ UPDATE ステートメントでカーソルを使用できるかどうかを指定します。指定可能な値は次のとおりです。
0 記述子がカーソルを記述していないか、WHERE CURRENT OF *cursor-name* を組み込んだ UPDATE ステートメントでカーソルを使用できません。
1 WHERE CURRENT OF *cursor-name* を組み込んだ UPDATE ステートメントでカーソルを使用できます。

DB2_MAX_ITEMS
ALLOCATE DESCRIPTOR ステートメントでの割り振り済み項目記述子の最大数として指定されている値を表します。 WITH MAX 文節が指定されなかった場合、値はデフォルトの ALLOCATE DESCRIPTOR ステートメントの最大項目数になります。

DB2_RESULT_SETS_COUNT
プロシージャーから戻された結果セットの数。この記述子がプロシージャーを記述していない場合は、値が 0 になります。

DYNAMIC_FUNCTION
文字ストリングとしての準備済み SQL ステートメントのタイプ。ステートメント・タイプの詳細については、 1511 ページの表 92 を参照してください。

DYNAMIC_FUNCTION_CODE
準備済み SQL ステートメントのタイプを表すステートメント・コード。ステートメント・コードの詳細については、 1511 ページの表 92 を参照してください。

KEY_TYPE
選択リストに含まれるキーのタイプ。考えられる値は、次のとおりです。
0 記述子が照会の列を記述していないか、または照会でキー列が参照されていないか、あるいは固有キーがありません。
1 選択リストに、照会によって参照される基本表の基本キーのすべての列が含まれています。
2 照会によって参照される表には基本キーがありませんが、選択リストには優先候補キーとして定義されている列のセットが含まれます。選択リストにそのような優先候補キーが複数含まれている場合、左端の優先候補キーが使用されます。

項目情報の取得

variable-2
変数を宣言する規則に従ってプログラム内で宣言された、変数を指定します。ただし、これはファイル参照変数またはグローバル変数であってはなりません。変数のデータ・タイプは、 1485 ページの表 89で指定されている記述子情報項目と互換性がなければなりません。変数は、(記憶域割り当て規則を使用して) 対応する記述子項目に割り当てられます。割り当て規則の詳細については、 113 ページの『割り当ておよび比較』を参照してください。

一般的に、DATA 項目を取得するときは変数のデータ・タイプ、長さ、精度、位取り、および CCSID が 1485 ページの表 89 で指定されているものと同じでなければなりません。可変長タイプの場合、可変長は記述子の LENGTH よりも小さくてはなりません。 C NUL 終了タイプの場合、可変長は記述子の LENGTH よりも少なくとも 1 大きくなければなりません。

CARDINALITY
配列データ・タイプのカーディナリティー。この記述子が DESCRIBE の結果である場合は、配列データ・タイプの最大カーディナリティーになります。カーディナリティーは他のすべてのデータ・タイプで 0 になります。

DATA
項目記述子によって記述されるデータの値。 INDICATOR の値が負の場合、DATA の値は未定義となり、INDICATOR 項目情報の取得 も同じステートメントに指定しなければなりません。

DATETIME_INTERVAL_CODE
特定の日時データ・タイプを定義するコード。
0 記述子項目には TYPE 値 9 がありません。
1 DATE
2 TIME
3 TIMESTAMP

DB2_BASE_CATALOG_NAME
項目記述子によって表される列の基本表のサーバー名。

DB2_BASE_COLUMN_NAME
記述される照会で参照される (おそらくビューを介して間接的に) 基本表で定義されている列の名前。列名を定義できないかまたは適用不可の場合、この項目には空ストリングが入ります。名前は、大/小文字の区別ありの区切り文字なしで戻されます。

DB2_BASE_SCHEMA_NAME
項目記述子によって表される列の基本表のスキーマ名。スキーマ名を定義できないかまたは適用不可の場合、この項目には空ストリングが入ります。名前は、大/小文字の区別ありの区切り文字なしで戻されます。

DB2_BASE_TABLE_NAME
項目記述子によって表される列の基礎となる基本表の表名。表名を定義できないかまたは適用不可の場合、この項目には空ストリングが入ります。名前は、大/小文字の区別ありの区切り文字なしで戻されます。

DB2_CCSID
文字、グラフィック、または XML データの CCSID。文字タイプ、グラフィッ

ク・ストリング・タイプ、または XML タイプに基づかないタイプではすべて、値はゼロになります。FOR BIT DATA 属性を持つバイナリー・タイプまたは文字タイプでは、値は 65535 になります。

DB2_COLUMN_CATALOG_NAME
項目記述子によって表されている列の参照される表またはビューのサーバー名。列カタログ名を定義できないかまたは適用不可の場合、この項目には空ストリングが入ります。

DB2_COLUMN_GENERATED
列が生成されるかどうかを示します。考えられる値は、次のとおりです。

0 生成されない
1 GENERATED ALWAYS
2 GENERATED BY DEFAULT

DB2_COLUMN_GENERATION_TYPE
列がどのように生成されるかを示します。考えられる値は、次のとおりです。

0 生成されない
1 IDENTITY 列
2 ROWID 列
4 行変更タイム・スタンプ列
5 ROW BEGIN 列
6 ROW END 列
7 TRANSACTION START ID 列
8 生成式列

DB2_COLUMN_HIDDEN
項目記述子によって表される列が隠されているかどうかを示します。考えられる値は、次のとおりです。

0 隠されていない
1 暗黙に隠されている
3 オプティミスティック・ロックに対して暗黙に隠されている

DB2_COLUMN_NAME
記述される照会で参照される表またはビューで定義されている列の名前。列名を定義できないかまたは適用不可の場合、この項目には空ストリングが入ります。名前は、大/小文字の区別ありの区切り文字なしで戻されます。

DB2_COLUMN_ROW_CHANGE
項目記述子によって表される列が、WITH ROW CHANGE COLUMNS 準備属性を使用した結果追加されたかどうかを示します。考えられる値は、次のとおりです。

-1 ROW CHANGE TOKEN (特殊)
-2 ROW CHANGE TOKEN (特殊でない)
-3 RID (リモート・リレーショナル・データベースからのみ有効)
-4 RID_BIT (リモート・リレーショナル・データベースからのみ有効)

DB2_COLUMN_SCHEMA_NAME
項目記述子によって表されている列の参照される表またはビューのスキーマ名。列スキーマ名を定義できないかまたは適用不可の場合、この項目には空ストリングが入ります。名前は、大/小文字の区別ありの区切り文字なしで戻されます。

DB2_COLUMN_TABLE_NAME
項目記述子によって表されている列の参照される表またはビューの表名またはビュー名。列表名を定義できないかまたは適用不可の場合、この項目には空ストリングが入ります。名前は、大/小文字の区別ありの区切り文字なしで戻されます。

DB2_COLUMN_UPDATABILITY
項目記述子によって表される列が更新可能かどうかを示します。考えられる値は、次のとおりです。
0 更新不可
1 更新可能

DB2_CORRELATION_NAME
常に空ストリングが戻されます。

DB2_CURSOR_NAME
この結果セットに対応するプロシージャー内のカーソルの名前。値が設定されるのは、記述子がプロシージャーを記述している場合に限られます。

DB2_LABEL
列に対して定義されるラベル。列のラベルがない場合、この項目には空ストリングが入ります。

DB2_PARAMETER_NAME
ストアード・プロシージャーのパラメーターの名前。 CALL ステートメントに対してのみ戻されます。名前は、大/小文字の区別ありの区切り文字なしで戻されます。

DB2_RESULT_SET_LOCATOR
この結果セットの結果セット・ロケーター。値が設定されるのは、記述子がプロシージャーを記述している場合に限られます。

DB2_RESULT_SET_ROWS
結果セット内の行の予測数。数が不明な場合は、値 -1 に設定されます。値が設定されるのは、記述子がプロシージャーを記述している場合に限られます。

DB2_SYSTEM_COLUMN_NAME
列のシステム名。システム名を定義できないかまたは適用不可の場合、この項目にはブランクが入ります。

INDICATOR
標識の値。この記述子項目に戻される値が DATA フィールドに指定されると、負でない値が使用されます。拡張標識変数が使用できない際に、この記述子項目に戻される値が NULL 値の場合、負の値が使用されます。拡張標識が使用可能な場合は、次のようになります。

- -1、-2、-3、-4、または -6 は、この記述子に戻される値が NULL 値であることを示します。
- -5 は、この記述子項目に戻される値が DEFAULT であることを示します。
- -7 は、この記述子項目に戻される値が UNASSIGNED であることを示します。

KEY_MEMBER
この列がキーの一部かどうかを示す標識。
0 この列はキーの一部ではありません。

1 この列は固有キーの一部です。
2 この列自体が固有キーです。

LENGTH
データの最大長を戻します。データ・タイプが文字またはグラフィック・ストリング、XML タイプまたは日時タイプの場合、長さは文字数を表します (バイト数ではない)。データ・タイプがバイナリー・ストリングまたは他のタイプの場合、長さはバイト数を表します。データ・タイプ・コードおよび長さの説明については、 1487 ページの表 90を参照してください。

LEVEL
項目記述子のレベル。値は 0 です。

NAME
項目記述子によって記述される選択リスト列に関連した名前。名前は、大/小文字の区別ありの区切り文字なしで戻されます。

NULLABLE
列またはパラメーター・マーカーがNULL 可能かどうかを示します。
0 選択リスト列またはパラメーター・マーカーに NULL 値を指定できません。
1 選択リスト列またはパラメーター・マーカーに NULL 値を指定できます。

OCTET_LENGTH
すべてのタイプのデータの最大長をバイトで戻します。データ・タイプ・コードおよび長さの説明については、 1487 ページの表 90を参照してください。

PARAMETER_MODE
CALL ステートメントでのパラメーター・マーカーのモード。
0 記述子は CALL ステートメントと関連付けられていません。
1 入力専用パラメーター。
2 入出力パラメーター。
4 出力専用パラメーター。

PARAMETER_ORDINAL_POSITION
CALL ステートメントでのパラメーター・マーカーの順序位置。記述子は CALL ステートメントと関連付けられていない場合、値は 0 になります。

PARAMETER_SPECIFIC_CATALOG
パラメーター・マーカーを含むプロシージャーのサーバー名。

PARAMETER_SPECIFIC_NAME
パラメーター・マーカーを含むプロシージャーの特定の名前。名前は、大/小文字の区別ありの区切り文字なしで戻されます。

PARAMETER_SPECIFIC_SCHEMA
パラメーター・マーカーを含むプロシージャーのスキーマ名。名前は、大/小文字の区別ありの区切り文字なしで戻されます。

PRECISION
データの精度を戻します。
SMALLINT
5

INTEGER
10
BIGINT
19
NUMERIC および **DECIMAL**
定義済み精度
REAL 24
DOUBLE
53
DECFLOAT(7)
7
DECFLOAT(16)
16
DECFLOAT(34)
34
TIME 0
TIMESTAMP
定義済み精度 (0-12)
その他のデータ・タイプ
0

RETURNED_CARDINALITY
FETCH または CALL から返される配列データ・タイプの現在のカーディナリティー。項目のデータ・タイプが配列でない場合は、値が 0 になります。

RETURNED_LENGTH
文字ストリングおよびグラフィック・ストリング、および XML のデータ・タイプの場合、戻される長さ (文字数)。バイナリー・ストリング・データ・タイプの場合、戻される長さ (バイト数)。

RETURNED_OCTET_LENGTH
すべてのストリング・データ・タイプに関して、戻される長さ (バイト数)。

SCALE
データ・タイプが DECIMAL または NUMERIC の場合、定義済みの位取りを戻します。位取りは他のすべてのデータ・タイプで 0 になります。

TYPE
項目のデータ・タイプを表すデータ・タイプ・コードを戻します。データ・タイプ・コードおよび長さの説明については、 1487 ページの表 90を参照してください。

UNNAMED
値 1 は、NAME 値がデータベース・マネージャーによって生成されることを示します。それ以外の場合、値はゼロで、NAME は選択リストの列の派生名になります。

USER_DEFINED_TYPE_CATALOG
ユーザー定義タイプのサーバー名。タイプがユーザー定義データ・タイプではない場合、この項目には空ストリングが入ります。

USER_DEFINED_TYPE_CODE
記述子項目のタイプがユーザー定義タイプかどうかを示します。

0 記述子項目はユーザー定義タイプではありません。
1 記述子項目はユーザー定義タイプです。

USER_DEFINED_TYPE_NAME
ユーザー定義データ・タイプの名前。タイプがユーザー定義データ・タイプではない場合、この項目には空ストリングが入ります。名前は、大/小文字の区別ありの区切り文字なしで戻されます。

USER_DEFINED_TYPE_SCHEMA
ユーザー定義データ・タイプのスキーマ名。タイプがユーザー定義データ・タイプではない場合、この項目には空ストリングが入ります。名前は、大/小文字の区別ありの区切り文字なしで戻されます。

注

項目のデータ・タイプ: 以下の表は、記述子項目ごとの SQL データ・タイプを示しています。記述子項目が変数に割り当てられるとき、変数は記述子項目のデータ・タイプと互換性がなければなりません。

表 *89. GET DESCRIPTOR* 項目のデータ・タイプ

項目名	データ・タイプ
ヘッダー情報	
COUNT	INTEGER
DB2_CURSOR_HOLDABILITY	INTEGER
DB2_CURSOR_RETURNABILITY	INTEGER
DB2_CURSOR_SCROLLABILITY	INTEGER
DB2_CURSOR_SENSITIVITY	INTEGER
DB2_CURSOR_UPDATABILITY	INTEGER
DB2_MAX_ITEMS	INTEGER
DB2_RESULT_SETS_COUNT	INTEGER
DYNAMIC_FUNCTION	VARCHAR(128)
DYNAMIC_FUNCTION_CODE	INTEGER
KEY_TYPE	INTEGER
項目情報	
CARDINALITY	BIGINT
DATA	TYPE によって指定されるデータ・タイプと一致
DATETIME_INTERVAL_CODE	INTEGER
DB2_BASE_CATALOG_NAME	VARCHAR(128)
DB2_BASE_COLUMN_NAME	VARCHAR(128)
DB2_BASE_SCHEMA_NAME	VARCHAR(128)
DB2_BASE_TABLE_NAME	VARCHAR(128)
DB2_CCSID	INTEGER
DB2_COLUMN_CATALOG_NAME	VARCHAR(128)
DB2_COLUMN_GENERATED	INTEGER
DB2_COLUMN_GENERATION_TYPE	INTEGER

表 89. GET DESCRIPTOR 項目のデータ・タイプ (続き)

項目名	データ・タイプ
DB2_COLUMN_HIDDEN	INTEGER
DB2_COLUMN_NAME	VARCHAR(128)
DB2_COLUMN_ROW_CHANGE	INTEGER
DB2_COLUMN_SCHEMA_NAME	VARCHAR(128)
DB2_COLUMN_TABLE_NAME	VARCHAR(128)
DB2_COLUMN_UPDATABILITY	INTEGER
DB2_CORRELATION_NAME	VARCHAR(128)
DB2_CURSOR_NAME	VARCHAR(128)
DB2_LABEL	VARCHAR(60)
DB2_PARAMETER_NAME	VARCHAR(128)
DB2_RESULT_SET_LOCATOR	結果セット・ロケーター
DB2_RESULT_SET_ROWS	BIGINT
DB2_SYSTEM_COLUMN_NAME	CHAR(10)
INDICATOR	INTEGER
KEY_MEMBER	INTEGER
LENGTH	INTEGER
LEVEL	INTEGER
NAME	VARCHAR(128)
NULLABLE	INTEGER
OCTET_LENGTH	INTEGER
PARAMETER_MODE	INTEGER
PARAMETER_ORDINAL_POSITION	INTEGER
PARAMETER_SPECIFIC_CATALOG	VARCHAR(128)
PARAMETER_SPECIFIC_NAME	VARCHAR(128)
PARAMETER_SPECIFIC_SCHEMA	VARCHAR(128)
PRECISION	INTEGER
RETURNED_CARDINALITY	INTEGER
RETURNED_LENGTH	INTEGER
RETURNED_OCTET_LENGTH	INTEGER
SCALE	INTEGER
TYPE	INTEGER
UNNAMED	INTEGER
USER_DEFINED_TYPE_CATALOG	VARCHAR(128)
USER_DEFINED_TYPE_NAME	VARCHAR(128)
USER_DEFINED_TYPE_SCHEMA	VARCHAR(128)
USER_DEFINED_TYPE_CODE	VARCHAR(128)

SQL データ・タイプ・コードおよび長さ**:** 以下の表は、TYPE、LENGTH、OCTET_LENGTH、および DATETIME_INTERVAL_CODE 記述子項目で考えられる値を表しています。

以下の表の値は、ISO および ANSI SQL Standard によって割り当てられていて、この規格の発展に応じて変更される可能性があります。これらの値を参照するときには、ライブラリー QSYSINC 内のインクルード・ソース・ファイルにあるインクルード *sqlscds* を使用してください。

表 *90. SQL* データ・タイプ・コードおよび長さ

データ・タイプ	データ・タイプ・コード	長さ	オクテット長
SMALLINT	5	2	2
INTEGER	4	4	4
BIGINT	25	8	8
DECIMAL	3	(精度/2)+1	(精度/2)+1
NUMERIC(n)	2	n	n
REAL	7	4	4
FLOAT	6	8	8
DOUBLE PRECISION	8	8	8
DECFLOAT(7)	-360	4	4
DECFLOAT(16)	-360	8	8
DECFLOAT(34)	-360	16	16
CHARACTER(n)	1	n	n
VARCHAR(n)	12	<=n	n
CLOB(n)	40	<=n	n
GRAPHIC (n)	-95	n	2*n
VARGRAPHIC (n)	–96	<=n	2*n
DBCLOB(n)	-350	<=n	2*n
BINARY(n)	–2	n	n
VARBINARY(n)	–3	<=n	n
BLOB(n)	30	n	n
DATE (DATETIME_INTERVAL_CODE = 1)	9	長さは日付形式に依存する	CCSID に基づく
TIME (DATETIME_INTERVAL_CODE = 2)	9	長さは時刻形式に依存する	CCSID に基づく
TIMESTAMP (DATETIME_INTERVAL_CODE = 3)	9	精度に応じて、19 から 32	精度に応じて、19 から 32 または 38 から 64 (CCSID によって異なる)
DATALINK(n)	70	<=n	n
ROWID	-904	40	40
XML	137	0	0
C NUL 終了 CHARACTER(n)	1	<=n	n
C NUL 終了 GRAPHIC(n)	–400	<=n	2*n
BLOB ファイル参照変数	-916	267	267
CLOB ファイル参照変数	-920	267	267

表 *90. SQL* データ・タイプ・コードおよび長さ *(*続き*)*

データ・タイプ	データ・タイプ・コード	長さ	オクテット長
DBCLOB ファイル参照変数	-924	267	267
結果セット・ロケーター	-972	8	8
アレイ	50	該当せず	該当せず

例

例 *1:* 記述子 'NEWDA' から記述子項目の数を取得します。

```
EXEC SQL GET DESCRIPTOR  'NEWDA'
  :numitems = COUNT;
```

例 *2:* 記述子 'NEWDA' の最初の項目記述子からデータ・タイプとオクテット長を取得します。

```
GET DESCRIPTOR 'NEWDA'
  VALUE 1 :dtype   = TYPE,
          :olength = OCTET_LENGTH;
```

GET DIAGNOSTICS

GET DIAGNOSTICS ステートメントは、直前に実行された SQL ステートメントに関する情報を取得します。

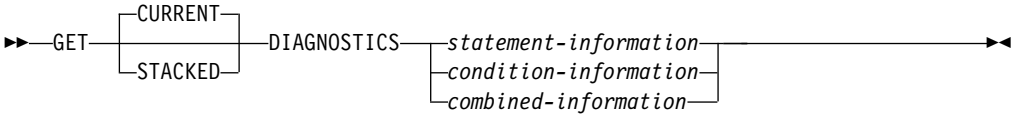
呼び出し

このステートメントは、アプリケーション・プログラム、SQL 関数、 SQL プロシージャー、またはトリガー内にのみ組み込むことができます。これを対話式に発行することはできません。これは実行可能ステートメントですが、動的に準備することはできません。REXX で指定してはなりません。

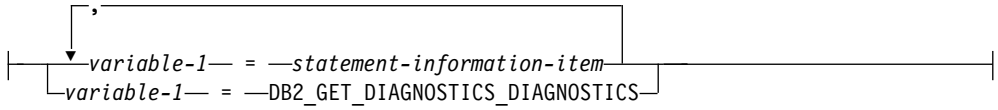
権限

権限は不要です。

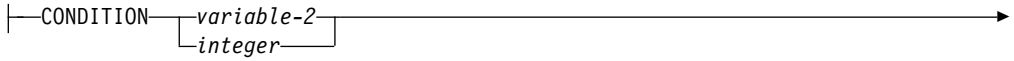
構文

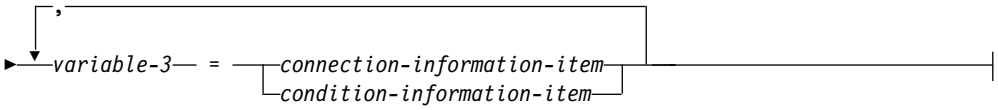


statement-information:

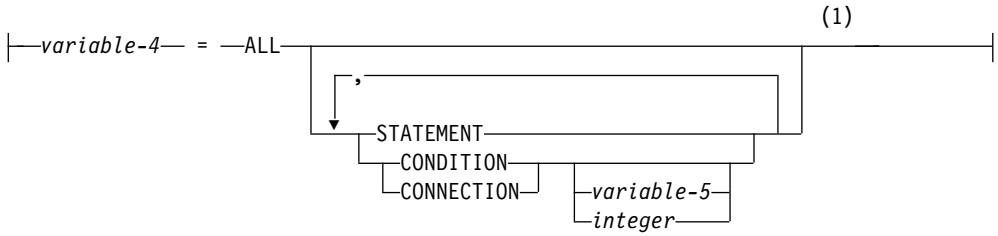


condition-information:





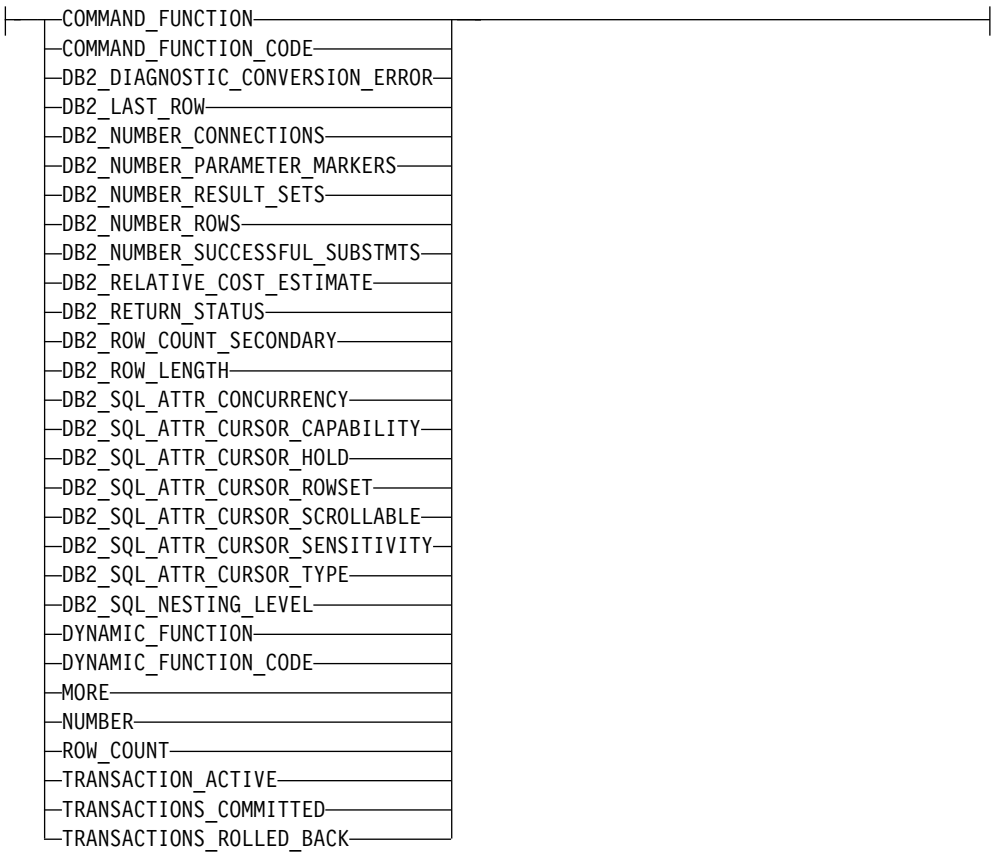
combined-information:



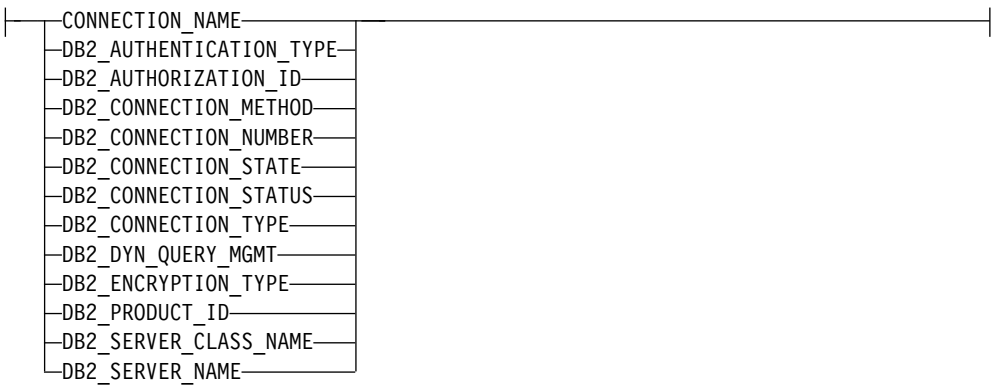
注:

1 STATEMENT は 1 回だけ指定できます。変数-*5* または整数 が指定されていない場合、 CONDITION および CONNECTION は 1 回だけ指定できます。

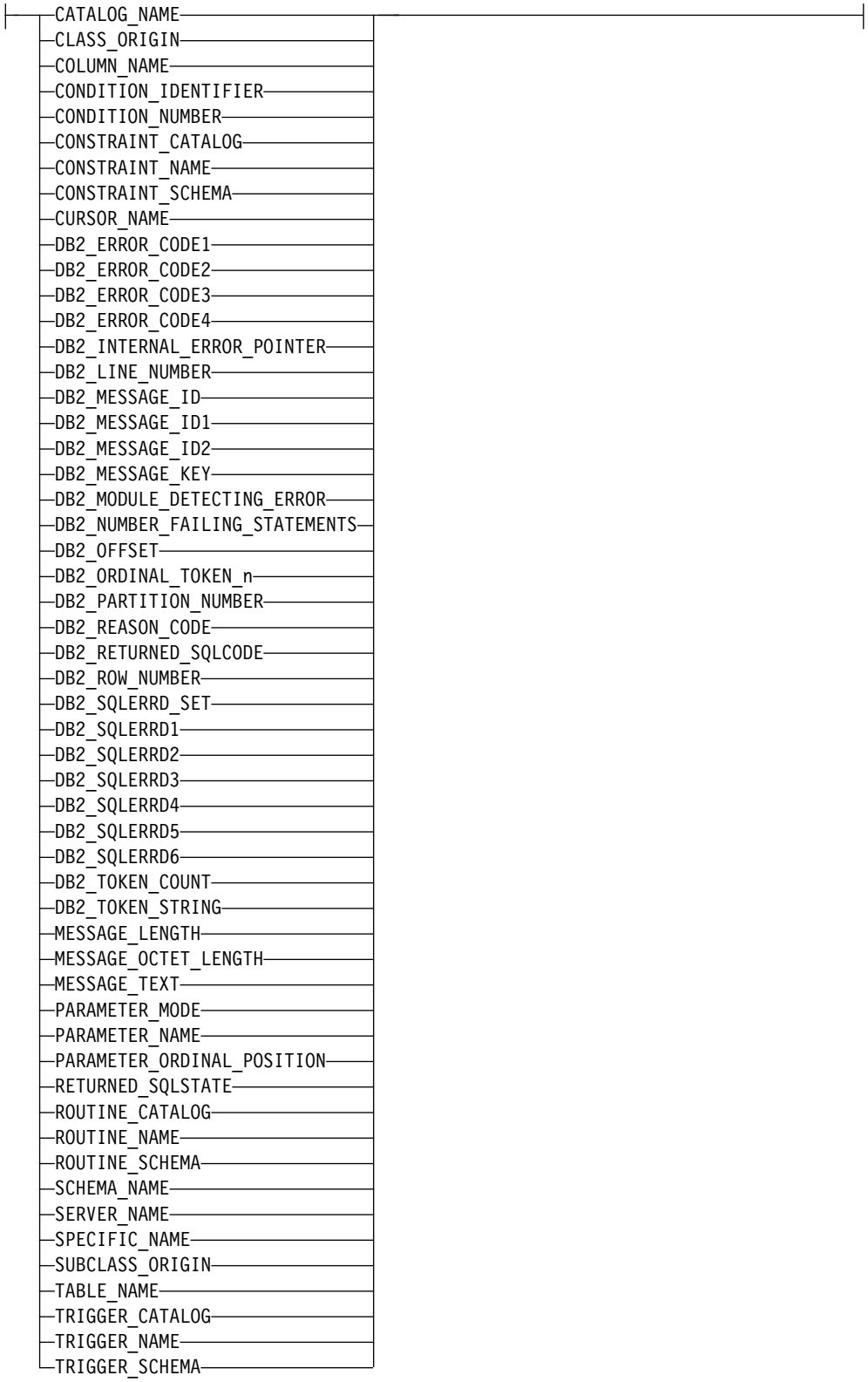
statement-information-item:



connection-information-item:



condition-information-item:



説明

CURRENT または **STACKED**
アクセスする診断領域を指定します。

CURRENT
最初の診断領域にアクセスするように指定します。これは、直前に実行された SQL ステートメントで、GET DIAGNOSTICS ステートメントでないものに対応します。これはデフォルトです。

STACKED
2 番目の診断領域にアクセスするように指定します。 2 番目の診断領域は、ハンドラー内だけで使用できます。これはハンドラーに入る前に実行された直前の SQL ステートメントで、 GET DIAGNOSTICS ではないものに対応します。 GET DIAGNOSTICS ステートメントがハンドラー内で最初のステートメントである場合、最初の診断領域と 2 番目の診断領域には同じ診断情報が含まれます。

statement-information
最後に実行された SQL ステートメントに関する情報を戻します。

variable-1
変数を宣言する規則に従ってプログラム内で宣言された、変数を指定します。変数をグローバル変数にすることはできません。変数のデータ・タイプは、 1508 ページの表 91 で指定の条件情報項目として指定されているデータ・タイプと互換性がなければなりません。変数には、 118 ページの『検索割り当て』で説明されている検索割り当て規則に従って、指定したステートメント情報項目の値が割り当てられます。その値が変数に割り当てられる際に切り捨てられる場合、警告 (SQLSTATE 01004) が戻されて診断領域の GET_DIAGNOSTICS_DIAGNOSTICS 項目にこの条件の詳細が追加されて更新されます。

指定の診断項目に診断情報が含まれない場合、変数はそのデータ・タイプに基づいてデフォルト値に設定されます。

- 厳密な数の値診断項目は 0、
- VARCHAR 診断項目は空ストリング、
- および CHAR 診断項目はブランクです。

condition-information
最後の SQL ステートメントの実行時に生じた 1 つ以上の条件に関する情報を戻します。

CONDITION *variable-2* または *integer*
情報が要求された診断を識別します。 SQL ステートメントを実行する際に生じる診断ごとに、1 つの整数が割り当てられます。値 1 は最初の診断を示し、2 は 2 番目の診断を示し、以下同様となります。値が 1 の場合、検索される診断情報は (GET DIAGNOSTICS ステートメント以外の) 直前の SQL ステートメントの実行によって実際に戻された SQLSTATE 値によって示される条件に対応します。指定される変数は、位取りがゼロの真数変数の宣言規則に従ってプログラム内で宣言されている必要があります。変数をグローバル変数にすることはできません。指定される値は、1 より小さくてはならず、使用可能な診断の数よりも大きくてもなりません。

variable-3
変数を宣言する規則に従ってプログラム内で宣言された、変数を指定します。変数をグローバル変数にすることはできません。変数のデータ・タイプは、 1508 ページの表 91 で指定の条件情報項目として指定されているデータ・タイプと互換性がなければなりません。変数には、 118 ページの『検索割り当て』で説明されている検索割り当て規則に従って、指定した条件情報項目の値が割り当てられます。その値が変数に割り当てられる際に切り捨てられる場合、エラーが戻されて診断領域の GET_DIAGNOSTICS_DIAGNOSTICS 項目にこの条件の詳細が追加されて更新されます。

指定の診断項目に診断情報が含まれない場合、変数はそのデータ・タイプに基づいてデフォルト値に設定されます。

- 厳密な数の値診断項目は 0、
- VARCHAR 診断項目は空ストリング、
- および CHAR 診断項目はブランクです。

combined-information
1 つのストリングに結合された複数の情報を戻します。

variable-4
変数を宣言する規則に従ってプログラム内で宣言された、変数を指定します。変数をグローバル変数にすることはできません。変数のデータ・タイプは、VARCHAR でなければなりません。変数は、 118 ページの『検索割り当て』で説明されている検索割り当て規則に従って割り当てられます。*variable-4* の長さが、戻される診断ストリング全体を保持するのに十分な長さでない場合、ストリングは切り捨てられ、エラーが戻されて診断領域の GET_DIAGNOSTICS_DIAGNOSTICS 項目にこの条件の詳細が追加されて更新されます。

ALL
最後に実行された SQL ステートメントに設定されたすべての診断項目を、 1 つのストリングに結合するように指示します。ストリングの形式は、以下の形式による、使用可能なすべての診断情報を含むセミコロンで分離したリストです。

項目名=文字形式による項目値;

文字形式による正の数値には、先頭の正符号 (+) は含まれません。ただし、項目が DB2_RETURNED_SQLCODE のときは例外です。その場合には、先頭に正符号 (+) が追加されます。例えば、次のようになります。

```
NUMBER=1;RETURNED_SQLSTATE=02000;DB2_RETURNED_SQLCODE=+100;
```

診断情報を含む項目だけが、ストリングに含まれます。このストリング内には、DB2_GET_DIAGNOSTICS_DIAGNOSTICS および DB2_SQL_NESTING_LEVEL 項目についてのエントリーもありません。

STATEMENT
最後に実行された SQL ステートメントの診断項目を含むすべてのステ

ートメント情報項目 診断項目を、 1 つのストリングに結合するように指示します。形式は、ALL についての上記の説明と同じです。

CONDITION
最後に実行された SQL ステートメントの診断情報を含む条件情報項目診断項目を、1 つのストリングに結合するように指示します。*variable-5* または *integer* を指定した場合、その形式は ALL オプションについての上記の説明と同じになります。*variable-5* または *integer* を指定しない場合、その形式には情報の先頭に、以下の形式による条件に対する条件番号項目が含まれます。

```
CONDITION_NUMBER=X;item-name=character-form-of-the-item-value;
```

X は、条件の番号です。例えば、次のようになります。

```
CONDITION_NUMBER=1;RETURNED_SQLSTATE=02000;DB2_RETURNED_SQLCODE=+100;
CONDITION_NUMBER=2;RETURNED_SQLSTATE=01004;
```

CONNECTION
最後に実行された SQL ステートメントの診断項目を含む接続情報項目診断項目を、1 つのストリングに結合するように指示します。*variable-5* または *integer* を指定した場合、その形式は ALL についての上記の説明と同じになります。*variable-5* または *integer* を指定しない場合、その形式には情報の先頭に、以下の形式による条件に対する接続番号項目が含まれます。

```
DB2_CONNECTION_NUMBER=X;item-name=character-form-of-the-item-value;
```

X は、条件の番号です。例えば、次のようになります。

```
DB2_CONNECTION_NUMBER=1;CONNECTION_NAME=SVL1;DB2_PRODUCT_ID=DSN07010;
```

variable-5 または *integer*
ALL CONDITION または ALL CONNECTION 情報が要求された診断を識別します。指定された変数は、整変数の宣言の規則に従ってプログラムで宣言されていなければなりません。変数をグローバル変数にすることはできません。指定される値は、1 より小さくてはならず、使用可能な診断の数よりも大きくてもなりません。

ステートメント情報項目

COMMAND_FUNCTION
直前の SQL ステートメントの名前を戻します。ステートメント・ストリング値についての詳細は、 1511 ページの表 92 を参照してください。

COMMAND_FUNCTION_CODE
直前の SQL ステートメントを識別する整数を戻します。ステートメント・コード値についての詳細は、 1511 ページの表 92 を参照してください。

DB2_DIAGNOSTIC_CONVERSION_ERROR
GET DIAGNOSTICS ステートメント値の 1 つのために文字データ値を変換するとき変換エラーが生じた場合、値の 1 を戻します。その他の場合は、値 0 を戻します。

DB2_GET_DIAGNOSTICS_DIAGNOSTICS
GET DIAGNOSTICS ステートメントの後に、GET DIAGNOSTICS ステートメントの実行中にエラーまたは警告が生じた場合、

DB2_GET_DIAGNOSTICS_DIAGNOSTICS はそれらのエラーまたは警告に関するテキスト情報を戻します。この情報の形式は、GET DIAGNOSTICS :hv = ALL ステートメントで戻される形式に似ています。

サーバーの DRDA レベルが要求元のクライアントよりも低い場合など、サーバーが理解できない情報項目についての要求が出された場合は、DB2_GET_DIAGNOSTICS_DIAGNOSTICS はテキスト 'Item not supported:' に続けて、要求されてはいてもサーバーがサポートしていない項目名のコンマで区切られたリストを戻します。

DB2_LAST_ROW

multiple-row-fetch ステートメントの場合、取り出された行のセットに、結果表に現在ある最後の行 (順方向に取り出しているカーソルの場合) が含まれているか、結果表に現在ある最初の行 (逆方向に取り出しているカーソルの場合) が含まれていると、値 +100 が戻されることがあります。戻された行の数が要求した行の数に等しい場合は、データ終わりの警告は発生しないことがあり、DB2_LAST_ROW は +100 に設定されないことがあります。

更新に影響を受けにくいカーソルの場合に DB2_LAST_ROW の値 +100 が戻されると、以降の FETCH は、SQLCODE が +100 に、SQLSTATE が「02000」に設定されて戻ります。更新に影響を受けやすいカーソルの場合は、以降の FETCH は、 FETCH が実行される前に挿入された行があれば、より多くのデータを戻すことがあります。

multiple-row-fetch ステートメント以外のステートメントの場合、または、結果表に現在ある最後の行 (順方向に取り出しているカーソルの場合) を含んでいないか、結果表に現在ある最初の行 (逆方向に取り出しているカーソルの場合) を含んでいない *multiple-row-fetch* ステートメントの場合、または、サーバーが SQLCA のみを戻す場合、ゼロ値が戻されます。

DB2_NUMBER_CONNECTIONS

クライアントの要求を実行するサーバーを取得するために確立された接続の数を戻します。それぞれの接続は、単一の条件に対して入手可能となる接続情報の項目領域を生成することがあります。

DB2_NUMBER_PARAMETER_MARKERS

PREPARE ステートメントの場合、準備済みステートメント内のパラメーター・マーカーの数を戻します。その他の場合は、値 0 を戻します。

DB2_NUMBER_RESULT_SETS

CALL ステートメントの場合は、プロシージャーから戻された結果セットの実際の数を戻します。その他の場合は、値 0 を戻します。

DB2_NUMBER_ROWS

直前の SQL ステートメントが OPEN または FETCH であり、それによって結果表のサイズが判明した場合、結果表の行数を戻します。 SENSITIVE カーソルでは、挿入および削除される行はこの値の次回の検索に影響を与えるので、この値は近似とみなされます。その他の場合は、値 0 を戻します。

DB2_NUMBER_SUCCESSFUL_SUBSTMTS

組み込み複合 SQL ステートメントでは、成功したサブステートメントの数を戻します。その他の場合は、値 0 を戻します。

DB2_RELATIVE_COST_ESTIMATE
PREPARE ステートメントの場合は、すべての実行で必要なリソースの相対的なコストの見積もりを戻します。必要な時間の見積もりは反映されません。動的に定義されるステートメントを準備するとき、この値は準備するステートメントの相対コストの標識として使用できます。この値は統計の変更に応じてさまざまであり、製品のリリースごとに異なることがあります。これはオプティマイザーによって選択されたアクセス・プランの見積もりコストです。ステートメントが PREPARE ステートメントではない場合、値のゼロが戻されます。

DB2_RETURN_STATUS
直前の SQL CALL ステートメントから戻された状況値を識別します。直前のステートメントが CALL ステートメントでない場合は、戻される値は意味がなく、予測不能です。詳しくは、 1835 ページの『戻り (RETURN) ステートメント』を参照してください。その他の場合は、値 0 を戻します。

外部プロシージャーについては、SQLCODE < 0 が戻された場合は、SQL_ERROR_CODE1 および DB2_RETURN_STATUS は -1 に設定され、それ以外の場合は、SQL_ERROR_CODE1 および DB2_RETURN_STATUS は 0 に設定されます。

DB2_ROW_COUNT_SECONDARY
直前に実行された SQL ステートメントの 2 次アクションに関連付けられた行数を識別します。直前の SQL ステートメントが DELETE または MERGE の場合、この値はカスケードされたアクションを含む参照制約、および活動化されたトリガーからのトリガー SQL ステートメントの処理の影響を受ける合計行数です。直前の SQL ステートメントが INSERT または UPDATE の場合、この値は、活動化されたトリガーからのトリガー SQL ステートメントの処理の結果として影響を受ける合計行数です。その他の場合は、値 0 を戻します。

分離レベルのコミット不可を使用して SQL ステートメントが実行される場合、この値はゼロになる場合があります。

DB2_ROW_LENGTH
FETCH の場合、結果行に LOB が含まれなければ、取り出された行の長さを戻します。 OPEN の場合は、結果行に LOB が含まれなければ、結果行の長さを戻します。 FETCH と OPEN で、結果に LOB が含まれる場合、戻される長さは予測不能です。その他の場合は、値 0 を戻します。

DB2_SQL_ATTR_CONCURRENCY
OPEN ステートメントの場合、読み取り専用、ロッキング、楽観的使用のタイム・スタンプ、または楽観的使用の値についての、並行性制御オプションを示します。

- R は、読み取り専用を示します。
- L は、ロッキングを示します。
- T は、タイム・スタンプまたは ROWID を使用して行バージョンを比較することを示します。
- V は、値を比較することを示します。

その他の場合は、ブランクを戻します。

DB2_SQL_ATTR_CURSOR_CAPABILITY
OPEN ステートメントの場合、カーソルが読み取り専用、削除可能、または更新可能であるかどうかという、カーソルの機能を示します。

- R は、カーソルが読み取り専用であることを示します。
- D は、カーソルを使用して読み取りおよび削除できることを示します。
- U は、カーソルを使用して読み取り、削除、および更新ができることを示します。

その他の場合は、ブランクを戻します。

DB2_SQL_ATTR_CURSOR_HOLD
OPEN ステートメントの場合、カーソルを複数の作業単位に渡ってオープンしたままにできるかどうかを示します。

- N は、このカーソルが複数の作業単位に渡ってオープンしたままにはならないことを示します。
- Y は、このカーソルが複数の作業単位に渡ってオープンしたままになることを示します。

その他の場合は、ブランクを戻します。

DB2_SQL_ATTR_CURSOR_ROWSET
OPEN ステートメントの場合、行セットによる位置指定を使用してカーソルにアクセスできるかどうかを示します。

- N は、このカーソルが行で位置指定する操作だけをサポートすることを示します。
- Y は、このカーソルが行セットによる位置指定をサポートすることを示します。

その他の場合は、ブランクを戻します。

DB2_SQL_ATTR_CURSOR_SCROLLABLE
OPEN ステートメントの場合、カーソルを前方および後方にスクロールできるかどうかを示します。

- N は、このカーソルがスクロール可能ではないことを示します。
- Y は、このカーソルがスクロール可能であることを示します。

その他の場合は、ブランクを戻します。

DB2_SQL_ATTR_CURSOR_SENSITIVITY
OPEN ステートメントの場合、カーソルが他の接続によるカーソル行の更新を表示するかどうかを示します。

- I は、反応しないことを示します。
- P は、部分的に反応することを示します。
- S は、反応することを示します。
- U は、指定されていないことを示します。

その他の場合は、ブランクを戻します。

DB2_SQL_ATTR_CURSOR_TYPE
OPEN ステートメントの場合、カーソル・タイプが動的、転送のみ、または静的であるかどうかを示します。

- D は、動的カーソルを示します。
- F は、前進のみのカーソルを示します。
- S は、静的カーソルを示します。

その他の場合は、ブランクを戻します。

DB2_SQL_NESTING_LEVEL
GET DIAGNOSTICS ステートメントが実行されたときに有効になっていたネストまたは再帰の現行レベルを識別します。各ネスト・レベルは、関数、プロシージャー、またはトリガーのネスト呼び出しまたは再帰的呼び出しに対応しています。GET DIAGNOSTICS ステートメントがネストのレベル外で実行される場合、値ゼロが戻されます。

並行して実行しているユーザー定義関数内で GET DIAGNOSTICS が発行される場合、ネスト・レベルは予測不能です。

DYNAMIC_FUNCTION
動的に準備または実行される SQL ステートメントのタイプを示す、文字ストリングを戻します。ステートメント・ストリング値についての詳細は、 1511 ページの表 92 を参照してください。

DYNAMIC_FUNCTION_CODE
動的に準備または実行される SQL ステートメントのタイプを示す、数値を戻します。ステートメント・コード値についての詳細は、 1511 ページの表 92 を参照してください。

MORE
処理可能な数よりも多くのエラーが発生したかどうかを示します。

- N は、直前の SQL ステートメントによるすべてのエラーと警告が、診断領域に保管されたことを示します。
- Y は、直前の SQL ステートメントによって発生したエラーと警告の数が、診断領域内の条件領域の数よりも多いことを示します。診断領域の最大サイズは 90K です。

NUMBER
直前の GET DIAGNOSTICS ステートメント以外の SQL ステートメントの実行によって検出された、診断領域に保管されたエラーと警告の数を戻します。直前の SQL ステートメントが成功 (SQLSTATE 00000) を戻したか、または実行された直前の SQL ステートメントが存在しない場合、戻される数値は 1 となります。 GET DIAGNOSTICS ステートメントそのものは SQLSTATE パラメーターを介して情報を戻すことがありますが、DB2_GET_DIAGNOSTICS_DIAGNOSTICS 項目を除いて、診断領域の以前の内容を変更することはありません。

ROW_COUNT
直前に実行された SQL ステートメントに関連付けられた行数を識別します。直前の SQL ステートメントが DELETE、INSERT、REFRESH、または UPDATE ステートメントの場合、ROW_COUNT は、そのステートメントによって削除、挿入、または更新された行数を識別します (ただし、トリガーまたは参照保全制約の影響を受けている行は除きます)。直前の SQL ステートメントが MERGE ステートメントの場合、ROW_COUNT は、そのステートメントによって削除、挿入、および更新された合計行数を識別します (ただし、トリガーま

たは参照保全制約の影響を受けている行は除きます)。直前の SQL ステートメントが複数行の 複数行取り出し、ROW_COUNT は取り出される行数を示します。その他の場合は、値 0 を戻します。

TRANSACTION_ACTIVE
SQL トランザクションが現在アクティブであれば値の 1 を戻し、 SQL トランザクションが現在アクティブでなければ 0 を戻します。

TRANSACTIONS_COMMITTED
直前のステートメントが CALL であった場合、 SQL または外部プロシージャーの実行中にコミットされたトランザクション数を戻します。その他の場合は、値 0 を戻します。

TRANSACTIONS_ROLLED_BACK
直前のステートメントが CALL であった場合、 SQL または外部プロシージャーの実行中にロールバックされたトランザクション数を戻します。その他の場合は、値 0 を戻します。

接続情報項目

CONNECTION_NAME
直前の SQL ステートメントが CONNECT、DISCONNECT、または SET CONNECTION の場合、直前のステートメントで指定されたサーバー名を戻します。その他の場合、現行接続の名前を戻します。

DB2_AUTHENTICATION_TYPE
認証タイプがサーバーかクライアントかを示します。

- C は、クライアント認証を示します。
- E は、DCE セキュリティー・サービス認証を示します。
- S は、サーバー認証を示します。

その他の場合は、ブランクを戻します。

DB2_AUTHORIZATION_ID
接続先のサーバーによって使用される認証 ID を戻します。ユーザー ID の翻訳および与信の出口プログラムのため、ローカルのユーザー ID はサーバーが使用する認証 ID と異なることがあります。

DB2_CONNECTION_METHOD
CONNECT または SET CONNECTION ステートメントでは、接続メソッドを戻します。

- D は、*DUW (分散作業単位) を戻します。
- R は、*RUW (リモート作業単位) を戻します。

DB2_CONNECTION_NUMBER
接続の数を戻します。

DB2_CONNECTION_STATE
接続状態が、接続かどうかを示します。

- -1 は、接続が未接続であることを示します。
- 1 は、接続が接続であることを示します。

その他の場合は、値 0 を戻します。

DB2_CONNECTION_STATUS
コミット可能な更新を実行できるかどうかを示します。

- 1 は、この作業単位の接続で、コミット可能な更新を行うことができることを示します。
- 2 は、この作業単位の接続で、コミット可能な更新を行うことはできないことを示します。

その他の場合は、値 0 を戻します。

DB2_CONNECTION_TYPE
接続タイプ (ローカル、リモート、またはドライバー・プログラム)、および会話が保護されているかどうかを示します。

- 1 は、ローカルのリレーショナル・データベースとの接続を示します。
- 2 は、会話が保護されない、遠隔のリモート・リレーショナル・データベースとの接続を示します。
- 3 は、会話が保護される、遠隔のリモート・リレーショナル・データベースとの接続を示します。
- 4 は、アプリケーション・リクエスターのドライバー・プログラムとの接続を示します。

その他の場合は、値 0 を戻します。

DB2_DYN_QUERY_MGMT
DYN_QUERY_MGMT データベース構成パラメーターが使用可能である場合、値の 1 を戻します。その他の場合は、値 0 を戻します。

DB2_ENCRYPTION_TYPE
暗号化のレベルを戻します。

- A は、認証されたトークン (認証 ID およびパスワード) だけが暗号化されることを示します。
- D は、すべてのデータが接続のために暗号化されていることを示します。

その他の場合は、ブランクを戻します。

DB2_PRODUCT_ID
製品シグニチャーを戻します。アプリケーション・サーバーが IBM リレーショナル・データベースのプロダクトである場合、形式は pppvvrrm となります。ここで、

- ppp は、以下のようにプロダクトを示します。ARI は Db2 (VM および VSE 版)、DSN は Db2 for z/OS、QSQ は Db2 for i、および SQL は他のすべての Db2 プロダクトです。
- vv は、2 桁のバージョン ID です (例えば、'04' など)。
- rr は、2 桁のリリース ID です (例えば、'01' など)。
- m は、1 桁のモディフィケーション・レベルを示します (例えば、'0' など)。

例えば、アプリケーション・サーバーが Db2 for z/OS のバージョン 7 である場合、この値は 'DSN07010' となります。その他の場合は、空ストリングを戻します。

DB2_SERVER_CLASS_NAME
サーバー・クラス名を戻します。例えば、Db2 for z/OS、Db2 for AIX、Db2 for Windows、および Db2 for i などです。

DB2_SERVER_NAME
CONNECT または SET CONNECTION ステートメントの場合、リレーショナル・データベース名を戻します。その他の場合は、空ストリングを戻します。

条件情報項目

CATALOG_NAME
戻される SQLSTATE が、

- クラス 09 (トリガー・アクション例外)、または
- クラス 23 (保全性制約違反)、または
- クラス 27 (トリガー・データ変更違反)、または
- 40002 (トランザクション・ロールバック - 保全性制約違反) の場合、

エラーを生じた制約が参照制約、検査制約、またはユニーク制約であれば、その制約を所有する表のサーバー名を戻します。

戻される SQLSTATE がクラス 42 (構文エラーまたはアクセス規則違反) の場合、エラーを生じた表のサーバー名を戻します。

戻される SQLSTATE がクラス 44 (WITH CHECK OPTION 違反) の場合、エラーを生じたビューのサーバー名を戻します。その他の場合は、空ストリングを戻します。

CLASS_ORIGIN
クラスが ISO 9075 で定義されている SQLSTATE に対しては、'ISO 9075' を戻します。クラスが SQL/MM で定義されている SQLSTATE に対しては、'ISO/IEC 13249' を戻します。クラスが IBM Db2 SQL で定義されている SQLSTATE に対しては、'Db2 SQL' を戻します。使用可能であれば、ユーザー作成コードによって設定された値を戻します。その他の場合は、空ストリングを戻します。

COLUMN_NAME
戻される SQLSTATE がクラス 42 (構文エラーまたはアクセス規則違反) であり、エラーがアクセス不能な列によって生じた場合は、エラーを生じた表のサーバー名を戻します。その他の場合は、空ストリングを戻します。

CONDITION_IDENTIFIER
RETURNED_SQLSTATE の値が 未処理のユーザー定義例外 (SQLSTATE 45000) に対応する場合、そのユーザー定義例外の条件名を戻します。

CONDITION_NUMBER
条件の数を戻します。

CONSTRAINT_CATALOG
戻される SQLSTATE が、

- クラス 23 (保全性制約違反)、または
- クラス 27 (トリガー・データ変更違反)、または
- 40002 (トランザクション・ロールバック - 保全性制約違反) の場合、

エラーを生じた制約を含む表を含む、サーバー名を戻します。その他の場合は、空ストリングを戻します。

CONSTRAINT_NAME
戻される SQLSTATE が、

- クラス 23 (保全性制約違反)、または
- クラス 27 (トリガー・データ変更違反)、または
- 40002 (トランザクション・ロールバック - 保全性制約違反) の場合、

エラーを生じた制約の名前を戻します。その他の場合は、空ストリングを戻します。

CONSTRAINT_SCHEMA
戻される SQLSTATE が、

- クラス 23 (保全性制約違反)、または
- クラス 27 (トリガー・データ変更違反)、または
- 40002 (トランザクション・ロールバック - 保全性制約違反) の場合、

エラーを生じた制約のスキーマ名を戻します。その他の場合は、空ストリングを戻します。

CURSOR_NAME
戻される SQLSTATE がクラス 24 (無効なカーソル状態) の場合、カーソル名を戻します。その他の場合は、空ストリングを戻します。

DB2_ERROR_CODE1
内部エラー・コードを戻します。その他の場合は、値 0 を戻します。

DB2_ERROR_CODE2
内部エラー・コードを戻します。その他の場合は、値 0 を戻します。

DB2_ERROR_CODE3
内部エラー・コードを戻します。その他の場合は、値 0 を戻します。

DB2_ERROR_CODE4
内部エラー・コードを戻します。その他の場合は、値 0 を戻します。

DB2_INTERNAL_ERROR_POINTER
いくつかのエラーについては、これは内部エラー・ポインターである負の値となります。その他の場合は、値 0 を戻します。

DB2_LINE_NUMBER
SQL プロシージャー本体を構文解析中にエラーが検出された SQL 関数、SQL プロシージャー、または SQL トリガーの CREATE PROCEDURE では、エラーが生じた可能性のある行番号を戻します。その他の場合は、値 0 を戻します。

DB2_MESSAGE_ID
MESSAGE_TEXT に対応するメッセージ ID を戻します。

DB2_MESSAGE_ID1
このエラーを最初に生じた、基礎となる IBM i CPF エスケープ・メッセージを戻します。その他の場合は、空ストリングを戻します。

DB2_MESSAGE_ID2
このエラーを最初に生じた、基礎となる IBM i CPD 診断メッセージを戻します。その他の場合は、空ストリングを戻します。

DB2_MESSAGE_KEY
CALL ステートメントの場合、プロシージャーが正常に実行されなかった原因となった、エラーの IBM i メッセージ・キーを戻します。
DELETE、INSERT、または UPDATE ステートメント内のトリガー・エラーの場合、トリガー・プログラムからシグナルで通知されたエラーのメッセージ・キーを戻します。 IBM i QMHRCVPM API は、そのメッセージ・キーおよびメッセージ・データのメッセージ記述を戻すために使用できます。その他の場合は、値 0 を戻します。

DB2_MODULE_DETECTING_ERROR
どのモジュールがエラーを検出したかを示す ID を戻します。ルーチンから発行される SIGNAL ステートメントの場合、値 'ROUTINE' を戻します。その他の SIGNAL ステートメントの場合、値 'PROGRAM' を戻します。

DB2_NUMBER_FAILING_STATEMENTS
NOT ATOMIC 組み込みコンパウンド SQL ステートメントの場合、失敗したステートメントの数を戻します。その他の場合は、値 0 を戻します。

DB2_OFFSET
SQL プロシージャー本体を構文解析中にエラーが検出された SQL プロシージャーの CREATE PROCEDURE では、使用可能な場合、エラーが生じた可能性のある行番号へのオフセットを戻します。ソース・ステートメントを構文解析中にエラーが検出された EXECUTE IMMEDIATE または PREPARE ステートメントでは、エラーが生じた可能性のあるソース・ステートメントへのオフセットを戻します。その他の場合は、値 0 を戻します。

DB2_ORDINAL_TOKEN_n
n 番目のトークンを戻します。 n は、1 から 100 までの値でなければなりません。例えば、DB2_ORDINAL_TOKEN_1 は最初のトークンを戻し、DB2_ORDINAL_TOKEN_2 は 2 番目のトークンを戻します。トークンの数値は、戻される前に文字に変換されます。トークンの値が存在しない場合、空ストリングを戻します。

DB2_PARTITION_NUMBER
パーティション・データベースの場合、エラーまたは警告が検出されたデータベース・パーティションのパーティション番号を戻します。エラーまたは警告が検出されなかった場合、現行ノードのパーティション番号を戻します。その他の場合は、値 0 を戻します。

DB2_REASON_CODE
メッセージ・テキスト内に理由コード・トークンがあるエラーの理由コードを戻します。その他の場合は、値 0 を戻します。

DB2_RETURNED_SQLCODE
指定された診断の SQLCODE を戻します。

DB2_ROW_NUMBER
直前の SQL ステートメントが複数行の挿入または複数行の取り出しである場合、条件が検出された行の番号が使用可能で適用可能な場合、その値を戻します。その他の場合は、値 0 を戻します。

DB2_SQLERRD_SET
DB2_SQLERRD1 から DB2_SQLERRD6 の項目が設定可能であることを示すには、Y を戻します。その他の場合は、ブランクを戻します。

DB2_SQLERRD1
サーバーによって戻された SQLCA から、値の SQLERRD(1) を戻します。

DB2_SQLERRD2
サーバーによって戻された SQLCA から、値の SQLERRD(2) を戻します。

DB2_SQLERRD3
サーバーによって戻された SQLCA から、値の SQLERRD(3) を戻します。

DB2_SQLERRD4
サーバーによって戻された SQLCA から、値の SQLERRD(4) を戻します。

DB2_SQLERRD5
サーバーによって戻された SQLCA から、値の SQLERRD(5) を戻します。

DB2_SQLERRD6
サーバーによって戻された SQLCA から、値の SQLERRD(6) を戻します。

DB2_TOKEN_COUNT
指定された診断での、使用可能なトークンの数を戻します。

DB2_TOKEN_STRING
指定された診断についての、X'FF' 区切り文字で区切られているトークンのストリングを戻します。

MESSAGE_LENGTH
直前に実行された SQL ステートメントから戻されたエラー、警告、または正常な完了のメッセージ・テキストの長さを (文字数で) 示します。

MESSAGE_OCTET_LENGTH
直前に実行された SQL ステートメントから戻されたエラー、警告、または正常な完了のメッセージ・テキストの長さを (バイト数で) 示します。

MESSAGE_TEXT
直前に実行された SQL ステートメントから戻されたエラー、警告、または正常な完了のメッセージ・テキストを示します。

SQLCODE が 0 の場合は、空ストリングが戻されます。これは、RETURNED_SQLSTATE の値が警告状態を示す場合にも該当します。

PARAMETER_MODE
戻される SQLSTATE が、

- クラス 39 (外部ルーチン呼び出し例外)、または
- クラス 38 (外部ルーチン例外)、または
- クラス 2F (SQL ルーチン例外)、または
- クラス 22 (データ例外)、または
- クラス 23 (保全性制約違反)、または
- クラス 01 (警告)

であり、条件がルーチンの *i* 番目のパラメーターに関連する場合、 *i* 番目のパラメーターのパラメーター・モードを戻します。その他の場合は、空ストリングを戻します。

PARAMETER_NAME

戻される SQLSTATE が、

- クラス 39 (外部ルーチン呼び出し例外)、または
- クラス 38 (外部ルーチン例外)、または
- クラス 2F (SQL ルーチン例外)、または
- クラス 22 (データ例外)、または
- クラス 23 (保全性制約違反)、または
- クラス 01 (警告)

であり、条件がルーチンの *i* 番目のパラメーターに関連していて、ルーチンの作成時にパラメーターに対してパラメーター名が指定されている場合、 *i* 番目のパラメーターのパラメーター名を戻します。その他の場合は、空ストリングを戻します。

PARAMETER_ORDINAL_POSITION

戻される SQLSTATE が、

- クラス 39 (外部ルーチン呼び出し例外)、または
- クラス 38 (外部ルーチン例外)、または
- クラス 2F (SQL ルーチン例外)、または
- クラス 22 (データ例外)、または
- クラス 23 (保全性制約違反)、または
- クラス 01 (警告)

であり、条件がルーチンの *i* 番目のパラメーターに関連する場合、 *i* の値を戻します。その他の場合は、空ストリングを戻します。

RETURNED_SQLSTATE

指定された診断の SQLSTATE を戻します。

ROUTINE_CATALOG

戻される SQLSTATE が、

- クラス 39 (外部ルーチン呼び出し例外)、または
- クラス 38 (外部ルーチン例外)、または
- クラス 2F (SQL ルーチン例外)、または

であり、条件がルーチンの *i* 番目のパラメーターに関連する場合、または戻される SQLSTATE が以下の場合であって、

- クラス 22 (データ例外)、または
- クラス 23 (保全性制約違反)、または
- クラス 01 (警告)

ルーチン呼び出しの際に SQL パラメーターに割り当てを行った結果として条件が発生した場合は、ルーチンのサーバー名を戻します。その他の場合は、空ストリングを戻します。

ROUTINE_NAME

戻される SQLSTATE が、

- クラス 39 (外部ルーチン呼び出し例外)、または
- クラス 38 (外部ルーチン例外)、または

- クラス 2F (SQL ルーチン例外)、または

であり、条件がルーチンの *i* 番目のパラメーターに関連する場合、または戻される SQLSTATE が以下の場合であって、

- クラス 22 (データ例外)、または
- クラス 23 (保全性制約違反)、または
- クラス 01 (警告)

ルーチン呼び出しの際に SQL パラメーターに割り当てを行った結果として条件が発生した場合は、ルーチンの名前を戻します。その他の場合は、空ストリングを戻します。

ROUTINE_SCHEMA
戻される SQLSTATE が、

- クラス 39 (外部ルーチン呼び出し例外)、または
- クラス 38 (外部ルーチン例外)、または
- クラス 2F (SQL ルーチン例外)、または

であり、条件がルーチンの *i* 番目のパラメーターに関連する場合、または戻される SQLSTATE が以下の場合であって、

- クラス 22 (データ例外)、または
- クラス 23 (保全性制約違反)、または
- クラス 01 (警告)

ルーチン呼び出しの際に SQL パラメーターに割り当てを行った結果として条件が発生した場合は、ルーチンのスキーマ名を戻します。その他の場合は、空ストリングを戻します。

SCHEMA_NAME
戻される SQLSTATE が、

- クラス 09 (トリガー・アクション例外)、または
- クラス 23 (保全性制約違反)、または
- クラス 27 (トリガー・データ変更違反)、または
- 40002 (トランザクション・ロールバック - 保全性制約違反) の場合、

エラーを生じた制約が参照制約、検査制約、またはユニーク制約であれば、その制約を所有する表のスキーマ名を戻します。

戻される SQLSTATE がクラス 42 (構文エラーまたはアクセス規則違反) の場合、エラーを生じた表のスキーマ名を戻します。

戻される SQLSTATE がクラス 44 (WITH CHECK OPTION 違反) の場合、エラーを生じたビューのスキーマ名を戻します。その他の場合は、空ストリングを戻します。

SERVER_NAME
直前の SQL ステートメントが CONNECT、DISCONNECT、または SET CONNECTION の場合、直前のステートメントで指定されたサーバー名を戻します。その他の場合、ステートメントが実行されたサーバーの名前を戻します。

SPECIFIC_NAME
戻される SQLSTATE が、

- クラス 39 (外部ルーチン呼び出し例外)、または
- クラス 38 (外部ルーチン例外)、または
- クラス 2F (SQL ルーチン例外)、または

であり、条件がルーチンの *i* 番目のパラメーターに関連する場合、または戻される SQLSTATE が以下の場合であって、

- クラス 22 (データ例外)、または
- クラス 23 (保全性制約違反)、または
- クラス 01 (警告)

ルーチン呼び出しの際に SQL パラメーターに割り当てを行った結果として条件が発生した場合は、プロシージャーまたは関数の特定名を戻します。その他の場合は、空ストリングを戻します。

SUBCLASS_ORIGIN
サブクラスが ISO 9075 で定義されている SQLSTATE に対しては、'ISO 9075' を戻します。サブクラスが RDA で定義されている SQLSTATE に対しては、'ISO/IEC 9579' を戻します。サブクラスが SQL/MM で定義されている SQLSTATE に対しては、 'ISO/IEC 13249-1'、'ISO/IEC 13249-2'、 'ISO/IEC 13249-3'、'ISO/IEC 13249-4'、または 'ISO/IEC 13249-5' を戻します。サブクラスが IBM Db2 SQL で定義されている SQLSTATE に対しては、'Db2 SQL' を戻します。使用可能であれば、ユーザー作成コードによって設定された値を戻します。その他の場合は、空ストリングを戻します。

TABLE_NAME
戻される SQLSTATE が、

- クラス 09 (トリガー・アクション例外)、または
- クラス 23 (保全性制約違反)、または
- クラス 27 (トリガー・データ変更違反)、または
- 40002 (トランザクション・ロールバック - 保全性制約違反) の場合、

エラーを生じた制約が参照制約、検査制約、またはユニーク制約であれば、その制約を所有する表の名前を戻します。

戻される SQLSTATE がクラス 42 (構文エラーまたはアクセス規則違反) の場合、エラーを生じた表の名前を戻します。

戻される SQLSTATE がクラス 44 (WITH CHECK OPTION 違反) の場合、エラーを生じた表の名前を戻します。その他の場合は、空ストリングを戻します。

TRIGGER_CATALOG
戻される SQLSTATE が、

- クラス 09 (トリガー・アクション例外)、または
- クラス 27 (トリガー・データ変更違反) の場合、

トリガーの名前を戻します。その他の場合は、空ストリングを戻します。

TRIGGER_NAME
戻される SQLSTATE が、

- クラス 09 (トリガー・アクション例外)、または
- クラス 27 (トリガー・データ変更違反) の場合、

トリガーの名前を戻します。その他の場合は、空ストリングを戻します。

TRIGGER_SCHEMA
戻される SQLSTATE が、

- クラス 09 (トリガー・アクション例外)、または
- クラス 27 (トリガー・データ変更違反) の場合、

トリガーのスキーマ名を戻します。その他の場合は、空ストリングを戻します。

注

診断領域に関する考慮事項**:** GET DIAGNOSTICS ステートメントは、DB2_GET_DIAGNOSTICS_DIAGNOSTICS を除き、診断領域の内容を変更することはありません。

SQL 関数、SQL プロシージャー、またはトリガー内で GET DIAGNOSTICS ステートメントが指定される場合、次のとおりです。

- エラーに関する情報が必要な場合、GET DIAGNOSTICS ステートメントは、エラーを処理するハンドラー内に指定される最初の実行可能ステートメントでなければなりません。
- 警告に関する情報が必要な場合:
  - ハンドラーがその警告条件に対する制御を取得する場合、GET DIAGNOSTICS ステートメントは、そのハンドラーに指定された最初のステートメントでなければなりません。
  - ハンドラーがその警告条件に対する制御を取得しない場合、GET DIAGNOSTICS ステートメントは、その直前のステートメントの次に実行されるステートメントでなければなりません。

それ以外の場合、GET DIAGNOSTICS ステートメントは、最後に実行されたステートメントに関する情報を戻します。

SQLCODE および **SQLSTATE SQL** 変数に関する考慮事項**:** GET DIAGNOSTICS ステートメントは、SQLSTATE および SQLCODE SQL 変数の値を変更します。

戻り値の大文字小文字の区別**:** 戻される診断項目に含まれる ID の値は、引用符で区切られず、大文字小文字を区別します。例えば、表の名前 "abc" は単に abc として戻されます。

変数の割り当て**:** 割り当てエラーが発生すると、変数の値は予測不能になります。

項目のデータ・タイプ**:** 以下の表は、診断項目ごとの SQL データ・タイプを示しています。診断項目が変数に割り当てられるとき、変数は診断項目のデータ・タイプと互換性がなければなりません。

表 91. GET DIAGNOSTICS 項目のデータ・タイプ

項目名	データ・タイプ
ステートメント情報項目	
COMMAND_FUNCTION	VARCHAR(128)
COMMAND_FUNCTION_CODE	INTEGER
DB2_DIAGNOSTIC_CONVERSION_ERROR	INTEGER

表 91. GET DIAGNOSTICS 項目のデータ・タイプ (続き)

項目名	データ・タイプ
DB2_GET_DIAGNOSTICS_DIAGNOSTICS	VARCHAR(32740)
DB2_LAST_ROW	INTEGER
DB2_NUMBER_CONNECTIONS	INTEGER
DB2_NUMBER_PARAMETER_MARKERS	INTEGER
DB2_NUMBER_RESULT_SETS	INTEGER
DB2_NUMBER_ROWS	DECIMAL(31,0)
DB2_NUMBER_SUCCESSFUL_SUBSTMTS	INTEGER
DB2_RELATIVE_COST_ESTIMATE	INTEGER
DB2_RETURN_STATUS	INTEGER
DB2_ROW_COUNT_SECONDARY	DECIMAL(31,0)
DB2_ROW_LENGTH	INTEGER
DB2_SQL_ATTR_CONCURRENCY	CHAR(1)
DB2_SQL_ATTR_CURSOR_CAPABILITY	CHAR(1)
DB2_SQL_ATTR_CURSOR_HOLD	CHAR(1)
DB2_SQL_ATTR_CURSOR_ROWSET	CHAR(1)
DB2_SQL_ATTR_CURSOR_SCROLLABLE	CHAR(1)
DB2_SQL_ATTR_CURSOR_SENSITIVITY	CHAR(1)
DB2_SQL_ATTR_CURSOR_TYPE	CHAR(1)
DB2_SQL_NESTING_LEVEL	INTEGER
DYNAMIC_FUNCTION	VARCHAR(128)
DYNAMIC_FUNCTION_CODE	INTEGER
MORE	CHAR(1)
NUMBER	INTEGER
ROW_COUNT	DECIMAL(31,0)
TRANSACTION_ACTIVE	INTEGER
TRANSACTIONS_COMMITTED	INTEGER
TRANSACTIONS_ROLLED_BACK	INTEGER
接続情報項目	
CONNECTION_NAME	VARCHAR(128)
DB2_AUTHENTICATION_TYPE	CHAR(1)
DB2_AUTHORIZATION_ID	VARCHAR(128)
DB2_CONNECTION_METHOD	CHAR(1)
DB2_CONNECTION_NUMBER	INTEGER
DB2_CONNECTION_STATE	INTEGER
DB2_CONNECTION_STATUS	INTEGER
DB2_CONNECTION_TYPE	SMALLINT
DB2_DYN_QUERY_MGMT	INTEGER
DB2_ENCRYPTION_TYPE	CHAR(1)
DB2_PRODUCT_ID	VARCHAR(8)
DB2_SERVER_CLASS_NAME	VARCHAR(128)

表 91. GET DIAGNOSTICS 項目のデータ・タイプ (続き)

項目名	データ・タイプ
DB2_SERVER_NAME	VARCHAR(128)
条件情報項目	
CATALOG_NAME	VARCHAR(128)
CLASS_ORIGIN	VARCHAR(128)
COLUMN_NAME	VARCHAR(128)
CONDITION_IDENTIFIER	VARCHAR(128)
CONDITION_NUMBER	INTEGER
CONSTRAINT_CATALOG	VARCHAR(128)
CONSTRAINT_NAME	VARCHAR(128)
CONSTRAINT_SCHEMA	VARCHAR(128)
CURSOR_NAME	VARCHAR(128)
DB2_ERROR_CODE1	INTEGER
DB2_ERROR_CODE2	INTEGER
DB2_ERROR_CODE3	INTEGER
DB2_ERROR_CODE4	INTEGER
DB2_INTERNAL_ERROR_POINTER	INTEGER
DB2_LINE_NUMBER	INTEGER
DB2_MESSAGE_ID	CHAR(10)
DB2_MESSAGE_ID1	VARCHAR(7)
DB2_MESSAGE_ID2	VARCHAR(7)
DB2_MESSAGE_KEY	INTEGER
DB2_MODULE_DETECTING_ERROR	VARCHAR(128)
DB2_NUMBER_FAILING_STATEMENTS	INTEGER
DB2_OFFSET	INTEGER
DB2_ORDINAL_TOKEN_n	VARCHAR(32740)
DB2_PARTITION_NUMBER	INTEGER
DB2_REASON_CODE	INTEGER
DB2_RETURNED_SQLCODE	INTEGER
DB2_ROW_NUMBER	INTEGER
DB2_SQLERRD_SET	CHAR(1)
DB2_SQLERRD1	INTEGER
DB2_SQLERRD2	INTEGER
DB2_SQLERRD3	INTEGER
DB2_SQLERRD4	INTEGER
DB2_SQLERRD5	INTEGER
DB2_SQLERRD6	INTEGER
DB2_TOKEN_COUNT	INTEGER
DB2_TOKEN_STRING	VARCHAR(1000)
MESSAGE_LENGTH	INTEGER
MESSAGE_OCTET_LENGTH	INTEGER

表 *91. GET DIAGNOSTICS* 項目のデータ・タイプ *(*続き*)*

項目名	データ・タイプ
MESSAGE_TEXT	VARCHAR(32740)
PARAMETER_MODE	VARCHAR(5)
PARAMETER_NAME	VARCHAR(128)
PARAMETER_ORDINAL_POSITION	INTEGER
RETURNED_SQLSTATE	CHAR(5)
ROUTINE_CATALOG	VARCHAR(128)
ROUTINE_NAME	VARCHAR(128)
ROUTINE_SCHEMA	VARCHAR(128)
SCHEMA_NAME	VARCHAR(128)
SERVER_NAME	VARCHAR(128)
SPECIFIC_NAME	VARCHAR(128)
SUBCLASS_ORIGIN	VARCHAR(128)
TABLE_NAME	VARCHAR(128)
TRIGGER_CATALOG	VARCHAR(128)
TRIGGER_NAME	VARCHAR(128)
TRIGGER_SCHEMA	VARCHAR(128)

SQL ステートメントのコードおよびストリング**:** 以下の表は、COMMAND_FUNCTION、COMMAND_FUNCTION_CODE、DYNAMIC_FUNCTION、および DYNAMIC_FUNCTION_CODE 診断項目の可能な値を示しています。

以下の表の値は、ISO および ANSI SQL Standard によって割り当てられていて、この規格の発展に応じて変更される可能性があります。これらの値を参照するときには、ライブラリー QSYSINC 内のインクルード・ソース・ファイルにあるインクルード *sqlscds* を使用してください。

表 *92. SQL* ステートメントのコードおよびストリング

ステートメントのタイプ	ステートメント・ストリング	ステートメント・コード
ALLOCATE CURSOR	ALLOCATE CURSOR	1
ALLOCATE DESCRIPTOR	ALLOCATE DESCRIPTOR	2
ALTER FUNCTION	ALTER ROUTINE	17
ALTER MASK	ALTER MASK	–100
ALTER PERMISSION	ALTER PERMISSION	–103
ALTER PROCEDURE	ALTER ROUTINE	17
ALTER SEQUENCE	ALTER SEQUENCE	134
ALTER TABLE	ALTER TABLE	4
ALTER TRIGGER	ALTER TRIGGER	–99
割り当て (assignment) ステートメント	ASSIGNMENT	5

表 92. SQL ステートメントのコードおよびストリング (続き)

ステートメントのタイプ	ステートメント・ストリング	ステートメント・コード
ASSOCIATE LOCATORS	ASSOCIATE LOCATORS	-6
CALL	CALL	7
CASE	CASE	86
CLOSE (静的 SQL)	CLOSE CURSOR	9
CLOSE (動的 SQL)	DYNAMIC CLOSE CURSOR	37
COMMENT	COMMENT	-7
COMMIT	COMMIT WORK	11
複合 (compound) ステートメント	BEGIN END	12
CONNECT	CONNECT	13
CREATE ALIAS	CREATE ALIAS	–8
CREATE FUNCTION	CREATE ROUTINE	14
CREATE INDEX	CREATE INDEX	-14
CREATE MASK	CREATE MASK	–101
CREATE PERMISSION	CREATE PERMISSION	–104
CREATE PROCEDURE	CREATE ROUTINE	14
CREATE SCHEMA	CREATE SCHEMA	64
CREATE SEQUENCE	CREATE SEQUENCE	133
CREATE TABLE	CREATE TABLE	77
CREATE TRIGGER	CREATE TRIGGER	80
CREATE TYPE	CREATE TYPE	83
CREATE VARIABLE	CREATE VARIABLE	-83
CREATE VIEW	CREATE VIEW	84
DEALLOCATE DESCRIPTOR	DEALLOCATE DESCRIPTOR	15
DECLARE GLOBAL TEMPORARY TABLE	DECLARE GLOBAL TEMPORARY TABLE	-21
位置指定された DELETE (静的 SQL)	DELETE CURSOR	18
位置指定された DELETE (動的 SQL)	DYNAMIC DELETE CURSOR	38
検索された DELETE	DELETE WHERE	19
DESCRIBE	DESCRIBE	20
DESCRIBE CURSOR	DESCRIBE CURSOR RESULT SET	-72
DESCRIBE PROCEDURE	DESCRIBE PROCEDURE	-23
DESCRIBE TABLE	DESCRIBE TABLE	-24
DISCONNECT	DISCONNECT	22
DROP ALIAS	DROP ALIAS	-25
DROP FUNCTION	DROP ROUTINE	30
DROP INDEX	DROP INDEX	-30
DROP MASK	DROP MASK	–102
DROP PACKAGE	DROP PACKAGE	-32

表 92. SQL ステートメントのコードおよびストリング (続き)

ステートメントのタイプ	ステートメント・ストリング	ステートメント・コード
DROP PERMISSION	DROP PERMISSION	–105
DROP PROCEDURE	DROP ROUTINE	30
DROP SCHEMA	DROP SCHEMA	31
DROP SEQUENCE	DROP SEQUENCE	135
DROP TABLE	DROP TABLE	32
DROP TRIGGER	DROP TRIGGER	34
DROP TYPE	DROP TYPE	35
DROP VARIABLE	DROP VARIABLE	-84
DROP XSROBJECT	DROP XSROBJECT	-95
DROP VIEW	DROP VIEW	36
EXECUTE	EXECUTE	44
EXECUTE IMMEDIATE	EXECUTE IMMEDIATE	43
FETCH (静的 SQL)	FETCH	45
FETCH (動的 SQL)	DYNAMIC FETCH	39
FOR	FOR	46
FREE LOCATOR	FREE LOCATOR	98
GET DESCRIPTOR	GET DESCRIPTOR	47
GOTO	GOTO	-37
GRANT (任意のタイプ)	GRANT	48
HOLD LOCATOR	HOLD LOCATOR	99
IF	IF	88
INSERT	INSERT	50
ITERATE	ITERATE	102
LABEL	LABEL	-39
LEAVE	LEAVE	89
LOCK TABLE	LOCK TABLE	-40
LOOP	LOOP	90
MERGE	MERGE	128
OPEN (静的 SQL)	OPEN	53
OPEN (動的 SQL)	DYNAMIC OPEN	40
PREPARE	PREPARE	56
準備済みの位置指定された DELETE (動的 SQL)	PREPARABLE DYNAMIC DELETE CURSOR	54
準備済みの位置指定された UPDATE (動的 SQL)	PREPARABLE DYNAMIC UPDATE CURSOR	55
REFRESH TABLE	REFRESH TABLE	-41
RELEASE (接続)	RELEASE CONNECTION	-42
RELEASE SAVEPOINT	RELEASE SAVEPOINT	57

表 92. SQL ステートメントのコードおよびストリング (続き)

ステートメントのタイプ	ステートメント・ストリング	ステートメント・コード
RENAME INDEX	RENAME INDEX	-43
RENAME TABLE	RENAME TABLE	-44
REPEAT	REPEAT	95
RESIGNAL	RESIGNAL	91
RETURN	RETURN	58
REVOKE (任意のタイプ)	REVOKE	59
ROLLBACK	ROLLBACK WORK	62
SAVEPOINT	SAVEPOINT	63
SELECT INTO	SELECT	65
選択ステートメント (動的 SQL)	SELECT CURSOR	85
SET CONNECTION	SET CONNECTION	67
SET CURRENT DEBUG MODE	SET CURRENT DEBUG MODE	-75
SET CURRENT DECFLOAT ROUNDING MODE	SET CURRENT DECFLOAT ROUNDING MODE	-82
SET CURRENT DEGREE	SET CURRENT DEGREE	-47
SET CURRENT IMPLICIT XMLPARSE OPTION	SET CURRENT IMPLICIT XMLPARSE OPT	-90
SET CURRENT TEMPORAL SYSTEM_TIME	SET CURRENT TEMPORAL SYSTEM_TIME	–98
SET DESCRIPTOR	SET DESCRIPTOR	70
SET ENCRYPTION PASSWORD	SET ENCRYPTION PASSWORD	-48
SET PATH	SET PATH	69
SET RESULT SETS	SET RESULT SETS	-64
SET SCHEMA	SET SCHEMA	74
SET SESSION AUTHORIZATION	SET SESSION AUTHORIZATION	76
SET TRANSACTION	SET TRANSACTION	75
SET 遷移変数	ASSIGNMENT	5
SET 変数	ASSIGNMENT	5
SIGNAL	SIGNAL	92
TRANSFER OWNERSHIP	TRANSFER OWNERSHIP	–77
TRUNCATE TABLE	TRUNCATE TABLE	–74
位置指定された UPDATE (静的 SQL)	UPDATE CURSOR	81
位置指定された UPDATE (動的 SQL)	DYNAMIC UPDATE CURSOR	42
検索された UPDATE	UPDATE WHERE	82
VALUES	STANDALONE FULLSELECT	-69
VALUES INTO	VALUES INTO	-66
WHILE	WHILE	97

表 92. SQL ステートメントのコードおよびストリング (続き)

ステートメントのタイプ	ステートメント・ストリング	ステートメント・コード
認識されないステートメント	長さがゼロのストリング	0

代替構文: 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード EXCEPTION を CONDITION の同義語として使用することができます。
- キーワード RETURN_STATUS を DB2_RETURN_STATUS の同義語として使用することができます。

例

SQL プロシージャーにおいて、GET DIAGNOSTICS ステートメントを実行して、更新された行数を判別します。

```
CREATE PROCEDURE sqlprocg (IN deptnbr VARCHAR(3))
  LANGUAGE SQL
  BEGIN
    DECLARE SQLSTATE CHAR(5);
    DECLARE rcount INTEGER;
    UPDATE CORPDATA.PROJECT
      SET PRSTAFF = PRSTAFF + 1.5
      WHERE DEPTNO = deptnbr;
    GET DIAGNOSTICS rcount = ROW_COUNT;
    /* At this point, rcount contains the number of rows that were updated. */
  END
```

SQL プロシージャー内部で、TRYIT というストアード・プロシージャーの呼び出しから戻された状況値を処理します。 TRYIT で RETURN ステートメントを使用して状況値を明示的に戻すこともでき、データベース・マネージャーから状況値が暗黙的に戻されることもあります。このプロシージャーは、正常に実行されると、値ゼロを戻します。

```
CREATE PROCEDURE TESTIT ()
  LANGUAGE SQL
  A1: BEGIN
      DECLARE RETVAL INTEGER DEFAULT 0;
      ...
      CALL TRYIT
      GET DIAGNOSTICS RETVAL = RETURN_STATUS;
      IF RETVAL <> 0 THEN
         ...
         LEAVE A1;
      ELSE
         ...
      END IF;
  END A1
```

SQL プロシージャーで、GET DIAGNOSTICS ステートメントを実行して、エラーのメッセージ・テキストを取り出します。

```
CREATE PROCEDURE divide2 ( IN  numerator INTEGER,
                           IN  denominator INTEGER,
                           OUT divide_result INTEGER,
```

```
                                OUT divide_error VARCHAR(70) )
LANGUAGE SQL
  BEGIN
     DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
        GET DIAGNOSTICS CONDITION 1
          divide_error = MESSAGE_TEXT;
     SET divide_result = numerator / denominator;
  END;
```

GRANT (関数特権またはプロシージャー特権)

この形式の GRANT ステートメントは、関数またはプロシージャーに対する特権を認可します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

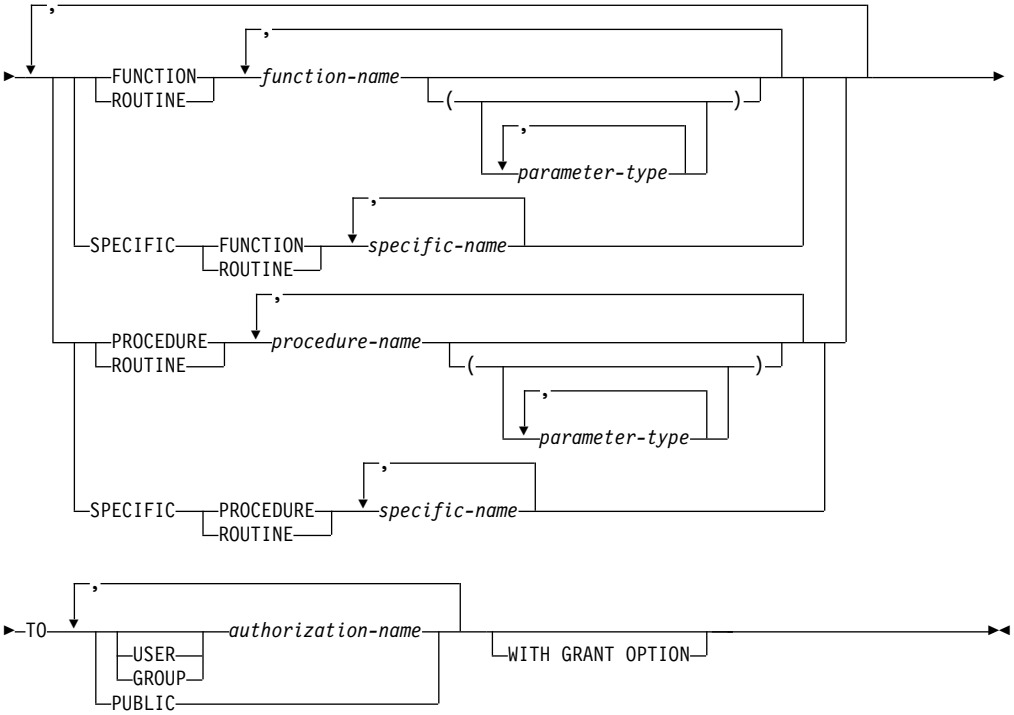
このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれの関数またはプロシージャーごとに、
  - このステートメントで指定されるすべての特権
  - その関数またはプロシージャーに対する *OBJMGT システム権限
  - その関数またはプロシージャーが入っているスキーマに対する USAGE 特権 (これが Java ルーチンの場合は、ディレクトリーに対する *EXECUTE システム権限)
- データベース管理者権限
- セキュリティー管理者権限

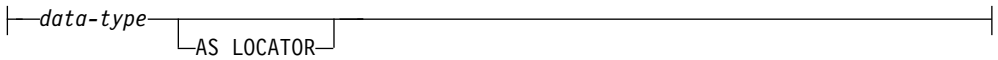
WITH GRANT OPTION を指定する場合、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- その関数またはプロシージャーの所有権
- データベース管理者権限
- セキュリティー管理者権限

構文

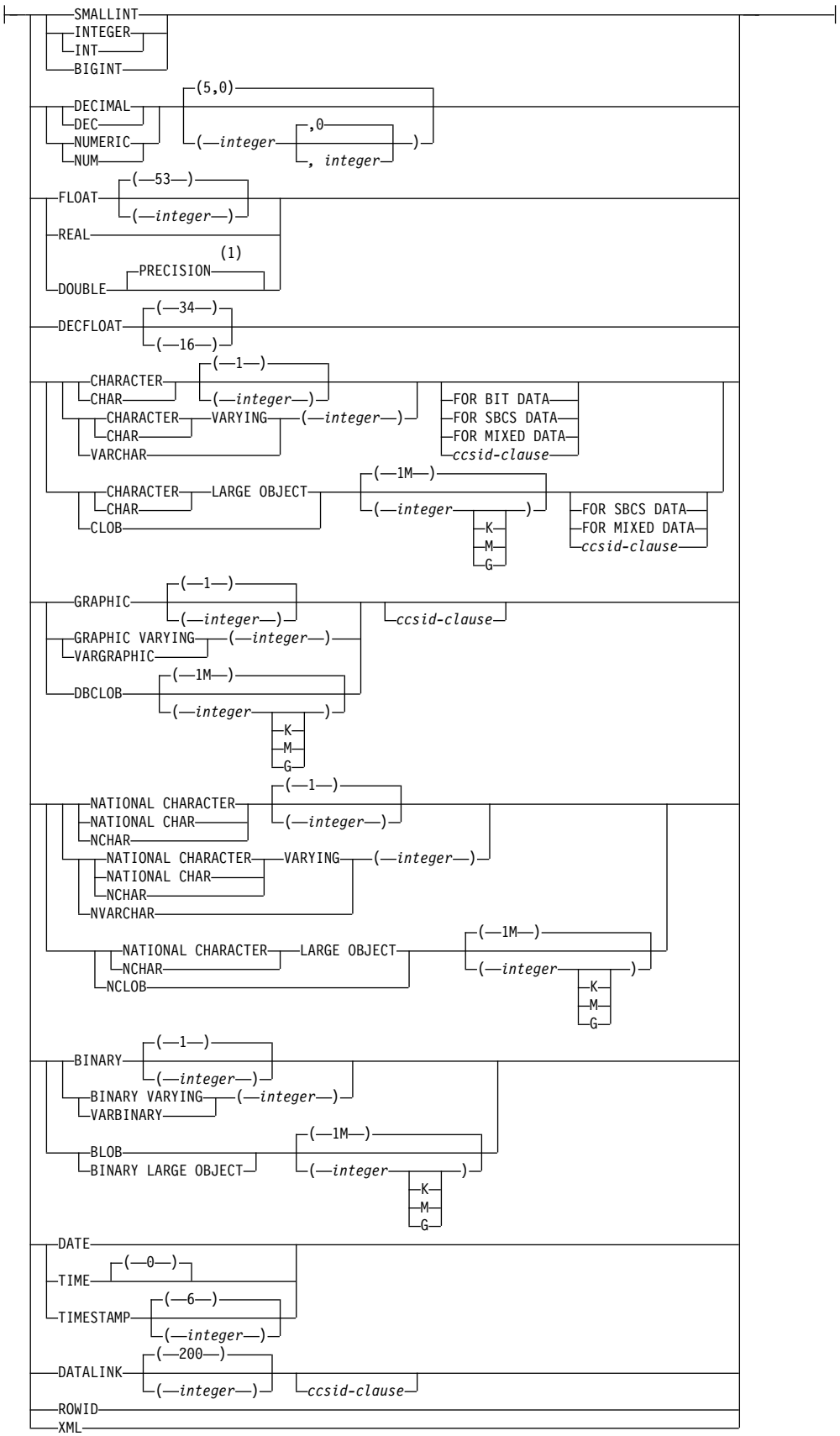


parameter-type:

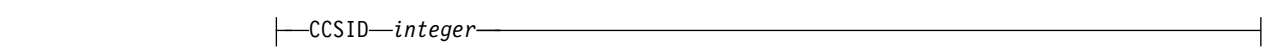


data-type:

built-in-type:


SMALLINT
INTEGER
INT
BIGINT
DECIMAL
DEC
NUMERIC
NUM
(5,0)
(integer
,0
, integer
)
FLOAT
(53)
(integer)
REAL
DOUBLE
PRECISION
(1)
DECFLOAT
(34)
(16)
CHARACTER
CHAR
(1)
(integer)
CHARACTER VARYING
CHAR VARYING
VARCHAR
FOR BIT DATA
FOR SBCS DATA
FOR MIXED DATA
ccsid-clause
CHARACTER LARGE OBJECT
CHAR LARGE OBJECT
CLOB
(1M)
K
M
G
GRAPHIC
GRAPHIC VARYING
VARGRAPHIC
DBCLOB
NATIONAL CHARACTER
NATIONAL CHAR
NCHAR
NATIONAL CHARACTER VARYING
NATIONAL CHAR VARYING
NVARCHAR
NATIONAL CHARACTER LARGE OBJECT
NCHAR LARGE OBJECT
NCLOB
BINARY
BINARY VARYING
VARBINARY
BLOB
BINARY LARGE OBJECT
DATE
TIME
(0)
TIMESTAMP
(6)
DATALINK
(200)
ROWID
XML

ccsid-clause:



注:

1 突き合わせは、データ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度に指定された値は、関数の作成時に指定された値に一致している必要はありません。

説明

ALL または ALL PRIVILEGES
1 つ以上の特権を認可します。認可される特権は、指定された関数またはプロシージャーに対してステートメントの権限 ID が持っている認可可能な特権のすべてです。関数またはプロシージャーに対する ALL PRIVILEGES を認可するのは、 *ALL システム権限を認可するのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つ以上を使用する必要があります。各キーワードは、そこで説明している特権を認可します。

ALTER
ALTER FUNCTION、ALTER PROCEDURE、または COMMENT ステートメントを使用する特権を許可します。

EXECUTE
関数またはプロシージャーを実行するための特権を認可します。

FUNCTION または SPECIFIC FUNCTION
特権が認可される関数を指定します。その関数は現行サーバーに存在していて、ユーザー定義関数でなければなりません。ただし、特殊タイプの作成時に暗黙に生成された関数であってはなりません。関数は、それぞれその名前、関数シグニチャー、あるいは特定名によって識別することができます。

FUNCTION *function-name*
関数を名前によって識別します。 *function-name* は厳密に 1 つの関数を示す必要があります。この関数には、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前の関数が複数ある場合、エラーが戻されます。

FUNCTION *function-name (parameter-type, ...)*
関数を一意的に識別する関数シグニチャーによって、関数を識別します。 *function-name (parameter-type,...)* は、指定された関数シグニチャーを持つ関数を識別しなければなりません。指定されたパラメーターは、関数の作成時に指定された、対応する位置にあるデータ・タイプと一致していなければなりません。特権が認可される関数インスタンスを識別する場合、データ・タイプの数とデータ・タイプの論理連結が使用されます。データ・タイプの同義語は、一致として扱われます。デフォルトがあるパラメーターは、このシグニチャーに含まれていなければなりません。

function-name () を指定する場合、識別される関数にパラメーターを使用することはできません。

function-name
関数の名前を識別します。

(parameter-type, ...)
関数のパラメーターを識別します。

非修飾の特殊タイプ名を指定する場合、データベース・マネージャーはその特殊タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、以下のいずれかを使用します。

- 空の括弧は、データベース・マネージャーがデータ・タイプの一致の判別に際して属性を無視することを示します。例えば、DEC() は、DEC(7,2) のデータ・タイプで定義された関数のパラメーターに一致するものとみなされます。ただし、FLOAT に空の括弧を指定することはできません。これは、そのパラメーター値が特定のデータ・タイプ (REAL または DOUBLE) を示しているからです。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を指定した場合、その値は、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプが FLOAT の場合、突き合わせはデータ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度は指定された値に厳密に一致している必要はありません。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。暗黙の長さは、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

FOR DATA 文節または CCSID 文節の指定はオプショナルです。いずれの文節も指定しないと、データ・タイプが一致するかどうかを判定する場合に、データベース・マネージャーが属性を無視することを示します。どちらか一方の文節を指定する場合は、CREATE FUNCTION ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

AS LOCATOR
関数が、このパラメーターのロケーターを受け取るように定義されることを示します。 AS LOCATOR を指定する場合は、データ・タイプは LOB または XML、あるいは LOB または XML に基づく特殊タイプでなければなりません。

SPECIFIC FUNCTION *specific-name*
関数を特定名によって識別します。 *specific-name* は、現行サーバーに存在する特定の関数を示している必要があります。

PROCEDURE または **SPECIFIC PROCEDURE**
特権が認可されるプロシージャーを指定します。このプロシージャー名 は、現行サーバーに存在しているプロシージャーを識別していなければなりません。

PROCEDURE *procedure-name*
プロシージャーを名前によって識別します。プロシージャー名 は、ただ 1

つのプロシージャーを識別していなければなりません。このプロシージャーには、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前のプロシージャーが複数ある場合、エラーが戻されます。

PROCEDURE *procedure-name (parameter-type, ...)*
プロシージャーを一意的に識別するプロシージャー・シグニチャーによって、プロシージャーを識別します。 *procedure-name (parameter-type,...)* では、指定されたプロシージャー・シグニチャーを持つプロシージャーを識別する必要があります。指定されたパラメーターは、プロシージャーの作成時に指定された、対応する位置にあるデータ・タイプと一致していなければなりません。認可するプロシージャー・インスタンスを識別する場合、データ・タイプの数とデータ・タイプの論理連結が使用されます。データ・タイプの同義語は、一致として扱われます。デフォルトがあるパラメーターは、このシグニチャーに含まれていなければなりません。

プロシージャー名 () を指定する場合、識別されるプロシージャーにパラメーターを使用することはできません。

procedure-name
プロシージャーの名前を識別します。

(parameter-type, ...)
プロシージャーのパラメーターを識別します。

非修飾の特殊タイプ名または配列タイプ名を指定する場合、データベース・マネージャーはその特殊タイプまたは配列タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、以下のいずれかを使用します。

- 空の括弧は、データベース・マネージャーがデータ・タイプの一致の判別に際して属性を無視することを示します。例えば、DEC() は、DEC(7,2) のデータ・タイプで定義されたプロシージャーのパラメーターに一致するものとみなされます。ただし、FLOAT に空の括弧を指定することはできません。これは、そのパラメーター値が特定のデータ・タイプ (REAL または DOUBLE) を示しているからです。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を指定する場合、その値は、 CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプが FLOAT の場合、突き合わせはデータ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度は指定された値に厳密に一致している必要はありません。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。暗黙の長さは、CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

FOR DATA 文節または CCSID 文節の指定はオプショナルです。いずれの文節も指定しないと、データ・タイプが一致するかどうかを判定する場合に、データベース・マネージャーが属性を無視することを示しま

す。どちらか一方の文節を指定する場合は、CREATE PROCEDURE ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

AS LOCATOR
プロシージャーが、このパラメーターのロケーターを受け取るように定義されることを示します。 AS LOCATOR を指定する場合は、データ・タイプは LOB または XML、あるいは LOB または XML に基づく特殊タイプでなければなりません。

SPECIFIC PROCEDURE *specific-name*
プロシージャーを特定名によって識別します。特定名 は、現行サーバーに存在している特定のプロシージャーを識別していなければなりません。

TO 特権を認可するユーザーを指定します。

USER
authorization-name がユーザー・プロファイルであることを指定します。USER が指定される場合、*authorization-name* はユーザー・プロファイルでなければなりません。

GROUP
authorization-name がグループ・プロファイルであることを指定します。GROUP が指定される場合、*authorization-name* はグループ・プロファイルでなければなりません。

authorization-name,...
1 つ以上の権限 ID をリストします。

PUBLIC
ユーザー (権限 ID) の集合に対して特権を認可します。詳しくは、 21 ページの『権限、特権、およびオブジェクト所有権』を参照してください。

WITH GRANT OPTION
指定した権限名 が、ON 文節で指定されている関数またはプロシージャーに対する特権を他のユーザーに認可できるようにします。

WITH GRANT OPTION の指定がない場合は、指定した権限名 は、ON 文節で指定されている関数またはプロシージャーに対する特権を別のユーザーに認可することができません。ただし、指定した権限名が、他の何らかの方法で認可できる権限を入手した場合 (例えば、*OBJMGT システム権限の認可) を除きます。

注

対応するシステム権限: SQL または外部関数か外部プロシージャーに認可された特権は、その関連のプログラム (*PGM) オブジェクトまたはサービス・プログラム (*SRVPGM) オブジェクトに認可されます。 Java 外部関数またはプロシージャーに認可された特権は、関連のクラス・ファイルまたは jar ファイルに対して認可されます。認可の実行時に関連プログラム、サービス・プログラム、クラス・ファイル、または jar ファイルが見つからない場合、エラーが戻されます。

GRANT および REVOKE ステートメントは、SQL オブジェクトに対するシステム権限の割り当ておよび除去を行います。次の表は、SQL 特権に対応するシステム権限を示しています。

表 93. 非 *Java* 関数またはプロシージャーに対して認可や取り消しが行われる特権

SQL の特権	関数またはプロシージャーに対する認可や取り消しに対応するシステム権限
ALL (ALL の認可または取り消しは、ステートメントの権限 ID が持つ特権のみを認可または取り消します。)	*OBJALTER *OBJOPR *EXECUTE *OBJMGT (取り消しのみ)
ALTER	*OBJALTER
EXECUTE	*EXECUTE *OBJOPR
WITH GRANT OPTION	*OBJMGT

表 94. *Java* 関数またはプロシージャーに対して認可や取り消しが行われる特権

SQL の特権	**Java** 関数またはプロシージャーに対する認可や取り消しに対応するデータ権限	**Java** 関数またはプロシージャーに対する認可や取り消しに対応するオブジェクト権限
ALL (ALL の認可または取り消しは、ステートメントの権限 ID が持つ特権のみを認可または取り消します。)	*RWX	*OBJEXIST *OBJALTER *OBJMGT (取り消しのみ)
ALTER	*R	*OBJALTER
EXECUTE	*RX	*EXECUTE
WITH GRANT OPTION	*RWX	*OBJMGT

関数またはプロシージャーへの権限を検査する際の対応するシステム権限**:** 次の表は、関数またはプロシージャーへの権限を検査する際の、SQL 特権に対応するシステム権限を示しています。左側の欄は、SQL 特権をリストしています。右側の欄は、同等のシステム権限をリストしています。

表 95. 非 *Java* 関数またはプロシージャーへの特権を検査する際の、対応するシステム権限

SQL の特権	対応するシステム権限
ALTER	*OBJALTER
EXECUTE	*EXECUTE および *OBJOPR

表 96. *Java* 関数またはプロシージャーへの特権を検査する際の、対応するシステム権限

SQL の特権	**Java** 関数またはプロシージャーへの特権を検査する際の、対応するデータ権限	**Java** 関数またはプロシージャーへの特権を検査する際の、対応するオブジェクト権限
ALTER	*R	*OBJALTER
EXECUTE	*RX	*EXECUTE

組み込み関数: 組み込み関数に関する特権を付与することはできません。

代替構文: 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- EXECUTE の同義語としてキーワード RUN を使用することができます。

例

プロシージャー PROCA に関する EXECUTE 特権を、PUBLIC に対して認可します。

```
GRANT EXECUTE
  ON PROCEDURE PROCA
  TO PUBLIC
```

GRANT (パッケージ特権)

この形式の GRANT ステートメントは、パッケージに対する特権を認可します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

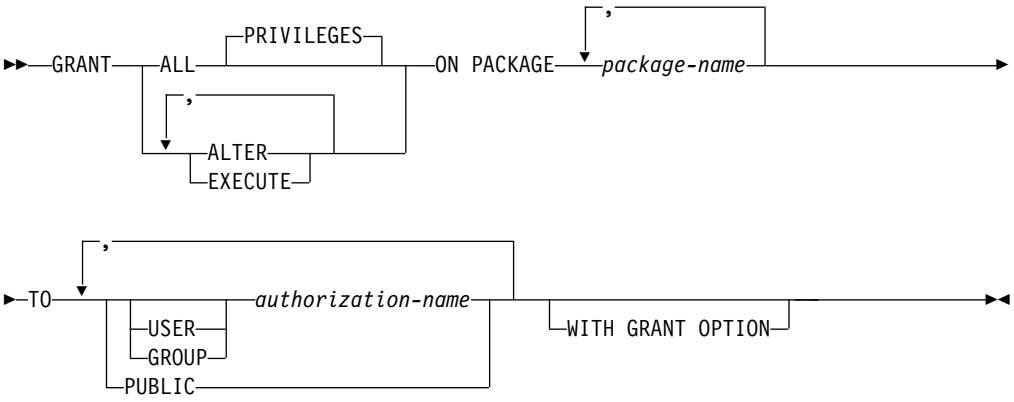
このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれのパッケージごとに、
  - このステートメントで指定されるすべての特権
  - パッケージに対する *OBJMGT システム権限
  - パッケージが含まれるスキーマに対する USAGE 特権
- データベース管理者権限
- セキュリティー管理者権限

WITH GRANT OPTION を指定する場合、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- そのパッケージの所有権
- データベース管理者権限
- セキュリティー管理者権限

構文



説明

ALL または ALL PRIVILEGES

1 つ以上の特権を認可します。認可される特権は、指定されたパッケージに対してステートメントの権限 ID が持っている認可可能な特権のすべてです。パッケージに対する ALL PRIVILEGES を認可することは、 *ALL システム権限を認可するのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つ以上を使用する必要があります。各キーワードは、そこで説明している特権を認可します。

ALTER
COMMENT ステートメントおよび LABEL ステートメントを使用するための特権を認可します。

EXECUTE
パッケージのステートメントを実行する特権を認可します。

ON PACKAGE *package-name*
特権を認可する対象のパッケージを識別します。このパッケージ名 は、現行サーバーに存在しているパッケージを識別していなければなりません。

TO 特権を認可するユーザーを指定します。

USER
authorization-name がユーザー・プロファイルであることを指定します。USER が指定される場合、*authorization-name* はユーザー・プロファイルでなければなりません。

GROUP
authorization-name がグループ・プロファイルであることを指定します。GROUP が指定される場合、*authorization-name* はグループ・プロファイルでなければなりません。

authorization-name,...
1 つ以上の権限 ID をリストします。

PUBLIC
ユーザー (権限 ID) の集合に対して特権を認可します。詳しくは、 21 ページの『権限、特権、およびオブジェクト所有権』を参照してください。

WITH GRANT OPTION
指定した権限名 が、ON 文節で指定されているパッケージに対する特権を他のユーザーに認可できるようにします。

WITH GRANT OPTION の指定がない場合は、指定した権限名 は、 ON 文節で指定されているパッケージに対する特権を別のユーザーに認可することができません。ただし、指定した権限名が、他の何らかの方法で認可できる権限を入手した場合 (例えば、*OBJMGT システム権限の認可) を除きます。

注

対応するシステム権限**:** GRANT および REVOKE ステートメントは、SQL オブジェクトに対するシステム権限の割り当ておよび除去を行います。次の表は、SQL 特権に対応するシステム権限を示しています。

表 97. パッケージについて、認可または取り消しの対象となる特権

SQL の特権	パッケージに対する認可または取り消しに対応するシステム権限
ALL (ALL の認可または取り消しは、ステートメントの権限 ID が持つ特権のみを認可または取り消します。)	*OBJALTER *OBJOPR *EXECUTE *OBJMGT (取り消しのみ)
ALTER	*OBJALTER
EXECUTE	*EXECUTE *OBJOPR
WITH GRANT OPTION	*OBJMGT

パッケージへの権限を検査する際の対応するシステム権限**:** 次の表は、パッケージへの権限を検査する際の、SQL 特権に対応するシステム権限を示しています。左側の欄は、SQL 特権をリストしています。右側の欄は、同等のシステム権限をリストしています。

表 98. パッケージへの特権を検査する際の、対応するシステム権限

SQL の特権	パッケージへの特権を検査する際の、対応するシステム権限
ALTER	*OBJALTER
EXECUTE	*EXECUTE および *OBJOPR

代替構文**:** 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- EXECUTE の同義語としてキーワード RUN を使用することができます。
- PACKAGE の同義語として、キーワード PROGRAM を使用することができます。

例

パッケージ PKGA に関する EXECUTE 特権を PUBLIC に対して認可します。

```
GRANT EXECUTE
  ON PACKAGE PKGA
  TO PUBLIC
```

GRANT (スキーマ特権)

この形式の GRANT ステートメントは、スキーマに対する特権を付与します。

呼び出し

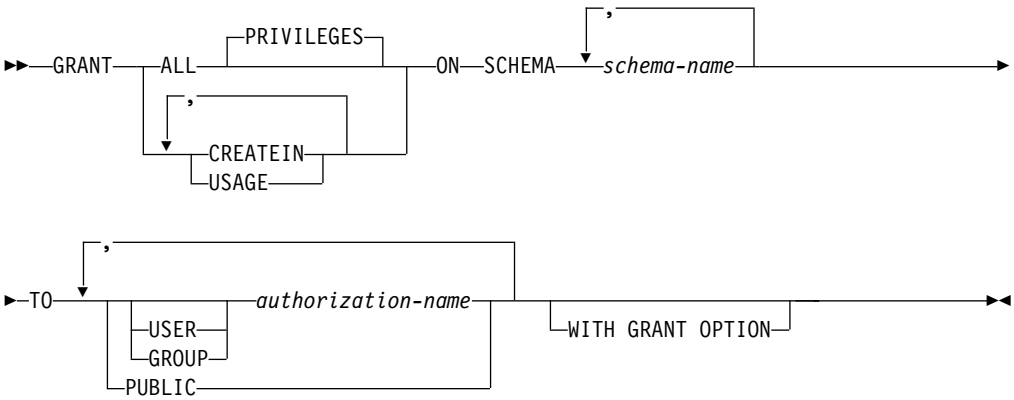
このステートメントは、アプリケーション・プログラムに組み込むことも、あるいは対話式に実行することもできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれのスキーマごとに、
  - このステートメントで指定されるすべての特権
  - スキーマに対する *OBJMGT のシステム権限
- データベース管理者権限
- セキュリティー管理者権限

構文



説明

ALL または ALL PRIVILEGES

1 つ以上の特権を認可します。認可される特権は、指定されたスキーマに対してステートメントの権限 ID が持っている認可可能な特権のすべてです。スキーマに対する ALL PRIVILEGES を認可することは、*ALL システム権限を認可するのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つ以上を使用する必要があります。各キーワードは、そこで説明している特権を認可します。

CREATEIN

スキーマにオブジェクトを作成する特権を与えます。 オブジェクトの作成に必要なその他の権限または特権は、これを指定しても必要です。

USAGE
スキーマを使用する特権を付与します。スキーマに存在するオブジェクトの参照には、USAGE 特権が必要です。

ON SCHEMA *schema-name*
付与する指定した特権の対象となるスキーマを指定します。

TO 特権を認可するユーザーを指定します。

USER
authorization-name がユーザー・プロファイルであることを指定します。USER が指定される場合、*authorization-name* はユーザー・プロファイルでなければなりません。

GROUP
authorization-name がグループ・プロファイルであることを指定します。GROUP が指定される場合、*authorization-name* はグループ・プロファイルでなければなりません。

authorization-name,...
1 つ以上の権限 ID をリストします。

PUBLIC
ユーザー (権限 ID) の集合に対して特権を認可します。詳しくは、 21 ページの『権限、特権、およびオブジェクト所有権』を参照してください。

WITH GRANT OPTION
指定した *authorization-name* が、ON 文節で指定されているスキーマに対する特権を他のユーザーに認可できるようにします。

注

対応するシステム権限**:** GRANT および REVOKE ステートメントは、SQL オブジェクトに対するシステム権限の割り当ておよび除去を行います。次の表は、スキーマに対して認可される SQL 特権に対応するシステム権限を示しています。左側の欄は、SQL 特権をリストしています。右側の欄は、認可または取り消しされる同等のシステム権限をリストしています。

表 *99.* スキーマに対して認可または取り消しされる特権

SQL の特権	スキーマに対する認可または取り消しに対応するシステム権限
ALL (ALL の認可または取り消しは、ステートメントの権限 ID が持つ特権のみを認可または取り消します。)	*OBJMGT *OBJOPR *READ *EXECUTE *ADD
CREATEIN	*OBJOPR *READ *EXECUTE *ADD[109]
USAGE	*OBJOPR *READ *EXECUTE

表 *99.* スキーマに対して認可または取り消しされる特権 (続き)

SQL の特権	スキーマに対する認可または取り消しに対応するシステム権限
WITH GRANT OPTION	*OBJMGT

スキーマへの権限を検査する際の対応するシステム権限**:** 次の表は、スキーマへの権限を検査する際の、SQL 特権に対応するシステム権限を示しています。左側の欄は、SQL 特権をリストしています。右側の欄は、同等のシステム権限をリストしています。

表 *100.* スキーマへの特権を検査する際の、対応するシステム権限

SQL の特権	対応するシステム権限
CREATEIN	*OBJOPR、*READ、*EXECUTE、および *ADD[110]
USAGE	*EXECUTE[110]

GRANT の規則**:** GRANT ステートメントで付与できるのは、ステートメントの権限 ID で付与できる特権に限られます。特権を付与できなかった場合は、エラーが返されます。

例

例 1: スキーマ T_SCORES に対する CREATEIN 特権をユーザー JONES に付与します。

```
GRANT CREATEIN
  ON SCHEMA T_SCORES
  TO JONES;
```

109. CREATEIN が取り消されると、*ADD のみが取り消されます。

110. *OBJOPR と *READ は特定のステートメント (例えば、CREATE など) の場合のみ検査されます。

GRANT (シーケンス特権)

この形式の GRANT ステートメントは、シーケンスに対する特権を認可します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

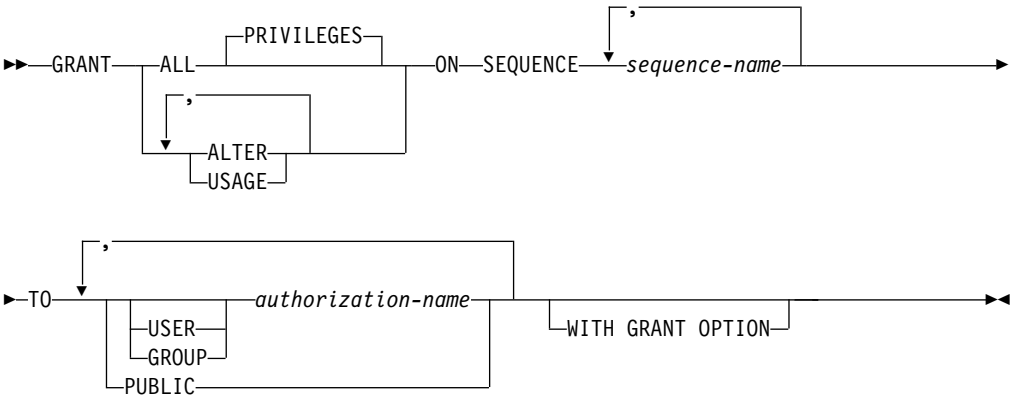
このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれのシーケンスごとに、
  - このステートメントで指定されるすべての特権
  - シーケンスに対する *OBJMGT システム権限
  - シーケンスが含まれるスキーマに対する USAGE 特権
- データベース管理者権限
- セキュリティー管理者権限

WITH GRANT OPTION を指定する場合、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- そのシーケンスの所有権
- データベース管理者権限
- セキュリティー管理者権限

構文



説明

ALL または ALL PRIVILEGES

1 つ以上の特権を認可します。認可される特権は、指定されたシーケンスに対してステートメントの権限 ID が持っている認可可能な特権のすべてです。シーケンスに対する ALL PRIVILEGES を認可することは、*ALL システム権限を認可するのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つ以上を使用する必要があります。各キーワードは、そこで説明している特権を認可します。

ALTER
シーケンスに対する ALTER SEQUENCE、COMMENT、および LABEL ステートメントを使用する特権を許可します。

USAGE
NEXT VALUE または PREVIOUS VALUE 式内のシーケンスを使用するための特権を認可します。

ON SEQUENCE *sequence-name*
特権が認可されるシーケンスを指定します。シーケンス名 は、現行サーバーに存在するシーケンスを示すものでなければなりません。

TO 特権を認可するユーザーを指定します。

USER
authorization-name がユーザー・プロファイルであることを指定します。USER が指定される場合、*authorization-name* はユーザー・プロファイルでなければなりません。

GROUP
authorization-name がグループ・プロファイルであることを指定します。GROUP が指定される場合、*authorization-name* はグループ・プロファイルでなければなりません。

authorization-name,...
1 つ以上の権限 ID をリストします。

PUBLIC
ユーザー (権限 ID) の集合に対して特権を認可します。詳しくは、 21 ページの『権限、特権、およびオブジェクト所有権』を参照してください。

WITH GRANT OPTION
指定した権限名 が、ON 文節で指定されているシーケンスに対する特権を他のユーザーに認可できるようにします。

WITH GRANT OPTION の指定がない場合は、指定した権限名 は、 USAGE 特権を別のユーザーに認可することができません。ただし、指定した権限名が、他の何らかの方法で認可できる権限を入手した場合 (例えば、*OBJMGT システム権限の認可) を除きます。

注

対応するシステム権限**:** GRANT および REVOKE ステートメントは、SQL オブジェクトに対するシステム権限の割り当ておよび除去を行います。次の表は、SQL 特権に対応するシステム権限を示しています。

表 *101.* シーケンスに対して認可または取り消しされる特権

SQL の特権	シーケンスに対する認可または取り消しに対応するシステム権限
ALL (ALL の認可または取り消しは、ステートメントの権限 ID が持つ特権のみを認可または取り消します。)	*OBJALTER *OBJOPR *EXECUTE *READ *ADD *DLT *UPD *OBJMGT (取り消しのみ)
ALTER	*OBJALTER
USAGE	*OBJOPR *EXECUTE *READ *ADD *DLT *UPD
WITH GRANT OPTION	*OBJMGT

シーケンスへの権限を検査する際の対応するシステム権限**:** 次の表は、シーケンスへの権限を検査する際の、SQL 特権に対応するシステム権限を示しています。左側の欄は、SQL 特権をリストしています。右側の欄は、同等のシステム権限をリストしています。

表 *102.* シーケンスへの特権を検査する際の、対応するシステム権限

SQL の特権	対応するシステム権限
ALTER	*OBJALTER
USAGE	*OBJOPR および *EXECUTE および *READ および *ADD および *DLT および *UPD

例

任意のユーザーに ORG_SEQ と呼ばれるシーケンスに対する USAGE 特権を認可します。

```
GRANT USAGE
  ON SEQUENCE ORG_SEQ
  TO PUBLIC
```

GRANT (表またはビューの特権)

この形式の GRANT ステートメントは、表またはビューに対する特権を認可します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

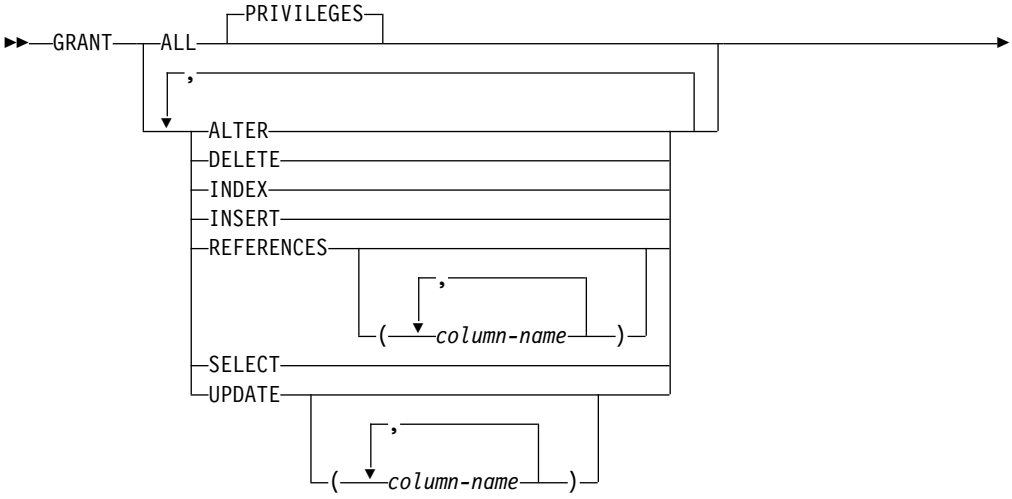
このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

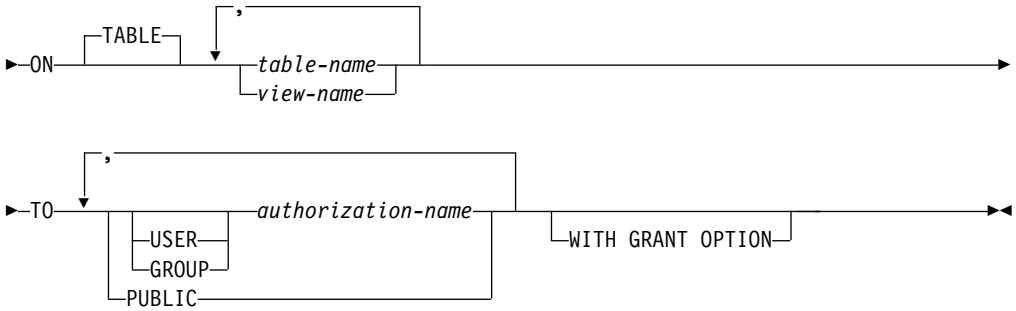
- ステートメント内で識別された、それぞれの表またはビューごとに、
  - このステートメントで指定されるすべての特権
  - その表またはビューに対する *OBJMGT システム権限
  - 表またはビューが含まれるスキーマに対する USAGE 特権
- データベース管理者権限
- セキュリティー管理者権限

WITH GRANT OPTION を指定する場合、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- その表の所有権
- データベース管理者権限
- セキュリティー管理者権限

構文





説明

ALL または ALL PRIVILEGES

1 つ以上の特権を認可します。認可される特権は、指定された表またはビューに対してステートメントの権限 ID が持っている認可可能な特権のすべてです。

表またはビューに対する ALL PRIVILEGES を認可することは、 *ALL システム権限を認可するのと同じではないことに注意する必要があります。

ALTER

指定の表を変更する特権、または指定の表でトリガーを作成または除去する特権を認可します。表およびビューに対する COMMENT および LABEL ステートメントを使用する特権を認可します。

DELETE

指定の表またはビューから行を削除する特権を認可します。ビューを指定する場合は、削除可能なビューでなければなりません。

INDEX

指定の表に索引を作成する特権を認可します。この特権は、ビューに対して認可することはできません。

INSERT

指定の表またはビューに行を挿入する特権を認可します。ビューを指定する場合は、挿入可能なビューでなければなりません。

REFERENCES

指定する表が親である場合に、参照制約を追加する特権を認可します。列のリストが指定されていない場合、または ALL PRIVILEGES を指定することによって REFERENCES が表またはビューのすべての列に対して認可されている場合、被認可者は、親キーとして ON 文節内に指定されている各表のすべての列を使用して、参照制約を追加することができます。これは、ALTER TABLE ステートメントによって後から追加された列も対象となります。この特権は、ビューの場合も認可できますが、ビューに対してはこの特権は使用されません。

REFERENCES (*column-name*,...)

指定する表が親である場合に、親キーとして列リストに指定されている列のみを使用して、参照制約を追加する特権を認可します。それぞれの列名 は、ON 文節に指定されている各表の列を識別する非修飾の名前でなければなりません。この特権は、ビューの列の場合も認可できますが、ビューについてはこの特権は使用されません。

SELECT
指定の表またはビューでビューを作成する特権またはデータを読み取る特権を認可します。例えば、表またはビューが照会に指定されている場合、SELECT 特権が必要です。

UPDATE
指定の表またはビューで行を更新する特権を認可します。列のリストが指定されていない場合、または ALL PRIVILEGES を指定することによって、 UPDATE が表またはビューのすべての列に対して認可されている場合、被認可者は、ON 文節に指定されている各表のすべての更新可能列を更新することができます。これは ALTER TABLE ステートメントによって後から追加された列も対象となります。ビューを指定する場合は、更新可能なビューでなければなりません。

UPDATE (*column-name***,...)**
列リストに示されている列のみを更新するために、UPDATE ステートメントを使用する特権を認可します。それぞれの列名は、ON 文節に指定されている各表およびビューの列を識別する非修飾の名前でなければなりません。ビューを指定する場合は、更新可能なビューでなければなりません。さらに、指定する列は更新可能な列でなければなりません。

ON *table-name* または *view-name,...*
特権を認可する表またはビューを識別します。表名 またはビュー名 は、現行サーバーにある表またはビューを示すものでなければなりませんが、宣言済み一時表を示すものであってはなりません。

TO 特権を認可するユーザーを指定します。

USER
authorization-name がユーザー・プロファイルであることを指定します。USER が指定される場合、*authorization-name* はユーザー・プロファイルでなければなりません。

GROUP
authorization-name がグループ・プロファイルであることを指定します。GROUP が指定される場合、*authorization-name* はグループ・プロファイルでなければなりません。

authorization-name,...
1 つ以上の権限 ID をリストします。

PUBLIC
ユーザー (権限 ID) の集合に対して特権を認可します。詳しくは、 21 ページの『権限、特権、およびオブジェクト所有権』を参照してください。

WITH GRANT OPTION
指定した権限名 が、ON 文節で指定されている表およびビューに対する特権を他のユーザーに認可できるようにします。

WITH GRANT OPTION の指定がない場合は、指定した権限名 は、ON 文節で指定されている表およびビューに対する特権を別のユーザーに認可することができません。ただし、指定した権限名が、他の何らかの方法で認可できる権限を入手した場合 (例えば、*OBJMGT システム権限の認可) を除きます。

注

対応するシステム権限: GRANT および REVOKE ステートメントは、SQL オブジェクトに対するシステム権限の割り当ておよび除去を行います。次の表は、表に対して認可される SQL 特権に対応するシステム権限を示しています。左側の欄は、SQL 特権をリストしています。右側の欄は、認可または取り消しされる同等のシステム権限をリストしています。

表 *103.* 表に対して認可または取り消しされる特権

SQL の特権	表に対する認可または取り消しに対応するシステム権限
ALL (ALL の GRANT または取り消しは、ステートメントの権限 ID が持つ特権のみを認可または取り消します。)	*OBJALTER [111] *OBJMGT (取り消しのみ) *OBJOPR *OBJREF *ADD *DLT *READ *UPD
ALTER	*OBJALTER [112]
DELETE	*OBJOPR [113] *DLT
INDEX	*OBJALTER [112]
INSERT	*OBJOPR[113] *ADD
REFERENCES	*OBJREF[112]
SELECT	*OBJOPR[113] *READ
UPDATE	*OBJOPR[113] *UPD
WITH GRANT OPTION	*OBJMGT

次の表は、ビューに対して認可される SQL 特権に対応するシステム権限を示しています。左側の欄は、SQL 特権をリストしています。中央の欄は、ビュー自体に対して認可または取り消しされる同等のシステム権限をリストしています。例えば、右側の欄にリストされているシステム権限は、ビューの定義内で参照さているすべての表およびビューに対して認可され、ビューが参照されている場合は、その定義で参照されているすべての表およびビューのすべてに対して認可されます。[114]

111. SQL の INDEX および ALTER 特権は、同じシステム権限 *OBJALTER に対応しています。 INDEX と ALTER の両方を認可しても、ユーザーに与えられる権限が増加するわけではありません。

112. WITH GRANT OPTION が与えられたユーザーは、ALTER および REFERENCES 権限によって与えられる機能も実行することができます。

113. *OBJOPR が取り消されるのは、指定の権限 ID または PUBLIC で *OBJOPR 以外の最後のシステム特権も取り消される場合に限られます。

114. 指定した権限が、ビューの定義で参照されている表およびビューに認可されるのは、権限の認可対象のユーザーが別の権限ソースからそのような権限 (例えば共通認可の権限) を入手していない場合だけに限られます。

ビューが複数の表やビューを参照している場合、*DLT、*ADD、および *UPD システム権限は、そのビューの定義の全選択の最初の表、またはビューについてのみ認可されます。 *READ システム権限は、そのビューの定義で参照されているすべての表およびビューに関して認可されます。

ある SQL 特権に対応して、複数のシステム権限が認可される場合に、それらの権限のいずれか 1 つを認可することができないと、警告が出され、その特権に対応する権限はいずれも認可されません。GRANT とは異なり、REVOKE はビューに関するシステム権限を取り消すだけです。参照される表やビューからシステム権限が取り消されることはありません。

表 *104.* ビューに対して認可または取り消しされる特権

SQL の特権	ビューに対する認可または取り消しに対応するシステム権限	参照された表およびビューに対する認可または取り消しに対応するシステム権限
ALL (ALL の GRANT または取り消しは、ステートメントの権限 ID が持つ特権のみを認可または取り消します。)	*OBJALTER *OBJMGT (取り消しのみ) *OBJOPR *OBJREF *ADD *DLT *READ *UPD	*ADD *DLT *READ *UPD
ALTER	*OBJALTER [112]	なし
DELETE	*OBJOPR[113] *DLT	*DLT
INDEX	該当しない	該当しない
INSERT	*OBJOPR[113] *ADD	*ADD
REFERENCES	*OBJREF[112]	なし
SELECT	*OBJOPR[113] *READ	*READ
UPDATE	*OBJOPR[113] *UPD	*UPD
WITH GRANT OPTION	*OBJMGT	なし

表またはビューへの権限を検査する際の対応するシステム権限**:** 次の表は、表への権限を検査する際の、SQL 特権に対応するシステム権限を示しています。左側の欄は、SQL 特権をリストしています。右側の欄は、同等のシステム権限をリストしています。

表 *105.* 表への特権を検査する際の、対応するシステム権限

SQL の特権	表への特権を検査する際の、対応するシステム権限
ALTER	*OBJALTER または *OBJMGT
DELETE	*OBJOPR および *DLT
INDEX	*OBJALTER または *OBJMGT

表 *105.* 表への特権を検査する際の、対応するシステム権限 (続き)

SQL の特権	表への特権を検査する際の、対応するシステム権限
INSERT	*OBJOPR および *ADD
REFERENCES	*OBJREF または *OBJMGT
SELECT	*OBJOPR および *READ
UPDATE	*OBJOPR および *UPD

次の表は、ビューに対する特権を検査する際の SQL 特権に対応するシステム権限を示しています。左側の欄は、SQL 特権をリストしています。中央の欄は、ビュー自体に対して検査される同等のシステム権限をリストしています。例えば、右側の欄にリストされているシステム権限は、ビューの定義内で参照さているすべての表およびビューで検査され、ビューが参照されている場合は、その定義で参照されているすべての表およびビューで検査されます。

表 *106.* ビューへの特権を検査する際の、対応するシステム権限

SQL の特権	ビューに対する対応するシステム権限	参照される表およびビューに対する対応するシステム権限
ALTER	*OBJALTER および *OBJMGT	なし
DELETE[115]	*OBJOPR および *DLT	*DLT
INDEX	該当しない	該当しない
INSERT[116]	*OBJOPR および *ADD	*ADD
REFERENCES	*OBJREF または *OBJMGT	なし
SELECT	*OBJOPR および *READ	*READ
UPDATE[117]	*OBJOPR および *UPD	*UPD

GRANT の規則: GRANT ステートメントで付与できるのは、ステートメントの権限 ID で付与できる特権に限られます。特権を付与できなかった場合は、エラーが返されます。

例

例 *1:* 表 WESTERN_CR に対するすべての特権を PUBLIC に認可します。

```
GRANT ALL PRIVILEGES ON WESTERN_CR
  TO PUBLIC
```

115. ビューが作成される際に、その所有者は、必ずしもそのビューに対する DELETE 特権を獲得するとは限りません。所有者が DELETE 特権を獲得するのは、そのビューが削除を許されており、しかも所有者が副選択で参照されている最初の表に対しても DELETE 特権を持っている場合だけです。

116. ビューが作成される際に、その所有者は、必ずしもそのビューに対する INSERT 特権を獲得するとは限りません。所有者が INSERT 特権を獲得するのは、そのビューが挿入を許されており、しかも所有者が副選択で参照されている最初の表に対しても INSERT 特権を持っている場合だけです。

117. ビューが作成される際に、その所有者は、必ずしもそのビューに対する UPDATE 特権を獲得するとは限りません。所有者が UPDATE 特権を獲得するのは、そのビューが更新を許可し、しかも所有者が副選択で参照されている最初の表に対して UPDATE 特権を保有している場合だけに限られます。

例 *2:* 表 CALENDAR に関する適切な特権を認可して、 PHIL および CLAIRE が表 CALENDAR を読み取って、新しい項目を挿入できるようにします。 PHIL および CLAIRE には、既存の項目の変更や削除は許しません。

```
GRANT SELECT, INSERT ON CALENDAR
  TO PHIL, CLAIRE
```

例 *3:* TABLE1 および VIEW1 関する列特権を FRED に認可します。この GRANT ステートメントの中に指定されている列が両方とも、TABLE1 と VIEW1 のどちらにもなければなりません。

```
GRANT UPDATE(column_1, column_2)
  ON  TABLE1, VIEW1
  TO FRED WITH GRANT OPTION
```

GRANT (タイプ特権)

この形式の GRANT ステートメントは、タイプに対する特権を認可します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

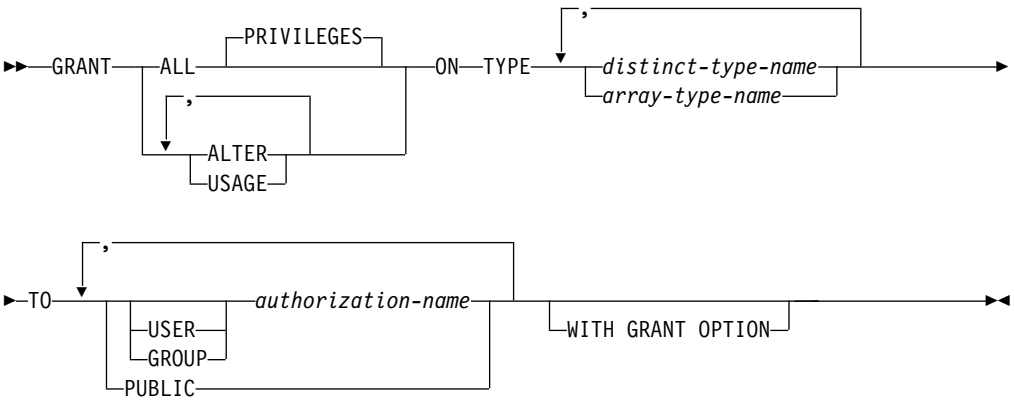
このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれの特殊タイプまたは配列タイプごとに、
  - このステートメントで指定されるすべての特権
  - タイプに対する *OBJMGT のシステム権限
  - タイプが含まれるスキーマに対する USAGE 特権
- データベース管理者権限
- セキュリティー管理者権限

WITH GRANT OPTION を指定する場合、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- タイプの所有権
- データベース管理者権限
- セキュリティー管理者権限

構文



説明

ALL または **ALL PRIVILEGES**
1 つ以上の特権を認可します。認可される特権は、指定した特殊タイプまたは配列タイプに対してステートメントの権限 ID が持っているすべての認可可能な

特権です。タイプに対する ALL PRIVILEGES を認可することは、*ALL システム権限を認可するのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つ以上を使用する必要があります。各キーワードは、そこで説明している特権を認可します。

ALTER
COMMENT および LABEL ステートメントを使用するための特権を認可します。

USAGE
表、関数、プロシージャー、CAST 式の中でタイプを使用するための特権を付与します。

ON TYPE *distinct-type-name*または*array-type-name*
特権が認可される特殊タイプまたは配列タイプを指定します。特殊タイプ名 または配列タイプ名 は、現行サーバーに存在するユーザー定義タイプを識別する必要があります。

TO 特権を認可するユーザーを指定します。

USER
authorization-name がユーザー・プロファイルであることを指定します。USER が指定される場合、*authorization-name* はユーザー・プロファイルでなければなりません。

GROUP
authorization-name がグループ・プロファイルであることを指定します。GROUP が指定される場合、*authorization-name* はグループ・プロファイルでなければなりません。

authorization-name,...
1 つ以上の権限 ID をリストします。

PUBLIC
ユーザー (権限 ID) の集合に対して特権を認可します。詳しくは、 21 ページの『権限、特権、およびオブジェクト所有権』を参照してください。

WITH GRANT OPTION
指定した権限名 が、ON 文節で指定されているタイプに対する特権を他のユーザーに認可できるようにします。

WITH GRANT OPTION の指定がない場合は、指定した権限名 は、 USAGE 特権を別のユーザーに認可することができません。ただし、指定した権限名が、他の何らかの方法で認可できる権限を入手した場合 (例えば、*OBJMGT システム権限の認可) を除きます。

注

対応するシステム権限**:** GRANT および REVOKE ステートメントは、SQL オブジェクトに対するシステム権限の割り当ておよび除去を行います。次の表は、SQL 特権に対応するシステム権限を示しています。

表 107. ユーザー定義タイプに対して認可または取り消しされる特権

SQL の特権	ユーザー定義タイプに対する認可または取り消しに対応するシステム権限
ALL (ALL の認可または取り消しは、ステートメントの権限 ID が持つ特権のみを認可または取り消します。)	*OBJALTER *OBJOPR *EXECUTE *OBJMGT (取り消しのみ)
ALTER	*OBJALTER
USAGE	*EXECUTE *OBJOPR
WITH GRANT OPTION	*OBJMGT

ユーザー定義タイプへの権限を検査する際の対応するシステム権限**:** 次の表は、タイプへの権限を検査する際の、SQL 特権に対応するシステム権限を示しています。左側の欄は、SQL 特権をリストしています。右側の欄は、同等のシステム権限をリストしています。

表 108. ユーザー定義タイプへの権限を検査する際の対応するシステム権限

SQL の特権	ユーザー定義タイプに対する認可または取り消しに対応するシステム権限
ALTER	*OBJALTER
USAGE	*EXECUTE および *OBJOPR

USAGE 特権が必要な場合**:** SQL ステートメント (例えば、CAST 仕様を含むステートメントや CREATE TABLE ステートメント) でタイプが明示的に参照されている場合、USAGE 特権が必要です。タイプが間接的に参照される場合、USAGE 特権は必要ありません。例えば、ビューが特殊データ・タイプを有する表の列を参照する場合などです。

代替構文**:** 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード DATA TYPE または DISTINCT TYPE を TYPE の同義語として使用することができます。

例

特殊タイプ SHOE_SIZE に関する USAGE 特権をユーザー JONES に認可します。この GRANT ステートメントでは、JONES に、特殊タイプ SHOE_SIZE に関連するキャスト関数を実行する特権は与えません。

```
GRANT USAGE
  ON DISTINCT TYPE SHOE_SIZE
  TO JONES
```

GRANT (変数特権)

この形式の GRANT ステートメントは、グローバル変数に対する特権を認可します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、あるいは対話式に実行することもできます。これは、動的に準備できる実行可能ステートメントです。

権限

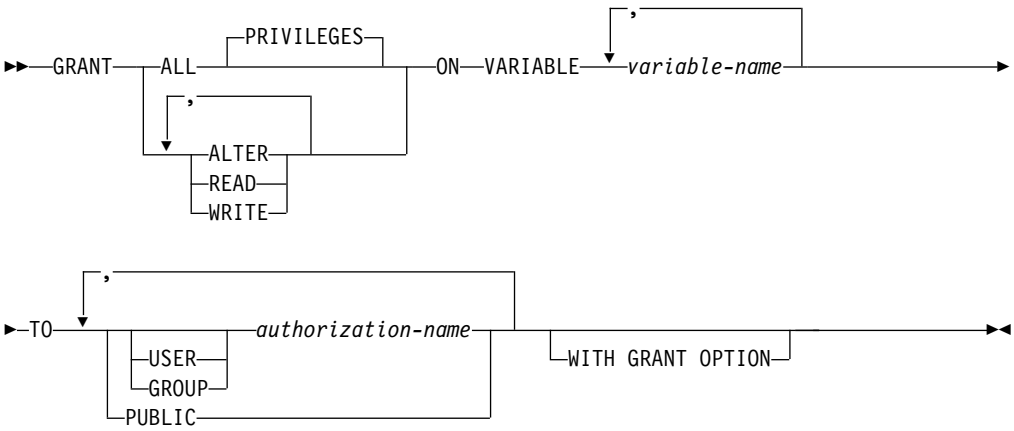
このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれのグローバル変数ごとに、
  - このステートメントで指定されるすべての特権
  - グローバル変数に対する *OBJMGT システム権限
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限
- セキュリティー管理者権限

WITH GRANT OPTION を指定する場合、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- グローバル変数の所有権
- データベース管理者権限
- セキュリティー管理者権限

構文



説明

ALL または **ALL PRIVILEGES**

1 つ以上の特権を認可します。認可される特権は、指定されたグローバル変数に対してステートメントの権限 ID が持っている認可可能な特権のすべてです。

グローバル変数に対する ALL PRIVILEGES を認可することは、*ALL システム権限を認可するのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つ以上を使用する必要があります。各キーワードは、そこで説明している特権を認可します。

ALTER
グローバル変数に対する COMMENT および LABEL ステートメントを使用する特権を認可します。

READ
グローバル変数の値を読み取る特権を付与します。

WRITE
グローバル変数に値を割り当てる特権を付与します。

ON VARIABLE *variable-name*
特権が認可されるグローバル変数を指定します。この変数名 は、現行サーバーに存在しているグローバル変数を識別していなければなりません。

TO 特権を認可するユーザーを指定します。

USER
authorization-name がユーザー・プロファイルであることを指定します。USER が指定される場合、*authorization-name* はユーザー・プロファイルでなければなりません。

GROUP
authorization-name がグループ・プロファイルであることを指定します。GROUP が指定される場合、*authorization-name* はグループ・プロファイルでなければなりません。

authorization-name,...
1 つ以上の権限 ID をリストします。

PUBLIC
ユーザー (権限 ID) の集合に対して特権を認可します。詳しくは、 21 ページの『権限、特権、およびオブジェクト所有権』を参照してください。

WITH GRANT OPTION
指定した権限名 が、ON 文節で指定されているグローバル変数に対する特権を他のユーザーに認可できるようにします。

注

対応するシステム権限**:** GRANT および REVOKE ステートメントは、SQL オブジェクトに対するシステム権限の割り当ておよび除去を行います。次の表は、SQL 特権に対応するシステム権限を示しています。

表 *109.* グローバル変数に対して認可または取り消しされる特権

SQL の特権	グローバル変数に対する認可または取り消しに対応するシステム権限
ALL (ALL の認可または取り消しは、ステートメントの権限 ID が持つ特権のみを認可または取り消します。)	*OBJALTER *OBJOPR *EXECUTE *UPD *READ *OBJMGT (取り消しのみ)
ALTER	*OBJALTER
READ	*OBJOPR *EXECUTE *READ
WRITE	*OBJOPR *EXECUTE *UPD
WITH GRANT OPTION	*OBJMGT

グローバル変数への権限を検査する際の対応するシステム権限**:** 次の表は、グローバル変数への権限を検査する際の、SQL 特権に対応するシステム権限を示しています。左側の欄は、SQL 特権をリストしています。右側の欄は、同等のシステム権限をリストしています。

表 *110.* グローバル変数への権限を検査する際の対応するシステム権限

SQL の特権	対応するシステム権限
ALTER	*OBJALTER
READ	*OBJOPR および *EXECUTE および *READ
WRITE	*OBJOPR および *EXECUTE および *UPD

例

グローバル変数 MYSCHEMA.MYJOB_PRINTER に関する READ 特権と WRITE 特権をユーザー ZUBIRI に付与します。

```
GRANT READ, WRITE
  ON VARIABLE MYSCHEMA.MYJOB_PRINTEER
  TO ZUBIRI
```

GRANT (XML スキーマ特権)

この形式の GRANT ステートメントは、XSR オブジェクトに対する特権を認可します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、あるいは対話式に実行することもできます。これは、動的に準備できる実行可能ステートメントです。

権限

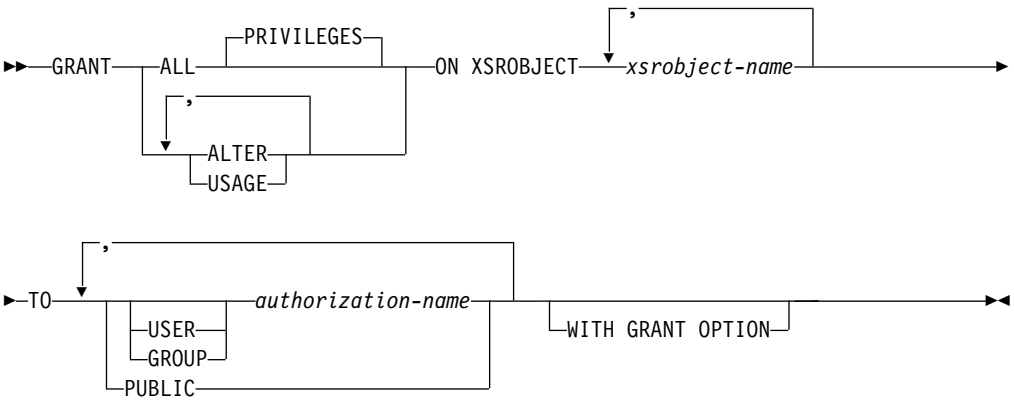
このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれの XSR オブジェクトごとに、
  - このステートメントで指定されるすべての特権
  - XSR オブジェクトに対する *OBJMGT システム権限
  - XSR オブジェクトが含まれるスキーマに対する USAGE 特権
- データベース管理者権限
- セキュリティー管理者権限

WITH GRANT OPTION を指定する場合、ステートメントの権限 ID によって保持される特権には、少なくとも次の 1 つが含まれていなければなりません。

- XSR オブジェクトの所有権
- データベース管理者権限
- セキュリティー管理者権限

構文



説明

ALL または ALL PRIVILEGES

1 つ以上の特権を認可します。認可される特権は、指定された XSR オブジェクトに対してステートメントの権限 ID が持っている認可可能な特権のすべてで

す。XSR オブジェクトに対する ALL PRIVILEGES を認可することは、*ALL システム権限を認可するのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つ以上を使用する必要があります。各キーワードは、そこで説明している特権を認可します。

ALTER
COMMENT ステートメントおよび LABEL ステートメントを使用するための特権を認可します。

USAGE
検証または分解のために XSR オブジェクトを使用する特権を付与します。

ON XSROBJECT *xsrobject-name*
特権が認可される XSR オブジェクトを指定します。この*XSR* オブジェクト名は、現行サーバーに存在しているXSR オブジェクトを示すものでなければなりません。

TO 特権を認可するユーザーを指定します。

USER
authorization-name がユーザー・プロファイルであることを指定します。USER が指定される場合、*authorization-name* はユーザー・プロファイルでなければなりません。

GROUP
authorization-name がグループ・プロファイルであることを指定します。GROUP が指定される場合、*authorization-name* はグループ・プロファイルでなければなりません。

authorization-name,...
1 つ以上の権限 ID をリストします。

PUBLIC
ユーザー (権限 ID) の集合に対して特権を認可します。詳しくは、 21 ページの『権限、特権、およびオブジェクト所有権』を参照してください。

WITH GRANT OPTION
指定した権限名 が、ON 文節で指定されている XSR オブジェクトに対する特権を他のユーザーに認可できるようにします。

WITH GRANT OPTION の指定がない場合は、指定した権限名 は、ON 文節で指定されている XSR オブジェクトに対する特権を別のユーザーに認可することができません。ただし、指定した権限名が、他の何らかの方法で認可できる権限を入手した場合 (例えば、*OBJMGT システム権限の認可) を除きます。

注

GRANT および REVOKE ステートメントは、SQL オブジェクトに対するシステム権限の割り当ておよび除去を行います。次の表は、SQL 特権に対応するシステム権限を示しています。

表 *111. XSR* オブジェクトに対して認可または取り消しされる特権

SQL の特権	**XSR** オブジェクトに対する認可または取り消しに対応するシステム権限
ALL (ALL の認可または取り消しは、ステートメントの権限 ID が持つ特権のみを認可または取り消します。)	*OBJALTER *OBJOPR *READ *EXECUTE *OBJMGT (取り消しのみ)
ALTER	*OBJALTER
USAGE	*OBJOPR *EXECUTE *READ
WITH GRANT OPTION	*OBJMGT

XSR オブジェクトへの権限を検査する際の対応するシステム権限**:** 次の表は、XSR オブジェクトへの権限を検査する際の、SQL特権に対応するシステム権限を示しています。左側の欄は、SQL 特権をリストしています。右側の欄は、同等のシステム権限をリストしています。

表 *112. XSR* オブジェクトへの権限を検査する際の対応するシステム権限

SQL の特権	対応するシステム権限
ALTER	*OBJALTER
USAGE	*OBJOPR および *EXECUTE および *READ

例

XSR オブジェクト XMLSCHEMA に関する USAGE 特権を PUBLIC に付与します。

```
GRANT USAGE
  ON XSROBJECT XMLSCHEMA
  TO PUBLIC
```

HOLD LOCATOR

HOLD LOCATOR ステートメントを使用すれば、作業単位が変わっても LOB ロケーター変数または XML ロケーター変数が値との関連を保持できるようになります。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。これを対話式に発行することはできません。これは、動的に準備できる実行可能ステートメントです。ただし、準備済みステートメントを実行するには、USING 文節を指定した EXECUTE ステートメントを使用しなければなりません。 HOLD LOCATOR は、EXECUTE IMMEDIATE ステートメントと併用することはできません。 REXX で指定してはなりません。

権限

権限は不要です。

構文

説明

変数,...

変数を指定します。この変数は、変数のロケーター変数を宣言する規則に従って宣言されていなければなりません。この変数に、標識変数を指定してはなりません。ロケーター変数のタイプは、バイナリー・ラージ・オブジェクト・ロケーター、文字ラージ・オブジェクト・ロケーター、2 バイト文字ラージ・オブジェクト・ロケーター、XML ロケーターのいずれかでなければなりません。

HOLD LOCATOR ステートメントが実行された後は、変数 リスト内の各ロケーター変数は保持プロパティーを持つことになります。

この変数には、現在ロケーターが割り当てられている必要があります。つまり、この作業単位中に (CALL、FETCH、SELECT INTO、SET 変数、または VALUES INTO ステートメントによって) ロケーターが割り当てられていなければならず、それ以降そのロケーターが (FREE LOCATOR ステートメントによって) 解放されていてはならない、ということです。そうでない場合には、エラーが発生します。

HOLD LOCATOR ステートメントに複数の変数が指定されていて、ロケーターの 1 つでエラーが発生した場合、どのロケーターも保持されません。

注記

保持プロパティーを持っている LOB ロケーター変数または XML ロケーター変数が解放される (つまり、変数と値の間の関連が除去される) のは、以下の場合です。

- そのロケーター変数を対象とする SQL FREE LOCATOR ステートメントが実行されたとき。

- HOLD オプションが指定されていない SQL ROLLBACK ステートメントが実行されたとき。
- SQL セッションが終了したとき。

例

従業員表に列 RESUME、HISTORY、および PICTURE が含まれていて、それらの列の値を表すためのロケーターがプログラムの中で確立されていると想定します。CLOB ロケーター変数 LOCRES および LOCHIST、および BLOB ロケーター変数 LOCPIC に、保持プロパティーを与えます。

```
HOLD LOCATOR :LOCRES,:LOCHIST,:LOCPIC
```

INCLUDE

INCLUDE ステートメントは、宣言とステートメントを含むアプリケーション・コードをソース・プログラムに挿入します。

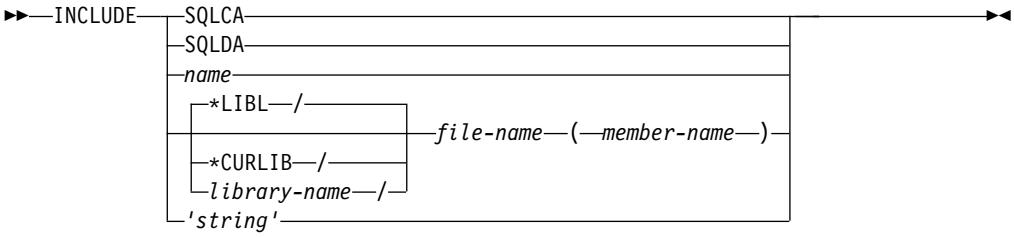
呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、実行可能ステートメントではありません。Java または REXX の場合、このステートメントを指定してはいけません。

権限

このステートメントの権限 ID は、メンバーを含むファイルについて、 *OBJOPR および *READ のシステム権限を持つ必要があります。

構文



説明

SQLCA

SQL 連絡域 (SQLCA) の記述を組み込むことを指定します。プログラムに独立型の SQLCODE または独立型の SQLSTATE を組み込む場合は、INCLUDE SQLCA を指定してはなりません。SQLCA は、C、C++、COBOL、および PL/I で指定できます。SQLCA を指定しない場合は、変数 SQLCODE または SQLSTATE をプログラム内で使用する必要があります。

INCLUDE SQLCA は、1 つのプログラムで一度しか指定できません。詳しくは、 876 ページの『SQL 診断情報』を参照してください。

RPG プログラムでは、SQLCA を指定してはなりません。 RPG プログラムの場合、プリコンパイラーによって自動的に SQLCA が組み込まれます。

SQLCA についての説明は、 1867 ページの『付録 C. SQLCA (SQL 連絡域)』を参照してください。

SQLDA

SQL 記述子域 (SQLDA) の記述を組み込むことを指定します。 INCLUDE SQLDA は、C、C++、COBOL、PL/I、および ILE RPG で指定できます。

SQLDA についての詳細は、 1877 ページの『付録 D. SQLDA (SQL 記述子域)』を参照してください。

name
ソース・プログラムに組み込むメンバーまたはソース・ストリーム・ファイルを識別します。

SRCFILE パラメーターを使用してプリコンパイルを行う場合、これは、CRTSQL*xxx* コマンドの INCFILE パラメーターで指定されているファイルから組み込むメンバーを識別します。

SRCSTMF パラメーターを使用してプリコンパイルを行う場合、これは、CRTSQL*xxx* コマンドの INCDIR パラメーターからのパスを使用して組み込むファイルを識別します。接尾部は、名前に付加されません。

このソースには、いずれかのホスト言語ステートメント、および INCLUDE ステートメント以外の SQL ステートメントを入れることができます。COBOL の場合、DATA DIVISION または PROCEDURE DIVISION 以外で INCLUDE メンバー名 を指定してはなりません。

file-name (member-name)
ソース・プログラムに組み込むソース・ファイルおよびメンバーを識別します。ソース・ファイルを含むライブラリーは、以下のいずれかの方法で指定します。

***LIBL**
そのジョブのライブラリー・リストのライブラリーが検索され、見つかった最初の表が使用されます。これはデフォルトです。

***CURLIB**
ジョブ用の現行ライブラリーが検索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL が使用されます。

library-name
検索するライブラリーの名前を識別します。

このソースには、いずれかのホスト言語ステートメント、および INCLUDE ステートメント以外の SQL ステートメントを入れることができます。COBOL の場合、DATA DIVISION または PROCEDURE DIVISION 以外で INCLUDE を指定してはなりません。

ソース・ファイルおよびメンバーの組み込みは、 SRCFILE パラメーターを使用してプリコンパイルするときにのみ可能です。

'string'
CRTSQL*xxx* コマンドの INCDIR パラメーターからのパスを使用して組み込むファイルを識別します。ストリングは、標準 SQL ストリング・リテラルとして処理されます。エスケープ文字に対するソース・ストリーム・ファイル規則には従いません。接尾部は、ストリングに付加されません。

このソースには、いずれかのホスト言語ステートメント、および INCLUDE ステートメント以外の SQL ステートメントを入れることができます。

ストリーム・ファイルの組み込みは、 SRCSTMF パラメーターを使用してプリコンパイルするときにのみ可能です。

INCLUDE ステートメントは、プログラムをプリコンパイルすると、ソース・ステートメントに置き換えられます。

このため、プログラムで INCLUDE ステートメントを指定する場合は、置き換え後のソース・ステートメントがコンパイラーに受け入れられるような場所に置かなければなりません。

注

CCSID に関する考慮事項: SRCFILE または SRCSTMF パラメーターで指定されたファイルの CCSID が INCLUDE ステートメントのソースの CCSID と異なる場合、INCLUDE ソースは、ソース・ファイルの CCSID に変換されます。

例

SQL 記述子域を C プログラムに組み込みます。

```
EXEC SQL INCLUDE SQLDA;

EXEC SQL DECLARE C1 CURSOR FOR
  SELECT DEPTNO, DEPTNAME, MGRNO FROM TDEPT
    WHERE ADMRDEPT = 'A00';

EXEC SQL OPEN C1;

while (SQLCODE==0) {
  EXEC SQL FETCH C1 INTO :dnum, :dname, :mnum;

/* Print results */

}

EXEC SQL CLOSE C1;
```

INSERT

INSERT ステートメントは、表またはビューに行を挿入します。ビューで INSTEAD OF INSERT トリガーが定義されていない場合にそのビューに行を挿入すると、そのビューの基礎になっている表にも行が挿入されます。そのようなトリガーが定義されていれば、代わりにこのトリガーが活動化されます。

このステートメントには、以下の 4 つの形式があります。

- *VALUES* を使用した *INSERT* の形式は、提供された値または参照された値を使用して、表またはビューに 1 つ以上の行を挿入する時に使用します。
- 全選択を使用する *INSERT* の形式は、他の表やビューからの値を使用して、表またはビューに 1 つ以上の行を挿入する時に使用します。
- *ROWS* を使用する *INSERT* 形式は、ホスト構造体配列で用意されている値を使用して、表またはビューに複数の行を挿入する場合に使用します。
- *INSERT DEFAULT VALUES* の形式は、すべての列にデフォルト値を使用して 1 つの行を挿入するときに使用します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは動的に準備できる実行可能なステートメントです。ただし、ROWS の形式は例外で、アプリケーション・プログラムに組み込まれる静的ステートメントでなければなりません。 REXX プロシージャーでは、n ROWS 形式は使用できません。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

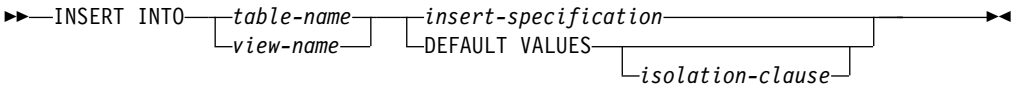
- ステートメントに指定された表またはビューに対して、
  - その表やビューについての INSERT 特権、および
  - 表またはビューが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

全選択 が指定されている場合、ステートメントの権限 ID によって保持される特権には次の 1 つが含まれていなければなりません。

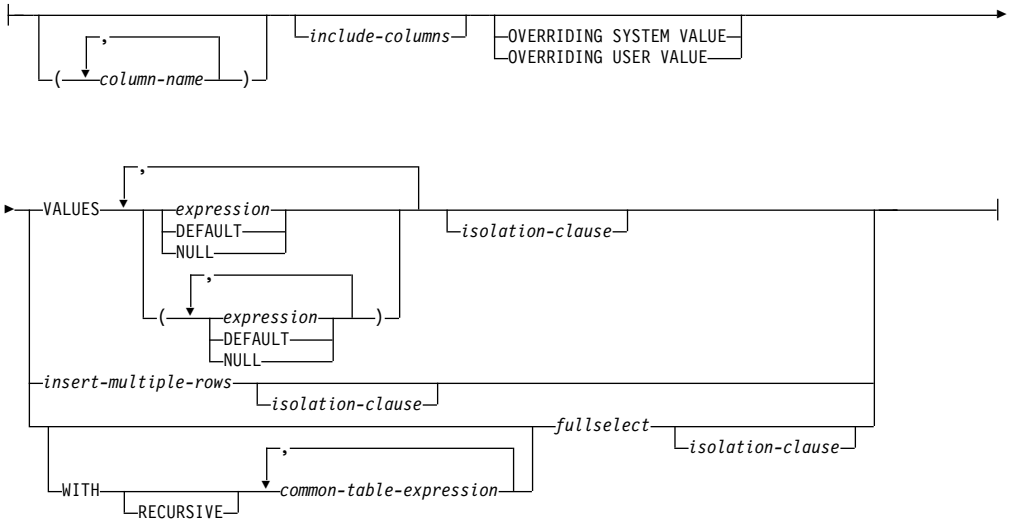
- 全選択 内で識別された、それぞれの表またはビューごとに、
  - 表やビューに対する SELECT 特権、および
  - 表またはビューが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

SQL 特権に対応するシステム権限については『表またはビューへの権限を検査する際の対応するシステム権限』を参照してください。

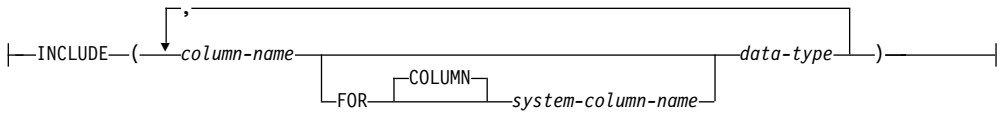
構文



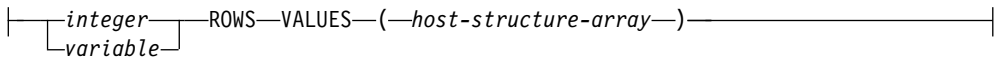
insert-specification:



include-columns:

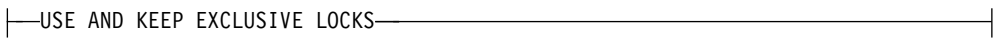


insert-multiple-rows:

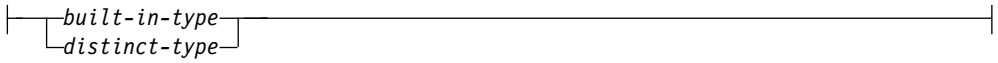


isolation-clause:

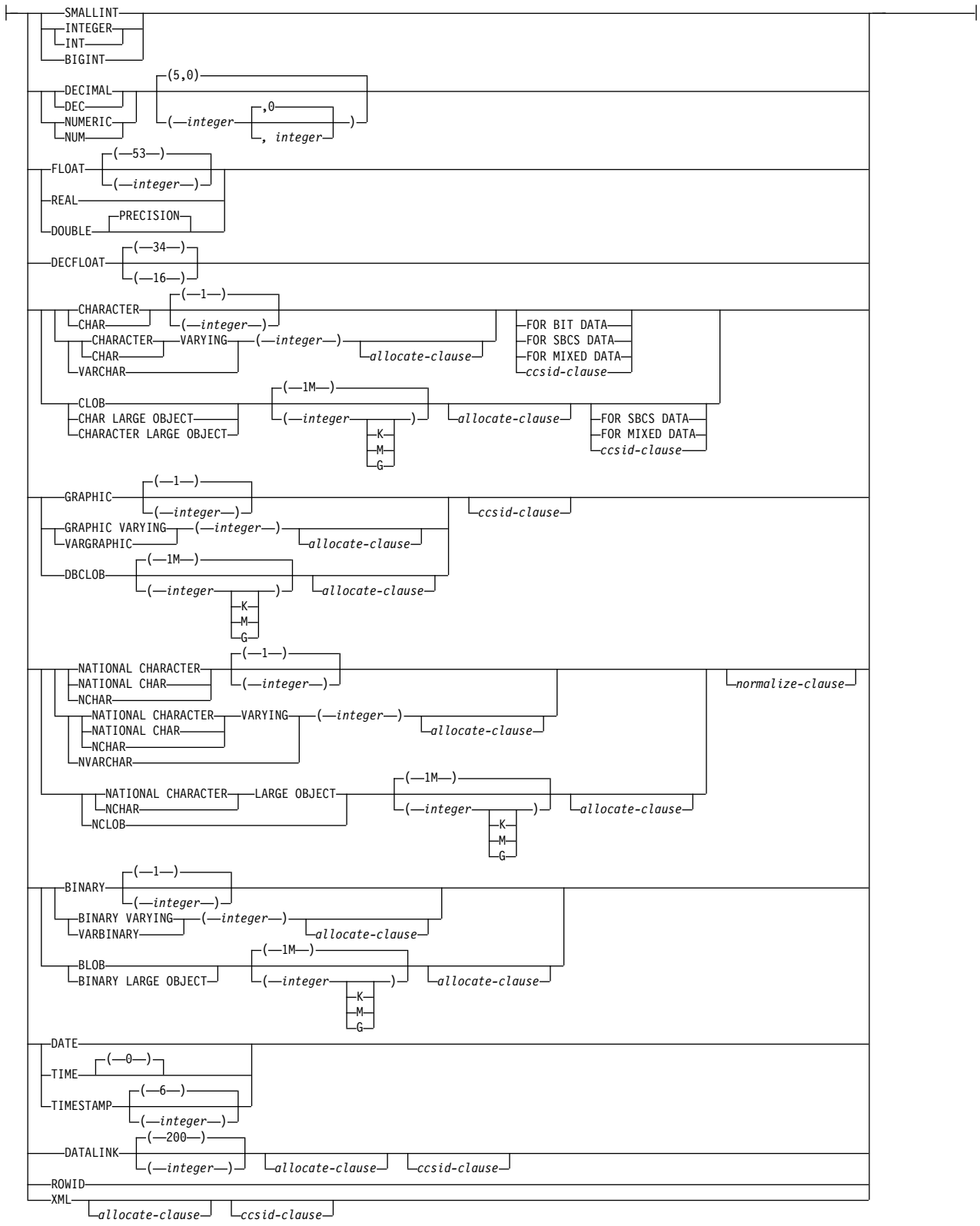
lock-clause:



data-type:

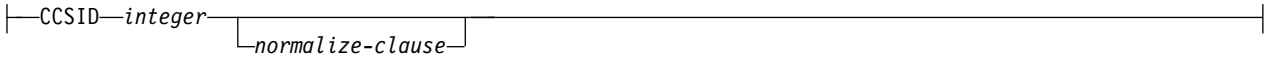


built-in-type:



allocate-clause:

ccsid-clause:



normalize-clause:

説明

INTO *table-name* または *view-name*
挿入操作の対象のオブジェクトを指定します。この名前は、現行サーバーに存在している表またはビューを識別していなければなりませんが、履歴表、カタログ表、カタログ表のビュー、または挿入可能ではないビューを識別するものであってはなりません。挿入可能なビューの説明については、 1342 ページの『CREATE VIEW』を参照してください。

DEFAULT VALUES
表内のすべての列にデフォルト値が割り当てられることを指定します。これは、すべての列にキーワード DEFAULT の値をそれぞれ含む値リストを指定することと同じです。挿入される値は、DEFAULT キーワードの説明にあるように、列がどのように定義されたかによって異なります。

(*column-name***,...)**
値を挿入する列を指定します。それぞれの名前は、表またはビューの列を識別する名前でなければなりません。同じ列を複数回指定することはできません。拡張標識変数が使用できないと、更新不可能なビュー列を識別することはできません。拡張標識変数が使用できず、挿入操作の対象となるビューに上記のような列がある場合は、列名のリストを指定しなければなりません。この列名のリストから、値を挿入できない列の名前を除外する必要があります。ビュー内の更新可能な列の説明については、 1342 ページの『CREATE VIEW』を参照してください。

列名のリストを指定しなかった場合は、該当する表またはビューにあるすべての列を左から右の順序で指定したものと見なされます。隠し属性で定義されたすべての列が省略されます。このリストは、ステートメントを準備するときに確立されるので、ステートメントを準備した後で表に追加した列をリストに指定してはなりません。

INSERT ステートメントがアプリケーションに組み込まれ、参照している表またはビューがプログラムの作成時に存在している場合には、そのステートメントはプログラムの作成時に準備されます。これ以外の場合には、INSERT ステートメントは、そのステートメントの最初の正常な実行時に準備されます。

include-columns
全選択の FROM 文節にネストされているとき、 *table-name* や *view-name* などの列と一緒に INSERT ステートメントの中間結果表に組み込まれている列セットを指定します。 *include-columns* は、*table-name* や *view-name* で指定されている列のリストの最後に付加されます。

INCLUDE
INSERT ステートメントの中間結果表に組み込まれる列のリストを指定します。この節は、INSERT ステートメントが全選択の FROM 文節にネストされている場合にのみ指定できます。

column-name
INSERT ステートメントの中間結果表の列を指定します。この名前は、他の組み込み列や *table-name* または *view-name* 内の列と同一の名前にはできません。

FOR COLUMN *system-column-name*
列の IBM i 名を指定します。この名前は、INCLUDE 列リスト内、または *table-name* や *view-name* 内のいずれかの *column-name* または *system-column-name* と同一であってはなりません。

data-type
組み込み列のデータ・タイプを指定します。 *data-type* の説明は、 1238 ページの『CREATE TABLE』を参照してください。 DATALINK データ・タイプを使用した場合は、FILE LINK CONTROL を使用できません。

OVERRIDING SYSTEM VALUE または **OVERRIDING USER VALUE**
ROWID、識別、または行変更タイム・スタンプ列に、システムが生成した値やユーザーが指定した値を使用するかどうかを指定します。OVERRIDING SYSTEM VALUE を指定する場合は、INSERT ステートメントの対象とする暗黙または明示的な列リストに、GENERATED ALWAYS として定義された ROWID 列、ID 列、または行変更タイム・スタンプ列が含まれていることが必要です。 OVERRIDING USER VALUE を指定する場合は、INSERT ステートメントの対象とする暗黙または明示的な列リストに、GENERATED ALWAYS または GENERATED BY DEFAULT として定義された列が含まれていることが必要です。

OVERRIDING SYSTEM VALUE
GENERATED ALWAYS として定義されている ROWID 列、ID 列、または行変更タイム・スタンプ列について、VALUES 文節に指定されている値または全選択の結果として得られた値を使用することを指定します。システム生成の値は挿入されません。

行開始列、行終了列、トランザクション開始 ID 列、または生成式列の値が提供された場合、それは DEFAULT でなければなりません。

OVERRIDING USER VALUE
GENERATED ALWAYS または GENERATED BY DEFAULT として定義されている列について、VALUES 文節に指定されている値または全選択の結果として得られた値を無視することを指定します。代わりにシステム生成の値が挿入され、ユーザー指定の値はオーバーライドされます。

OVERRIDING SYSTEM VALUE と OVERRIDING USER VALUE のどちらも指定しない場合は、以下のようになります。

- GENERATED ALWAYS として定義された ROWID 列、ID 列、行変更タイム・スタンプ列、行開始列、行終了列、トランザクション開始 ID 列、および生成式列に、DEFAULT 以外の値を指定することはできません。

- GENERATED BY DEFAULT として定義された ROWID、識別、または行変更タイム・スタンプ列には、値を指定することができます。値を指定した場合は、この列にその値が割り当てられます。ただし、BY DEFAULT として定義された ROWID 列に値を挿入できるのは、指定された値が、Db2 for z/OS または Db2 for i によって既に生成されている有効な行 ID 値である場合のみです。 BY DEFAULT として定義された識別または行変更タイム・スタンプ列に値を挿入した場合は、その識別または行変更タイム・スタンプ列が固有制約または固有索引内の唯一のキーである場合以外は、データベース・マネージャーはその指定された値が該当の列についての固有な値であるかどうかを検査しません。固有制約も固有索引もない場合は、データベース・マネージャーは、NO CYCLE が有効である場合に限り、システム生成の値のセットの中でのみ各値の固有性を保証します。

  値が指定されていない場合は、データベース・マネージャーは新しい値を生成します。

VALUES

挿入する 1 つ以上の新しい行を指定します。

この文節に指定する各変数は、ホスト構造体や変数の宣言の規則に従って宣言されているホスト構造体または変数を識別していなければなりません。このステートメントの操作形式では、ホスト構造体に対する参照は、その個々の変数それぞれに対する参照によって置き換えられます。変数と構造についての詳細は、172 ページの『ホスト変数に対する参照』および 179 ページの『ホスト構造』を参照してください。

VALUES 文節内の各行の値の数は、暗黙的または明示的な列のリスト、および INCLUDE 文節で識別された列にある名前の数と一致していなければなりません。リストの最初の列には VALUES 文節の最初の値が挿入され、リストの 2 番目の列には VALUES 文節の 2 番目の値が挿入されるというように、指定した列に対応する値が順に挿入されます。

expression

集約関数または列名を含まない、 196 ページの『式』で説明されているタイプの *expression*。式 が変数 の場合、その変数は構造体を識別できます。拡張標識変数が使用可能で、式が単一変数でない場合、DEFAULT および UNASSIGNED の拡張標識変数値は、その式に対して使用してはなりません。

DEFAULT

列にデフォルト値を割り当てることを指定します。挿入する値は、次のように、列がどのように定義されたかによって異なります。

- 式に基づく生成列として列が定義されている場合は、その式に基づいた列の値がデータベース・マネージャーによって生成されます。
- 列が ROWID 列、ID 列、行変更タイム・スタンプ列、行開始列、行終了列、またはトランザクション開始 ID 列の場合、データベース・マネージャーは新しい値を生成します。
- WITH DEFAULT 文節が使用される場合、挿入されるデフォルト値は、その列に関して定義されている値になります (1238 ページの『CREATE TABLE』の列定義 のデフォルト文節 を参照してください)。

- WITH DEFAULT 文節または NOT NULL 文節が使用されない場合、挿入される値は NULL です。
- NOT NULL 文節が使用されていて、WITH DEFAULT 文節が使用されていないか、 DEFAULT NULL が使用されている場合、その列については DEFAULT キーワードは指定できません。

GENERATED ALWAYS として定義されている ROWID 列または 生成列については、OVERRIDING USER VALUE を指定することによりユーザー指定の値を無視してシステム生成の固有値を挿入することを指示した場合以外は、DEFAULT を指定する必要があります。

NULL
列の値を NULL 値にすることを指定します。NULL は、NULL 可能列にのみ指定してください。

WITH *common-table-expression*
共通表式を指定します。共通表式については、 848 ページの『共通表式』を参照してください。

fullselect
全選択の結果表の形式で、新しい行の集合を指定します。結果表が空の場合、SQLSTATE は「02000」に設定されます。

全選択 の説明については、 841 ページの『全選択』を参照してください。

INSERT の基本オブジェクトと、全選択 内のいずれかの副選択の基本オブジェクトが同じ表であるとき、その選択ステートメントは、行が挿入される前にすべて評価されます。

結果表の列の数と、列名 のリストに暗黙的または明示的に指定した名前の数は同じでなければなりません。リストの最初の列には、結果表の最初の列の値が挿入され、リストの 2 番目の列には、結果表の 2 番目の列が挿入されるというように、対応する列の値が順に挿入されます。

isolation-clause
このステートメントに関して使用する分離レベルを指定します。

WITH

分離レベルを指定します。次のいずれかになります。
- *RR* 反復可能読み取り
- *RS* 読み取り固定
- *CS* カーソル固定
- *UR* 非コミット読み取り
- *NC* コミットなし

ISOLATION 文節 を指定しなかった場合は、デフォルトの分離レベルが使用されます。デフォルトの判別方法については、 859 ページの『ISOLATION 文節』を参照してください。

insert-multiple-rows

整数 または 変数 ROWS
挿入する行数を指定します。変数 を指定する場合、その変数は位取りゼロの数値でなければならず、また標識変数を含むことはできません。

VALUES (ホスト構造体配列)
ホスト構造体の配列の形式で新しい一連の行を指定します。ホスト構造体配列は、その宣言の規則に従ってプログラムで宣言されていなければなりません。ホスト構造体配列 名の代わりに、パラメーター・マーカーを使用することはできません。

ホスト構造の変数の数は、暗黙的または明示的な列のリスト、および INCLUDE 文節で識別された列にある名前の数と一致していなければなりません。配列の最初のホスト構造体は最初の行に対応し、配列の 2 番目のホスト構造体は 2 番目の行に対応します。以下同様です。さらに、ホスト構造体の最初の変数は、該当の行の最初の列に対応し、ホスト構造体の 2 番目の変数は、該当の行の 2 番目の列に対応します。以下同様です。

ホスト構造の配列についての説明は、 181 ページの『ホスト構造配列』を参照してください。

現行接続が非リモート・サーバーへの接続の場合、*insert-multiple-rows* は使用できません。 RPG/400 または PL/I プログラムのデータ変更参照では、複数行挿入 は使用できません。

INSERT の規則

デフォルト値**:** 列のリストに指定されていない列には、その列のデフォルト値が挿入されます。デフォルト値を持たない列は、列リストに含めなければなりません。同様に、INSTEAD OF INSERT トリガーを使用せずビューに値を挿入する場合に、そのビューに含まれていない基本表の列があれば、基本表の該当する列には、デフォルト値が挿入されます。このため、ビューに入っていない基本表の列は、すべてデフォルト値を持っています。

割り当て**:** 挿入する値は、 57 ページの『第 2 章 言語エレメント』で説明されている記憶域割り当て規則に従って、列に割り当てられます。

妥当性検査**:** 挿入操作は、以下の規則に従う必要があります。以下の規則に従わない場合や、INSERT ステートメントの実行中にその他のエラーが発生した場合は、行が挿入されません。ただし、COMMIT(*NONE) を指定した場合は例外です。

- 固有制約および固有索引*:* 識別された表、または識別されたビューの基本表が 1 つ以上の固有索引または固有制約を持つ場合は、表に挿入される各行は、それらの索引および制約によって課せられる制限に適合しなければなりません (SQLSTATE 23505)。

  すべての固有性検査は、COMMIT(*NONE) の指定がある場合を除き、そのステートメントの終わりで実際に行われます。複数行 INSERT ステートメントの場合、このチェックはすべての行が挿入された後に行われます。COMMIT(*NONE) が指定されている場合は、各行が挿入されるごとにチェックが行われます。
- 検査制約*:* 識別された表、または識別されたビューの基本表が、1 つ以上の検査制約を持つ場合、表に挿入される各行ごとに、検査制約は真または不明でなければなりません (SQLSTATE 23513)。

  検査制約は、ステートメントの終わりで必ずチェックされます。複数行 INSERT ステートメントの場合、このチェックはすべての行が挿入された後に行われます。

- ビューと *CHECK OPTION* 文節*:* ビューが識別されている場合は、挿入される行は適用される CHECK OPTION 文節に適合しなければなりません (SQLSTATE 44000)。詳しくは、 1342 ページの『CREATE VIEW』を参照してください。

トリガー**:** 識別された表または識別されたビューの基本表が挿入トリガーを持つ場合、トリガーが起動されます。トリガーが起動された結果、挿入する値に応じて、他のステートメントが実行されたり、エラー条件が発生したりすることがあります。 INSERT ステートメントをデータ変更表参照 として使用すると、挿入行の変更を試みる AFTER INSERT トリガーがエラーを起こします。

参照保全**:** 外部キーの非 NULL の挿入値は、関連の親表の親キーの値のいずれかに等しくなければなりません。

参照制約 (RESTRICT 削除規則を伴う参照制約以外の) は、ステートメントの終わりで実際上チェックされます。複数行 INSERT ステートメントの場合、これはすべての行が挿入され、関連のトリガーが起動された後で行われます。

INSERT ステートメントをデータ変更表参照 として使用すると、挿入行の変更を試みるいずれかの参照制約がエラーを起こします。

XML 値**:** XML 列に挿入する値は、整形式 XML 文書でなければなりません。

行アクセス制御または列アクセス制御が適用されている表への行の挿入**:** 行アクセス制御または列アクセス制御が適用されている表に対して INSERT ステートメントが発行されると、有効な行の許可または列マスクに指定されている規則によって、行を挿入できるかどうかが判定されます。通常、これらの規則は、ステートメントの許可 ID に基づきます。以下の規則は、INSERT 中に、使用可能な行の許可および列マスクが使用される方法について説明しています。

- 新しい行の値を導出するときに列が参照されている場合に、その列に対して有効な列マスクがあれば、マスクされた値が新しい値を導出するために使用されます。オブジェクト表でも列アクセス制御がアクティブになっている場合、新しい値を導出するために適用される列マスクは、定数や式ではなく、列自体を戻す必要があります。列マスクを列に適用した結果が列自体にならない場合、新しい値は挿入に使用できず、エラーが戻されます。

  OVERRIDING USER VALUE 文節を指定する場合は、新しい行内の対応する値は無視されるので、列マスクに関する上記の規則はこれらの値には適用できません。
- 行を挿入でき、表に関する BEFORE INSERT トリガーがある場合、このトリガーがアクティブになる。

  トリガー・アクション内で、挿入対象の新しい値を遷移変数中で変更できます。トリガーから値が戻る際に、新しい値については最後の値が挿入対象になります。
- 挿入される行は、有効な行の許可に準拠しなければならない。

  表に関する使用可能な行の許可が複数定義されている場合、行アクセス制御検索条件は、それぞれの使用可能な行の許可内の検索条件に対して論理 OR 演算子を

適用することにより派生します。有効な行の許可のすべてに準拠する行は、挿入された場合にも、行アクセス制御検索条件を使用して取り戻すことができます。

- 行を挿入でき、表に関する AFTER INSERT トリガーがある場合、このトリガーがアクティブになる。

前述の規則は *include-columns* には適用されません。 *include-columns* は、マスクされているかどうかに関わらず、割り当てられるすべての値を受け入れます。

マスクされたデータを、挿入操作の値として使用される変数に割り当てることができます。列に挿入違反チェック制約が存在しない場合、マスクされたデータが列に挿入され、エラーは発行されません。

拡張標識変数の使用**:** 使用可能な場合は、正の値および 0 (ゼロ) から -7 以外の標識変数値を設定しないでください。 DEFAULT および UNASSIGNED 拡張標識変数値を、それらがサポートされていないコンテキストに指定しないでください。

拡張標識変数**:** INSERT ステートメントにおける UNASSIGNED の拡張標識値には、列をデフォルト値に設定する効果があります。

更新可能でないターゲット列は、GENERATED ALWAYS として定義されている生成列でない限り、拡張標識変数値 UNASSIGNED が割り当てられなければなりません。ターゲット列が GENERATED ALWAYS として定義される生成列である場合は、拡張標識変数値 DEFAULT または UNASSIGNED をこれに割り当てる必要があります。

拡張標識変数と挿入トリガー**:** 拡張標識変数の有無により、挿入トリガーの起動に変更が生じることはありません。暗黙的または明示的な列リスト内のすべての列に、拡張標識変数に基づく UNASSIGNED 値または DEFAULT が割り当てられている場合、すべての列に個別の DEFAULT 値を持つ挿入が試行され、これが正常に完了すると、挿入トリガーが起動されます。

拡張標識変数とエラー検査の据え置き**:** 拡張標識変数が使用可能な場合、更新不可能な列への挿入を認識するために、ステートメント準備の間に通常行われる妥当性検査は、ステートメントが実行されるまで据え置かれます。

注

挿入操作エラー**:** COMMIT(*NONE) を指定していない状態で、挿入値がいずれかの制約に違反している場合や、その他のエラーが INSERT ステートメントの実行中に発生した場合は、そのステートメントとトリガー SQL ステートメントによる変更がすべてロールバックされます。ただし、エラーが発生する前に、その作業単位の中で行われていたその他の変更はロールバックされません。COMMIT(*NONE) が指定されていれば、変更がロールバックされることはありません。

挿入された行数**:** INSERT ステートメントの実行後、 SQL 診断領域の ROW_COUNT ステートメント情報項目 (または SQLCA の SQLERRD(3)) は、データベース・マネージャーが挿入した行の数となります。 ROW_COUNT 項目には、トリガーの結果として挿入された行の数は含まれません。

ROW_COUNT についての説明は、 1489 ページの『GET DIAGNOSTICS』を参照してください。SQLCA についての説明は、 1867 ページの『付録 C. SQLCA (SQL 連絡域)』を参照してください。

ロッキング: COMMIT(*RR)、COMMIT(*ALL)、COMMIT(*CS)、または COMMIT(*CHG) が指定されている場合は、正常に実行される INSERT ステートメントの実行中に、 1 つ以上の排他的ロックが掛けられます。そのようなロックがコミットまたはロールバック操作によって解放されるまで、挿入された行は、以下によってのみアクセスすることができます。

- その挿入を行ったアプリケーション・プロセス
- 読み取り専用操作を介して、COMMIT(*NONE) または COMMIT(*CHG) を使用する別のアプリケーション・プロセス

ロックは、他のアプリケーション・プロセスがその表の操作を行うのを防止します。ロッキングの詳細については、 1038 ページの『COMMIT』、 1661 ページの『ROLLBACK』、および 1580 ページの『LOCK TABLE』 ステートメントの説明を参照してください。また、 32 ページの『分離レベル』および「データベース・プログラミング」も参照してください。

FINAL TABLE を指定した際に、INSERT をデータ変更表参照 として使用すると、SELECT が完了するまで、ロックは挿入行に置かれます。これらのロックは、挿入行の変更を試みる AFTER TRIGGER などによる、同一のジョブ内からの挿入行に対する間接的な変更を妨げる可能性があります。これらのロックは、COMMIT(*NONE) を含め、すべての分離レベルに対して獲得されます。

COMMIT(*RR)、COMMIT(*ALL)、COMMIT(*CS)、または COMMIT(*CHG) を指定した場合は、1 つの INSERT ステートメントで最高 500 000 000 行を挿入または変更することができます。変更される行の数には、トリガーの結果として同じコミットメント定義のもとで挿入、更新、または削除される行が含まれます。

生成列: GENERATED ALWAYS として定義された生成列は、VALUES リスト中の対応する項目が DEFAULT でない限り、列リスト に指定しないでください。ユーザーは、OVERRIDING USER VALUE 文節を指定して、ユーザー指定の値が無視され、INSERT の時点のシステム生成値がこの列に挿入されるように指示することができます。

行開始列、行終了列、またはトランザクション開始 **ID** 列を持つ表への挿入: これらの生成列を持つ表 (例えば、システム期間テンポラル表) に行を挿入するとき、データベース・マネージャーは以下の列に値を割り当てます。

- 行開始列に割り当てられる値は、次のいずれかの場合に時刻機構を読み取ることによって生成されます。(1) トランザクションの中で、表に含まれる行開始列またはトランザクション開始 ID 列に値を割り当てる必要があるようなデータ変更ステートメントを最初に実行するとき。(2) システム期間テンポラル表に含まれる行を削除するとき。行開始列の値は、トランザクション全体にわたり固有になるようにデータベース・マネージャーによって生成されます。単一の SQL トランザクション内で複数の行が挿入される場合、行開始列の値はすべての行において同じになり、別のトランザクションでその列のために生成された値とは異なる固有の値になります。

- 行終了列には、この列 (9999-12-30-00.00.00.000000000000) の最大値が割り当てられます。
- トランザクション開始 ID 列には、トランザクションごとに固有のタイム・スタンプ値、または NULL 値が割り当てられます。トランザクション開始 ID 列が NULL 可能である場合には、この列に NULL 値が割り当てられます。それ以外の場合、この値は、次のいずれかの場合に時刻機構を読み取ることによって生成されます。(1) トランザクションの中で、表に含まれる行開始列またはトランザクション開始 ID 列に値を割り当てる必要があるようなデータ変更ステートメントを最初に実行するとき。(2) システム期間テンポラル表に含まれる行を削除するとき。単一の SQL トランザクション内で複数の行が挿入される場合、トランザクション開始 ID 列の値はすべての行において同じになり、別のトランザクションでその列のために生成された値とは異なる固有の値になります。

システム期間テンポラル表への挿入**:** システム期間テンポラル表に行を挿入するとき、データベース・マネージャーは、行開始列、行終了列、またはトランザクション開始 ID 列を持つ表に対する指示に従って列に値を割り当てます。また、行を挿入するとき、そのシステム期間テンポラル表に関連付けられた履歴表に行は追加されません。

CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターに NULL 以外の値が設定され、 SYSTIME オプションの値が YES の場合は、INSERT ステートメントの基礎ターゲットを、システム期間テンポラル表にすることはできません。この制限は、システム期間テンポラル表への参照が直接か間接かにかかわらず適用されます。

REXX: 変数は、REXX プロシージャー内の INSERT ステートメントでは使用できません。INSERT を使用する場合は、必ず、パラメーター・マーカーを使用する PREPARE および EXECUTE の対象として使用してください。

代替構文**:** 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード NONE を NC の同義語として使用することができます。
- キーワード CHG を UR の同義語として使用することができます。
- キーワード ALL を RS の同義語として使用することができます。

例

例 *1:* 以下の仕様を持つ新規の部門を、DEPARTMENT 表に挿入します。

- 部門番号 (DEPTNO) は 'E31'
- 部門名 (DEPTNAME) は 'ARCHITECTURE'
- その管理者の社員番号 (MGRNO) は '00390'
- 報告先の部門 (ADMRDEPT) は 'E01'

```
INSERT INTO DEPARTMENT
  VALUES ('E31', 'ARCHITECTURE', '00390', 'E01')
```

例 *2:* 例 1 と同様に新規の部門を DEPARTMENT 表に挿入します。ただし、この新規の部門には管理者を割り当てません。

```
INSERT INTO DEPARTMENT (DEPTNO, DEPTNAME, ADMRDEPT)
  VALUES ('E31', 'ARCHITECTURE', 'E01')
```

例 *3:* EMPPROJACT 表と同じ列構成で、表 MA_EMPPROJACT を作成します。EMPPROJACT 表からプロジェクト番号 (PROJNO) が文字「MA」で始まっている行を、データとして MA_EMPPROJACT に追加します。

```
CREATE TABLE MA_EMPPROJACT LIKE EMPPROJACT

INSERT INTO MA_EMPPROJACT
  SELECT * FROM EMPPROJACT
    WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

例 *4:* Java プログラムのステートメントを使用して、接続コンテキスト「ctx」上の PROJECT 表にプロジェクトの骨組みを追加します。プロジェクト番号 (PROJNO)、プロジェクト名 (PROJNAME)、部門番号 (DEPTNO)、および管理担当者 (RESPEMP) の値は、ホスト変数から入手します。プロジェクト開始日付 (PRSTDATE) には、現在の日付を使用します。表内のその他の列には、NULL 値を割り当てておきます。

```
#sql [ctx] { INSERT INTO PROJECT (PROJNO, PROJNAME, DEPTNO, RESPEMP, PRSTDATE)
             VALUES (:PRJNO, :PRJNM, :DPTNO, :REMP, CURRENT DATE) };
```

例 *5:* 例 2 と同じように 1 つのステートメントを使用して 2 つの新しい部門を表 DEPARTMENT に挿入します。ただし、この新しい部門には管理担当者を割り当てません。

```
INSERT INTO DEPARTMENT (DEPTNO, DEPTNAME, ADMRDEPT)
  VALUES ('B11', 'PURCHASING', 'B01'),
         ('E41', 'DATABASE ADMINISTRATION', 'E01')
```

例 *6:* PL/I プログラムで、複数行 INSERT を使用して、表 DEPARTMENT に 10 行を追加します。挿入するデータは、ホスト構造体配列 DEPT に入っています。

```
DCL  1  DEPT(10),
        3  DEPT  CHAR(3),
        3  LASTNAME  CHAR(29)  VARYING,
        3  WORKDEPT  CHAR(6),
        3  JOB  CHAR(3);

EXEC SQL INSERT INTO DEPARTMENT 10 ROWS VALUES (:DEPT);
```

例 *7:* READ UNCOMMITTED (UR, CHG) オプションを使用して、EMPPROJACT 表に新しいプロジェクトを挿入します。

```
INSERT INTO EMPPROJACT
  VALUES ('000140', 'PL2100', 30)
  WITH CHG
```

例 *8:* SELECT ステートメントで、INSERT ステートメントをデータ変更表参照 として指定します。 VALUE 節で値が指定されている組み込み列を余分に定義し、それを、挿入される行の配列用の列として使用します。

```
SELECT inorder, ordernum
  FROM FINAL TABLE (INSERT INTO ORDERS (CUSTNO)
                                INCLUDE(INSERTNUM INTEGER)
                    VALUES (:cnum1, 1),
                           (:cnum2, 2)) InsertedOrders
  ORDER BY insertnum
```

LABEL

LABEL ステートメントは、種々のデータベース・オブジェクトのカタログ記述にラベルを追加したり、置換したりします。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

表、ビュー、別名、列、タイプ、パッケージ、シーケンス、または XSR オブジェクトに対してラベルを付けるには、ステートメントの権限 ID が保持する特権に、次のうち少なくとも 1 つを含める必要があります。

- ステートメントで識別されている表、ビュー、別名、タイプ、パッケージ、シーケンス、変数、XSR オブジェクト、関数、またはプロシージャーの場合
  - その表、ビュー、別名、タイプ、パッケージ、シーケンス、変数、XSR オブジェクト、関数、またはプロシージャーに対する ALTER 特権、および
  - その表、ビュー、別名、タイプ、パッケージ、シーケンス、変数、XSR オブジェクト、関数、またはプロシージャーが入っているスキーマに対する USAGE 特権
- データベース管理者権限

制約またはトリガーに対してラベルを付けるには、ステートメントの権限 ID が保持する特権に、次のうち少なくとも 1 つを含める必要があります。

- ステートメント内の制約またはトリガーのサブジェクト表に対して、
  - 対象表に対する ALTER 特権
  - 対象表が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

索引にラベルを付けるためには、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメントで識別される索引に対して、
  - その索引についての *OBJALTER システム権限
  - 索引が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

関数にラベルを付けるためには、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- SYSFUNCS および SYSROUTINES カタログ・ビューと表の場合
  - SYSROUTINES に対する UPDATE 特権
  - SYSFUNCS に対する *OBJOPR システム権限、および
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

プロシージャーにラベルを付けるためには、ステートメントの権限 ID によって保持される特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- SYSPROCS および SYSROUTINES カタログ・ビューと表の場合:
  - SYSROUTINES に対する UPDATE 特権
  - SYSPROCS に対する *OBJOPR システム権限、および
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

マスクまたは許可にラベルを付けるためには、次が必要です。

- このステートメントの許可 ID には、セキュリティー管理者権限 がなければなりません。 21 ページの『管理権限』を参照してください。

シーケンスにラベルを付けるためには、ステートメントの権限 ID によって保持される特権にも、少なくとも次のいずれか 1 つが含まれなければなりません。

- データ域変更 (CHGDTAARA) CL コマンドに対する *USE 権限
- データベース管理者権限

ステートメントの権限 ID は、次の場合に別名に対する ALTER 権限を保有します。

- その別名の所有者である場合。
- その別名の *OBJALTER または *OBJMGT のいずれかのシステム権限が認可されている場合。

変数にラベルを付けるためには、ステートメントの権限 ID によって保持される特権にも、少なくとも次のいずれか 1 つが含まれなければなりません。

- SYSVARIABLES カタログ表の場合
  - SYSVARIABLES に対する UPDATE 特権
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

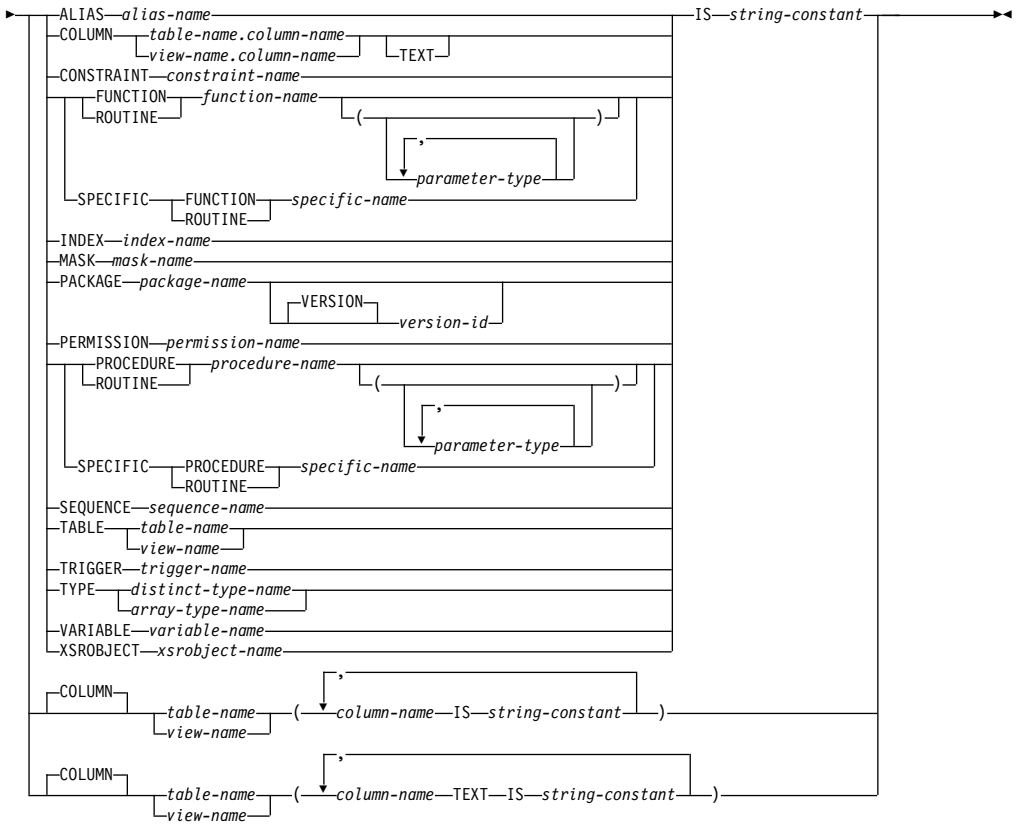
XSR オブジェクトにラベルを付けるためには、ステートメントの権限 ID によって保持される特権にも、少なくとも次のいずれか 1 つが含まれなければなりません。

- XSROBJECTS カタログ表の場合
  - XSROBJECTS に対する UPDATE 特権
  - スキーマ QSYS2 に対する USAGE 特権
- データベース管理者権限

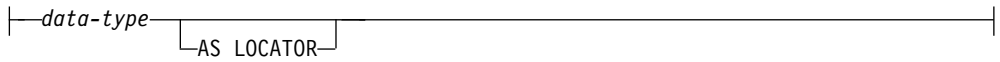
SQL 特権に対応するシステム権限の説明については、表またはビューへの権限を検査する際の対応するシステム権限、シーケンスへの権限を検査する際の対応するシステム権限、およびパッケージへの権限を検査する際の対応するシステム権限を参照してください。

構文

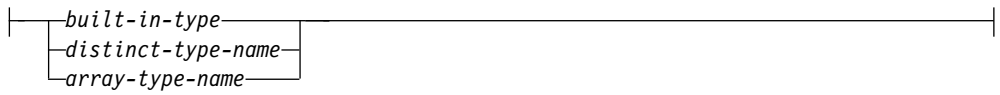
LABEL



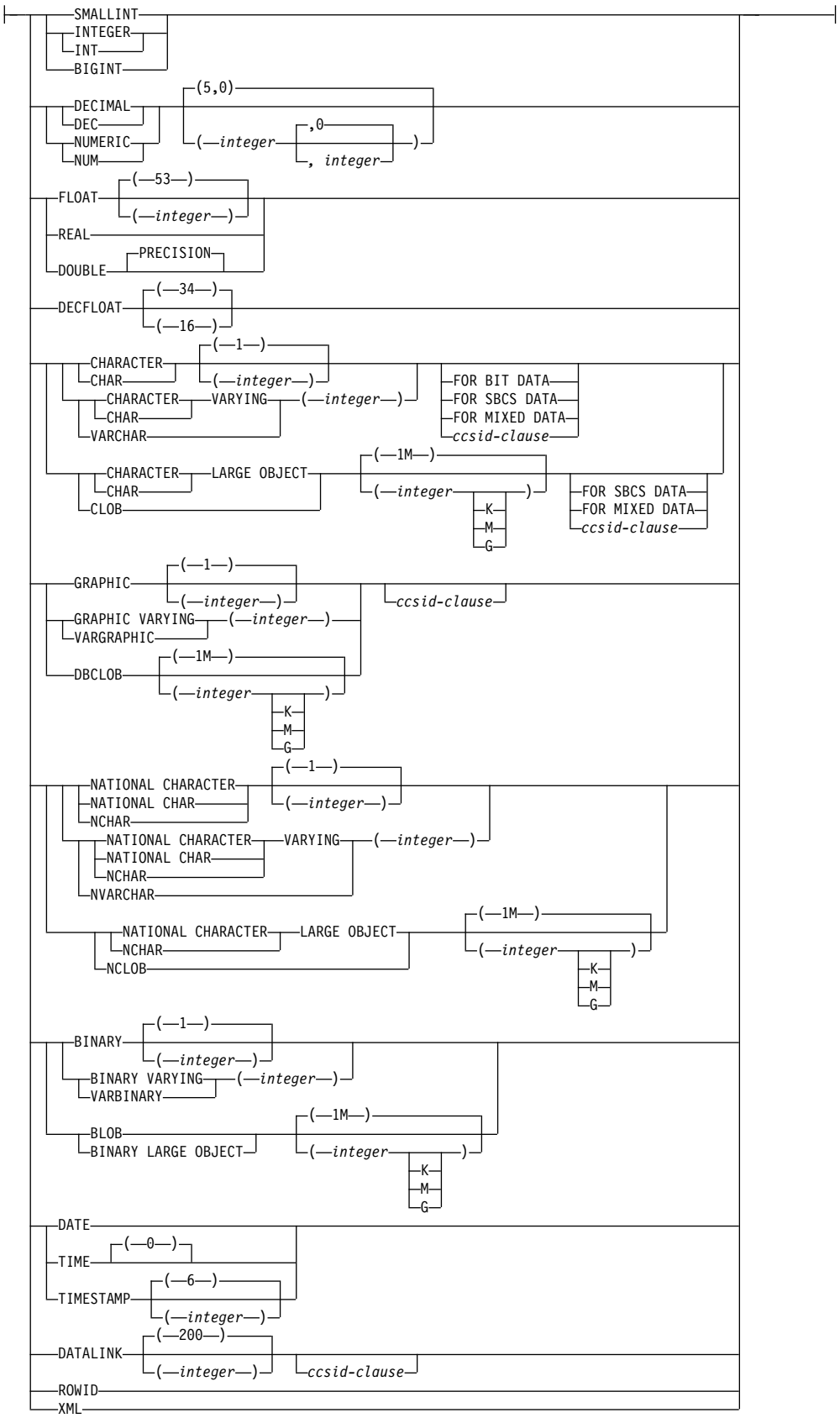
parameter-type:



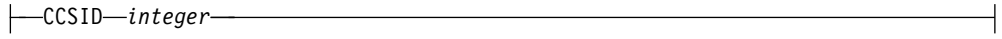
data-type:



built-in-type:


SMALLINT
INTEGER
INT
BIGINT
DECIMAL
DEC
NUMERIC
NUM
(5,0)
(integer
,0
, integer
)
FLOAT
(53)
(integer)
REAL
DOUBLE
PRECISION
DECFLOAT
(34)
(16)
CHARACTER
CHAR
(1)
(integer)
CHARACTER VARYING
CHAR VARYING
VARCHAR
FOR BIT DATA
FOR SBCS DATA
FOR MIXED DATA
ccsid-clause
CHARACTER LARGE OBJECT
CHAR LARGE OBJECT
CLOB
(1M)
K
M
G
GRAPHIC
GRAPHIC VARYING
VARGRAPHIC
DBCLOB
NATIONAL CHARACTER
NATIONAL CHAR
NCHAR
NATIONAL CHARACTER VARYING
NATIONAL CHAR VARYING
NVARCHAR
NATIONAL CHARACTER LARGE OBJECT
NCHAR LARGE OBJECT
NCLOB
BINARY
BINARY VARYING
VARBINARY
BLOB
BINARY LARGE OBJECT
DATE
TIME
(0)
TIMESTAMP
(6)
DATALINK
(200)
ROWID
XML

ccsid-clause:



説明

ALIAS
別名のラベルであることを指定します。別名のラベルは、システム・オブジェクト・テキストとして付加されます。

alias-name
ラベルを適用する別名を識別します。この名前は、現行サーバーに存在している別名を識別していなければなりません。

COLUMN
列のラベルであることを指定します。列のラベルは、システム列見出しまたは列テキストとして付加されます。列見出しは、照会の結果を表示または印刷する場合に使用されます。

table-name.column-name **or** *view-name.column-name*
ラベルを適用する列を指定します。表名 またはビュー名 は、現行サーバーにある表またはビューを示すものでなければなりませんが、宣言済み一時表を示すものであってはなりません。列名 は、その表またはビューの列を識別するものでなければなりません。

TEXT
IBM i の列テキストを使用することを指定します。 TEXT を省略すると、列見出しが指定されます。

CONSTRAINT
制約に対するラベルであることを指定します。

constraint-name
ラベルを適用する制約を識別します。 *constraint-name* は、現行サーバーに存在する制約を識別する必要があります。

FUNCTION または SPECIFIC FUNCTION
ラベルの適用対象である関数を識別します。この関数は現行サーバーに存在していなければならず、ユーザー定義関数である必要があります。関数は、それぞれその名前、関数シグニチャー、あるいは特定名によって識別することができます。

FUNCTION *function-name*
関数を名前によって識別します。 *function-name* は厳密に 1 つの関数を示す必要があります。この関数には、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前の関数が複数ある場合、エラーが戻されます。

FUNCTION *function-name (parameter-type, ...)*
関数を一意的に識別する関数シグニチャーによって、関数を識別します。 *function-name (parameter-type,...)* は、指定された関数シグニチャーを持つ関数を識別しなければなりません。指定されたパラメーターは、関数の作成時に指定された、対応する位置にあるデータ・タイプと一致していなければなりません。ラベルを付ける関数インスタンスを識別するには、データ・タイ

プの数とデータ・タイプの論理連結を使用します。データ・タイプの同義語は、一致として扱われます。デフォルトがあるパラメーターは、このシグニチャーに含まれていなければなりません。

function-name () を指定する場合、識別される関数にパラメーターを使用することはできません。

function-name
関数の名前を識別します。

(parameter-type, ...)
関数のパラメーターを識別します。

非修飾の特殊タイプ名を指定する場合、データベース・マネージャーはその特殊タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、以下のいずれかを使用します。

- 空の括弧は、データベース・マネージャーがデータ・タイプの一致の判別に際して属性を無視することを示します。例えば、DEC() は、DEC(7,2) のデータ・タイプで定義された関数のパラメーターに一致するものとみなされます。ただし、FLOAT に空の括弧を指定することはできません。これは、そのパラメーター値が特定のデータ・タイプ (REAL または DOUBLE) を示しているからです。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を指定した場合、その値は、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプが FLOAT の場合、突き合わせはデータ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度は指定された値に厳密に一致している必要はありません。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。暗黙の長さは、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

FOR DATA 文節または CCSID 文節の指定はオプショナルです。いずれの文節も指定しないと、データ・タイプが一致するかどうかを判定する場合に、データベース・マネージャーが属性を無視することを示します。どちらか一方の文節を指定する場合は、CREATE FUNCTION ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

AS LOCATOR
関数が、このパラメーターのロケーターを受け取るように定義されることを示します。 AS LOCATOR を指定する場合は、データ・タイプは LOB または XML、あるいは LOB または XML に基づく特殊タイプでなければなりません。 AS LOCATOR を指定した場合、FOR SBCS DATA または FOR MIXED DATA を指定してはなりません。

SPECIFIC FUNCTION *specific-name*
関数を特定名によって識別します。 *specific-name* は、現行サーバーに存在する特定の関数を示している必要があります。

INDEX
索引のラベルであることを指定します。索引のラベルは、システム・オブジェクト・テキストとして付加されます。

index-name
ラベルを適用する索引を識別します。この名前は、現行サーバーに存在している索引を識別していなければなりません。

MASK
マスクのラベルであることを指定します。

mask-name
ラベルを適用するマスクを指定します。この名前は、現行サーバーに存在するマスクを示している必要があります。

PACKAGE
パッケージのラベルであることを指定します。パッケージのラベルは、システム・オブジェクト・テキストとして付加されます。

package-name
ラベルを適用するパッケージを識別します。この名前は、現行サーバーに存在しているパッケージを識別していなければなりません。

VERSION *version-id*
バージョン *ID* は、作成時にパッケージに割り当てられたバージョン ID です。バージョン *ID* を指定しない場合、バージョン ID として NULL ストリングが使用されます。

PERMISSION
許可のラベルであることを指定します。

permission-name
ラベルを適用する権限を指定します。この名前は、現行サーバーに存在する許可を示すものでなければなりません。

PROCEDURE または **SPECIFIC PROCEDURE**
ラベルを適用するプロシージャーを識別します。このプロシージャー名 は、現行サーバーに存在しているプロシージャーを識別していなければなりません。

PROCEDURE *procedure-name*
プロシージャーを名前によって識別します。プロシージャー名 は、ただ 1 つのプロシージャーを識別していなければなりません。このプロシージャーには、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前のプロシージャーが複数ある場合、エラーが戻されます。

PROCEDURE *procedure-name (parameter-type, ...)*
プロシージャーを一意的に識別するプロシージャー・シグニチャーによって、プロシージャーを識別します。 *procedure-name (parameter-type,...)* では、指定されたプロシージャー・シグニチャーを持つプロシージャーを識別する必要があります。指定されたパラメーターは、プロシージャーの作成時に指定された、対応する位置にあるデータ・タイプと一致していなければな

りません。ラベルを付ける対象のプロシージャー・インスタンスを識別するには、データ・タイプの数とデータ・タイプの論理連結を使用します。データ・タイプの同義語は、一致として扱われます。デフォルトがあるパラメーターは、このシグニチャーに含まれていなければなりません。

プロシージャー名 *()* を指定する場合、識別されるプロシージャーにパラメーターを使用することはできません。

procedure-name
プロシージャーの名前を識別します。

(parameter-type, ...)
プロシージャーのパラメーターを識別します。

非修飾の特殊タイプ名または配列タイプ名を指定する場合、データベース・マネージャーはその特殊タイプまたは配列タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、以下のいずれかを使用します。

- 空の括弧は、データベース・マネージャーがデータ・タイプの一致の判別に際して属性を無視することを示します。例えば、DEC() は、DEC(7,2) のデータ・タイプで定義されたプロシージャーのパラメーターに一致するものとみなされます。ただし、FLOAT に空の括弧を指定することはできません。これは、そのパラメーター値が特定のデータ・タイプ (REAL または DOUBLE) を示しているからです。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を指定する場合、その値は、 CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプが FLOAT の場合、突き合わせはデータ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度は指定された値に厳密に一致している必要はありません。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。暗黙の長さは、CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

FOR DATA 文節または CCSID 文節の指定はオプショナルです。いずれの文節も指定しないと、データ・タイプが一致するかどうかを判定する場合に、データベース・マネージャーが属性を無視することを示します。どちらか一方の文節を指定する場合は、CREATE PROCEDURE ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

AS LOCATOR
プロシージャーが、このパラメーターのロケーターを受け取るように定義されることを示します。 AS LOCATOR を指定する場合は、データ・タイプは LOB または XML、あるいは LOB または XML に基づく特殊タイプでなければなりません。 AS LOCATOR を指定した場合、FOR SBCS DATA または FOR MIXED DATA を指定してはなりません。

SPECIFIC PROCEDURE *specific-name*
プロシージャーを特定名によって識別します。特定名 は、現行サーバーに存在している特定のプロシージャーを識別していなければなりません。

SEQUENCE
シーケンスのラベルであることを指定します。シーケンスのラベルは、システム・オブジェクト・テキストとして付加されます。

sequence-name
シーケンスを識別します。ここで指定した表またはビューに関するラベルが追加されます。シーケンス名 は、現行サーバーに存在するシーケンスを示すものでなければなりません。

TABLE
表またはビューのラベルであることを指定します。表またはビューのラベルは、システム・オブジェクト・テキストとして付加されます。

table-name または *view-name*
表またはビューを識別します。ここで指定した表またはビューに関するラベルが追加されます。表名 またはビュー名 は、現行サーバーにある表またはビューを示すものでなければなりませんが、宣言済み一時表を示すものであってはなりません。

TRIGGER
トリガーに対するラベルであることを指定します。

trigger-name
ラベルを追加したいトリガーを識別します。トリガー名 は、現行サーバーに存在しているトリガーを識別していなければなりません。

TYPE *distinct-type-name* または *array-type-name*
ラベルを適用する特殊タイプまたは配列タイプを識別します。特殊タイプ名 または 配列タイプ名 は、現行サーバーに存在するタイプを示すものでなければなりません。

VARIABLE *variable-name*
ラベルを適用する変数を識別します。この変数名 は、現行サーバーに存在している変数を識別していなければなりません。

XSROBJECT *xsrobject-name*
ラベルを適用する XSR オブジェクトを識別します。この*XSR* オブジェクト名は、現行サーバーに存在しているXSR オブジェクトを示すものでなければなりません。

IS この後に、付加したいラベルを指定します。

string-constant
列の見出しの場合は、60 バイトまでの長さ、オブジェクト・テキストまたは列のテキストの場合は、50 バイトまでの長さであればどのような SQL 文字ストリング定数でも構いません。この定数には、1 バイト文字および 2 バイト文字を入れることができます。

列見出しとしてのラベルは、 20 バイトまでの 3 つの部分 (セグメント) から構成されています。対話式 SQL、Query for IBM i、IBM Db2 Query Manager and SQL Development Kit for i、およびその他のプロダクトによって、各 20 バイトのセグメントをそれぞれ別の行に表示または印刷する

ことができます。列のラベルに混合データが使用される場合、20 バイトのそれぞれのセグメントは、有効な混合データ文字ストリングでなければなりません。シフト文字は、 20 バイトのセグメントの中で対になっていなければなりません。

注

列見出し**:** 列見出しは、照会の結果を表示または印刷する場合に使用されます。最初の列見出しは最初の行に、2 番目の列見出しは 2 行目に、3 番目の列見出しは 3 行目に、それぞれ表示または印刷されます。列見出しは最高 60 バイトまでですが、最初の 20 バイトが最初の列見出し、 2 番目の 20 バイトが 2 番目の列見出し、3 番目の 20 バイトが 3 番目の列見出しになります。ブランクは、それぞれの 20 バイトの列見出しの終わりから削除されます。

列見出し情報の 60 バイトすべてがカタログ・ビュー SYSCOLUMNS で使用可能です。ただし、DESCRIBE または DESCRIBE TABLE ステートメントの SQLDA で戻されるのは、最初の列見出しだけです。

DESCRIBE または DESCRIBE TABLE ステートメントでは、列テキストが戻されません。データベース・マネージャーが、共用されているレコード様式の記述の列見出しの情報を変更すると、その変更は、その様式の記述を共用するすべてのファイルに反映されます。ファイルが他のファイルと様式を共用しているかどうかを調べるには、 CL コマンドのデータベース関係表示 (DSPDBR) の RCDFMT パラメーターを使用します。

代替構文**:** 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- PACKAGE の同義語として、キーワード PROGRAM を使用することができます。
- キーワード DATA TYPE または DISTINCT TYPE を TYPE の同義語として使用することができます。

例

例 *1:* 表 DEPARTMENT の DEPTNO 列にラベルを付けます。

```
LABEL ON COLUMN DEPARTMENT.DEPTNO
  IS 'DEPARTMENT NUMBER'
```

例 *2:* 列見出しが 2 行に表示されている、表 DEPARTMENT の列 DEPTNO にラベルを付けます。

```
LABEL ON COLUMN DEPARTMENT.DEPTNO
  IS 'Department          Number'
```

例 *3:* パッケージ PAYROLL にラベルを付けます。

```
LABEL ON PACKAGE PAYROLL
  IS 'Payroll Package'
```

LOCK TABLE

LOCK TABLE ステートメントは、並行して実行されるアプリケーション・プロセスによる表の変更や表の使用を防止します。

呼び出し

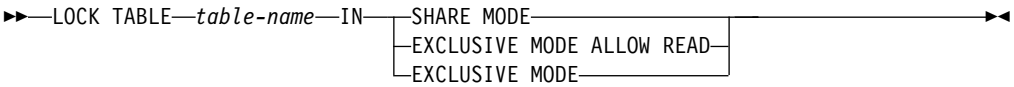
このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメントで識別される表に対して、
  - その表についての *OBJOPR システム権限、および
  - 表が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

構文



説明

table-name
ロックする表を識別します。この表名では、現行サーバーに存在する基本表を指定する必要があります。カタログ表または宣言済み一時表は、指定しないでください。

IN SHARE MODE
並行して実行されているアプリケーション・プロセスが、この表に対して読み取り専用操作以外の操作を実行できないようにします。

このステートメントが実行されるアプリケーション・プロセス用の共用ロック (*SHRNUP) が確立されます。他のアプリケーション・プロセスが共用ロック (*SHRNUP) を確立することもあり、その場合、このアプリケーション・プロセスが読み取り専用以外の操作を実行できないようにします。

IN EXCLUSIVE MODE ALLOW READ
並行して実行されているアプリケーション・プロセスが、この表に対して読み取り専用操作以外の操作を実行できないようにします。

このステートメントが実行されるアプリケーション・プロセス用の排他読み取り許可ロック (*EXCLRD) が確立されます。他のアプリケーション・プロセスは共用ロック (*SHRNUP) を確立できず、その場合、このアプリケーション・プロセスが該当の表に対して更新、削除、および挿入を実行できないようにすることは不可能です。

IN EXCLUSIVE MODE
並行して実行されるアプリケーション・プロセスが、この表に対していかなる操作も実行できないようにします。

このステートメントが実行されるアプリケーション・プロセス用の排他ロック (*EXCL) が確立されます。

注

取得されたロック: ロッキングは、並行操作を防止するために使用されます。

ロックは、以下の時点で解放されます。

- 作業単位が終了したとき。ただし、作業単位が COMMIT HOLD または ROLLBACK HOLD によって終了した場合を除きます。
- プログラム・スタックの中の最初の SQL プログラムが終了したとき。ただし、CLOSQLCSR(*ENDJOB) または CLOSQLCSR(*ENDACTGRP) が CRTSQLxxx コマンドで指定されていた場合を除きます。
- 活動化グループが終了したとき。
- 接続が CONNECT (タイプ 1) ステートメントの使用により変更されたとき。
- そのロックに関連する接続が DISCONNECT ステートメントの使用により切り離されたとき。
- 接続が解除保留状態にあり、正常な COMMIT が行われたとき。

また、オブジェクト割り振り解除 (DLCOBJ) コマンドを出して、表をアンロックすることもできます。

ロック待機時間: 競合するロックを既に他のアプリケーション・プロセスが保持していると、現在使用しているアプリケーションは待ち状態になります (待ち時間の上限は、ジョブのデフォルトの待ち時間です)。

例

表 DEPARTMENT をロックします。

```
LOCK TABLE DEPARTMENT IN EXCLUSIVE MODE
```

MERGE

MERGE ステートメントは、ソース (表参照の結果) のデータを使用してターゲット (表またはビュー) を更新します。入力データと一致する行がターゲットに存在する場合は、その行を指定内容に基づいて更新/削除できます。ターゲットに存在しない行を指定内容に基づいて挿入することも可能です。ビューの行を更新/削除/挿入すると、そのビューで INSTEAD OF トリガーが定義されていない限り、ビューの基本表の行も更新/削除/挿入されます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、あるいは対話式に実行することもできます。これは、動的に準備できる実行可能ステートメントです。このステートメントは、REXX プロシージャーでは使用できません。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- データベース管理者権限
- 挿入操作を指定する場合は、以下のようになります。
  - その表やビューについての INSERT 特権、および
  - 表またはビューが含まれるスキーマに対する USAGE 特権
- 削除操作を指定する場合は、以下のようになります。
  - その表またはビューに対する DELETE 特権
  - 表またはビューが含まれるスキーマに対する USAGE 特権
- 更新操作を指定する場合は、以下のようになります。
  - 表やビューに対する UPDATE 特権、または
  - 更新される各列に対する UPDATE 特権、または
  - 表またはビューが含まれるスキーマに対する USAGE 特権

さらに、*search-condition*、*insert-operation*、*assignment-clause* のいずれかに全選択を組み込む場合は、ステートメントの権限 ID の特権に少なくとも以下のいずれか 1 つの権限が含まれていなければなりません。

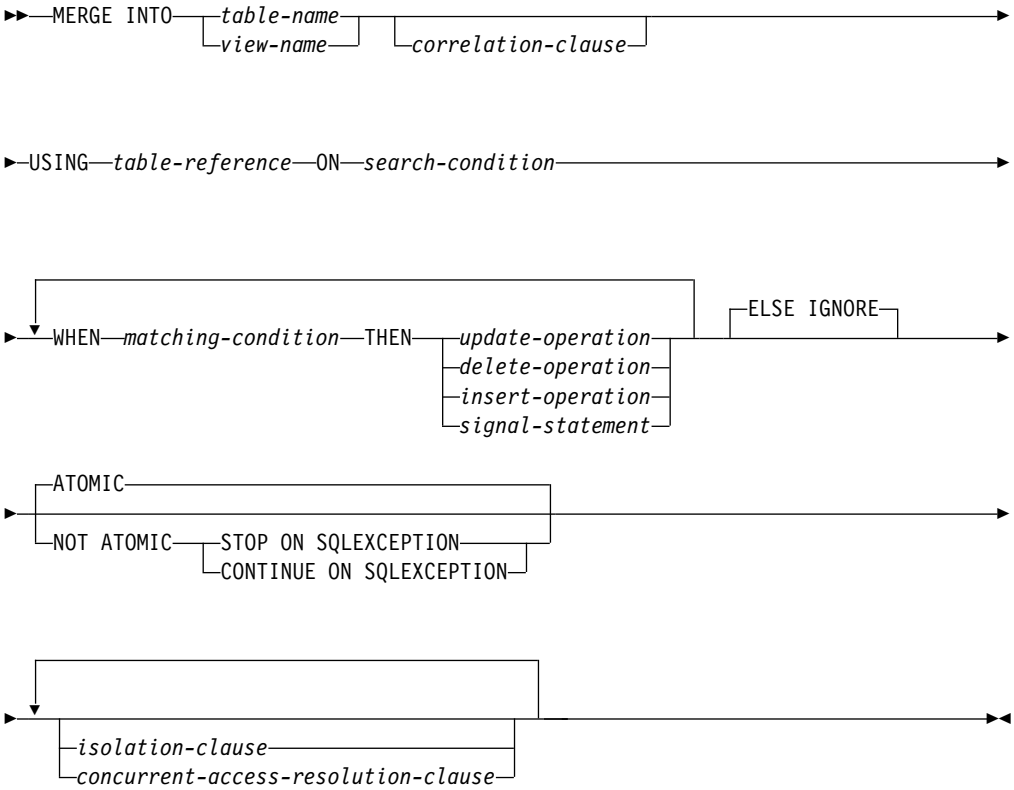
- 全選択 内で識別された、それぞれの表またはビューごとに、
  - 表やビューに対する SELECT 特権、および
  - 表またはビューが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

さらに、表またはビューの列を参照する照会を *table-reference* に組み込む場合は、ステートメントの権限 ID の特権に少なくとも以下のいずれか 1 つの権限が含まれていなければなりません。

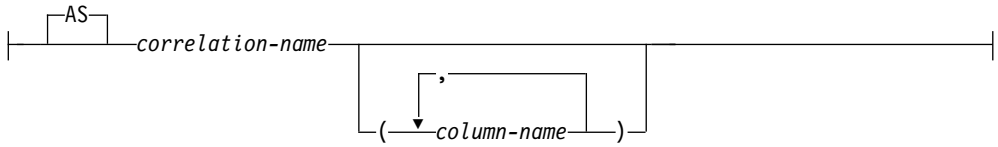
- 全選択 内で識別された、それぞれの表またはビューごとに、
  - 表やビューに対する SELECT 特権、および
  - 表またはビューが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

SQL 特権に対応するシステム権限については、『表またはビューへの権限を検査する際の対応するシステム権限』を参照してください。

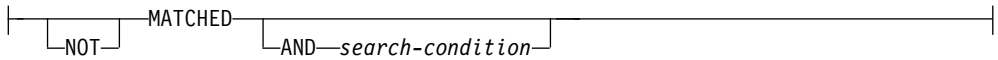
構文



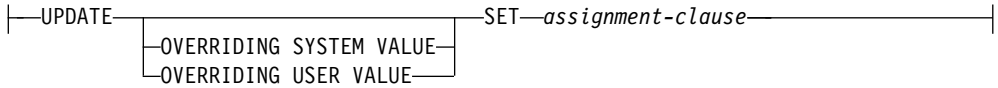
correlation-clause:



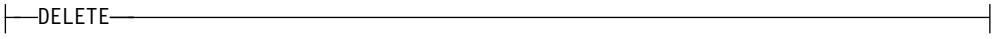
matching-condition:



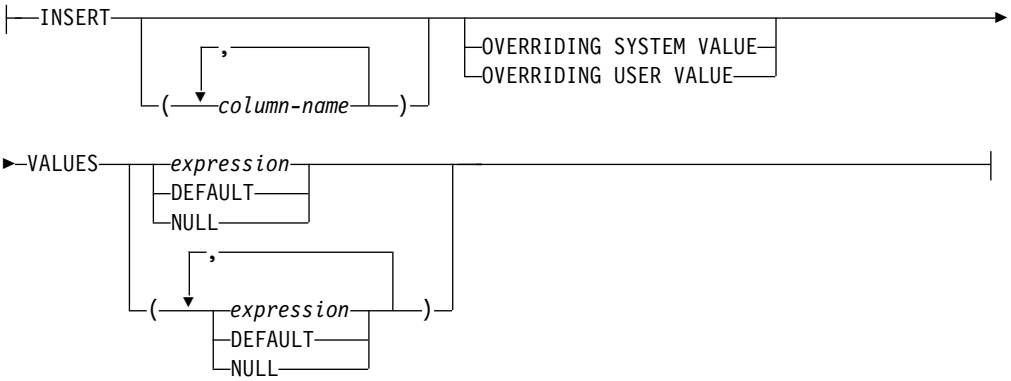
update-operation:



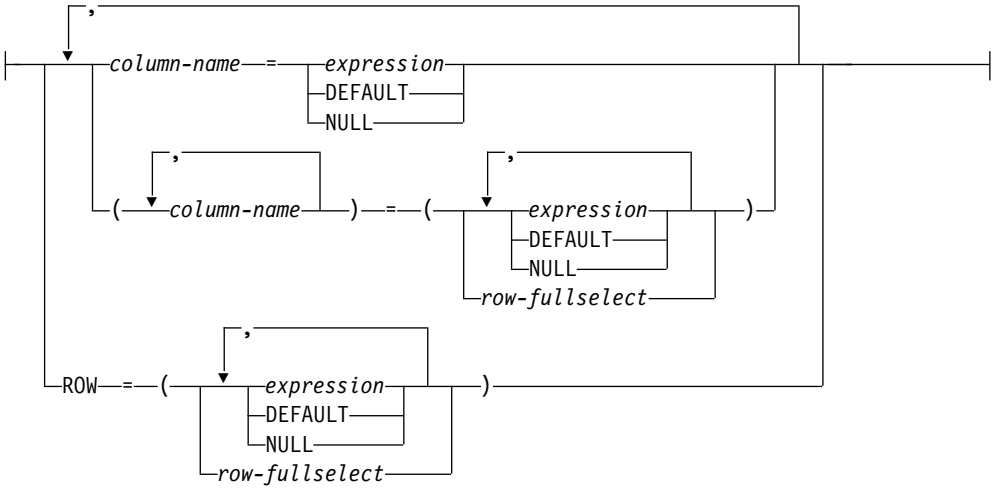
delete-operation:



insert-operation:



割り当て文節**:**



isolation-clause:

説明

table-name または *view-name*
マージの更新、挿入、削除の操作のターゲットを指定します。この名前は、現行サーバーに存在する表またはビューを示していなければなりませんが、カタログ表、カタログ表のビュー、読み取り専用ビュー、または、ビューに対して定義さ

れた INSTEAD OF トリガーに無関係の削除不能ビューを示すものであってはなりません。*table-name* が履歴表を識別する場合、MERGE ステートメントに更新操作および挿入操作を指定することはできません。

AS *correlation-name*
検索条件 やマッチング条件 の中で、または割り当て文節 の右側で、ターゲット表またはビューを指定するために使用できます。表またはビューの列に対する参照を修飾するためには、*correlation-name* を使用します。相関名 を指定する場合は、列名 も同時に指定することによって、表名 またはビュー名 の列に名前を指定することができます。列リストを指定する場合は、その列リストの中で、表またはビューの各列について、それぞれ名前を指定する必要があります。詳しくは、 163 ページの『相関名』を参照してください。

USING *table-reference*
ターゲットにマージする結果表として行のセットを指定します。結果表が空の場合、警告が戻されます。

ON *search-condition*
ソース表 からの行とターゲット表の行が一致するかどうかを判定するために使用される述部を指定します。

search-condition に組み込む各 *column-name* では、ターゲット表またはターゲット・ビューまたは *table-reference* の列名を指定しなければなりません (ただし、副照会の内部は除きます)。検索条件に副照会を組み込む場合に、マージとその副照会の基本オブジェクトが同じ表であれば、行の更新/挿入操作が実行される前にその副照会が完全に評価されます。

search-condition は、ターゲット表と *table-reference* の結果表の各行に適用されます。 *table-reference* の結果表の行のうち、*search-condition* の結果が真になった行では、指定した更新/削除操作が実行されます。結果表の行のうち、*search-condition* の結果が真にならなかった行では、指定した挿入操作が実行されます。

定量副照会、副選択を入れた IN 述部、EXISTS 副照会を search-condition に組み込むことはできません。基本述部副照会またはスカラー全選択は含めることができます。集約関数または非決定的なスカラー関数を使用した式を組み込むこともできません。

WHEN *matching-condition*
update-operation、*delete-operation*、*insert-operation*、または *signal-statement* が実行される条件を指定します。それぞれの *matching-condition* は、指定順で評価されます。*matching-condition* の評価が真になった行は、その後のマッチング条件の対象になりません。

MATCHED
ON *search-condition* が真になった行で実行する操作を指定します。THEN の後に指定できるのは、UPDATE、DELETE、*signal-statement* に限られます。

AND *search-condition*
THEN の後に指定する操作の実行に関する ON 検索条件に合致した行に適用する追加の検索条件を指定します。

検索条件 には、EXISTS または IN 述部に副照会を組み込むことはできません。

NOT MATCHED
ON *search-condition* が偽または不明になった行で実行する操作を指定します。THEN の後に指定できるのは、INSERT または *signal-statement* に限られます。

AND *search-condition*
THEN の後に指定する操作の実行に関する ON 検索条件に合致しなかった行に適用する追加の検索条件を指定します。

検索条件 には、EXISTS または IN 述部に副照会を組み込むことはできません。

THEN
マッチング条件 が真と評価された時に実行する操作を指定します。

update-operation
matching-condition の評価が真になった行で実行する更新操作を指定します。

assignment-clause
列の更新情報のリストを指定します。

column-name
更新する列を識別します。*column-name* では、ターゲット表またはターゲット・ビューの列を指定する必要があります。スカラー関数、定数、式から派生するビューの列を *column-name* で指定することはできません。列名を複数回指定することはできません。

1 つのビューに同じ列から得られる 2 つの列がある場合、その列の値を更新することは可能ですが、その 2 つの列を同一の MERGE ステートメントで更新することはできません。

ROW
ターゲット表またはビューのすべての列 (ただし、隠し属性を指定して定義された列以外) を識別します。ビューが指定されている場合、そのビューの列がまったく、スカラー関数、定数、または式から派生していない場合があります。

式、NULL、および DEFAULT の数 (または *row-fullselect* の結果列の数) は、行の列の数と一致していなければなりません。

ビューに、そのビューの別の列から派生したビュー列が含まれている場合、そのビューに ROW を指定することはできません。これは、両方の列を同じ UPDATE 内で更新できないためです。

expression
列の新しい値を示します。この式の中で集約関数を指定することはできません (ただし、集約関数がスカラー全選択の中にある場合を除きます)。

ターゲットの *table-name* または *view-name* の列に対する参照をこの式に組み込むこともできます。更新する行ごとに、式に組み込むターゲット列の参照の値として、行の更新前の列の値を指定します。

expression がソース表の単一の列への参照である場合、ソース表の列の値は、拡張標識変数の値で指定された可能性があります。そのような標識変数は、*assignment-clause* の対応するターゲット列に影響を与えます。

拡張標識変数が使用可能であり、式が以下の参照よりも複雑な場合には、拡張標識変数の値 DEFAULT (-5) または UNASSIGNED (-7) を使用してはなりません。

- ソース表の単一の列
- 単一のホスト変数

DEFAULT
列にデフォルト値を割り当てるように指定します。DEFAULT は、デフォルト値のある列のみに対して指定できます。デフォルト値について詳しくは、 1238 ページの『CREATE TABLE』 の DEFAULT 節の説明を参照してください。

ユーザー指定の値をすべて無視して固有のシステム生成値を使用するよう指示する OVERRIDING USER VALUE が指定されていない場合、GENERATED ALWAYS として定義された列には DEFAULT を指定する必要があります。GENERATED BY DEFAULT として定義される列に関しては、有効な値を指定することができます。

NULL
列の新しい値に NULL 値を指定します。NULL 可能列にのみ NULL を指定します。

delete-operation
matching-condition の評価が真になった行で実行する削除操作を指定します。

insert-operation
matching-condition の評価が真になった行で実行する挿入操作を指定します。

INSERT
挿入操作で使用する列名と行値式のリストの前に記述します。

行値式にある値の数は、暗黙的または明示的な挿入列リストに含まれている名前の数と同じでなければなりません。リストの最初の列には VALUES 文節の最初の値が挿入され、リストの 2 番目の列には VALUES 文節の 2 番目の値が挿入されるというように、指定した列に対応する値が順に挿入されます。

(*column-name***,...)**
値を挿入する列を指定します。それぞれの名前は、表またはビューの列を識別する名前でなければなりません。同じ列を複数回指定することはできません。拡張標識変数が使用できないと、更新不可能なビュー列を

識別することはできません。拡張標識変数が使用できず、挿入操作の対象となるビューに上記のような列がある場合は、列名のリストを指定しなければなりません。この列名のリストから、値を挿入できない列の名前を除外する必要があります。ビュー内の更新可能な列の説明については、 1342 ページの『CREATE VIEW』を参照してください。

列名のリストを指定しなかった場合は、該当する表またはビューにあるすべての列を左から右の順序で指定したものと見なされます。隠し属性で定義されたすべての列が省略されます。このリストは、ステートメントを準備するときに確立されるので、ステートメントを準備した後で表に追加した列をリストに指定してはなりません。

VALUES

挿入する新しい行を指定します。

この節の各変数は、変数宣言規則に従って宣言された変数を示していなければなりません。ホスト構造は使用できません。変数について詳しくは、 172 ページの『ホスト変数に対する参照』を参照してください。

expression

集約関数または列名を含まない、 196 ページの『式』で説明されているタイプの *expression*。

拡張標識変数が使用可能であり、式が以下の参照よりも複雑な場合には、拡張標識変数の値 DEFAULT (-5) または UNASSIGNED (-7) を使用してはなりません。

- ソース表の単一の列
- 単一のホスト変数
- 明示的にキャストされるホスト変数

DEFAULT

列にデフォルト値を割り当てるように指定します。DEFAULT は、デフォルト値のある列のみに対して指定できます。デフォルト値について詳しくは、 1238 ページの『CREATE TABLE』 の DEFAULT 節の説明を参照してください。

ユーザー指定の値をすべて無視して固有のシステム生成値を使用するよう指示する OVERRIDING USER VALUE が指定されていない場合、GENERATED ALWAYS として定義された列には DEFAULT を指定する必要があります。GENERATED BY DEFAULT として定義される列に関しては、有効な値を指定することができます。

NULL

列の値を NULL 値にすることを指定します。NULL は、NULL 可能列にのみ指定してください。

OVERRIDING SYSTEM VALUE または OVERRIDING USER VALUE

ROWID、識別、または行変更タイム・スタンプ列に、システムが生成した値やユーザーが指定した値を使用するかどうかを指定します。

OVERRIDING SYSTEM VALUE を指定する場合は、INSERT の暗黙的または明示的な列リストまたは UPDATE の SET 文節に、GENERATED ALWAYS として定義された ROWID 列、ID 列、または行変更タイム・ス

タンプ列が含まれていることが必要です。 OVERRIDING USER VALUE を指定する場合は、INSERT の暗黙的または明示的な列リストまたは UPDATE の SET 文節に、GENERATED ALWAYS または GENERATED BY DEFAULT として定義されている列を組み込む必要があります。

OVERRIDING SYSTEM VALUE
GENERATED ALWAYS として定義されている列について、VALUES または SET 文節に指定されている値を使用することを指定します。システム生成の値は使用されません。

行開始列、行終了列、トランザクション開始 ID 列、または生成式列の値が提供された場合、それは DEFAULT でなければなりません。

OVERRIDING USER VALUE
GENERATED ALWAYS または GENERATED BY DEFAULT として定義されている列について、VALUES または SET 文節に指定されている値を無視することを指定します。代わりにシステム生成の値が使用され、ユーザー指定の値はオーバーライドされます。

OVERRIDING SYSTEM VALUE と OVERRIDING USER VALUE のどちらも指定しない場合は、以下のようになります。

- GENERATED ALWAYS として定義された ROWID 列、ID 列、行変更タイム・スタンプ列、行開始列、行終了列、トランザクション開始 ID 列、および生成式列に、値を指定することはできません。
- GENERATED BY DEFAULT として定義された ROWID、識別、または行変更タイム・スタンプ列には、値を指定することができます。値を指定した場合は、この列にその値が割り当てられます。ただし、BY DEFAULT として定義された ROWID 列に値を割り当てることができるのは、指定された値が、Db2 for z/OS または Db2 for i によって既に生成されている有効な行 ID 値である場合のみです。BY DEFAULT として定義された識別または行変更タイム・スタンプ列に値を挿入または更新した場合は、その識別または行変更タイム・スタンプ列が固有制約または固有索引内の唯一のキーである場合以外は、データベース・マネージャーはその指定された値が該当の列についての固有な値であるかどうかを検査しません。固有制約も固有索引もない場合は、データベース・マネージャーは、NO CYCLE が有効である場合に限り、システム生成の値のセットの中でのみ各値の固有性を保証します。

  値が指定されていない場合は、データベース・マネージャーは新しい値を生成します。

signal-statement
matching-condition の評価が真になったときにエラーを返すために実行する SIGNAL ステートメントを指定します。 1740 ページの『SIGNAL』を参照してください。

ELSE IGNORE
USING *table-reference* の行ですべての WHEN 文節の *matching-condition* が偽になった場合に何の操作も実行しない、という動作を指定します。その場合は、ELSE IGNORE を指定したかどうかにかかわりなく、何の操作も実行されません。

ATOMIC または NOT ATOMIC
エラーを処理する方法を指定します。

ATOMIC
update-operation、*delete-operation*、または *insert-operation* の実行中にエラーが発生した場合に MERGE ステートメント全体をロールバックする、という動作を指定します。

NOT ATOMIC
update-operation、*delete-operation*、または *insert-operation* の実行中にエラーが発生した場合に、その *update-operation*、*delete-operation*、または *insert-operation* のみがロールバックされることを指定します。

STOP ON SQL EXCEPTION
update-operation、*delete-operation*、または *insert-operation* の実行中にエラーが発生した場合に MERGE ステートメントの処理を停止する、という動作を指定します。

CONTINUE ON SQL EXCEPTION
update-operation、*delete-operation*、または *insert-operation* の実行中にエラーが発生した場合に MERGE ステートメントの処理を続行する、という動作を指定します。

isolation-clause
このステートメントに関して使用する分離レベルを指定します。

WITH

分離レベルを指定します。次のいずれかになります。
- *RR* 反復可能読み取り
- *RS* 読み取り固定
- *CS* カーソル固定
- *UR* 非コミット読み取り
- *NC* コミットなし

ISOLATION 文節 を指定しなかった場合は、デフォルトの分離レベルが使用されます。デフォルトの判別方法については、 859 ページの『ISOLATION 文節』を参照してください。

concurrent-access-resolution-clause
SELECT ステートメントで使用する並行アクセスの解決方法を指定します。詳しくは、 861 ページの『concurrent-access-resolution-clause』を参照してください。

MERGE の規則

- 単一の MERGE ステートメントに、複数の *update-operation*、*delete-operation*、*insert-operation*、または *signal-statement* を指定できます。
- ターゲットの各行を操作できるのは、1 回に限られます。ターゲットの行は、*table-reference* の結果表の 1 つの行に対してのみ MATCHED として評価されます。ネストした SQL 操作 (INSTEAD OF トリガー以外のトリガーまたは RI) で、UPDATE、DELETE、INSERT、MERGE の各ステートメントのターゲットとして、ターゲット表 (または同じ階層にある表) を指定することはできません。

MERGE ステートメントの更新、挿入、削除の部分に当てはまる他の規則については、それぞれ対応するステートメントの『規則』のセクションにある説明を参照してください。

拡張標識変数の使用**:** 使用可能な場合は、正の値および 0 (ゼロ) から -7 以外の標識変数値を設定しないでください。 DEFAULT および UNASSIGNED 拡張標識変数値を、それらがサポートされていないコンテキストに指定しないでください。

拡張標識変数**:** MERGE ステートメントの挿入部分で、拡張標識変数値 UNASSIGNED は、列をデフォルト値に設定する効果があります。

MERGE 制約事項**:** MERGE ステートメントのターゲット表にトリガーがあるか、または、ターゲット表が参照整合性制約における親である場合、 *update-operation* または *insert-operation* に、グローバル変数、関数、または副選択が含まれていてはなりません。

注

処理の論理順序**:** NOT ATOMIC の MERGE ステートメントの場合は、各ソース行がそれぞれ別個に処理されます。つまり、各ソース行で別々の MERGE ステートメントが実行されるような動作になります。例えば、ターゲット行の更新を引き起こすソース行は、行の更新が実行されるときに、トリガー (ステートメント・レベルのトリガーを含む) を起動します。したがって、5 つの行が更新されるとすれば、UPDATE トリガー (ステートメント・レベルの UPDATE トリガーを含む) が 5 回起動されます。

ATOMIC の MERGE ステートメントの場合は、ソース行がまとめて処理されます。つまり、それぞれの WHEN 文節で 1 セットの行が処理されるような動作になります。例えば、5 つの行が更新されるとすれば、行レベルの UPDATE トリガーが 5 回起動されますが、ステートメント・レベルの UPDATE トリガーが起動されるのは、*n* 個になります (*n* は、UPDATE が含まれている WHEN 文節の数です。ソース行を処理しなかった UPDATE が含まれている WHEN 文節もその数に入ります)。ATOMIC の MERGE の処理の論理順序を以下にまとめます。

1. 1 つのセットとしてまとめて処理する行をソースとターゲットから判別します。このステートメントで特殊レジスターの CURRENT DATE、CURRENT TIME、CURRENT TIMESTAMP のいずれかを使用する場合は、ステートメント全体でクロックの読み取りが 1 回だけ発生します。
2. ON 文節を使用して、それらの行を MATCHED または NOT MATCHED のいずれかとして分類します。
3. WHEN 文節に含まれている *matching-condition* を評価します。
4. *assignment-clause* と *insert-operation* に含まれている式を評価します。
5. それぞれの *signal-statement* を実行します。
6. 指定された順序に従って、それぞれの *update-operation*、*delete-operation*、または *insert-operation* を該当する行に適用します。それぞれの *update-operation*、*delete-operation*、または *insert-operation* によってアクティブにされたトリガーが実行されます。ステートメント・レベルのトリガーは、*update-operation*、*delete-operation*、または *insert-operation* の基準を満たす行がない場合でもアクティブになります。それぞれの *update-operation*、*delete-operation*、または

insert-operation は、後続のそれぞれの *update-operation*、*delete-operation*、または *insert-operation* のトリガーに影響する可能性があります。

更新行の数**:** MERGE ステートメントの実行後、SQL 診断域の ROW_COUNT ステートメント情報項目 (または SQLCA の SQLERRD(3)) には、MERGE ステートメントによって操作された行の数が入ります (ただし、ELSE IGNORE 文節の対象になった行は除外されます)。ROW_COUNT 項目と SQLERRD(3) には、トリガーの結果として操作された行の数は含まれません。
DB2_ROW_COUNT_SECONDARY ステートメント情報項目 (または SQLCA の SQLERRD(5)) の値には、それらの行の数が含まれています。

ROW_COUNT および DB2_ROW_COUNT_SECONDARY についての説明は、1489 ページの『GET DIAGNOSTICS』を参照してください。SQLCA についての説明は、 1867 ページの『付録 C. SQLCA (SQL 連絡域)』を参照してください。

GET DIAGNOSTICS に関する注意点**:** MERGE ステートメントの完了時に 1 つ以上のエラーが発生している場合は、MERGE ステートメントの実行後に GET DIAGNOSTICS ステートメントを使用して、どの入力行でエラーが発生したのかを確認できます。GET DIAGNOSTICS のステートメント情報項目 NUMBER には、MERGE ステートメントの実行時に検出された条件 (エラーまたは警告) の数が入ります。さらに、GET DIAGNOSTICS の条件情報項目 DB2_ROW_NUMBER では、それぞれの条件ごとに、エラーの原因になった入力ソース行を確認できます。

挿入行の更新はできない**:** MERGE ステートメントの実行前に存在しなかったターゲット行 (つまり、MERGE ステートメントによって挿入された行) を更新することはできません。

MERGE のターゲット行の並行変更**:** MERGE 処理では、*update-operations*、*delete-operations*、または *insert-operations* の実行前に、MERGE で影響を受けるターゲット行が確認されます。制限の強い分離レベル (反復可能読み取りなど) を使用する場合を除き、影響を受ける一連のターゲット行が確認されてから、特定の行の *update-operation*、*delete-operation*、または *insert-operation* が処理されるまでの間に、並行処理によって MERGE のターゲット行が挿入/変更されることもあり得ます。そのような並行アクティビティーが発生すると、エラーになる可能性があります。例えば、MERGE 処理で、ソース行がターゲットに存在しない、という状況が確認されたとします。ターゲットでは、1 つの列の値にユニーク・キー制約がかかっています。MERGE がソース・データに基づいて新しい行を挿入しようとする前に、並行処理が同じキー値で行を挿入してしまう可能性があります。そのような状態で MERGE 処理がその行を挿入しようとすると、重複キー・エラーが発生します。

ロッキング**:** COMMIT(*RR)、COMMIT(*ALL)、COMMIT(*CS)、または COMMIT(*CHG) が指定されている場合は、正常に実行される MERGE ステートメントの実行中に、 1 つまたは複数の排他的ロックが掛けられます。そのようなロックがコミットまたはロールバック操作によって解放されるまで、挿入または更新された行は、以下によってのみアクセスすることができます。

- 挿入または更新を行ったアプリケーション・プロセス
- 読み取り専用操作を介して、COMMIT(*NONE) または COMMIT(*CHG) を使用する別のアプリケーション・プロセス

ロックは、他のアプリケーション・プロセスがその表の操作を行うのを防止します。ロッキングの詳細については、 1038 ページの『COMMIT』、 1661 ページの『ROLLBACK』、および 1580 ページの『LOCK TABLE』 ステートメントの説明を参照してください。また、 32 ページの『分離レベル』および「データベース・プログラミング」も参照してください。

COMMIT(*RR)、COMMIT(*ALL)、COMMIT(*CS)、または COMMIT(*CHG) を指定した場合は、1 つの MERGE ステートメントで最高 500 000 000 行を獲得することができます。行ロックの数には、MERGE のターゲットで挿入/更新/削除された行と、トリガーの結果として同じコミットメント定義で挿入/更新/削除された行が含まれます。さらに、COMMIT(*ALL) を指定した場合は、USING *table-reference* で参照されているソース行も行ロックの数に含まれます。

アクティブな行および列アクセス制御がある表**:** 使用可能な行の許可と列マスクが、MERGE ステートメント内の更新操作や挿入操作にどのように影響を及ぼすかについては、INSERT ステートメントと UPDATE ステートメントの情報を参照してください。

NOT ATOMIC の処理**:** NOT ATOMIC を指定した場合は、ソース・データ行が別々に処理されます。MERGE ステートメントで特殊レジスター (CURRENT TIMESTAMP など) を参照した場合は、ソース・データの各行が処理されるたびにその参照が評価されます。ステートメント・レベルのトリガーもソース・データの各行の処理時に活動化されます。

ソース・データの行の操作中にエラーが発生すると、そのエラーの時点で処理の対象になっていた行は、挿入/更新/削除されません。個々の行の処理はアトミック・オペレーションになります。MERGE ステートメントの処理で既に実行された他の変更は、ロールバックされません。CONTINUE ON EXCEPTION を指定した場合は、次の行に移って実行処理が続行されます。

システム期間テンポラル表**:** システム期間テンポラル表に関する MERGE ステートメントが処理される場合、行は特定のデータ変更操作が呼び出された場合と同様に影響を受けます。

例

例 1: アクティビティーの説明が変更されていた場合は、アーカイブ表でその説明を更新します。新しいアクティビティーがあれば、アーカイブ表にそのアクティビティーを挿入します。アーカイブ表でもアクティビティー表でも、アクティビティーが主キーになっています。

```
MERGE INTO archive ar
  USING (SELECT activity, description FROM activities) ac
  ON (ar.activity = ac.activity)
  WHEN MATCHED THEN
    UPDATE SET description = ac.description
  WHEN NOT MATCHED THEN
    INSERT (activity, description) VALUES(ac.activity, ac.description)
```

例 2: 出荷表を使用して、在庫表に行をマージします。行が一致した場合は、出荷表の部品数の分だけ数量を増やします。そうでない場合は、新しい *partno* を在庫表に挿入します。

```
MERGE INTO inventory AS in
  USING (SELECT partno, description, count FROM shipment
          WHERE shipment.partno IS NOT NULL) AS sh
  ON (in.partno = sh.partno)
  WHEN MATCHED THEN
    UPDATE SET description = sh.description,
               quantity = in.quantity + sh.count
  WHEN NOT MATCHED THEN
    INSERT (partno, description, quantity)
           VALUES (sh.partno, sh.description, sh.count)
```

例 *3:* 取引表を使用して、口座表に行をマージします。口座 ID に対する一連の取引に基づいて残高を更新し、存在しない口座については、取引の合計に基づいて新しい口座を挿入します。

```
MERGE INTO account AS a
    USING (SELECT id, SUM(amount) sum_amount FROM transaction
             GROUP BY id) AS t
    ON a.id = t.id
    WHEN MATCHED THEN
      UPDATE SET balance = a.balance + t.sum_amount
    WHEN NOT MATCHED THEN
      INSERT (id, balance) VALUES (t.id, t.sum_amount)
```

例 *4:* トランザクション・ログ表を使用して、従業員ファイル表に行を挿入します。トランザクションの時刻に基づいてトランザクション・ログ表の最新の行を確認し、その内容によって電話番号と支社の列を更新します。一致する行がない場合は、従業員ファイル表に新しい行を挿入します。

```
MERGE INTO employee_file AS e
  USING (SELECT empid, phone, office
              FROM (SELECT empid, phone, office,
                           ROW_NUMBER() OVER (PARTITION BY empid
                                              ORDER BY transaction_time DESC) rn
                    FROM transaction_log) AS nt
              WHERE rn = 1) AS t
  ON e.empid = t.empid
  WHEN MATCHED THEN
    UPDATE SET (phone, office) = (t.phone, t.office)
  WHEN NOT MATCHED THEN
    INSERT (empid, phone, office)
           VALUES(t.empid, t.phone, t.office)
```

例 *5:* 従業員行の動的作成値を使用して、データが既存の従業員に対応する場合はマスター従業員表を更新し、データが新しい従業員に対応する場合はその行を挿入します。以下に示すのは、C プログラムのコード断片です。

```
hv1 =
"MERGE INTO employee AS t
   USING (VALUES(CAST(? AS CHAR(6)), CAST(? AS VARCHAR(12)),
                      CAST(? AS CHAR(1)), CAST(? AS VARCHAR(15)),
                      CAST(? AS SMALLINT), CAST(? AS INTEGER)))
         s (empno, firstnme, midinit, lastname, edlevel, salary)
   ON t.empno = s.empno
   WHEN MATCHED THEN
     UPDATE SET salary = s.salary
   WHEN NOT MATCHED THEN
     INSERT (empno, firstnme, midinit, lastname, edlevel, salary)
            VALUES (s.empno, s.firstnme, s.midinit, s.lastname, s.edlevel,
                    s.salary)";
EXEC SQL PREPARE s1 FROM :hv1;
EXEC SQL EXECUTE s1 USING :hv2, :hv3, :hv4, :hv5, :hv6, :hv7;
```

例 *6:* グループ A が企画したアクティビティーのリストをアーカイブ表で更新します。期日が過ぎたアクティビティーは、すべて削除します。アクティビティーの情報 (説明と日付) に変更があれば、アーカイブ表の情報を更新します。今後の新しいアクティビティーがあれば、その情報をアーカイブ表に挿入します。アクティビティーのデータが不明な場合は、エラーを通知します。アーカイブ表では、アクティビティーの日付の指定が必須です。各グループにそれぞれのアクティビティー表があります。例えば、activities_groupA には、グループ A が企画したアクティビティーが含まれています。一方、アーカイブ表には、社内の各グループが企画した今後のすべてのアクティビティーが入っています。アーカイブ表では、グループとアクティビティーが主キーになっていて、データを NULL にすることはできません。すべてのアクティビティー表では、アクティビティーが主キーになっています。アーカイブ表の最終変更列では、CURRENT TIMESTAMP がデフォルト値として定義されています。

```
MERGE INTO archive ar
  USING (SELECT activity, description, date, last_modified
           FROM activities_groupA) ac
  ON (ar.activity = ac.activity) AND ar.group = 'A'
  WHEN MATCHED AND ac.date IS NULL THEN
    SIGNAL SQLSTATE '70001'
           SET MESSAGE_TEXT = 'Activity cannot be modified.  Reason: date is not known'
  WHEN MATCHED and ac.date < CURRENT DATE THEN
    DELETE
  WHEN MATCHED AND as.last_modified < ac.last_modified THEN
    UPDATE SET (description, date, last_modified)
               = (ac.description, ac.date, DEFAULT)
  WHEN NOT MATCHED AND ac.date IS NULL THEN
    SIGNAL SQLSTATE '70002'
           SET MESSAGE_TEXT = 'Activity cannot be inserted.  Reason: date is not known'

  WHEN NOT MATCHED AND ac.date >= CURRENT DATE THEN
    INSERT (group, activity, description, date)
           VALUES ('A', ac.activity, ac.description, ac.date)
  ELSE IGNORE
```

OPEN

OPEN ステートメントは、カーソルをオープンして、カーソルの結果表から行を取り出せるようにします。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。これは実行可能ステートメントですが、動的に準備することはできません。Java では指定できません。

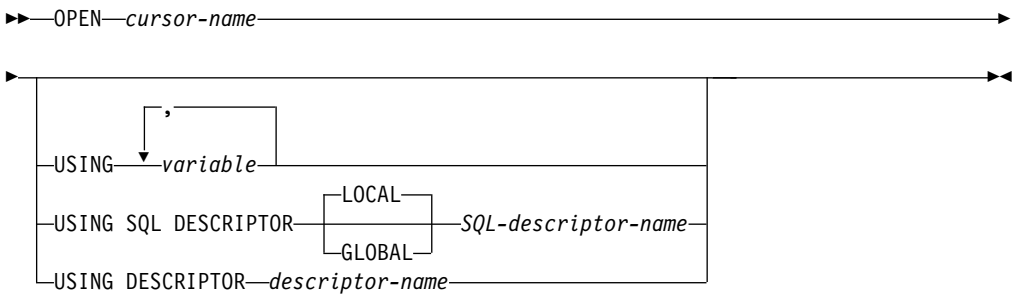
権限

ステートメントでグローバル変数を参照する場合は、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別されるグローバル変数に対して、
  - そのグローバル変数に対する READ 特権
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

カーソルの使用に必要な権限については、 1353 ページの『DECLARE CURSOR』を参照してください。

構文



説明

cursor-name

オープンするカーソルを識別します。この *cursor-name* は、宣言されているカーソルを識別しなければなりません。カーソルの宣言については、DECLARE CURSOR ステートメントの Notes の項を参照してください。このカーソルは、OPEN ステートメントを実行するときには、クローズ状態になければなりません。

カーソルに関連付けられる SELECT ステートメントは、以下のいずれかです。

- DECLARE CURSOR ステートメントで指定した選択ステートメント、または
- DECLARE CURSOR ステートメントで指定したステートメント名 によって識別される準備済み選択ステートメント。このステートメントが正しく準備されていない場合や、選択ステートメント でない場合は、カーソルを正常にオープンすることはできません。

カーソルの対象となる結果表は、SELECT ステートメントを評価することによって得られます。 SELECT ステートメントを評価するときには、SELECT ステートメントに特殊レジスターが指定されていれば、その特殊レジスターの現行値が使用され、OPEN ステートメントの USING 文節または SELECT ステートメントに変数が指定されていれば、その変数の現行値が使用されます。結果表の行は、OPEN ステートメントの実行時に取得され、一時表に保持される場合と、後続の FETCH ステートメントの実行時に取得される場合があります。どちらの場合も、カーソルはオープン状態になり、結果表の最初の行の前に位置付けられます。ただし、表が空であれば、実際のカーソルの位置は「最終行の後」になります。

USING

この後に変数のリストを指定します。準備済みステートメントのパラメーター・マーカー (疑問符) は、ここに指定した変数の値によって置き換えられます。パラメーター・マーカーの説明については、 1603 ページの『PREPARE』を参照してください。

- DECLARE CURSOR ステートメントでパラメーター・マーカーを組み込んだ *statement-name* を指定していた場合は、USING を使用する必要があります。準備済みステートメントにパラメーター・マーカーが入っていない場合は、USING は無視されます。
- DECLARE CURSOR ステートメントで *select-statement* を指定していた場合は、USING を使用して変数値をオーバーライドできます。詳細については、変数値のオーバーライドを参照してください。

DECLARE CURSOR ステートメントでパラメーター・マーカーの入った準備済みステートメントを指定している場合は、USING を使用する必要があります。準備済みステートメントにパラメーター・マーカーが入っていない場合は、USING は無視されます。

variable,...

ホスト構造体または変数を指定します。指定するホスト構造体または変数はそれらの宣言の規則に従ってプログラムで宣言されていなければなりません。ホスト構造体に対する参照は、その個々の変数に対する参照に置き換えられます。変数の数は、準備済みステートメントのパラメーター・マーカーの数と同じでなければなりません。*n* 番目の変数は、準備済みステートメントの *n* 番目のパラメーター・マーカーに対応します。

現在の接続がローカル接続である (DRDA 接続ではない) 場合のみ、グローバル変数が使用できます。

USING SQL DESCRIPTOR *SQL-descriptor-name*

SQL 記述子を識別します。

LOCAL

記述子の名前の有効範囲はプログラム呼び出しのローカルであることを指定します。

GLOBAL

記述子の名前の有効範囲は SQL セッション全体であることを指定します。

SQL-descriptor-name
SQL 記述子の名前を指定します。名前は、指定の有効範囲を持つ既存の記述子を識別するものでなければなりません。

SQL 記述子内の情報の説明については、 1688 ページの『SET DESCRIPTOR』を参照してください。

USING DESCRIPTOR *descriptor-name*
SQLDA を識別します。この SQLDA には、入力変数の有効な記述が入っていなければなりません。

OPEN ステートメントを処理する前に、ユーザーは SQLDA の以下のフィールドをセットしておく必要があります。(REXX の場合は、規則が異なります。詳しくは、組み込み SQL プログラミングを参照してください。)

- SQLN (SQLDA に用意する SQLVAR のオカレンスの数を示します。)
- SQLDABC (SQLDA 用に割り振る記憶域のバイト数を示します。)
- SQLD (ステートメントを処理するときに、SQLDA で使用する変数の個数を指示します。)
- SQLVAR の各オカレンス (変数の属性を指示します。)

SQLDA の記憶域は、SQLVAR のオカレンスをすべて収容するのに十分な大きさで割り振らなければなりません。 LOB または特殊タイプが結果の中に存在する場合、各パラメーター・マーカーごとに追加の SQLVAR 項目が必要です。 SQLVAR の説明、SQLVAR オカレンスの回数を判別する方法など、SQLDA について詳しくは、 1877 ページの『付録 D. SQLDA (SQL 記述子域)』を参照してください。

SQLD には、ゼロ以上で SQLN 以下の値をセットしなければなりません。この値は、準備済みステートメント内のパラメーター・マーカーの個数と同じでなければなりません。 SQLDA によって *n* 番目に記述される変数は、準備済みステートメントの *n* 番目のパラメーター・マーカーに対応します。

RPG/400 はポインターを設定する機能を用意しておらず、SQLDA はポインターを使用して、適切な変数を見つけるため、ユーザーは、RPG/400 アプリケーションの外側でポインターを設定しなければならないことに注意する必要があります。

注

クローズ状態のカーソル**:** 以下の時点では、プログラム内のすべてのカーソルはクローズ状態にあります。

- プログラムが呼び出されたとき。
  - CLOSQLCSR(*ENDPGM) が指定されている場合、プログラムが呼び出されるたびに、すべてのカーソルがクローズ状態になります。
  - CLOSQLCSR(*ENDSQL) が指定されている場合、1 つの SQL プログラムが呼び出しスタックに残っている間は、プログラムが初めて呼び出されるときに限って、すべてのカーソルがクローズ状態になります。

  - CLOSQLCSR(*ENDJOB) が指定されている場合は、ジョブが活動状態である限りは、プログラムが初めて呼び出された時に限って、すべてのカーソルがクローズ状態になります。
  - CLOSQLCSR(*ENDMOD) が指定されている場合、モジュールが開始されるたびに、すべてのカーソルがクローズ状態になります。
  - CLOSQLCSR(*ENDACTGRP) が指定されている場合は、プログラム内のモジュールが活動化グループ内で最初に開始された時に限って、すべてのカーソルがクローズ状態になります。
- プログラムは、HOLD オプションの指定がない COMMIT ステートメントを実行して、新規の作業単位を開始します。 HOLD オプションを指定して宣言されたカーソルは、HOLD オプションの指定がない COMMIT ステートメントではクローズされません。 COMMIT HOLD ステートメントは、HOLD オプションで宣言されているかどうかに関わらず、カーソルをクローズしません。
- プログラムは、HOLD オプションの指定がない ROLLBACK ステートメントを実行して、新規の作業単位を開始します。 ROLLBACK HOLD ステートメントは、HOLD オプションで宣言されているかどうかに関わらず、カーソルをクローズしません。
- CONNECT (タイプ 1) ステートメントが実行されたとき。

また、次の場合に、カーソルがクローズ状態になることもあります。
- CLOSE ステートメントが実行されたとき。
- DISCONNECT ステートメントによって、そのカーソルが関連する接続が切り離されたとき。
- そのカーソルが関連した接続が解除保留状態にあり、正常な COMMIT が行われたとき。
- CONNECT (タイプ 1) ステートメントが実行されたとき。

カーソルの結果表から行を検索するには、カーソルがオープン状態であるときに FETCH ステートメントを実行しなければなりません。クローズ状態のカーソルをオープン状態に変更する方法は、OPEN ステートメントを実行する以外にはありません。

一時表の影響**:** カーソルの結果表が読み取り専用でなければ、その結果表の行は後続の FETCH ステートメントを実行したときに取得されます。これと同じ方式は、読み取り専用の結果表にも使用されます。ただし、結果表が読み取り専用である場合は、Db2 for i がこの方式に代えて一時表方式の使用を選択することがあります。一時表を使用する方式では、OPEN ステートメントの実行時に、結果表全体が一時表に挿入されます。一時表を使用した場合は、プログラムの結果が以下のいくつかの点で異なります。
- 通常は、後続の FETCH ステートメントが実行されるまでは発生しないエラーが、 OPEN ステートメントの実行時に発生する可能性がある。
- カーソルがオープンされている時に INSERT、UPDATE、および DELETE ステートメントを実行すると、結果表に影響を与えない可能性がある。
- SELECT ステートメント内にある任意の NEXT VALUE 式は、 OPEN 中に結果表のすべての行に対して評価されます。そのため、OPEN 状態のときに結果表のすべての行に対してシーケンス値が生成されます。

- 任意の関数は、OPEN 中に結果表のすべての行に対して評価されます。そのため、関数内にある SQL データを変更する外部アクションおよび SQL ステートメントは、 OPEN 状態のときに結果表のすべての行に対して実行されます。

逆に、一時表を使用しない場合は、カーソルがオープンされている間に INSERT、UPDATE、および DELETE ステートメントを実行すると、結果表に影響を与える可能性があり、SELECT ステートメント内のすべての NEXT VALUE 式および関数は、各行が取り出される際に評価されます。このような操作の影響は、常に予測できるとは限りません。例えば、SELECT * FROM T として定義されている結果表の行にカーソル CUR が位置付けられているときに、 T に対して行を挿入した場合は、その行の順序が定まっていないことから、その挿入が結果表に与える影響は予測できないものになります。後続の FETCH CUR で、T の新しい行が取り出されることも、取り出されないこともあります。

パラメーター・マーカーの置換**:** ステートメント内にある各パラメーター・マーカーは、実際には、カーソルに関連する SELECT ステートメントが評価されるときに、対応する変数の値に置き換えられます。パラメーター・マーカーの置き換えは、変数の値をソースとし、データベース・マネージャー内部の変数をターゲットとする割り当て演算によって処理されます。タイプ付きパラメーター・マーカーの場合、ターゲット変数の属性は、CAST によって指定されたものになります。タイプ無しパラメーター・マーカーの場合、ターゲット変数の属性は、パラメーター・マーカーのコンテキストによって決まります。パラメーター・マーカーに影響を及ぼす規則については、 1612 ページの表 113を参照してください。

V が、パラメーター・マーカー P に対応する変数を指すものとします。V の値は、値を列に割り当てる場合の規則に従って、P のターゲット変数に割り当てられます。したがって、次のことがいえます。

- V は、ターゲットと互換性のあるものでなければなりません。
- V が数値ならば、V の整数部の絶対値は、ターゲットの整数部の絶対値の最大を超えてはなりません。
- V の属性がターゲットの属性と一致しない場合は、ターゲットの属性に合わせて値が変換されます。
- ターゲットに NULL を入れることができない場合は、V の値は NULL であってはなりません。

ただし、値を列に割り当てる場合の規則とは、以下の点が異なります。

- V がストリングで、その長さがターゲットの長さ属性より大きければ、V の値は途中で切り捨てられます (エラーは出されません)。

カーソルの SELECT ステートメントが評価されるときに、P の代わりに使用される値は、P のターゲット変数の値です。例えば、V が CHAR(6) で、ターゲットが CHAR(8) の場合は、P の代わりに使用される値は、 V の値に 2 つのブランクが埋め込まれた値になります。

USING 文節は、パラメーター・マーカーが入っている準備済み SELECT ステートメントを対象としたものです。しかし、カーソルに関連する SELECT ステートメントが、DECLARE CURSOR ステートメントの一部として入っているときにも、USING 文節を使用することができます。この場合、OPEN ステートメントは、SELECT ステートメント内の変数の属性がターゲット変数の属性と同じであること

を除けば、SELECT ステートメント内のそれぞれの変数がパラメーター・マーカーである場合と同じように実行されます。このため、カーソルに関連する SELECT ステートメントにある変数の値は、 USING 文節内で指定された変数の値に変更されることになります。

変数値のオーバーライド: USING 文節の主な対象になっているのは、パラメーター・マーカーが入っている準備済み SELECT ステートメントです。しかし、カーソルに関連する SELECT ステートメントが、DECLARE CURSOR ステートメントの一部として入っているときにも、 USING 文節を使用することができます。その場合、OPEN ステートメントは、SELECT ステートメント内の変数の属性がターゲット変数の属性と同じであることを除けば、SELECT ステートメント内のそれぞれの変数がパラメーター・マーカーである場合と同じように実行されます。このため、カーソルに関連する SELECT ステートメントにある変数の値は、 USING 文節内で指定された変数の値に変更されることになります。

例

例 1: COBOL プログラム内に、以下のような処理を行う組み込みステートメントを書き込みます。

1. カーソル C1 を定義します。このカーソルは、ADMRDEPT 部門「A00」によって管理されている各部門の DEPARTMENT 表からすべての行を取り出すためのカーソルです。
2. 最初に取り出す行の前に、カーソル C1 を位置付けます。

```
EXEC SQL  DECLARE C1 CURSOR FOR
           SELECT DEPTNO, DEPTNAME, MGRNO FROM DEPARTMENT
              WHERE ADMRDEPT = 'A00'  END-EXEC.

EXEC SQL  OPEN C1  END-EXEC.
```

例 2: C プログラムに OPEN ステートメントをコーディングして、カーソル DYN_CURSOR を、動的に定義される選択ステートメント に関連付けます。準備済み選択ステートメント の選択リストには、既に 2 つの項目が定義されていているものとします。最初の項目のデータ・タイプは整数で、2 番目の項目のデータ・タイプは VARCHAR(64) です。 (以下の例には、関連するホスト変数の定義、PREPARE ステートメント、および DECLARE CURSOR ステートメントも示しています。)

```
EXEC SQ  BEGIN DECLARE SECTION;
   static short hv_int;
   char hv_vchar64[64];
   char stmt1_str[200];
EXEC SQL  END DECLARE SECTION;

EXEC SQL  PREPARE STMT1_NAME FROM :stmt1_str;

EXEC SQL  DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

EXEC SQL  OPEN DYN_CURSOR USING :hv_int, :hv_vchar64;
```

例 3: 例 2 と同じような OPEN ステートメントをコーディングします。ただし、この例では、選択ステートメント内の項目の数とデータ・タイプは分かっていません。

```
EXEC SQL  BEGIN DECLARE SECTION;
   char stmt1_str[200];
EXEC SQL  END DECLARE SECTION;
EXEC SQL  INCLUDE SQLDA;

EXEC SQL  PREPARE STMT1_NAME FROM :stmt1_str;
EXEC SQL  DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

EXEC SQL  OPEN DYN_CURSOR USING DESCRIPTOR :sqlda;
```

PREPARE

PREPARE ステートメントは、文字ストリング形式のステートメントから実行可能な形式の SQL ステートメントを作成します。このような文字ストリング形式は、ステートメント・ストリング と呼ばれ、実行可能な形式は、準備済みステートメント と呼ばれます。

呼び出し

このステートメントは、アプリケーション・プログラム、SQL 関数、 SQL プロシージャー、またはトリガー内にのみ組み込むことができます。これは実行可能ステートメントですが、動的に準備することはできません。Java では指定できません。

権限

権限の規則は、その PREPARE ステートメントに指定された SQL ステートメントに対して定義されている規則と同じです。例えば、SELECT ステートメントを準備する場合に適用される権限規則については、 847 ページの『選択ステートメント』を参照してください。

CRTSQLxxx コマンドに DLYPRP(*NO) が指定されていると、以下の場合を除き、権限の検査は該当のステートメントが準備される時点で行われます。

- DROP SCHEMA ステートメントが準備済みの場合、スキーマの中の全オブジェクトに対する特権は、そのステートメントが実行されるまで検査されません。
- DROP TABLE ステートメントが準備済みの場合、該当の表を参照するすべてのビュー、索引、および論理ファイルに対する特権は、そのステートメントが実行されるまで検査されません。
- DROP VIEW ステートメントが準備済みの場合、そのビューを参照するすべてのビューに対する特権は、そのステートメントが実行されるまで検査されません。
- CREATE TRIGGER ステートメントが準備済みの場合、*triggered-action* で参照されるオブジェクトに対する特権は、ステートメントが実行されるまで検査されません。
- FUNCTION、PROCEDURE、SEQUENCE、TYPE、TRIGGER、VARIABLE、または XSROBJECT ステートメントの DROP、COMMENT、または LABEL が準備済みの場合、権限は、そのステートメントが実行されるまで検査されません。
- GRANT または REVOKE ステートメントが準備済みの場合、権限は、そのステートメントが実行されるまで検査されません。

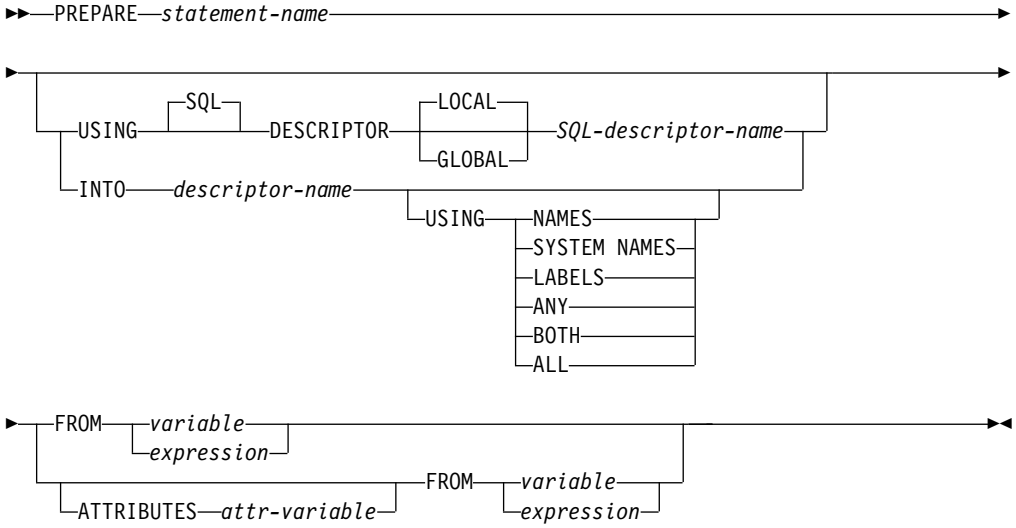
CRTSQLxxx コマンドに DLYPRP(*YES) が指定されている場合、該当のステートメントが実行されるか、または OPEN ステートメントで使用されるまで、権限の検査はすべて据え置かれます。

ステートメントの権限 ID は、プログラムが作成されたときに CRTSQLxxx コマンドに USRPRF(*OWNER) および DYNUSRPRF(*OWNER) が指定された場合を除いて、実行時の権限 ID です。詳しくは、 78 ページの『権限 ID と権限名』を参照してください。

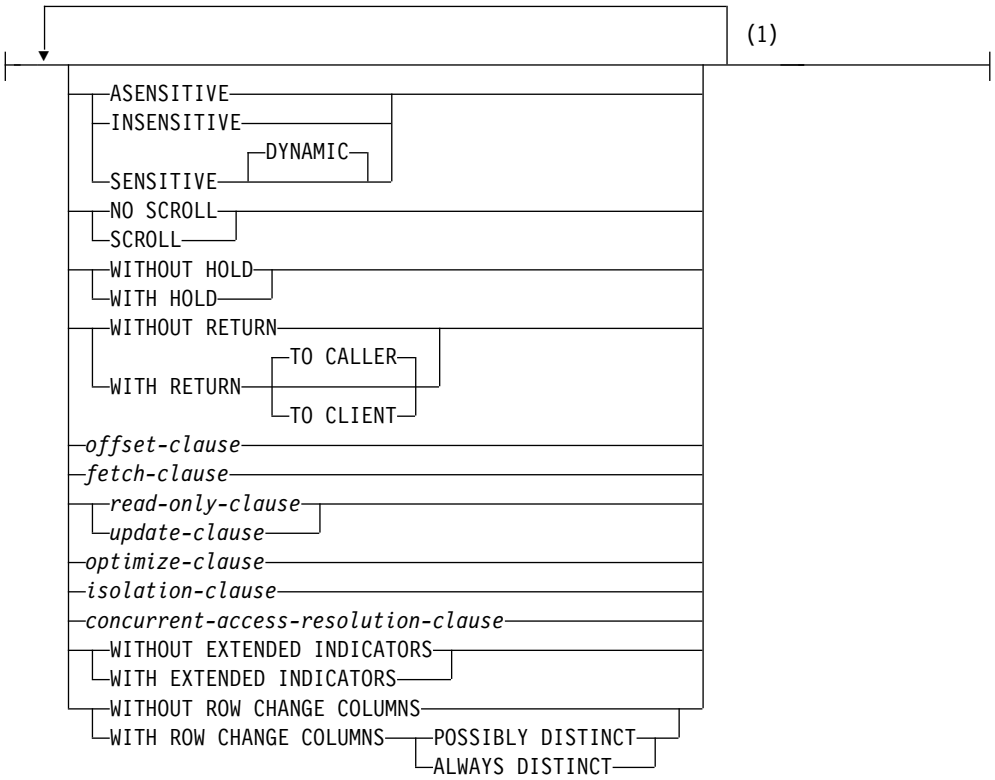
ステートメントでグローバル変数を参照する場合は、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別されるグローバル変数に対して、
  - そのグローバル変数に対する READ 特権
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

構文



attribute-string:



注:

1 同じ文節を複数回指定することはできません。オプションが指定されていない場合には、関連付けられた DECLARE CURSOR ステートメントおよび準備済み SELECT ステートメントのオプションに指定されているものがそのデフォルトとなります。

説明

statement-name

準備済みステートメントの名前を指定します。この名前に、既存の準備済みステートメントを指定すると、その準備済みステートメントは次の場合に破棄されます。

- そのステートメントが同じプログラムの同じインスタンス内で準備された場合。
- 両方のステートメントに関連した CRTSQLxxx コマンドに CLOSQLCSR(*ENDJOB)、 CLOSQLCSR(*ENDACTGRP)、または CLOSQLCSR(*ENDSQL) が指定されている場合。

この名前に、プログラムの同じインスタンスの中のオープン・カーソルに関連する SELECT ステートメントを指定してはなりません。

USING SQL DESCRIPTOR *SQL-descriptor-name*

SQL 記述子を識別します。USING が指定されると、PREPARE ステートメン

トが正常に実行されたときに、準備済みステートメントに関する情報が、*SQL 記述子名* で指定した SQL 記述子内に入ります。したがって、次の PREPARE ステートメントは、

```
EXEC SQL PREPARE S1 USING SQL DESCRIPTOR :sqldescriptor FROM :V1;
```

上記のステートメントは、次のステートメントと同等です。

```
EXEC SQL PREPARE S1 FROM :V1;
EXEC SQL DESCRIBE S1 USING SQL DESCRIPTOR :sqldescriptor;
```

LOCAL
記述子の名前の有効範囲はプログラム呼び出しのローカルであることを指定します。

GLOBAL
記述子の名前の有効範囲は SQL セッション全体であることを指定します。

SQL-descriptor-name
SQL 記述子の名前を指定します。名前は、指定の有効範囲を持つ既存の記述子を識別するものでなければなりません。

SQL 記述子に入る情報の説明については、 1476 ページの『GET DESCRIPTOR』を参照してください。

INTO
INTO を使用すると、PREPARE ステートメントが正常に実行されたときに、準備済みステートメントに関する情報が、記述子名 で指定した SQLDA 内に入ります。したがって、次の PREPARE ステートメントは、

```
EXEC SQL PREPARE S1 INTO :SQLDA FROM :V1;
```

上記のステートメントは、次のステートメントと同等です。

```
EXEC SQL PREPARE S1 FROM :V1;
EXEC SQL DESCRIBE S1 INTO :SQLDA;
```

descriptor-name
SQL 記述子域 (SQLDA) を指定します。これについては、 1877 ページの『付録 D. SQLDA (SQL 記述子域)』で説明しています。PREPARE ステートメントを実行する前に、SQLDA に次の変数を設定しておく必要があります。(REXX の場合の規則は異なります。詳しくは、「組み込み SQL プログラミング」トピック集を参照してください。):

SQLN
SQLVAR によって表される変数の個数を示します。 (SQLN によって、SQLVAR 配列の大きさ (エレメント数) が指定されます。) SQLN は PREPARE ステートメントの実行に先立ってゼロよりも大きいか、または等しい値に設定しなければなりません。必要なオカレンスの数を決定する手法については、 1880 ページの『必要な SQLVAR オカレンスの数の決定』を参照してください。

SQLDA に入れられる情報の説明については、 1413 ページの『DESCRIBE』を参照してください。

USING
SQLDA のそれぞれの SQLNAME 変数に、どのような値を割り当てるかを指

定します。要求した値が存在しない場合または名前が 30 より長い場合、SQLNAME の長さは 0 にセットされます。

NAMES
列の名前を割り当てます。これはデフォルトです。準備されたステートメントで名前がその選択リストに明示的に指定されている場合、指定されたそれらの名前が戻されます。

SYSTEM NAMES
列のシステム列名を割り当てます。

LABELS
列のラベルを割り当てます。(列のラベルは、LABEL ステートメントによって定義されます。) ラベルの最初の 20 バイトだけが戻されます。

ANY
列のラベルを割り当てます。列がラベルを持たない場合、ラベルとして列名が使用されます。

BOTH
列のラベルと名前の両方を割り当てます。この場合、追加情報に応じるために、1 つの列ごとに SQLVAR の 2 つから 3 つのオカレンスが必要になりますが、その数は、結果セットに特殊タイプが入っているか否かによって決まります。この拡張の SQLVAR 配列を指定するには、SQLN を 2*n* か 3*n* (この場合の *n* は、表やビュー内の列数) に設定します。 SQLVAR の最初の *n* 個のオカレンスには、列の名前が入り、 2 番目または 3 番目の *n* オカレンスには、列のラベルが含まれます。 特殊タイプがない場合、SQLVAR 項目の 2 番目のセットにそのラベルが戻されます。それ以外の場合、ラベルは、SQLVAR 項目の 3 番目のセット内に戻されます。

同じ SQLDA を以後の FETCH ステートメントで使用する場合には、その PREPARE が完了したあと、SQLN を *n* に設定してください。

ALL
ラベル、列名、およびシステム列名を割り当てます。この場合、追加情報に応じるために、 1 つの列ごとに SQLVAR の 3 つから 4 つのオカレンスが必要になりますが、その数は、結果セットに特殊タイプが入っているか否かによって決まります。この拡張の SQLVAR 配列を指定するには、SQLN を 3*n* か 4*n* (この場合の *n* は、結果表内の列数) に設定します。
SQLVAR の最初の *n* オカレンスには、システム列名が入ります。 2 番目または 3 番目の *n* オカレンスには、列のラベルが含まれます。列名がシステム列名とは異なる場合、列名は 3 番目または 4 番目の *n* オカレンスに含まれます。特殊タイプが指定されていない場合、ラベルは、SQLVAR 記入項目の 2 番目のセット内に戻され、列名は、SQLVAR 記入項目の 3 番目のセット内に戻されます。それ以外の場合、ラベルは、SQLVAR 記入項目の 3 番目のセット内に戻され、列名は、 SQLVAR 記入項目の 4 番目のセット内に戻されます。

同じ SQLDA を以後の FETCH ステートメントで使用する場合には、その PREPARE が完了したあと、SQLN を *n* に設定してください。

ATTRIBUTES *attr-variable*
対応する属性が、関連付けられた SELECT ステートメントの最外部の全選択の一部として指定されていない場合に、このカーソルに有効な属性を指定します。

属性が最外部の全選択に指定されている場合、それらは PREPARE ステートメント上で指定された対応する属性の代わりに使用されます。逆に、属性が PREPARE ステートメントに指定されている場合、DECLARE CURSOR ステートメントに指定された対応する属性の代わりにそれらの属性が使用されます。

作成したステートメントが *select-statement* ではない場合、USE CURRENTLY COMMITTED および WAIT FOR OUTCOME 以外のすべての属性は無視されます。

attr-variable は、ストリング変数の宣言に関する規則に従ってプログラム内で宣言される、文字ストリングまたは Unicode グラフィック変数を識別する必要があります。*attr-variable* は、長さ属性が VARCHAR の最大長を超えないストリング変数 (固定長または可変長のいずれか) でなければなりません。先頭ブランクおよび末尾ブランクは、変数の値から除去されます。変数には、有効な *attribute-string* が含まれている必要があります。

標識変数を使用して、属性が PREPARE ステートメント上に実際に指定されているかどうかを示すことができます。このようにして、属性を指定する必要があるかどうかには関係なく、アプリケーションは同じ PREPARE ステートメントを使用できます。 *attribute-string* の一部として指定できるオプションは以下のとおりです。

ASENSITIVE、SENSITIVE、または INSENSITIVE
カーソルが変更に対して反応を決めない、反応する、または反応しないことを指定します。詳しくは、 1353 ページの『DECLARE CURSOR』を参照してください。

SENSITIVE を指定した場合、*fetch-clause* は指定できません。INSENSITIVE を指定した場合、*update-clause* は指定できません。

NO SCROLL または SCROLL
カーソルがスクロール可能かどうかを指定します。詳しくは、 1353 ページの『DECLARE CURSOR』を参照してください。

WITHOUT HOLD または WITH HOLD
コミット操作の結果として、カーソルがクローズされるのを防止するかどうかを指定します。詳しくは、 1353 ページの『DECLARE CURSOR』を参照してください。

WITHOUT RETURN または WITH RETURN
カーソルの結果表をプロシージャーから戻される結果セットとして使用するかどうか指定します。詳しくは、 1353 ページの『DECLARE CURSOR』を参照してください。

offset-clause
行が取得される前にスキップする行の数を指定します。詳しくは、 836 ページの『offset-clause』を参照してください。 *offset-row-count* は定数でなければなりません。

fetch-clause
最大数の行を検索するように指定します。詳しくは、 837 ページの『fetch-clause』を参照してください。 *fetch-row-count* は定数でなければなりません。代替の LIMIT 構文はサポートされていません。

read-only-clause または *update-clause*
結果表が読み取り専用であるか更新可能であるかを指定します。
update-clause は、列名なしで指定する必要があります (FOR UPDATE)。詳しくは、 857 ページの『READ-ONLY 文節』、および 856 ページの『UPDATE 文節』を参照してください。

optimize-clause
データベース・マネージャーが、プログラムが整数 で指定された行数を超えて結果表から検索を行う意図はないことを想定するように指定します。詳しくは、 858 ページの『OPTIMIZE 文節』を参照してください。

isolation-clause
SELECT ステートメントを実行する分離レベルを指定します。詳しくは、859 ページの『ISOLATION 文節』を参照してください。

concurrent-access-resolution-clause
SELECT ステートメントで使用する並行アクセスの解決方法を指定します。詳しくは、 861 ページの『concurrent-access-resolution-clause』を参照してください。

WITHOUT EXTENDED INDICATORS または **WITH EXTENDED INDICATORS**
INSERT または UPDATE の実行中に標識変数に指定された値が NULL 値を示すための標準 SQL セマンティクスに従っているかどうか、または拡張機能を使用して DEFAULT または UNASSIGNED 値の割り当てを示すことができるかどうかを指定します。

WITH EXTENDED INDICATORS を指定する必要があるのは、このステートメントが INSERT ステートメントの VALUES 形式を使用する INSERT である場合、UPDATE ステートメントである場合、またはこのステートメントに INSERT ステートメントの VALUES 形式を使用する INSERT が含まれる場合のみです。

WITHOUT ROW CHANGE COLUMNS、**WITH ROW CHANGE COLUMNS POSSIBLY DISTINCT** 、または **WITH ROW CHANGE COLUMNS ALWAYS DISTINCT**
準備された選択ステートメント の結果セットに列を追加するかどうかを指定します。この列は後で、その行の列の値が変更された可能性があるかどうかを識別するために使用できます。この行変更列は、単一の表 (または更新可能ビュー) が最外部の副選択で参照される場合にのみ追加されます。
DESCRIBE および GET DESCRIPTOR ステートメントは、どの行が追加されたかを示します。

WITHOUT ROW CHANGE COLUMNS
行変更列は、結果セットには追加されません。これはデフォルトです。

WITH ROW CHANGE COLUMNS POSSIBLY DISTINCT
行変更列は、単一の行を一意的に表していない場合でも結果セットに追加されます。追加された列を使用して、その行の列の値が最初に取り出された後に変更された可能性があるかどうかを判別することができます。

- 行変更列の値が最初の取り出し以降に変更されていない場合は、その行のどの列も最初の取り出し以降に変更されていません。

- 行変更列の値が最初の取り出し以降に変更された場合は、行変更値が単一の行を表しているかどうかが確実ではないため、その行の列は変更された可能性も、変更されていない可能性もあります。

WITH ROW CHANGE COLUMNS ALWAYS DISTINCT
行変更列は、単一の行を一意的に表している場合にのみ結果セットに追加されます。そうでない場合、行変更列は結果セットに追加されません。追加された列を使用して、その行の列の値が最初の取り出し以降に変更されたかどうかを判別することができます。 (行の行変更列が確実に単一の行を一意的に識別するには、表に行変更タイム・スタンプ列が必要となることに注意してください。)

- 行変更列の値が最初の取り出し以降に変更されていない場合は、その行のどの列も最初の取り出し以降に変更されていません。
- 行変更列の値が最初の取り出し以降に変更された場合、その行の列は変更されています。

WITH ROW CHANGE COLUMNS ALWAYS DISTINCT が指定され、データベース・マネージャーが特殊な行変更列を返すことができない場合は、警告が返されます (SQLSTATE 0168T)。

FROM
ステートメント・ストリングを指定します。ステートメント・ストリングは、指定される *expression*、*string-expression*、または *variable* の値です。

variable
variable を識別します。この変数は、文字ストリングまたは Unicode グラフィック変数の宣言に関する規則に従ってプログラム内で宣言されます。この変数に、標識変数を指定してはなりません。

expression
集約関数または列名を含まない、 196 ページの『式』で説明されているタイプの *expression*。これは、文字ストリングまたは Unicode グラフィック・ストリングの値を戻す必要があります。*expression* の中に変数が指定される場合、その変数の CCSID は 65535 であってはなりません。[118]

ステートメント・ストリングは、以下の SQL ステートメントのいずれかである必要があります。

ALLOCATE CURSOR	INSERT	SET CURRENT DEGREE
ALTER	LABEL	SET CURRENT IMPLICIT XMLPARSE OPTION
ASSOCIATE LOCATORS	LOCK TABLE	SET CURRENT TEMPORAL SYSTEM_TIME
CALL	MERGE	SET ENCRYPTION PASSWORD
COMMENT	REFRESH TABLE	SET PATH
COMMIT	RELEASE SAVEPOINT	SET SCHEMA
コンパウンド (動的)	RENAME	SET SESSION AUTHORIZATION
CREATE	REVOKE	SET TRANSACTION

118. PL/I プログラム内では、PL/I ストリング式も指定できます。

119. SET 変数ステートメントのターゲットはグローバル変数である必要があります。

DECLARE GLOBAL TEMPORARY TABLE	ROLLBACK	SET 変数[119]
DELETE	SAVEPOINT	TRANSFER OWNERSHIP
DROP	*select-statement*	TRUNCATE
FREE LOCATOR	SET CURRENT DEBUG MODE	UPDATE
GRANT	SET CURRENT DECFLOAT ROUNDING MODE	VALUES INTO
HOLD LOCATOR		

ステートメント・ストリングは、次のようなストリングであってはなりません。

- EXEC SQL で始める。
- END-EXEC またはセミコロンで終了する。
- 変数への参照を含むストリング。グローバル変数を使用できます。

注

パラメーター・マーカー: ステートメント・ストリングにホスト変数、SQL 変数、および SQL パラメーターへの参照を組み込むことはできませんが、パラメーター・マーカー またはグローバル変数を組み込むことはできます。パラメーター・マーカーは、準備されたステートメントの実行時点で、変数の値によって置き換えられます。パラメーター・マーカーは疑問符 (?) で表し、そのステートメント・ストリングが静的 SQL ステートメントであった場合に変数を使用することができる個所で使用します。パラメーター・マーカーが、値によってどのように置き換えられるかについては、 1596 ページの『OPEN』および 1458 ページの『EXECUTE』を参照してください。

パラメーター・マーカーには、次の 2 つのタイプがあります。

型付きパラメーター・マーカー
: ターゲット・データ・タイプが一緒に指定されているパラメーター・マーカー。その汎用形式は、次のとおりです。

```
CAST(? AS データ・タイプ)
```

この表記は関数呼び出しではありませんが、実行時のそのパラメーターのタイプが指定のデータ・タイプになるか、指定のデータ・タイプに変換できるデータ・タイプになることを「約束」します。例えば、次の場合、

```
UPDATE EMPLOYEE
  SET LASTNAME = TRANSLATE(CAST(? AS VARCHAR(12)))
 WHERE EMPNO = ?
```

TRANSLATE 関数の引数の値は、実行時に提供されます。その値のデータ・タイプは、VARCHAR(12)、あるいは VARCHAR(12) に変換できるデータ・タイプになります。詳しくは、 218 ページの『CAST の指定』を参照してください。

型なしパラメーター・マーカー
: ターゲット・データ・タイプが指定されていないパラメーター・マーカー。その形式は、単一の疑問符 (?) です。型なしパラメーター・マーカーのデータ・タイプは、コンテキストによって提供されます。例えば、上記の更新ステートメントの述部にある型なしパラメーター・マーカーは、EMPNO 列のデータ・タイプと同じになります。

タイプ付きパラメーター・マーカーは、変数がサポートされており、データ・タイプが CAST 関数の約束に基づいていれば、動的 SQL ステートメント内のどこでも使用できます。

型なしパラメーター・マーカーは、変数がサポートされている場合、動的 SQL ステートメント内の選択された場所で使用できます。使用する場所とその結果のデータ・タイプを、以下の表にまとめます。

表 *113.* 式における型なしパラメーター・マーカーの使用 *(*選択リスト、*CASE*、および *VALUES* を含む*)*

型なしパラメーター・マーカーの場所	データ・タイプ
副照会内でない選択リストで単独で使用	エラー
EXISTS 副照会内の選択リストで単独で使用	エラー
副照会内の選択リストで単独で使用	副照会の他のオペランドのデータ・タイプ。[120]
offset-clause で *offset-row-count* として単独で使用。	BIGINT
fetch-clause で *fetch-row-count* として単独で使用。	BIGINT
単一算術演算子の両方のオペランド (演算子優先順位と演算順序の規則を考慮して) 次の場合も含まれます。 `? + ? + 10`	DECFLOAT(34)
算術式 (日時式は除く) の単一演算子の一方のオペランド 次の場合も含まれます。 `? + ? * 10`	他方のオペランドのデータ・タイプ。
単位タイプが SECONDS 以外の、日時式内のラベル付き期間。(ラベル付き期間のうち、単位のタイプを示す部分はパラメーター・マーカーにできません)	DECIMAL(15,0)
単位タイプが SECONDS の、日時式内のラベル付き期間。(ラベル付き期間のうち、単位のタイプを示す部分はパラメーター・マーカーにできません)	DECIMAL(27,12)
日時式のその他のオペランド (例えば、'timecol + ?' や '? - datecol')	エラー
CONCAT 演算子の両オペランドとして使用。	DBCLOB(1G) CCSID 1200
他のオペランドが CLOB 以外の文字データ・タイプである場合に、CONCAT 演算子の 1 つのオペランドとして使用。	他のオペランドと同じ CCSID の VARCHAR(32740)
他のオペランドが DBCLOB 以外のグラフィック・データ・タイプである場合に、CONCAT 演算子の 1 つのオペランドとして使用。	他のオペランドと同じ CCSID の VARGRAPHIC(16370)

表 113. 式における型なしパラメーター・マーカーの使用 (選択リスト、*CASE*、および *VALUES* を含む) (続き)

型なしパラメーター・マーカーの場所	データ・タイプ
他のオペランドが BLOB 以外のバイナリー・タイプである場合に、CONCAT 演算子の 1 つのオペランドとして使用。	VARBINARY(32740)
他のオペランドがラージ・オブジェクト・ストリングである場合に、CONCAT 演算子の 1 つのオペランドとして使用。	他のオペランドのデータ・タイプと同じ。
CASE 式の CASE キーワードの後の式	型なしパラメーター・マーカー以外の WHEN キーワードの後の式に対して、 133 ページの『結果のデータ・タイプに関する規則』を適用した結果。
CASE 式 (単純および検索) 内の結果式の少なくとも 1 つ。残りの結果式は、型なしパラメーター・マーカーか NULL のどちらかである場合。	エラー
単純 CASE 式内の WHEN の後の任意または全部の式	CASE の後の式と、型なしパラメーター・マーカーでない WHEN の後の式に対して、 133 ページの『結果のデータ・タイプに関する規則』 を適用した結果。
CASE 式 (単純と検索の両方) 内の結果式。その式の少なくとも 1 つの結果式が、非 NULL で、型なしパラメーター・マーカーでもない場合。	NULL または型なしパラメーター・マーカー以外のすべての結果式に対して、 133 ページの『結果のデータ・タイプに関する規則』を適用した結果。
INSERT ステートメントの中になく、MERGE ステートメントの挿入操作の VALUES 文節内にない単一行の VALUES 文節で、*column-expression* として単独で使用。	エラー
INSERT ステートメント内になく、他のすべての行式 で同じ位置にある列式 が型なしパラメーター・マーカーである複数行 VALUES 文節の列式 として単独で。	エラー
INSERT ステートメント内になく、他の行式のうちの少なくとも 1 つで同じ位置にある列式 が型なしパラメーター・マーカーでも NULL でもない複数行 VALUES 文節の列式として単独で。	型なしパラメーター・マーカー以外のすべてのオペランドに 133 ページの『結果のデータ・タイプに関する規則』を適用した結果。
INSERT ステートメント内の単一行 VALUES 文節で列式 として単独で使用。	列のデータ・タイプ。その列がユーザー定義特殊タイプとして定義されている場合は、ユーザー定義特殊タイプのソース・データ・タイプ。[120]
INSERT ステートメント内の複数行 VALUES 文節で列式 として単独で使用。	列のデータ・タイプ。その列がユーザー定義特殊タイプとして定義されている場合は、ユーザー定義特殊タイプのソース・データ・タイプ。[120]
MERGE ステートメントのソース表の VALUES 文節で、列式 として単独で使用	エラー

表 113. 式における型なしパラメーター・マーカーの使用 (選択リスト、*CASE*、および *VALUES* を含む) (続き)

型なしパラメーター・マーカーの場所	データ・タイプ
MERGE ステートメントの挿入操作の VALUES 文節で、列式 として単独で使用。	列のデータ・タイプ。その列がユーザー定義特殊タイプとして定義されている場合は、ユーザー定義特殊タイプのソース・データ・タイプ。[120]
MERGE ステートメントの更新操作の場合に、割り当て文節の右側で、列式 として単独で使用。	列のデータ・タイプ。その列がユーザー定義特殊タイプとして定義されている場合は、ユーザー定義特殊タイプのソース・データ・タイプ。[120]
UPDATE ステートメントの SET 文節の右側の値として単独で使用。	列のデータ・タイプ。その列がユーザー定義特殊タイプとして定義されている場合は、ユーザー定義特殊タイプのソース・データ・タイプ。[120]
INSERT ステートメントの *insert-multiple-rows* の中の値として	INTEGER
SET 特殊レジスター・ステートメントの右側の値として	特殊レジスターのデータ・タイプ。
関連付けられた式がグローバル変数である VALUES INTO ステートメントの VALUES 文節内の値として	グローバル変数のデータ・タイプ。
VALUES INTO ステートメントの INTO 文節内の値として	関連式のデータ・タイプ。 [120]
FREE LOCATOR または HOLD LOCATOR ステートメントの中の値として	ロケーター
SET ENCRYPTION PASSWORD ステートメントの中のパスワードの値として	VARCHAR(128)
SET ENCRYPTION PASSWORD ステートメントの中のヒントの値として	VARCHAR(32)

表 114. 述部での型なしパラメーター・マーカーの使用法

型なしパラメーター・マーカーの場所	データ・タイプ
比較演算子または DISTINCT 述部の両方のオペランド	VARGRAPHIC(16370) CCSID 1200
比較演算子または DISTINCT 述部の一方のオペランド。他方のオペランドが、非型付きパラメーター・マーカーまたは特殊タイプ以外の場合。	他のオペランドのデータ・タイプ。[120]
比較演算子の一方のオペランド。他方のオペランドが特殊タイプの場合。	エラー
BETWEEN 述部のすべてのオペランド	VARGRAPHIC(16370) CCSID 1200
BETWEEN 述部の 2 つのオペランド	唯一の非パラメーター・マーカーのデータ・タイプと同じ。

表 *114.* 述部での型なしパラメーター・マーカーの使用法 (続き)

型なしパラメーター・マーカーの場所	データ・タイプ
BETWEEN 述部の 1 つのみのオペランド	型なしパラメーター・マーカー以外のすべてのオペランドに 133 ページの『結果のデータ・タイプに関する規則』を適用した結果。ただし、CCSID 属性は、実行時に指定された値の CCSID になります。
IN 述部のすべてのオペランド。例えば、? IN (?,?,?)	VARGRAPHIC(16370) CCSID 1200
IN 述部の第 1 オペランド。右側が全選択の場合 (例えば、? IN (全選択))	選択された列のデータ・タイプ。
IN 述部の第 1 オペランド。右側が全選択でない場合 (例えば、? IN (?,A,B) または ? IN (A,?,B,?) など) .	IN リスト内の、型なしパラメーター・マーカー以外のすべてのオペランド (IN キーワードの右側のオペランド) に対して、 133 ページの『結果のデータ・タイプに関する規則』を適用した結果。ただし、CCSID 属性は、実行時に指定された値の CCSID になります。
IN 述部の IN リストの任意または全部のオペランド (例えば、IN (?,B,?))	IN 述部の、型なしパラメーター・マーカー以外のすべてのオペランド (IN 述部の左右のオペランド) に対して、 133 ページの『結果のデータ・タイプに関する規則』を適用した結果。ただし、CCSID 属性は、実行時に指定された値の CCSID になります。
IN 述部の行値表現 の中の任意のオペランド。例えば、(c1,?) IN ...	エラー
IN 述部で *row-value-expression* が指定される場合、副照会内の任意の選択リスト項目。例えば、(c1,c2) IN (SELECT ?, c1 FROM ...)	エラー
IS JSON 述部の第 1 オペランド	CLOB(2G) CCSID 1208
JSON_EXISTS 述部の *json-expression* または *sql-json-path-expression*	CLOB(2G) CCSID 1208
LIKE 述部の 3 つのオペランドすべて	一致式 (オペランド 1) とパターン式 (オペランド 2) は VARCHAR(32740)。エスケープ式 (オペランド 3) は、ジョブの CCSID の VARCHAR(1) [121]。
LIKE 述部の一致式。パターン式またはエスケープ式がタイプなしパラメーター・マーカーではない場合。	VARCHAR(32740)、VARGRAPHIC(16370)、または VARBINARY(32740) のいずれか。タイプなしパラメーター・マーカーではない第 1 オペランドのデータ・タイプによって決まります。CCSID は、最初のオペランドの CCSID に基づきます。

表 *114.* 述部での型なしパラメーター・マーカーの使用法 (続き)

型なしパラメーター・マーカーの場所	データ・タイプ
LIKE 述部のパターン式。一致式またはエスケープ式がタイプなしパラメーター・マーカーではない場合。	VARCHAR(32740)、VARGRAPHIC(16370)、または VARBINARY(32740) のいずれか。タイプなしパラメーター・マーカーではない第 1 オペランドのデータ・タイプによって決まります。CCSID は、最初のオペランドの CCSID に基づきます。 パターンの値の固定長変数の使用法については、 262 ページの『LIKE 述部』を参照してください。
LIKE 述部のエスケープ式。一致式またはパターン式が非型付きパラメーター・マーカーではない場合。	非型付きパラメーター・マーカー以外のすべてのオペランドに 133 ページの『結果のデータ・タイプに関する規則』を適用した結果に基づいて、VARCHAR(1) [121]、VARGRAPHIC(1)、または VARBINARY(1) のいずれか。CCSID も、これらの規則を適用した結果に基づきます。
NULL 述部のオペランド	VARGRAPHIC(16370) CCSID 1200

表 *115.* 組み込み関数での型なしパラメーター・マーカーの使用法

型なしパラメーター・マーカーの場所	データ・タイプ
BITAND、 BITANDNOT、BITOR、BITXOR、BITNOT、 COALESCE、IFNULL、LAND、LOR、MIN、 MAX、NULLIF、VALUE、または XOR のすべての引数	エラー
少なくとも 1 つの引数が非型付きパラメーター・マーカー以外の、COALESCE、 IFNULL、LAND、LOR、MIN、MAX、 NULLIF、VALUE、または、XOR の引数。	タイプ無しパラメーター・マーカー以外のすべての引数に 133 ページの『結果のデータ・タイプに関する規則』 を適用した結果。結果が特殊タイプの場合、エラーが戻されます。
他方の引数が非型付きパラメーター・マーカー以外の、BITAND、 BITANDNOT、BITOR、および BITXOR の引数。	他方の引数が SMALLINT、 INTEGER、または BIGINT の場合、他方の引数のデータ・タイプ。それ以外の場合は DECFLOAT(34)。
BSON_TO_JSON の最初の引数	BLOB(2G)
COMPARE_DECFLOAT、 DECFLOAT_SORTKEY、 NORMALIZE_DECFLOAT、 QUANTIZE、 および TOTALORDER のすべての引数	DECFLOAT(34)
DAYNAME の最初の引数	TIMESTAMP(12)
DECFLOAT_FORMAT の最初の引数	VARGRAPHIC(16370) CCSID 1200
DECFLOAT_FORMAT の 2 番目の引数	エラー
HASH、HASH_MD5、 HASH_SHA1、HASH_SHA256、または HASH_SHA512 の最初の引数	DBCLOB(1G) CCSID 1200

表 115. 組み込み関数での型なしパラメーター・マーカーの使用法 (続き)

型なしパラメーター・マーカーの場所	データ・タイプ
HASH の 2 番目の引数	INTEGER
JSON_ARRAY、JSON_OBJECT、JSON_QUERY、または JSON_VALUE の *JSON-expression* (FORMAT BSON が指定された場合) の引数	BLOB(2G)
JSON_ARRAY、JSON_OBJECT、JSON_QUERY、または JSON_VALUE の *JSON-expression* (FORMAT BSON が指定されない場合)、 *key-name-expression*、または *sql-json-path-expression* の引数	CLOB(2G) CCSID 1208
JSON_TO_BSON の最初の引数	CLOB(2G) CCSID 1208
LOCATE、POSITION、または POSSTR の 2 つの引数	DBCLOB(1G) CCSID 1200
他の引数が文字データ・タイプの場合、LOCATE、POSITION、または POSSTR の 1 つの引数。	他方の引数の CCSID の VARCHAR(32740)
他の引数がグラフィック・データ・タイプの場合、LOCATE、POSITION、または POSSTR の 1 つの引数。	他方の引数の CCSID の VARGRAPHIC(16370)
他の引数が 2 進データ・タイプの場合、LOCATE、POSITION、または POSSTR の 1 つの引数。	VARBINARY(32740)
LOCATE_IN_STRING または OVERLAY の最初と 2 番目の引数の両方	DBCLOB(1G) CCSID 1200
他方のストリング引数が文字データ・タイプである場合、LOCATE_IN_STRING または OVERLAY の最初または 2 番目の引数	他方の引数の CCSID の VARCHAR(32740)
他方のストリング引数がグラフィック・データ・タイプである場合、LOCATE_IN_STRING または OVERLAY の最初または 2 番目の引数	他方の引数の CCSID の VARGRAPHIC(16370)
他方のストリング引数がバイナリー・データ・タイプである場合、LOCATE_IN_STRING または OVERLAY の最初または 2 番目の引数	VARBINARY(32740)
LOCATE_IN_STRING または OVERLAY の 3 番目または 4 番目の引数	INTEGER
LPAD または RPAD の 2 番目の引数	INTEGER
LPAD または RPAD の 3 番目の引数	VARCHAR(32740)
LTRIM または RTRIM の最初の引数	DBCLOB(1G) CCSID 1200

表 *115.* 組み込み関数での型なしパラメーター・マーカーの使用法 (続き)

型なしパラメーター・マーカーの場所	データ・タイプ
LTRIM または RTRIM の 2 番目の引数	最初の引数が文字タイプの場合、VARCHAR(32740)。最初の引数がバイナリー・タイプの場合、VARBINARY(32740)。最初の引数がグラフィック・タイプの場合、VARGRAPHIC(16370)。CCSID は、最初の引数の CCSID に基づきます。
UPPER、LOWER、UCASE、および LCASE の引数	DBCLOB(1G) CCSID 1200
MONTHNAME の最初の引数	TIMESTAMP(12)
MONTHS_BETWEEN の最初の引数と 2 番目の引数として使用。	TIMESTAMP(12)
REGEXP_LIKE、REGEXP_INSTR、REGEXP_SUBSTR、REGEXP_COUNT、および REGEXP_REPLACE の第 1 オペランド	DBCLOB(1G) CCSID 1200
REGEXP_LIKE、REGEXP_INSTR、REGEXP_SUBSTR、REGEXP_COUNT、および REGEXP_REPLACE の第 2 オペランド	DBCLOB(32K) CCSID 1200
REGEXP_REPLACE の第 3 オペランド	DBCLOB(32K) CCSID 1200
REGEXP_LIKE、REGEXP_COUNT、REGEXP_INSTR、REGEXP_SUBSTR、および REGEXP_REPLACE の *source-string* オペランド	DBCLOB(32K) CCSID 1200
REGEXP_LIKE、REGEXP_COUNT、REGEXP_INSTR、REGEXP_SUBSTR、および REGEXP_REPLACE の *pattern-expression*	DBCLOB(32K) CCSID 1200
REGEXP_REPLACE の *replacement-string* オペランド	DBCLOB(32K) CCSID 1200
REGEXP_LIKE、REGEXP_COUNT、REGEXP_INSTR、REGEXP_SUBSTR、および REGEXP_REPLACE の *start* オペランド	INTEGER
REGEXP_LIKE、REGEXP_COUNT、REGEXP_INSTR、REGEXP_SUBSTR、および REGEXP_REPLACE の *flags* オペランド	VARCHAR(6)
REGEXP_INSTR、REGEXP_SUBSTR、および REGEXP_REPLACE の *occurrence* オペランド	INTEGER
REGEXP_INSTR の *return-option* オペランド	INTEGER
REGEXP_INSTR および REGEXP_SUBSTR の *group* オペランド	INTEGER
SUBSTR (最初の引数)	DBCLOB(1G) CCSID 1200
SUBSTR (2 番目と 3 番目の引数)	INTEGER
TRANSLATE の最初の引数	エラー

表 *115.* 組み込み関数での型なしパラメーター・マーカーの使用法 (続き)

型なしパラメーター・マーカーの場所	データ・タイプ
TRANSLATE の 2 番目と 3 番目の引数	最初の引数が文字タイプの場合、VARCHAR(32740)。最初の引数がグラフィック・タイプの場合、VARGRAPHIC(16370)。CCSID は、最初の引数の CCSID に基づきます。
TRANSLATE の 4 番目の引数	最初の引数が文字タイプの場合、VARCHAR(1)。最初の引数がグラフィック・タイプの場合、VARGRAPHIC(1)。CCSID は、最初の引数の CCSID に基づきます。
TIMESTAMP または TIMESTAMP_ISO の最初の引数	エラー
TIMESTAMP の 2 番目の引数	TIME
TIMESTAMP_FORMAT の最初の引数	VARGRAPHIC(16370) CCSID 1200
VARCHAR_FORMAT の最初の引数	TIMESTAMP(12)
TIMESTAMP_FORMAT または VARCHAR_FORMAT の 2 番目の引数	エラー
VERIFY_GROUP_FOR_USER の最初の引数より後のすべての引数	VARCHAR(128)
XMLVALIDATE の最初の引数	XML [122]
XMLPARSE の最初の引数	CLOB(2G) または DBCLOB(1G)。照会オプション SQL_XML_DATA_CCSID の CCSID 値によって決まります。
XMLCOMMENT の最初の引数	VARCHAR(32740) または VARGRAPHIC(16370)。照会オプション SQL_XML_DATA_CCSID の CCSID 値によって決まります。
XMLTEXT の最初の引数	VARCHAR(32740) または VARGRAPHIC(16370)。照会オプション SQL_XML_DATA_CCSID の CCSID 値によって決まります。
XMLPI の 2 番目の引数	VARCHAR(36740) または VARGRAPHIC(16370)。照会オプション SQL_XML_DATA_CCSID の CCSID 値によって決まります。
XMLSERIALIZE の最初の引数	XML [123]
XMLDOCUMENT のすべての引数	XML [122]
XMLCONCAT のすべての引数	XML [122]
XSLTRANSFORM の最初、2 番目、および 3 番目の引数	XML [123]
ARRAY の配列指標	BIGINT
単項減算で使用。	DECFLOAT(34)
単項加算で使用。	DECFLOAT(34)
その他のすべてのスカラー関数のその他のすべての引数。	エラー

表 115. 組み込み関数での型なしパラメーター・マーカーの使用法 (続き)

型なしパラメーター・マーカーの場所	データ・タイプ
MEDIAN の引数	DECFLOAT(34)
CORRELATION、COVARIANCE、COVARIANCE_SAMP、またはいずれかの回帰関数の最初または 2 番目の引数。	他方の引数が DECFLOAT の場合、DECFLOAT(34)。それ以外の場合は、DOUBLE。
JSON_ARRAYAGG または JSON_OBJECTAGG の *JSON-expression* (FORMAT BSON が指定された場合) の引数	BLOB(2G)
JSON_ARRAYAGG または JSON_OBJECTAGG の *JSON-expression* (FORMAT BSON が指定されない場合) または *key-name-expression* の引数	CLOB(2G) CCSID 1208
LISTAGG の 2 番目の引数	最初の引数が文字タイプの場合、VARCHAR(32740)。最初の引数がバイナリー・タイプの場合、VARBINARY(32740)。最初の引数がグラフィック・タイプの場合、VARGRAPHIC(16370)。CCSID は、最初の引数の CCSID に基づきます。
PERCENTILE_CONT または PERCENTILE_DISC の引数または ORDER BY 式	DECFLOAT(34)
他のすべての集約関数の引数	エラー
JSON_TABLE の *JSON-expression* または *sql-json-path-expression* の引数	CLOB(2G) CCSID 1208

表 116. ユーザー定義ルーチンでの型なしパラメーター・マーカーの使用法

型なしパラメーター・マーカーの場所	データ・タイプ
関数の引数	関数の作成時に定義された、パラメーターのデータ・タイプ
プロシージャーの引数	プロシージャーの作成時に定義された、パラメーターのデータ・タイプ

エラー検査**:** PREPARE ステートメントが実行されると、ステートメント・ストリングが解析され、エラーがないか検査されます。ステートメント・ストリングが無効な場合は、準備済みステートメントは作成されず、エラーが戻されます。

120. データ・タイプが DATE、TIME、または TIMESTAMP の場合、VARCHAR(32740) が使用されます。

121. エスケープ式が MIXED データの場合、データ・タイプは VARCHAR(4) です。

122. XML の CCSID は、 101 ページの『XML 値』で説明されているように決定されます。

123. CCSID は、 218 ページの『CAST の指定』で説明されているように、AS 節に指定された *data-type* の属性に基づいて決定されます。 *data-type* がバイナリー・ストリングまたはビット・データの場合、SQL_XML_DATA_CCSID が CCSID 属性に使用されます。

ローカルおよびリモート処理では、DLYPREP(*YES) オプションを指定すると、一部の SQL ステートメントで「遅延」エラーを受け取ることがあります。例えば、DESCRIBE、EXECUTE、および OPEN で、通常は PREPARE 処理中に出される SQLCODE を受け取ることがあります。

参照および実行の規則: 準備済みステートメントは、以下のようなステートメントから参照できます (ただし、ステートメントによっては、参照できる準備済みステートメントが制約されることがあります)。

ステートメント	準備済みステートメントの制約事項
DESCRIBE	なし
DECLARE CURSOR	カーソルがオープンされているときは SELECT する必要がある。
EXECUTE	SELECT してはならない。

準備済みステートメントは、何度でも実行することができます。準備済みステートメントを一度しか実行せず、ステートメントの中でパラメーター・マーカーも使用しない場合は、PREPARE ステートメントと EXECUTE ステートメントを使用するより、EXECUTE IMMEDIATE ステートメントを使用した方が効率的です。

拡張標識の使用法: EXTENDED INDICATORS 節は、拡張標識変数値が UPDATE ステートメントの SET 割り当て文節、INSERT ステートメントの VALUES 式リスト、または MERGE ステートメントの挿入操作または更新操作において使用可能かどうかを示します。

拡張標識変数とエラー検査の据え置き: 拡張標識変数が使用可能な場合、UNASSIGNED 標識変数値により、ステートメントから確実にそのターゲット列が除外されます。このため、通常はステートメント準備中に行われる妥当性検査は、ステートメントの実行時まで遅延されます。

準備済みステートメントの持続性: 準備済みステートメントはすべて、次の場合に破棄されます。[124]

- CONNECT (タイプ 1) ステートメントが実行された場合。
- 準備済みステートメントが関連する接続が、DISCONNECT ステートメントにより切り離された場合。
- 準備済みステートメントが解除保留の接続に関連し、正常なコミットが行われた場合。
- SQL ステートメントに関連した有効範囲 (ジョブ、活動化グループ、またはプログラム) が終了した場合。

ステートメントの有効範囲: *statement-name* の有効範囲は、それが定義されているソース・プログラムです。準備済みステートメントを他の SQL ステートメントから参照できるのは、その SQL ステートメントが PREPARE ステートメントによってプリコンパイルされたものである場合だけです。例えば、別にコンパイルされた他のプログラムから呼び出されたプログラムは、呼び出し側プログラムによって作成された準備済みステートメントを使用することができません。

また、ステートメント名の有効範囲は、そのステートメントを含むプログラムが実行しているスレッドに限定されます。例えば、同じジョブの中にある別々の 2 つの

124. 準備済みステートメントは、キャッシュに入れて、実際に破棄しないことも可能です。ただし、キャッシュに入れたステートメントは、同一のステートメントを再度準備するときにしか使用できません。

スレッドで同じプログラムが実行している場合、2 番目のスレッドは、最初のスレッドが準備したステートメントを使用することができません。

ステートメントが定義されているプログラムがそのステートメントの有効範囲ですが、そのプログラムから作成された各パッケージはそれぞれ準備されたステートメントの別個のインスタンスを含み、実行時に準備されたステートメントが複数存在することがあります。例えば、CONNECT (タイプ 2) ステートメントを使用して、次の順序でロケーション X とロケーション Y に接続するプログラムを想定します。

```
EXEC SQL CONNECT TO X;
EXEC SQL PREPARE S FROM :hv1;
EXEC SQL EXECUTE S;
.
.
.
EXEC SQL CONNECT TO Y;
EXEC SQL PREPARE S FROM :hv1;
EXEC SQL EXECUTE S;
```

S の 2 番目の準備は、Y で S の別個のインスタンスを準備します。

CRTSQLxxx コマンドに CLOSQLCSR(*ENDJOB)、CLOSQLCSR(*ENDACTGRP)、または CLOSQLCSR(*ENDSQL) が指定されていない場合、準備済みステートメントを参照できるのは、プログラム・スタックにあるプログラムの同一のインスタンスに制限されます。

- CLOSQLCSR(*ENDJOB) が指定されている場合、プログラム・スタックにあるそのプログラム (ステートメントを準備したプログラム) のどのインスタンスでも準備済みステートメントを参照できます。この場合、準備済みステートメントはジョブの終わりに破棄されます。
- CLOSQLCSR(*ENDSQL) が指定されている場合、プログラム・スタック内の最後の SQL プログラムが終了するまでの間、 プログラム・スタックにあるそのプログラム (ステートメントを準備したプログラム) のどのインスタンスでも、準備済みステートメントを参照できます。この場合、準備済みステートメントはプログラム・スタックの最後の SQL プログラムが終了すると破棄されます。
- CLOSQLCSR(*ENDACTGRP) が指定されている場合、その活動化グループが終了するまでは、そのステートメントを準備したプログラムのモジュールのすべてのインスタンスで、その準備済みステートメントを参照できます。この場合、準備済みステートメントは活動化グループが終了すると破棄されます。

SQL 記述子の割り振り**:** USING 文節が指定される場合、PREPARE ステートメントを実行する前に、ALLOCATE DESCRIPTOR ステートメントを使用して SQL 記述子を割り振らなければなりません。割り振られた記述子項目の数が結果列の数よりも小さい場合、警告 (SQLSTATE 01005) が戻されます。

PREPARE および ***LIBL:** 通常、オブジェクトの非修飾名は、ステートメントが準備される際に解決されます。したがって、ステートメントが準備された後に CURRENT SCHEMA または CURRENT PATH が変更されても、ステートメントの実行またはオープン時にどのオブジェクトが参照されるかに対しては影響しません。ただし、システム名が使用され、オブジェクト名が *LIBL で暗黙に修飾されている場合、オブジェクトは実行時またはオープン時に解決されます。ステートメン

トが準備された後、ただし、実行時またはオープン時より前にライブラリー・リストが変更されると、ステートメントの実行時またはオープン時にどのオブジェクトが参照されるかに影響します。

例

例 *1:* COBOL プログラムで、*select-statement* 以外のステートメントを準備して実行します。このステートメントは、変数 HOLDER に入っているものとします。プログラムでは、ユーザーからの何らかの指示に基づいて、変数 HOLDER にステートメント・ストリングを入れます。準備するステートメントには、パラメーター・マーカーは入っていません。

```
EXEC SQL  PREPARE STMT_NAME FROM :HOLDER  END-EXEC.

EXEC SQL  EXECUTE STMT_NAME  END-EXEC.
```

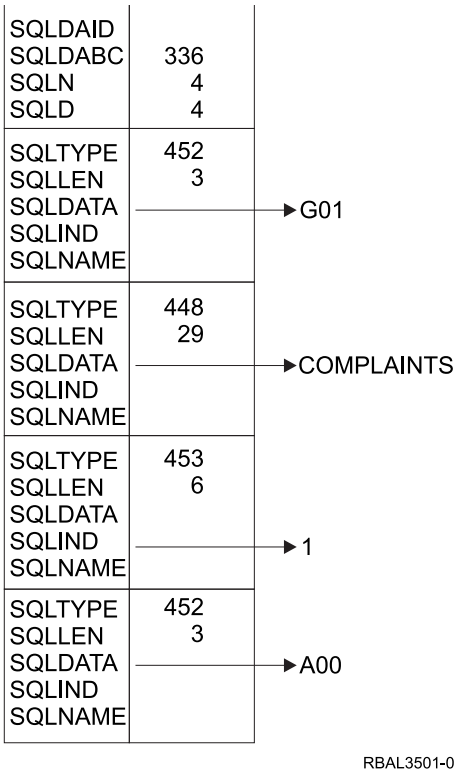
例 *2:* 例 1 と同様に、*select-statement* 以外のステートメントを準備して実行しますが、準備するステートメントには、任意の数のパラメーター・マーカーを含めることができるものとします。

```
EXEC SQL  PREPARE STMT_NAME FROM :HOLDER  END-EXEC.

EXEC SQL  EXECUTE STMT_NAME USING DESCRIPTOR :INSERT_DA  END-EXEC.
```

以下のステートメントを準備するものとします。

```
INSERT INTO DEPARTMENT VALUES(?, ?, ?, ?)
```

部門番号 G01、部門名 COMPLAINTS、管理者なし、報告先部門は部門 A00 という行を挿入するとすれば、 EXECUTE ステートメントを実行する前に、構造体 INSERT_DA には次のような値が入っていなければなりません。



RBAL3501-0

REFRESH TABLE

REFRESH TABLE ステートメントは、マテリアライズ照会表のデータをリフレッシュします。このステートメントはマテリアライズ照会表のすべての行を削除してから、マテリアライズ照会表の定義で指定された選択ステートメント にある結果行を挿入します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

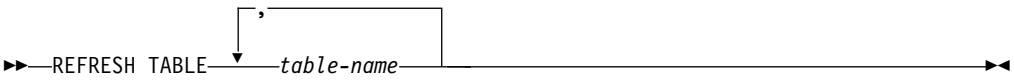
権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメントで識別される表に対して、
  - 表に対するシステム権限 *OBJMGT
  - 表に対する DELETE 特権
  - 表に対する INSERT 特権
  - 表が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

SQL 特権に対応するシステム権限については、『表またはビューへの権限を検査する際の対応するシステム権限』を参照してください。

構文



説明

table-name
リフレッシュするマテリアライズ表を識別します。 *table-name* では、現行サーバーに存在するマテリアライズ照会表を指定する必要があります。 REFRESH TABLE はマテリアライズ照会表の定義にある 選択ステートメント を評価して、表をリフレッシュします。

注

マテリアライズ照会表のリフレッシュの使用**:** マテリアライズ照会表は、REFRESH TABLE ステートメントの処理中に選択ステートメント を評価するためには使用されません。

リフレッシュ分離レベル**:** 選択ステートメント の評価に使用される分離レベルは、次のいずれかです。

- 選択ステートメント の分離レベル 文節に指定されている分離レベル、または

- 分離レベル 文節が指定されていない場合、 CREATE TABLE または ALTER TABLE の発行時に記録されたマテリアライズ照会表の分離レベル。

行数: REFRESH TABLE ステートメントが正常に実行された後、 SQL 診断領域の ROW_COUNT ステートメント情報項目 (または SQLCA の SQLERRD(3)) には、マテリアライズ照会表に挿入された行数が含まれます。

例

TRANSCOUNT マテリアライズ照会表のデータをリフレッシュします。

```
REFRESH TABLE TRANSCOUNT
```

RELEASE (接続)

RELEASE ステートメントは、1 つ以上の接続を解放ペンディング状態にします。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことと、対話式に呼び出すことだけが可能です。これは実行可能ステートメントですが、動的に準備することはできません。Java および REXX では指定できません。

RELEASE は、トリガーおよび関数で使用できません。

権限

ステートメントでグローバル変数を参照する場合は、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別されるグローバル変数に対して、
  - そのグローバル変数に対する READ 特権
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

構文

説明

server-name または *variable*

指定したサーバー名、または指定した変数に入っているサーバー名によって接続を識別します。スキーマ名で修飾すれば、グローバル変数を使用することもできます。 変数を指定する場合

- その変数は、文字ストリング変数でなければなりません。
- 標識変数を伴っていてはなりません。
- サーバー名は、その変数内で左寄せし、通常 ID の形成の規則に従っていなければなりません。
- サーバー名の長さが、変数の長さよりも短い場合、右側をブランクで埋めなければなりません。

RELEASE ステートメントが実行される時点で、指定したサーバー名、または指定の変数に入っているサーバー名は、活動化グループの既存の接続を識別していなければなりません。

CURRENT

活動化グループの現行接続を識別します。活動化グループは接続状態でなければなりません。

ALL または **ALL SQL**
活動化グループの既存のすべての接続 (ローカルおよびリモートの接続の両方) を識別します。

このステートメントの実行時に接続が存在しない場合、エラーや警告は起こりません。

ALL という名前のアプリケーション・サーバーを指定する場合は、変数または区切り ID しか使用できません。

RELEASE ステートメントが正常に実行された場合は、識別されている各接続は解除保留状態になり、したがって、次のコミット操作中に終了することになります。RELEASE ステートメントが不成功の場合には、その活動化グループの接続状態およびその接続の状態は変わりません。

注

RELEASE と CONNECT (タイプ 1): CONNECT (タイプ 1) の使用は、RELEASE の使用を妨げることはありません。

RELEASE の有効範囲: RELEASE は、カーソルをクローズしません。また、どのようなリソースも解放しません。さらに、該当の接続をさらに使用するのを妨げることはありません。

リモート接続のリソースに関する考慮事項**:** リモート接続を作成し維持するためには、リソースが必要です。したがって、再使用の予定がないリモート接続は、解除保留状態にする必要があり、再使用の予定があるリモート接続は、解除保留状態にしてはなりません。

接続状態**:** ROLLBACK は、接続を解除保留状態から保留状態にリセットすることはしません。

コミット操作が行われる時点で現行接続が解除保留状態にある場合は、その接続は終了し、その活動化グループは未接続状態になります。この場合は、次に実行される SQL ステートメントは、 CONNECT または SET CONNECTION であることが必要です。

RELEASE ALL は、ローカルの解除保留への接続を解除保留状態にします。解除保留状態の接続は、WITH HOLD 文節を指定して定義したオープン・カーソルを持つ場合でも、コミット操作中に終了します。

例

例 1 : 次の作業単位では、TOROLAB1 との接続は必要としません。次のステートメントは、次のコミット操作の過程で既存の接続を終了させます。

```
EXEC SQL RELEASE TOROLAB1;
```

例 2 : 次の作業単位では、現行接続は必要としません。次のステートメントは、次のコミット操作の過程で既存の接続を終了させます。

```
EXEC SQL RELEASE CURRENT;
```

例 *3* : 次の作業単位では、既存の接続はいずれも必要としません。次のステートメントは、次のコミット操作の過程で既存の接続を終了させます。

```
EXEC SQL RELEASE ALL;
```

RELEASE SAVEPOINT

RELEASE SAVEPOINT ステートメントは、現行サーバーの 1 つの作業単位内で、指定されたセーブポイントとそれ以降に確立されたすべてのセーブポイントを解放します。

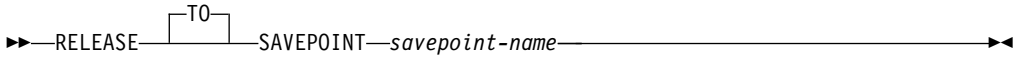
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

権限は不要です。

構文



説明

savepoint-name
解放するセーブポイントを指定します。この名前は、現行サーバーに存在するセーブポイントを示すものでなければなりません。現行サーバーにおける指定したセーブポイントと、この作業単位内でそれ以降に確立されているすべてのセーブポイントが解放されます。解放された後は、そのセーブポイントは維持されないので、そのセーブポイントまでのロールバックはできなくなります。

注

セーブポイント名**:** 解放したセーブポイントの名前は、別の SAVEPOINT ステートメントで再使用することができます。同じセーブポイント名が指定されている前の SAVEPOINT ステートメントで、UNIQUE キーワードが指定されていても構いません。

分離レベルの制約事項**:** 対象の活動化グループについてコミットメント制御が活動状態にない場合は、RELEASE SAVEPOINT ステートメントは使用できません。どのコミットメント定義が使用されているかを判別する方法については、COMMIT ステートメントの 1039 ページの『注』を参照してください。

例

あるメイン・ルーチンが、セーブポイント A を設定した後で、セーブポイント B および C を設定するサブルーチンを呼び出すものとします。メイン・ルーチンに制御が戻ると、メイン・ルーチンは、セーブポイント A とそれ以降に設定されたすべてのセーブポイントを解放します。つまり、A のほかに、サブルーチンが設定したセーブポイント B および C が解放されます。

```
RELEASE SAVEPOINT A
```

RENAME

RENAME ステートメントは、表、ビュー、または索引の名前を変更します。表、ビュー、または索引の名前またはシステム・オブジェクト名 (あるいは、その両方) を変更できます。

呼び出し

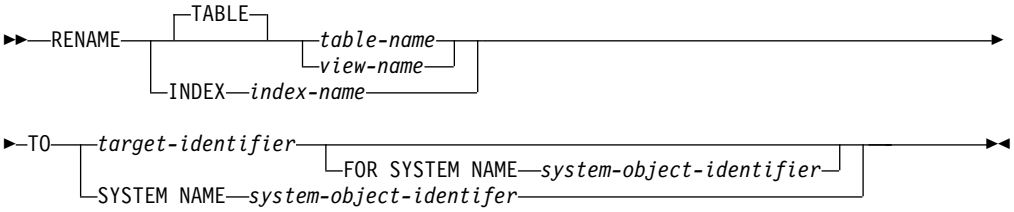
このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- 次のシステム権限
  - オブジェクト名を変更する場合、
    - 名前を変更する表、ビュー、または索引に対する *OBJMGT システム権限。
    - 名前を変更する表、ビュー、または索引が入っているスキーマに対する USAGE 特権。
  - オブジェクトのシステム名を変更する場合、
    - 名前を変更する表、ビュー、または索引に対する *OBJMGT システム権限。
    - 名前を変更する表、ビュー、または索引が入っているスキーマに対する USAGE 特権および *UPD システム権限。
- データベース管理者権限

構文



説明

TABLE *table-name* または *view-name*

名前の変更をする表またはビューを示します。*table-name* または *view-name* は、現行サーバーにある表またはビューを示すものでなければなりません。しかし、カタログ表または宣言済み一時表を示すものであってはなりません。指定された名前が別名であっても構いません。指定した表またはビューは、新しい名前に変更されます。その表あるいはビューに関する特権、制約、索引、トリガー、ビュー、および論理ファイルはすべてそのままの状態に保持されます。

該当の表あるいはビューを参照するアクセス・プランはいずれも、そのアクセス・プランを使用するプログラムが次回実行される時に再度暗黙的に準備されます。プログラムは元の名前で表あるいはビューを参照するので、その時に元の名前の表あるいはビューが存在しない場合、エラーが戻されます。

INDEX *index-name*
名前を変更する索引を示します。この索引名 は、現行サーバーに存在している索引を示すものでなければなりません。指定した索引は、新しい名前に変更されます。

この索引を参照するアクセス・プランは、名前変更によって影響は受けません。

target-identifier
それぞれ、表、ビュー、索引の新しい表名、ビュー名、索引名 を示します。*target-identifier* は、現行サーバーに既に存在する表、ビュー、別名、または索引と同一であってはなりません。*target-identifier* は、非修飾 SQL ID でなければなりません。

SYSTEM NAME *system-object-identifier*
それぞれ、表、ビュー、索引の、新しいシステム・オブジェクト *ID* を示します。システム・オブジェクト *ID* は、現行サーバーに既に存在する表、ビュー、別名、または索引と同一であってはなりません。システム・オブジェクト *ID* は、非修飾システム ID でなければなりません。

オブジェクトの名前とオブジェクトのシステム名が同一であり、 *target-identifier* が指定されていない場合、 *system-object-identifier* を指定すると、それが新規の名前およびシステム・オブジェクト名になります。それ以外の場合には、システム・オブジェクト *ID* の指定は、オブジェクトのシステム名だけに影響を与え、オブジェクトの名前には影響を与えません。

target-identifier と *system-object-identifier* の両方が指定されている場合、両方を有効なシステム・オブジェクト名にすることはできません。

注

ステートメントの効果**:** 指定した表、ビュー、または索引が新しい名前に変更されます。名前変更された表では、その表のすべての特権と索引が保持されます。名前変更された索引では、すべての特権が保持されます。

パッケージとアクセス・プランの無効化**:** その表を参照するアクセス・プランは、すべて無効になります。詳しくは、 17 ページの『パッケージとアクセス・プラン』を参照してください。

別名に関する考慮事項**:** *table-name* の別名が指定されている場合、その表は現行サーバーに存在していなければならず、その別名により識別される表が名前変更されます。その別名自体の名前は変更されないので、名前変更の後も引き続き古い表名を参照することになります。

RENAME ステートメントを使用して別名の名前を変更するためのサポートはありません。別名が参照する名前を変更するには、その別名を除去してから再作成する必要があります。

名前変更の新規**:** 名前変更の操作は、指定された新しい名前に応じて実行されます。

- 新しい名前が有効なシステム ID の場合、
  - 代替名がある場合は、それが除去されます。
  - システム・オブジェクト名は、新しい名前に変更されます。
- 新しい名前が有効な ID でない場合、
  - 代替名が追加されるか、新しい名前に変更されます。
  - システム・オブジェクト名 (表またはビューの) が、名前を変更する表、ビューまたは索引として指定された場合、新しいシステム・オブジェクト名が生成されます。表名の生成規則についての詳細は、 1298 ページの『表名の生成の規則』を参照してください。

表名 の別名が指定されている場合は、その別名は現行サーバーに存在していなければならず、そしてその別名により識別される表が名前変更されます。その別名自体の名前は変更されないので、名前変更の後も引き続き古い表を参照することになります。別名の名前を変更するためのサポートはありません。

例

例 *1*: EMPLOYEE 表の名前を CURRENT_EMPLOYEES に変更します。

```
RENAME TABLE EMPLOYEE
TO CURRENT_EMPLOYEES
```

例 *2*: XEMP1 という EMPNO を使用する固有索引の名前を UXEMPNO に変更します。

```
RENAME INDEX XEMP1
  TO UXEMPNO
```

例 *3*: MY_IN_TRAY という名前の表を MY_IN_TRAY_94 に名前変更します。システム・オブジェクト名は、そのまま変更されません (MY_IN_TRAY)。

```
RENAME TABLE MY_IN_TRAY TO MY_IN_TRAY_94
  FOR SYSTEM NAME MY_IN_TRAY
```

REVOKE (関数特権またはプロシージャー特権)

この形式の REVOKE ステートメントは、関数またはプロシージャーに対する特権を除去します。

呼び出し

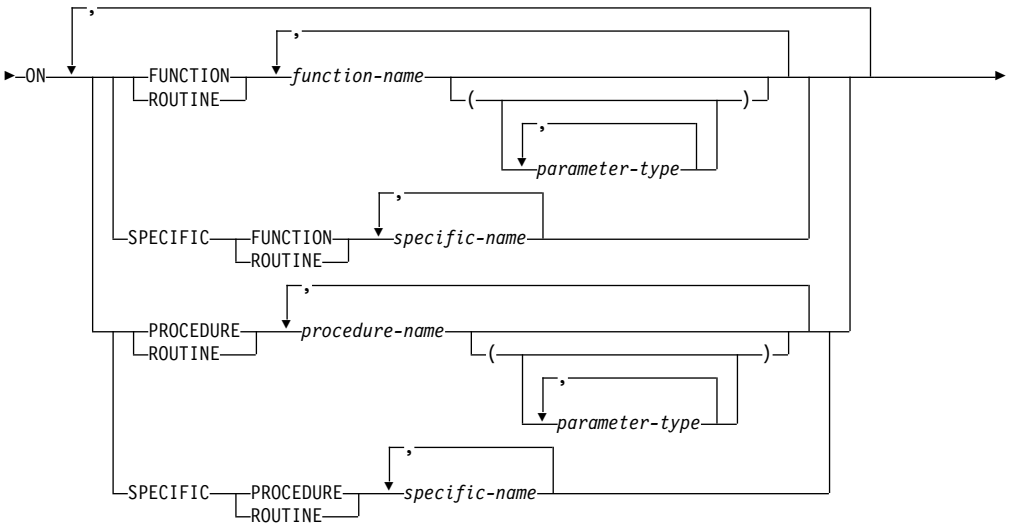
このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

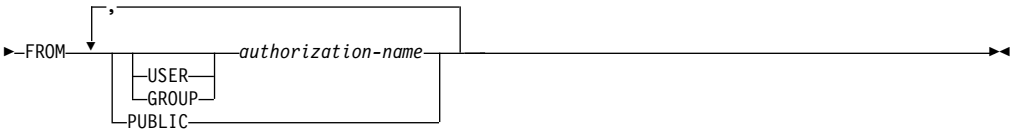
権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれの関数またはプロシージャーごとに、
  - このステートメントで指定されるすべての特権
  - その関数またはプロシージャーに対する *OBJMGT システム権限
  - その関数またはプロシージャーが入っているスキーマに対する USAGE 特権 (これが Java ルーチンの場合は、ディレクトリーに対する *EXECUTE システム権限)
- データベース管理者権限
- セキュリティー管理者権限

構文

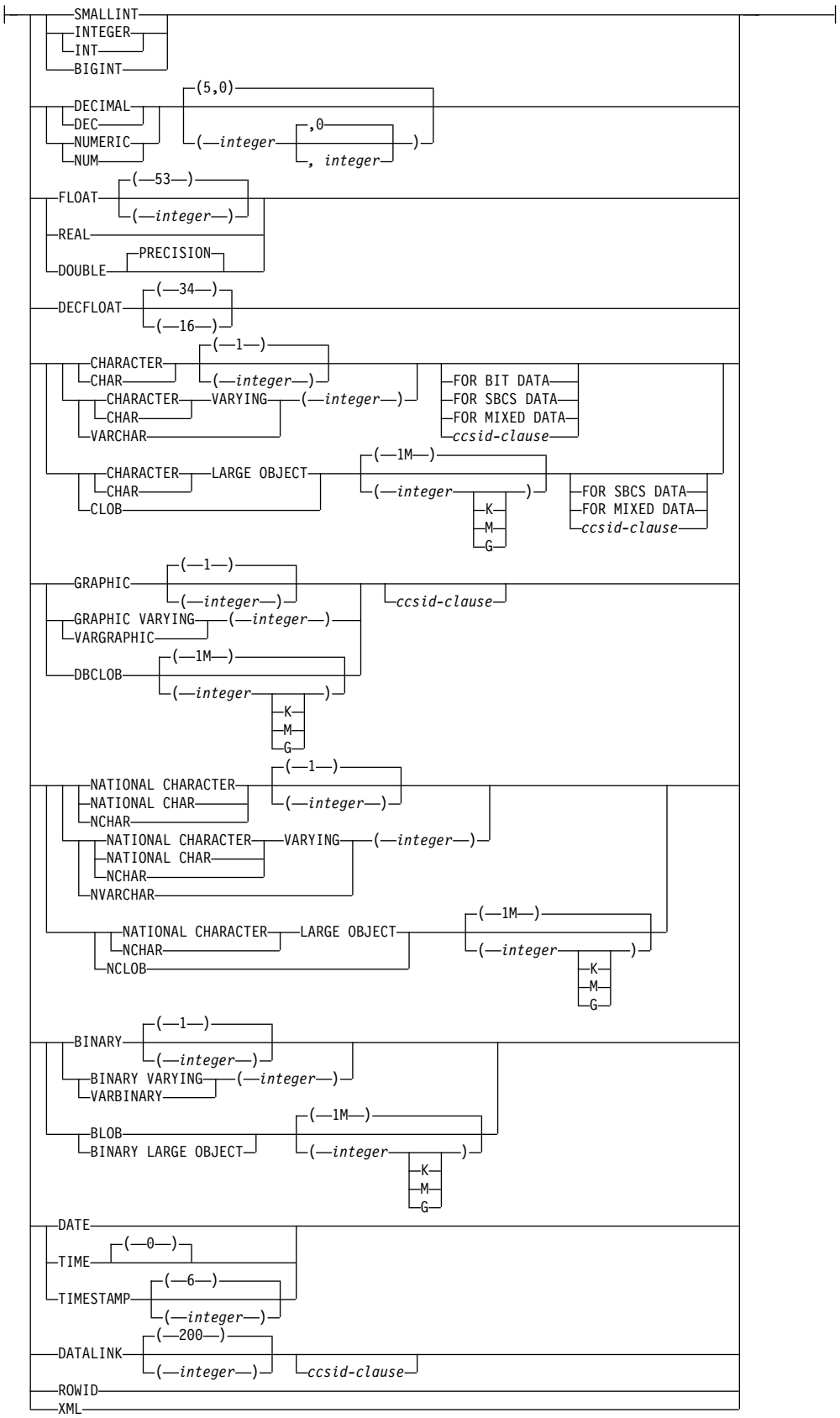




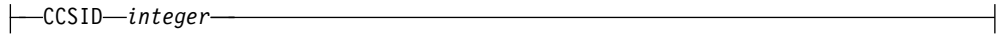
parameter-type:

data-type:

built-in-type:


SMALLINT
INTEGER
INT
BIGINT
DECIMAL
DEC
NUMERIC
NUM
(5,0)
(integer
,0
, integer
)
FLOAT
(53)
(integer)
REAL
DOUBLE
PRECISION
DECFLOAT
(34)
(16)
CHARACTER
CHAR
(1)
(integer)
CHARACTER VARYING
CHAR VARYING
VARCHAR
FOR BIT DATA
FOR SBCS DATA
FOR MIXED DATA
ccsid-clause
CHARACTER LARGE OBJECT
CHAR LARGE OBJECT
CLOB
(1M)
K
M
G
GRAPHIC
VARGRAPHIC
GRAPHIC VARYING
DBCLOB
NATIONAL CHARACTER
NATIONAL CHAR
NCHAR
NATIONAL CHARACTER VARYING
NATIONAL CHAR VARYING
NVARCHAR
NATIONAL CHARACTER LARGE OBJECT
NCHAR LARGE OBJECT
NCLOB
BINARY
BINARY VARYING
VARBINARY
BLOB
BINARY LARGE OBJECT
DATE
TIME
(0)
TIMESTAMP
(6)
DATALINK
(200)
ROWID
XML

ccsid-clause:



説明

ALL または ALL PRIVILEGES
各権限名 から 1 つ以上の関数またはプロシージャー特権を取り消します。取り消される特権は、識別された関数またはプロシージャーに関して、権限名 に認可されていた特権です。関数またはプロシージャーに対する ALL PRIVILEGES を取り消すのは、*ALL システム権限を取り消すのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つ以上を使用する必要があります。各キーワードは、そこで説明されている特権を取り消します。

ALTER
ALTER FUNCTION、ALTER PROCEDURE、または COMMENT ステートメントを使用するための特権を取り消します。

EXECUTE
関数またはプロシージャーを実行するための特権を取り消します。

FUNCTION または SPECIFIC FUNCTION
特権が取り除かれる関数を指定します。その関数は現行サーバーに存在していて、ユーザー定義関数でなければなりません。ただし、特殊タイプの作成時に暗黙に生成された関数であってはなりません。関数は、それぞれその名前、関数シグニチャー、あるいは特定名によって識別することができます。

FUNCTION *function-name*
関数を名前によって識別します。 *function-name* は厳密に 1 つの関数を示す必要があります。この関数には、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前の関数が複数ある場合、エラーが戻されます。

FUNCTION *function-name (parameter-type, ...)*
関数を一意的に識別する関数シグニチャーによって、関数を識別します。 *function-name (parameter-type,...)* は、指定された関数シグニチャーを持つ関数を識別しなければなりません。指定されたパラメーターは、関数の作成時に指定された、対応する位置にあるデータ・タイプと一致していなければなりません。特権が取り除かれる関数インスタンスを識別する場合、データ・タイプの数とデータ・タイプの論理連結が使用されます。データ・タイプの同義語は、一致として扱われます。デフォルトがあるパラメーターは、このシグニチャーに含まれていなければなりません。

function-name () を指定する場合、識別される関数にパラメーターを使用することはできません。

function-name
関数の名前を識別します。

(parameter-type, ...)
関数のパラメーターを識別します。

非修飾の特殊タイプ名を指定する場合、データベース・マネージャーはその特殊タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、以下のいずれかを使用します。

- 空の括弧は、データベース・マネージャーがデータ・タイプの一致の判別に際して属性を無視することを示します。例えば、DEC() は、DEC(7,2) のデータ・タイプで定義された関数のパラメーターに一致するものとみなされます。ただし、FLOAT に空の括弧を指定することはできません。これは、そのパラメーター値が特定のデータ・タイプ (REAL または DOUBLE) を示しているからです。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を指定した場合、その値は、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプが FLOAT の場合、突き合わせはデータ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度は指定された値に厳密に一致している必要はありません。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。暗黙の長さは、CREATE FUNCTION ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

FOR DATA 文節または CCSID 文節の指定はオプショナルです。いずれの文節も指定しないと、データ・タイプが一致するかどうかを判定する場合に、データベース・マネージャーが属性を無視することを示します。どちらか一方の文節を指定する場合は、CREATE FUNCTION ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

AS LOCATOR
関数が、このパラメーターのロケーターを受け取るように定義されることを示します。 AS LOCATOR を指定する場合は、データ・タイプは LOB または XML、あるいは LOB または XML に基づく特殊タイプでなければなりません。

SPECIFIC FUNCTION *specific-name*
関数を特定名によって識別します。 *specific-name* は、現行サーバーに存在する特定の関数を示している必要があります。

PROCEDURE または SPECIFIC PROCEDURE
特権が取り除かれるプロシージャーを指定します。このプロシージャー名 は、現行サーバーに存在しているプロシージャーを識別していなければなりません。

PROCEDURE *procedure-name*
プロシージャーを名前によって識別します。プロシージャー名 は、ただ 1 つのプロシージャーを識別していなければなりません。このプロシージャーには、パラメーターをいくつでも定義することができます。指定されたスキーマまたは暗黙のスキーマの中に、指定された名前のプロシージャーが複数ある場合、エラーが戻されます。

PROCEDURE *procedure-name (parameter-type, ...)*

プロシージャーを一意的に識別するプロシージャー・シグニチャーによって、プロシージャーを識別します。 *procedure-name (parameter-type,...)* では、指定されたプロシージャー・シグニチャーを持つプロシージャーを識別する必要があります。指定されたパラメーターは、プロシージャーの作成時に指定された、対応する位置にあるデータ・タイプと一致していなければなりません。取り除くプロシージャー・インスタンスを識別する場合、データ・タイプの数とデータ・タイプの論理連結が使用されます。データ・タイプの同義語は、一致として扱われます。デフォルトがあるパラメーターは、このシグニチャーに含まれていなければなりません。

プロシージャー名 *()* を指定する場合、識別されるプロシージャーにパラメーターを使用することはできません。

procedure-name

プロシージャーの名前を識別します。

(parameter-type, ...)

プロシージャーのパラメーターを識別します。

非修飾の特殊タイプ名または配列タイプ名を指定する場合、データベース・マネージャーはその特殊タイプまたは配列タイプのスキーマ名を解決するための SQL パスを検索します。

長さ属性、精度属性、あるいは位取り属性があるデータ・タイプの場合、以下のいずれかを使用します。

- 空の括弧は、データベース・マネージャーがデータ・タイプの一致の判別に際して属性を無視することを示します。例えば、DEC() は、DEC(7,2) のデータ・タイプで定義されたプロシージャーのパラメーターに一致するものとみなされます。ただし、FLOAT に空の括弧を指定することはできません。これは、そのパラメーター値が特定のデータ・タイプ (REAL または DOUBLE) を示しているからです。
- 長さ属性、精度属性、あるいは位取り属性に特定の値を指定する場合、その値は、 CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。データ・タイプが FLOAT の場合、突き合わせはデータ・タイプ (REAL または DOUBLE) に基づいて行われるので、精度は指定された値に厳密に一致している必要はありません。
- 長さ属性、精度属性、または位取り属性が明示的に指定されておらず、空の括弧も指定されていない場合、該当のデータ・タイプのデフォルト属性が暗黙指定されます。暗黙の長さは、CREATE PROCEDURE ステートメントの中で暗黙的または明示的に指定された値と正確に一致している必要があります。

FOR DATA 文節または CCSID 文節の指定はオプショナルです。いずれの文節も指定しないと、データ・タイプが一致するかどうかを判定する場合に、データベース・マネージャーが属性を無視することを示します。どちらか一方の文節を指定する場合は、CREATE PROCEDURE ステートメントに暗黙的または明示的に指定されている値と一致させる必要があります。

AS LOCATOR
プロシージャーが、このパラメーターのロケーターを受け取るように定義されることを示します。 AS LOCATOR を指定する場合は、データ・タイプは LOB または XML、あるいは LOB または XML に基づく特殊タイプでなければなりません。

SPECIFIC PROCEDURE *specific-name*
プロシージャーを特定名によって識別します。特定名 は、現行サーバーに存在している特定のプロシージャーを識別していなければなりません。

FROM
特権を取り消すユーザーを識別します。

USER
authorization-name がユーザー・プロファイルであることを指定します。USER が指定される場合、*authorization-name* はユーザー・プロファイルでなければなりません。

GROUP
authorization-name がグループ・プロファイルであることを指定します。GROUP が指定される場合、*authorization-name* はグループ・プロファイルでなければなりません。

authorization-name,...
1 つ以上の権限 ID をリストします。同じ権限名 は、複数回指定してはなりません。

PUBLIC
PUBLIC に対する特権の付与を取り消します。詳しくは、 21 ページの『権限、特権、およびオブジェクト所有権』を参照してください。

注

複数の認可: 関数またはプロシージャーに対する特権を取り消した場合は、どのユーザーが認可を行ったかには関係なく、その関数またはプロシージャーに対する特権の認可はすべて無効になります。

WITH GRANT OPTION の取り消し: WITH GRANT OPTION を取り消す唯一の方法は、ALL を指定して取り消すことです。

特権の警告: ユーザーから特定の特権を取り消しても、そのユーザーがその特権を必要とする操作を実行できなくなるとは限りません。例えば、そのユーザーは引き続き PUBLIC による特権またはデータベース管理者権限を持つ場合があります。

対応するシステム権限: 関数またはプロシージャーの特権を取り消すと、対応するシステム権限が取り消されます。SQL 特権に対応するシステム権限の説明については、 1517 ページの『GRANT (関数特権またはプロシージャー特権)』を参照してください。

SQL、または外部関数か外部プロシージャーに関して取り消された特権は、その関連のプログラム (*PGM) オブジェクトまたはサービス・プログラム (*SRVPGM) オブジェクトについて取り消されます。 Java 外部関数またはプロシージャーに関して取り消された特権は、関連のクラス・ファイルまたは jar ファイルに関して取り

消されます。取り消しの実行時に関連プログラム、サービス・プログラム、クラス・ファイル、または jar ファイルが見つからない場合、エラーが戻されます。

代替構文: 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- EXECUTE の同義語としてキーワード RUN を使用することができます。

例

PUBLIC に対するプロシージャー PROCA に関する EXECUTE 特権を取り消します。

```
REVOKE EXECUTE
  ON PROCEDURE PROCA
  FROM PUBLIC
```

REVOKE (パッケージ特権)

この形式の REVOKE ステートメントは、パッケージに対する特権を除去します。

呼び出し

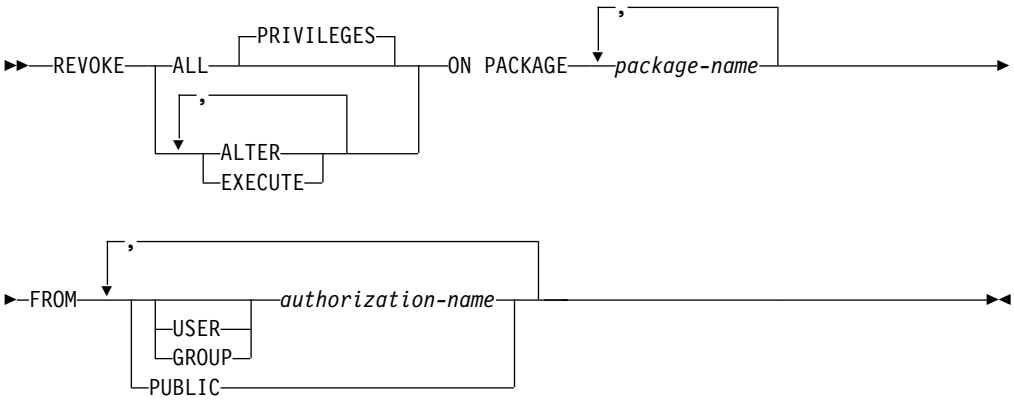
このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれのパッケージごとに、
  - このステートメントで指定されるすべての特権
  - パッケージに対する *OBJMGT システム権限
  - パッケージが含まれるスキーマに対する USAGE 特権
- データベース管理者権限
- セキュリティー管理者権限

構文



説明

ALL または ALL PRIVILEGES

各権限名 から 1 つ以上のパッケージ特権を取り消します。取り消される特権は、識別されたパッケージに関して、権限名 に認可されていた特権です。パッケージに対する ALL PRIVILEGES を取り消すのは、*ALL システム権限を取り消すのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つ以上を使用する必要があります。各キーワードは、そこで説明されている特権を取り消します。

ALTER

COMMENT および LABEL ステートメントを使用する特権を取り消します。

EXECUTE
パッケージ内のステートメントを実行する特権を取り消します。

ON PACKAGE *package-name*
EXECUTE 特権が取り消されるパッケージを指定します。このパッケージ名は、現行サーバーに存在しているパッケージを識別していなければなりません。

FROM
特権を取り消すユーザーを識別します。

USER
authorization-name がユーザー・プロファイルであることを指定します。USER が指定される場合、*authorization-name* はユーザー・プロファイルでなければなりません。

GROUP
authorization-name がグループ・プロファイルであることを指定します。GROUP が指定される場合、*authorization-name* はグループ・プロファイルでなければなりません。

authorization-name,...
1 つ以上の権限 ID をリストします。同じ権限名 は、複数回指定してはなりません。

PUBLIC
PUBLIC に対する特権の付与を取り消します。詳しくは、 21 ページの『権限、特権、およびオブジェクト所有権』を参照してください。

注

複数の認可: パッケージに対する特権を取り消した場合は、どのユーザーが認可を行ったかには関係なく、そのパッケージに対する特権の認可はすべて無効になります。

WITH GRANT OPTION の取り消し: WITH GRANT OPTION を取り消す唯一の方法は、ALL を指定して取り消すことです。

特権の警告: ユーザーから特定の特権を取り消しても、そのユーザーがその特権を必要とする操作を実行できなくなるとは限りません。例えば、そのユーザーは引き続き PUBLIC による特権またはデータベース管理者権限を持つ場合があります。

対応するシステム権限: パッケージ特権を取り消すと、対応するシステム権限が取り消されます。 SQL 特権に対応するシステム権限の説明については、 1526 ページの『GRANT (パッケージ特権)』を参照してください。

代替構文: 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- EXECUTE の同義語としてキーワード RUN を使用することができます。
- PACKAGE の同義語として、キーワード PROGRAM を使用することができます。

例

例 *1:* PUBLIC から、パッケージ PKGA に関する EXECUTE 特権を取り消します。

```
REVOKE EXECUTE
  ON PACKAGE PKGA
  FROM PUBLIC
```

例 *2:* ユーザー FRANK および PUBLIC から、パッケージ RRSP_PKG に関する EXECUTE 特権を取り消します。

```
REVOKE EXECUTE
  ON PACKAGE RRSP_PKG
  FROM FRANK, PUBLIC
```

REVOKE (スキーマ特権)

この形式の REVOKE ステートメントは、スキーマに対する特権を除去します。

呼び出し

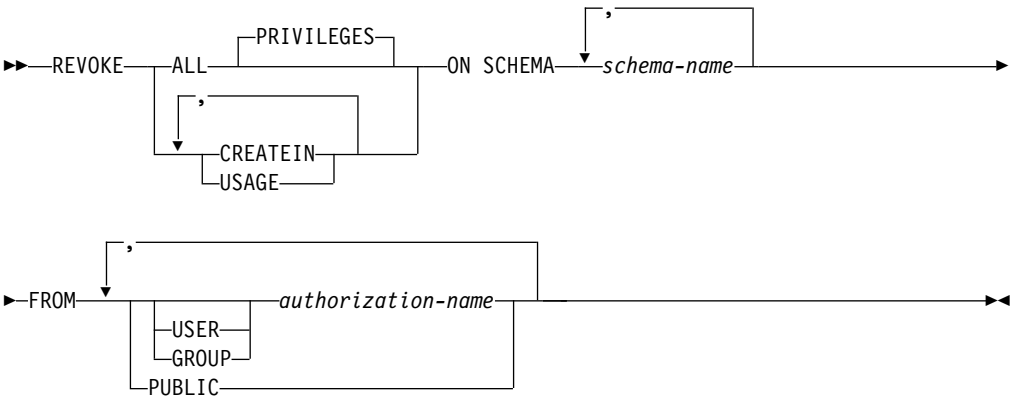
このステートメントは、アプリケーション・プログラムに組み込むことも、あるいは対話式に実行することもできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれのスキーマごとに、
  - このステートメントで指定されるすべての特権
  - スキーマに対する *OBJMGT のシステム権限
- データベース管理者権限
- セキュリティー管理者権限

構文



説明

ALL または ALL PRIVILEGES
各 *authorization-name* から 1 つ以上のスキーマ特権を取り消します。取り消される特権は、識別されたスキーマに関して、*authorization-name* に認可されていた特権です。スキーマに対する ALL PRIVILEGES を取り消すのは、*ALL システム権限を取り消すのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つ以上を使用する必要があります。各キーワードは、そこで説明されている特権を取り消します。

CREATEIN
スキーマにオブジェクトを作成する特権を取り消します。

USAGE
スキーマを使用する特権を取り消します。スキーマに存在するオブジェクトの参照には、USAGE 特権が必要です。

ON SCHEMA *schema-name*
特権を取り消すスキーマを指定します。

FROM
特権を取り消すユーザーを識別します。

USER
authorization-name がユーザー・プロファイルであることを指定します。USER が指定される場合、*authorization-name* はユーザー・プロファイルでなければなりません。

GROUP
authorization-name がグループ・プロファイルであることを指定します。GROUP が指定される場合、*authorization-name* はグループ・プロファイルでなければなりません。

authorization-name,...
1 つ以上の権限 ID をリストします。同じ権限名 は、複数回指定してはなりません。

PUBLIC
PUBLIC に対する特権の付与を取り消します。詳しくは、 21 ページの『権限、特権、およびオブジェクト所有権』を参照してください。

注

複数の認可: スキーマに対する特権を取り消した場合は、どのユーザーが認可を行ったかには関係なく、そのスキーマに対する特権の認可はすべて無効になります。

WITH GRANT OPTION の取り消し: WITH GRANT OPTION を取り消す唯一の方法は、ALL を指定して取り消すことです。

特権の警告: ユーザーから特定の特権を取り消しても、そのユーザーがその特権を必要とする操作を実行できなくなるとは限りません。例えば、そのユーザーは引き続き PUBLIC による特権またはデータベース管理者権限を持つ場合があります。

対応するシステム権限: スキーマ特権を取り消すと、対応するシステム権限が取り消されます。SQL 特権に対応するシステム権限の説明については、 1529 ページの『GRANT (スキーマ特権)』を参照してください。

例

例 *1:* スキーマ T_SCORES に対する CREATEIN 特権をユーザー JONES から取り消します。

```
REVOKE CREATEIN
  ON SCHEMA T_SCORES
  FROM JONES;
```

REVOKE (シーケンス特権)

この形式の REVOKE ステートメントは、シーケンスに対する特権を除去します。

呼び出し

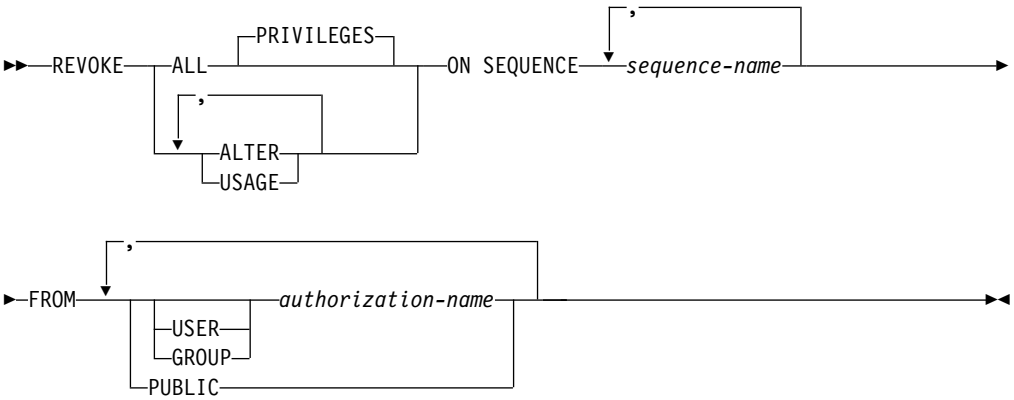
このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれのシーケンスごとに、
  - このステートメントで指定されるすべての特権
  - シーケンスに対する *OBJMGT システム権限
  - シーケンスが含まれるスキーマに対する USAGE 特権
- データベース管理者権限
- セキュリティー管理者権限

構文



説明

ALL または ALL PRIVILEGES

各権限名 から 1 つ以上のシーケンス特権を取り消します。取り消される特権は、識別されたシーケンスに関して、権限名 に認可されていた特権です。シーケンスに対する ALL PRIVILEGES を取り消すのは、*ALL システム権限を取り消すのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つ以上を使用する必要があります。各キーワードは、そこで説明されている特権を取り消します。

ALTER

シーケンスに対する ALTER SEQUENCE、COMMENT、および LABEL ステートメントを使用する特権を取り消します。

USAGE
NEXT VALUE または PREVIOUS VALUE 式内のシーケンスを使用するための特権を取り消します。

ON SEQUENCE *sequence-name*
特権が取り消されるシーケンスを指定します。シーケンス名 は、現行サーバーに存在するシーケンスを示すものでなければなりません。

FROM
特権を取り消すユーザーを識別します。

USER
authorization-name がユーザー・プロファイルであることを指定します。USER が指定される場合、*authorization-name* はユーザー・プロファイルでなければなりません。

GROUP
authorization-name がグループ・プロファイルであることを指定します。GROUP が指定される場合、*authorization-name* はグループ・プロファイルでなければなりません。

authorization-name,...
1 つ以上の権限 ID をリストします。同じ権限名 は、複数回指定してはなりません。

PUBLIC
PUBLIC に対する特権の付与を取り消します。詳しくは、 21 ページの『権限、特権、およびオブジェクト所有権』を参照してください。

注

複数の認可: シーケンスに対する特権を取り消した場合は、どのユーザーが認可を行ったかには関係なく、そのシーケンスに対する特権の認可はすべて無効になります。

WITH GRANT OPTION の取り消し: WITH GRANT OPTION を取り消す唯一の方法は、ALL を指定して取り消すことです。

特権の警告: ユーザーから特定の特権を取り消しても、そのユーザーがその特権を必要とする操作を実行できなくなるとは限りません。例えば、そのユーザーは引き続き PUBLIC による特権またはデータベース管理者権限を持つ場合があります。

対応するシステム権限: シーケンス特権を取り消すと、対応するシステム権限が取り消されます。 SQL 特権に対応するシステム権限の説明については、 1532 ページの『GRANT (シーケンス特権)』を参照してください。

例

ORG_SEQ と呼ばれるシーケンスに対する USAGE 特権を PUBLIC から取り消します。

```
REVOKE USAGE
  ON SEQUENCE ORG_SEQ
  FROM PUBLIC
```

REVOKE (表またはビューの特権)

この形式の REVOKE ステートメントは、表またはビューに対する特権を除去します。

呼び出し

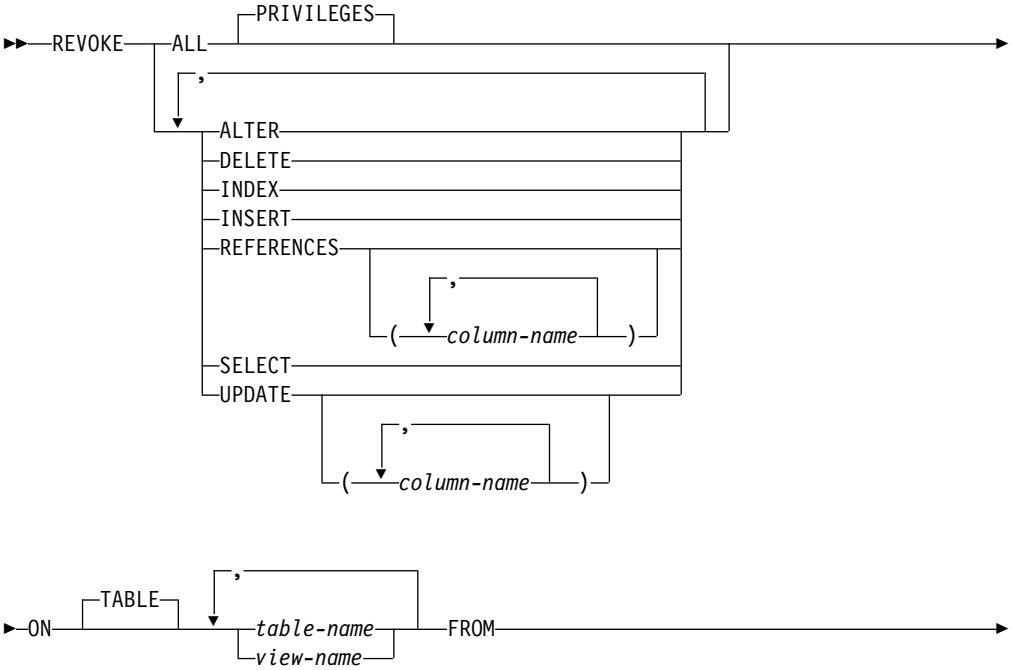
このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

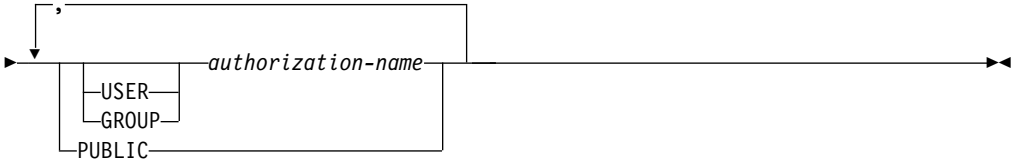
権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれの表またはビューごとに、
  - このステートメントで指定されるすべての特権
  - その表またはビューに対する *OBJMGT システム権限
  - 表またはビューが含まれるスキーマに対する USAGE 特権
- データベース管理者権限
- セキュリティー管理者権限

構文





説明

ALL または ALL PRIVILEGES
各権限名 から 1 つ以上の特権を取り消します。取り消される特権は、識別された表およびビューに関して、権限名 に認可されていた特権です。表またはビューに対する ALL PRIVILEGES を取り消すのは、*ALL システム権限を取り消すのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つ以上を使用する必要があります。各キーワードはそこで説明されている特権を取り消しますが、ON 文節で指定された表およびビューに当てはまる特権だけが取り消されます。

ALTER
指定した表を変更する特権や、指定した表またはビューに対してコメントを追加したり、ラベルを追加したり、索引を作成したりするための特権を取り消します。

DELETE
指定の表またはビューから行を削除する特権を取り消します。

INDEX
指定の表に索引を作成する特権を取り消します。

INSERT
指定した表またはビューに行を挿入するための特権を取り消します。

REFERENCES
その表が親になる参照制約を追加する特権を取り消します。

REFERENCES (*column-name*,...)
親キーで指定された列を使用して参照制約を追加する特権を取り消します。それぞれの列名は、ON 文節に指定されている各表の列を識別する非修飾の名前でなければなりません。

SELECT
SELECT または CREATE VIEW ステートメントを使用する特権を取り消します。

UPDATE
UPDATE ステートメントを使用する特権を取り消します。

UPDATE (*column-name*,...)
指定された列を更新する特権を取り消します。それぞれの列名は、ON 文節に指定されている各表の列を識別する非修飾の名前でなければなりません。

ON *table-name* または *view-name, ...*
特権を取り消す表またはビューを指定します。表名 またはビュー名 は、現行サーバーにある表またはビューを示すものでなければなりませんが、宣言済み一時表を示すものであってはなりません。

FROM
特権を取り消すユーザーを識別します。

USER
authorization-name がユーザー・プロファイルであることを指定します。USER が指定される場合、*authorization-name* はユーザー・プロファイルでなければなりません。

GROUP
authorization-name がグループ・プロファイルであることを指定します。GROUP が指定される場合、*authorization-name* はグループ・プロファイルでなければなりません。

authorization-name,...
1 つ以上の権限 ID をリストします。同じ権限名 は、複数回指定してはなりません。

PUBLIC
PUBLIC に対する特権の付与を取り消します。詳しくは、 21 ページの『権限、特権、およびオブジェクト所有権』を参照してください。

注

複数の認可**:** 同じ特権が同じユーザーに対して複数回認可されている場合は、そのユーザーからその特権を取り消すと、それらの認可はすべて無効になります。

ある特権を取り消すと、その特権がどのようなユーザーに認可されているかには関係なく、その特権の認可がすべて取り消されます。

WITH GRANT OPTION の取り消し**:** WITH GRANT OPTION を取り消す唯一の方法は、ALL を指定して取り消すことです。

特権の警告**:** ユーザーから特定の特権を取り消しても、そのユーザーがその特権を必要とする操作を実行できなくなるとは限りません。例えば、そのユーザーは引き続き PUBLIC による特権またはデータベース管理者権限を持つ場合があります。

複数のシステム権限を 1 つの SQL 特権の取り消しで取り消す場合、それらのシステム権限の中に 1 つでも取り消すことができないものがあると、警告が出され、その特権の取り消しでは権限は取り消されません。

対応するシステム権限**:** 表特権を取り消すと、対応するシステム権限が取り消されます。ただし、以下の例外があります。

- 表またはビューに対する特権を取り消した際に、*OBJOPR が取り消されるのは、 *ADD、*DLT、 *READ、および *UPD もすべて取り消された場合だけです。
- ビューに対する権限を取り消す場合、そのビューの定義の全選択で参照されている、どのような表やビューからも権限は取り消されません。

SQL 特権に対応するシステム権限の説明については、 1535 ページの『GRANT (表またはビューの特権)』を参照してください。

INDEX または ALTER 特権のいずれかを取り消すと、システム権限 *OBJALTER が取り消されます。

例

例 *1:* 表 EMPLOYEE に対する SELECT 特権を、ユーザー ENGLES から取り消します。

```
REVOKE SELECT
  ON TABLE EMPLOYEE
  FROM ENGLES
```

例 *2:* 以前にはすべてのユーザーに認可していた表 EMPLOYEE に対する更新特権を取り消します。特定のユーザーに対する認可には、影響を与えないことに注意してください。

```
REVOKE UPDATE
  ON TABLE EMPLOYEE
  FROM PUBLIC
```

例 *3:* 表 EMPLOYEE に対するすべての特権を、ユーザー PELLOW および ANDERSON から取り消します。

```
REVOKE ALL
  ON TABLE  EMPLOYEE
  FROM PELLOW, ANDERSON
```

例 *4:* VIEW1 の column_1 を更新する特権を、FRED から取り消します。

```
REVOKE UPDATE(column_1)
  ON VIEW1
  FROM FRED
```

REVOKE (タイプ特権)

この形式の REVOKE ステートメントは、タイプに対する特権を除去します。

呼び出し

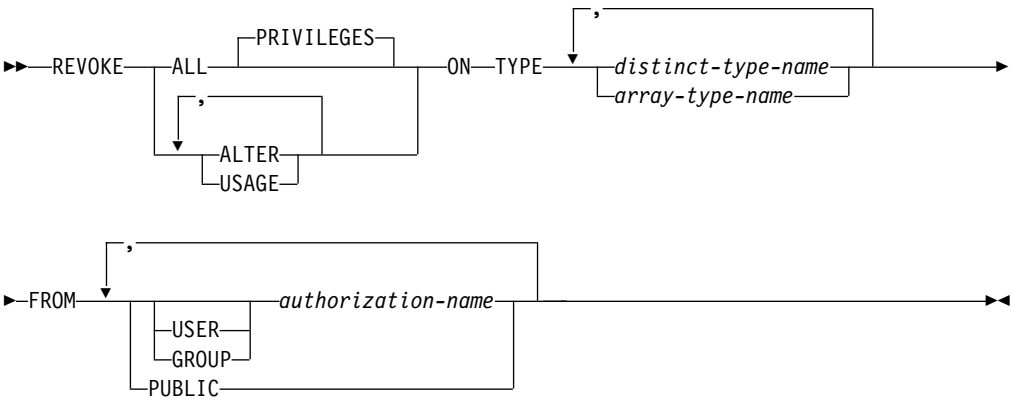
このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれの特殊タイプまたは配列タイプごとに、
  - このステートメントで指定されるすべての特権
  - タイプに対する *OBJMGT のシステム権限
  - タイプが含まれるスキーマに対する USAGE 特権
- データベース管理者権限
- セキュリティー管理者権限

構文



説明

ALL または ALL PRIVILEGES
各権限名 から 1 つ以上のタイプ特権を取り消します。取り消される特権は、識別されたユーザー定義タイプに関して、権限名 に認可されていた特権です。タイプに対する ALL PRIVILEGES を取り消すのは、*ALL システム権限を取り消すのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つ以上を使用する必要があります。各キーワードは、そこで説明されている特権を取り消します。

ALTER
COMMENT および LABEL ステートメントを使用する特権を取り消します。

USAGE
表、関数、プロシージャー内で、あるいは CREATE TYPE ステートメントの中のソース・タイプとしてユーザー定義タイプを使用するための特権を取り消します。

ON TYPE *distinct-type-name* または *array-type-name*
特権が取り除かれる特殊タイプを指定します。特殊タイプ名 または配列タイプ名 は、現行サーバーに存在するユーザー定義タイプを識別する必要があります。

FROM
特権を取り消すユーザーを識別します。

USER
authorization-name がユーザー・プロファイルであることを指定します。USER が指定される場合、*authorization-name* はユーザー・プロファイルでなければなりません。

GROUP
authorization-name がグループ・プロファイルであることを指定します。GROUP が指定される場合、*authorization-name* はグループ・プロファイルでなければなりません。

authorization-name,...
1 つ以上の権限 ID をリストします。同じ権限名 は、複数回指定してはなりません。

PUBLIC
PUBLIC に対する特権の付与を取り消します。詳しくは、 21 ページの『権限、特権、およびオブジェクト所有権』を参照してください。

注

複数の認可: 権限 ID A が同じ特権を権限 ID B に対して複数回認可した場合は、B からその特権を取り消すと、それらの認可はすべて無効になります。

WITH GRANT OPTION の取り消し: WITH GRANT OPTION を取り消す唯一の方法は、ALL を指定して取り消すことです。

特権の警告: ユーザーから特定の特権を取り消しても、そのユーザーがその特権を必要とする操作を実行できなくなるとは限りません。例えば、そのユーザーは引き続き PUBLIC による特権またはデータベース管理者権限を持つ場合があります。

対応するシステム権限: タイプ特権を取り消すと、対応するシステム権限が取り消されます。SQL 特権に対応するシステム権限の説明については、 1542 ページの『GRANT (タイプ特権)』を参照してください。

代替構文: 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード DATA TYPE または DISTINCT TYPE を TYPE の同義語として使用することができます。

例

特殊タイプ SHOESIZE に関する USAGE 特権を、ユーザー JONES から取り消します。

```
REVOKE USAGE
  ON DISTINCT TYPE SHOESIZE
  FROM JONES
```

REVOKE (変数特権)

この形式の REVOKE ステートメントは、作成されたグローバル変数に対する特権を除去します。

呼び出し

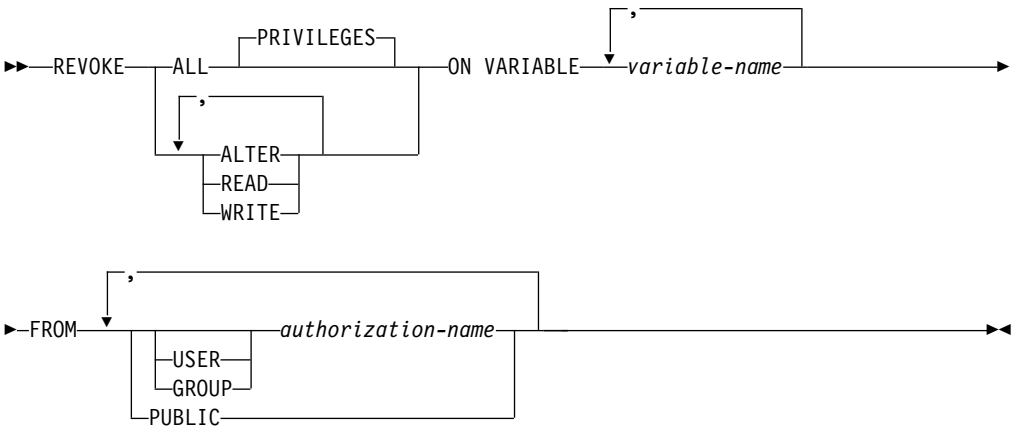
このステートメントは、アプリケーション・プログラムに組み込むことも、あるいは対話式に実行することもできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれの変数ごとに、
  - このステートメントで指定されるすべての特権
  - グローバル変数に対する *OBJMGT システム権限
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限
- セキュリティー管理者権限

構文



説明

ALL または **ALL PRIVILEGES**

各権限名 から 1 つ以上のグローバル変数特権を取り消します。取り消される特権は、識別されたグローバル変数に関して、権限名 に認可されていた特権です。グローバル変数に対する ALL PRIVILEGES を取り消すのは、*ALL システム権限を取り消すのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つ以上を使用する必要があります。各キーワードは、そこで説明されている特権を取り消します。

ALTER
指定したグローバル変数に対して、COMMENT および LABEL ステートメントを使用する特権を取り消します。

READ
指定したグローバル変数の値を読み取る特権を取り消します。

WRITE
指定したグローバル変数に値を割り当てる特権を取り消します。

ON VARIABLE *variable-name*
1 つ以上の特権を取り消すグローバル変数を指定します。各変数名 は、現行サーバーに存在しているグローバル変数を識別していなければなりません。

FROM
特権を取り消すユーザーを識別します。

USER
authorization-name がユーザー・プロファイルであることを指定します。USER が指定される場合、*authorization-name* はユーザー・プロファイルでなければなりません。

GROUP
authorization-name がグループ・プロファイルであることを指定します。GROUP が指定される場合、*authorization-name* はグループ・プロファイルでなければなりません。

authorization-name,...
1 つ以上の権限 ID をリストします。同じ権限名 は、複数回指定してはなりません。

PUBLIC
PUBLIC に対する特権の付与を取り消します。詳しくは、 21 ページの『権限、特権、およびオブジェクト所有権』を参照してください。

注

複数の認可: 変数に対する特権を取り消した場合は、どのユーザーが認可を行ったかには関係なく、その変数に対する特権の認可はすべて無効になります。

WITH GRANT OPTION の取り消し: WITH GRANT OPTION を取り消す唯一の方法は、ALL を指定して取り消すことです。

特権の警告: ユーザーから特定の特権を取り消しても、そのユーザーがその特権を必要とする操作を実行できなくなるとは限りません。例えば、そのユーザーは引き続き PUBLIC による特権またはデータベース管理者権限を持つ場合があります。

対応するシステム権限: グローバル変数特権を取り消すと、対応するシステム権限が取り消されます。SQL 特権に対応するシステム権限の説明については、 1545 ページの『GRANT (変数特権)』を参照してください。

例

ユーザー ZUBIRI からグローバル変数 MYSCHEMA.MYJOB_PRINTER における WRITE 特権を取り消します。

```
REVOKE WRITE
  ON VARIABLE MYSCHEMA.MYJOB_PRINTER
  FROM ZUBIRI
```

REVOKE (XML スキーマ特権)

この形式の REVOKE ステートメントは、XSR オブジェクトに対する特権を除去します。

呼び出し

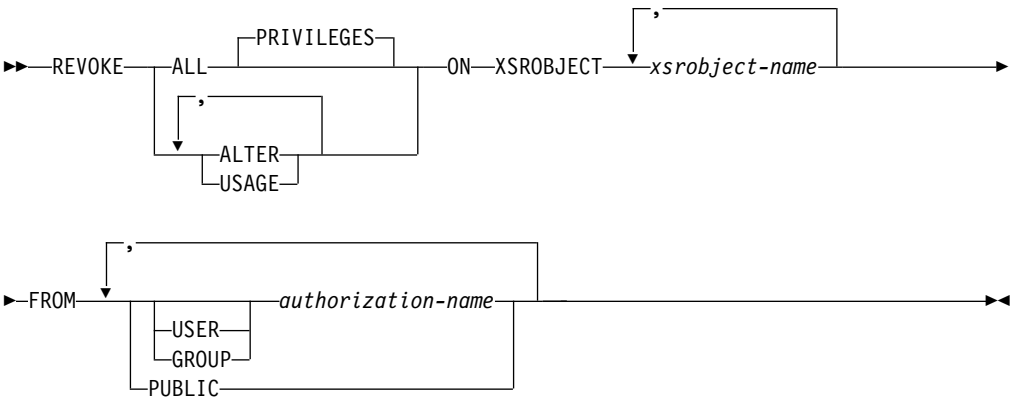
このステートメントは、アプリケーション・プログラムに組み込むことも、あるいは対話式に実行することもできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメント内で識別された、それぞれの XSR オブジェクトごとに、
  - このステートメントで指定されるすべての特権
  - XSR オブジェクトに対する *OBJMGT システム権限
  - XSR オブジェクトが含まれるスキーマに対する USAGE 特権
- データベース管理者権限
- セキュリティー管理者権限

構文



説明

ALL または ALL PRIVILEGES

各権限名 から 1 つ以上の XSR オブジェクト特権を取り消します。取り消される特権は、識別された XSR オブジェクトに関して、権限名 に認可されていた特権です。XSR オブジェクトに対する ALL PRIVILEGES を取り消すのは、*ALL システム権限を取り消すのと同じではないことに注意する必要があります。

ALL を使用しない場合には、以下にリストしたキーワードの 1 つ以上を使用する必要があります。各キーワードは、そこで説明されている特権を取り消します。

ALTER
COMMENT および LABEL ステートメントを使用する特権を取り消します。

USAGE
XSR オブジェクトを使用する特権を取り消します。

ON XSROBJECT *xsrobject-name*
特権が取り除かれる XSR オブジェクトを指定します。この*XSR* オブジェクト名 は、現行サーバーに存在しているXSR オブジェクトを示すものでなければなりません。

FROM
特権を取り消すユーザーを識別します。

USER
authorization-name がユーザー・プロファイルであることを指定します。USER が指定される場合、*authorization-name* はユーザー・プロファイルでなければなりません。

GROUP
authorization-name がグループ・プロファイルであることを指定します。GROUP が指定される場合、*authorization-name* はグループ・プロファイルでなければなりません。

authorization-name,...
1 つ以上の権限 ID をリストします。同じ権限名 は、複数回指定してはなりません。

PUBLIC
PUBLIC に対する特権の付与を取り消します。詳しくは、 21 ページの『権限、特権、およびオブジェクト所有権』を参照してください。

注

複数の認可: XSR オブジェクトに対する特権を取り消した場合は、どのユーザーが認可を行ったかには関係なく、その XSR オブジェクトに対する特権の認可はすべて無効になります。

WITH GRANT OPTION の取り消し: WITH GRANT OPTION を取り消す唯一の方法は、ALL を指定して取り消すことです。

特権の警告: ユーザーから特定の特権を取り消しても、そのユーザーがその特権を必要とする操作を実行できなくなるとは限りません。例えば、そのユーザーは引き続き PUBLIC による特権またはデータベース管理者権限を持つ場合があります。

対応するシステム権限: XSR オブジェクト特権を取り消すと、対応するシステム権限が取り消されます。SQL 特権に対応するシステム権限の説明については、1548 ページの『GRANT (XML スキーマ特権)』を参照してください。

例

PUBLIC から XSR オブジェクト XMLSCHEMA における USAGE 特権を取り消します。

```
REVOKE USAGE
  ON XSRSOBJECT XMLSCHEMA
  FROM PUBLIC
```

ROLLBACK

ROLLBACK ステートメントは、変更をバックアウトするために使用します。

ROLLBACK ステートメントは次の目的に使用できます。

- 作業単位を終了させ、その作業単位でリレーショナル・データベースに対して行われたすべての変更をバックアウトする。アプリケーション・プロセスが使用しているリカバリー可能リソースがリレーショナル・データベースだけである場合は、ROLLBACK は作業単位も終了します。
- 作業単位を終了させずに、その作業単位内に設定されたセーブポイント以降に行われた変更のみをバックアウトする。セーブポイントまでのロールバックにより、選択した変更を取り消すことができます。

呼び出し

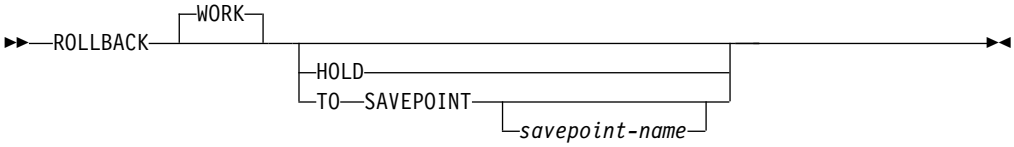
このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

トリガー・プログラムとその対象となるプログラムが同じコミットメント定義のもとで実行される場合、トリガーでは ROLLBACK は許されません。リモート・アプリケーション・サーバーへの分散作業単位接続時に呼び出されるプロシージャー、または ATOMIC として定義されているプロシージャーでは、ROLLBACK を使用することはできません。関数内では ROLLBACK を使用できません。

権限

権限は不要です。

構文



説明

TO SAVEPOINT 文節なしの ROLLBACK を使用すると、このステートメントが実行される作業単位が終了します。作業単位の中で SQL スキーマ・ステートメントおよび SQL データ変更ステートメントにより行われたすべての変更がバックアウトされます。詳しくは、 865 ページの『第 7 章 ステートメント』を参照してください。

識別値の生成は、トランザクションの制御下にありません。識別列を持つ表への行の挿入によって生成され、消費される値は、ROLLBACK ステートメントの実行に依存しません。また、ROLLBACK ステートメントの実行は、IDENTITY_VAL_LOCAL 関数に影響を与えません。

特殊レジスターは、トランザクションの制御下にありません。 ROLLBACK ステートメントの実行は、特殊レジスターに影響を与えません。

シーケンスは、トランザクションの制御下にありません。 ROLLBACK ステートメントの実行は、NEXT VALUE 式の実行によって生成され、消費される現行値に影響を与えません。

グローバル変数は、トランザクションの制御下にありません。ROLLBACK ステートメントの実行は、インスタンス化されたグローバル変数の値に影響を与えません。

宣言済み一時表の内容に対する ROLLBACK または ROLLBACK TO SAVEPOINT の影響は、DECLARE GLOBAL TEMPORARY TABLE ステートメントの ON ROLLBACK 文節の設定によって決まります。

WORK
ROLLBACK WORK と ROLLBACK の効果は同じです。

HOLD
リソースを保持するように指示します。これを指定すると、現在オープンされているカーソルは、HOLD オプションで宣言されているかどうかに関係なく、クローズされません。作業単位の過程で獲得したリソース (表の行に対するロックは除く) はすべて保持されます。ただし、その作業単位の過程で特定の行に対して暗黙に掛けられたロックは解放されます。

次のような条件が当てはまらない場合、ROLLBACK HOLD が終了したときのカーソル位置は、該当する作業単位を開始したときと同じになります。

- カーソルを含むプログラムまたはルーチンの作成時に ALWBLK (*ALLREAD) が指定された場合
- カーソルを含むプログラムまたはルーチンの作成時に ALWBLK(*READ) および ALWCPYDTA(*OPTIMIZE) が指定された場合

TO SAVEPOINT
作業単位を終了せずに、部分ロールバック (セーブポイントまで) のみを行うことを指定します。セーブポイント名を指定しなかった場合は、最後の活動セーブポイントまでのロールバックが行われます。例えば、ある作業単位の中でセーブポイント A、B、および C がこの順序で設定されているときに、C が解放されたとすると、ROLLBACK TO SAVEPOINT によりセーブポイント B までのロールバックが行われます。アクティブなセーブポイントが存在しない場合は、エラーが戻されます。

savepoint-name
どのセーブポイントまでロールバックするかを指定します。この名前は、現行サーバーに存在しているセーブポイントを識別していなければなりません。

ROLLBACK TO SAVEPOINT が正常に完了した後も、セーブポイントは存続します。

セーブポイントが設定された後で行われたすべてのデータベース変更 (ON ROLLBACK PRESERVE ROWS 文節によって宣言済みの一時表に対する変更も含む) がバックアウトされます。ロックおよび LOB ロケーターはすべて保持されます。

ROLLBACK TO SAVEPOINT によるカーソルへの影響は、セーブポイントに含まれるステートメントによって決まります。

- セーブポイントに、カーソルが依存している SQL スキーマ・ステートメントが含まれている場合は、そのカーソルはクローズされます。 ROLLBACK TO SAVEPOINT の後でこのようなカーソルを使用しようとすると、エラーが起こります。
- その他の場合は、カーソルは ROLLBACK TO SAVEPOINT の影響を受けません (オープンされ、位置付けされたままの状態を維持します)。

ロールバックの対象となったセーブポイントより後で設定された、現行サーバー上のセーブポイントは、すべて解放されます。ロールバックの対象となったセーブポイントは解放されません。

注

推奨されるコーディング方法**:** 明示的な COMMIT または ROLLBACK ステートメントを、アプリケーション・プロセスの最後にコーディングしてください。アプリケーション環境に応じて、暗黙的なコミットまたはロールバック操作のいずれかが、アプリケーション・プロセスの終わりに実行されます。このため、移植可能なアプリケーションでは、明示的な COMMIT または ROLLBACK が許可された環境で実行が終了する前に、COMMIT または ROLLBACK を明示的に実行する必要があります。

ロールバックによるその他の影響**:** TO SAVEPOINT 文節および HOLD 文節を指定せずにロールバックすると、次が発生します。

- 作業単位の実行中にオープンされたすべてのカーソルは、HOLD オプションによって宣言されているかどうかに関係なくクローズされます。
- すべての LOB ロケーター (保持されているものも含む) が解放されます。
- この作業単位のコミットメント定義のもとで獲得されたロックはすべて、解放されます。

ROLLBACK は、接続の状態に影響を与えません。

暗黙的なロールバック**:** デフォルトの活動化グループが終了すると、暗黙のロールバックが行われます。したがって、明示的な COMMIT または ROLLBACK ステートメントは、デフォルトの活動化グループが終了する前に出しておかなければなりません。

次のような場合は、ROLLBACK が自動的に実行されます。

1. デフォルトの活動化グループが最後に COMMIT を出さずに終了した場合。
2. 活動化グループの作業の完了を妨げるような障害 (例えば、電源障害など) が発生した場合。

   障害が起こった時点で COMMIT が進行中であったためにその作業単位が準備状態である場合、ロールバックは行われません。代わりに、その作業単位に関連するすべての接続の再同期化が行われます。詳しくは、「コミットメント制御」トピック集を参照してください。
3. アプリケーション・サーバーとの接続が失われるような障害 (例えば、通信回線の障害など) が発生した場合。

障害が起こった時点で COMMIT が進行中であったためにその作業単位が準備状態である場合、ロールバックは行われません。代わりに、その作業単位に関連するすべての接続の再同期化が行われます。詳しくは、「コミットメント制御」トピック集を参照してください。

4. デフォルトの活動化グループ以外の活動化グループは、異常終了します。

行ロックの制限**:** 1 つの作業単位には、最高 400 万までの行の処理を含めることができますが、これには、SELECT INTO または FETCH ステートメントの過程で取り出された行、および INSERT、DELETE、および UPDATE 操作の一環として挿入、削除、または更新された行も含まれます。[126]

影響されないステートメント**:** コミットおよびロールバック操作が DROP SCHEMA ステートメントに影響することはありません。したがって、このステートメントは、現行の分離レベルがコミット不可 (NC) の場合に使用できます。

ROLLBACK の制約事項**:** 対象の活動化グループについてコミットメント制御が活動状態にない場合は、ROLLBACK ステートメントは使用できません。どのコミットメント定義が使用されているかを判別する方法については、COMMIT ステートメントの項のコミットメント定義に関する説明を参照してください。

2 次スレッドのユーザー定義関数でコミットまたはロールバックは使用できません。

ROLLBACK は、接続の状態に影響を与えません。

1 つの作業単位の中で、CLOSE の後に ROLLBACK を実行すると、その作業単位の中で行われた変更はすべてバックアウトされます。ただし、CLOSE 自体はバックアウトされないので、ファイルが再オープンされることはありません。

例

例 *1:* ROLLBACK ステートメントを使用した例については、 1040 ページの『例』の COMMIT の項を参照してください。

例 *2:* あるリカバリー単位の開始後に、A、B、C の 3 つのセーブポイントが設定され、その後 C が解放されたとします。

```
SAVEPOINT A ON ROLLBACK RETAIN CURSORS;
...
SAVEPOINT B ON ROLLBACK RETAIN CURSORS;
....
SAVEPOINT C ON ROLLBACK RETAIN CURSORS;
...
RELEASE SAVEPOINT C
```

125. COMMIT(*CHG) または COMMIT(*CS) を指定した場合は除きます。指定した場合、これらの行はこの合計数に含まれません。

126. この制約には以下も含まれます。

- 高水準言語のファイル処理機能によるコミットメント制御のもとでオープンされたファイルに基づいてアクセスまたは変更された行。
- トリガー、または CASCADE、SET NULL、あるいは SET DEFAULT 参照保全削除規則の結果として削除、更新、または挿入された行。

セーブポイント A までのデータベース変更のみをすべてロールバックします。

```
ROLLBACK WORK TO SAVEPOINT A
```

セーブポイント名が指定されていない場合 (つまり ROLLBACK WORK TO SAVEPOINT の場合)、最後に設定されたアクティブ・セーブポイント (B) までのロールバックが行われます。

SAVEPOINT

SAVEPOINT ステートメントは、作業単位内にセーブポイントを設定します。セーブポイントは作業単位内の特定時点を表すもので、リレーショナル・データベースに対する変更をその時点までロールバックすることができます。

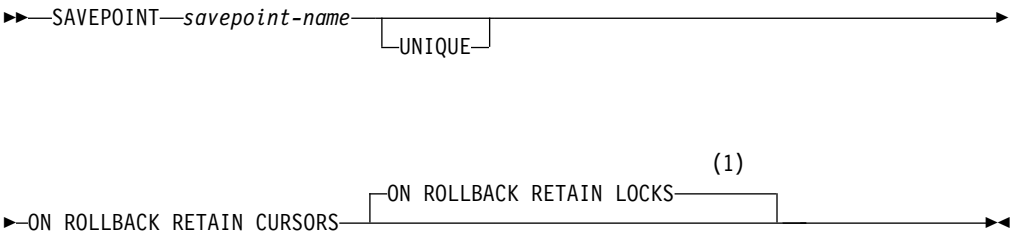
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

権限は不要です。

構文



注:

1 ROLLBACK のオプションは、どのような順序で指定しても構いません。

説明

savepoint-name
新しいセーブポイント を指定します。「SYS」で始まる *savepoint-name* は使用できません。

UNIQUE
アプリケーション・プログラムが、この作業単位内でこのセーブポイント名を再使用できないことを指定します。この作業単位内に、既にセーブポイント名 と同じ名前が存在している場合は、エラーが起こります。

UNIQUE を省略した場合は、アプリケーションが作業単位内でこのセーブポイント名を再使用できることを示します。セーブポイント名 がこの作業単位内の既存のセーブポイントのどれかと同じであっても、その既存のセーブポイントの作成時に UNIQUE オプションが指定されていなければ、その既存のセーブポイントは破棄され、新しいセーブポイントが作成されます。セーブポイントを破棄して、その名前を他のセーブポイントに再使用することは、セーブポイントを解放することとは異なります。 1 つのセーブポイント名を再使用した場合、破棄されるセーブポイントは 1 つだけです。しかし、RELEASE SAVEPOINT ステートメントを使用してセーブポイントの 1 つを解放すると、そのセーブポイント以降に設定されているすべてのセーブポイントが解放されます。

ON ROLLBACK RETAIN CURSORS
このセーブポイントへのロールバックが行われたときに、このセーブポイントの設定後にオープンされたカーソルをクローズしないことを指定します。

- SAVEPOINT ステートメントの有効範囲内の表またはビューに対して SQL スキーマ・ステートメントが実行される場合、その表またはビューを参照するカーソルはすべてクローズされます。 ROLLBACK TO SAVEPOINT の後でこのようなカーソルを使用しようとすると、エラーが起こります。
- その他の場合は、カーソルは ROLLBACK TO SAVEPOINT の影響を受けません (オープンされ、位置付けされたままの状態を維持します)。

上記のカーソルは、セーブポイントへのロールバックの後もオープン状態のままになっていますが、使用不能になることもあります。例えば、セーブポイントへのロールバックが原因で、カーソルが位置付けされている行の挿入がロールバックされることになった場合、カーソルを使用してその行を更新または削除しようとすると、エラーが起こります。

ON ROLLBACK RETAIN LOCKS
このセーブポイントへのロールバックが行われたときに、セーブポイントの設定後に獲得されたロックをどれも解放しないことを指定します。

注

セーブポイントの持続性**:** セーブポイント *S* は次の場合に破棄されます。

- COMMIT ステートメントまたは ROLLBACK (TO SAVEPOINT 節を指定しない) ステートメントを実行する場合。
- ROLLBACK TO SAVEPOINT ステートメントを、セーブポイント *S* を指定して、または作業単位で *S* よりも前に設定されたセーブポイントを指定して実行する場合。
- RELEASE SAVEPOINT ステートメントを、セーブポイント *S* を指定して、または作業単位で *S* よりも前に設定されたセーブポイントを指定して実行する場合。
- SAVEPOINT ステートメントで、UNIQUE キーワードを指定して作成しなかった既存のセーブポイントと同じ名前を指定する場合。

INSERT に対する影響**:** アプリケーションでは、挿入操作がバッファーに入れられることがあります。 SAVEPOINT、ROLLBACK、または RELEASE TO SAVEPOINT ステートメントを実行すると、バッファーはフラッシュされます。

SAVEPOINT の制約事項**:** 対象の活動化グループについてコミットメント制御が活動状態にない場合は、SAVEPOINT ステートメントは使用できません。どのコミットメント定義が使用されているかを判別する方法については、COMMIT ステートメントの 1039 ページの『注』を参照してください。

例

ある作業単位内の 3 箇所にセーブポイントを設定したいとします。第 1 のセーブポイントには A と命名し、このセーブポイント名は再使用できるようにします。第 2 のセーブポイントには B と命名し、この名前は再使用できないようにします。第 3 のセーブポイントが設定可能な状態になった時点ではセーブポイント A は不要になるので、第 3 のセーブポイントの名前として A を再使用します。

```
SAVEPOINT A ON ROLLBACK RETAIN CURSORS;
.
.
.
SAVEPOINT B UNIQUE ON ROLLBACK RETAIN CURSORS;
.
.
.
SAVEPOINT A ON ROLLBACK RETAIN CURSORS;
```

SELECT

SELECT ステートメントは、照会の形式の 1 つです。 SQLJ アプリケーション・プログラムに組み込むことも、対話式に実行することもできます。

詳しくは、 789 ページの『SELECT 文節』および 787 ページの『第 6 章 照会』を参照してください。

SELECT INTO

SELECT INTO ステートメントは、1 行以内で構成される結果表を作成し、その行の値を変数に割り当てます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。これは実行可能ステートメントですが、動的に準備することはできません。REXX で指定してはなりません。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

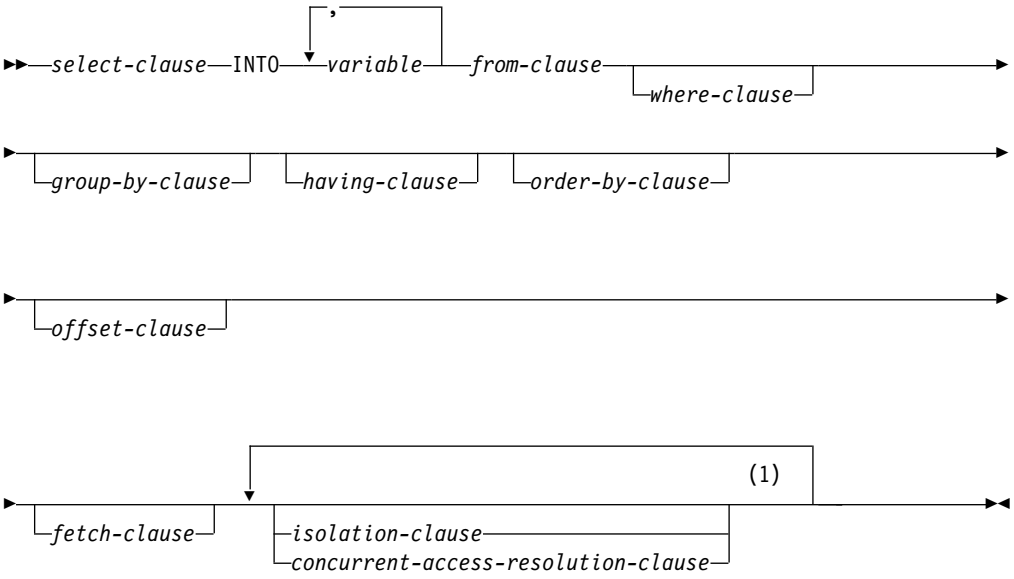
- ステートメント内で識別された、それぞれの表またはビューごとに、
  - 表やビューに対する SELECT 特権、および
  - 表またはビューが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

INTO 変数リスト内にグローバル変数が指定されている場合、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つが含まれていなければなりません。

- グローバル変数に対する WRITE 特権
- データベース管理者権限

SQL 特権に対応するシステム権限については、『表またはビューへの権限を検査する際の対応するシステム権限』を参照してください。

構文



注:

1 各文節はそれぞれ 1 回だけ指定できます。

説明

結果表は、*isolation-clause*、*concurrent-access-resolution-clause*、*from-clause*、*where-clause*、*group-by-clause*、*having-clause*、*order-by-clause*、*offset-clause*、*fetch-clause*、および *select-clause* をこの順序で評価することによって得られます。

select-clause、*from-clause*、*where-clause*、*group-by-clause*、*having-clause*、*order-by-clause*、*offset-clause*、*fetch-clause*、*isolation-clause*、および *concurrent-access-resolution-clause* の説明については、 787 ページの『第 6 章 照会』を参照してください。

INTO *variable*,...
1 つ以上のホスト構造体、あるいは変数を指定します。これらのホスト構造体や変数は、その宣言の規則に従ってプログラムで宣言する必要があります。
INTO 文節の操作形式では、ホスト構造体に対する参照は、その個々の変数に対する参照によって置き換えられます。結果の行の最初の値がリストの最初の変数に割り当てられ、2 番目の値が 2 番目の変数に割り当てられます。以下同様です。各変数のデータ・タイプは、それぞれに対応する列と互換性がなければなりません。

注

変数の割り当て: 変数への割り当ては、それぞれ 113 ページの『割り当ておよび比較』で説明されている検索割り当て規則に従って行われます。[127] 変数の数が行の中の値の数より少ない場合、 SQL 警告 (SQLSTATE 01503) が戻されます (そして、SQLCA の SQLWARN3 フィールドに 'W' が設定されます)。結果の列の数よりも変数の数が多い場合には、警告は出されない点に注意してください。値が NULL の場合は、その値に対して標識変数が用意されている必要があります。

変数として文字変数を指定し、その変数が、結果を収容するのに十分な大きさを持っていない場合には、警告 (SQLSTATE 01004) が戻され (そして SQLCA の SQLWARN1 に 'W' が割り当てられ) ます。標識変数が用意されている場合、結果の実際の長さが、その変数に関連する標識変数に戻されることがあります。詳しくは、 170 ページの『変数』を参照してください。

割り当てエラーが発生した場合は、該当の変数の値、および後に続く変数の値は予期できなくなります。ただし、変数に既に割り当てられている値があれば、その値は割り当てられたままです。

複数の割り当て: 複数の変数を INTO 文節で指定すると、割り当てを行う前にその照会が完全に評価されます。したがって、選択リスト内の変数 への参照は、常時、SELECT INTO ステートメント内のどの割り当てよりも前の変数 の値です。

空の結果表: 結果表が空である場合は、このステートメントは '02000' を SQLSTATE 変数に割り当て、変数には値を割り当てません。

127. SQL 変数または SQL パラメーターへの割り当て、および標準オプションが指定された場合は、ストレージ割り当て規則が適用されます。標準オプションについては、 xi ページの『標準への準拠』を参照してください。

複数の行がある結果表**:** 検索条件に該当する行が複数ある場合、ステートメントの処理は終了し、エラーが戻されます (SQLSTATE 21000)。結果表に複数の行があるためにエラーが生じた場合、変数には値が割り当てられることもあれば、割り当てられないこともあります。変数に値が割り当てられる場合、その値がどの行から得られるかは不定であり、予期できません。

結果列の評価に関する注意点**:** SELECT INTO ステートメントの選択リストにある結果列の評価でエラーが発生すると (例えば、算術式でゼロによる除算やオーバーフローなどが発生したり、数値や文字の変換エラーが発生したりすると)、結果は NULL 値になります。他の NULL 値の場合と同様に、標識変数を用意しなければなりません。該当の変数の値は、未定義になります。ただし、この場合、標識変数は -2 の値にセットされます。ステートメントの処理は続行して、警告が戻されます。標識変数が指定されない場合、エラーが戻されて、変数には値が割り当てられなくなります。エラーが戻されるとき、既にいくつかの値が変数に割り当てられていることがあり、それらの値は割り当てられたままになります。

日時値が戻されるとき、変数にはその値を完全に保管できるだけの長さが必要です。長さが不足する場合、切り捨てなければならない値の量に応じて、警告またはエラーが戻されます。詳しくは、120 ページの『日時割り当て』を参照してください。

例

例 *1:* COBOL プログラムのステートメントを使用して、表 EMPLOYEE の給与 (SALARY) 最高額を、ホスト変数 MAX-SALARY (DECIMAL(9,2)) に入れます。

```
EXEC SQL  SELECT MAX(SALARY)
            INTO :MAX-SALARY
            FROM EMPLOYEE WITH CS
END-EXEC.
```

例 *2:* Java プログラムのステートメントを使用して、接続コンテキスト 'ctx' にある EMPLOYEE 表から、従業員番号 (EMPNO) の値がホスト変数 HOST_EMP に保管されている値 (java.lang.String) と同じである行を選択します。さらに、選択した行にある名字 (LASTNAME) および教育レベル (EDLEVEL) を、それぞれホスト変数 HOST_NAME (ストリング) および HOST_EDUCATE (整数) に入れます。

```
#sql [ctx] {  SELECT LASTNAME, EDLEVEL
              INTO :HOST_NAME, :HOST_EDUCATE
              FROM EMPLOYEE
              WHERE EMPNO = :HOST_EMP   };
```

例 *3:* 従業員 528671 の行を、EMPLOYEE 表からホスト構造 EMPREC に入れます。行は後で更新され、照会の実行時にロックされると想定します。

```
EXEC SQL  SELECT *
            INTO :EMPREC
            FROM EMPLOYEE
            WHERE EMPNO = '528671'
            WITH RS USE AND KEEP EXCLUSIVE LOCKS
END-EXEC.
```

SET CONNECTION

SET CONNECTION ステートメントは、既存の接続の 1 つを識別することによって、活動化グループの現行サーバーを確立します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことと、対話式に呼び出すことだけが可能です。これは実行可能ステートメントですが、動的に準備することはできません。Java および REXX では指定できません。

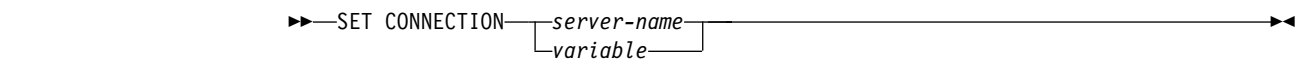
SET CONNECTION はトリガーおよび関数では使用できません。

権限

ステートメントでグローバル変数を参照する場合は、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別されるグローバル変数に対して、
  - そのグローバル変数に対する READ 特権
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

構文



説明

サーバー名 または 変数

指定したサーバー名、または指定した変数に入っているサーバー名によって接続を識別します。スキーマ名で修飾すれば、グローバル変数を使用することもできます。変数を指定する場合

- これは長さ属性が 18 以下の文字ストリング変数でなければなりません。
- 標識変数を伴っていてはなりません。
- サーバー名は、その変数内で左寄せし、通常 ID の形成の規則に従っていなければなりません。
- サーバー名の長さが、変数の長さよりも短い場合、右側をブランクで埋めなければなりません。

以下の説明で S は、指定したサーバー名または変数に入っているサーバー名を表しています。 S は該当のアプリケーション・プロセスの既存の接続を識別していなければなりません。 S が現行の接続を識別している場合は、S の状態およびアプリケーション・プロセスの他の接続すべての状態は変わりませんが、 S に関する情報が SQLCA のフィールド SQLERRP に入れられます。S が休止接続を識別している場合、次の規則が適用されます。

SET CONNECTION ステートメントが成功した場合、

- 接続 S は、現行状態になります。

- S が特殊レジスター CURRENT SERVER に入れられます。
- アプリケーション・サーバーについての情報は、 SQL 診断領域の接続情報項目に入れられます。
- アプリケーション・サーバー S に関する情報も、SQLCA の SQLERRP フィールドに入れられます。アプリケーション・サーバーが IBM リレーショナル・データベース製品の場合、その情報は *pppvvrrm* の形式です。ここで、
  - *ppp* は、製品を次のように識別します。
    - ARI (Db2 UDB (VSE および VM 版) の場合)
    - DSN (Db2 for z/OS の場合)
    - QSQ (Db2 for i の場合)
    - 他のすべての Db2 プロダクトの場合は SQL
  - *vv* は、2 桁のバージョン ID です (例えば、'04' など)。
  - *rr* は、2 桁のリリース ID です (例えば、'01' など)。
  - *m* は、1 桁のモディフィケーション・レベルを示します (例えば、'0' など)。

  例えば、アプリケーション・サーバーがバージョン 4 の Db2 for z/OS の場合、SQLERRP の値は「DSN04010」になります。
- 接続に関するその他の情報は、SQL 診断領域の DB2_CONNECTION_STATUS および DB2_CONNECTION_TYPE 接続情報項目から入手できます。

  DB2_CONNECTION_STATUS 接続情報項目は、この作業単位に対する接続の状態を示しています。次の値のいずれかが入れられます。
  - 1 - この作業単位の接続では、コミット可能な更新を行うことができる。
  - 2 - この作業単位の接続では、コミット可能な更新を行うことはできない。

  DB2_CONNECTION_TYPE 接続情報項目は、接続のタイプを示しています。次の値のいずれかが入れられます。
  - 1 - ローカルのリレーショナル・データベースとの接続。
  - 2 - 会話が保護されない、遠隔のリモート・リレーショナル・データベースとの接続。
  - 3 - 会話が保護される、遠隔のリモート・リレーショナル・データベースとの接続。
  - 4 - アプリケーション・リクエスターのドライバー・プログラムとの接続。
- 接続に関する追加の情報も SQLCA のフィールド SQLERRD(4) に入れられます。 SQLERRD(4) には、該当のアプリケーション・サーバーがコミット可能な更新を実行するのを許可するかどうかを示す値が入ります。以下は、この CONNECT に関して SQLCA のフィールド SQLERRD(4) に入れられる値とその意味を示しています。
  - 1 - コミット可能な更新を実行できます。接続は、無保護会話を使用する接続であるか、CONNECT (タイプ 1) ステートメントによってアプリケーション・リクエスターのドライバー・プログラムに対して確立された接続であるか、CONNECT (タイプ 1) ステートメントによって確立されたローカル接続であるかのいずれかです。
  - 2 - コミット可能な更新は行うことができません。会話は無保護です。

  - 3 - コミット可能な更新を行うことができるか否かは不明です。会話は保護会話です。
  - 4 - コミット可能な更新を行うことができるか否かは不明です。会話は無保護です。
  - 5 - コミット可能な更新を行うことができるかどうかは不明で、その接続は、CONNECT (タイプ 2) ステートメントを使用して確立されたローカル接続、または CONNECT (タイプ 2) ステートメントを使用して確立されたアプリケーション・リクエスター・ドライバー・プログラムとの接続です。
- 接続についての追加情報は SQLCA のフィールド SQLERRMC に入れられます。フィールド SQLERRMC の中の情報の説明については、付録 B、「SQL 連絡域」を参照してください。
- それ以前の現行接続は、休止状態になります。

SET CONNECTION ステートメントが不成功であった場合、該当の活動化グループの接続状態およびその接続の状態は変わりません。

注

CONNECT に対する **SET CONNECTION (**タイプ **1)**: CONNECT (タイプ 1) ステートメントの使用は SET CONNECTION の使用を妨げることはありませんが、休止状態の接続は存在しないので、そのステートメントは失敗するか何も行わないかのどちらかです。

接続がリストアされた後の状態: 同一の作業単位の中で接続が使用され、休止状態になり、その後で現行状態に復元された場合は、その接続に関するロック、カーソル、および準備済みステートメントの状況は、その活動化グループによる最後の使用を反映しています。

ローカル接続: 現行の独立補助記憶域プール (IASP) ネーム・スペースがローカル接続のリレーショナル・データベースに一致しない場合は、そのローカル接続に対する SET CONNECTION は失敗します。

例

TOROLAB1 で SQL ステートメントを実行し、次に TOROLAB2 で SQL ステートメントを実行し、最後に TOROLAB1 でさらに SQL ステートメントを実行します。

```
EXEC SQL CONNECT TO TOROLAB1;

(TOROLAB1 のオブジェクトを参照するステートメントを実行する)

EXEC SQL CONNECT TO TOROLAB2;

(TOROLAB2 のオブジェクトを参照するステートメントを実行する)

EXEC SQL SET CONNECTION TOROLAB1;

(TOROLAB1 のオブジェクトを参照するステートメントを実行する)
```

最初の CONNECT ステートメントは、TOROLAB1 との接続を確立し、2 番目の CONNECT ステートメントはその接続を休止状態にし、 SET CONNECTION ステートメントはその接続を現行状態に戻します。

SET CURRENT DEBUG MODE

SET CURRENT DEBUG MODE ステートメントは、値を CURRENT DEBUG MODE 特殊レジスターに割り当てます。

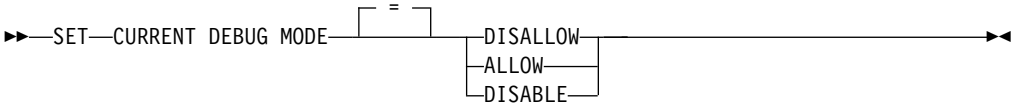
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

不要です。

構文



説明

CURRENT DEBUG MODE の値は、以下に示す指定されたキーワードによって置き換えられます。

DISALLOW
Unified Debugger がデバッグできないようにプロシージャーが作成されます。プロシージャーの DEBUG MODE 属性が DISALLOW である場合、後にそのプロシージャーを変更して DEBUG MODE 属性を変更することができます。

ALLOW
Unified Debugger がデバッグできるようにプロシージャーが作成されます。プロシージャーの DEBUG MODE 属性が ALLOW である場合、後にそのプロシージャーを変更して DEBUG MODE 属性を変更することができます。

DISABLE
Unified Debugger がデバッグできないようにプロシージャーが作成されます。プロシージャーの DEBUG MODE 属性が DISABLE である場合、後にそのプロシージャーを変更して DEBUG MODE 属性を変更することはできません。

注

トランザクションに関する考慮事項: SET CURRENT DEBUG MODE ステートメントは、コミット可能な操作ではありません。 ROLLBACK は、現行のデバッグ・モードには影響を与えません。

最初の現行デバッグ・モード: 現行デバッグ・モードの初期値は DISALLOW です。

現行デバッグ・モードの有効範囲: 現行デバッグ・モードの有効範囲は活動化グループです。

例

例 *1:* 以下のステートメントは、CREATE PROCEDURE (SQL) ステートメントによって作成される後続のプロシージャーがデバッグ可能となるように CURRENT DEBUG MODE を設定します。

```
SET CURRENT DEBUG MODE = ALLOW
```

例 *2:* 以下のステートメントは、CREATE PROCEDURE (SQL) ステートメントによって作成される後続のプロシージャーがデバッグ不可能となり、それらのプロシージャーがデバッグ可能に変更されないように CURRENT DEBUG MODE を設定します。

```
SET CURRENT DEBUG MODE = DISABLE
```

SET CURRENT DECFLOAT ROUNDING MODE

SET CURRENT DECFLOAT ROUNDING MODE ステートメントは、CURRENT DECFLOAT ROUNDING MODE 特殊レジスターの値を変更します。

呼び出し

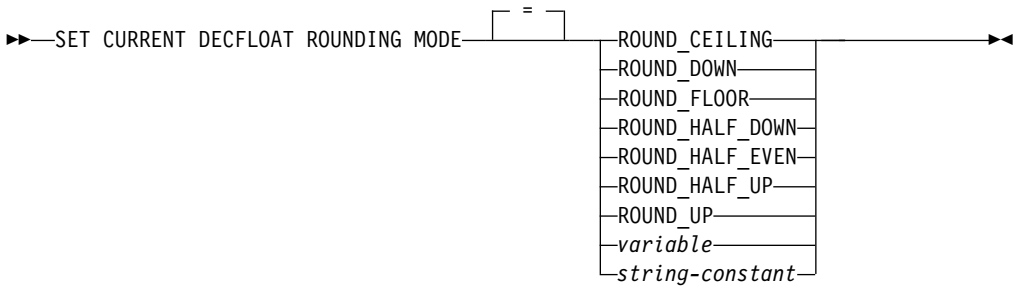
このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

ステートメント内のグローバル変数を参照する場合は、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別されるグローバル変数に対して、
  - そのグローバル変数に対する READ 特権
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

構文



説明

ROUND_CEILING
正の無限大の方向に丸めます。廃棄される桁がすべてゼロであるか、符号が負である場合、廃棄される桁が除去されることを除いて結果は変わりません。それ以外の場合、結果の係数は 1 だけ増分されます (切り上げられます)。

ROUND_DOWN
ゼロの方向に丸めます (切り捨て)。廃棄される桁は無視されます。

ROUND_FLOOR
負の無限大の方向に丸めます。廃棄される桁がすべてゼロであるか、符号が正である場合、廃棄される桁が除去されることを除いて結果は変わりません。それ以外の場合、符号は負となり、結果の係数は 1 だけ増分されます。

ROUND_HALF_DOWN
最も近い値に丸めます。等距離である場合、切り捨てます。廃棄される桁が左隣の桁の値の半分 (0.5) より大きい値を表す場合、結果の係数は 1 増分されます (切り上げ)。そうでない場合、廃棄される桁は無視されます。

この丸めモードは浮動小数点の算術計算に関する IEEE ドラフト標準でサポートされていないため、移植可能アプリケーションの作成時にはお勧めしません。

ROUND_HALF_EVEN
最も近い値に丸めます。等距離である場合、最後の桁が偶数になるように丸めます。廃棄される桁が左隣の桁の値の半分 (0.5) より大きい値を表す場合、結果の係数は 1 増分されます (切り上げ)。半分未満の場合、結果の係数は調整されません (つまり、廃棄される桁は無視されます)。上記がいずれも該当しない場合 (つまり正確に半分を表す場合)、結果の係数は、右端の桁が偶数の場合は変更されず、右端の桁が奇数 (偶数桁を作成するために) の場合は 1 だけ増分されます (切り上げられます)。

ROUND_HALF_UP
最も近い値に丸めます。等距離である場合、切り上げます。廃棄される桁が左隣の桁の値の半分 (0.5) 以上の値を表す場合、結果の係数は 1 増分されます (切り上げられます)。そうでない場合、廃棄される桁は無視されます。

ROUND_UP
ゼロから離れる方向に丸めます。廃棄される桁がすべてゼロである場合、廃棄される桁が除去されることを除いて結果は変わりません。それ以外の場合、結果の係数は 1 だけ増分されます (切り上げられます)。

この丸めモードは浮動小数点の算術計算に関する IEEE ドラフト標準でサポートされていないため、移植可能アプリケーションの作成時にはお勧めしません。

string-constant
丸めモードの指定を含む文字定数。

variable
CURRENT DECFLOAT ROUNDING MODE の値を含む変数を指定します。内容が大文字に変換されることはありません。

変数は、次の条件に合っていなければなりません。

- 文字ストリングまたは Unicode グラフィック・ストリングの変数でなければなりません。
- その後に標識変数が続くことはできません。
- 7 つの丸めモード・キーワードの 1 つを含んでいなければなりません。
- 変数が固定長である場合、右側はブランクで埋め込まれています。

注

トランザクションに関する考慮事項**:** SET CURRENT DECFLOAT ROUNDING MODE ステートメントはコミット可能な操作ではありません。ROLLBACK は、CURRENT DECFLOAT ROUNDING MODE に影響を与えません。

初期 **CURRENT DECFLOAT ROUNDING MODE:** 活動化グループ内の CURRENT DECFLOAT ROUNDING MODE の初期値は、その活動化グループ内で実行される最初の SQL ステートメントによって設定されます。

- 活動化グループにおける最初の SQL ステートメントが、SQL プログラムまたは SQL パッケージから実行される場合、CURRENT DECFLOAT ROUNDING MODE 特殊レジスターは DECFLTRND パラメーターの値に設定されます。
- それ以外の場合、初期値は ROUND_HALF_EVEN です。

CURRENT DECFLOAT ROUNDING MODE の有効範囲は活動化グループです。

例

例 *1:* CURRENT DECFLOAT ROUNDING MODE 特殊レジスターを、ストリング定数およびキーワードを使用して ROUND_DOWN に設定します。

```
SET CURRENT DECFLOAT ROUNDING MODE = 'ROUND_DOWN'
SET CURRENT DECFLOAT ROUNDING MODE = ROUND_DOWN
```

SET CURRENT DEGREE

SET CURRENT DEGREE ステートメントは、値を CURRENT DEGREE 特殊レジスターに割り当てます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。REXX で指定してはなりません。

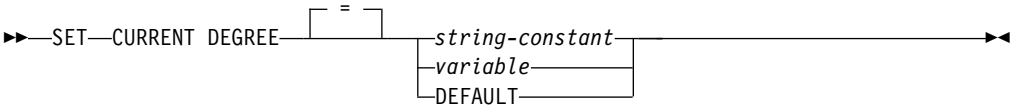
権限

このステートメントの権限 ID によって保持される特権には、特殊権限 *JOBCTL が含まれていなければなりません。その他の場合には、System i ナビゲーターのアプリケーション管理を介して、IBM i の SQL 管理機能に対して許可を与える必要があります。機能 ID を QIBM_DB_SQLADM に設定して機能使用法の変更 (CHGFCNUSG) コマンドを使用することによっても、許可されるユーザーのリストを変更することができます。

ステートメント内のグローバル変数を参照する場合は、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別されるグローバル変数に対して、
  - そのグローバル変数に対する READ 特権
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

構文



説明

CURRENT DEGREE の値は、ストリング定数または変数の値によって置き換えられます。

string-constant
文字ストリング定数を指定します。内容が大文字に変換されることはありません。

string-constant の長さは、先行ブランクおよび末尾ブランクを切り取った後 5 を超えてはなりません。

variable
CURRENT DEGREE の値を含む変数を指定します。

変数は、次の条件に合っていなければなりません。

- CHAR、VARCHAR、Unicode GRAPHIC、または Unicode VARGRAPHIC 変数でなければなりません。変数 の内容の実際の長さは、先頭ブランクおよび末尾ブランクを切り取った後 5 を超えてはなりません。
- NULL 値とすることはできません。
- 内容は大文字にしてください。すべての文字には大文字小文字の区別があり、大文字に変換されることはありません。

DEFAULT
現行照会オプション・ファイル (QAQQINI) の PARALLEL_DEGREE パラメーターが指定される場合、CURRENT DEGREE は PARALLEL_DEGREE にリセットされます。指定されない場合は、CURRENT DEGREE は、QQRYDEGREE システム値によって指定される度合いからリセットされます。

ストリング定数または変数の値は、以下のいずれかでなければなりません。

1 並列処理は許可されません。

2 から 32767
使用する並列処理の度合いを指定します。

ANY
I/O か SMP のいずれの並列処理の場合でも、データベース・マネージャーが任意の数のタスクを選択できることを指定します。

並列処理の使用および使用されるタスク数は、システムで使用可能なプロセッサーの数、ジョブが実行されているプール内の使用可能なアクティブ・メモリーに関するジョブの割り当て分、および操作に予想される経過時間が CPU 処理または I/O リソースによって限定されるかどうかに基づいて決定されます。データベース・マネージャーは、プール内のメモリーのこのジョブが占める割合に基づいて、経過時間を最小化するインプリメンテーションを選択します。

NONE
並列処理は許可されません。

MAX
I/O か SMP のいずれの並列処理の場合でも、データベース・マネージャーは任意の数のタスクを選択できます。MAX は、プール内のすべてのアクティブ・メモリーを使用できることをデータベース・マネージャーが前提とする点を除いて、ANY と類似しています。

IO データベース・マネージャーが照会に I/O 並列処理の使用を選択した場合、任意の数のタスクを使用できます。SMP は許可されません。

注

トランザクションに関する考慮事項**:** SET CURRENT DEGREE ステートメントは、コミット可能な操作ではありません。 ROLLBACK は、CURRENT DEGREE には影響を与えません。

最初の現行の度合い**:** CURRENT DEGREE の初期値は、CHGQRYA CL コマンド、現行照会オプション・ファイル (QAQQINI) の PARALLEL_DEGREE パラメーター、または QQRYDEGREE システム値の有効な並列処理の度合いと同じです。

並列処理の度合いの優先順位**:** 並列処理の度合いの制御はさまざまな方法で行えます。実際に使用される並列処理の度合いは次のように決定されます。

- SET CURRENT DEGREE ステートメント、または DEGREE キーワードを指定した CHGQRYA CL コマンドが実行された場合、いずれかの最新のものによって指定される並列処理の度合いの値は CURRENT DEGREE になります。
- SET CURRENT DEGREE ステートメント、および DEGREE キーワードを指定した CHGQRYA CL コマンドのどちらも実行されなかった場合、
  - PARALLEL_DEGREE パラメーターを指定した現行照会オプション・ファイル (QAQQINI) が指定された場合、QAQQINI ファイルによって指定される並列処理の度合いの値は CURRENT DEGREE になります。
  - 指定されない場合は、QQRYDEGREE システム値によって指定される並列処理の度合いの値は CURRENT DEGREE になります。

詳細については、「データベース・パフォーマンスおよび Query の最適化」トピック集を参照してください。

現行の度合いの有効範囲**:** CURRENT DEGREE の有効範囲はジョブです。

並列処理に関する制限**:** Db2 SMP 機能がインストールされていない場合、警告が返され、並列処理は使用されません。

SQL ステートメントの中にも、並列処理を使用できないものがあります。

例

例 1: 次のステートメントは、並列処理を禁止するよう CURRENT DEGREE を設定します。

```
SET CURRENT DEGREE = '1'
```

例 2: 次のステートメントは、並列処理を許可するよう CURRENT DEGREE を設定します。

```
SET CURRENT DEGREE = 'ANY'
```

SET CURRENT IMPLICIT XMLPARSE OPTION

SET CURRENT IMPLICIT XMLPARSE OPTION ステートメントは、CURRENT IMPLICIT XMLPARSE OPTION 特殊レジスターの値を変更します。

呼び出し

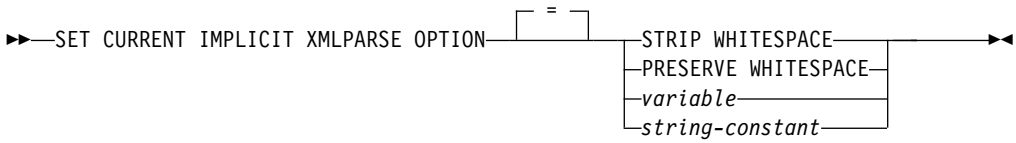
このステートメントは、アプリケーション・プログラムに組み込むことも、あるいは対話式に実行することもできます。これは、動的に準備できる実行可能ステートメントです。

権限

ステートメント内のグローバル変数を参照する場合は、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別されるグローバル変数に対して、
  - そのグローバル変数に対する READ 特権
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

構文



説明

STRIP WHITESPACE
暗黙的な XMLPARSE で空白を除去します。

PRESERVE WHITESPACE
暗黙的な XMLPARSE で空白を除去しません。

variable
CURRENT IMPLICIT XMLPARSE OPTION の値を含む変数を指定します。内容が大文字に変換されます。

変数は、次の条件に合っていなければなりません。

- 文字ストリングまたは Unicode グラフィック・ストリングの変数でなければなりません。
- その後に標識変数が続くことはできません。
- 2 つの暗黙的な XMLPARSE オプションのうちいずれかを含んでいなければなりません。
- 変数が固定長である場合、右側はブランクで埋め込まれています。

string-constant
暗黙的な XMLPARSE オプションの指定を含む文字定数。この値は左寄せされたストリングでなければならず、キーワード間に含まれるブランク文字が 1 つ

だけである「STRIP WHITESPACE」か「PRESERVE WHITESPACE」にする必要があります。内容が大文字に変換されます。

注

トランザクションに関する考慮事項**:** SET CURRENT IMPLICIT XMLPARSE OPTION ステートメントは、コミット可能な操作ではありません。 ROLLBACK は、CURRENT IMPLICIT XMLPARSE OPTION には影響を与えません。

初期 **CURRENT IMPLICIT XMLPARSE OPTION:** CURRENT IMPLICIT XMLPARSE OPTION の初期値は「STRIP WHITESPACE」です。

この特殊レジスターは、静的および動的の両方のステートメントに影響を与えます。

CURRENT IMPLICIT XMLPARSE OPTION の有効範囲は接続です。

例

CURRENT IMPLICIT XMLPARSE OPTION 特殊レジスターの値を「PRESERVE WHITESPACE」に設定します。

```
SET CURRENT IMPLICIT XMLPARSE OPTION = PRESERVE WHITESPACE
```

SET CURRENT TEMPORAL SYSTEM_TIME

SET CURRENT TEMPORAL SYSTEM_TIME ステートメントは、CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターの値を変更します。

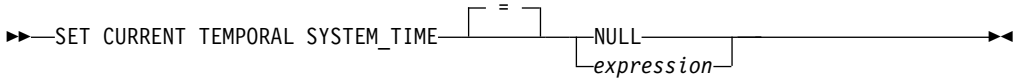
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、あるいは対話式に実行することもできます。これは、動的に準備できる実行可能ステートメントです。

権限

権限は不要です。

構文



説明

NULL
NULL 値を指定します。

expression
TIMESTAMP(12) に割り当てることができる式。式に、スカラー全選択を含めてはなりません。また、非決定的な関数、外部アクションである関数、および SQL データを変更する関数を含めることもできません。

注

トランザクションの考慮事項**:** SET CURRENT TEMPORAL SYSTEM_TIME ステートメントはコミット可能な操作ではありません。 ROLLBACK は CURRENT TEMPORAL SYSTEM_TIME に影響を及ぼしません。

CURRENT TEMPORAL SYSTEM_TIME の有効範囲**:** CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターの有効範囲は、活動化グループおよび接続です。

例

例 1: CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターを前月に設定します。

```
SET CURRENT TEMPORAL SYSTEM_TIME = CURRENT TIMESTAMP - 1 MONTH
```

例 2: CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターを NULL 値に設定します。

```
SET CURRENT TEMPORAL SYSTEM_TIME = NULL
```

SET DESCRIPTOR

SET DESCRIPTOR ステートメントは、SQL 記述子に情報を設定します。

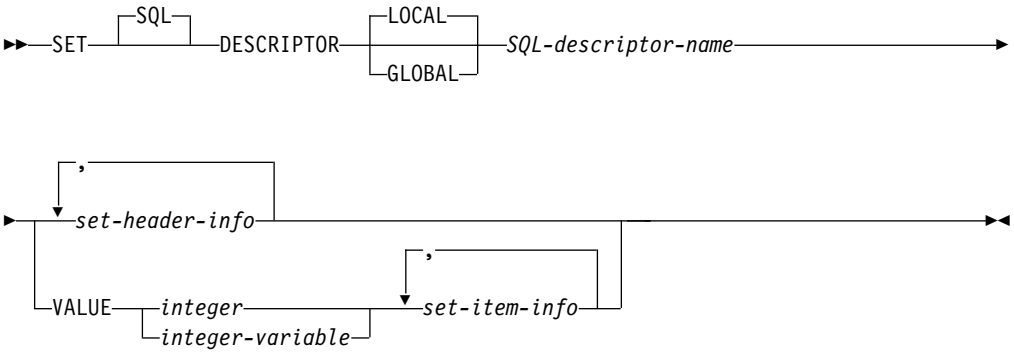
呼び出し

このステートメントは、アプリケーション・プログラム、SQL 関数、 SQL プロシージャー、またはトリガー内にのみ組み込むことができます。これを対話式に発行することはできません。これは実行可能ステートメントですが、動的に準備することはできません。REXX で指定してはなりません。

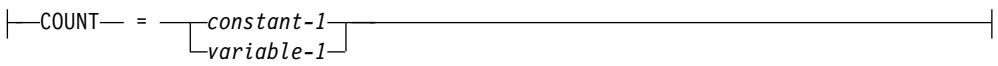
権限

権限は不要です。

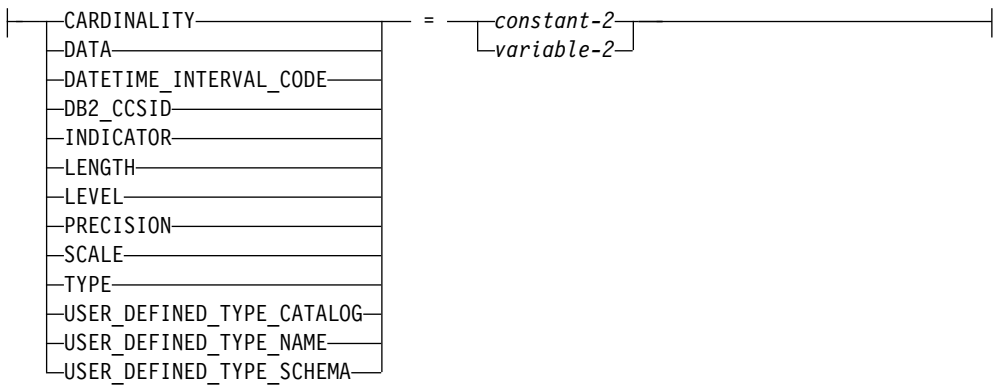
構文



set-header-info:



set-item-info:



説明

LOCAL
記述子の名前の有効範囲はプログラム呼び出しのローカルであることを指定します。提供される情報は、このローカルの有効範囲で認識される記述子に設定されます。

GLOBAL
記述子の名前の有効範囲は SQL セッション全体であることを指定します。提供される情報は、同じデータベース接続を使用して実行するプログラムによって認識される記述子に設定されます。

SQL-descriptor-name
SQL 記述子の名前を指定します。名前は、指定の有効範囲を持つ既存の記述子を識別するものでなければなりません。

set-header-info
SQL 記述子に属性を設定します。同じ記述子項目を、1 つの SET DESCRIPTOR ステートメントで複数回指定することはできません。

VALUE
指定された情報が設定される項目数を指定します。記述子に割り振られる項目の最大数よりも項目数が大きいか、または項目数が 1 より小さい場合、エラーが戻されます。

integer
1 から SQL 記述子の割り振り済み項目数の範囲の整数定数。

整変数
変数を宣言する規則に従ってプログラム内で宣言された、変数を指定します。変数をグローバル変数にすることはできません。変数のデータ・タイプは、SMALLINT、INTEGER、BIGINT、または DECIMAL、あるいは位取りがゼロの NUMERIC でなければなりません。変数の値は、1 から SQL 記述子の割り振り済み項目の最大数の範囲でなければなりません。

set-item-info
特定の項目に関する情報を SQL 記述子に設定します。同じ記述子項目を、1 つの SET DESCRIPTOR ステートメントで複数回指定することはできません。指定されたタイプに適用できない項目は無視されます。

set-header-info

COUNT
記述子に指定される項目の数。

variable-1
変数を宣言する規則に従ってプログラム内で宣言された、変数を指定します。ただし、これはファイル参照変数またはグローバル変数であってはなりません。変数のデータ・タイプは、 1485 ページの表 89で指定されている COUNT ヘッダー項目と互換性がなければなりません。変数は、(記憶域割り当て規則を使用して) COUNT ヘッダー項目に割り当てられます。割り当て規則の詳細については、 113 ページの『割り当ておよび比較』を参照してください。

constant-1
COUNT ヘッダー項目の設定に使用される定数値を識別します。定数のデー

タ・タイプは、 1485 ページの表 89で指定されている COUNT ヘッダー項目と互換性がなければなりません。定数は、(記憶域割り当て規則を使用して) COUNT ヘッダー項目に割り当てられます。割り当て規則の詳細については、113 ページの『割り当ておよび比較』を参照してください。

set-item-info

CARDINALITY
項目のカーディナリティーを指定します。許可されるのは、TYPE が配列の場合のみです。

DATA
項目記述子によって記述されるデータの値を指定します。 INDICATOR の値が負の場合、DATA の値は未定義となります。割り当てられる値は定数になれません。

DATETIME_INTERVAL_CODE
特定の日時データ・タイプを指定します。 TYPE が 9 に設定される場合、DATETIME_INTERVAL_CODE の指定は必須です。

1 DATE
2 TIME
3 TIMESTAMP

DB2_CCSID
文字、グラフィック、XML、および日時データの CCSID を指定します。これ以外のデータ・タイプの値はすべて適用できません。 DB2_CCSID が指定されないか、0 が指定される場合があります。

- XML データの場合、SQL_XML_DATA_CCSID QAQQINI オプション設定が使用されます。
- それ以外の場合は、変数の CCSID はジョブの CCSID によって決定されます。

INDICATOR
標識の値を指定します。負でない値は、この記述子項目に DATA 値が指定されることを示します。拡張標識変数が使用可能になっていない場合、負の値は、この記述子項目によって記述される値が NULL 値であることを示します。設定されない場合、INDICATOR の値は 0 です。拡張標識変数が使用可能な場合は、次のとおりです。

- -1、-2、-3、-4、または -6 は、この記述子項目によって記述される値が NULL 値であることを示します。
- -5 は、この記述子項目によって記述される値が DEFAULT 値であることを示します。
- -7 は、この記述子項目によって記述される値が UNASSIGNED 値であることを示します。
- 0 または正の値は、この記述子項目に DATA 値が提供されることを示します。

LENGTH
データの最大長を指定します。データ・タイプが文字またはグラフィック・ストリング・タイプ、XML タイプまたは日時タイプの場合、長さは文字数を表します (バイト数ではない)。データ・タイプがバイナリー・ストリングまたは他の

タイプの場合、長さはバイト数を表します。 LENGTH が指定されない場合、デフォルトの長さが使用されます。デフォルトの説明については、 1692 ページの表 117を参照してください。

LEVEL

項目記述子のレベル。

0 1 次記述子項目です。

1 2 次記述子項目用です。これは、配列項目用です。

PRECISION

データ・タイプ DECIMAL、NUMERIC、DECFLOAT、DOUBLE、REAL、FLOAT、および TIMESTAMP の記述子項目についての精度を指定します。 PRECISION が指定されない場合、デフォルトの精度が使用されます。デフォルトの説明については、 1692 ページの表 117を参照してください。

SCALE

データ・タイプ DECIMAL または NUMERIC の記述子項目についての位取りを指定します。 SCALE が指定されない場合、デフォルトの位取りが使用されます。デフォルトの説明については、 1692 ページの表 117を参照してください。

TYPE

記述子項目のデータ・タイプを表すデータ・タイプ・コードを指定します。データ・タイプ・コードおよび長さの説明については、 1487 ページの表 90を参照してください。それぞれの記述子項目ごとに、TYPE または USER_DEFINED_TYPE_NAME および USER_DEFINED_TYPE_SCHEMA のいずれか (両方ではない) を指定しなければなりません。

USER_DEFINED_TYPE_CATALOG

ユーザー定義タイプのサーバー名を指定します。USER_DEFINED_TYPE_CATALOG が指定される場合、これは現行サーバーと等しくなければなりません。指定されない場合、USER_DEFINED_TYPE_CATALOG が現行サーバーとなります。

USER_DEFINED_TYPE_NAME

ユーザー定義データ・タイプの名前を指定します。それぞれの記述子項目ごとに、TYPE または USER_DEFINED_TYPE_NAME および USER_DEFINED_TYPE_SCHEMA のいずれか (両方ではない) を指定しなければなりません。

USER_DEFINED_TYPE_SCHEMA

ユーザー定義タイプが入っているスキーマを指定します。それぞれの記述子項目ごとに、TYPE または USER_DEFINED_TYPE_NAME および USER_DEFINED_TYPE_SCHEMA のいずれか (両方ではない) を指定しなければなりません。

variable-2

変数を宣言する規則に従ってプログラム内で宣言された、変数を指定します。ただし、これはファイル参照変数またはグローバル変数であってはなりません。変数のデータ・タイプは、 1485 ページの表 89で指定されている記述子情報項目と互換性がなければなりません。変数は、(記憶域割り当て規則を使用して) 対応する記述子項目に割り当てられます。割り当て規則の詳細については、 113 ページの『割り当ておよび比較』を参照してください。

一般的に、DATA 項目を設定するときは変数のデータ・タイプ、長さ、精度、位取り、および CCSID が 1485 ページの表 89 で指定されているものと同じでなければなりません。可変長タイプの場合、可変長は記述子の LENGTH よりも小さくてはなりません。 C NUL 終了タイプの場合、可変長は記述子の LENGTH よりも少なくとも 1 大きくなければなりません。

constant-2
記述子項目の設定に使用される定数値を識別します。定数のデータ・タイプには、 1485 ページの表 89 で指定されているのと同じデータ・タイプ、長さ、精度、位取り、および CCSID がなければなりません。定数は、(記憶域割り当て規則を使用して) 対応する記述子項目に割り当てられます。割り当て規則の詳細については、 113 ページの『割り当ておよび比較』を参照してください。

記述子項目が DATA に設定される場合、定数-2 は指定できません。

注

記述子項目のデフォルト値**:** 以下の表は、記述子項目のデフォルト値が指定されない場合の LENGTH、PRECISION、および SCALE のデフォルト値を表しています。

表 *117.* デフォルトの *LENGTH*、*PRECISION*、および *SCALE*

データ・タイプ	**LENGTH**	**PRECISION**	**SCALE**
DECIMAL および NUMERIC		5	0
FLOAT		53	0
DECFLOAT		34	
CHARACTER、VARCHAR、および CLOB	1		
GRAPHIC、VARGRAPHIC、および DBCLOB	1		
BINARY、VARBINARY、および BLOB	1		
DATE	10		
TIME	8		
TIMESTAMP	26	6	
XML	1		

例

例 *1:* 記述子 'NEWDA' の項目数を :numitems の値に設定します。

```
EXEC SQL SET DESCRIPTOR  'NEWDA'
  COUNT = :numitems;
```

例 *2:* 記述子 'NEWDA' の最初の項目記述子のタイプおよび長さの値を設定します。

```
SET DESCRIPTOR 'NEWDA'
  VALUE 1 TYPE   = :dtype,
          LENGTH = :olength;
```

SET ENCRYPTION PASSWORD

SET ENCRYPTION PASSWORD ステートメントは、暗号化および暗号化解除の機能で使用されるデフォルトのパスワードおよびヒントを設定します。このパスワードは認証に関するものではなく、データの暗号化および暗号化解除にのみ使用されます。

このステートメントの使用について詳しくは、 450 ページの『ENCRYPT_AES』、 453 ページの『ENCRYPT_RC2』、 456 ページの『ENCRYPT_TDES』、 432 ページの『DECRYPT_BIT、
DECRYPT_BINARY、
DECRYPT_CHAR、および DECRYPT_DB』 を参照してください。

呼び出し

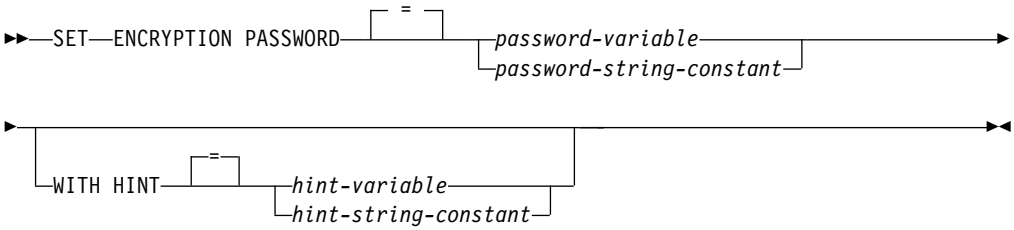
このステートメントは、アプリケーション・プログラムに組み込むことも、あるいは対話式に実行することもできます。これは、動的に準備できる実行可能ステートメントです。

権限

ステートメント内のグローバル変数を参照する場合は、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別された、それぞれのグローバル変数ごとに、
  - そのグローバル変数に対する READ 特権
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

構文



説明

password-variable
暗号化パスワードを含む変数を指定します。

変数は、次の条件に合っていなければなりません。

- CHAR、VARCHAR、Unicode GRAPHIC、または Unicode VARGRAPHIC 変数でなければなりません。変数の内容の実際の長さは、6 以上 127 以下であるか、または空ストリングでなければなりません。空ストリングを指定する場合、デフォルトの暗号化パスワードは値なしに設定されます。
- NULL 値とすることはできません。

- すべての文字には大文字小文字の区別があり、大文字に変換されることはありません。

password-string-constant
文字定数。この定数の長さは、6 以上 127 以下であるか、または空ストリングでなければなりません。空ストリングを指定する場合、デフォルトの暗号化パスワードは値なしに設定されます。リテラル形式のパスワードは、静的 SQL または REXX では許可されていません。

WITH HINT
データ所有者がパスワードを思い出すための値 ('Pacific' を思い出すための 'Ocean' など) が指定されていることを示します。ヒント値が指定されている場合、そのヒントは暗号化関数のデフォルトとして使用されます。暗号値のヒントは、後に GETHINT 関数を使用して検索できます。この文節を指定しない場合、および暗号化関数に対してヒントを明示的に指定しない場合には、出力される暗号化データにヒントは組み込まれません。

hint-variable
暗号化パスワードのヒントを含む変数を指定します。

変数は、次の条件に合っていなければなりません。

- CHAR、VARCHAR、Unicode GRAPHIC、または Unicode VARGRAPHIC 変数でなければなりません。変数の内容の実際の長さは、32 を超えてはなりません。空ストリングを指定する場合、デフォルトの暗号化パスワード・ヒントは値なしに設定されます。
- NULL 値とすることはできません。
- すべての文字には大文字小文字の区別があり、大文字に変換されることはありません。

hint-string-constant
文字定数。この定数の長さは、32 よりも大きくすることはできません。空ストリングを指定する場合、デフォルトの暗号化パスワード・ヒントは値なしに設定されます。

注

パスワード保護: 暗号化されたパスワードに不注意によりアクセスすることのないように、プログラム、プロシージャー、または関数のソースにはパスワード・ストリング定数 を指定しないでください。その代わりに、変数を使用してください。

リモート・リレーショナル・データベースに接続しているとき、指定されたパスワード自体は「平文で」送信されます。つまり、パスワード自体は暗号化されません。このようなケースでパスワードを保護するには、 IPSEC (または IBM i 製品同士の接続の場合は SSL) などの通信暗号化メカニズムを使用することを考慮してください。

トランザクションに関する考慮事項: SET ENCRYPTION PASSWORD ステートメントは、コミット可能な操作ではありません。 ROLLBACK は、デフォルトの暗号化パスワードまたはデフォルトの暗号化パスワード・ヒントには影響を与えません。

暗号化パスワードの初期値**:** デフォルトの暗号化パスワードおよびデフォルトの暗号化パスワード・ヒントの初期値は、どちらも空ストリング ('') です。

暗号化パスワードの有効範囲**:** デフォルトの暗号化パスワードおよびデフォルトの暗号化パスワード・ヒントの有効範囲は、活動化グループおよび接続です。

例

ENCRYPTION PASSWORD を :hv1 の値に設定します。

```
SET ENCRYPTION PASSWORD :hv1
```

SET OPTION

SET OPTION ステートメントは、SQL ステートメントで使用される処理オプションを設定します。

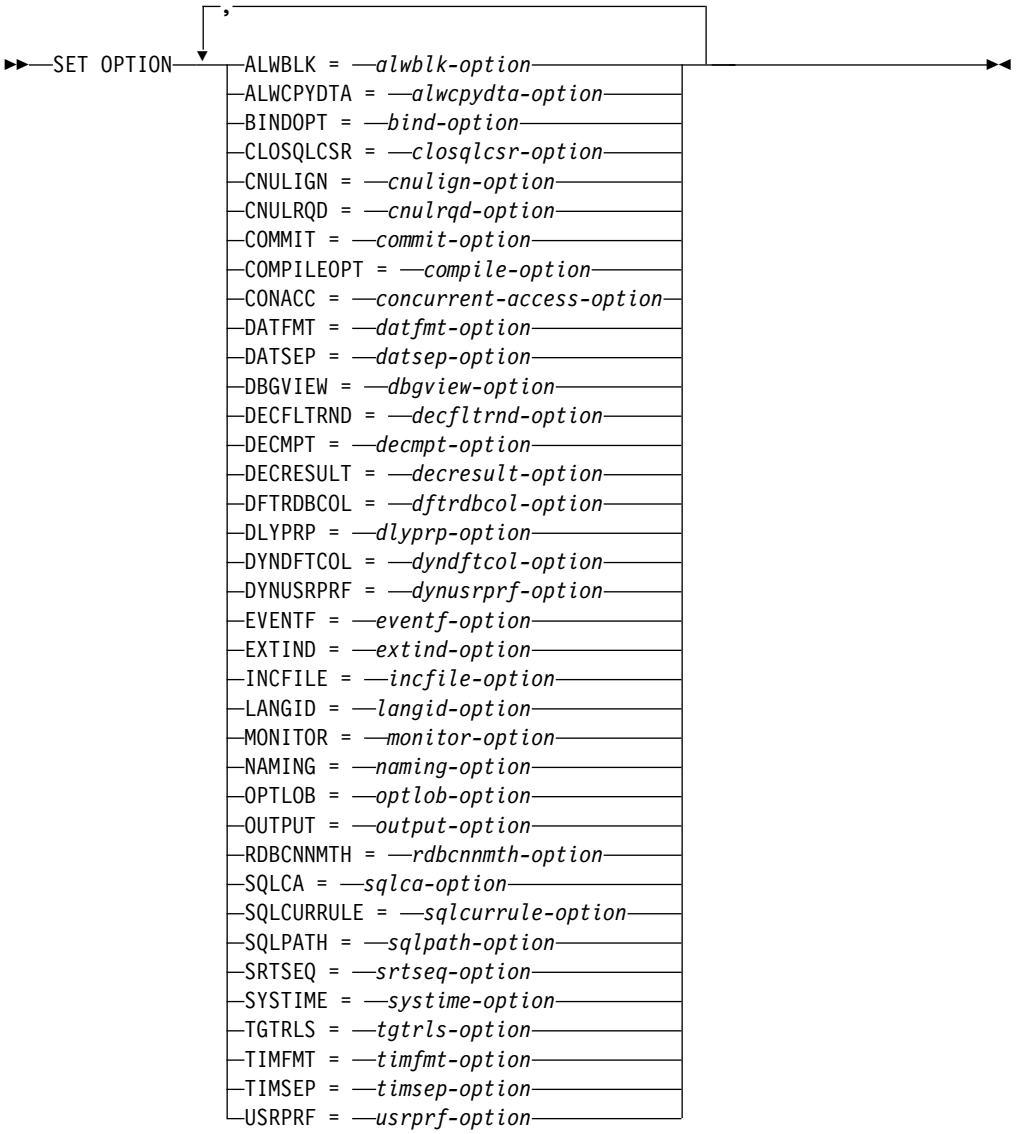
呼び出し

このステートメントは、REXX プロシージャーで使用、またはアプリケーション・プログラムに組み込むことができます。 REXX プロシージャーで使用する場合、このステートメントは実行可能ステートメントです。アプリケーション・プログラムに組み込む場合、このステートメントは実行可能ではなく、他のどの SQL ステートメントよりも先に行う必要があります。このステートメントは、動的に準備することができません。

権限

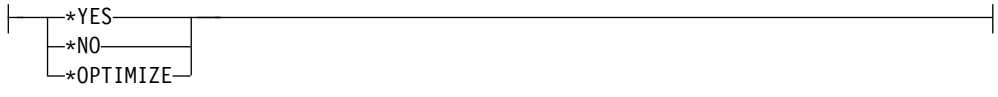
権限は不要です。

構文

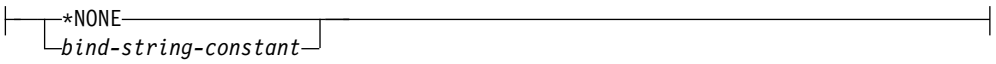


alwblk-option:

alwcpydta-option:



bind-option:



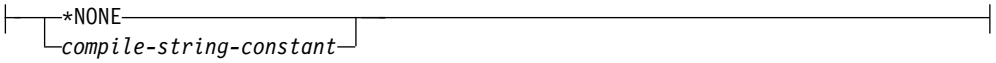
closqlcsr-option:

cnulign-option:

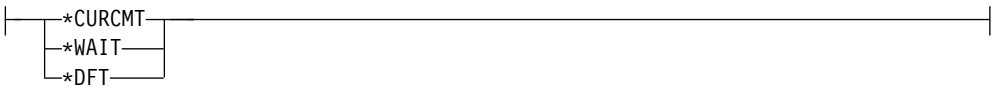
cnulrqd-option:

commit-option:

compile-option:



concurrent-access-option:



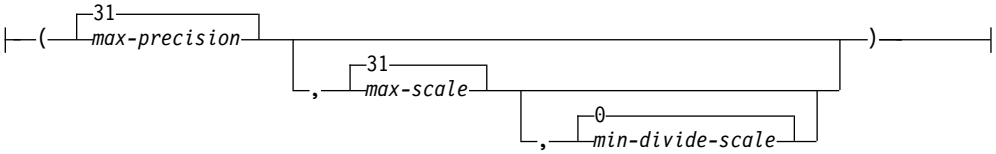
datfmt-option:

datsep-option:

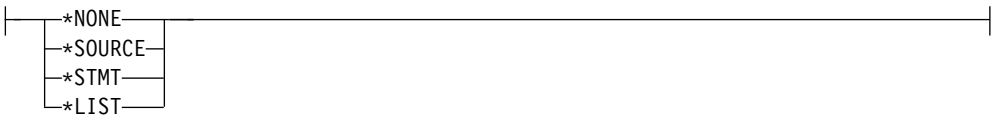
decfltrnd-option:

decmpt-option:

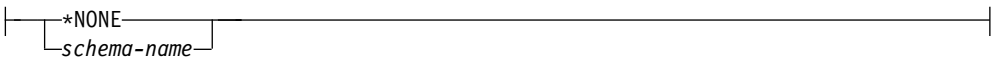
decresult-option:



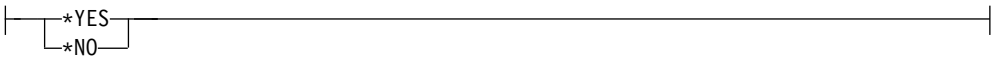
dbgview-option:



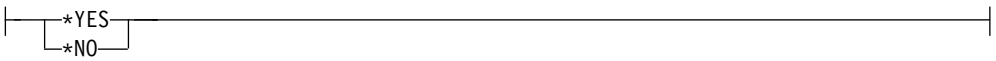
dftrdbcol-option:



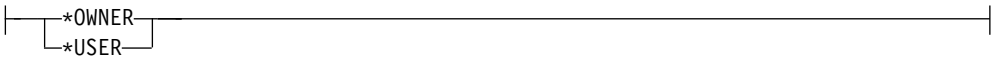
dlyprp-option:



dyndftcol-option:

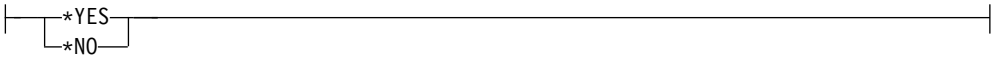


dynusrprf-option:



eventf-option:

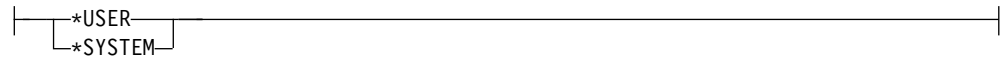
extind-option:



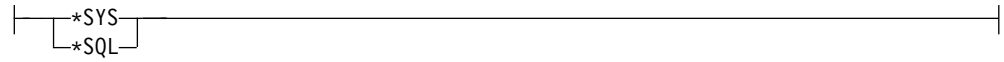
incfile-option:

langid-option:

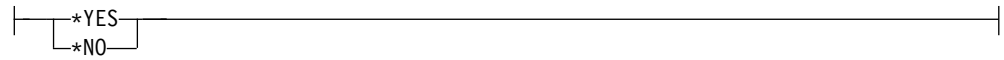
monitor-option:



naming-option:

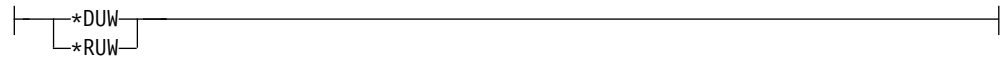


optlob-option:



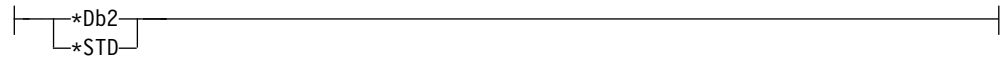
output-option:

rdbcnnmth-option:



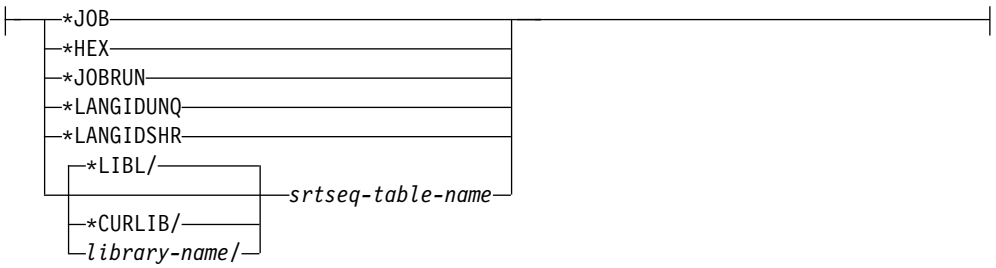
sqlca-option:

sqlcurrule-option:

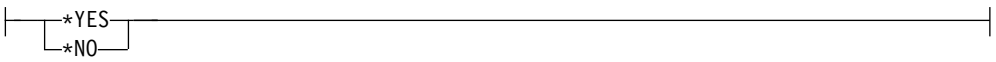


sqlpath-option:

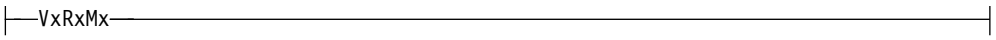
srtseq-option:



systime-option:



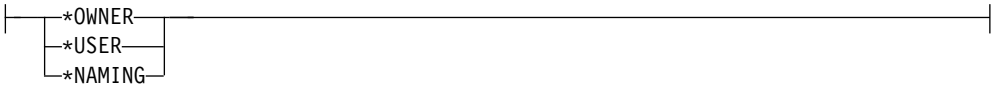
tgtrls-option:



timfmt-option:

timsep-option:

usrprf-option:



説明

ALWBLK

データベース・マネージャーが行ブロッキングを使用できるかどうか、およびブロッキングを読み取り専用カーソルに使用できる範囲を指定します。このオプションは、REXX では無視されます。

***ALLREAD**

COMMIT が *NONE、*CHG、または *CS の場合、読み取り専用カーソルでは行がブロックされます。プログラム内にある、明示的に更新できないカーソルはすべて EXECUTE または EXECUTE IMMEDIATE ステートメントがそのプログラム内にある可能性があっても、読み取り専用処理用にオープンされます。

*ALLREAD を指定すると、

- *READ で許可されているブロッキングに加えて、コミットメント制御レベル *CHG および *CS のもとでの行ブロッキングが可能になります。
- プログラム内のほとんどすべての読み取り専用カーソルのパフォーマンスを上げることができますが、以下のやり方で照会が制限されます。
  - ロールバック (ROLLBACK) コマンド、ホスト言語での ROLLBACK ステートメント、または ROLLBACK HOLD SQL ステートメントは、次のような場合には読み取り専用カーソルの位置変更をしません。
    - カーソルを含むプログラムまたはルーチンの作成時に ALWBLK(*ALLREAD) が指定された場合
    - カーソルを含むプログラムまたはルーチンの作成時に ALWBLK(*READ) および ALWCPYDTA(*OPTIMIZE) が指定された場合
  - 位置指定 UPDATE または DELETE ステートメントを動的に実行 (例えば、EXECUTE IMMEDIATE を使用して) しても、カーソルの DECLARE ステートメントに FOR UPDATE 文節が含まれていない場合は、そのカーソル内の行を更新することはできません。

***NONE**

カーソルに関するデータの検索のために、行はブロックされません。

*NONE を指定すると、

- 検索されるデータが必ず現行のデータになります。
- 照会用のデータの最初の行を検索するために要する時間が短縮される場合があります。
- 照会がクローズする前に、その照会の最初の数行しか検索されないときは、データベース・マネージャーがプログラムによって使用されないデータ行のブロックを検索するのを、取り止めるようにします。
- 多数の行を検索する照会の場合、その照会全体のパフォーマンスを低下させる場合があります。

***READ**

次の場合に、カーソルに関するデータの読み取り専用検索で、行がブロックされます。

- COMMIT パラメーターに *NONE が指定され、コミットメント制御が使用されないことが指示されたとき。
- FOR READ ONLY 文節によってカーソルが宣言されたとき、またはカーソルに関して位置指定 UPDATE ステートメントまたは DELETE ステートメントを実行できる動的ステートメントがないとき。

*READ を指定すると、上記の条件を満たし、かつ大量の行を検索する照会の全体のパフォーマンスを上げることができます。

ALWCPYDTA
データのコピーを SELECT ステートメント内で使用できるかどうかを指定します。このオプションは、REXX では無視されます。

***OPTIMIZE**
システムが、データベースから直接検索されたデータを使用するか、そのデータのコピーを使用するかを決定します。この決定は、どちらの方法が最高のパフォーマンスを発揮するかに基づいて行われます。 COMMIT が *CHG または *CS で、ALWBLK が *ALLREAD でない場合、または COMMIT が *ALL または *RR の場合には、照会の実行が必要な場合に限りデータのコピーが使用されます。

***YES**
データのコピーは、必要な場合にだけ使用されます。

***NO**
データのコピーを使用することはできません。そのデータの一時コピーが照会の実行に必要な場合、エラー・メッセージが戻されます。

BINDOPT
SQL 関数、SQL プロシージャー、または SQL トリガーの作成時に使用する CRTPGM コマンドまたは CRTSRVPGM CL コマンドで使用する追加のパラメーターを指定します。 BINDOPT ストリングは、プリコンパイラーによって生成された CRTPGM コマンドまたは CRTSRVPGM CL コマンドに追加されます。ストリングの内容は妥当性検査されません。不正なパラメーターが存在する場合、バインド・コマンドはエラーを発行します。バインド・オプション ACTGRP、ALWRINZ、AUT、ENTMOD、
EXPORT、MODULE、REPLACE、STGMDL、TEXT、TGTRLS、および USRPRF は、プリコンパイラーで設定されるため、このストリングに指定することはできません。

このオプションは、SQL 関数、SQL プロシージャー、または SQL トリガー内でのみ使用できます。このオプションは、REXX では無視されます。

***NONE**
CRTPGM または CRTSRVPGM CL コマンドで使用される追加のパラメーターはありません。

bind-string-constant
バインド・オプションを含む 5000 文字以下の文字定数。

CLOSQLCSR
SQL カーソルが暗黙的にクローズされるとき、SQL 準備済みステートメントが暗黙的に廃棄され、 LOCK TABLE ロックが解除されることを指定します。SQL カーソルは、CLOSE、COMMIT、または ROLLBACK (HOLD はなし) の各 SQL ステートメントを出すと、明示的にクローズされます。このオプションは、REXX では無視されます。*ENDACTGRP および *ENDMOD は、ILE プログラムおよびモジュール、SQL 関数、SQL プロシージャー、または SQL トリガーが使用するためのものです。*ENDPGM、*ENDSQL、および *ENDJOB は、非 ILE プログラムが使用します。

SQL スカラー関数、SQL プロシージャー、および SQL トリガーは、デフォルトとして *ENDMOD を使用します。SQL 表関数は常に *ENDACTGRP を使用して作成されます。

***ENDACTGRP**
活動化グループが終了すると、SQL カーソルはクローズし、SQL 準備済みステートメントは暗黙的に廃棄され、 LOCK TABLE ロックは解除されます。

***ENDMOD**
モジュールが終了すると、SQL カーソルはクローズし、SQL 準備済みステートメントは暗黙的に廃棄されます。 LOCK TABLE ロックは、呼び出しスタックの最初の SQL プログラムが終了すると解除されます。カーソルは論理的にのみクローズされる可能性があり、カーソルが物理的にクローズされるのは、スタックで SQL の最初のプログラムが終了した時点、そしてそのプログラムが *ENDACTGRP を使用してコンパイルされなかった場合のみである点に留意してください。

***ENDPGM**
プログラムが終了すると、SQL カーソルはクローズし、SQL 準備済みステートメントは廃棄されます。 LOCK TABLE ロックは、呼び出しスタックの最初の SQL プログラムが終了すると解除されます。

***ENDSQL**
SQL カーソルは、呼び出しから次の呼び出しまでの間もオープンしたままで、新たに SQL OPEN 実行しなくても取り出すことができます。この場合、呼び出しスタック上で高位にあるプログラムの 1 つが、少なくとも 1 個の SQL ステートメントを実行していなければなりません。呼び出しスタックの最初の SQL プログラムが終了すると、SQL カーソルはクローズし、SQL 準備済みステートメントは廃棄され、LOCK TABLE ロックは解除されます。最初に呼び出された SQL プログラム (呼び出しスタック上の最初の SQL プログラム) に *ENDSQL が指定されると、そのプログラムは *ENDPGM が指定された場合と同様に扱われます。

***ENDJOB**
SQL カーソルは、呼び出しから次の呼び出しまでの間もオープンしたままで、新たに SQL OPEN 実行しなくても取り出すことができます。呼び出しスタック上で高位にあるプログラムが、SQL ステートメントを実行している必要はありません。呼び出しスタックの最初の SQL プログラムが終了しても、SQL カーソルはオープンしたままで、 SQL 準備済みステートメントは保存され、LOCK TABLE ロックは保持されます。ジョブが終了すると、SQL カーソルはクローズされ、SQL 準備済みステートメントは廃棄され、 LOCK TABLE ロックは解除されます。

CNULIGN
文字およびグラフィック・ホスト変数に NUL 終了文字を無視するかどうかを指定します。このオプションは、C および C++ プログラム内の SQL ステートメントにしか使用されません。

このオプションは、SQL 関数、SQL プロシージャー、または SQL トリガーでは使用できません。

***YES**
NUL 文字で終了するように定義された文字およびグラフィックのホスト変数は、INSERT ステートメントと UPDATE ステートメントの固定長の変数として扱われます。 NUL 終了文字は、データの一部と見なされます。

***NO**
NUL 終了の文字およびグラフィックのホスト変数が、INSERT ステートメントと UPDATE ステートメントに NUL 終了文字を使用します。

CNULRQD
文字およびグラフィック・ホスト変数に NUL 終了文字を戻すかどうかを指定します。このオプションは、C および C++ プログラム内の SQL ステートメントにしか使用されません。

このオプションは、SQL 関数、SQL プロシージャー、または SQL トリガーでは使用できません。

***YES**
出力の文字およびグラフィック・ホスト変数に、常に NUL 終了文字が含まれます。 NUL 終了文字用のスペースが不足している場合、データが切り捨てられ、NUL 終了文字が追加されます。入力の文字およびグラフィック・ホスト変数には、NUL 終了文字が必須です。

***NO**
出力の文字およびグラフィック・ホスト変数の場合、ホスト変数の長さがデータとまったく同じ場合、 NUL 終了文字は戻されません。入力の文字およびグラフィック・ホスト変数には、NUL 終了文字は必要ありません。

COMMIT
使用される分離レベルを指定します。REXX では、ソースで参照されるファイルはこのオプションの影響を受けません。 SQL ステートメントで参照される表、ビュー、およびパッケージだけが影響を受けます。分離レベルの詳細については、 32 ページの『分離レベル』を参照してください。

***CHG**
非コミット読み取りの分離レベルを指定します。

***NONE**
コミットなしの分離レベルを指定します。REXX プロシージャーに DROP SCHEMA ステートメントが入っている場合、 *NONE を使用する必要があります。

***CS**
カーソル固定の分離レベルを指定します。

***ALL**
読み取り固定の分離レベルを指定します。

***RR**
反復可能読み取りの分離レベルを指定します。

COMPILEOPT
コンパイラー・コマンドで使用する追加のパラメーターを指定します。
COMPILEOPT ストリングは、プリコンパイラーによって作成されたコンパイラー・コマンドに追加されます。ストリングのどこかに 'INCDIR(' が存在する場合、プリコンパイラーは SRCSTMF パラメーターを使用してコンパイラーを

呼び出します。ストリングの内容は妥当性検査されません。不正なパラメーターが存在する場合、コンパイラー・コマンドはエラーを発行します。プリコンパイラーがコンパイラーに渡すキーワードのいずれかを使用すると、パラメーターが重複するためにコンパイラー・コマンドは失敗します。プリコンパイラーがコンパイラー・コマンド用に生成するパラメーターのリストについては、「組み込み SQL プログラミング」トピック集を参照してください。このオプションは、REXX では無視されます。

このオプションは、SQL 関数、SQL プロシージャー、または SQL トリガーでは使用できません。

***NONE**
コンパイラー・コマンドで使用される追加のパラメーターはありません。

character-string
コンパイラー・オプションを含む 5000 文字以下の文字定数。

CONACC
選択ステートメントで使用する同時アクセスの解決を指定します。

***CURCMT**
データベース・マネージャーが更新プロセスまたは削除プロセスの過程にある行を検出するときに、現時点でコミット済みのバージョンのデータを使用するように指定します。挿入プロセスの過程に行は、スキップされます。この値は、分離レベル *CS が可能な場合に有効です。

***WAIT**
別のトランザクションによって更新プロセスまたは削除プロセスにあるデータのコミットまたはロールバックを待機するように指定します。この値は、分離レベル *CS および *ALL が可能な場合に有効です。

***DFT**
対象プログラムに関して、同時アクセス・オプションが明示的に設定されないように指定します。プログラムの呼び出し時に有効な値が使用されます。

DATFMT
日付結果列にアクセスするときに使用する形式を指定します。日付の出力フィールドは、すべて指定した形式で戻されます。入力日付ストリングのときは、日付が有効な形式で指定されたかどうかを判別するために、指定した値が使用されます。

注: *USA、*ISO、*EUR、または *JIS の形式を使用する入力日付ストリングは、常に有効です。

***JOB:**
ジョブに指定された形式が使用されます。ジョブの現行日付形式を決定するには、ジョブ表示 (DSPJOB) コマンドを使用してください。

***ISO**
国際標準化機構 (ISO) の日付形式 (yyyy-mm-dd) が使用されます。

***EUR**
欧州の日付形式 (dd.mm.yyyy) が使用されます。

***USA**
米国の日付形式 (mm/dd/yyyy) が使用されます。

***JIS**
日本工業規格 (JIS) の日付形式 (yyyy-mm-dd) が使用されます。

***MDY**
日付形式 (mm/dd/yy) が使用されます。

***DMY**
日付形式 (dd/mm/yy) が使用されます。

***YMD**
日付形式 (yy/mm/dd) が使用されます。

***JUL**
年間通算日形式 (yy/ddd) が使用されます。

DATSEP
日付の結果列にアクセスする場合に使用される、区切り文字を指定します。

注: このパラメーターは、*JOB、*MDY、*DMY、*YMD、または *JUL が DATFMT パラメーターで指定されたときだけ適用されます。

***JOB**
そのジョブで指定されている日付区切り文字が使用されます。ジョブ表示 (DSPJOB) コマンドを使用すると、ジョブの現在の値を確かめることができます。

***SLASH または '/'**
スラッシュ (/) が使用されます。

***PERIOD または '.'**
ピリオド (.) が使用されます。

***COMMA または ','**
コンマ (,) が使用されます。

***DASH または '-'**
ダッシュ (-) が使用されます。

***BLANK または ' '**
ブランク ( ) が使用されます。

DBGVIEW
システムのデバッグ機能によってオブジェクトをデバッグできるかどうか、またコンパイラーが提供するデバッグ情報のタイプを指定します。 DBGVIEW パラメーターは、SQL 関数、プロシージャー、およびトリガーの本体でのみ指定できます。

CREATE PROCEDURE または ALTER PROCEDURE ステートメントの DEBUG MODE が指定される場合は、SET OPTION ステートメントの DBGVIEW オプションを指定してはなりません。

選択可能な項目は、次のとおりです。

***NONE**
デバッグ表示は生成されません。

***SOURCE**
SQL ステートメント・ソースを使用して、コンパイル済みモジュール・オブジェクトをデバッグできます。 *SOURCE を指定した場合、変更された

ソースは作成された関数、プロシージャー、またはトリガーと同じスキーマ内のソース・ファイル QSQDSRC に保管されます。

***STMT**
プログラム・ステートメント番号と記号 ID を使用して、コンパイル済みモジュール・オブジェクトをデバッグできます。

***LIST**
コンパイル済みモジュール・オブジェクトのデバッグのリスト表示を生成します。

DEBUG MODE を指定せずに SET OPTION ステートメント内の DBGVIEW オプションを指定した場合、プロシージャーを Unified Debugger でデバッグすることはできませんが、システム・デバッグ機能を使用してデバッグすることは可能です。 DEBUG MODE オプションも DBGVIEW オプションも指定しない場合は、CURRENT DEBUG MODE 特殊レジスターでのデバッグ・モードが使用されます。

DECFLTRND
静的 SQL ステートメントに使用する DECFLOAT 丸めモードを指定します。選択可能な項目は、次のとおりです。

***CEILING**
正の無限大の方向に丸めます。廃棄される桁がすべてゼロであるか、符号が負である場合、廃棄される桁が除去されることを除いて結果は変わりません。それ以外の場合、結果の係数は 1 だけ増分されます (切り上げられます)。

***DOWN**
ゼロの方向に丸めます (切り捨て)。廃棄される桁は無視されます。

***FLOOR**
負の無限大の方向に丸めます。廃棄される桁がすべてゼロであるか、符号が正である場合、廃棄される桁が除去されることを除いて結果は変わりません。それ以外の場合、符号は負となり、結果の係数は 1 だけ増分されます。

***HALFDOWN**
最も近い値に丸めます。等距離である場合、切り捨てます。廃棄される桁が左隣の桁の値の半分 (0.5) より大きい値を表す場合、結果の係数は 1 増分されます (切り上げ)。そうでない場合、廃棄される桁は無視されます。

***HALFEVEN**
最も近い値に丸めます。等距離である場合、最後の桁が偶数になるように丸めます。廃棄される桁が左隣の桁の値の半分 (0.5) より大きい値を表す場合、結果の係数は 1 増分されます (切り上げ)。半分未満の場合、結果の係数は調整されません (つまり、廃棄される桁は無視されます)。上記がいずれも該当しない場合 (つまり正確に半分を表す場合)、結果の係数は、右端の桁が偶数の場合は変更されず、右端の桁が奇数 (偶数桁を作成するために) の場合は 1 だけ増分されます (切り上げられます)。

***HALFUP**
最も近い値に丸めます。等距離である場合、切り上げます。廃棄される桁が

左隣の桁の値の半分 (0.5) 以上の値を表す場合、結果の係数は 1 増分されます (切り上げられます)。そうでない場合、廃棄される桁は無視されます。

***UP**
ゼロから離れる方向に丸めます。廃棄される桁がすべてゼロである場合、廃棄される桁が除去されることを除いて結果は変わりません。それ以外の場合、結果の係数は 1 だけ増分されます (切り上げられます)。

DECMPT
小数点を表すのに使用する記号を指定します。選択可能な項目は、次のとおりです。

***PERIOD**
小数点を表すのにピリオドを使用します。

***COMMA**
小数点を表すのにコンマを使用します。

***SYSVAL**
小数点の表現は、システム値 (QDECFMT) に従います。

***JOB**
小数点を表すのに、ジョブ値 (DECFMT) を使用します。

DECRESULT
10 進数での算術など、10 進演算で使用する最大精度、最大位取り、および最小除算位取りを指定します。指定する制限は、NUMERIC および DECIMAL データ・タイプだけに適用されます。

max-precision
10 進演算から戻される最大精度を示す整数定数。この値は 31 または 63 となります。デフォルト値は 31 です。

max-scale
10 進演算から戻される最大位取りを示す整数定数。この値は、0 から最大精度までの範囲から指定できます。デフォルト値は 31 です。

min-divide-scale
割り算演算から戻される最小位取りを示す整数定数。この値は、1 から 9 の範囲内でなければならず、 *max-scale* より大きくてはなりません。デフォルトは 0 で、0 は最小の位取りが指定されないことを示します。

DFTRDBCOL
表、ビュー、索引、および SQL パッケージの非修飾名に使用するスキーマ名を指定します。このパラメーターは、静的 SQL ステートメントにのみ適用します。このオプションは、REXX では無視されます。

***NONE**
OPTION プリコンパイル・パラメーターに指定された、または SET OPTION NAMING オプションによって指定された命名規則が使用されます。

schema-name
スキーマの名前を指定します。この値は、OPTION プリコンパイル・パラメーターに指定された、または SET OPTION NAMING オプションによって指定された命名規則の代わりに使用されます。

DLYPRP

PREPARE ステートメントの動的ステートメント妥当性検査を、 OPEN、EXECUTE、または DESCRIBE ステートメントが実行されるまで遅らせるかどうかを指定します。妥当性検査を遅らせると、冗長妥当性検査が行われなくなるのでパフォーマンスが上がります。このオプションは、REXX では無視されます。

***NO**

動的ステートメント妥当性検査は遅れません。動的ステートメントが準備されると、アクセス・プランの妥当性検査が行われます。動的ステートメントが OPEN または EXECUTE ステートメントで使用されると、アクセス・プランの妥当性検査が再度行われます。動的ステートメントによって参照されるオブジェクトの権限または存在は変わる可能性があるため、OPEN または EXECUTE ステートメントを出した後でも SQLCODE または SQLSTATE をチェックして、その動的ステートメントがまだ有効であるか確認する必要があります。

***YES**

動的ステートメントの妥当性検査は、その動的ステートメントが OPEN、EXECUTE、または DESCRIBE SQL ステートメントで使用されるまで遅れます。動的ステートメントが使用された時点で、妥当性検査は完了し、アクセス・プランが作成されます。*YES を指定する場合、OPEN、EXECUTE、または DESCRIBE ステートメントを実行した後に SQLCODE および SQLSTATE をチェックして、その動的ステートメントが有効であるか確認する必要があります。

注: *YES を指定すると、PREPARE ステートメントで INTO 文節が使用された場合、または DESCRIBE ステートメントで、そのステートメントに OPEN が出される前に動的ステートメントが使用された場合、パフォーマンスは上がりません。

DYNDFTCOL

DFTRDBCOL パラメーターに指定されたスキーマ名が、動的ステートメントにも使用されることを指定します。このオプションは、REXX では無視されます。

***NO**

表、ビュー、索引、および SQL パッケージの非修飾名として DFTRDBCOL に指定された値を、動的 SQL ステートメントには使用しません。OPTION プリコンパイル・パラメーターに指定された、または SET OPTION NAMING オプションによって指定された命名規則が使用されます。

***YES**

DFTRDBCOL に指定されたスキーマ名が、動的 SQL ステートメントの中で、表、ビュー、索引、および SQL パッケージの非修飾名として使用されます。

DYNUSRPRF

動的 SQL ステートメントにユーザー・プロファイルを使用することを指定します。このオプションは、REXX では無視されます。

***USER**
ローカル動的 SQL ステートメントが、ジョブのユーザー・プロファイルのもとで実行されます。分散動的 SQL ステートメントは、アプリケーション・サーバー・ジョブのユーザー・プロファイルのもとで実行されます。

***OWNER**
ローカル動的 SQL ステートメントが、プログラムの所有者のユーザー・プロファイルのもとで実行されます。分散動的 SQL ステートメントは、SQL パッケージの所有者のユーザー・プロファイルのもとで実行されます。

EVENTF
イベント・ファイルを生成するかどうかを指定します。連携開発環境/400® (CODE/400) は、イベント・ファイルを使用して、CODE/400 エディターと統合されたエラー・フィードバックを提供します。

***YES**
コンパイラーは、連携開発環境/400 (CODE/400 が使用するイベント・ファイルを生成します。

***NO**
コンパイラーは、連携開発環境/400 (CODE/400 が使用するイベント・ファイルを生成しません。

EXTIND
SQL ステートメントで渡される標識変数値を扱う方法を指定します。

***NO**
拡張標識変数を使用可能にしないこと、および選択ステートメント の暗黙または明示的な UPDATE 文節内では非更新可能な列を使用できないことを指定します。

***YES**
拡張標識変数を使用可能にすること、および *SELECT* ステートメント の暗黙または明示的な UPDATE 文節内では非更新可能な列を使用できることを指定します。

INCFILE
INCLUDE SQL ステートメントに使用するソース・ファイルの名前を指定します。 INCLUDE SQL ステートメントにソース・ファイル名が指定されなかったときに、INCLUDE SQL ステートメントにリストされたソース・メンバーを検索するためにこれが使用されます。

このオプションは、SQL 関数、SQL プロシージャー、または SQL トリガー内でのみ使用できます。このオプションは、REXX では無視されます。

file-name
使用されるソース・ファイルの名前を指定します。ファイルの名前は、次のライブラリー値のいずれかによって修飾できます。

***LIBL**
そのジョブのライブラリー・リストのユーザーおよびシステム部分のすべてのライブラリーが検索され、見つかった最初の表が使用されます。

***CURLIB**
ジョブ用の現行ライブラリーが検索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name
検索したいライブラリーの名前を指定します。

LANGID
SRTSEQ(*LANGIDUNQ) または SRTSEQ(*LANGIDSHR) が指定されているときに使用される、言語 ID を指定します。

***JOB** または ***JOBRUN**
そのジョブの LANGID の値が使用されます。

分散アプリケーションの場合、LANGID(*JOBRUN) が有効なのは、SRTSEQ(*JOBRUN) も指定されている場合だけです。

language-id
使用したい言語の ID を指定します。言語 ID として使用できる値についての説明は、「グローバリゼーション」トピック集にあるトピック言語 ID を参照してください。

MONITOR
データベース・モニターの実行時にステートメントがユーザー・ステートメントとして識別されるか、またはシステム・ステートメントとして識別されるかを指定します。

***USER**
SQL ステートメントはユーザー・ステートメントとして識別されます。これはデフォルトです。

***SYSTEM**
SQL ステートメントはシステム・ステートメントとして識別されます。

NAMING
SQL 命名規則とシステム命名規則のどちらを使用するかを指定します。このオプションは、SQL 関数、SQL プロシージャー、または SQL トリガーでは使用できません。

選択可能な項目は、次のとおりです。

***SYS**
システム命名規則が使用されます。

***SQL**
SQL 命名規則が使用されます。

OPTLOB
XML および LOB へのアクセスが、DRDA を介してアクセスする場合に最適化できるかどうかを指定します。選択可能な項目は、次のとおりです。

***YES**
LOB および XML アクセスは最適化されます。カーソルの最初の FETCH によって、それ以降のすべての FETCH においてそのカーソルが LOB および XML でどのように使用されるかが決定されます。このオプションは、そのカーソルがクローズされるまで有効です。

最初の FETCH で LOB または XML 列にアクセスするためにロケーターを使用すると、それ以降、そのカーソルの FETCH で、その LOB または XML 列を LOB または XML 変数内に取り出すことはできません。

最初の FETCH で LOB または XML 変数内に LOB または XML 列が置かれると、それ以降、そのカーソルの FETCH でその列用にロケーターを使用することができません。

***NO**
LOB アクセスは最適化されません。列を LOB ロケーター内に取り出すか、 LOB 変数内に取り出すかについての制約はありません。このオプションによって、パフォーマンスが低下する場合があります。

OUTPUT
プリコンパイラーおよびコンパイラー・リストを生成するかどうかを指定します。OUTPUT パラメーターは、SQL 関数、プロシージャー、およびトリガーの本体でのみ指定できます。選択可能な項目は、次のとおりです。

***NONE**
プリコンパイラーおよびコンパイラー・リストは生成されません。

***PRINT**
プリコンパイラーおよびコンパイラー・リストが生成されます。

RDBCNNMTH
CONNECT ステートメントに使用する意味体系を指定します。このオプションは、REXX では無視されます。

***DUW**
CONNECT (タイプ 2) の意味体系は、分散作業単位をサポートするのに使用されます。追加のリレーショナル・データベースに対して CONNECT ステートメントを連続して使用しても、それ以前の接続は切断されません。

***RUW**
CONNECT (タイプ 1) の意味体系は、リモート作業単位をサポートするのに使用されます。連続する CONNECT ステートメントによって、新しい接続が確立される前に直前の接続が切断されることになります。

SQLCA
SQLCA 内のフィールドが各 SQL ステートメントの後に設定されるかどうかを指定します。 SQLCA オプションを指定できるのは、ILE C、ILE C++、ILE COBOL、および ILE RPG だけです。 このオプションは、SQL 関数、SQL プロシージャー、または SQL トリガーでは使用できません。

選択可能な項目は、次のとおりです。

***YES**
SQLCA 内のフィールドは各 SQL ステートメントの後に設定されます。ユーザー・プログラムは、SQL ステートメントの実行の後に SQLCA 内のすべての値を参照できます。

***NO**
SQLCA 内のフィールドは各 SQL ステートメントの後に設定されません。ユーザー・プログラムは GET DIAGNOSTICS ステートメントを使用して、 SQL ステートメントの実行に関する情報を検索します。

SQLCA(*NO) は、通常は SQLCA(*YES) よりも良好に実行します。

他のホスト言語では SQLCA が必要となり、 SQLCA 内のフィールドは各 SQL ステートメントの後に設定されます。

SQLCURRULE

SQL ステートメントに使用する意味体系を指定します。

***Db2**

すべての SQL ステートメントの意味体系は、デフォルトにより、Db2 用に設定された規則に従います。以下の意味体系は、このオプションによって制御されます。

- 16 進定数が文字データとして処理されます。
- Unicode グラフィック・ストリング定数は UCS-2 (CCSID 13488) です。
- ルーチンまたはトリガーの本体の中の SQL 変数および SQL パラメーターへの割り当てでは、検索割り当て規則が使用されます。
- SQLDA および SQLN 内への SELECT ステートメントの記述が必要な SQLVAR 項目数より少ない場合、結果表に LOB または UDT が含まれる場合にのみ SQLSTATE 01005 が返されます。

***STD**

すべての SQL ステートメントの意味体系は、デフォルトにより、 ISO および ANSI SQL の規格用に設定された規則に従います。以下の意味体系は、このオプションによって制御されます。

- 16 進定数が 2 進データとして処理されます。
- Unicode グラフィック・ストリング定数は UTF-16 (CCSID 1200) です。
- ルーチンまたはトリガーの本体の中の SQL 変数および SQL パラメーターへの割り当てでは、記憶域割り当て規則が使用されます。
- SQLDA および SQLN 内への SELECT ステートメントの記述が必要な SQLVAR 項目数より少ない場合、常に SQLSTATE 01005 が返されます。

SQLPATH

静的 SQL ステートメント内で、プロシージャー、関数、およびユーザー定義タイプを見つけるために使用するパスを指定します。このオプションは、REXX では無視されます。

***LIBL**

使用されるパスは、実行時のライブラリー・リストです。

character-string

コンマで区切られた 1 つ以上のスキーマ名を持つ文字定数。指定できるのは、システム・スキーマ名だけです。

SRTSEQ

SQL ステートメントの中のストリング比較に使用される照合順序表を指定します。

注: *HEX を指定する必要があるのは、REXX プロシージャーを接続するアプリケーション・サーバーが Db2 for i、または V2R3M0 より前のリリース・レベルの IBM i 製品ではない場合です。

***JOB または *JOBRUN**
そのジョブの SRTSEQ の値が使用されます。

***HEX**
照合順序表は使用しません。照合順序を決定するには、文字の 16 進値を使用します。

***LANGIDUNQ**
照合順序表には、コード・ページの各文字ごとに固有の重み付けが含まれていなければなりません。

***LANGIDSHR**
指定された LANGID の共用重み付けソート表が使用されます。

srtseq-table-name
そのプログラムで使用する照合順序表の名前を指定します。照合順序表の名前は、次のライブラリー値のいずれかによって修飾できます。

***LIBL**
そのジョブのライブラリー・リストのユーザーおよびシステム部分のすべてのライブラリーが検索され、見つかった最初の表が使用されます。

***CURLIB**
ジョブ用の現行ライブラリーが検索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name
検索したいライブラリーの名前を指定します。

SYSTIME
CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターが静的および動的 SQL ステートメントに影響するかどうかを指定します。

選択可能な項目は、次のとおりです。

***YES**
システム期間テンポラル表への参照は、CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターの値の影響を受けます。

***NO**
システム期間テンポラル表への参照は、CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターの値の影響を受けません。

TGTRLS
ユーザーが作成するオブジェクトを使用するオペレーティング・システムのリリースを指定します。TGTRLS パラメーターは、 SQL 関数、プロシージャー、およびトリガーの本体に指定するか、または、外部関数またはプロシージャー作成ステートメントの一部としてのみ指定できます。選択可能な項目は、次のとおりです。

VxRxMx
VxRxMx 形式でリリースを指定します。ここで、Vx はバージョン、Rx は

リリース、Mx は修正レベルを表します。例えば、V7R1M0 は、バージョン 7、リリース 1 修正レベル 0 です。オブジェクトは、指定のリリースまたはそれ以降のリリースのオペレーティング・システムがインストールされたシステム上で使用できます。

有効な値は、現行のバージョン、リリース、および修正レベルによって決まり、新規リリースのたびに変更されます。データベース・マネージャーによってサポートされる最も古いリリース・レベルより前のリリース・レベルを指定すると、サポートされる最も古いリリースを示したエラー・メッセージが送られます。

TIMFMT

時刻の結果列にアクセスするときに使用する形式を指定します。時刻の出力フィールドはすべて指定した形式で戻されます。入力時刻ストリングの場合は、指定された値を使用して、時刻が有効な形式で指定されているかどうかを判別します。

注**:** *USA、*ISO、*EUR、または *JIS の形式を使用する入力時刻ストリングは、常に有効です。

***HMS**
形式 (hh:mm:ss) が使用されます。

***ISO**
国際標準化機構 (ISO) の時刻形式 (hh.mm.ss) が使用されます。

***EUR**
欧州の時刻形式 (hh.mm.ss) が使用されます。

***USA**
米国の時刻形式 (hh:mm *xx*) が使用されます。ここで、*xx* は AM または PM です。

***JIS**
日本工業規格 (JIS) の時刻形式 (hh:mm:ss) が使用されます。

TIMSEP

時刻の結果列にアクセスする場合に使用される区切り文字を指定します。

注**:** このパラメーターは、TIMFMT パラメーターで *HMS が指定されたときだけ適用されます。

***JOB**
そのジョブに指定されている時刻の区切り文字が使用されます。ジョブ表示 (DSPJOB) コマンドを使用すると、ジョブの現在の値を確かめることができます。

***COLON または ':'**
コロン (:) が使用されます。

***PERIOD または '.'**
ピリオド (.) が使用されます。

***COMMA または ','**
コンマ (,) が使用されます。

***BLANK または ' '**
ブランク ( ) が使用されます。

USRPRF
コンパイル済みプログラム・オブジェクトが実行される際に使用されるユーザー・プロファイル (そのプログラム・オブジェクトが静的 SQL ステートメント内の各オブジェクトごとに保有する権限も含む) を指定します。プログラムの所有者またはプログラム・ユーザーのいずれかのプロファイルが、プログラム・オブジェクトがどのオブジェクトを使用できるかを制御するために使用されます。このオプションは、REXX では無視されます。

***NAMING**
ユーザー・プロファイルは、命名規則によって決定されます。命名規則が *SQL の場合、USRPRF(*OWNER) が使用されます。命名規則が *SYS の場合、USRPRF(*USER) が使用されます。

***USER**
プログラム・オブジェクトを実行しているユーザーのプロファイルが使用されます。

***OWNER**
プログラムの所有者とプログラム・ユーザーの両方のユーザー・プロファイルが、プログラムの実行時に使用されます。

注

デフォルト値**:** オプションのデフォルト値は、言語、オブジェクト・タイプ、および作成時に有効なオプションに依存します。

- SQL プロシージャー、SQL 関数、または SQL トリガーの作成時のオプションのデフォルト値は、オブジェクトの作成時に有効なデフォルト値となります。例えば、SQL プロシージャーが作成され、その時点の COMMIT オプションが *CS である場合、*CS が COMMIT のデフォルト・オプションになります。これにより、各オプションは SET OPTION ステートメント内で検出されると更新されます。
- REXX 以外のアプリケーション・プログラムでは、オプションのデフォルト値は、CRTSQLxxx コマンドで指定した値になります。これにより、各オプションは、SET OPTION ステートメント内で検出されると更新されます。 SET OPTION ステートメントはすべて、他のどの組み込み SQL よりも先に置く必要があります。
- REXX プロシージャーの開始時に、オプションはそれぞれのデフォルト値に設定されます。各オプションのデフォルト値は、構文図に最初にリストされている値です。オプションが SET OPTION ステートメントによって変更されると、新しい値は、そのオプションが再度変更されるか、またはその REXX プロシージャーが終了するまで有効です。

代替構文**:** 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- *UR は *CHG の同義語として使用できます。
- *NC は *NONE の同義語として使用できます。

- *RS は *ALL の同義語として使用できます。

例

例 *1* : 分離レベルを *ALL に、命名モードを SQL 名に設定します。

```
EXEC SQL SET OPTION COMMIT =*ALL, NAMING =*SQL
```

例 *2* : 日付形式を欧州形式に、分離レベルを *CS に、小数点をコンマに設定します。

```
EXEC SQL SET OPTION DATFMT = *EUR, COMMIT = *CS, DECMPT = *COMMA
```

SET PATH

SET PATH ステートメントは、CURRENT PATH 特殊レジスターの値を変更します。

呼び出し

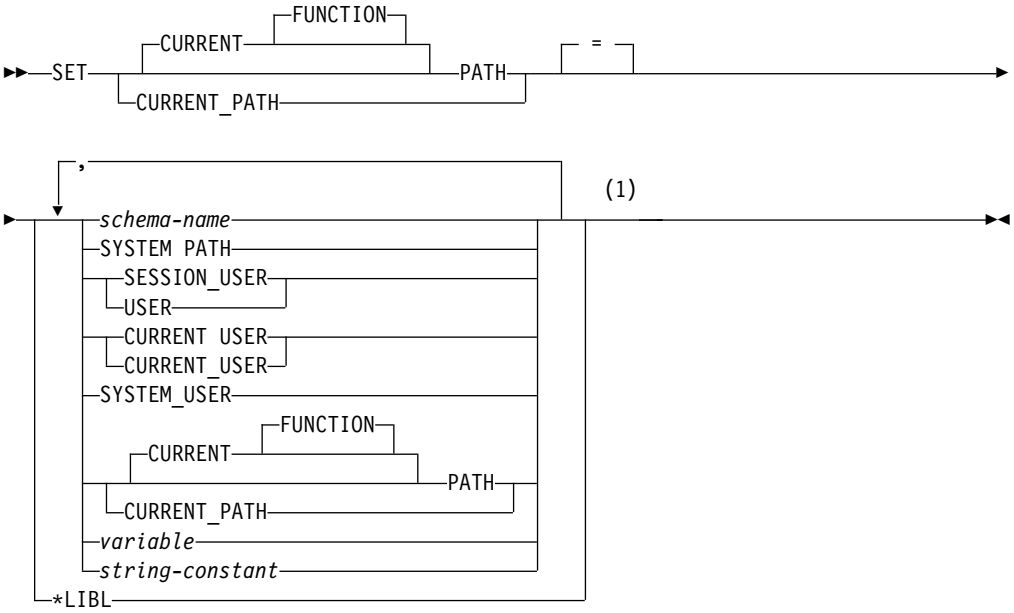
このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

ステートメント内のグローバル変数を参照する場合は、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別された、それぞれのグローバル変数ごとに、
  - そのグローバル変数に対する READ 特権
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

構文



注:

1 SYSTEM PATH、SESSION_USER、USER、CURRENT_USER、SYSTEM_USER、および CURRENT PATH は、ステートメントの右側におのおの 1 回まで指定できます。

説明

schema-name
スキーマを識別します。指定されたスキーマ名がシステム・スキーマ名である場合、PATH の設定時には、そのスキーマが存在するかどうかについての検査は

行われません。例えば、スキーマ名 をミススペルした場合、後続の SQL の操作はその影響を受けます。指定されたスキーマ名がシステム・スキーマ名ではない場合、PATH の設定時に、そのスキーマが存在していなければなりません。

推奨されていることではありませんが、PATH が「PATH」として指定されている場合、それをスキーマ名 として指定できます。

SYSTEM PATH
システム・パスのスキーマ名を指定します。この値を指定することは、スキーマ名として "QSYS"、"QSYS2"、 "SYSPROC"、"SYSIBMADM" を指定する場合と同じになります。

SESSION_USER または USER
SESSION_USER 特殊レジスターの値を指定します。

CURRENT USER
CURRENT USER 特殊レジスターの値を指定します。

SYSTEM_USER
SYSTEM_USER 特殊レジスターの値を指定します。

CURRENT PATH
このステートメントの実行前の CURRENT PATH 特殊レジスターの値を指定します。現行パスが *LIBL の場合、CURRENT PATH は許されません。

variable
コンマで区切られた、1 つ以上のスキーマ名を含む変数を指定します。スキーマ名で修飾すれば、グローバル変数を使用することもできます。

変数は、次の条件に合っていなければなりません。

- CHAR、VARCHAR、Unicode GRAPHIC、または Unicode VARGRAPHIC 変数でなければなりません。変数 の内容の実際の長さは、パスの最大長を超えることはできません。
- その後に標識変数が続くことはできません。
- NULL 値とすることはできません。
- 各スキーマ名は、通常の ID または区切り文字付き ID の形成に関する規則に従っていなければなりません。
- 各スキーマ名には、通常の ID に指定できない小文字の文字が含まれていることはできません。
- 変数が固定長文字である場合、右側はブランクで埋め込まれています。
- SET ステートメントが SQL ルーチン、SQL トリガー、または複合ステートメントで指定された場合、 *variable* を非修飾名のグローバル変数にすることはできません。

string-constant
コンマで区切られた 1 つ以上のスキーマ名を持つ文字定数。

ストリング定数は次の条件を満たしている必要があります。

- 各スキーマ名は、通常の ID または区切り文字付き ID の形成に関する規則に従っていなければなりません。
- 各スキーマ名には、通常の ID に指定できない小文字の文字が含まれていることはできません。

***LIBL**
パスは現行スレッドのライブラリー・リストに設定されます。

注

トランザクションに関する考慮事項: SET PATH ステートメントは、コミット可能な操作ではありません。 ROLLBACK は、CURRENT PATH には影響を与えません。

SQL パスの内容についての規則:

- スキーマ名は、パス内で複数回使用できません。
- 指定可能なスキーマの数は、CURRENT PATH 特殊レジスターの合計長によって制限されます。特殊レジスター・ストリングは、指定された各スキーマ名から末尾ブランクを除去し、二重引用符によって区切り、コンマで各スキーマ名を区切って作成します。作成されたストリングの長さが 3483 バイトを超えると、エラーが戻されます。パスには最大 268 のスキーマ名を表示できます。
- 単一のキーワード (USER、PATH、CURRENT_PATH など) を単一のキーワードとして指定することと、区切り文字付き ID として指定することとは違います。単一のキーワードとして指定された特殊レジスターの現行値を SQL パスで使用するように指定するには、その特殊レジスターの名前をキーワードとして指定します。代わりに、特殊レジスターの名前を区切り文字付き ID ("USER" など) として指定した場合、それはその値のスキーマ名 ('USER') として解釈されます。例えば、USER 特殊レジスターの現行値が SMITH であるとすると、SET PATH = SYSIBM、USER、"USER" の結果は、"SYSIBM"、"SMITH"、"USER" の CURRENT PATH 値となります。
- SET PATH ステートメントに指定した値が変数であるかスキーマ名 であるかは、以下の規則によって判別されます。
  - 名前 が SQL プロシージャー内のパラメーターまたは SQL 変数と等しい場合、名前 はパラメーターまたは SQL 変数として解釈され、名前 内の値が PATH に割り当てられます。
  - 名前 が SQL プロシージャー内のパラメーターまたは SQL 変数と等しくない場合、名前 はスキーマ名 として解釈され、名前 が値として PATH に割り当てられます。

システム・パス: SYSTEM PATH は、プラットフォームのシステム・パスを参照します。スキーマ QSYS、QSYS2、SYSPROC、および SYSIBMADM は指定する必要がありません。これらは、パスに含まれない場合、最後のスキーマとして暗黙的に想定されます (この場合、CURRENT PATH 特殊レジスターに含まれていません)。

CURRENT PATH 特殊レジスターの初期値は、活動化グループ内で実行された最初の SQL ステートメントにシステム命名が使用された場合は、*LIBL になります。最初の SQL ステートメントに SQL 命名が使用された場合、初期値は "QSYS"、"QSYS2"、"SYSPROC"、"X" (X は USER 特殊レジスターの値) になります。

SQL パスの使用: CURRENT PATH 特殊レジスターは、動的 SQL ステートメント内でユーザー定義タイプ、関数、およびプロシージャーを解決するために使用されます。詳しくは、 72 ページの『SQL パス』を参照してください。

例

次のステートメントは、CURRENT PATH 特殊レジスターを設定します。

```
SET PATH = FERMAT, "McDuff", SYSIBM
```

次のステートメントは、SQL パス特殊レジスターの現行値を検索して、CURPATH というホスト変数に入れます。

```
EXEC SQL VALUES (CURRENT PATH) INTO :CURPATH;
```

直前の例で設定されている場合、この値は "FERMAT"、"McDuff"、"SYSIBM" となります。

SET RESULT SETS

SET RESULT SETS ステートメントでは、プロシージャーから返すことができる結果セットを指定します。

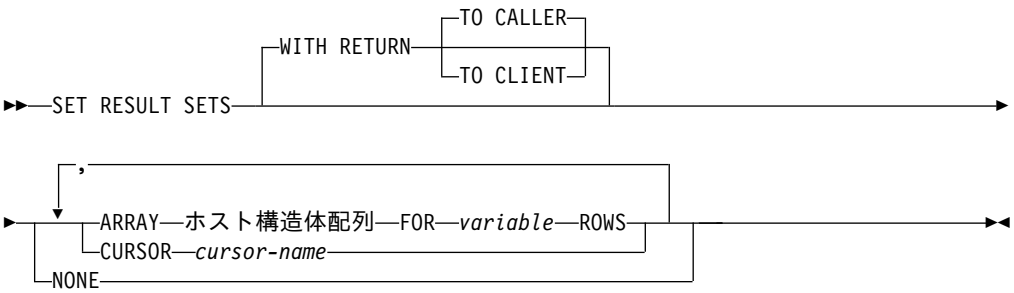
呼び出し

アプリケーション・プログラムまたは SQL プロシージャーに組み込むことが、このステートメントの唯一の使用方法です。これは実行可能ステートメントですが、動的に準備することはできません。このステートメントは、Java または REXX プロシージャーでは使用できません。

権限

権限は不要です。

構文



説明

WITH RETURN
カーソルの結果表を、プロシージャーから戻される結果セットとして使用するように指定します。

スクロール可能ではないカーソルの場合、結果セットには、現行カーソル位置から結果表の最後までのすべての行が含まれます。スクロール可能なカーソルの場合、結果セットには、結果表のすべての行が含まれます。

TO CALLER
カーソルがプロシージャーの呼び出し側に結果セットを戻せることを指定します。例えば、呼び出し側がクライアント・アプリケーションである場合、結果セットはそのクライアント・アプリケーションに戻されます。

TO CLIENT
カーソルがクライアント・アプリケーションに結果セットを戻せることを指定します。このカーソルは、中間にネストされたプロシージャーからは見えません。関数またはトリガーが直接または間接的にプロシージャーを呼び出すと、結果セットはクライアントに返されず、プロシージャーの終了後にカーソルがクローズされます。

CURSOR *cursor-name*
プロシージャーから戻される可能性がある結果セットを定義するために使用するカーソルを識別します。このカーソル名 は、 DECLARE CURSOR ステートメ

ントに関する 1354 ページの『説明』で説明している宣言されたカーソルを識別していなければなりません。SET RESULT SETS ステートメントの実行時点では、カーソルはオープン状態であることが必要です。割り当てカーソルは指定できません。

ARRAY *host-structure-array*
ホスト構造体配列 は、ホスト構造体の宣言に関する規則に従って定義されているホスト構造体の配列を識別します。この配列には、C の NUL で終了するホスト変数を入れることはできません。

配列の最初の構造体が最初の行に対応し、配列の 2 番目の構造体が 2 番目の行に対応するというように、順番に対応しています。さらに、行の最初の値が構造体内の最初の項目に対応し、行の 2 番目の値が構造体の 2 番目の項目に対応するというように、これも順番に対応しています。

DRDA を使用する場合に、LOB と XML を配列で返すことはできません。

1 つの SET RESULT SET ステートメントに指定できるのは 1 つの配列のみで、これにはプロシージャーへのネストされた呼び出しからの RETURN TO CLIENT 配列結果セットがすべて含まれます。

FOR *variable* **ROWS**
結果セットの行数を指定します。変数 は、位取りがゼロの数値変数であることが必要であり、標識変数を含んでいてはなりません。変数をグローバル変数にすることはできません。指定された行の数は、ホスト構造体配列の次元以下でなければなりません。

NONE
結果セットを戻さないことを指定します。プロシージャーが終了した際に、オープン状態のカーソルは戻されません。

注

結果セットの詳細については、プロシージャーからの結果セットおよび WITH RETURN 文節を参照してください。

外部プロシージャー**:** 外部プロシージャーから結果セットを戻すには、次の 3 つの方法があります。

- SET RESULT SETS ステートメントがプロシージャーで実行される場合は、その SET RESULT SETS ステートメントが結果セットを識別します。結果セットは、SET RESULT SETS ステートメントで指定した順序で戻されます。
- SET RESULT SETS ステートメントがプロシージャーで実行されない場合
  - WITH RETURN 文節でカーソルが指定されていない場合、プロシージャーがオープンし、オープン状態のまま戻す各カーソルが、それぞれ 1 つの結果セットを識別します。結果セットは、カーソルがオープンされた順序で戻されます。
  - WITH RETURN 文節でカーソルが指定されている場合、WITH RETURN 文節で定義されたカーソルのうち、プロシージャーがオープンし、オープン状態のまま戻す各カーソルが、それぞれ 1 つの結果セットを識別します。結果セットは、カーソルがオープンされた順序で戻されます。

オープン・カーソルを使用して結果セットが戻される場合、現行カーソル位置から始まる行が戻されます。

プロシージャーから結果セットを戻すには、ALTER PROCEDURE (外部)、CREATE PROCEDURE (外部) ステートメント、または DECLARE PROCEDURE ステートメントで RESULT SETS 文節を指定してください。戻される結果セットの最大数は、ALTER PROCEDURE (外部)、CREATE PROCEDURE (外部) ステートメントまたは DECLARE PROCEDURE ステートメントに指定した数を超えることはできません。

SQL プロシージャー**:** SQL プロシージャーから結果セットを戻すためには、RESULT SETS 文節を指定してプロシージャーを作成する必要があります。 WITH RETURN 文節で定義されたカーソルのうち、プロシージャーがオープンし、オープン状態のまま戻す各カーソルが、それぞれ 1 つの結果セットを識別します。

- プロシージャー内で SET RESULT SETS ステートメントが実行される場合は、その SET RESULT SETS ステートメントに指定されている結果セットが戻されます。結果セットは、SET RESULT SETS ステートメントで指定した順序で戻されます。
- プロシージャー内で SET RESULT SETS ステートメントが実行されない場合は、結果セットはカーソルがオープンされた順序で戻されます。

オープン・カーソルを使用して結果セットが戻される場合、現行カーソル位置から始まる行が戻されます。

SQL プロシージャーからなんらかの結果セットを戻すには、 CREATE PROCEDURE (SQL) ステートメントで RESULT SETS 文節を指定する必要があります。戻される結果セットの最大数は、CREATE PROCEDURE ステートメントに指定した数を超えることはできません。

例

次の SET RESULT SETS ステートメントは、カーソル X を、プロシージャーが呼び出されるときに戻される結果セットとして指定します。ODBC クライアントからの結果セットの使用についての詳細な説明と例については、「IBM i Access Family」トピック集を参照してください。

```
EXEC SQL SET RESULT SETS CURSOR X;
```

SET SCHEMA

SET SCHEMA ステートメントは、CURRENT SCHEMA 特殊レジスターの値を変更します。

呼び出し

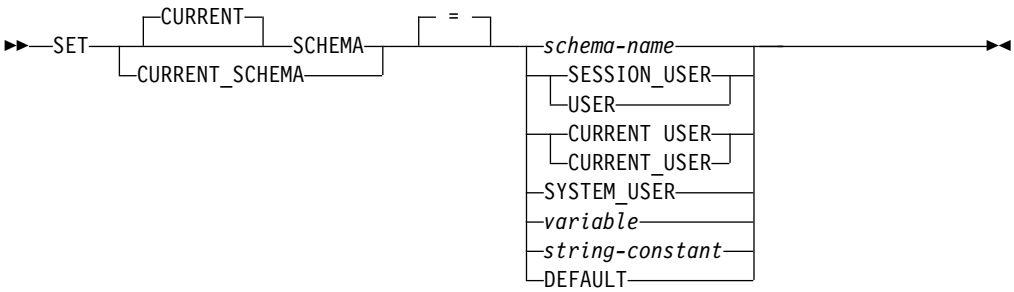
このステートメントは、アプリケーション・プログラムに組み込むことができ、また対話式に呼び出すこともできます。これは、動的に準備できる実行可能ステートメントです。

権限

ステートメント内のグローバル変数を参照する場合は、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別されるグローバル変数に対して、
  - そのグローバル変数に対する READ 特権
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

構文



説明

schema-name
スキーマを識別します。指定されたスキーマ名がシステム・スキーマ名である場合、現行スキーマの設定時には、そのスキーマが存在するかどうかについての検査は行われません。指定されたスキーマ名がシステム・スキーマ名ではない場合、現行スキーマの設定時に、そのスキーマが存在していなければなりません。

指定された値がスキーマ名 に関する規則と一致しない場合、エラーが戻されます。

SESSION_USER または **USER**
この値は SESSION_USER 特殊レジスターです。

CURRENT USER
CURRENT USER 特殊レジスターの値を指定します。

SYSTEM_USER
この値は SYSTEM_USER 特殊レジスターです。

variable
スキーマ名を含む変数を指定します。スキーマ名で修飾すれば、グローバル変数を使用することもできます。内容が大文字に変換されることはありません。

変数は、次の条件に合っていなければなりません。

- 文字ストリングまたは Unicode グラフィック変数でなければなりません。末尾ブランクを切り取った後の *variable* の内容の実際の長さは、スキーマ名の長さを超えてはなりません。 1845 ページの『付録 A. SQL の制約』を参照してください。
- その後に標識変数が続くことはできません。
- NULL 値とすることはできません。
- 通常の ID または区切り文字付き ID の形成に関する規則に従っていなければなりません。
- 変数が固定長である場合、右側はブランクで埋め込まれています。
- キーワード SESSION_USER、CURRENT_USER、SYSTEM_USER、または USER であってはなりません。
- SET ステートメントが SQL ルーチン、SQL トリガー、または複合ステートメントで指定された場合、 *variable* を非修飾名のグローバル変数にすることはできません。

string-constant
スキーマ名を含む文字定数。

ストリング定数は次の条件を満たしている必要があります。

- 末尾ブランクを切り取った後の長さが、スキーマ名の最大長を超えないこと。
- スキーマ名が左寄せで入っており、通常 ID または区切り文字付き ID の形式の規則に準拠していること。
- キーワード SESSION_USER、CURRENT_USER、 SYSTEM_USER、または USER であってはなりません。

DEFAULT
CURRENT SCHEMA は初期値に設定されます。 SQL 命名規則の場合の初期値は USER です。システム命名規則の場合の初期値は *LIBL です。

注

キーワードに関する考慮事項**:** 単一のキーワード (USER など) を単一のキーワードとして指定することと、区切り文字付き ID として指定することとは違います。USER 特殊レジスターの現行値を現行スキーマの設定のために使用することを指示するには、USER をキーワードとして指定してください。代わりに USER を区切り文字付き ID ("USER" など) として指定した場合、それはその値のスキーマ名 ("USER") として解釈されます。

トランザクションに関する考慮事項**:** SET SCHEMA ステートメントは、コミット可能な操作ではありません。ROLLBACK は、CURRENT SCHEMA には影響を与えません。

その他の特殊レジスターに対する影響**:** CURRENT SCHEMA 特殊レジスターの設定により、CURRENT PATH 特殊レジスターが影響を受けることはありません。し

たがって、SQL パスには CURRENT SCHEMA は組み込まれないため、関数、プロシージャー、およびユーザー定義タイプの解決でこれらのオブジェクトが見つからないことがあります。現行スキーマの値を SQL パスに組み込むには、SET SCHEMA ステートメントを発行するときに、必ず、SET SCHEMA ステートメントからのスキーマ名を含む SET PATH ステートメントも発行するようにしてください。

CURRENT SCHEMA: CURRENT SCHEMA 特殊レジスターの値は、DYNDFTCOL が指定されているプログラムの場合を除き、すべての動的 SQL ステートメントの中の一部の非修飾名の修飾子として使用されます。プログラム内で DYNDFTCOL が指定されている場合は、そのスキーマ名が CURRENT SCHEMA のスキーマ名の代わりに使用されます。名前の修飾については、 72 ページの『非修飾オブジェクト名の修飾』を参照してください。

SQL 命名規則の場合は、CURRENT SCHEMA 特殊レジスターの初期値は USER になります。システム命名規則の場合は、CURRENT SCHEMA 特殊レジスターの初期値は '*LIBL' です。

代替の構文**:** CURRENT SCHEMA の同義語として、CURRENT SQLID を使用できます。 SET CURRENT SQLID ステートメントの効果は、SET CURRENT SCHEMA ステートメントと同じです。他の効果 (ステートメント権限の変更など) は発生しません。

SET SCHEMA を使用することは、QSQCHGDC API を呼び出すことと同じです。

例

例 1: 次のステートメントは、CURRENT SCHEMA 特殊レジスターを設定します。

```
SET SCHEMA = RICK
```

例 2: 次の例では、CURRENT SCHEMA 特殊レジスターの現行値を検索して、CURSCHEMA というホスト変数に入れます。

```
EXEC SQL VALUES(CURRENT SCHEMA) INTO :CURSCHEMA
```

値は、例えば例 1 で設定された RICK です。

SET SESSION AUTHORIZATION

SET SESSION AUTHORIZATION ステートメントは、SESSION_USER および USER 特殊レジスターの値を変更します。また、現行スレッドに関連したユーザー・プロファイルの名前も変更します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、対話式に呼び出すことも可能です。これは、動的に準備できる実行可能ステートメントです。REXX で指定してはなりません。

SET SESSION AUTHORIZATION は、SQL トリガー、SQL 関数、または SQL プロシージャーでは使用できません。

権限

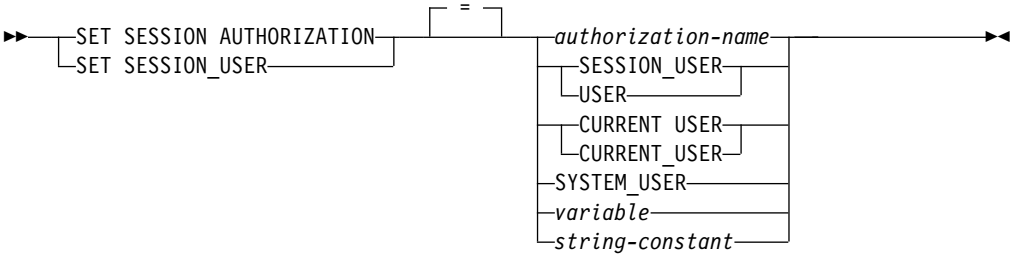
ステートメントで指定された権限名が SYSTEM_USER 特殊レジスターの値と異なる場合、このステートメントの権限 ID によって保持される特権には、システム権限 *ALLOBJ が含まれていなければなりません。

ステートメントで指定された権限名が SYSTEM_USER 特殊レジスターと同じ場合、このステートメントを実行するための権限は不要です。

ステートメント内のグローバル変数を参照する場合は、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別されるグローバル変数に対して、
  - そのグローバル変数に対する READ 特権
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

構文



説明

authorization-name

SESSION_USER 特殊レジスターの新しい値として使用される権限 ID と、実行時の権限 ID を識別します。

権限 ID は、現行サーバーに存在する有効なユーザー・プロファイルまたはグループ・ユーザー・プロファイルでなければなりません。ユーザー・プロファイ

ル・ハンドルを持たないいくつかのシステム・ユーザー・プロファイルは使用できません。詳しくは、『Get Profile Handle API』を参照してください。

CURRENT USER
SESSION_USER 特殊レジスターおよび実行時の権限 ID は、CURRENT USER 特殊レジスターに設定されます。

SESSION_USER または USER
SESSION_USER 特殊レジスターおよび実行時の権限 ID は、USER 特殊レジスターに設定されます。

SYSTEM_USER
SESSION_USER 特殊レジスターおよび実行時の権限 ID は、SYSTEM_USER 特殊レジスターに設定されます。

variable
権限 ID 名を含む変数。スキーマ名で修飾すれば、グローバル変数を使用することもできます。

変数は、次の条件に合っていなければなりません。

- 文字ストリング変数です。
- 変数 に標識変数が関連付けられている場合、標識変数の値は NULL 値を示すものであってはなりません。
- 左寄せされている権限 ID を含み、通常のまたは区切り文字付き ID の形成の規則に従っています。
- 右側はブランクで埋め込まれています。
- NULL 値とすることはできません。
- キーワード USER、SESSION_USER、SYSTEM_USER、または CURRENT_USER であってはなりません。

string-constant
権限 ID を含む文字定数。

注

SET SESSION AUTHORIZATION のその他の影響**:** SET SESSION AUTHORIZATION によって次の現象が発生します。

- 作業単位中にオープンされたすべてのカーソルがクローズされます。
- すべての LOB ロケーターは解放されます。
- この作業単位のコミットメント定義のもとで獲得されたロックはすべて、解放されます。
- 準備済みステートメントはすべて破棄されます。
- SQL 記述子域はすべて割り振り解除されます。
- プロシージャーの結果セットはすべて消去されます。
- 暗号化パスワードはリセットされます。
- オープンされたネイティブ・データベース・ファイルおよび統合ファイル・システム (IFS) ファイルはすべてクローズされます (ソケット、NTC セッション、およびメモリー・マップを含む)。

SET SESSION AUTHORIZATION の実行時に、グローバル変数および宣言済み一時表を含む、その他のリソースが保存されます。SET SESSION AUTHORIZATION ステートメントを実行する前に、すべての宣言済み一時表を削除または消去し、グローバル変数を消去することが推奨されています。

SET SESSION AUTHORIZATION に関する制約事項**:** このステートメントは、作業単位中にバックアウトされる可能性のある作業になる最初のステートメントとしてのみ発行可能です。次の実行可能ステートメントは、SET SESSION AUTHORIZATION を実行する前に発行することができます。

- すべての SQL トランザクション・ステートメント
- すべての SQL 接続ステートメント
- すべての SQL セッション・ステートメント
- GET DIAGNOSTICS

現行サーバーへの接続以外の接続が存在する場合 (デフォルトの活動化グループ以外のすべてのローカル接続を含む)、SET SESSION AUTHORIZATION は使用できません。

保留中のカーソルがオープンされているか、または保留中のロケーターが存在する場合、SET SESSION AUTHORIZATION は使用できません。

SET SESSION AUTHORIZATION の有効範囲**:** SET SESSION AUTHORIZATION の有効範囲は現行スレッドです。その他のアプリケーション・プロセスのスレッドは影響を受けません。

例

例 *1:* 以下のステートメントは、SESSION_USER 特殊レジスターを設定します。

```
SET SESSION_USER = RAJIV
```

例 *2:* セッション許可 ID (SESSION_USER 特殊レジスター) を、ステートメント発行元の接続を確立する際に使用されたシステム許可 ID の値にします。

```
SET SESSION AUTHORIZATION SYSTEM_USER
```

SET TRANSACTION

SET TRANSACTION ステートメントは、現行の作業単位の分離レベル、読み取り専用属性、または診断領域サイズを設定します。

呼び出し

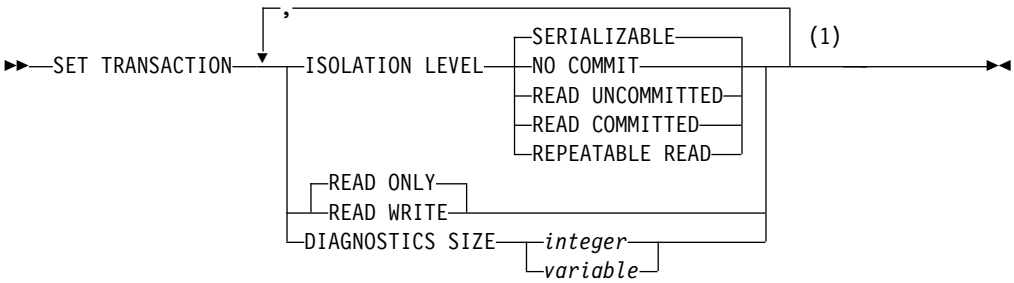
このステートメントは、アプリケーション・プログラムに組み込むことも、対話式に呼び出すことも可能です。これは、動的に準備できる実行可能ステートメントです。

権限

ステートメント内のグローバル変数を参照する場合は、ステートメントの権限 ID が保持する特権に、少なくとも次のいずれか 1 つが含まれなければなりません。

- ステートメント内で識別されるグローバル変数に対して、
  - そのグローバル変数に対する READ 特権
  - グローバル変数が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

構文



注:

1 ISOLATION LEVEL 文節は 1 つだけ指定でき、 READ WRITE または READ ONLY 文節はどちらか一方を 1 つだけ指定でき、 DIAGNOSTICS SIZE 文節は 1 つだけ指定できます。

説明

ISOLATION LEVEL
トランザクションの分離レベルを指定します。 ISOLATION LEVEL 文節を指定しなかった場合は、ISOLATION LEVEL SERIALIZABLE が暗黙指定されます。

NO COMMIT
分離レベル NC (COMMIT(*NONE)) を指定します。

READ UNCOMMITTED
分離レベル UR (COMMIT(*CHG)) を指定します。

READ COMMITTED
分離レベル CS (COMMIT(*CS)) を指定します。

REPEATABLE READ
分離レベル RS (COMMIT(*ALL)) を指定します。

SERIALIZABLE
分離レベル RR (COMMIT(*RR)) を指定します。

READ WRITE または **READ ONLY**
このトランザクションでデータ変更操作が許されるかどうかを指定します。

READ WRITE
すべての SQL 操作が許されることを指定します。 ISOLATION LEVEL READ UNCOMMITTED を指定した場合以外は、これがデフォルト値です。

READ ONLY
SQL データを変更しない SQL 操作のみが許されることを指定します。 ISOLATION LEVEL READ UNCOMMITTED を指定した場合は、これがデフォルト値です。

DIAGNOSTICS SIZE
現在のトランザクションに対する GET DIAGNOSTICS 条件領域の最大値を指定します。 GET DIAGNOSTICS ステートメント情報項目 MORE は、ステートメントが現行トランザクションの条件領域の最大数を超過する場合、現行のステートメントに関して 'Y' に設定されます。診断領域の最大サイズは 90K です。 指定する条件領域の最大数は、1 から 32767 までの値でなければなりません。

integer
現在のトランザクションに対する条件領域の最大値を指定する整数定数。

variable
現在のトランザクションに対する条件領域の最大値を含む変数を識別します。変数はゼロのスケールの数値変数でなければならず、標識変数が続いてはなりません。

注

SET TRANSACTION の有効範囲**:** SET TRANSACTION ステートメントは、そのプロセスの現行活動化グループの SQL ステートメントの分離レベルを設定します。その活動化グループのコミットメント制御の有効範囲がそのジョブの範囲である場合、 SET TRANSACTION ステートメントは同一のジョブ・コミット有効範囲を持つ他の活動化グループすべての分離レベルを設定します。

SQL ステートメント内に分離節が指定されている場合、その分離レベルがトランザクション分離レベルをオーバーライドし、動的 SQL ステートメントに使用されます。

SET TRANSACTION ステートメントの有効範囲は、そのステートメントが実行される文脈に基づいています。トリガーで SET TRANSACTION ステートメントが実行される場合は、指定した分離レベルは、別の SET TRANSACTION ステートメントが実行されるか、またはそのトリガーが終了するかのいずれかが起こるまで、後続のすべての SQL ステートメントに適用されます。 SET TRANSACTION ステートメントがトリガーの外部で実行される場合は、指定した分離レベルは、COMMIT

または ROLLBACK 操作が行われるまで、後続のすべての SQL ステートメントに適用されます (ただし、トリガー内にあって、SET TRANSACTION の後で実行されるステートメントを除きます)。

分離レベルの詳細については、 32 ページの『分離レベル』を参照してください。

SET TRANSACTION の制約事項: SET TRANSACTION ステートメントは、以下の場合を除き、作業単位の最初の SQL ステートメントである場合にのみ実行することができます。

- この作業単位で以前に実行されたすべてのステートメントが、 SET TRANSACTION ステートメントまたは分離レベル NC で実行されたステートメントである場合、または
- それがトリガーによって実行された場合。

トリガーでは、READ ONLY を指定した SET TRANSACTION はコミット境界でのみ使用できます。トリガーでは、いつでも SET TRANSACTION ステートメントを実行することができますが、そのトリガーの最初のステートメントとして実行することをお勧めします。 SET TRANSACTION ステートメントがトリガー内で役立つのは、トリガーの中の SQL ステートメントの分離レベルを、そのトリガーを起動させたアプリケーションと同じレベルに設定する場合です。

現行接続がリモート・アプリケーション・サーバーとの接続である場合、SET TRANSACTION ステートメントは、現行サーバーのトリガーに入っていない限り使用できません。SET TRANSACTION ステートメントが実行されると、該当の作業単位がコミットまたはロールバックされるまで、 CONNECT および SET CONNECTION ステートメントは使用できません。

2 次スレッド内の最初のステートメントとして SET TRANSACTION を使用することはできません。

SET TRANSACTION ステートメントは、SET TRANSACTION ステートメントの実行時にまだオープンされている WITH HOLD カーソルに対しては無効です。

代替構文: 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- NO COMMIT の同義語として、キーワード NC または NONE を使用できます。
- READ UNCOMMITTED の同義語として、キーワード UR および CHG を使用できます。
- READ COMMITTED の同義語として、キーワード CS を使用することができます。
- REPEATABLE READ の同義語として、キーワード RS または ALL を使用できます。
- SERIALIZABLE の同義語として、キーワード RR を使用できます。

例

例 *1:* 次の SET TRANSACTION ステートメントは、分離レベルを NONE に設定します (SQL プリコンパイラーのコマンドで *NONE を指定するのと同等です)。

```
EXEC SQL SET TRANSACTION ISOLATION LEVEL NO COMMIT;
```

例 *2:* 次の SET TRANSACTION は、分離レベルを SERIALIZABLE に設定します。

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
```

SET 変数

SET 変数ステートメントは、1 行以内で構成される結果表を作成し、その行の値を変数に割り当てます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができます。これは、設定されているすべての変数がグローバル変数の場合には、動的に準備可能な実行可能ステートメントです。REXX で指定してはなりません。

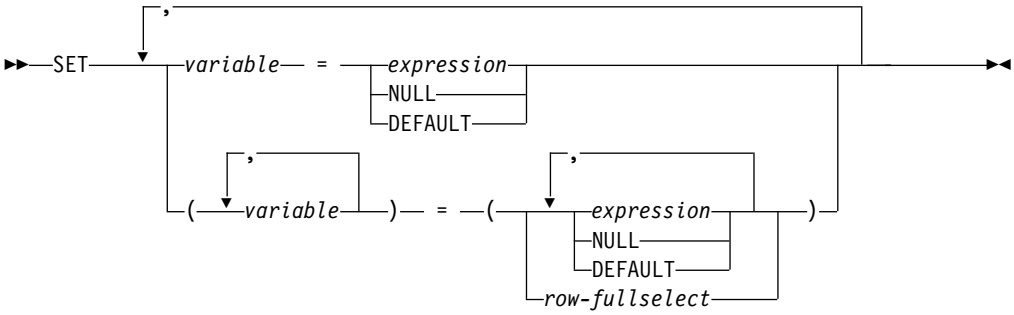
権限

行全選択 が指定されている場合、 841 ページの『全選択』で各副選択に必要な権限についての説明を参照してください。

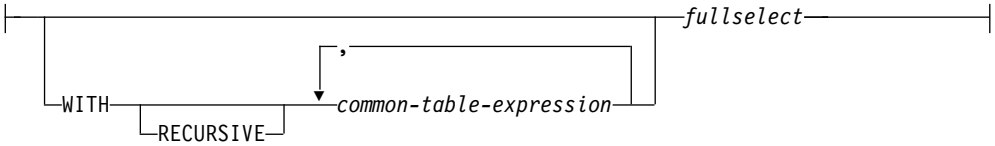
割り当ての左側にグローバル変数が指定される場合、ステートメントの権限 ID が保持する特権に以下の 1 つ以上が含まれている必要があります。

- グローバル変数に対する WRITE 特権
- データベース管理者権限

構文



行全選択:



説明

変数，...

1 つ以上の変数、あるいはホスト構造体を指定します。これらは、変数の宣言に関する規則に従って宣言されていなければなりません (172 ページの『ホスト変数に対する参照』を参照)。ホスト構造体は、ホスト構造体の各エレメントを表す変数のリストによって、論理的に置き換えられます。

各変数 に割り当てられる値は、変数 の直後に指定することができます。例えば、変数 = 式、変数 = 式 のように指定します。または、対になっている括弧

を使用すると、すべての変数 とすべての値を指定することができます。つまり、(変数、変数) = (式、式) などのように指定します。

各変数のデータ・タイプは、それぞれに対応する結果列と互換性がなければなりません。これらの割り当てはそれぞれ、 113 ページの『割り当ておよび比較』で説明されている規則に従って行われます。等号演算子の左側に指定する変数の数は、それに対応して等号演算子の右側に指定されている結果の値の数と同じでなければなりません。値が NULL の場合は、標識変数が用意されている必要があります。割り当てでエラーが起こった場合、その値は変数に割り当てられず、それ以後の変数への値の割り当ては行われません。ただし、変数に既に割り当てられている値があれば、その値は割り当てられたままです。

副選択の式 あるいは SELECT リストの算術式の結果、エラーが発生した (ゼロによる除算やオーバーフローなど) 場合、または文字変換エラーが起こった場合、結果は NULL 値になります。他の NULL 値の場合と同様に、標識変数を用意しなければなりません。該当の変数の値は、未定義になります。ただし、この場合、標識変数は -2 にセットされます。ステートメントの処理は、エラーが起こらなかったものとして続行されます。(ただし、警告が戻されます。) 標識変数を用意していない場合は、エラーが戻されます。エラーが生じた時点で、既にいくつかの値が変数に割り当てられていることがあり、それらの値は割り当てられたままになります。

expression
変数の新しい値を指定します。*expression* は、 196 ページの『式』で説明したタイプの任意の式です。この式の中で列名を使用してはなりません。

NULL
変数の新しい値を NULL 値にすることを指定します。

DEFAULT
変数の新しい値をデフォルトの初期値にすることを指定します。 DEFAULT を割り当てることができるのは、グローバル変数に対してのみです。DEFAULT が使用される場合、SET ステートメントに割り当てることができる変数は 1 つだけです。

row-fullselect
1 つの結果行を戻す全選択。結果列の値は、対応する各変数 に割り当てられます。全選択の結果に行が含まれない場合、NULL 値が割り当てられます。結果の中に複数の行がある場合には、エラーが戻されます。

WITH *common-table-expression*
共通表式を指定します。共通表式については、 848 ページの『共通表式』を参照してください。

fullselect
1 つの結果行を戻す *fullselect*。結果列の値は、対応する各変数 に割り当てられます。 *fullselect* の結果に行が含まれない場合、NULL 値が割り当てられます。結果の中に複数の行がある場合には、エラーが戻されます。

注

変数の割り当て**:** 変数として文字変数を指定し、その変数が、結果を収容するのに十分な大きさを持っていない場合には、警告 (SQLSTATE 01004) が戻され (そして SQLCA の SQLWARN1 に 'W' が割り当てられ) ます。標識変数が用意されてい

る場合、結果の実際の長さは、その変数に関連する標識変数に戻されます。

変数として C の NUL で終了する変数を指定し、その変数が、結果および NUL 終了文字を入れられるだけの十分な大きさを持っていない場合は、以下のようになります。

- CRTSQLCI コマンドまたは CRTSQLCPPI コマンドに *CNULRQD オプションを指定した場合 (または SET OPTION ステートメントに CNULRQD(*YES) を指定した場合)、以下のようになります。
  - 結果が切り捨てられます。
  - 最後の文字は NUL 終了文字になります。
  - SQLCA の SQLWARN1 に値'W' が割り当てられます。
- CRTSQLCI コマンドまたは CRTSQLCPPI コマンドに *NOCNULRQD オプションを指定した場合 (または SET OPTION ステートメントに CNULRQD(*NO) を指定した場合)、以下のようになります。
  - NUL 終了文字は戻されません。
  - SQLCA の SQLWARN1 に値'N' が割り当てられます。

複数の割り当て**:** 複数の割り当てを同一 SET ステートメント内で指定すると、割り当てを行う前にすべての式 および行全選択 の評価が完全に行われます。そのため、式 または行全選択 でのターゲット変数への参照は、常に SET ステートメントでの割り当ての前のターゲット変数の値となります。

例

例 *1:* CURRENT PATH 特殊レジスターの値を、ホスト変数 HV1 に割り当てます。

```
EXEC SQL SET :HV1 = CURRENT PATH;
```

例 *2:* LOB ロケーター LOB1 が CLOB 値と関連していると想定します。 CLOB 値の一部を、LOB ロケーターを使用してホスト変数 DETAILS に割り当てます。

```
EXEC SQL SET :DETAILS = SUBSTR(:LOB1,1,35);
```

SIGNAL

SIGNAL ステートメントは、エラー条件または警告条件を通知します。これは、指定の SQLSTATE とオプションの条件情報項目 を使用して、エラーまたは警告を戻します。

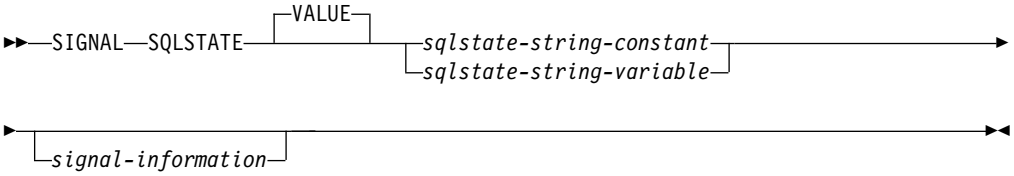
呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。これは実行可能ステートメントですが、動的に準備することはできません。REXX で指定してはなりません。

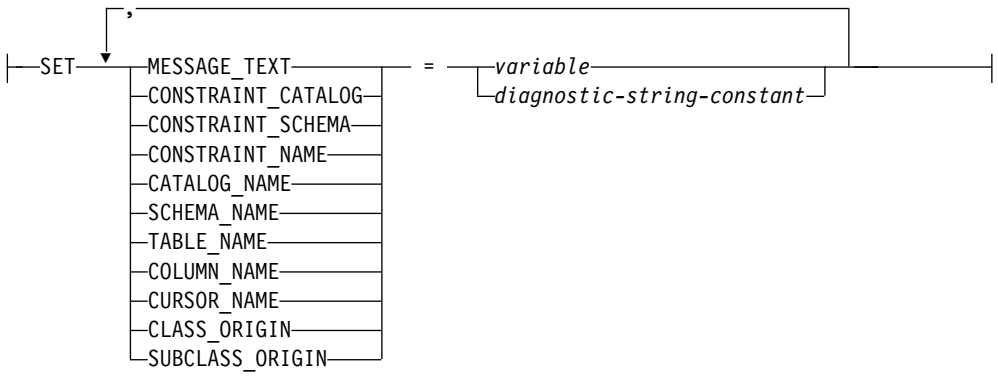
権限

権限は不要です。

構文



signal-information:



説明

SQLSTATE VALUE

通知する SQLSTATE を指定します。指定値は NULL 不可で、次の SQLSTATE の規則に従わなければなりません。

- 各文字は、数字 ('0' から '9') またはアクセント記号なしの英大文字 ('A' から 'Z') でなければなりません。
- SQLSTATE クラス (最初の 2 文字) は、'00' であってはなりません (これは正常終了を表します)。

SQLSTATE がこの規則に従っていないと、エラーが戻されます。

SQLSTATE ストリング定数
SQLSTATE ストリング定数 は、厳密に 5 文字の文字ストリング定数でなければなりません。

SQLSTATE ストリング変数
SQLSTATE ストリング変数 は、文字または Unicode グラフィック変数でなければなりません。この変数は、グローバル変数にすることはできません。変数 の内容の実際の長さは、5 でなければなりません。

SET
条件情報項目 への値の割り当てを指定します。条件情報項目 値は、 GET DIAGNOSTICS ステートメントを使用してアクセスできます。 SQLCA 内でアクセス可能な条件情報項目 は、MESSAGE_TEXT だけです。

MESSAGE_TEXT
エラーまたは警告を説明するストリングを指定します。

SQLCA を使用する場合、

- このストリングは、SQLCA の SQLERRMC フィールドに戻されます。
- ストリングの実際の長さが 1000 バイトを超える場合、警告せずに切り捨てられます。

CONSTRAINT_CATALOG
通知されたエラーまたは警告に関連する制約を含む、データベースの名前を示すストリングを指定します。

CONSTRAINT_SCHEMA
通知されたエラーまたは警告に関連する制約を含む、スキーマの名前を示すストリングを指定します。

CONSTRAINT_NAME
通知されたエラーまたは警告に関連する制約の名前を示すストリングを指定します。

CATALOG_NAME
通知されたエラーまたは警告に関連する表またはビューを含む、データベースの名前を示すストリングを指定します。

SCHEMA_NAME
通知されたエラーまたは警告に関連する表またはビューを含む、スキーマの名前を示すストリングを指定します。

TABLE_NAME
通知されたエラーまたは警告に関連する表またはビューの名前を示すストリングを指定します。

COLUMN_NAME
通知されたエラーまたは警告に関連する表またはビューの列名を示すストリングを指定します。

CURSOR_NAME
通知されたエラーまたは警告に関連するカーソルの名前を示すストリングを指定します。

CLASS_ORIGIN
通知されたエラーまたは警告に関連する SQLSTATE クラスの起点を示すストリングを指定します。

SUBCLASS_ORIGIN
通知されたエラーまたは警告に関連する SQLSTATE サブクラスの起点を示すストリングを指定します。

variable
変数を指定します。この変数は、変数を宣言する規則に従って宣言されていなければなりません (172 ページの『ホスト変数に対する参照』を参照)。変数には、条件情報項目 に割り当てる値が含まれます。この変数は、CHAR、VARCHAR、Unicode GRAPHIC、または Unicode VARGRAPHIC 変数として定義する必要があります。この変数は、グローバル変数にすることはできません。

診断ストリング定数
条件情報項目 に割り当てる値を含む、文字ストリング定数を指定します。

注

SQLSTATE 値: SIGNAL ステートメントには、任意の有効な SQLSTATE 値を使用できますが、プログラマーは、アプリケーション用に予約されている範囲に基づいて、新規の SQLSTATE を定義することをお勧めします。そうすれば、使用する SQLSTATE 値が今後リリースされるデータベース・マネージャーで定義される値と重なってしまう可能性を防止できます。

SQLSTATE 値は、2 文字のクラス・コード値と、それに続く 3 文字のサブクラス・コード値から構成されます。クラス・コード値は、成否の実行条件のクラスを表します。

- 文字 '7' から '9' または 'I' から 'Z' で始まる SQLSTATE クラスは、定義しても構いません。これらのクラス内では、任意のサブクラスを定義できます。
- 文字 '0' から '6' または 'A' から 'H' で始まる SQLSTATE クラスは、データベース・マネージャー用に予約されています。これらのクラス内では、文字 '0' から 'H' で始まるサブクラスは、データベース・マネージャー用に予約されています。文字 'I' から 'Z' で始まるサブクラスは、定義しても構いません。

SQLSTATE の詳細については、「SQL メッセージおよびコード」トピック集を参照してください。

割り当て: SIGNAL ステートメントが実行されるとき、指定した各ストリング定数および変数 の値は、対応する条件情報項目 に割り当てられます。ただし、ストリング定数 または変数 の長さが対応する条件情報項目 の最大長を超える場合は、警告せずに切り捨てられます。割り当て規則の詳細については、 113 ページの『割り当ておよび比較』を参照してください。特定の条件情報項目 の最大長については、 1489 ページの『GET DIAGNOSTICS』を参照してください。

SIGNAL ステートメントの処理: SIGNAL ステートメントが実行された場合、SQLCODE は、次のように SQLSTATE 値に基づいて設定されます。

- 指定された SQLSTATE クラスが '01' または '02' の場合、警告または NOT FOUND が通知され、SQLCODE は +438 に設定されます。

- それ以外の場合、例外が通知され、SQLCODE は -438 に設定されます。

例

例 *1:* 説明メッセージ・テキストを付加して、 SQLSTATE '75002' を通知します。

```
EXEC SQL SIGNAL SQLSTATE '75002'
            SET MESSAGE_TEXT = 'Customer number is not known';
```

例 *2:* 説明メッセージ・テキストおよびエラーの生じた関連する特定の表を付加して、 SQLSTATE '75002' を通知します。

```
EXEC SQL SIGNAL SQLSTATE '75002'
            SET MESSAGE_TEXT = 'Customer number is not known',
                SCHEMA_NAME = 'CORPDATA',
                TABLE_NAME = 'CUSTOMER';
```

TRANSFER OWNERSHIP

TRANSFER OWNERSHIP ステートメントは、データベース・オブジェクトの所有権を転送します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、あるいは対話式に実行することもできます。これは、動的に準備できる実行可能ステートメントです。

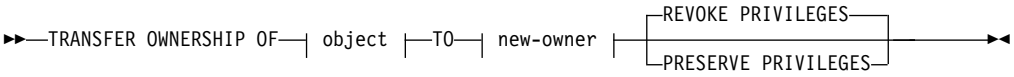
権限

表、ビュー、または索引の所有権を移動するには、ステートメントの権限 ID は対象のオブジェクトの所有者でなければならず、保持する特権には少なくとも次の 1 つが含まれていなければなりません。

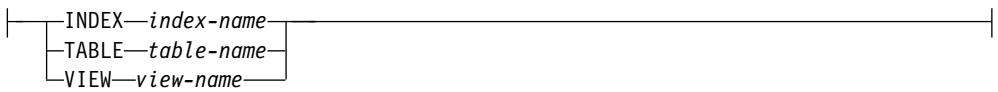
- 次のシステム権限
  - 転送したいオブジェクトについての *OBJOPR および *OBJEXIST システム権限。
  - 転送したいオブジェクトが含まれるスキーマに対する USAGE 特権
- データベース管理者権限
- セキュリティー管理者権限

SQL 特権に対応するシステム権限については、『表またはビューへの権限を検査する際の対応するシステム権限』を参照してください。

構文



オブジェクト:



new-owner:

説明

INDEX *index-name*
所有権を転送する索引を指定します。この索引名 は、現行サーバーに存在している索引を示すものでなければなりません。

TABLE *table-name*
所有権を転送する表を指定します。表名 は、現行サーバーに存在している基本表を示していなければなりませんが、宣言されたグローバル一時表またはカタログ表を示すものであってはなりません。

VIEW *view-name*
所有権を転送するビューを指定します。このビュー名 は、現行サーバーに存在しているビューを識別していなければなりませんが、カタログ・ビューを識別するものであってはなりません。

USER *authorization-name*
オブジェクトの所有権の転送先となる許可 ID を指定します。

CURRENT USER または **CURRENT_USER**
CURRENT USER 特殊レジスターの値を、オブジェクトの所有権の転送先となる許可 ID として使用することを指定します。

SESSION_USER または **USER**
SESSION_USER 特殊レジスターの値を、オブジェクトの所有権の転送先となる許可 ID として使用することを指定します。

SYSTEM_USER
SYSTEM_USER 特殊レジスターの値を、オブジェクトの所有権の転送先となる許可 ID として使用することを指定します。

REVOKE PRIVILEGES または **PRESERVE PRIVILEGES**
所有権が転送された後の、オブジェクトの現行所有者の特権を指定します。

REVOKE PRIVILEGES
現行所有者は転送完了後はオブジェクトに対する明示的な特権を何も保有しなくなることを指定します。

PRESERVE PRIVILEGES
所有権を転送するオブジェクトの現行所有者が、転送後もオブジェクトに対する既存の特権を引き続き保持することを指定します。例えば、ビューの作成者に付与された特権は、所有権が別のユーザーに譲渡された後でも、元の所有者によって引き続き保持されます。

規則

大部分のシステム定義オブジェクトの所有権は譲渡できません。

名前が「SYS」または「Q」で始まるスキーマ内のオブジェクトの所有権は譲渡できません。

セキュリティー管理者権限を持つ権限 ID は、まだオブジェクトの所有者ではない場合はオブジェクトの所有権を自分自身に譲渡することはできません。

注

- 現行所有者がオブジェクトに対して保有しているすべての特権が新規所有者に譲渡されます。
- データベース・オブジェクトの所有権を譲渡する場合、新規所有者はそのオブジェクトに従属するものに対する特権を必ずしも保有するとは限りません。
- オブジェクトの所有権をその所有者に譲渡しようとする場合、警告が戻されます。

例

例 *1:* 表 T1 の所有権を PAUL に転送します。

```
TRANSFER OWNERSHIP OF TABLE WALID.T1
     TO USER PAUL PRESERVE PRIVILEGES
```

Paul は表 WALID.T1 の所有者になり、この表の元の所有者が持っていたすべての特権を付与されます。前の所有者もすべての特権を保持し続けます。

例 *2:* ビュー V1 の所有権を HENRY に譲渡し、前の所有者から特権を除去します。

```
TRANSFER OWNERSHIP OF VIEW V1
  TO USER HENRY
```

Henry はビュー V1 の所有者になり、このビューの元の所有者が持っていたすべての特権を付与されます。前の所有者は、このビューに対する明示的な特権を何も持たなくなります。

TRUNCATE

TRUNCATE ステートメントは、表からすべての行を削除します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことも、あるいは対話式に実行することもできます。これは、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメントで識別される表に対して:
  - その表の DELETE 特権、および
  - 表が含まれるスキーマに対する USAGE 特権
- データベース管理者権限

IGNORE DELETE TRIGGERS オプションが指定される場合、ステートメントの権限 ID が保持する特権には以下が含まれていなければなりません。

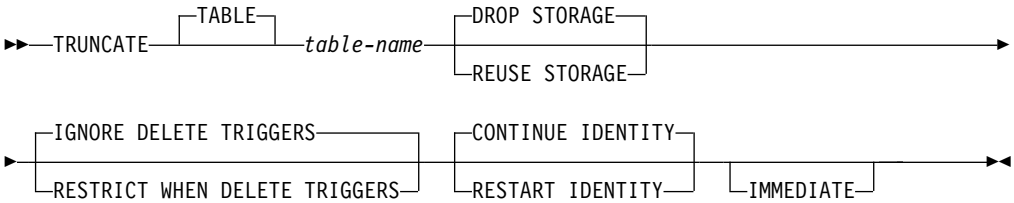
- 表に対する ALTER 特権、および表に対する *OBJOPR システム権限

表に対して行アクセス制御または列アクセス制御がアクティブになっている場合、ステートメントの権限 ID が保持する特権には以下が含まれていなければなりません。

- 表についての *OBJOPR および *OBJEXIST システム権限。

SQL 特権に対応するシステム権限については、『表またはビューへの権限を検査する際の対応するシステム権限』を参照してください。

構文



説明

table-name
削除する行がある表を指定します。この名前は、現行サーバーに存在する表を示すものでなければなりません。また、カタログ表、ビュー、およびシステム期間テンポラル表を示すものであってはなりません。

DROP STORAGE または **REUSE STORAGE**
表に割り振られている既存のストレージをドロップするか、再利用するかを指定します。

DROP STORAGE
表に割り振られているすべてのストレージが解放され、使用可能になります。これはデフォルトです。

REUSE STORAGE
表に割り振られているすべてのストレージは、その表に引き続き割り振られますが、ストレージは空と見なされます。

IGNORE DELETE TRIGGERS または RESTRICT WHEN DELETE TRIGGERS
表に削除トリガーが定義されている場合の処理を指定します。

IGNORE DELETE TRIGGERS
表に定義されている削除トリガーは、切り捨て操作では活動化されないことを指定します。これはデフォルトです。

RESTRICT WHEN DELETE TRIGGERS
表に削除トリガーが定義されている場合はエラーを戻すことを指定します。

CONTINUE IDENTITY または RESTART IDENTITY
ID 列値の生成を処理する方法を指定します。

CONTINUE IDENTITY
表に ID 列が存在する場合、生成される次の ID 列の値は、TRUNCATE ステートメントが実行されなかった場合に生成される次の値に進みます。これはデフォルトです。

RESTART IDENTITY
表に ID 列が存在する場合、生成される次の ID 列値は、その ID 列が定義されたときに指定された初期値です。

IMMEDIATE
切り捨て操作は即時に処理され、取り消しできないことを指定します。

切り捨てられた表は、同じ作業単位で使用できるように直ちに使用可能になります。 ROLLBACK ステートメントは、TRUNCATE ステートメントの実行後に実行できますが、切り捨て操作は取り消されず、表は、切り捨てられた状態のままになります。例えば、TRUNCATE IMMEDIATE ステートメントの実行後に表に対して別のデータ変更操作が実行され、その後で ROLLBACK ステートメントが実行された場合、切り捨て操作は取り消されませんが、その他のデータ変更操作はすべて取り消されます。

いずれかのセッションで、表に対してオープンしているカーソルがあるか、表にロックがかかっている場合、切り捨て操作は実行できません。

IMMEDIATE が無指定の場合、ROLLBACK ステートメントは切り捨て操作を取り消せます。

注

参照整合性**:** 指定される表を参照制約の親表とすることはできません。

削除行数**:** SQL 診断領域内の ROW_COUNT 条件領域項目 (または SQLCA 内の SQLERRD(3)) は、切り捨て操作の場合は -1 に設定されます。表から削除された行の総数は戻されません。

表に行が存在しない場合、SQLSTATE 値「02000」が戻されます。

SQLCA についての説明は、 1867 ページの『付録 C. SQLCA (SQL 連絡域)』を参照してください。

例

例 *1:* 既存のトリガーにかかわらず未使用の在庫表を空にし、表に割り振られていたスペースを戻します。

```
TRUNCATE TABLE INVENTORY
     DROP STORAGE
     IGNORE DELETE TRIGGERS;
```

例 *2:* 既存のトリガーにかかわらず未使用の在庫表を空にしますが、後で再使用できるように、その割り振られていたスペースを保持します。

```
TRUNCATE TABLE INVENTORY
     REUSE STORAGE
     IGNORE DELETE TRIGGERS;
```

例 *3:* 既存のトリガーにかかわらず未使用の在庫表を永久に空にし (IMMEDIATE オプションが指定されている場合は ROLLBACK ステートメントは切り捨て操作を取り消せません)、再使用できるように、表に割り振られていたスペースを保持します。

```
TRUNCATE TABLE INVENTORY
     REUSE STORAGE
     IGNORE DELETE TRIGGERS
     IMMEDIATE;
```

UPDATE

UPDATE ステートメントは、表またはビューの行の指定した列の値を更新します。このビューに対して INSTEAD OF UPDATE トリガーが定義されていない場合、ビューの行を更新すると、そのビューの基本表の行が更新されます。そのようなトリガーが定義されていれば、代わりにこのトリガーが活動化されます。

このステートメントには、以下の 2 つの形式があります。

- 検索条件付き UPDATE 形式は、 1 つ以上の行 (任意指定の検索条件によって決まる) を更新する場合に使用されます。
- 位置指定 UPDATE 形式は、 1 行 (カーソルの現在位置によって決まる) だけを更新する場合に使用されます。

呼び出し

検索 UPDATE ステートメントは、アプリケーションに組み込むか、または対話式に呼び出すことができます。位置指定 UPDATE ステートメントは、アプリケーション・プログラムに組み込んで使用しなければなりません。どちらの形式も、動的に準備できる実行可能ステートメントです。

権限

このステートメントの権限 ID が保持する特権には、少なくとも以下の 1 つが含まれていなければなりません。

- ステートメントに指定された表またはビューに対して、
  - 表やビューに対する UPDATE 特権、または
  - 更新される各列に対する UPDATE 特権、または
  - 表またはビューが含まれるスキーマに対する USAGE 特権
- データベース管理者権限

割り当て文節 の式 にその表またはビューの列に対する参照が含まれている場合、または検索 UPDATE の検索条件 にその表またはビューの列に対する参照が含まれている場合は、ステートメントの権限 ID が保持する特権には、以下のいずれか 1 つも含まれていなければなりません。

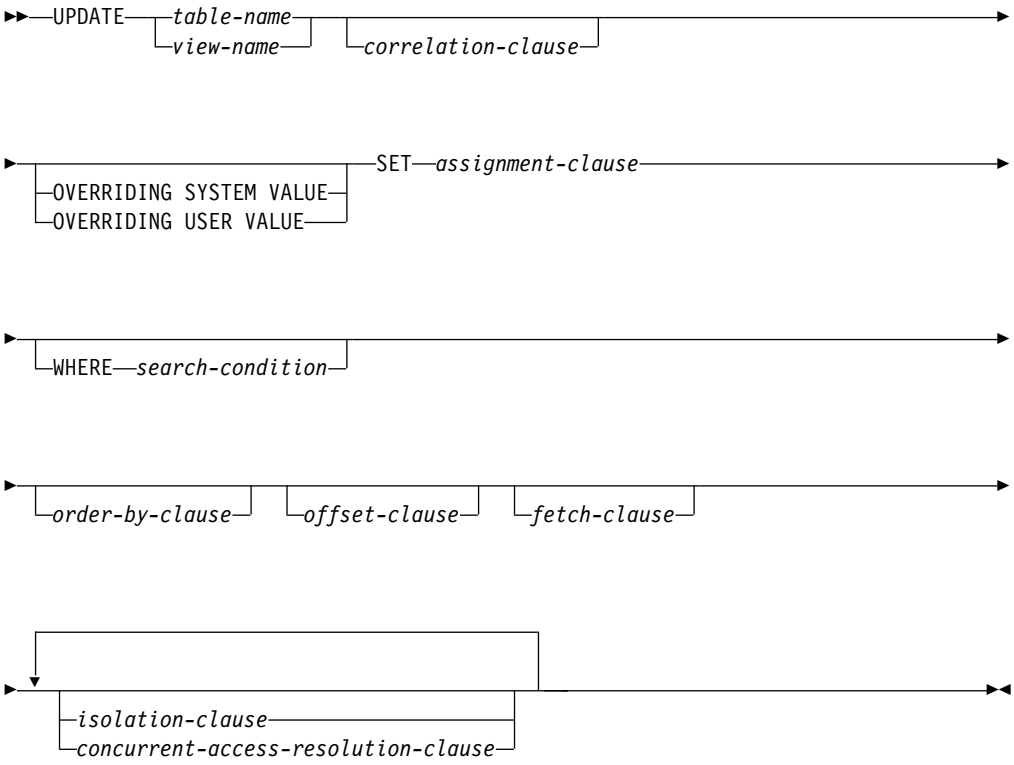
- その表またはビューについての SELECT 特権
- セキュリティー管理者権限

検索条件 に副照会が含まれている場合、または割り当て文節 にスカラー全選択 か行全選択 が含まれている場合、 787 ページの『第 6 章 照会』で、各副選択に必要な権限の説明を参照してください。

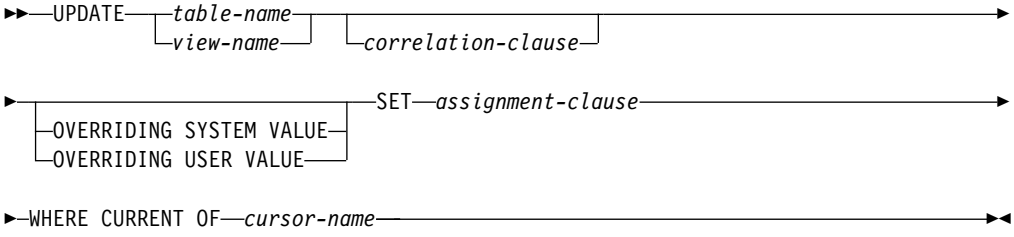
SQL 特権に対応するシステム権限については、『表またはビューへの権限を検査する際の対応するシステム権限』を参照してください。

構文

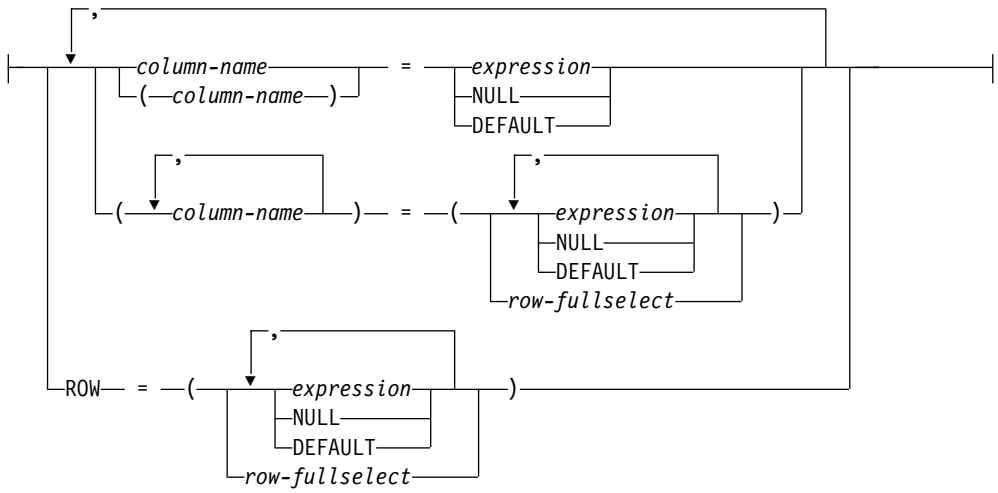
検索 **UPDATE:**



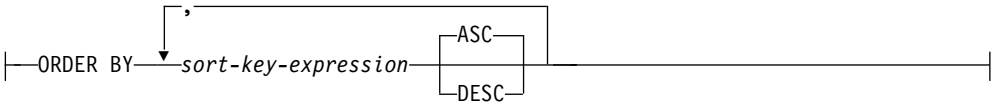
位置指定 **UPDATE:**



割り当て節:



ORDER BY 文節:



ISOLATION 文節:

説明

table-name または *view-name*
更新する表またはビューを指定します。この名前は、現行サーバーに存在している表またはビューを識別していなければなりませんが、履歴表、カタログ表、カタログ表のビュー、または読み取り専用のビューを識別するものであってはなりません。読み取り専用ビューおよび更新可能ビューについての説明は、 1342 ページの『CREATE VIEW』を参照してください。

相関文節
表またはビューを示すために、*search-condition* または *assignment-clause* 内で使うことができます。相関文節 の説明については、 794 ページの『table-reference』を参照してください。相関名 の説明については、 163 ページの『相関名』を参照してください。

OVERRIDING SYSTEM VALUE または **OVERRIDING USER VALUE**
ROWID 列、識別列、または行変更タイム・スタンプ列にシステムが生成した値またはユーザーが指定した値を使用するかどうかを指定します。
OVERRIDING SYSTEM VALUE を指定する場合は、SET 文節内の暗黙的または明示的な列リストに、GENERATED ALWAYS として定義された ROWID 列、ID 列、または行変更タイム・スタンプ列が含まれていることが必要です。
OVERRIDING USER VALUE を指定する場合は、SET 文節内の暗黙または明示的な列リストに、GENERATED ALWAYS または GENERATED BY DEFAULT として定義された列が含まれていることが必要です。

OVERRIDING SYSTEM VALUE
GENERATED ALWAYS として定義されている ROWID 列、ID 列、または行変更タイム・スタンプ列について、SET 文節に指定されている値を使用することを指定します。システム生成の値は使用されません。

行開始列、行終了列、トランザクション開始 ID 列、または生成式列の値が提供された場合、それは DEFAULT でなければなりません。

OVERRIDING USER VALUE
GENERATED ALWAYS または GENERATED BY DEFAULT として定義されている列について、SET 文節に指定されている値を無視することを指定します。代わりにシステム生成の値が使用され、ユーザー指定の値はオーバーライドされます。

OVERRIDING SYSTEM VALUE と OVERRIDING USER VALUE のどちらも指定しない場合は、以下のようになります。

- GENERATED ALWAYS として定義された ROWID 列、ID 列、行変更タイム・スタンプ列、行開始列、行終了列、トランザクション開始 ID 列、および生成式列に、値を指定することはできません。
- GENERATED BY DEFAULT として定義されている ROWID 列については、値を指定することができます。値を指定した場合は、この列にその値が割り当てられます。ただし、BY DEFAULT として定義された ROWID 列の値を更新できるのは、指定した値が Db2for z/OS または Db2 for i により既に生成されている有効な行 ID 値である場合に限られます。
- GENERATED BY DEFAULT として定義されている識別列または行変更タイム・スタンプ列には、値を指定することができます。 BY DEFAULT として定義された識別列または行変更タイム・スタンプ列の値を更新すると、その識別列または行変更タイム・スタンプ列が固有制約または固有索引内の唯一のキーである場合以外は、データベース・マネージャーは指定された値がその列で固有な値であるかどうかを検査しません。固有制約も固有索引もない場合は、データベース・マネージャーは、NO CYCLE が有効である場合に限り、システム生成の値のセットの中でのみ各値の固有性を保証します。

  値が指定されていない場合は、データベース・マネージャーは新しい値を生成します。

SET

列名への値の割り当てを指定します。

assignment-clause

column-name

更新する列を識別します。列名 は、指定された表またはビューの列を識別するものでなければなりません。拡張標識変数が使用不可の場合、その列は、スカラー関数、定数、または式から得られるビューの列を識別するものであってはなりません。列を複数回指定することはできません。

位置指定 UPDATE の場合 :

- UPDATE 文節をカーソルに関する SELECT ステートメントに指定する場合は、 SET リストのそれぞれの列名を、UPDATE 文節にも指定しなければなりません。
- UPDATE 文節をカーソルに関する SELECT ステートメントに指定しない場合は、更新可能な任意の列の名前を指定することができます。

詳しくは、 856 ページの『UPDATE 文節』を参照してください。

1 つのビューに同じ列から得られる 2 つの列がある場合、その列の値を更新することは可能ですが、その 2 つの列を同一の UPDATE ステートメントで更新することはできません。

列名 のリストを指定する場合は、式、NULL、および DEFAULT の数が列名 の数に一致していなければなりません。

ROW

指定された表またはビューのすべての列 (ただし、隠し属性を指定して定義

された列以外) を識別します。ビューが指定されている場合、そのビューの列がまったく、スカラー関数、定数、または式から派生していない場合があります。

式、NULL、および DEFAULT の数 (または行全選択 からの結果列の数) は、行の列の数と一致していなければなりません。

位置指定 UPDATE の場合、カーソルの SELECT ステートメント内に UPDATE 文節が指定されている場合、その UPDATE 文節にも表またはビューの各列を指定しなければなりません。詳しくは、 856 ページの『UPDATE 文節』を参照してください。

ビューに、そのビューの別の列から派生したビュー列が含まれている場合、そのビューに ROW を指定することはできません。これは、両方の列を同じ UPDATE ステートメント内で更新できないためです。

expression
列の新しい値を指定します。 *expression* は、 196 ページの『式』で説明したタイプの任意の式です。この式の中で、集約関数を使用してはなりません。

式の中の列名 は、指定した表またはビューの列の名前を指定するものでなければなりません。行が更新されるたびに、その行の列の値 (行を更新する前の値) がこの式の列の値になります。

この文節に指定する各変数は、ホスト構造体や変数の宣言の規則に従って宣言されているホスト構造体または変数を識別していなければなりません。このステートメントの操作形式では、ホスト構造体に対する参照は、その個々の変数それぞれに対する参照によって置き換えられます。式 が単一のホスト変数である場合、そのホスト変数には、拡張標識変数で使用可能な標識を組み込むことができます。拡張標識変数が使用可能で、割り当て文節内の式が単一のホスト変数でない場合、拡張標識変数値 DEFAULT および UNASSIGNED を使用してはなりません。変数と構造についての詳細は、172 ページの『ホスト変数に対する参照』および 179 ページの『ホスト構造』を参照してください。ホスト構造を指定した場合、キーワード ROW を指定しなければなりません。

NULL
列の新しい値を NULL 値にすることを指定します。NULL は、NULL 可能列にのみ指定してください。

DEFAULT
列にデフォルト値を割り当てることを指定します。使用される値は、次のように、列がどのように定義されたかによって異なります。

- 式に基づいて生成列として列が定義されている場合は、その式に基づいた列の値がデータベース・マネージャーによって生成されます。
- 列が ROWID 列、ID 列、行開始列、行終了列、またはトランザクション開始 ID 列の場合、データベース・マネージャーは新しい値を生成します。
- WITH DEFAULT 文節が使用される場合、使用されるデフォルト値は、その列に関して定義されている値になります (1238 ページの『CREATE TABLE』の列定義 のデフォルト文節 を参照してください)。

- WITH DEFAULT 文節または NOT NULL 文節が使用されない場合、使用される値は NULL です。
- 列が行変更タイム・スタンプ列として定義されると、新規の行変更タイム・スタンプ値が列に割り当てられます。

NOT NULL 文節が使用されていて、WITH DEFAULT 文節が使用されていない場合、または DEFAULT NULL が使用されている場合、その列については DEFAULT キーワードは指定できません。

DEFAULT は、GENERATED ALWAYS として定義されている ID 列には指定する必要があります。ただし、OVERRIDING SYSTEM VALUE が使用される場合は例外です。

行開始列、行終了列、トランザクション開始 ID 列、または生成式列に設定可能な値は、DEFAULT のみです。

row-fullselect
1 つの結果行を戻す全選択。選択リスト内の結果列の数は、割り当てのために指定された列名 の数 (または ROW が指定されている場合は、その行の列の数) と一致していなければなりません。結果列の値は、対応する各列名に割り当てられます。全選択の結果に行が含まれない場合、NULL 値が割り当てられます。結果の中に複数の行がある場合には、エラーが戻されます。

行全選択 には、 UPDATE ステートメントのターゲット表の列に対する参照が含まれている場合があります。行が更新されるたびに、その行の列の値 (行を更新する前の値) がこの式の列の値になります。

WHERE
更新する行を指定します。文節を省略するか、あるいは検索条件 またはカーソル名 を指定できます。この文節を指定しなかった場合は、指定した表またはビューのすべての行が更新されます。

search-condition
275 ページの『検索条件』で説明している、いずれかの検索条件を指定します。検索条件の中のそれぞれの列名 (副照会の中は除く) は、指定した表またはビューにある列の名前を指定するものでなければなりません。
UPDATE と副照会の基本オブジェクトが両方とも同じ表になる場合に、検索条件に副照会が含まれるときは、その副照会の評価が完了してから行が更新されます。

検索条件 は、表またはビューの各行に適用されます。更新された行は、検索条件 の結果が真であるものです。

検索条件に副照会が含まれている場合は、いずれかの行に検索条件 が適用されるたびにその副照会が実行され、副照会の結果が検索条件 の適用に使用されると考えることができます。実際には、相関参照のない副照会は一度しか実行されないことがあります。各行ごとに 1 回ずつ実行する必要があるのは、相関参照がある副照会です。

CURRENT OF *cursor-name*
更新操作で使用するカーソルを識別します。カーソル名 は、 1353 ページの『DECLARE CURSOR』の説明にしたがって宣言されているカーソルを識別しなければなりません。

更新を指定した表またはビューは、このカーソルに関する SELECT ステートメントの FROM 文節でも指定されていなければなりません。また、このカーソルの結果表が読み取り専用であってはなりません。読み取り専用の結果表の説明については、 1353 ページの『DECLARE CURSOR』を参照してください。

DECLARE CURSOR ステートメントには、UPDATE ステートメントで使用される表またはビューの period-specification があってはなりません。

UPDATE ステートメントの実行時点では、カーソルはある行に位置付けられていなければなりません。その行が更新されます。

order-by-clause

offset-clause と *fetch-clause* を適用する行の順序を指定します。*offset-clause* と *fetch-clause* に基づいて更新される行セットを決定するために予測どおりの順序になるように、*order-by-clause* を指定する必要があります。

sort-key-expression

更新操作の対象となる行の順序付けに使用される値を指定する式。 1 つの *sort-key-expression* を指定すると、行はその *sort-key-expression* の値の順に並べられます。複数の *sort-key-expression* を指定すると、行は最初の *sort-key-expression* の値の順に並べられ、次に 2 番目の *sort-key-expression* の値の順に、というように順に並べられます。

sort-key-expression の結果は、DATALINK または XML であってはなりません。

ASC

sort-key-expression の値を昇順に使用します。これはデフォルトです。

DESC

sort-key-expression の値を降順に使用します。

順序付けは、 57 ページの『第 2 章 言語エレメント』で説明した比較規則にしたがって行われます。 NULL 値は、他のすべての値より高位となります。ユーザーの順序付けの指定によって完全な順序が判別できない場合は、最後に指定された *sort-key-expression* の値が重複する行は、順序が不定になります。ORDER BY を指定しなければ、更新される行は任意の順序で並べられます。

offset-clause

限定した行のサブセットをスキップして、更新の対象範囲を制限します。*offset-clause* について詳しくは、 836 ページの『offset-clause』 を参照してください。

fetch-clause

限定した行のサブセットだけに更新の対象範囲を制限します。*fetch-clause* について詳しくは、 837 ページの『fetch-clause』 を参照してください。

ISOLATION 文節

このステートメントに関して使用する分離レベルを指定します。

WITH

分離レベルを指定します。次のいずれかになります。

- *RR* 反復可能読み取り
- *RS* 読み取り固定

- *CS* カーソル固定
- *UR* 非コミット読み取り
- *NC* コミットなし

ISOLATION 文節 を指定しなかった場合は、デフォルトの分離レベルが使用されます。デフォルトの判別方法については、 859 ページの『ISOLATION 文節』を参照してください。

concurrent-access-resolution-clause
SELECT ステートメントで使用する並行アクセスの解決方法を指定します。詳しくは、 861 ページの『concurrent-access-resolution-clause』を参照してください。

UPDATE の規則

割り当て**:** 更新する値は、 113 ページの『割り当ておよび比較』で説明されている記憶域割り当て規則に従って、列に割り当てられます。

妥当性検査**:** 更新は、以下の規則に従う必要があります。従わない場合、または UPDATE ステートメントの実行中にその他のエラーが生じる場合、行は更新されません。

- 全選択*:* 行全選択 またはスカラー全選択 は、複数の行を戻してはなりません。(SQLSTATE 21000)。
- 固有制約および固有索引*:* 識別された表、または識別されたビューの基本表が 1 つ以上の固有索引または固有制約を持つ場合は、その表の更新される各行は、それらの索引および制約によって課せられる制限に適合しなければなりません (SQLSTATE 23505)。

  すべての固有性検査は、ステートメントの終わりに実際に行われます。固有索引または固有制約に関連する列の複数行 UPDATE ステートメントの場合、これはすべての行が更新された後で行われます。
- 検査制約*:* 識別された表、または識別されたビューの基本表が、 1 つ以上の検査制約を持つ場合、表で更新される各行ごとに、検査制約は真または不明でなければなりません (SQLSTATE 23513)。

  すべての検査制約は、ステートメントの終わりで必ず検証されます。複数行 UPDATE ステートメントの場合、これはすべての行が更新された後で行われます。
- ビューと *WITH CHECK OPTION:* ビューが識別されている場合は、更新された行は適用される WITH CHECK OPTION に適合しなければなりません (SQLSTATE 44000)。詳しくは、 1342 ページの『CREATE VIEW』を参照してください。

トリガー**:**識別された表または識別されたビューの基本表が更新トリガーを持つ場合、トリガーが起動されます。 トリガーが起動された結果、更新される値に応じて、他のステートメントが実行されたり、エラー条件が発生したりすることがあります。

参照保全**:** 親行の親キーの値を変更してはなりません。

更新値が NULL 値以外の外部キーを生成する場合、その外部キーは関連する親表の親キーの何らかの値に等しくなければなりません。

参照制約 (RESTRICT 削除規則を伴う参照制約以外の) は、ステートメントの終わりで実際上チェックされます。複数行 UPDATE ステートメントの場合、これはすべての行が更新された後で行われます。

XML 値: XML 列を更新する場合、新しい値は整形式 XML 文書でなければなりません。

行アクセス制御または列アクセス制御が適用されている表の行の更新**:** 行アクセス制御または列アクセス制御が適用されている表に対して UPDATE ステートメントが発行されると、有効な行の許可または列マスクに指定されている規則によって、行を更新できるかどうかが判定されます。通常、これらの規則は、ステートメントの許可 ID に基づきます。以下に、有効になっている行の許可または列マスクが UPDATE 時にどのように使用されるかについて説明します。

- 行の許可を使用して、更新される行のセットが識別されます。

  1 つの表に対して、有効な行の許可が複数定義されている場合は、有効になっている各許可の検索条件に論理 OR 演算子を適用することにより、行アクセス制御検索条件が導出されます。この行アクセス制御検索条件が表に適用されて、UPDATE ステートメントの権限 ID でアクセス可能な行が決定されます。WHERE 節を UPDATE ステートメントに指定した場合は、アクセス可能な行に対してユーザーが指定した述部が適用されて、更新される行が決定されます。WHERE 節がない場合、更新される行は、すべてのアクセス可能な行です。
- 更新される行が存在する場合、それらの行を更新できるかどうかは、以下の規則によって決まります。
  - 新しい行の値を導出するときに列を参照する場合に、その列に有効になっている列マスクが存在すると、新しい値は、マスクされた状態の値を使用して導出されます。オブジェクト表でも列アクセス制御がアクティブになっている場合、新しい値を導出するために適用される列マスクは、定数や式ではなく、列自体を戻す必要があります。列マスクを列に適用した結果が列自体にならない場合、新しい値は更新に使用できず、エラーが戻されます。
  - 行が更新可能な場合、表に BEFORE UPDATE トリガーが存在すれば、そのトリガーがアクティブ化されます。

    トリガー・アクション中に、遷移変数内の更新用の新しい値が変更されることがあります。最終的な値がトリガーから戻されると、その新しい値が更新に使用されます。
  - 更新される行は、有効になっている行の許可に準拠していなければなりません。

    更新される各行について、古い値が、UPDATE ステートメントに指定された新しい値に置き換えられます。有効になっている行の許可に準拠する行とは、更新された場合に、導出される行アクセス制御検索条件を使用して取得可能な行のことです。
  - 行が更新可能な場合、表に AFTER UPDATE トリガーが存在すれば、そのトリガーがアクティブ化されます。

マスクされたデータを、更新操作の値として使用される変数に割り当てることができます。列に更新違反チェック制約が存在しない場合、マスクされたデータで列が更新され、エラーは発行されません。

パーティション化された表のパーティション・キーの更新: パーティション化された表の行のパーティション・キーが、その行が別のパーティションに属すように更新される場合:

- 指定されるターゲット表は、表の単一のパーティションを参照する別名であってはなりません。
- 指定されるターゲット表は、ジャーナル処理の対象でなければなりません。

拡張標識変数の使用: 拡張標識変数が使用可能な場合、正の値および 0 (ゼロ) から -7 以外の標識変数値を使用しないでください。 DEFAULT および UNASSIGNED 拡張標識変数値を、それらがサポートされていないコンテキストに指定しないでください。

拡張標識変数: UPDATE ステートメントの割り当て文節 では、単一のホスト変数を参照する式を使用すると、拡張標識変数の値を割り当てることができます。拡張標識変数値 UNASSIGNED の割り当ては、列が *assignment-clause* に指定されていない場合と同様の影響を及ぼします。拡張標識変数値 DEFAULT の割り当ては、デフォルト値を持つ列、または GENERATED ALWAYS として定義されていて新しい値が生成される列 (行変更タイム・スタンプ列など) に対してのみ使用できます。

更新可能でないターゲット列は、GENERATED ALWAYS として定義されている生成列でない限り、拡張標識変数値 UNASSIGNED が割り当てられなければなりません。ターゲット列が GENERATED ALWAYS として定義される生成列である場合は、拡張標識変数値 DEFAULT または UNASSIGNED をこれに割り当てる必要があります。

UPDATE ステートメントでは、拡張標識変数値 UNASSIGNED をすべてのターゲット列に割り当ててはなりません。

拡張標識変数および更新トリガー: ターゲット列に拡張標識変数値 UNASSIGNED が割り当てられている場合、その列は更新されたとは見なされません。

拡張標識変数とエラー検査の据え置き: 拡張標識変数が使用可能な場合、非更新可能列の更新を認知するためにステートメント作成中に行われるはずになっている妥当性検査は、そのステートメントが実行されるまで据え置かれます。

システム期間テンポラル表に関する考慮事項: システム期間テンポラル表の行を更新すると、データベース・マネージャーによって、行開始列およびトランザクション開始 ID 列の値が以下のように更新されます。

- 行開始列に割り当てられる値は、次のいずれかの場合に時刻機構を読み取ることによって生成されます。(1) トランザクションの中で、表に含まれる行開始列またはトランザクション開始 ID 列に値を割り当てる必要があるようなデータ変更ステートメントを最初に実行するとき。(2) システム期間テンポラル表に含まれる行を削除するとき。行開始列の値は、トランザクション全体にわたり固有になるようにデータベース・マネージャーによって生成されます。関連した履歴表に挿入される行の終了タイム・スタンプ値が開始タイム・スタンプ値より大きくなるように、タイム・スタンプ値が調整される可能性があります。これは、競合す

るトランザクションがシステム期間テンポラル表の同じ行を更新しているときに行われる場合があります。このタイム・スタンプ値の調整を行うには、SYSTIME_PERIOD_ADJ QAQQINI オプションを *ADJUST に設定する必要があります。単一の SQL トランザクション内で複数の行が更新され、調整が必要ではない場合、行開始列の値はすべての行において同じになり、別のトランザクションでその列のために生成された値とは異なる固有の値になります。

- トランザクション開始 ID 列には、トランザクションごとに固有のタイム・スタンプ値、または NULL 値が割り当てられます。トランザクション開始 ID 列が NULL 可能で、値を調整する必要がない行開始列が表にある場合には、その列に NULL 値が割り当てられます。それ以外の場合、この値は、次のいずれかの場合に時刻機構を読み取ることによって生成されます。(1) トランザクションの中で、表に含まれる行開始列またはトランザクション開始 ID 列に値を割り当てる必要があるようなデータ変更ステートメントを最初に実行するとき。(2) システム期間テンポラル表に含まれる行を削除するとき。単一の SQL トランザクション内で複数の行が更新される場合、トランザクション開始 ID 列の値はすべての行において同じになり、別のトランザクションでその列のために生成された値とは異なる固有の値になります。

UPDATE ステートメントに、履歴行を参照する (明示的に履歴表の名前を参照するか、FROM 節の期間指定を使用することにより暗黙的に参照する) 相関副照会が含まれる検索条件がある場合、履歴行として (履歴表がある場合にはそこに) 挿入される更新行の古いバージョンは、そのステートメントにおいて以後処理される行の更新操作で可視になる可能性があります。

CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターが NULL 以外の値に設定され、SYSTIME オプションの値が YES の場合は、UPDATE ステートメントの基礎ターゲット (直接または間接) を、システム期間テンポラル表にすることはできません。

CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターが NULL 以外の値に設定され、SYSTIME オプションの値が YES の場合は、NO SQL 以外のデータ・アクセス標識を持つ外部ルーチンの呼び出しを含む WHERE 文節がビュー定義にある場合、WITH CHECK OPTION を指定して定義されたビューを、UPDATE ステートメントのターゲットにすることはできません。

履歴表に関する考慮事項**:** システム期間テンポラル表の 1 つの行が更新されるとき、その行の履歴コピーが対応する履歴表に挿入され、履歴行の終了タイム・スタンプが、データ変更操作の時刻に対応するシステム判別値の形式でキャプチャーされます。データベース・マネージャーが割り当てるこの値は、次のいずれかの場合に時刻機構を読み取ることによって生成されます。(1) トランザクションの中で、表に含まれる行開始列またはトランザクション開始 ID 列に値を割り当てる必要があるようなデータ変更ステートメントを最初に実行するとき。(2) システム期間テンポラル表に含まれる行を削除するとき。終了列の値は、トランザクション全体の履歴表で固有になるようにデータベース・マネージャーによって生成されます。履歴表に挿入される行の終了タイム・スタンプ値が開始タイム・スタンプ値より大きくなるように、タイム・スタンプ値が調整される可能性があります。これは、競合するトランザクションがシステム期間テンポラル表の同じ行を更新しているときに行われる場合があります。このタイム・スタンプ値の調整を行うには、

SYSTIME_PERIOD_ADJ QAQQINI オプションを *ADJUST に設定する必要があります。こうしないと、エラーが戻されます。

更新操作では、この調整は、システム期間テンポラル表に関連付けられた履歴表の行終了列に対応する終了列の値にのみ影響があります。この表への以降の参照ではこれらの調整を考慮に入れ、システム期間テンポラル表に関連付けられた期間の行開始列および行終了列に対応する列の値の中でトランザクション開始時刻が検索されるかどうかについて確認してください。

注

更新操作エラー**:** 更新値がいずれかの制約に違反した場合、またはその他のエラーが UPDATE ステートメントの実行中に発生し、しかも COMMIT(*NONE) が指定されていた場合は、そのステートメントの実行中に行われた変更はすべて撤回されます。ただし、エラーが発生する前に、その作業単位の中で行われていたその他の変更は撤回されません。 COMMIT(*NONE) が指定されていれば、変更が撤回されることはありません。

エラーの発生によって、カーソルの状態が予期できないものになることがあります。

更新された行数**:** UPDATE ステートメントの実行が完了した後、更新された行数は SQL 診断領域の ROW_COUNT ステートメント情報項目 (および SQLCA の SQLERRD(3)) に戻されます。 SQLCA についての説明は、 1867 ページの『付録 C. SQLCA (SQL 連絡域)』を参照してください。

ROW_COUNT についての説明は、 1489 ページの『GET DIAGNOSTICS』を参照してください。SQLCA についての説明は、 1867 ページの『付録 C. SQLCA (SQL 連絡域)』を参照してください。

ロッキング**:** 適切なロックがまだ存在していなければ、 UPDATE ステートメントが正しく実行されることにより、1 つ以上の排他ロックが獲得されます。これらのロックがコミットまたはロールバックの操作によって解放されるまで、更新された行へのアクセスは、以下に限定されます。

- その更新を行ったアプリケーション・プロセス
- 読み取り専用操作を介して、COMMIT(*NONE) または COMMIT(*CHG) を使用する別のアプリケーション・プロセス

ロックは、他のアプリケーション・プロセスがその表の操作を行うのを防止します。ロックについての詳細は、COMMIT、ROLLBACK、および LOCK TABLE ステートメント、および 32 ページの『分離レベル』の分離レベルの項を参照してください。また、「データベース・プログラミング」トピック集も参照してください。

COMMIT(*RR)、COMMIT(*ALL)、COMMIT(*CS)、または COMMIT(*CHG) が指定されている場合は、1 つの UPDATE ステートメントで、最高 500 000 000 行を更新または変更することができます。変更される行の数には、トリガーの結果として同じコミットメント定義のもとで挿入、更新、または削除される行が含まれます。

REXX: 変数は、REXX プロシージャー内の UPDATE ステートメントでは使用できません。UPDATE を使用する場合は、必ず、パラメーター・マーカーを使用する PREPARE および EXECUTE の対象として使用してください。

データ・リンク**:** DATALINK 列の URL 値を更新した場合、それは古い DATALINK 値を削除して新しい値を挿入するのと同じ結果になります。まず、古い値がいずれかのファイルにリンクしていた場合は、そのファイルへのリンクが解除されます。次に、その DATALINK 値のリンケージ属性が空であれば、指定したファイルがその列にリンクされます。

DATALINK 列のコメント値は、URL パスとして (例えば DLVALUE スカラー関数のデータ位置引数として) 空のストリングを指定するか、または新しい値に古い値と同じ値を指定することにより、ファイルに再リンクせずに更新することができます。 DATALINK 列を NULL 値で更新した場合は、既存の DATALINK 値を削除した場合と同じ結果になります。

既存の値または新しい値のいずれかのファイル・サーバーがデータベース・サーバーに登録されていない場合は、DATALINK 値を更新しようとしたときにエラーが起きることがあります。

代替構文**:** 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード NONE を NC の同義語として使用することができます。
- キーワード CHG を UR の同義語として使用することができます。
- キーワード ALL を RS の同義語として使用することができます。

例

例 *1:* EMPLOYEE 表において、従業員番号 (EMPNO) '000290' のジョブ (JOB) を 'LABORER' に変更します。

```
UPDATE EMPLOYEE
  SET JOB = 'LABORER'
  WHERE EMPNO = '000290'
```

例 *2:* PROJECT 表において、部門 (DEPTNO) 'D21' が担当しているすべてのプロジェクトについて、プロジェクトのスタッフ・レベル (PRSTAFF) を 1.5 増やします。

```
UPDATE PROJECT
  SET PRSTAFF = PRSTAFF + 1.5
  WHERE DEPTNO = 'D21'
```

例 *3:* 部門 (WORKDEPT) 'E21' の管理者以外の全従業員が一時的に配置替えになったとします。これを示すために、表 EMPLOYEE の対象従業員のジョブ (JOB) を NULL に、給与 (SALARY、BONUS、 COMM) の値をゼロに変更します。

```
UPDATE EMPLOYEE
  SET JOB=NULL, SALARY=0, BONUS=0, COMM=0
  WHERE WORKDEPT = 'E21' AND JOB <> 'MANAGER'
```

例 *4:* Java プログラムで、接続コンテキスト 'ctx' 上の表 EMPLOYEE にある行を表示した上で、要求があれば、特定の従業員のジョブ (JOB) を入力された新しいジョブ (NEWJOB) に変更します。

```
#sql iterator empIterator implements sqlj.runtime.ForUpdate
     with( updateColumns='JOB' )
     ( ... );
empIterator C1;

#sql [ctx] C1 = { SELECT * FROM EMPLOYEE };

#sql { FETCH :C1 INTO ... };
while ( !C1.endFetch() )  {
   System.out.println( ... );
                ...
   if ( condition for updating row ) {
       #sql [ctx] { UPDATE EMPLOYEE
                       SET JOB = :NEWJOB
                       WHERE CURRENT OF :C1 };
   }

   #sql { FETCH :C1 INTO ... };
}
C1.close();
```

VALUES

VALUES ステートメントは、照会の 1 つの形式です。 SQLJ アプリケーション・プログラムに組み込むことも、対話式に実行することもできます。

詳しくは、 841 ページの『全選択』を参照してください。

VALUES INTO

VALUES INTO ステートメントは、1 行以内で構成される結果表を作成し、その行の値を変数に割り当てます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができます。これは動的に準備できる実行可能ステートメントですが、割り当てられるすべての変数がグローバル変数でない場合は対話式に発行することはできません。REXX または Java では指定できません。

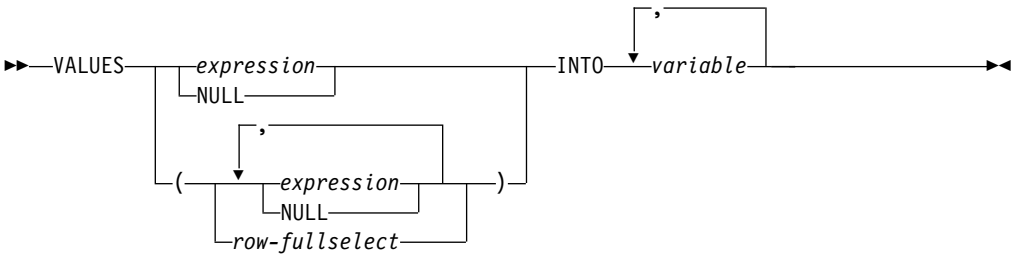
権限

行全選択 が指定されている場合、 787 ページの『第 6 章 照会』で各副選択に必要な権限についての説明を参照してください。

割り当ての右側にグローバル変数が指定される場合、ステートメントの権限 ID が保持する特権には以下の 1 つ以上が含まれていなければなりません。

- グローバル変数に対する WRITE 特権
- データベース管理者権限

構文



説明

VALUES
1 つ以上の列からなる単一行をこの後に指定します。

expression
変数の新しい値を指定します。*expression* は、 196 ページの『式』で説明したタイプの任意の式です。この式の中で列名を使用してはなりません。ホスト構造体はサポートされません。

NULL
変数の新しい値を NULL 値にすることを指定します。

row-fullselect
1 つの結果行を戻す全選択。結果列の値は、対応する各変数 に割り当てられます。全選択の結果に行が含まれない場合、NULL 値が割り当てられます。結果の中に複数の行がある場合には、エラーが戻されます。

INTO *variable*,...
1 つ以上のホスト構造体、あるいは変数を指定します。これらのホスト構造体や

変数は、その宣言の規則に従ってプログラムで宣言する必要があります。この INTO の操作形式では、ホスト構造体に対する参照は、その個々の変数それぞれに対する参照によって置き換えられます。指定した最初の値が最初の変数に割り当てられ、 2 番目の値が 2 番目の変数に割り当てられます。以下同様です。

注

変数の割り当て**:** 変数への割り当ては、それぞれ 113 ページの『割り当ておよび比較』で説明されている検索割り当て規則に従って行われます。[128] 変数の数が行の中の値の数より少ない場合、 SQL 警告 (SQLSTATE 01503) が戻されます (そして、SQLCA の SQLWARN3 フィールドに 'W' が設定されます)。結果の列の数よりも変数の数が多い場合には、警告は出されない点に注意してください。値が NULL の場合は、その値に対して標識変数が用意されている必要があります。

変数として文字変数を指定し、その変数が、結果を収容するのに十分な大きさを持っていない場合には、警告 (SQLSTATE 01004) が戻され (そして SQLCA の SQLWARN1 に 'W' が割り当てられ) ます。標識変数が用意されている場合、結果の実際の長さが、その変数に関連する標識変数に戻されることがあります。詳しくは、 170 ページの『変数』を参照してください。

割り当てでエラーが起こった場合、その値は変数に割り当てられず、それ以後の変数への値の割り当ては行われません。ただし、変数に既に割り当てられている値があれば、その値は割り当てられたままです。

変数として C の NUL で終了するホスト変数を指定し、その変数が、結果および NUL 終了文字を入れられるだけの十分な大きさを持っていない場合は、以下のようになります。

- CRTSQLCI コマンドまたは CRTSQLCPPI コマンドに *CNULRQD オプションを指定した場合 (または SET OPTION ステートメントに CNULRQD(*YES) を指定した場合)、以下のようになります。
  - 結果が切り捨てられます。
  - 最後の文字は NUL 終了文字になります。
  - SQLCA の SQLWARN1 に値'W' が割り当てられます。
- CRTSQLCI コマンドまたは CRTSQLCPPI コマンドに *NOCNULRQD オプションを指定した場合 (または SET OPTION ステートメントに CNULRQD(*NO) を指定した場合)、以下のようになります。
  - NUL 終了文字は戻されません。
  - SQLCA の SQLWARN1 に値'N' が割り当てられます。

結果列の評価に関する考慮事項**:** 算術式の結果 (ゼロによる除算やオーバーフローなど) または数値や文字の変換エラーの結果として、 VALUES INTO ステートメントの式リストにある結果列を評価する際にエラーが生じた場合、結果は NULL 値になります。他の NULL 値の場合と同様に、標識変数を用意しなければなりません。該当の変数の値は、未定義になります。ただし、この場合、標識変数は -2 の値にセットされます。ステートメントの処理は続行して、警告が戻されます。標識

128. SQL 変数または SQL パラメーターへの割り当て、および標準オプションが指定された場合は、ストレージ割り当て規則が適用されます。標準オプションについては、 xi ページの『標準への準拠』を参照してください。

変数が指定されない場合、エラーが戻されて、変数には値が割り当てられなくなります。エラーが戻されるとき、既にいくつかの値が変数に割り当てられていることがあり、それらの値は割り当てられたままになります。

日時値が戻されるとき、変数にはその値を完全に保管できるだけの長さが必要です。長さが不足する場合、切り捨てなければならない値の量に応じて、警告またはエラーが戻されます。詳しくは、 120 ページの『日時割り当て』を参照してください。

複数の割り当て**:** 複数の変数を INTO 文節で指定すると、割り当てを行う前にすべての式 が評価されます。したがって、式 内の変数 への参照は、常時、VALUES INTO ステートメント内のどの割り当てよりも前の変数 の値です。

例

例 1: CURRENT PATH 特殊レジスターの値を、ホスト変数 HV1 に割り当てます。

```
EXEC SQL VALUES CURRENT PATH
         INTO :HV1;
```

例 2: LOB ロケーター LOB1 が CLOB 値と関連していると想定します。 CLOB 値の一部を、LOB ロケーターを使用してホスト変数 DETAILS に割り当てて、CURRENT TIMESTAMP をホスト変数 TIMETRACK に割り当てます。

```
EXEC SQL VALUES (SUBSTR(:LOB1,1,35), CURRENT TIMESTAMP)
         INTO :DETAILS, :TIMETRACK;
```

WHENEVER

WHENEVER ステートメントは、指定した例外条件が発生した場合にとるべきアクションを指定します。

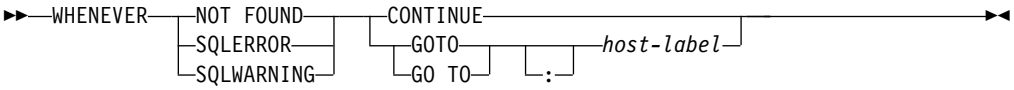
呼び出し

このステートメントは、アプリケーション・プログラムに組み込んで使用します。それ以外の使用法はありません。このステートメントは、実行可能ステートメントではありません。 Java および REXX では指定できません。REXX におけるエラー処理の説明については、「組み込み SQL プログラミング」トピック集を参照してください。

権限

権限は不要です。

構文



説明

NOT FOUND、SQLERROR、または SQLWARNING 文節は、例外条件のタイプを識別するために使用します。

NOT FOUND
SQLSTATE が '02000' に、または SQLCODE が +100 になる結果をもたらす、すべての条件を識別します。

SQLERROR
SQLCODE が負になる条件を指定します。

SQLWARNING
最初の 2 文字が '01' である SQLSTATE 値となる条件、警告状態 (SQLWARN0 の値が 'W') となる条件、または +100 以外の正の SQLCODE になる条件を識別します。

CONTINUE または GO TO 文節は、指定したタイプの例外条件が存在した場合に、次に実行するステートメントを指定するのに使用します。

CONTINUE
ソース・プログラムにある、次の順番の命令を指定します。

GOTO または **GO TO** *host-label*
ホスト・ラベル によって識別されるステートメントを実行するように指定します。ホスト・ラベルの部分には、単独のトークンを指定します (必要に応じて、トークン前にコロンを付けます)。このトークンの形式は、ホスト言語によって異なります。例えば、COBOL プログラムでは、セクション名 または修飾のない段落名 を指定します。

注

WHENEVER ステートメントの有効範囲**:** プログラム内のそれぞれの実行可能 SQL ステートメントは、どれか 1 つのタイプの暗黙または明示的な WHENEVER ステートメントの有効範囲内に含まれます。 WHENEVER ステートメントの有効範囲は、プログラム内のステートメントのリスト順によって決まり、実行順序には関連がありません。

SQL ステートメントは、ソース・プログラム内でそのステートメントの前に指定されている最後の WHENEVER ステートメント (上記の 3 つのタイプのどれか) の有効範囲内に含まれます。 SQL ステートメントの前に、いずれのタイプの WHENEVER ステートメントも指定されていなければ、その SQL ステートメントは、CONTINUE が指定されているタイプの暗黙的 WHENEVER ステートメントの有効範囲に含まれます。

SQL は、COBOL、C、および RPG でのネストされたプログラムをサポートします。しかし、SQL は通常の COBOL、C、または RPG の有効範囲規則に従いません。つまり、ネストされたプロシージャーよりも前にプログラム・ソースで指定された最後の WHENEVER ステートメントが、まだ、そのネストされたプロシージャーについては有効です。 WHENEVER ステートメントで参照されるラベルは、その内部プログラムで複写されたものである必要があります。他に、その内部プログラムで新しい WHENEVER ステートメントを指定することもできます。

例

以下のステートメントは、COBOL プログラム内に組み込むことができます。

例 *1:* ステートメントでエラーが発生した場合は、必ずラベル HANDLER に進む。

```
EXEC SQL  WHENEVER SQLERROR GOTO HANDLER  END-EXEC.
```

例 *2:* どのステートメントで警告が発生しても処理は続行する。

```
EXEC SQL  WHENEVER SQLWARNING CONTINUE  END-EXEC.
```

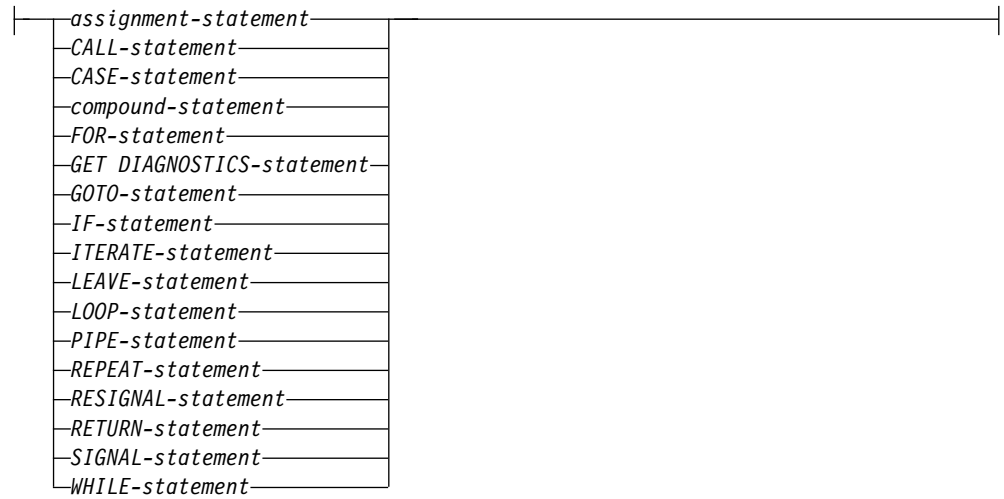
例 *3:* データを戻すべきステートメントがデータを戻さなかった場合は、ラベル ENDDATA に進む。

```
EXEC SQL  WHENEVER NOT FOUND GOTO ENDDATA  END-EXEC.
```

第 8 章 SQL 制御ステートメント

制御ステートメントは SQL ステートメントの一種で、これを使用することで、構造化プログラミング言語でプログラムを書く場合と同じような方法で SQL が使用できるようになります。 SQL 制御ステートメントは、ロジック・フローを制御し、変数の宣言と設定をし、警告および例外を処理する能力を提供します。一部の SQL 制御ステートメントには、ネストされた他の SQL ステートメントが組み込まれることもあります。

SQL-control-statement:



制御ステートメントは、SQL プロシージャー、 SQL 関数、SQL トリガー、および複合 (動的) ステートメント内でサポートされます。

SQL プロシージャーは、CREATE PROCEDURE ステートメントに SQL ルーチン本体を指定して作成します。 SQL 関数は、CREATE FUNCTION ステートメントに SQL ルーチン本体を指定して作成します。 SQL ルーチンは、SQL プロシージャーまたは SQL 関数です。 SQL トリガーは、CREATE TRIGGER ステートメントに SQL ルーチン本体を指定して作成します。複合 (動的) ステートメントは、SQL ルーチン本体を複合 (動的) ステートメントに指定することによって定義されます。

SQL プロシージャーは変更も可能です。その場合は、新しい SQL ルーチン本体を ALTER PROCEDURE ステートメントに指定します。SQL 関数も変更が可能です。この場合は、新しい SQL ルーチン本体を ALTER FUNCTION ステートメントに指定します。

SQL ルーチン本体は、単一の SQL ステートメントでなければならず、SQL 制御ステートメントであっても構いません。

SQL ルーチン本体は、プロシージャー、関数、またはトリガーの実行可能部分で、データベース・マネージャーによってプログラムまたはサービス・プログラムに変

換されます。 SQL ルーチン、トリガーまたはグローバル変数が作成されるときに、SQL は、組み込み SQL ステートメントを含む C ソース・コードが入った一時ソース・ファイル (QTEMP/QSQLSRC および QTEMP/QSQLT00000) を作成します。 SQL がこれらの一時ソース・ファイルを作成する場合、レコード長 160 が使用され、ソース・ステートメントの CCSID 値が新規ファイルの CCSID 値として設定されます。これらのソース・ファイルのいずれかがある場合、必要であればソースと同じ CCSID を持つように変更されます。これらのソース・ファイルのレコード長は 160 にする必要があります。そうでないと、予期しない結果が発生する可能性があります。ソース・ファイル・メンバーの名前は、ルーチン、トリガー、またはグローバル変数のシステム名と同じです。 DBGVIEW(*SOURCE) が指定されていた場合は、SQL はルーチンまたはトリガー用のルート・ソースを、プロシージャー、関数、またはトリガーと同じライブラリーにあるソース・ファイル QSQDSRC の中に作成します。

SQL プロシージャーまたは SQL トリガーは、CRTPGM コマンドを使用してプログラム (*PGM) オブジェクトとして作成されます。 SQL 関数は、CRTSRVPGM コマンドを使用してサービス・プログラム (*SRVPGM) オブジェクトとして作成されます。このプログラムまたはサービス・プログラムは、プロシージャー名、関数名、またはトリガー名の暗黙的または明示的な修飾子となるライブラリー内に作成されます。

プログラムまたはサービス・プログラムが作成されると、特定の制御ステートメント以外の SQL ステートメントは、そのプログラムまたはサービス・プログラム内の組み込み SQL ステートメントになります。 CALL、SIGNAL、RESIGNAL、および GET DIAGNOSTIC 制御ステートメントも、そのプログラムまたはサービス・プログラム内の組み込み SQL ステートメントになります。

指定されたプロシージャーまたは関数は、SYSROUTINES および SYSPARMS カタログ表内で登録され、プログラムに対する内部リンクが SYSROUTINES から作成されます。プロシージャーが SQL CALL ステートメントを使用して呼び出されるとき、あるいは関数が SQL ステートメント内で呼び出されるときに、そのルーチンに関連したプログラムが呼び出されます。指定された SQL トリガーは、SYSTRIGGER カタログ表内で登録されます。

SQL パラメーターおよび変数の参照

SQL パラメーターおよび SQL 変数は、式または変数が指定できる SQL プロシージャー・ステートメント内の任意の場所で参照することができます。

ホスト変数を、SQL 関数、 SQL プロシージャー、SQL トリガー、または複合 (動的) ステートメント中に指定することはできません。 SQL パラメーターは、ルーチン内の任意の場所で参照でき、そのルーチン名で修飾することができます。 SQL 変数名は、それが宣言された複合ステートメント内の任意の場所で参照することができます。その複合ステートメント内で直接または間接的にネストされたステートメントも同様です。変数が宣言されている複合ステートメントにラベルが付いていると、変数名の参照をそのラベルで修飾できます。

NOT NULL と明示的に宣言された変数を除き、 SQL パラメーターと SQL 変数はすべてNULL 可能と見なされます。 SQL ルーチン内の SQL パラメーターまたは SQL 変数の名前は、そのルーチン内で参照される表またはビュー内の列の名前と同じにすることができます。 SQL 変数の名前を、同じルーチンで宣言されている別の SQL 変数名と同じ名前にすることもできます。このことは、2 つの SQL 変数が別々の複合ステートメント で宣言される場合に生じる可能性があります。SQL 変数の宣言を含む複合ステートメント は、その変数の有効範囲を決定します。詳しくは、 1791 ページの『複合 (compound) ステートメント』を参照してください。

同じ名前は、明示的に修飾する必要があります。名前の修飾によって、名前が列、グローバル変数、SQL 変数、または SQL パラメーターのどれを指しているのかが明確に示されます。名前を修飾しない場合、または修飾されていても未確定な場合は、以下の規則に則って、名前の解決法が決まります。名前は、以下の順序で一致が検査されて解決されます。

- SQL ルーチン本体内に指定されている表またはビューが、ルーチンの作成時に存在している場合、その名前は最初に列名としてチェックされます。
- 列として見つからなければ、その名前は SQL 変数名としてチェックされます。この SQL 変数を宣言できるのは、その参照を含む *compound-statement* 内、またはその複合ステートメントをネストしている複合ステートメント内です。同一の有効範囲内に 2 つの SQL 変数があって、同一の名前を持つ場合、[129] 最も内側の複合ステートメントで宣言された SQL 変数が使用されます。
- SQL 変数名として見つからなければ、その名前は SQL パラメーター名としてチェックされます。

その名前が依然として列、SQL 変数、または SQL パラメーターとして解決できず、名前の有効範囲に現行サーバーには存在しない表またはビューが含まれている場合には、その名前は列またはグローバル変数であると想定されます。現行サーバーに表とビューのすべてが存在する場合には、その名前はグローバル変数であると想定されます。SQL_GVAR_BUILD_RULE QAQQINI オプションが *EXIST であり、そのグローバル変数が存在しない場合、エラーが発行されます。

SQL ルーチン中の SQL 変数または SQL パラメーターの名前を、特定の SQL ステートメント中で使用されている ID 名と同じ名前にすることもできます。その名

129. これは、これらの変数が別々の複合ステートメント内で宣言されている場合に生じる可能性があります。

前を修飾しない場合、次の規則によって、その名前が ID を参照するのか、SQL パラメーターまたは SQL 変数を参照するのかが記述されます。修飾された SQL パラメーターおよび SQL 変数をこれらの名前に使用することはサポートされていません。

- SET PATH ステートメントおよび SET SCHEMA ステートメントでは、その名前は SQL パラメーター名または SQL 変数名としてチェックされます。 SQL 変数名または SQL パラメーター名として見つからなければ、その名前は ID として使用されます。
- CONNECT、DISCONNECT、RELEASE、および SET CONNECTION ステートメント内では、その名前は ID として使用されます。
- CALL ステートメント内では、その名前は ID として使用されます。
- ASSOCIATE LOCATORS および DESCRIBE PROCEDURE ステートメント内では、 CREATE TRIGGER ステートメントで使用される場合は、その名前は ID として使用されます。

SQL 条件名の参照

SQL 条件名を、同じルーチンで宣言されている別の SQL 条件名と同じ名前にすることもできます。このことは、2 つの SQL 条件が別々の複合ステートメント で宣言される場合に生じる可能性があります。

SQL 条件名の宣言を含む複合ステートメント は、この条件名の有効範囲を決定します。条件名は、それが宣言される複合ステートメント内で固有でなければなりません。ただし、そのような複合ステートメント内にネストされた複合ステートメント内の宣言は例外です。条件名は、それが宣言された複合ステートメント内でのみ参照が可能です。複合ステートメント内でネストされた複合ステートメントも同様です。条件名の参照がある場合、最も内側の複合ステートメントで宣言された条件が、使用される条件です。詳しくは、 1791 ページの『複合 (compound) ステートメント』を参照してください。

SQL カーソル名の参照

SQL カーソル名を、同じルーチンで宣言されている別の SQL カーソル名と同じ名前にすることもできます。このことは、2 つの SQL カーソルが別々の複合ステートメント で宣言される場合に生じる可能性があります。 FOR ステートメントに指定されたカーソル名は、同じ複合ステートメント で宣言されている別の SQL カーソルの名前と同じにすることができます。

SQL カーソルの宣言を含む複合ステートメント は、このカーソル名の有効範囲を決定します。カーソル名は、それが宣言される複合ステートメント内で固有でなければなりません。ただし、そのような複合ステートメント内にネストされた複合ステートメント内の宣言は例外です。カーソル名は、それが宣言された複合ステートメント内でのみ参照が可能です。複合ステートメント内でネストされた複合ステートメントも同様です。カーソル名の参照がある場合、最も内側の複合ステートメントで宣言されたカーソルが使用されるカーソルです。詳しくは、 1791 ページの『複合 (compound) ステートメント』を参照してください。

SQL ラベルの参照

ラベルは、ほとんどの SQL プロシージャー・ステートメントの先頭に指定することができます。 SQL プロシージャー・ステートメントにラベルを指定する場合、同じ有効範囲内で他のラベルと異なる固有のものである必要があります。ラベルは同じ複合ステートメント内の他のどのラベルとも同じであってはならず、複合ステートメントそのものに指定されたラベルとも同じであってはなりません。また、複合ステートメントが他の複合ステートメント内にネストされている場合、ラベルは他のどの上位レベルの複合ステートメントに指定されたラベルとも同じであってはなりません。ラベルは、SQL 関数、SQL プロシージャー、またはその SQL プロシージャー・ステートメントを含む SQL トリガーの名前と同じであってはなりません。

SQL プロシージャー・ステートメントにラベルを指定することにより、そのラベルが定義され、そのラベルの有効範囲が決定されます。ラベル名は、それが定義された複合ステートメント内でのみ参照が可能です。その複合ステートメント内で直接または間接的にネストされたステートメントも同様です。ラベルは GOTO、LEAVE、または ITERATE ステートメントのターゲットとして指定することができ、そのラベルをターゲットとして参照するステートメントに関する規則の対象となります。

ネストされた複合ステートメントでの名前のスコーピングの要約

SQL 変数宣言、カーソル、条件名、および条件ハンドラーのスコープを定義するために、SQL ルーチン内でネストされた複合ステートメントを使用することができます。

さらに、ネストされた複合ステートメントのコンテキスト内で、ラベルにスコープが定義されます。ただし、名前空間の規則および非固有名の参照方法は、名前のタイプによって異なります。次の表に、これらの違いを要約します。

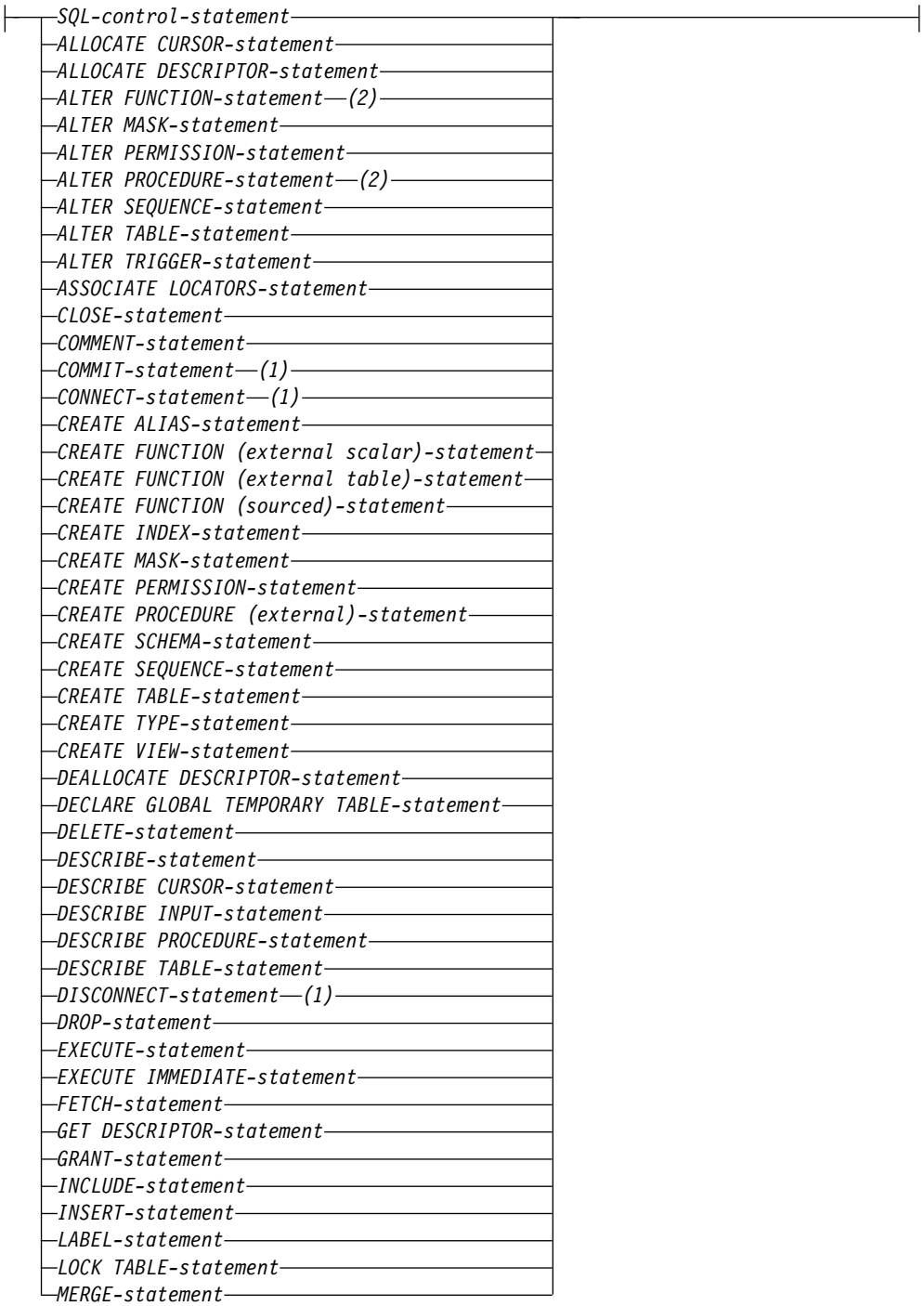
表 *118.* ネストされた複合ステートメントでの名前のスコーピングの要約

名前のタイプ	固有でなければならない場所	修飾の可・不可	参照可能な場所
SQL 変数	これが宣言される複合ステートメント。ただし、この複合ステートメント内にネストされている複合ステートメント内の宣言は除外されます。	可。この変数が宣言された複合ステートメントのラベルによる修飾が可能。	これが宣言される複合ステートメント。この複合ステートメント内にネストされている複合ステートメントをすべて含みます。複数の SQL 変数が同じ名前で定義されている場合、ラベルを使用してスコープ内で最もローカルなもの以外の特定の変数を明示的に参照することができます。
条件	これが宣言される複合ステートメント。ただし、この複合ステートメント内にネストされている複合ステートメント内の宣言は除外されます。	不可	これが宣言される複合ステートメント。この複合ステートメント内にネストされている複合ステートメントをすべて含みます。条件ハンドラーの宣言内、または SIGNAL あるいは RESIGNAL ステートメント内で使用できます。 注**:** 複数の条件が同じ名前で定義されている場合、スコープ内で最もローカルなもの以外の条件を明示的に参照する方法はありません。
カーソル	これが宣言される複合ステートメント。ただし、この複合ステートメント内にネストされている複合ステートメント内の宣言は除外されます。	不可	これが宣言される複合ステートメント。この複合ステートメント内にネストされている複合ステートメントをすべて含みます。 注**:** 複数のカーソルが同じ名前で定義されている場合、スコープ内で最もローカルなもの以外のカーソルを明示的に参照する方法はありません。ただし、そのカーソルが結果セット・カーソルとして定義されている場合 (例えば、カーソル宣言の一部として WITH RETURN 文節が指定された場合)、呼び出し側のアプリケーションはその結果セットにアクセスすることができます。
ラベル	変数が宣言された複合ステートメント。この複合ステートメント内にネストされている複合ステートメント内の宣言を含みます。	不可	これが宣言される複合ステートメント。この複合ステートメント内にネストされている複合ステートメントをすべて含みます。 ラベルは、SQL 変数の名前を修飾するため、または GOTO、LEAVE、または ITERATE ステートメントのターゲットとして使用します。

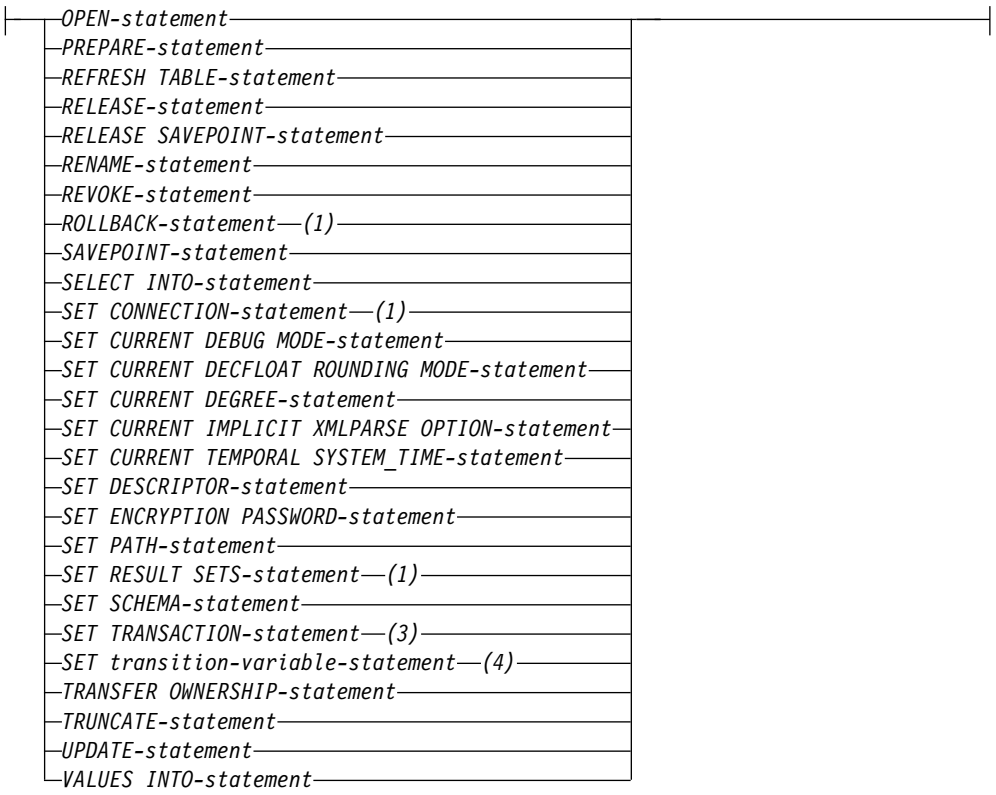
SQL プロシージャー・ステートメント

SQL 制御ステートメントでは、その SQL 制御ステートメントの内部に、複数の SQL ステートメントを指定することができます。これらのステートメントは、SQL プロシージャー・ステートメントとして定義されます。

構文:



構文 (続き):



注:

1. COMMIT、ROLLBACK、CONNECT、DISCONNECT、SET CONNECTION、および SET RESULT SETS ステートメントは、SQL プロシージャーでしか使用できません。
2. REPLACE キーワードを指定する ALTER PROCEDURE (SQL)、ALTER FUNCTION (SQL スカラー)、および ALTER FUNCTION (SQL 表) ステートメントは、*SQL-routine-body* では使用できません。
3. SET TRANSACTION ステートメントは、SQL 関数またはトリガーでしか使用できません。
4. *SET transition-variable-statement* は、トリガーでしか使用できません。*fullselect* および *VALUES-statement* もトリガーで指定できます。

注

コメント: SQL プロシージャーの本体内に、コメントを含めることができます。二重ダッシュ形式のコメント (--) に加えて、/* で始まり */ で終わるコメントも使用できます。この形式のコメントには、以下の規則が適用されます。

- 開始文字の /* は、隣接させて同一行に置く必要があります。
- 終了文字の */ は、隣接させて同一行に置く必要があります。
- コメントは、スペースを入れることができるところならば、どこからでも開始できます。
- コメントは、次の行に続けることができます。

エラーおよび警告条件の検出および処理:: SQL ステートメントが実行されると、データベース・マネージャーは、その SQL ステートメントの記述に特別な指定がない限り、そのステートメントの処理に関する情報を診断領域に保管します (SQLSTATE および SQLCODE を含む)。完了条件では、SQL ステートメントの正常完了、警告条件付きの完了、NOT FOUND 条件付きの完了のいずれかが示されます。例外条件は、SQL ステートメントが正常に実行されなかったことを示します。

例外条件、警告条件、または NOT FOUND 条件が発生した場合に実行されるように、複合ステートメント内に条件ハンドラーを定義することができます。条件ハンドラーの宣言には、条件ハンドラーを活動化するときに実行するコードが含まれます。正常完了以外の条件が *SQL* プロシージャー・ステートメント の処理内で発生すると、その条件を処理できる条件ハンドラーが有効範囲内にある場合、その条件ハンドラーがこの条件を処理するために活動化されます。条件ハンドラーの定義については、 1791 ページの『複合 (compound) ステートメント』を参照してください。条件ハンドラー内のコードは、警告条件、NOT FOUND 条件、または例外条件があるか確認し、適切なアクションをとることができます。条件ハンドラー活動化の要因となった診断領域内の条件を確認するには、その条件ハンドラーの本体の開始時に以下のいずれかの方法を使用します。

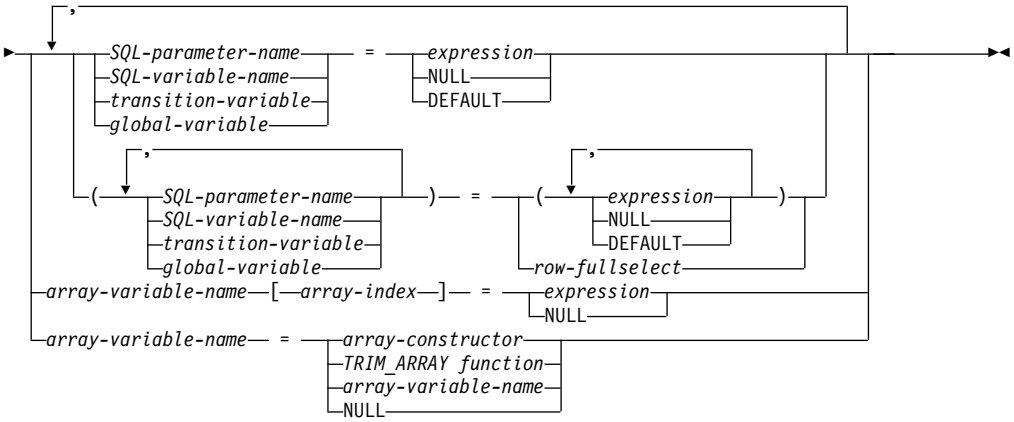
- GET DIAGNOSTICS ステートメントを発行して、条件情報を要求する。 1804 ページの『GET DIAGNOSTICS ステートメント』を参照してください。
- SQL 変数 SQLSTATE および SQLCODE をテストする。

条件が警告であり、その条件を処理するハンドラーがない場合、条件の確認を行う対象となるステートメントの直後に、上記の 2 つの方法を条件ハンドラー本体の外部で使用することもできます。条件がエラーであり、その条件を処理するハンドラーがない場合、ルーチンまたはトリガーがそのエラー条件によって終了されます。

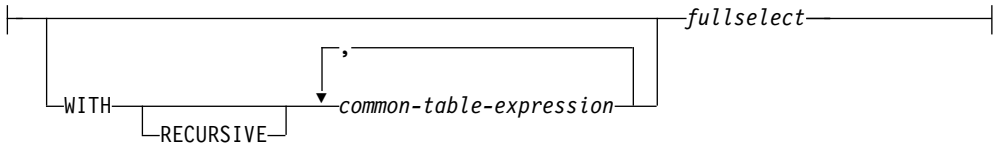
割り当てステートメント

割り当てステートメント は、SQL パラメーター、SQL 変数、または遷移変数 に値を割り当てます。

構文



行全選択:



説明

label
割り当てステートメント のラベルを指定します。このラベル名は、ルーチン名または同じ有効範囲内の別のラベルと同じものにすることはできません。詳しくは、 1777 ページの『SQL ラベルの参照』を参照してください。

SQL-parameter-name
割り当てのターゲットの SQL パラメーターを識別します。この SQL パラメーターは、CREATE PROCEDURE ステートメントまたは CREATE FUNCTION ステートメントのパラメーター宣言 に指定しておく必要があります。

SQL-variable-name
割り当てのターゲットの SQL 変数を識別します。SQL 変数は *compound-statement* でのみ宣言することができ、使用する前に宣言しておく必要があります。

transition-variable
新しい行の中のどの列を更新するかを指定します。 *transition-variable* は、トリガーの対象表の中の列を示すものでなければならず、必要に応じて、新しい値を

示す相関名により修飾することができます。OLD *transition-variable* を指定してはなりません。遷移変数 は、BEFORE トリガーでのみ変更できます。

遷移変数 は、1 つの割り当てステートメントで 1 回しか識別することができません。

値は、ストレージ割り当て規則に従って *transition-variable* に割り当てられます。詳しくは、 113 ページの『割り当ておよび比較』を参照してください。

global-variable
割り当てのターゲットとなるグローバル変数を指定します。

expression
割り当てのソースの式または値を指定します。

expression には、OLD および NEW *transition-variable* に対する参照を含めることができます。CREATE TRIGGER ステートメントに OLD 文節と NEW 文節の両方が含まれている場合は、*transition-variable* への参照を *correlation-name* によって修飾する必要があります。

NULL
NULL 値を指定します。 *transition-variable* の場合、NULL は、NULL 可能列に対してのみ指定できます。

DEFAULT
デフォルト値を使用することを指定します。 DEFAULT は、グローバル変数に対して、および SQL トリガー内で遷移変数に対してのみ指定できます。
DEFAULT は、 GENERATED 列に関連付けられた遷移変数 には使用できません。 DEFAULT が使用される場合、SET ステートメントに割り当てることができるグローバル変数は 1 つだけです。

row-fullselect
1 つの結果行を戻す全選択。結果列の値は、対応する変数またはパラメーターに割り当てられます。全選択の結果に行が含まれない場合、NULL 値が割り当てられます。結果の中に複数の行がある場合には、エラーが戻されます。

WITH *common-table-expression*
共通表式を指定します。共通表式については、 848 ページの『共通表式』を参照してください。

fullselect
1 つの結果行を戻す *fullselect*。結果列の値は、対応する各変数 に割り当てられます。 *fullselect* の結果に行が含まれない場合、NULL 値が割り当てられます。結果の中に複数の行がある場合には、エラーが戻されます。

array-variable-name
SQL 変数またはパラメーターを識別します。変数またはパラメーターは、配列タイプでなければなりません。

[*array-index***]**
配列内のどのエレメントを割り当てのターゲットにするかを指定する数値式。配列指標は、ゼロのスケールの厳密な数値タイプでなければならず、NULL にすることはできません。値は、1 から配列で定義されている最大カーディナリティーまでの値でなければなりません。

array-constructor
配列コンストラクターの値を指定します。 213 ページの『ARRAY コンストラクター』を参照してください。

TRIM_ARRAY 関数
TRIM_ARRAY スカラー関数を指定します。 662 ページの『TRIM_ARRAY』を参照してください。

注

割り当て規則**:** 割り当てステートメント内の割り当ては、 113 ページの『割り当ておよび比較』で説明されている SQL 検索割り当て規則に準拠する必要があります。[130]

`SET A[idx] = rhs` という形式 (*A* は、配列変数名、*idx* は、配列指標として使用する式、*rhs* は、配列タイプとして使用する互換タイプの式) を使用して割り当てる場合は、以下のようにします。

1. 配列 *A* が NULL 値の場合は、*A* に空の配列を設定します。
2. *C* を配列 *A* のカーディナリティーとします。
3. *idx* が *C* 以下の場合は、*idx* で指定する位置の値が *rhs* の値で置き換えられます。
4. *idx* が *C* より大きい場合は、以下のようになります。
   a. *i* (*i* は *C* より大きく *idx* より小さい値) の位置の値が NULL 値に設定されます。
   b. *idx* の位置の値が *rhs* の値に設定されます。
   c. *A* のカーディナリティーが *idx* に設定されます。

SQL パラメーターを含む割り当て**:** IN パラメーターは、割り当てステートメントの左側または右側に指定することができます。制御が呼び出し元に戻るときには、IN パラメーターのオリジナル値が保存されています。 OUT パラメーターは、割り当てステートメント の左側または右側に指定することができます。最初の指定時に値を割り当てておかなかった場合、値は NULL になります。制御が呼び出し元に戻るときに、OUT パラメーターに最後に割り当てられた値が呼び出し元に戻されます。 INOUT パラメーターの場合、このパラメーターの最初の値は呼び出し元により決定され、パラメーターに最後に割り当てられた値が呼び出し元に戻されます。

複数の **SQL** パラメーターまたは **SQL** 変数の割り当て**:** 複数の変数が割り当てステートメント のターゲットとして指定されている場合、割り当てを行う前に割り当てステートメント のターゲットが完全に評価されます。こうして、ターゲット式内の変数への参照は、常時、どの割り当てよりも前の変数の値になります。

配列**:** 配列または配列エレメントへの割り当ての場合、ステートメント内では 1 つの割り当てのみが許可されます。

130. SQL 変数または SQL パラメーターへの割り当て、および標準オプションが指定される場合、記憶域割り当て規則が適用されます。標準オプションについては、 xi ページの『標準への準拠』を参照してください。

特殊レジスター**:** 特殊レジスターの名前 (PATH など) に一致する ID を使用して変数を宣言した場合は、その変数を区切り文字で囲んで、特殊レジスターに対する割り当てと区別する必要があります (例えば、PATH という名前の変数を整数として宣言する場合は、SET "PATH" = 1)。

SQLSTATE および **SQLCODE** 変数の考慮事項**:** これらの変数への割り当ては禁止されていません。ただし、割り当ては診断領域に影響を与えることも、条件ハンドラーを活動化することもないので、お勧めしません。割り当てステートメント ごとに、SQLCODE および SQLSTATE がリセットされ、診断領域または SQLCA が初期化されますが、次のような割り当てステートメント については例外です。

- SQLSTATE または SQLCODE 変数を他の変数に割り当てるもの、または
- 定数値を SQLSTATE または SQLCODE 変数内に設定するもの。

例

例 *1:* SQL 変数の p_salary を 10% 増やします。

```
SET p_salary = p_salary + (p_salary * .10)
```

例 *2:* SQL 変数の p_salary を NULL 値に設定します。

```
SET p_salary = NULL
```

例 *3:* SQL 配列変数 p_phonenumbers を固定の番号の配列に設定します。

```
SET p_phonenumbers = ARRAY[9055553907, 4165554213, 4085553678]
```

例 *4:* SQL 配列変数 p_phonenumbers を、PHONENUMBER 表から取得した番号の配列に設定します。

```
SET p_phonenumbers = ARRAY
  [SELECT NUMBER FROM PHONENUMBERS
          WHERE EMPID = 624]
```

例 *5:* SQL 配列変数 p_phonenumbers の 1 番目と 10 番目のエレメントに p_mynumber を割り当てます。 1 番目の割り当ての後に、p_phonenumbers のカーディナリティーは 1 になります。2 番目の割り当ての後に、カーディナリティーは 10 になり、2 から 9 までのエレメントには、暗黙的に NULL 値が割り当てられます。

```
SET p_phonenumbers[1] = p_mynumber

SET p_phonenumbers[10] = p_mynumber
```

例 *6:* 給与列の値が 50000 を超えないようにします。新しい値が 50000 より大きい場合は、50000 に設定します。

```
CREATE TRIGGER LIMIT_SALARY
  BEFORE INSERT ON EMPLOYEE
  REFERENCING NEW AS NEW_VAR
  FOR EACH ROW MODE DB2SQL
  WHEN (NEW_VAR.SALARY > 50000)
    BEGIN ATOMIC
    SET NEW_VAR.SALARY = 50000;
  END
```

例 *7:* 職名が更新されたときに、新しい職名に基づいて給与が増額されるようにします。そして、その地位での年数を 0 に設定します。

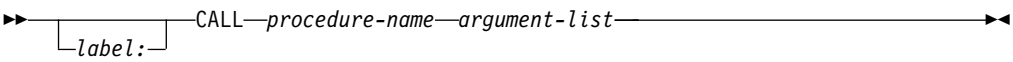
```
CREATE TRIGGER SET_SALARY
  BEFORE UPDATE OF JOB ON STAFF
  REFERENCING OLD AS OLD_VAR
              NEW AS NEW_VAR
  FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
    SET (NEW_VAR.SALARY, NEW_VAR.YEARS) =
        (OLD_VAR.SALARY * CASE NEW_VAR.JOB
            WHEN 'Sales' THEN 1.1
            WHEN 'Mgr'   THEN 1.05
            ELSE 1 END ,0);
  END
```

CALL ステートメント

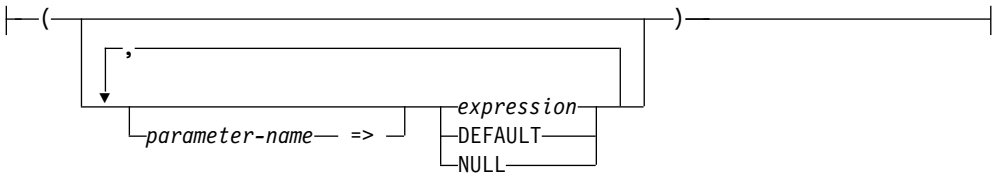
CALL ステートメントは、プロシージャーを呼び出します。 SQL 関数、SQL プロシージャー、または SQL トリガー内の CALL の構文は、他のコンテキストで CALL ステートメントとしてサポートされているもののサブセットです。

詳しくは、 1015 ページの『CALL』を参照してください。

構文



argument-list:



説明

label
CALL ステートメントのラベルを指定します。このラベル名は、ルーチン名または同じ有効範囲内の別のラベルと同じものにすることはできません。詳しくは、 1777 ページの『SQL ラベルの参照』を参照してください。

procedure-name
呼び出すプロシージャーを識別します。このプロシージャー名 は、現行サーバーに存在しているプロシージャーを識別していなければなりません。

引数リスト
パラメーターとしてプロシージャーに渡される値のリストを識別します。 *n* 番目の名前付きでない引数が、プロシージャー内の *n* 番目のパラメーターに対応します。

CREATE PROCEDURE を使用して OUT として定義する各パラメーターは、*SQL-variable-name* または *SQL-parameter-name* として指定する必要があります。OUT パラメーターにはデフォルトを指定できません。INOUT パラメーターにデフォルトが使用される場合、そのデフォルト式は、プロシージャーへの入力用のパラメーターを初期化するために使用されます。プロシージャーの終了時には、このパラメーターに対して値は何も戻されません。

プロシージャーが呼び出されるときには、デフォルト値を持つよう定義されていないすべてのパラメーターに対して引数が指定される必要があります。名前付き構文を使用して、ある引数がパラメーターに代入される場合、それに続く引数もすべて名前付き構文を使用して代入される必要があります 。

日付、時刻、またはタイム・スタンプの特殊レジスター値への引数リスト内での参照では、デフォルト式に 1 つのクロック読み取りが使用され、明示的引数内での参照には別のクロック読み取りが使用されます。

指定された引数の数は、指定されたプロシージャー名 を持つ現行のサーバーで定義されたパラメーター数と同じでなければなりません。

アプリケーション・リクエスターは、変数であるパラメーターをすべて INOUT パラメーターと見なします。変数以外のパラメーターは、すべて入力パラメーターであると見なされます。パラメーターの実際の属性は、現行サーバーによって決定されます。

parameter-name
引数値が割り当てられるパラメーターの名前。この名前は、プロシージャーに定義されているパラメーター名と一致しなければなりません。名前付き引数は、引数リスト内に指定される順序とは無関係に、同じ名前を持つパラメーターに対応します。引数が、名前によってパラメーターに割り当てられる場合、それ以降の引数もすべて名前によって割り当てられる必要があります。

名前付き引数は、一回だけ (暗黙的または明示的に) 指定される必要があります。

名前付き引数は、CREATE PROCEDURE ステートメントを使用して定義されていないプロシージャーの呼び出しでは許可されません。

expression
集約関数または列名を含まない、 196 ページの『式』で説明されているタイプの *expression*。

DEFAULT
CREATE PROCEDURE ステートメントに定義されたデフォルトがプロシージャーへの引数として使用されることを指定します。そうでない場合は NULL が使用されます。

NULL
NULL 値をプロシージャーへの引数として指定します。

注

OUT および **INOUT** パラメーターへの引数の規則**:** 各 OUT または INOUT パラメーターは、 SQL パラメーターまたは SQL 変数として指定する必要があります。

特殊レジスター**:** プロシージャー内の特殊レジスターの初期値は、そのプロシージャーの呼び出し元から継承されます。プロシージャー内で特殊レジスターに割り当てられた値は、その SQL プロシージャー全体で使用され、そのプロシージャーから呼び出される後続のすべてのプロシージャーで継承されます。プロシージャーがその呼び出し元に戻るときは、特殊レジスターは呼び出し元のオリジナルの値に復元されます。

関連情報**:** 詳しくは、 1015 ページの『CALL』を参照してください。

例

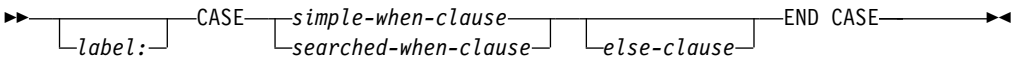
プロシージャー *proc1* を呼び出し、SQL 変数をパラメーターとして渡します。

```
CALL proc1(v_empno, v_salary)
```

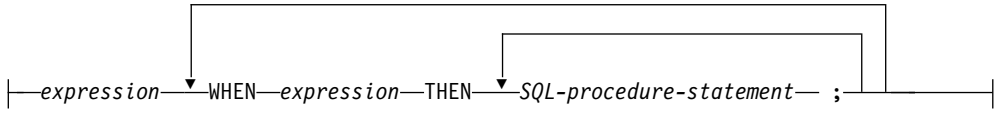
ケース (CASE) ステートメント

CASE ステートメントは、複数の条件に基づいて実行パスを選択します。

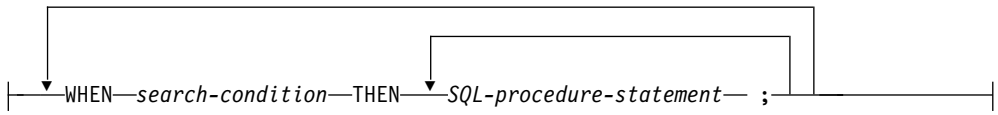
構文



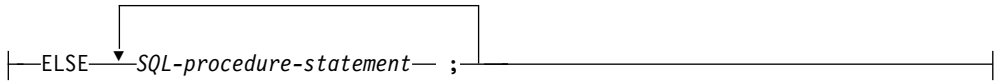
simple-when-clause:



searched-when-clause:



else-clause:



説明

label

CASE ステートメントのラベルを指定します。このラベル名は、ルーチン名または同じ有効範囲内の別のラベルと同じものにすることはできません。詳しくは、 1777 ページの『SQL ラベルの参照』を参照してください。

単純 *WHEN* 文節

最初の WHEN キーワードの前の式 の値が、WHEN キーワードの後のそれぞれの式 の値と等しいかどうかテストされます。この比較が真の場合、関連の THEN 文節内のステートメントが実行され、CASE ステートメントの処理は終了します。比較の結果が不明または偽であれば、処理は次の比較から続けられます。結果がどの比較でも一致せず、ELSE 文節が存在する場合、ELSE 文節内のステートメントが実行されます。

検索 *WHEN* 文節

WHEN キーワードの後にある検索条件 が評価されます。それが真と評価された場合、関連の THEN 文節内のステートメントが実行され、CASE ステートメントの処理は終了します。評価の結果が偽、または不明であれば、次の検索条件 が評価されます。どの *search-condition* も真と評価されず、ELSE 文節が存在する場合、ELSE 文節内のステートメントが実行されます。

else-clause
単純 *WHEN* 文節 または検索 *WHEN* 文節 に指定された条件がいずれも真でない場合は、*ELSE* 文節 内のステートメントが実行されます。

WHEN の中で指定された条件がいずれも真でない場合に ELSE 文節が指定されていなければ、実行時にエラーが出され、CASE ステートメントの実行は終了します (SQLSTATE 20000)。

SQL-procedure-statement
実行するステートメントを指定します。 1779 ページの『SQL プロシージャー・ステートメント』を参照してください。

注

CASE ステートメントのネスト **:** 単純 *WHEN* 文節 を使用する CASE ステートメントは、最大 3 レベルまでネストすることができます。検索 *WHEN* 文節 を使用する CASE ステートメントでは、ネスト・レベル数に制限はありません。

SQLSTATE および **SQLCODE** 変数の考慮事項**:** CASE ステートメント内の最初の *SQL-procedure-statement* が実行されると、SQLSTATE および SQLCODE SQL 変数は、その CASE ステートメントの *expressions* または *search-conditions* を評価した結果を反映します。CASE ステートメントに ELSE 文節を組み込まなければ、どの *search-conditions* の評価も真にならない場合に、*expression* からエラーが返されます。

例

例 *1:*SQL 変数 v_workdept の値に応じて、表 DEPARTMENT 内の列 DEPTNAME を該当の名前で更新します。

次の例は、単純 *WHEN* 文節 の構文を使用してこれを行う方法を示しています。

```
CASE v_workdept
  WHEN 'A00'
    THEN UPDATE department SET
                deptname = 'DATA ACCESS 1';
  WHEN 'B01'
    THEN UPDATE department SET
                deptname = 'DATA ACCESS 2';
    ELSE UPDATE department SET
                deptname = 'DATA ACCESS 3';
END CASE
```

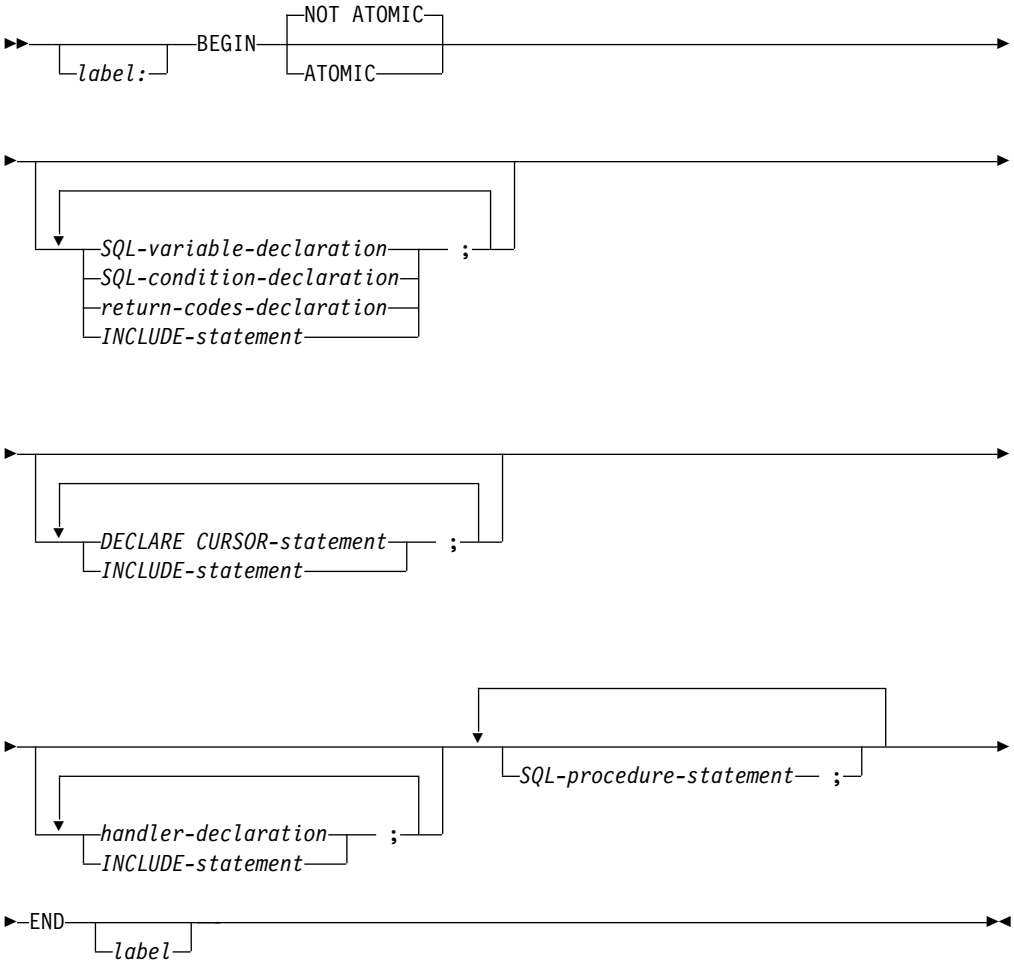
例 *2:* 次の例は、検索 *WHEN* 文節 の構文を使用してこれを行う方法を示しています。

```
CASE
  WHEN v_workdept = 'A00'
    THEN UPDATE department SET
                deptname = 'DATA ACCESS 1';
  WHEN v_workdept = 'B01'
    THEN UPDATE department SET
                deptname = 'DATA ACCESS 2';
    ELSE UPDATE department SET
                deptname = 'DATA ACCESS 3';
END CASE
```

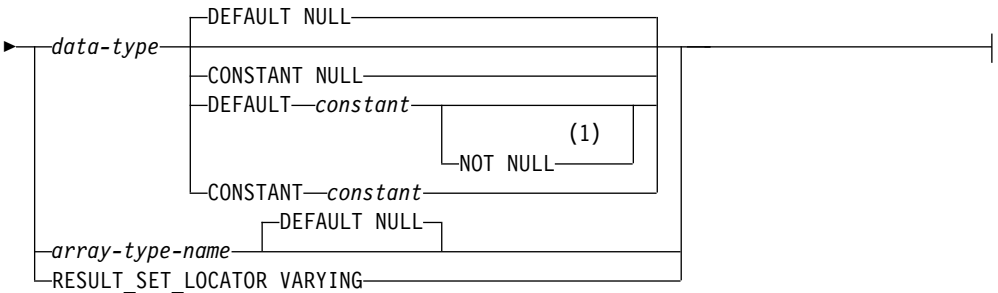
複合 (compound) ステートメント

複合ステートメントは、他のステートメントを 1 つの SQL プロシージャーの中にグループとしてまとめます。複合ステートメントによって、SQL 変数、カーソル、および条件ハンドラーを宣言することができます。

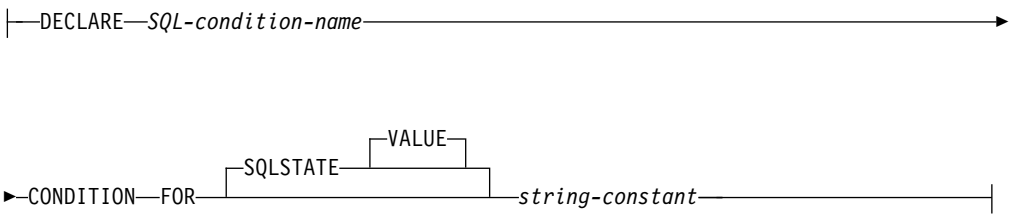
構文



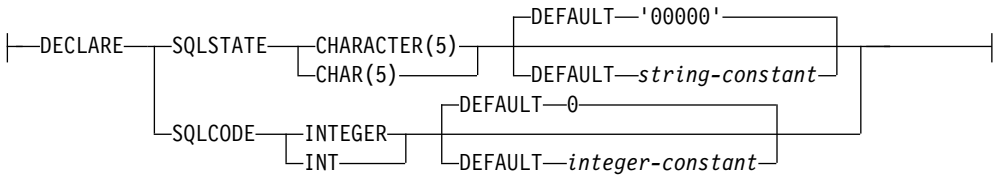
SQL-variable-declaration:



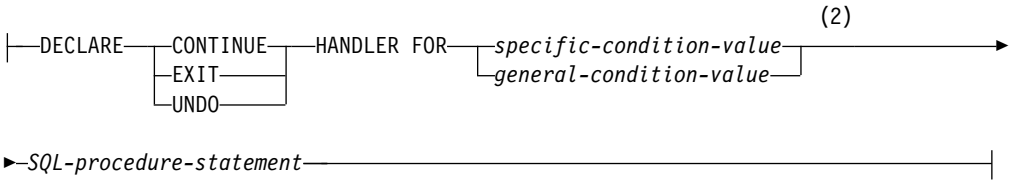
SQL-condition-declaration:



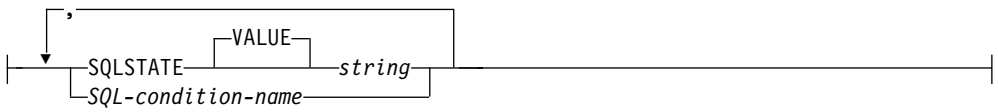
return-codes-declaration:



handler-declaration:



specific-condition-value:



general-condition-value:

data-type:

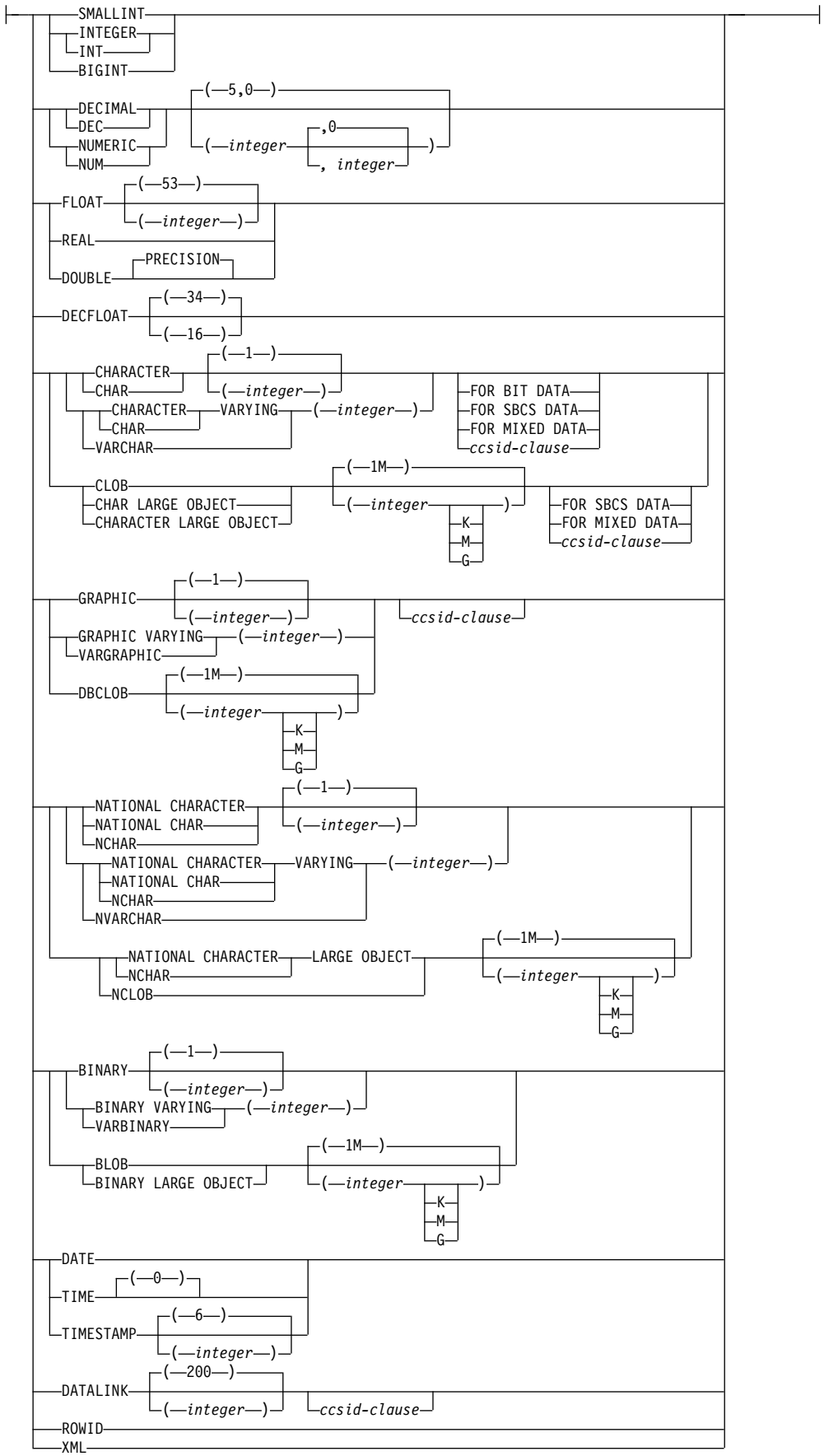
注:

1 DEFAULT 文節と NOT NULL 文節は、どちらの順序で指定しても構いません。

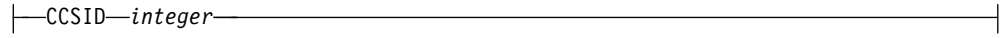
2 特定条件値 と一般条件値 を同一のハンドラー宣言に同時に指定することはできません。

3 同じ文節を複数回指定することはできません。

built-in-type:


SMALLINT
INTEGER
INT
BIGINT
DECIMAL
DEC
NUMERIC
NUM
(5,0)
(integer)
,0
, integer
FLOAT
(53)
(integer)
REAL
DOUBLE
PRECISION
DECFLOAT
(34)
(16)
CHARACTER
CHAR
(1)
(integer)
CHARACTER VARYING
CHAR VARYING
VARCHAR
FOR BIT DATA
FOR SBCS DATA
FOR MIXED DATA
ccsid-clause
CLOB
CHAR LARGE OBJECT
CHARACTER LARGE OBJECT
(1M)
K
M
G
GRAPHIC
GRAPHIC VARYING
VARGRAPHIC
DBCLOB
NATIONAL CHARACTER
NATIONAL CHAR
NCHAR
NATIONAL CHARACTER VARYING
NVARCHAR
NATIONAL CHARACTER LARGE OBJECT
NCLOB
BINARY
BINARY VARYING
VARBINARY
BLOB
BINARY LARGE OBJECT
DATE
TIME
(0)
TIMESTAMP
(6)
DATALINK
(200)
ROWID
XML

ccsid-clause:



説明

label
複合ステートメント のラベルを指定します。終了ラベルを指定する場合、開始ラベルと同じにしなければなりません。このラベル名は、ルーチン名または同じ有効範囲内の別のラベルと同じものにすることはできません。詳しくは、 1777 ページの『SQL ラベルの参照』を参照してください。

ATOMIC
ATOMIC は、複合ステートメント 内の処理されない例外条件により複合ステートメント がロールバックされることを示します。 ATOMIC を指定した場合は、複合ステートメント内で COMMIT または ROLLBACK ステートメントを指定することはできません (ROLLBACK TO SAVEPOINT を指定できます)。

NOT ATOMIC
NOT ATOMIC は、複合ステートメント 内で処理されない例外条件がある場合に複合ステートメント をロールバックしないことを示します。NOT ATOMIC は、SQL トリガーの最外部の複合ステートメント内に指定されている場合は、ATOMIC として処理されます。

ATOMIC が SQL プロシージャーまたは関数で指定されている場合、当該ルーチンのデータ種別は MODIFIES SQL DATA でなければなりません。

SQL-variable-declaration
複合ステートメント に対してローカルな変数を宣言します。

SQL-variable-name
ローカル SQL 変数の名前を定義します。データベース・マネージャーは、区切り文字のない SQL 変数名はすべて大文字に変換します。同じ複合ステートメント の内部にある別の SQL 変数と同じ名前を指定しないでください。ただし、その複合ステートメント の内部でネストされている複合ステートメント 内の宣言は例外です。列名やパラメーター名と同じ名前を SQL 変数に付けることもしないでください。同一ステートメント内に同名の列が複数個ある場合に、SQL 変数名がどのように解決されるかについては、1773 ページの『SQL パラメーターおよび変数の参照』を参照してください。変数名は 'SQL' で始まっていてはなりません。

data-type
変数のデータ・タイプを指定します。データ・タイプの説明については、1238 ページの『CREATE TABLE』を参照してください。

データ・タイプ がグラフィック・ストリング・データ・タイプの場合は、UTF-16 または UCS-2 データを指す CCSID 1200 または 13488 を指定してください。 CCSID が指定されていない場合は、グラフィック・ストリング変数の CCSID は、そのジョブに関連付けられている DBCS CCSID です。

配列タイプ名
この SQL 変数は、CREATE TYPE (配列) ステートメントで定義する配列である、ということを指定します。

DEFAULT *constant* または **NULL**
SQL 変数のデフォルト値を定義します。指定された定数は、 113 ページの『割り当ておよび比較』で説明している割り当て規則に従って、その変数に割り当てることができる値を表している必要があります。変数が初期化されるのは、その変数が宣言されている複合ステートメント が入力されるときです。デフォルト値の指定がない場合は、SQL 変数は NULL に初期化されます。 XML タイプの SQL 変数でデフォルト値を指定することはできません。配列タイプ の SQL 変数は、常に NULL に初期化されます。

NOT NULL
SQL 変数に NULL 値が入るのを防ぎます。NOT NULL を指定しないことは、その変数がヌルであってもよいことを意味します。XML タイプの SQL 変数で NOT NULL を指定することはできません。

CONSTANT *constant* または **NULL**
SQL 変数に変更できない固定値があることを指定します。 CONSTANT を使用して定義された SQL 変数は、すべての割り当て操作のターゲットとして使用できません。指定された定数は、 113 ページの『割り当ておよび比較』で説明している割り当て規則に従って、その変数に割り当てることができる値を表している必要があります。

RESULT_SET_LOCATOR VARYING
結果セット・ロケーター変数のデータ・タイプを指定します。

SQL-condition-declaration
条件名と対応する SQLSTATE 値を宣言します。

SQL-condition-name
条件の名前を指定します。条件名は、自身の宣言がある複合ステートメントの内部で固有の名前でなければなりません (ただし、複合ステートメントの内部にネストされている複合ステートメント 内の宣言は例外です)。

FOR SQLSTATE *string-constant*
この条件に関連する SQLSTATE を指定します。 ストリング定数は、5 文字で指定しなければなりません。 SQLSTATE クラス (最初の 2 文字) は「00」であってはなりません。

return-codes-declaration
GET DIAGNOSTICS や空の複合ステートメント 以外の SQL ステートメントの実行後に、診断域の最初の条件で設定される SQLSTATE と SQLCODE という特殊な SQL 変数を宣言します。 SQLSTATE 変数および SQLCODE 変数は、SQL プロシージャー、SQL 関数、または SQL トリガーの最外部の複合ステートメント の中でのみ宣言できます。

SQLSTATE および SQLCODE 特殊変数は、GET DIAGNOSTICS 以外の直前の SQL ステートメントを処理した結果得られる SQL 戻りコードを取得する手段としてのみ使用されます。SQLSTATE および SQLCODE 値を使用しようとする場合は、即時にそれらの値を他の SQL 変数に保管して、次の SQL ステートメントの実行後に戻される SQL 戻りコードでそれらの値が置き換えられないようにしてください。SQLSTATE を処理するハンドラーが定義される場合、割

り当てがそのハンドラーの最初のステートメントである場合、割り当てステートメントを使用してその SQLSTATE (または関連した SQLCODE) 値を別の SQL 変数に保管することができます。

これらの変数に値を割り当てることは、禁止されてはいませんが、お勧めしません。これらの特殊変数への割り当ては、条件ハンドラーによって無視されます。SQLSTATE および SQLCODE 特殊変数は NULL に設定できません。

DECLARE CURSOR ステートメント
ルーチン本体でカーソルを宣言します。カーソル名は、複合ステートメント 内部で固有の名前でなければなりません。ただし、複合ステートメント の内部にネストされている複合ステートメント の宣言を除きます。

カーソル名 は、それが宣言されている複合ステートメント 内でのみ参照することができます。これには、その複合ステートメント 内にネストされている任意の複合ステートメント を含みます。

このカーソルをオープンするには OPEN ステートメントを使用し、このカーソルを使用して行を読み取るには FETCH ステートメントを使用します。カーソルが SQL プロシージャー内にあり、結果セットとして使用する場合、

- カーソルを宣言する際に WITH RETURN を指定する
- DYNAMIC RESULT SETS 文節にゼロ以外の値を指定して使用し、プロシージャーを作成する
- 複合ステートメント 内で CLOSE ステートメントを指定しない。

これらの基準に適合しないカーソルは、複合ステートメント の終わりでクローズされます。

カーソルの宣言について詳しくは、 1353 ページの『DECLARE CURSOR』を参照してください。

handler-declaration
ハンドラー、つまり複合ステートメント 内で例外条件または完了条件が発生したときに実行される *SQL* プロシージャー・ステートメント を指定します。

条件ハンドラー宣言は、同じ条件値または SQLSTATE 値を複数回参照することはできず、また、1 つの SQLSTATE 値と、それと同じ SQLSTATE 値を表す条件名を参照することもできません。 SQLSTATE 値のリストおよび詳細については、「SQL メッセージおよびコード」トピック集を参照してください。

さらに、1 つの複合ステートメントで複数の条件ハンドラーが宣言される場合、2 つの条件ハンドラー宣言が以下のものを指定することはできません。

- 同一の一般条件カテゴリー、または
- 同一の特定条件 (SQLSTATE 値として、またはそれと同一の値を表す条件名として)

条件ハンドラーは、それが宣言されている複合ステートメント (ネストされた複合ステートメントを含む) 内のハンドラー宣言 に従う *SQL* プロシージャー・ステートメント のセットに対してアクティブです。

ある条件のハンドラーは、ネストされた複合ステートメントの複数のレベルに存在することがあります。例えば、複合ステートメント *n1* に別の複合ステートメント *n2* が含まれ、それにさらに別の複合ステートメント *n3* が含まれている場合を想定してください。例外条件が *n3* 内で生じると、 *n3* 内のアクティ

ブなハンドラーが最初にその条件を処理することを許可されます。適切なハンドラーが *n3* に存在しない場合、その条件は *n2* に再通知されて、 *n2* 内のアクティブなハンドラーがその状態を処理できるようになります。適切なハンドラーが *n2* に存在しない場合、その条件は *n1* に再通知されて、 *n1* 内のアクティブなハンドラーがその状態を処理できるようになります。 *n1* 内に適切なハンドラーがない場合、その状態は未処理と見なされます。

条件ハンドラーには以下の 3 つのタイプがあります。

CONTINUE
条件ハンドラーが活動化され、正常に完了した後、例外を起こしたステートメントの後の SQL ステートメントに制御が戻されることを指定します。IF、 CASE、 FOR、 WHILE、または REPEAT の中で比較を実行しているときにエラーが発生すると、それに対応する END IF、 END CASE、 END FOR、 END WHILE、または END REPEAT の後のステートメントに制御が戻されます。

EXIT
条件ハンドラーが活動化され、正常に完了した後、条件ハンドラーを宣言した複合ステートメントの終わりに制御が戻されることを指定します。

UNDO
条件ハンドラーが活動化されると、複合ステートメント によって加えられた変更がロールバックされることを指定します。このハンドラーが正常に完了すると、複合ステートメント の終わりに制御が戻されます。 UNDO を指定する場合は、ATOMIC を指定する必要があります。

UNDO は、SQL 関数または SQL トリガーの最外部の複合ステートメントの中には指定できません。

このハンドラーが活動化される条件は以下のとおりです。

SQLSTATE *string*
特定の SQLSTATE が発生したときにハンドラーを呼び出すことを指定します。SQLSTATE クラス (最初の 2 文字) は「00」であってはなりません。

SQL-condition-name
条件名に関連した特定の SQLSTATE が発生したときにハンドラーを呼び出すことを指定します。*SQL* 条件名は、*SQL* 条件宣言 の中で既に定義されていなければなりません。

SQLEXCEPTION
例外条件が発生したときにハンドラーを呼び出すことを指定します。例外条件は、最初の 2 文字が '00'、'01'、または '02' ではない SQLSTATE 値によって表されます。

SQLWARNING
警告条件が発生したときにハンドラーを呼び出すことを指定します。警告条件は、最初の 2 文字が '01' である SQLSTATE 値によって表されます。

NOT FOUND
NOT FOUND 条件が発生したときにハンドラーを呼び出すことを指定します。 NOT FOUND 条件は、最初の 2 文字が '02' である SQLSTATE 値によって表されます。

注

ネストする複合ステートメント: 複合ステートメントはネストすることができます。ネストされた複合ステートメントを使用して、変数定義、条件名、条件ハンドラー、およびカーソルの有効範囲を複合ステートメント 内のステートメントのサブセットに指定することができます。これにより、各 SQL プロシージャー・ステートメントに対する処理を単純化できます。ネストされた複合ステートメントのサポートにより、条件ハンドラーの宣言内で複合ステートメントを使用できるようになります。

条件ハンドラー: 複合ステートメント 内の条件ハンドラーは、外部 SQL アプリケーション・プログラムで使用される WHENEVER ステートメントとほぼ同じです。例外、警告、または Not Found 条件が発生すると自動的に制御を得るように、条件ハンドラーを定義できます。条件ハンドラーの本体には、その条件ハンドラーが活動化されるときに実行されるコードが含まれています。SQL ステートメントの処理のためにデータベース・マネージャーが戻す例外、警告、または Not Found 条件の結果として、条件ハンドラーを活動化することができます。または、プロシージャー本体内で SIGNAL または RESIGNAL ステートメントが発行された結果を、活動化の条件とすることができます。

条件ハンドラーは複合ステートメント内で宣言され、その条件ハンドラーが宣言される複合ステートメント内のすべての条件ハンドラー宣言の後に続く、一連の *SQL* プロシージャー・ステートメント に対してアクティブになります。もっと具体的には、条件ハンドラー宣言 H の有効範囲は、H が表示される複合ステートメント内に含まれている条件ハンドラー宣言の後に続く *SQL* プロシージャー・ステートメント のリストです。したがって、H の有効範囲には、条件ハンドラー H の本体中に含まれるステートメントは入りません。つまり、条件ハンドラーは条件ハンドラー自体の本体中で生じる条件を処理できないことになります。同様に、同一の複合ステートメント中で H1 と H2 という 2 つの条件ハンドラーが宣言されている場合、H1 は H2 の本体中で生じる条件を処理できず、H2 は H1 の本体中で生じる条件を処理できません。

条件ハンドラーの宣言は、それを活動化する条件、条件ハンドラーのタイプ (CONTINUE、EXIT、または UNDO)、およびハンドラーのアクションを指定します。条件ハンドラーのタイプにより、ハンドラー・アクションが正常に完了した後に制御がどこに戻されるかが決まります。

条件ハンドラーの活動化: *SQL* プロシージャー・ステートメント の処理で、正常終了以外の条件が発生すると、その条件を処理できる条件ハンドラーが有効範囲内にある場合、その条件を処理するためにその条件ハンドラーが活動化されます。

複合ステートメントがネストされているルーチンでは、特定の条件を処理できる条件ハンドラーが、ネストされている複合ステートメントの複数のレベルに存在する場合があります。活動化される条件ハンドラーは、条件が検出された有効範囲の最も内側で宣言される条件ハンドラーです。そのネスト・レベルの複数の条件ハンドラーが条件を処理できる場合、活動化される条件ハンドラーは、その複合ステートメントで宣言された最も適したハンドラーです。

最も適したハンドラーとは、該当の例外条件または完了条件の SQLSTATE に最も一致度の高い複合ステートメント に定義されているハンドラーです。

例えば、最も内側の複合ステートメントが SQLSTATE 22001 に対する特定のハンドラーとともに、SQLEXCEPTION に対するハンドラーも宣言する場合、SQLSTATE 22001 が検出されるときに、SQLSTATE 22001 に対する特定ハンドラーが最も適したハンドラーです。この場合、その特定ハンドラーが活動化されます。

条件ハンドラーがアクティブ化されると、その条件ハンドラーのアクションが実行されます。ハンドラー・アクションが正常に完了するか、未処理の警告を出して完了する場合、診断域がクリアされ、条件ハンドラーのタイプ (CONTINUE、EXIT、または UNDO ハンドラー) により、どこに制御が戻されるかが決まります。さらに、ハンドラーが正常に完了するか、未処理の警告で完了する場合、SQLSTATE および SQLCODE SQL 変数がクリアされます。

ハンドラー・アクションが正常に完了しないときに、ハンドラー・アクションで検出された条件に対して適切なハンドラーが存在する場合、その条件ハンドラーが活動化されます。それ以外の場合、条件ハンドラー内で検出される条件は処理されません。

未処理条件**:** 条件が検出されるときに、その条件に対して適したハンドラーが存在しない場合、その条件は処理されません。

- 未処理条件が例外である場合、失敗したステートメントを含む SQL プロシージャー、SQL 関数、または SQL トリガーは、未処理例外条件で終了します。
- 未処理条件が警告または Not Found 条件である場合、次のステートメントに移って処理が続けられます。次の SQL ステートメントの処理により、診断域内の未処理条件に関する情報が上書きされ、未処理条件の形跡が存在しなくなることに注意してください。

ネストされた複合ステートメントと一緒に **SIGNAL** または **RESIGNAL** ステートメントを使用する場合の考慮事項: 条件ハンドラーで指定された *SQL* プロシージャー・ステートメント が、例外 SQLSTATE を持つ SIGNAL または RESIGNAL ステートメントのどちらかである場合、複合ステートメントは、指定された例外を出して終了します。これは、この条件ハンドラー、または同じ複合ステートメント内の別の条件ハンドラーが CONTINUE を指定する場合であっても起こります。これらの条件ハンドラーがこの例外の有効範囲内にないからです。複合ステートメントが別の複合ステートメント内にネストされている場合、上位の複合ステートメントにある条件ハンドラーが例外の有効範囲内にあるため、それらのハンドラーがその例外を処理する場合があります。

SQL パラメーターおよび **SQL** 変数中の **NULL** 値: SQL パラメーターまたは SQL 変数の値が NULL で、標識変数が許可されていない SQL ステートメント (CONNECT あるいは DESCRIBE など) に使用されている場合、エラーが戻されます。

オープン・カーソルに対する影響: 何らかの理由で複合ステートメント を終了した後、その複合ステートメントで宣言されるすべてのオープン・カーソルはクローズされます。ただし、結果セットを戻すように宣言される場合、または *ENDACTGRP が指定されている場合を除きます。

SQLSTATE および **SQLCODE SQL** 変数に関する考慮事項**:** 複合ステートメント自体は、SQLSTATE および SQLCODE SQL 変数に影響を与えません。ただし、複

合ステートメント内にある SQL ステートメントは、SQLSTATE 変数および SQLCODE SQL 変数に影響する可能性があります。複合ステートメントの終わりで、SQLSTATE および SQLCODE SQL 変数は、その複合ステートメント内で実行され、SQLSTATE および SQLCODE SQL 変数を変更した最後の SQL ステートメントの結果を反映します。SQLSTATE および SQLCODE 変数が複合ステートメント内で変更されなかった場合、それらの変数には、複合ステートメントが入力されたときと同じ値が入っています。

例

以下のアクションを実行する、複合ステートメントのあるプロシージャー本体を作成します。

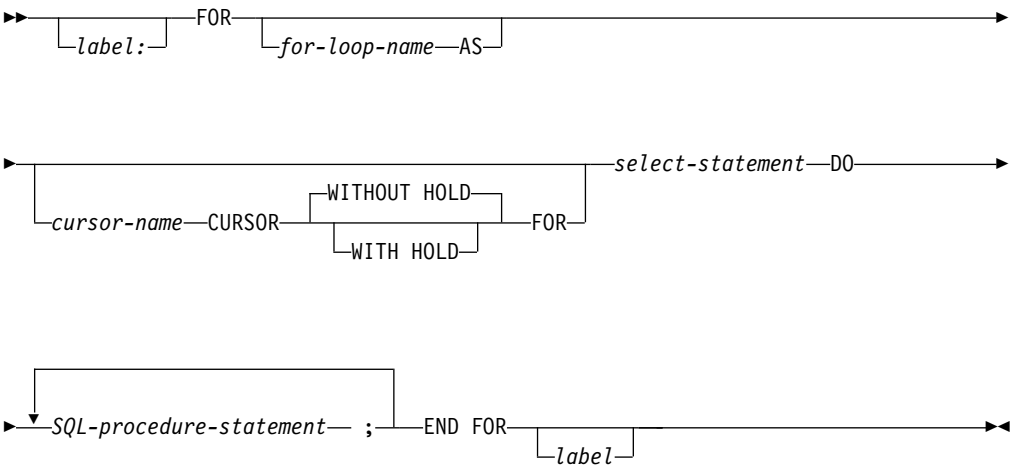
1. SQL 変数を宣言します。
2. 従業員の給与を IN パラメーターで判別される部門に戻すカーソルを宣言します。
3. 値 6666 を OUT パラメーター medianSalary に割り当てる条件 NOT FOUND (ファイルの終わり) のための EXIT ハンドラーを宣言します。
4. 指定の部門に属する従業員の数を選択して、SQL 変数 v_numRecords に入れます。
5. 従業員の 50% + 1 が検索されるまで、WHILE ループ内のカーソルから行を取り出します。
6. 給与の中央値を戻します。

```
CREATE PROCEDURE DEPT_MEDIAN
   (IN  deptNumber   SMALLINT,
    OUT medianSalary DOUBLE)
   LANGUAGE SQL
   BEGIN
    DECLARE v_numRecords INTEGER DEFAULT 1;
    DECLARE v_counter INTEGER DEFAULT 0;
    DECLARE c1 CURSOR FOR
       SELECT salary FROM staff
          WHERE DEPT = deptNumber
          ORDER BY salary;
    DECLARE EXIT HANDLER FOR NOT FOUND
       SET medianSalary = 6666;
       /* initialize OUT parameter */
       SET medianSalary = 0;
       SELECT COUNT(*) INTO v_numRecords FROM staff
          WHERE DEPT = deptNumber;
       OPEN c1;
       WHILE v_counter < (v_numRecords / 2 + 1) DO
          FETCH c1 INTO medianSalary;
          SET v_counter = v_counter + 1;
       END WHILE;
       CLOSE c1;
    END
```

FOR ステートメント

FOR ステートメントは、表の行ごとに 1 つのステートメントまたは一群のステートメントを実行します。

構文



説明

label

FOR ステートメントのラベルを指定します。終了ラベルを指定する場合、開始ラベルと同じにしなければなりません。このラベル名は、ルーチン名または同じ有効範囲内の別のラベルと同じものにすることはできません。詳しくは、 1777 ページの『SQL ラベルの参照』を参照してください。

for-loop-name

FOR ステートメントを実装するために生成する暗黙的な複合ステートメントのラベルを指定します。複合ステートメント のラベルの規則が適用されますが、FOR ステートメントの ITERATE、GOTO、LEAVE の各ステートメントでは使用できません。この *for-loop-name* を使用するのは、指定した *SELECT* ステートメント から返される列名を修飾するためです。同じ有効範囲にある他のラベルと同じ名前を指定することはできません。詳しくは、 1777 ページの『SQL ラベルの参照』を参照してください。

for-loop-name または *label* を使用して、ステートメント内の他の SQL 変数の名前を修飾できます。

for-loop-name を指定した場合は、SQL 関数、SQL プロシージャー、SQL トリガーのデバッグ時に、そのステートメントの他のすべての SQL 変数名もその *for-loop-name* で修飾する必要があります。

cursor-name

カーソルの名前を指定します。

WITH HOLD

コミット操作の結果として、カーソルがクローズされるのを防止します。WITH HOLD 文節を使用して宣言されたカーソルがコミット時点で暗黙にクローズす

るのは、そのカーソルに関連する接続がコミット操作中に終了する場合だけです。詳しくは、 1353 ページの『DECLARE CURSOR』を参照してください。

select-statement
カーソルの選択ステートメントを指定します。

選択リスト内のそれぞれの式には名前が必要です。式の名前が単純な列名ではない場合、その式の名前を指定するには AS 文節を使用しなければなりません。AS 文節を指定すると、その名前は変数に使用されますが、必ず固有の名前でなければなりません。

SQL-procedure-statement
カーソルの結果表の行ごとに実行する SQL ステートメントを指定します。この SQL ステートメントには、FOR ステートメントのカーソル名を指定する OPEN、FETCH、または CLOSE が含まれていてはなりません。

注

FOR ステートメントの規則: FOR ステートメントは、カーソルの結果表の行ごとに 1 つまたは複数のステートメントを実行します。カーソルは、選択された列および行を記述する選択リストを指定することによって定義されます。 FOR ステートメント内のステートメントは、選択されたそれぞれの行ごとに実行されます。

選択リストには固有の列名を組み込む必要があります。*SELECT* ステートメント で参照するオブジェクトは、関数、プロシージャー、トリガーの作成時に存在していなければなりません。

FOR ステートメントに指定されているカーソルは、その FOR ステートメント以外の場所で参照することはできず、OPEN、FETCH、または CLOSE ステートメントに指定できません。

ハンドラー警告: ハンドラーを使用して、カーソルをオープンする際、または FOR ステートメントでカーソルを使用して行を取り出す際に生じたエラーを処理することができます。これらのオープンまたは取り出し条件を処理するハンドラーとして CONTINUE ハンドラーを定義することは、 FOR ステートメントが永久にループする結果となる可能性があるので避けてください。

例

この例では、FOR ステートメントが、employee 表から 3 列を選択するカーソルを指定するために使用されています。選択されたすべての行について、SQL 変数 *fullname* の設定が、ラストネーム、コンマ、ファーストネーム、ブランク、ミドルネームのイニシャル、の順で行われます。 *fullname* のそれぞれの値が、表 TNAMES に挿入されます。

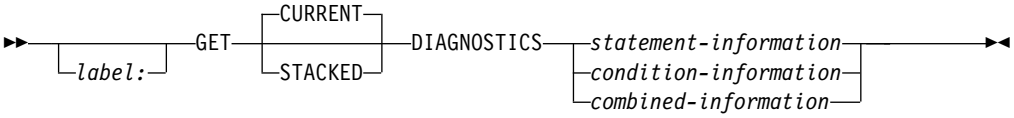
```
BEGIN
  DECLARE fullname CHAR(40);
  FOR v1 AS
      c1 CURSOR FOR
    SELECT firstnme, midinit, lastname FROM employee
     DO
       SET fullname =
          lastname || ', ' || firstnme ||' ' || midinit;
       INSERT INTO TNAMES VALUES ( fullname );
     END FOR;
END;
```

GET DIAGNOSTICS ステートメント

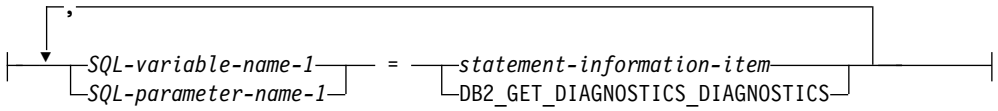
GET DIAGNOSTICS ステートメントは、直前に実行された SQL ステートメントに関する情報を取得します。 SQL 関数、SQL プロシージャー、または SQL トリガー内の GET DIAGNOSTICS の構文は、他のコンテキストで GET DIAGNOSTICS ステートメントとしてサポートされているもののサブセットです。

詳しくは、 1489 ページの『GET DIAGNOSTICS』を参照してください。

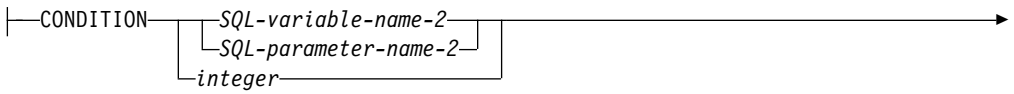
構文

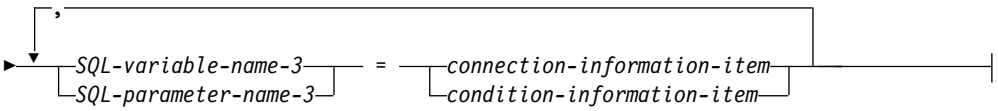


statement-information:

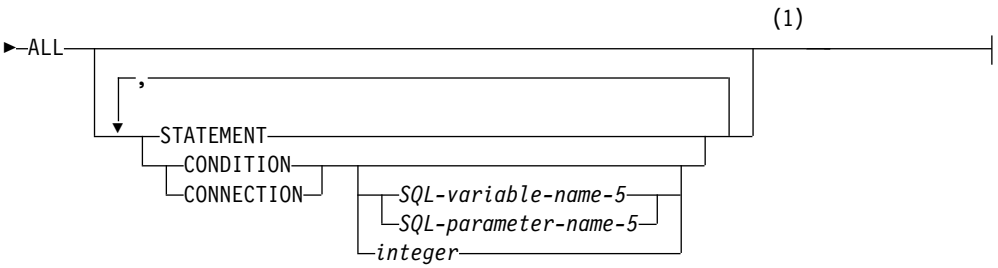


condition-information:





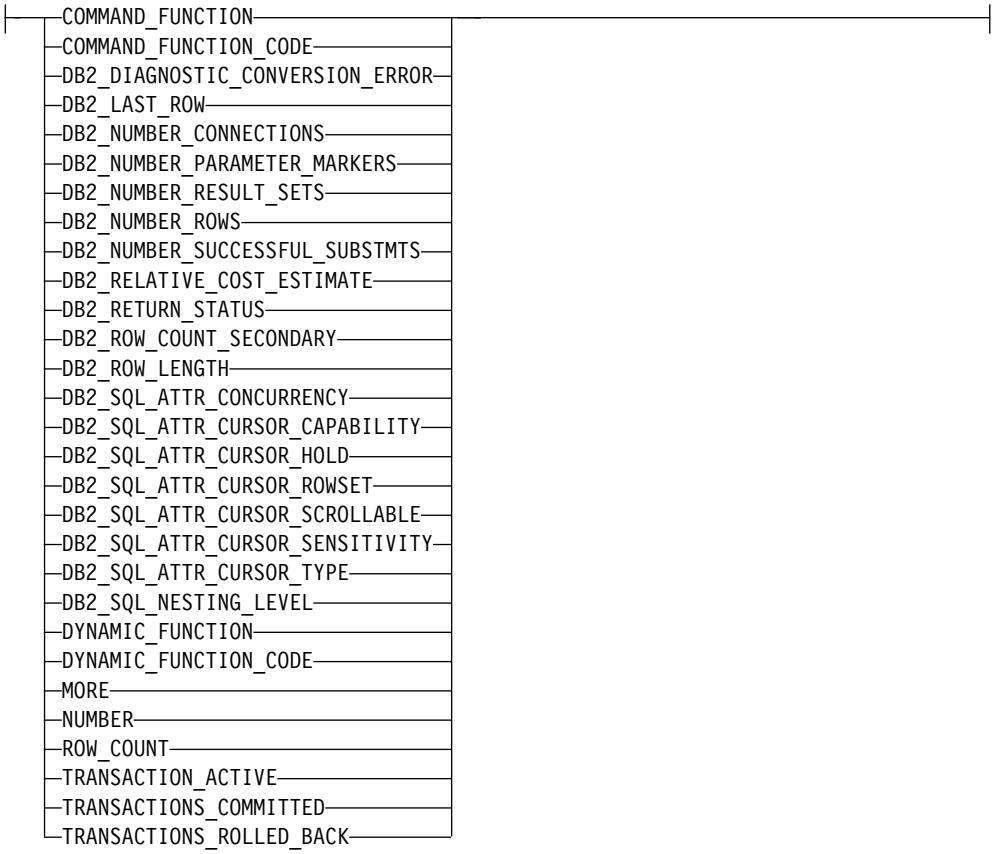
combined-information:



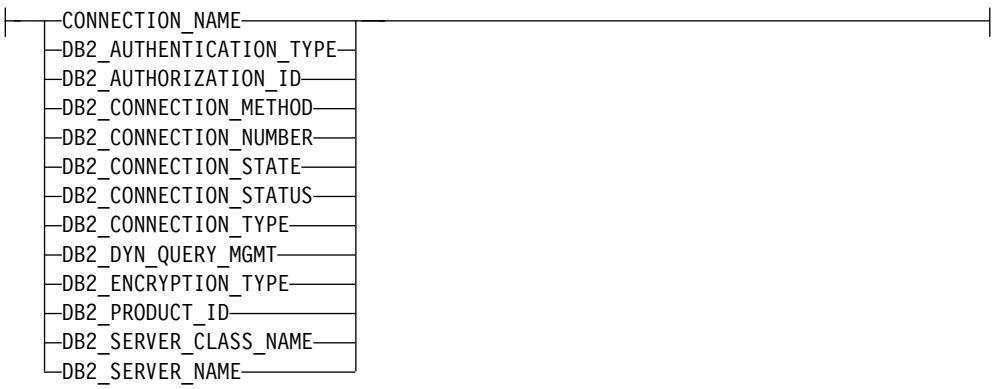
注:

1 STATEMENT は 1 回だけ指定できます。 *SQL* 変数名 *5*、 *SQL* パラメーター名 *5*、または整数 が指定されていない場合、 CONDITION および CONNECTION は 1 回だけ指定できます。

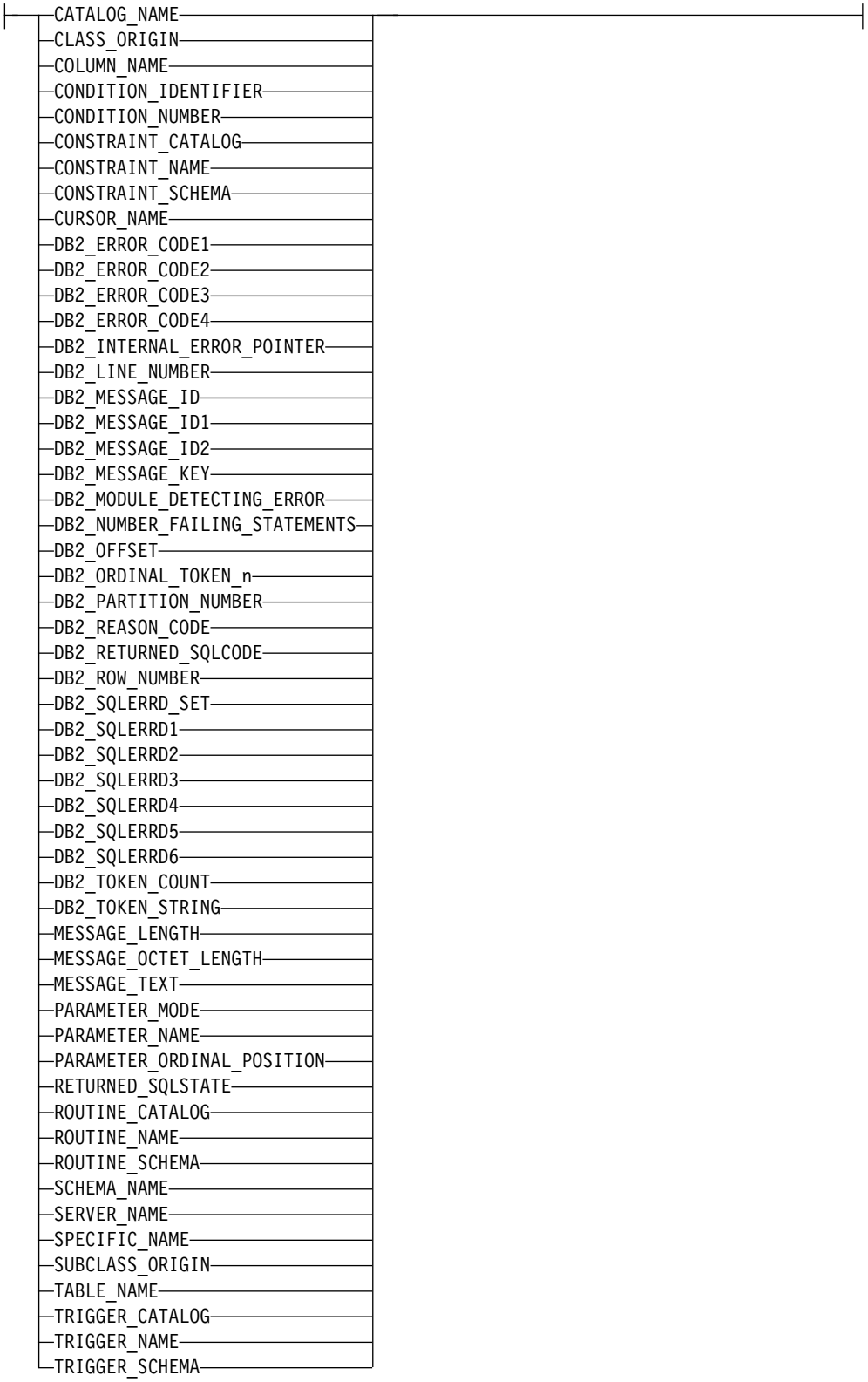
statement-information-item:



connection-information-item:



condition-information-item:



説明

label
GET DIAGNOSTICS ステートメントのラベルを指定します。終了ラベルを指定する場合、開始ラベルと同じにしなければなりません。このラベル名は、ルーチン名または同じ有効範囲内の別のラベルと同じものにすることはできません。詳しくは、 1777 ページの『SQL ラベルの参照』を参照してください。

CURRENT または **STACKED**
アクセスする診断領域を指定します。

CURRENT
最初の診断領域にアクセスするように指定します。これは、直前に実行された SQL ステートメントで、GET DIAGNOSTICS ステートメントでないものに対応します。これはデフォルトです。

STACKED
2 番目の診断領域にアクセスするように指定します。 2 番目の診断領域は、ハンドラー内だけで使用できます。これはハンドラーに入る前に実行された直前の SQL ステートメントで、 GET DIAGNOSTICS ではないものに対応します。 GET DIAGNOSTICS ステートメントがハンドラー内で最初のステートメントである場合、最初の診断領域と 2 番目の診断領域には同じ診断情報が含まれます。

statement-information
最後に実行された SQL ステートメントに関する情報を戻します。

SQL 変数名 *1* または *SQL* パラメーター名 *1*
SQL 変数および SQL パラメーターを宣言する規則に従ってプログラムに記述された、変数を指定します。 SQL 変数および SQL パラメーターのデータ・タイプは、 1508 ページの表 91 で指定の条件情報項目として指定されているデータ・タイプと互換性がなければなりません。変数には、 118 ページの『検索割り当て』で説明されている検索割り当て規則に従って、指定したステートメント情報項目の値が割り当てられます。その値が SQL 変数または SQL パラメーターに割り当てられる際に切り捨てられる場合、警告が戻されて (SQLSTATE 01004) 診断領域の GET_DIAGNOSTICS_DIAGNOSTICS 項目にこの条件の詳細が追加されて更新されます。

指定の診断項目に診断情報が含まれない場合、 SQL 変数または SQL パラメーターはそのデータ・タイプに基づいてデフォルト値に設定されます。厳密な数値診断項目は 0、VARCHAR 診断項目は空ストリング、CHAR 診断項目はブランクです。

condition-information
最後の SQL ステートメントの実行時に生じた 1 つ以上の条件に関する情報を戻します。

CONDITION *SQL-variable-name-2* または *SQL-parameter-name-2* または *integer*
情報が要求された診断を識別します。 SQL ステートメントを実行する際に生じる診断ごとに、1 つの整数が割り当てられます。値 1 は最初の診断を示し、2 は 2 番目の診断を示し、以下同様となります。値が 1 の場合、検索される診断情報は (GET DIAGNOSTICS ステートメント以外の) 直前の

SQL ステートメントの実行によって実際に戻された SQLSTATE 値によって示される条件に対応します。指定される SQL 変数または SQL パラメーターは、位取りがゼロの真数変数の宣言規則に従ってプログラム内で記述されている必要があります。指定される値は、1 より小さくてはならず、使用可能な診断の数よりも大きくてもなりません。

SQL 変数名 3 または SQL パラメーター名 3
SQL 変数または SQL パラメーターを宣言する規則に従ってプログラムに記述された、変数を指定します。 SQL 変数および SQL パラメーターのデータ・タイプは、 1508 ページの表 91 で指定の条件情報項目として指定されているデータ・タイプと互換性がなければなりません。 SQL 変数または SQL パラメーターには、 118 ページの『検索割り当て』で説明されている検索割り当て規則に従って、指定した条件情報項目の値が割り当てられます。その値が SQL 変数または SQL パラメーターに割り当てられる際に切り捨てられる場合、警告が戻されて (SQLSTATE 01004) 診断領域の GET_DIAGNOSTICS_DIAGNOSTICS 項目にこの条件の詳細が追加されて更新されます。

指定の診断項目に診断情報が含まれない場合、 SQL 変数または SQL パラメーターはそのデータ・タイプに基づいてデフォルト値に設定されます。厳密な数値診断項目は 0、VARCHAR 診断項目は空ストリング、CHAR 診断項目はブランクです。

combined-information
1 つのストリングに結合された複数の情報を戻します。

GET DIAGNOSTICS ステートメントが SQL 関数、SQL プロシージャー、またはトリガーに指定されている場合、 GET DIAGNOSTICS ステートメントはエラーを処理するハンドラーに指定された最初のステートメントでなければなりません。

警告に関する情報が必要な場合、

- ハンドラーがその警告条件に対する制御を取得する場合、GET DIAGNOSTICS ステートメントは、そのハンドラーに指定された最初のステートメントでなければなりません。
- ハンドラーがその警告条件に対する制御を取得しない場合、GET DIAGNOSTICS ステートメントは、その直前のステートメントの次に実行されるステートメントでなければなりません。

SQL 変数名 4 または SQL パラメーター名 4
SQL 変数または SQL パラメーターを宣言する規則に従ってプログラムに記述された、変数を指定します。 SQL 変数および SQL パラメーターのデータ・タイプは、VARCHAR でなければなりません。 SQL 変数または SQL パラメーターは、 118 ページの『検索割り当て』で説明されている検索割り当て規則に従って割り当てられます。*SQL 変数名 4* または *SQL パラメーター名 4* に戻される診断ストリング全体を入れるための十分の長さがない場合、ストリングは切り捨てられ、警告が戻されて (SQLSTATE 01004) 診断領域の GET_DIAGNOSTICS_DIAGNOSTICS 項目にこの条件の詳細が追加されて更新されます。

ALL
最後に実行された SQL ステートメントに設定されたすべての診断項目を、 1 つのストリングに結合するように指示します。ストリングの形式は、以下の形式による、使用可能なすべての診断情報を含むセミコロンで分離したリストです。

```
項目名=文字形式による項目値;
```

文字形式による正の数値には、先頭の正符号 (+) は含まれません。ただし、項目が RETURNED_SQLCODE のときは例外です。その場合には、先頭に正符号 (+) が追加されます。例えば、次のようになります。

```
NUMBER=1;RETURNED_SQLSTATE=02000;DB2_RETURNED_SQLCODE=+100;
```

診断情報を含む項目だけが、ストリングに含まれます。このストリング内には、DB2_GET_DIAGNOSTICS_DIAGNOSTICS および DB2_SQL_NESTING_LEVEL 項目についてのエントリーもありません。

STATEMENT
最後に実行された SQL ステートメントの診断項目を含むすべてのステートメント情報項目 診断項目を、 1 つのストリングに結合するように指示します。形式は、ALL についての上記の説明と同じです。

CONDITION
最後に実行された SQL ステートメントの診断情報を含む条件情報項目 診断項目を、1 つのストリングに結合するように指示します。*SQL* 変数名 *5*、*SQL* パラメーター名 *5*、または整数 を指定した場合、その形式は ALL オプションについての上記の説明と同じになります。 *SQL* 変数名 *5*、*SQL* パラメーター名 *5*、または整数 を指定しない場合、その形式には情報の先頭に、以下の形式による条件に対する条件番号項目が含まれます。

```
CONDITION_NUMBER=X;item-name=character-form-of-the-item-value;
```

X は、条件の番号です。例えば、次のようになります。

```
CONDITION_NUMBER=1;RETURNED_SQLSTATE=02000;RETURNED_SQLCODE=+100;
CONDITION_NUMBER=2;RETURNED_SQLSTATE=01004;
```

CONNECTION
最後に実行された SQL ステートメントの診断項目を含む接続情報項目 診断項目を、1 つのストリングに結合するように指示します。 *SQL* 変数名 *5*、*SQL* パラメーター名 *5*、または整数 を指定した場合、その形式は ALL についての上記の説明と同じになります。 *SQL* 変数名 *5*、*SQL* パラメーター名 *5*、または整数 を指定しない場合、その形式には情報の先頭に、以下の形式による条件に対する接続番号項目が含まれます。

```
CONNECTION_NUMBER=X;item-name=character-form-of-the-item-value;
```

X は、条件の番号です。例えば、次のようになります。

```
CONNECTION_NUMBER=1;CONNECTION_NAME=SVL1;DB2_PRODUCT_ID=DSN07010;
```

SQL 変数名 *5* または *SQL* パラメーター名 *5* または整数
ALL CONDITION または ALL CONNECTION 情報が要求された診

断を識別します。ここで指定する SQL 変数または SQL パラメーターは、整数の SQL 変数または SQL パラメーターの宣言に関する規則に基づいてプログラムで記述されている SQL 変数または SQL パラメーターでなければなりません。指定される値は、1 より小さくてはならず、使用可能な診断の数よりも大きくてもなりません。

ステートメント情報項目
: ステートメント情報項目 については、 1494 ページの『ステートメント情報項目』を参照してください。

接続情報項目
: 接続情報項目 については、 1499 ページの『接続情報項目』を参照してください。

条件情報項目
: 条件情報項目 については、 1501 ページの『条件情報項目』 を参照してください。

注

ステートメントの影響**:** GET DIAGNOSTICS ステートメントは、DB2_GET_DIAGNOSTICS_DIAGNOSTICS を除き、診断領域の内容を変更することはありません。

SQLCODE および **SQLSTATE SQL** 変数に関する考慮事項**:** GET DIAGNOSTICS ステートメントは、SQLSTATE および SQLCODE SQL 変数の値を変更します。

戻り値の大文字小文字の区別: 戻される診断項目に含まれる ID の値は、引用符で区切られず、大文字小文字を区別します。例えば、表の名前 "abc" は単に abc として戻されます。

項目のデータ・タイプ: 診断項目が SQL 変数または SQL パラメーターに割り当てられるとき、SQL 変数または SQL パラメーターは診断項目のデータ・タイプと互換性がなければなりません。詳しくは、 1508 ページの表 91を参照してください。

同義のキーワード**:** 以下のキーワードは、旧リリースとの互換性を維持するためにサポートされている同義語です。これらのキーワードは標準キーワードではないので、原則として使用しないようにしてください。

- キーワード EXCEPTION を CONDITION の同義語として使用することができます。
- キーワード RETURN_STATUS を DB2_RETURN_STATUS の同義語として使用することができます。

例

例 *1:* SQL プロシージャーにおいて、GET DIAGNOSTICS ステートメントを実行して、更新された行数を判別します。

```
CREATE PROCEDURE sqlprocg (IN deptnbr VARCHAR(3))
  LANGUAGE SQL
  BEGIN
    DECLARE SQLSTATE CHAR(5);
    DECLARE rcount INTEGER;
    UPDATE CORPDATA.PROJECT
```

```
    SET PRSTAFF = PRSTAFF + 1.5
    WHERE DEPTNO = deptnbr;
  GET DIAGNOSTICS rcount = ROW_COUNT;
  /* At this point, rcount contains the number of rows that were updated. */
END
```

例 *2:* SQL プロシージャー内部で、TRYIT というストアード・プロシージャーの呼び出しから戻された状況値を処理します。TRYIT で RETURN ステートメントを使用して状況値を明示的に戻すこともでき、データベース・マネージャーから状況値が暗黙的に戻されることもあります。このプロシージャーは、正常に実行されると、値ゼロを戻します。

```
CREATE PROCEDURE TESTIT ()
  LANGUAGE SQL
  A1: BEGIN
      DECLARE RETVAL INTEGER DEFAULT 0;
      ...
      CALL TRYIT
      GET DIAGNOSTICS RETVAL = RETURN_STATUS;
      IF RETVAL <> 0 THEN
         ...
         LEAVE A1;
      ELSE
         ...
      END IF;
  END A1
```

例 *3:* SQL プロシージャーで、GET DIAGNOSTICS ステートメントを実行して、エラーのメッセージ・テキストを取り出します。

```
CREATE PROCEDURE divide2 ( IN numerator INTEGER,
                           IN denominator INTEGER,
                          OUT divide_result INTEGER,
                              OUT divide_error VARCHAR(70) )
LANGUAGE SQL
  BEGIN
     DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
        GET DIAGNOSTICS EXCEPTION 1
          divide_error = MESSAGE_TEXT;
     SET divide_result = numerator / denominator;
  END;
```

GOTO ステートメント

GOTO ステートメントは、SQL 関数、SQL プロシージャー、または SQL トリガー内部のユーザー定義のラベルに分岐します。

構文

説明

ラベル *1*

GOTO ステートメントのラベルを指定します。このラベル名は、ルーチン名または同じ有効範囲内の別のラベルと同じものにすることはできません。詳しくは、 1777 ページの『SQL ラベルの参照』を参照してください。

label2

処理を継続するラベル付きステートメントを指定します。ラベル付きステートメントと GOTO ステートメントは、両方とも同じ有効範囲内に存在しなければなりません。

- FOR ステートメントの中で GOTO ステートメントを定義する場合は、同じ FOR ステートメントの中で *label2* を定義する必要があります (ただし、ネストされた FOR ステートメントまたはネストされた複合ステートメントは除きます)。
- FOR ステートメントの外で GOTO ステートメントを定義する場合は、FOR ステートメントまたはネストされた複合ステートメントの中で *label2* を定義しないでください。
- 異常事態処理ルーチンの中で GOTO ステートメントを定義する場合は、同じ処理ルーチンの中で *label2* を定義する必要があります。
- 異常事態処理ルーチンの外で GOTO ステートメントを定義する場合は、異常事態処理ルーチンの中で *label2* を定義しないでください。

ラベル *2* が GOTO ステートメントで到達可能な有効範囲内に定義されていない場合は、エラーが戻されます。

注

GOTO ステートメントの使用: GOTO ステートメントは、控えめに使用することをお勧めします。このステートメントは、SQL ステートメントの通常の処理順序を乱すので、ルーチンの読み取りや維持が難しくなります。GOTO ステートメントを使用する前に、IF ステートメントや LEAVE ステートメントなど別のステートメントを代わりに使用することで、GOTO ステートメントを使わずに済まないか検討してください。

オープン・カーソルに対する影響: GOTO ステートメントが制御を複合ステートメントから移動すると、その GOTO ステートメントを含んでいる該当の複合ステートメントで宣言されているすべてのオープン・カーソルはクローズされます。ただし、結果セットを戻すように宣言される場合、または *ENDACTGRP が指定されている場合を除きます。

ATOMIC 複合ステートメントに対する影響: GOTO ステートメントが制御を ATOMIC 複合ステートメントから移動すると、その ATOMIC 複合ステートメントが入力されたときに暗黙的に開始されたセーブポイントが解放されます。

SQLSTATE および **SQLCODE** 変数に関する考慮事項**:** GOTO ステートメントは、SQLSTATE および SQLCODE SQL 変数に影響を与えません。 GOTO ステートメントの終わりで、SQLSTATE および SQLCODE SQL 変数は、その GOTO ステートメントの前に実行された最後のステートメントの結果を反映します。

例

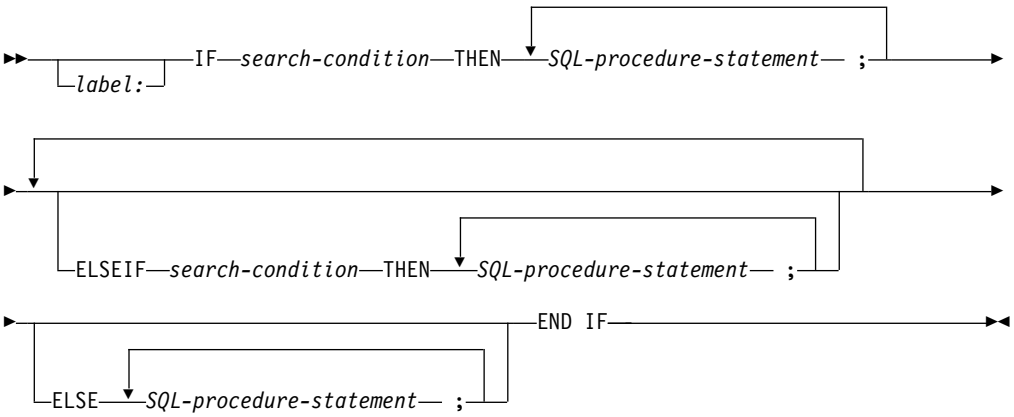
次のステートメントでは、パラメーター *rating* と *v_empno* がプロシージャーに渡されます。サービスの時間が、出力パラメーター *return_parm* で日付期間として戻されます。会社のサービスの時間が 6 カ月未満の場合、GOTO ステートメントは制御をプロシージャーの最後に移動し、*new_salary* は変更されないままになります。

```
CREATE PROCEDURE adjust_salary
  (IN v_empno CHAR(6),
   IN rating INTEGER,
   OUT return_parm DECIMAL(8,2))
  LANGUAGE SQL
  MODIFIES SQL DATA
    BEGIN
       DECLARE new_salary DECIMAL(9,2);
       DECLARE service DECIMAL(8,2);
            SELECT salary, CURRENT_DATE - hiredate
               INTO new_salary, service
               FROM employee
               WHERE empno = v_empno;
            IF service < 600
              THEN GOTO exit1;
            END IF;
            IF rating = 1
             THEN SET new_salary = new_salary + (new_salary * .10);
             ELSEIF rating = 2
             THEN SET new_salary = new_salary + (new_salary * .05);
            END IF;
            UPDATE employee
               SET salary = new_salary
               WHERE empno = v_empno;

      EXIT1: SET return_parm = service;
    END
```

IF ステートメント

IF ステートメントは、検索条件の結果に基づいて、異なるセットの SQL ステートメントを実行します。

構文



説明

label
IF ステートメントのラベルを指定します。このラベル名は、ルーチン名または同じ有効範囲内の別のラベルと同じものにすることはできません。詳しくは、1777 ページの『SQL ラベルの参照』を参照してください。

search-condition
SQL ステートメントを実行することが必要になる検索条件 を指定します。この条件が不明または偽であれば、条件が真になるかまたは ELSE 文節に到達するまで、処理は次の検索条件から続けられます。

SQL-procedure-statement
直前の *search-condition* が真になった場合に実行する SQL ステートメントを指定します。

注

SQLSTATE および SQLCODE SQL 変数に関する考慮事項: IF ステートメントの最初の *SQL procedure-statement* を実行する際は、SQLSTATE および SQLCODE SQL 変数はその IF ステートメントの検索条件 の評価結果を反映します。 IF ステートメントに ELSE 文節が含まれず、検索条件 の評価がいずれも真でなかった場合、IF ステートメントの後のステートメントの実行時に、SQLSTATE および SQLCODE SQL 変数はその IF ステートメントの検索条件の評価結果を反映します。

例

次の SQL プロシージャーは、2 つの IN パラメーター (従業員番号と従業員考課) を受け入れます。 *rating* (考課) の値に応じて、従業員表が更新され、給与および賞与の列に新しい値が入ります。

```
CREATE PROCEDURE UPDATE_SALARY_IF
   (IN employee_number CHAR(6), INOUT rating SMALLINT)
   LANGUAGE SQL
   MODIFIES SQL DATA
   BEGIN
      DECLARE not_found CONDITION FOR SQLSTATE '02000';
      DECLARE EXIT HANDLER FOR not_found
         SET rating = -1;
      IF rating = 1
         THEN UPDATE employee
         SET salary = salary * 1.10, bonus = 1000
         WHERE empno = employee_number;
      ELSEIF rating = 2
         THEN UPDATE employee
         SET salary = salary * 1.05, bonus = 500
         WHERE empno = employee_number;
      ELSE UPDATE employee
         SET salary = salary * 1.03, bonus = 0
         WHERE empno = employee_number;
      END IF;
   END
```

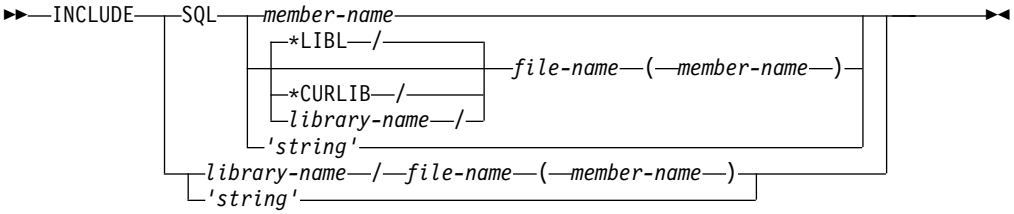
INCLUDE ステートメント

INCLUDE ステートメントは、宣言やステートメントなどのアプリケーション・コードを SQL ルーチン本体に挿入します。インクルードされるコードには、ルーチン定義に即時に追加される SQL コード、または SQL プロシージャー、SQL 関数、または SQL トリガーの作成時に追加される ILE C コードのいずれかが可能です。

権限

このステートメントの権限 ID は、メンバーを含むファイルについて、 *OBJOPR および *READ のシステム権限を持つ必要があります。

構文



説明

INCLUDE ステートメントは、ソースの中で、インクルードされるソース・ステートメントが構文的に受け入れられる場所に指定しなければなりません。 INCLUDE ステートメントのどちらの形式も、複合ステートメント の本体内の *SQL* プロシージャー・ステートメント として指定できます。 INCLUDE SQL ステートメントは、複合ステートメント の宣言およびハンドラー内でも指定可能です。 1 つのステートメントで指定できるインクルードの数に制限はありませんが、インクルードされるソースが、埋め込みのインクルードを含むことはできません。

SQL

インクルードされるソースには、*SQL* ルーチン本体 または *SQL* トリガー本体に使用できる SQL 構文のみが含まれます。これらのステートメントは、*SQL* ルーチン本体 または *SQL* トリガー本体 の一部としてインラインで展開されます。これらは、ステートメントのステートメント・テキストに直接含まれるかのように処理されます。 SQL プロシージャー、関数、またはトリガー用に保管されたステートメント・テキストは、このインクルードされたソースを含みます。 QSYS2.GENERATE_SQL プロシージャーが使用されると、生成ソースでは、INCLUDE SQL が使用されたところで、展開された SQL ステートメントが表示されます。

member-name

SQL プロシージャー、SQL 関数、または SQL トリガーの中にインクルードするメンバーを識別します。ソース・ファイルとライブラリーは、SET OPTION ステートメントの INCFILE オプションを使用して判別されます。詳しくは、 1696 ページの『SET OPTION』を参照してください。

file-name (member-name)

SQL プロシージャー、SQL 関数、または SQL トリガーの中にインクルー

ドするソース・ファイルおよびメンバーを識別します。ソース・ファイルを含むライブラリーは、以下のいずれかの方法で指定します。

***LIBL**
そのジョブのライブラリー・リストのすべてのライブラリーが検索され、見つかった最初の表が使用されます。これはデフォルトです。

***CURLIB**
ジョブ用の現行ライブラリーが検索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL が使用されます。

library-name
ライブラリーの名前を識別します。

'string'
インクルードされるソース・ストリーム・ファイルを識別します。ストリングは、標準 SQL ストリング・リテラルとして処理されます。エスケープ文字に対するソース・ストリーム・ファイル規則には従いません。接尾部は、ストリングに付加されません。

SQL を指定しない場合
インクルードされるソースは、EXEC SQL が前に付いた ILE C 言語ステートメントまたは 組み込み SQL ステートメントのみを含みます。対応する EXEC SQL INCLUDE ステートメントは、この SQL プロシージャー、SQL 関数、または SQL トリガーに生成されるプログラム・ソースの一部として生成されます。

プロシージャー、関数、またはトリガーの作成中に追加のサービス・プログラムをバインドすることがインクルードで要求される場合は、 SET OPTION ステートメントで BINDOPT オプションを使用して、他のサービス・プログラムにあるエクスポートを参照するために BNDSRVPGM パラメーターを追加することができます。詳しくは、 1696 ページの『SET OPTION』を参照してください。

library-name / file-name (member-name)
SQL プロシージャー、SQL 関数、または SQL トリガーの作成時にインクルードする ILE C 言語ステートメントを含む完全修飾ソース・ファイルおよびメンバーを指定します。

'string'
SQL プロシージャー、SQL 関数、または SQL トリガーの作成時にインクルードする ILE C 言語ステートメントを含むソース・ストリーム・ファイルの完全パスを指定します。

注

CCSID に関する考慮事項**:** SQL ステートメントの CCSID が INCLUDE ステートメントのソースの CCSID と異なる場合、INCLUDE ソースは、SQL ステートメントの CCSID に変換されます。

ソースに関する考慮事項**:** C インクルードのソースは、160 ソース列を超えてはなりません。それより先は切り捨てられます。

SQL インクルードのストリング・リテラルは、組み込んでいるソースの幅より長くてはなりません。

SQL インクルードのソースは、完全な SQL ステートメントを含む必要があります。これは強制ではありませんが、この規則に違反すると予測不能な結果が発生する可能性があります。

SQL トリガーに関する考慮事項**:** SQL トリガーが INCLUDE ステートメントを使用して ILE C をインクルードする場合、トリガー・プログラムの再作成を要求する操作はすべて、インクルード・ファイルへのランタイム依存関係を持ちます。トリガー・プログラムが見つからない場合は、トリガーが起動されたとき、表がコピーされたとき、または表が復元されたときに、データベースによってそれを自動的に再作成することができます。インクルード・ファイルが使用可能でない場合は、トリガーは作成されず、トリガーを要求する関数は失敗します。

例

- INCLUDE ステートメントを使用して、SQL 関数に共通の条件ハンドラーを埋め込みます。

```
CREATE FUNCTION H1 ()
RETURNS INT
DETERMINISTIC
NO EXTERNAL ACTION
NOT FENCED
BEGIN
  DECLARE RES INT;
  INCLUDE SQL SQLINCLUDE(COMMONCOND);

  SET RES = AFUNCTION();
  RETURN RES;
END;
```

  ライブラリー・リスト内のファイル SQLINCLUDE の COMMONCOND には、以下が含まれます。この条件ハンドラーは、特定の not found 条件に対して SQL プロシージャー ERR_PROC を呼び出します。メッセージ処理の中で、ROUTINE_SCHEMA および ROUTINE_SPECIFIC_NAME の組み込みグローバル変数を使用して、エラーのソースを示します。

```
DECLARE NO_FUNC CONDITION FOR SQLSTATE '42704';
DECLARE CONTINUE HANDLER FOR NO_FUNC
  BEGIN
    CALL ERR_PROC(SYSIBM.ROUTINE_SCHEMA CONCAT '.' CONCAT
                  SYSIBM.ROUTINE_SPECIFIC_NAME CONCAT
                  ': Function not found');
  END;
```

- INCLUDE ステートメントを使用して、C 関数の呼び出しを埋め込みます。これは、Qp0zLprintf インターフェースを使用して入力パラメーターをプリントします。

```
CREATE PROCEDURE LPRINTF(P1 VARCHAR(1000))
BEGIN
  IF P1 IS NOT NULL THEN
    INCLUDE QGPL/CINCLUDE(MYLPRINTF);
  END IF;
END;
```

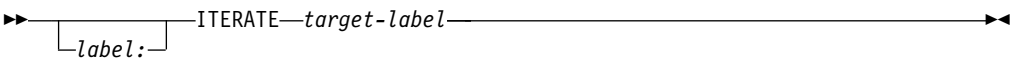
ライブラリー QGPL のファイル CINCLUDE 内のメンバー MYLPRINTF は、以下のコードを含みます。このインクルードは、関数の入力パラメーターへの参照を含みます。生成された C コードを調べて名前を判別する必要があります。

```
{
  /* declare prototype for Qp0zLprintf */
  extern int  Qp0zLprintf (char *format, ...);

  /* print input parameter to job log */
  Qp0zLprintf("%.*s\n", LPRINTF.P1.LEN, LPRINTF.P1.DAT);
}
```

ITERATE ステートメント

ITERATE ステートメントは、制御のフローをラベル付きループの先頭に戻します。

構文



説明

label

ITERATE ステートメントのラベルを指定します。このラベル名は、ルーチン名または同じ有効範囲内の別のラベルと同じものにすることはできません。詳しくは、 1777 ページの『SQL ラベルの参照』を参照してください。

target-label

制御のフローを渡す FOR、LOOP、REPEAT、または WHILE のいずれかのステートメントのラベルを指定します。*target-label* は、FOR、LOOP、REPEAT、または WHILE ステートメントのラベルとして定義する必要があります。

ITERATE ステートメントはその FOR、LOOP、REPEAT、または WHILE ステートメントの中、またはそのいずれかのステートメント内に直接または間接的にネストされたコードのブロック内になければならず、以下の制限の対象となります。

- ITERATE ステートメントが条件ハンドラー内にある場合、*target-label* はその条件ハンドラー内で定義する必要があります。
- ITERATE ステートメントが条件ハンドラー内にない場合、*target-label* を条件ハンドラー内で定義してはなりません。
- ITERATE ステートメントが FOR ステートメント内にある場合、*target-label* はその FOR ステートメント上の該当するラベルでなければなりません。あるいはそのラベルをこの FOR ステートメントの中で定義する必要があります。

注

SQLSTATE および **SQLCODE** 変数に関する考慮事項**:** ITERATE ステートメントは SQLSTATE および SQLCODE SQL 変数には影響しません。SQLSTATE および SQLCODE SQL 変数は、ITERATE ステートメントの終了時に、その ITERATE ステートメントの前に、最後に実行されたステートメントの結果を反映します。

例

この例では、カーソルを使用して、新規部門に関する情報を戻します。 *not_found* 条件ハンドラーが呼び出されると、制御のフローがループの外側に渡されます。*v_dept* の値が 'D11' の場合は、ITERATE ステートメントは制御のフローを LOOP ステートメントの先頭に戻します。それ以外の場合は、新規の行が DEPARTMENT 表に挿入されます。

```
CREATE PROCEDURE ITERATOR ()
   LANGUAGE SQL
   MODIFIES SQL DATA
   BEGIN
```

```
   DECLARE v_dept CHAR(3);
   DECLARE v_deptname VARCHAR(29);
   DECLARE v_admdept CHAR(3);
   DECLARE at_end INTEGER DEFAULT 0;
   DECLARE not_found CONDITION FOR SQLSTATE '02000';
   DECLARE c1 CURSOR FOR
     SELECT deptno,deptname,admrdept
     FROM department
     ORDER BY deptno;
   DECLARE CONTINUE HANDLER FOR not_found
     SET at_end = 1;
   OPEN c1;
   ins_loop:
   LOOP
      FETCH c1 INTO v_dept, v_deptname, v_admdept;
      IF at_end = 1 THEN
         LEAVE ins_loop;
      ELSEIF v_dept ='D11' THEN
         ITERATE ins_loop;
      END IF;
      INSERT INTO department (deptno,deptname,admrdept)
         VALUES('NEW', v_deptname, v_admdept);
   END LOOP;
   CLOSE c1;
END
```

終了 (LEAVE) ステートメント

LEAVE ステートメントは、ブロックまたはループを終了することによって実行を継続します。

構文

説明

ラベル *1*

LEAVE ステートメントのラベルを指定します。このラベル名は、ルーチン名または同じ有効範囲内の別のラベルと同じものにすることはできません。詳しくは、 1777 ページの『SQL ラベルの参照』を参照してください。

label2

終了する複合、FOR、LOOP、REPEAT、または WHILE ステートメントのラベルを指定します。

LEAVE ステートメントはハンドラーを終了する場合には使用できません。

注

オープン・カーソルに対する影響**:** LEAVE ステートメントが複合ステートメントの外部に制御を転送する際に、LEAVE ステートメントを含む複合ステートメント内で宣言されたすべてのオープン・カーソルは、それらが結果セットを返すために宣言されている場合、または *ENDACTGRP が指定されている場合を除き、クローズされます。

SQLSTATE および **SQLCODE** 変数に関する考慮事項**:** LEAVE ステートメントは SQLSTATE および SQLCODE SQL 変数には影響しません。SQLSTATE および SQLCODE SQL 変数は、LEAVE ステートメントの終了時に、その LEAVE ステートメントの前に、最後に実行されたステートメントの結果を反映します。

例

この例には、カーソル *c1* のデータを取り出すループが含まれています。 SQL 変数 *at_end* の値がゼロでない場合、LEAVE はループの外部に制御を転送します。

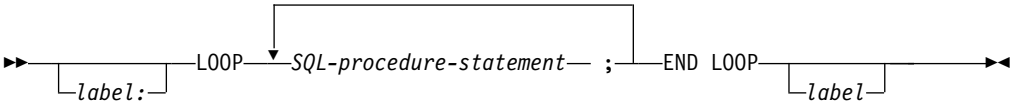
```
CREATE PROCEDURE LEAVE_LOOP (OUT COUNTER INTEGER)
   LANGUAGE SQL
   BEGIN
      DECLARE v_counter INTEGER;
      DECLARE v_firstnme VARCHAR(12);
      DECLARE v_midinit CHAR(1);
      DECLARE v_lastname VARCHAR(15);
      DECLARE at_end SMALLINT DEFAULT 0;
      DECLARE not_found CONDITION FOR SQLSTATE '02000';
      DECLARE c1 CURSOR FOR
         SELECT firstnme, midinit, lastname
         FROM employee;
      DECLARE CONTINUE HANDLER FOR not_found
         SET at_end = 1;
      SET v_counter = 0;
      OPEN c1;
```

```
   fetch_loop:
   LOOP
      FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
      IF at_end <> 0 THEN
         LEAVE fetch_loop;
      END IF;
      SET v_counter = v_counter + 1;
   END LOOP fetch_loop;
   SET counter = v_counter;
   CLOSE c1;
END
```

ループ (LOOP) ステートメント

LOOP ステートメントは、1 つのステートメントまたは 1 グループのステートメントの実行を繰り返します。

構文



説明

label
LOOP ステートメントのラベルを指定します。終了ラベルを指定する場合、開始ラベルと同じにしなければなりません。このラベル名は、ルーチン名または同じ有効範囲内の別のラベルと同じものにすることはできません。詳しくは、1777 ページの『SQL ラベルの参照』を参照してください。

SQL プロシージャー・ステートメント
ループで実行する SQL ステートメントを指定します。

注

診断領域に関する考慮事項**:** LOOP ステートメントの最初の繰り返しの開始時、および以降のすべての繰り返し時に、診断領域はクリアされます。

SQLSTATE および **SQLCODE** 変数に関する考慮事項**:** LOOP ステートメント内の最初の *SQL* プロシージャー・ステートメント を実行する前に、SQLSTATE および SQLCODE の値は、LOOP ステートメントの前に設定された最後の値を反映します。 GOTO または LEAVE ステートメントによってループが終了された場合、SQLSTATE および SQLCODE の値はそのステートメントの正常終了を反映します。LOOP ステートメントが繰り返されるときに、SQLSTATE および SQLCODE の値は LOOP ステートメント内で最後に実行された SQL ステートメントの結果を反映します。

例

このプロシージャーでは、LOOP ステートメントを使用して従業員表から値を取り出します。ループが繰り返されるたびに、OUT パラメーターの *counter* が増分し、*v_midinit* の値がチェックされて、その値がシングル・スペース (' ') でないかどうかが確認されます。 *v_midinit* が単一スペースの場合、 LEAVE ステートメントは制御のフローをループの外側に渡します。

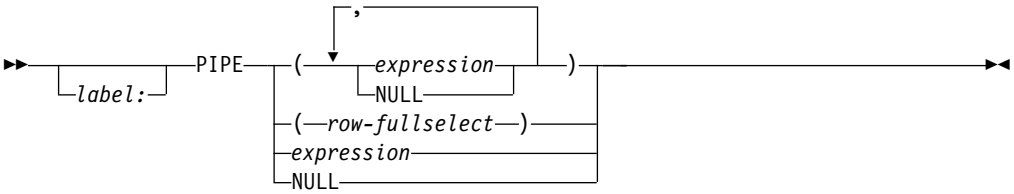
```
CREATE PROCEDURE LOOP_UNTIL_SPACE (OUT COUNTER INTEGER)
   LANGUAGE SQL
   BEGIN
      DECLARE v_counter INTEGER DEFAULT 0;
      DECLARE v_firstnme VARCHAR(12);
      DECLARE v_midinit CHAR(1);
      DECLARE v_lastname VARCHAR(15);
      DECLARE c1 CURSOR FOR
         SELECT firstnme, midinit, lastname
         FROM employee;
```

```
   DECLARE CONTINUE HANDLER FOR NOT FOUND
      SET counter = -1;
   OPEN c1;
   fetch_loop:
   LOOP
      FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
      IF v_midinit = ' ' THEN
         LEAVE fetch_loop;
      END IF;
      SET v_counter = v_counter + 1;
   END LOOP fetch_loop;
   SET counter = v_counter;
   CLOSE c1;
END
```

PIPE ステートメント

PIPE ステートメントは、表関数から 1 行を戻します。 PIPE ステートメントを使用する SQL 表関数は、パイプライン関数と呼ばれます。

構文



説明

label
PIPE ステートメントのラベルを指定します。このラベル名は、ルーチン名または同じ有効範囲内の別のラベルと同じものにすることはできません。詳しくは、1777 ページの『SQL ラベルの参照』を参照してください。

(*expression***, ...)**
関数から戻される行の値を指定します。リスト内の式または NULL キーワードの数は、関数結果の列の数と一致していなければなりません。各式のデータ・タイプは、 113 ページの『割り当ておよび比較』に説明されている記憶域割り当て規則を使用して、関数結果に対して定義されている対応する列のデータ・タイプに割り当てることができるものでなければなりません。

row-fullselect
単一行を返す全選択を指定します。全選択内の列数は、関数の結果の列数に一致していなければなりません。全選択の結果表の列のデータ・タイプは、 113 ページの『割り当ておよび比較』で説明されている記憶域割り当て規則を使用して、関数結果に定義された列のデータ・タイプに割り当て可能でなければなりません。行全選択の結果が 0 行である場合は、列ごとに NULL が戻されます。結果の中に複数の行がある場合には、エラーが戻されます。

expression
スカラー値が関数から返されるように指定します。表関数は単一の列を戻さなければならず、式の値はその列に割り当て可能なものでなければなりません。

NULL
関数から NULL 値が返されるよう指定します。複数の結果列がある場合は、列ごとに NULL 値が戻されます。

例

PIPE ステートメントを使用して、SQL 表関数からの行を戻します。

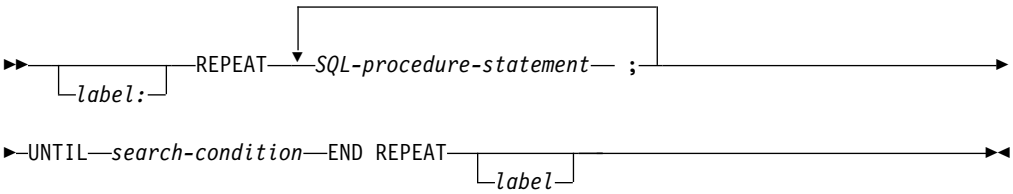
```
CREATE FUNCTION TRANSFORM() RETURNS TABLE ( EMPLOYEE_NAME CHAR(20), UNIQUE# INT )
BEGIN
  DECLARE EMPNAME VARCHAR(15);
  DECLARE MYRECNUM INTEGER DEFAULT 1;
  DECLARE AT_END INTEGER DEFAULT 0;
  DECLARE EMP_CURSOR CURSOR FOR SELECT lastname FROM employee;
  DECLARE CONTINUE HANDLER FOR SQLSTATE '02000'
```

```
    SET AT_END = 1;

  OPEN EMP_CURSOR;
  MYLOOP: LOOP
    FETCH EMP_CURSOR INTO EMPNAME;
    IF AT_END = 1 THEN
      LEAVE MYLOOP;
    END IF;
    PIPE (EMPNAME, MYRECNUM);  -- return single row
    SET MYRECNUM = MYRECNUM + 1;
  END LOOP;

  CLOSE EMP_CURSOR;
  RETURN;
END;
```

反復 (REPEAT) ステートメント

REPEAT ステートメントは、検索条件が真になるまで、1 つのステートメントまたは 1 グループのステートメントの実行を繰り返します。

構文



説明

label
REPEAT ステートメントのラベルを指定します。終了ラベルを指定する場合、開始ラベルと同じにしなければなりません。このラベル名は、ルーチン名または同じ有効範囲内の別のラベルと同じものにすることはできません。詳しくは、1777 ページの『SQL ラベルの参照』を参照してください。

SQL-procedure-statement
REPEAT ループで実行する SQL ステートメントを指定します。

search-condition
search-condition は、毎回、REPEAT ループの実行後に評価されます。この条件が真であれば、REPEAT ループは終了します。この条件が不明または偽であれば、ループは継続します。

注

診断領域に関する考慮事項**:** REPEAT ステートメントの最初の繰り返しの開始時、および以降のすべての繰り返し時に、診断領域はクリアされます。

SQLSTATE および **SQLCODE SQL** 変数に関する考慮事項**:** REPEAT ステートメントの繰り返しごとに、SQLSTATE および SQLCODE SQL 変数は、REPEAT ステートメント内の最初の *SQL-procedure-statement* が実行される前にクリアされます。 REPEAT ステートメントの最初の繰り返しの開始時に、SQLSTATE および SQLCODE 値は、REPEAT ステートメントの前に設定された最後の値を反映します。REPEAT ステートメントの繰り返し 2 から n の開始時に、SQLSTATE および SQLCODE 値は、その REPEAT ステートメントの UNTIL 文節における検索条件の評価結果を反映します。 GOTO、ITERATE、または LEAVE ステートメントによってループが終了された場合、SQLSTATE および SQLCODE の値はそのステートメントの正常終了を反映します。そうでない場合、REPEAT ステートメントの END REPEAT の完了後に、SQLSTATE および SQLCODE 値は、その REPEAT ステートメントの UNTIL 文節における検索条件の評価結果を反映します。

例

REPEAT ステートメントは、*not_found* 条件ハンドラーが呼び出されるまで、表から行を取り出します。

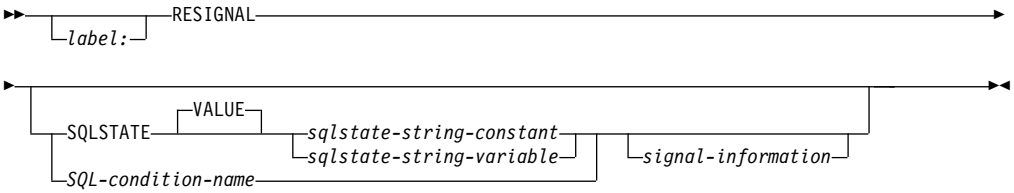
```
CREATE PROCEDURE REPEAT_STMT (OUT COUNTER INTEGER)
   LANGUAGE SQL
   BEGIN
      DECLARE v_counter INTEGER DEFAULT 0;
      DECLARE v_firstnme VARCHAR(12);
      DECLARE v_midinit CHAR(1);
      DECLARE v_lastname VARCHAR(15);
      DECLARE at_end SMALLINT DEFAULT 0;
      DECLARE not_found CONDITION FOR SQLSTATE '02000';
      DECLARE c1 CURSOR FOR
         SELECT firstnme, midinit, lastname
         FROM employee;
      DECLARE CONTINUE HANDLER FOR not_found
         SET at_end = 1;
      OPEN c1;
      fetch_loop:
      REPEAT
         FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
         SET v_counter = v_counter + 1;
         UNTIL at_end > 0
      END REPEAT fetch_loop;
      SET counter = v_counter;
      CLOSE c1;
   END
```

再通知 (RESIGNAL) ステートメント

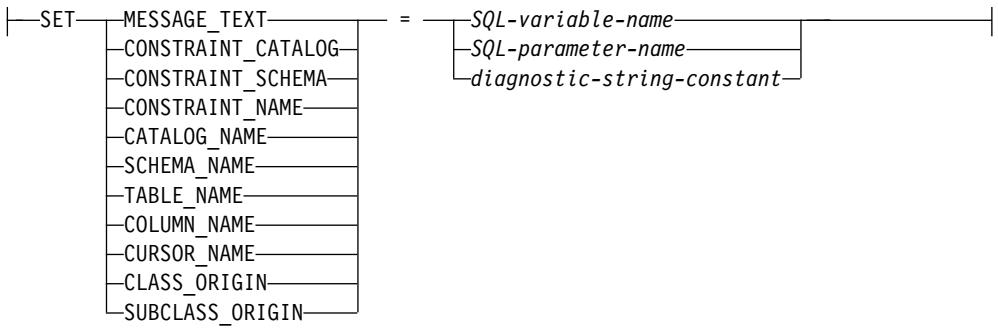
RESIGNAL ステートメントは、現行条件を再発生、または代替条件を発生させて、その処理がより高いレベルで行われるようにするために条件ハンドラー内で使用されます。これにより、オプションのメッセージ・テキストと一緒に例外、警告、または NOT FOUND 条件が返されます。

オペランドを指定せずに RESIGNAL ステートメントを実行すると、現行条件が外部に渡されます。

構文



signal-information:



説明

label

RESIGNAL ステートメントのラベルを指定します。このラベル名は、ルーチン名または同じ有効範囲内の別のラベルと同じものにすることはできません。詳しくは、 1777 ページの『SQL ラベルの参照』を参照してください。

SQLSTATE VALUE

再通知可能な SQLSTATE を指定します。任意の有効な SQLSTATE 値を使用できます。指定値は SQLSTATE の以下の規則に従っていなければなりません。

- 各文字は、数字 ('0' から '9') またはアクセント記号なしの英大文字 ('A' から 'Z') でなければなりません。
- SQLSTATE クラス (最初の 2 文字) は、'00' であってはなりません (これは正常終了を表します)。

SQLSTATE がこの規則に従っていないと、エラーが戻されます。

SQLSTATE ストリング定数
SQLSTATE ストリング定数 は、厳密に 5 文字の文字ストリング定数でなければなりません。

SQLSTATE ストリング変数
SQLSTATE ストリング変数 は、文字または Unicode グラフィック変数でなければなりません。*SQLSTATE* ストリング変数 の内容の実際の長さは、5 でなければなりません。

SQL-condition-name
戻される条件の名前を指定します。 *SQL-condition-name* は *compound-statement* の中で宣言する必要があります。

SET
条件情報項目 への値の割り当てを指定します。条件情報項目 値は、 GET DIAGNOSTICS ステートメントを使用してアクセスできます。 SQLCA 内でアクセス可能な条件情報項目 は、MESSAGE_TEXT だけです。

MESSAGE_TEXT
エラーまたは警告を説明するストリングを指定します。

SQLCA を使用する場合、

- このストリングは、SQLCA の SQLERRMC フィールドに戻されます。
- ストリングの実際の長さが 1000 バイトを超える場合、警告せずに切り捨てられます。

CONSTRAINT_CATALOG
通知されたエラーまたは警告に関連する制約を含む、データベースの名前を示すストリングを指定します。

CONSTRAINT_SCHEMA
通知されたエラーまたは警告に関連する制約を含む、スキーマの名前を示すストリングを指定します。

CONSTRAINT_NAME
通知されたエラーまたは警告に関連する制約の名前を示すストリングを指定します。

CATALOG_NAME
通知されたエラーまたは警告に関連する表またはビューを含む、データベースの名前を示すストリングを指定します。

SCHEMA_NAME
通知されたエラーまたは警告に関連する表またはビューを含む、スキーマの名前を示すストリングを指定します。

TABLE_NAME
通知されたエラーまたは警告に関連する表またはビューの名前を示すストリングを指定します。

COLUMN_NAME
通知されたエラーまたは警告に関連する表またはビューの列名を示すストリングを指定します。

CURSOR_NAME
通知されたエラーまたは警告に関連するカーソルの名前を示すストリングを指定します。

CLASS_ORIGIN
通知されたエラーまたは警告に関連する SQLSTATE クラスの起点を示すストリングを指定します。

SUBCLASS_ORIGIN
通知されたエラーまたは警告に関連する SQLSTATE サブクラスの起点を示すストリングを指定します。

SQL-variable-name
条件情報項目 に割り当てる値を含む、複合ステートメント 内で宣言される SQL 変数を示します。 SQL 変数は、CHAR、VARCHAR、Unicode GRAPHIC、または Unicode VARGRAPHIC 変数として定義する必要があります。

SQL-parameter-name
条件情報項目 に割り当てる値を含む、複合ステートメント 内で宣言される SQL パラメーターを示します。 SQL パラメーターは、CHAR、VARCHAR、Unicode GRAPHIC、または Unicode VARGRAPHIC 変数として定義する必要があります。

診断ストリング定数
条件情報項目 に割り当てる値を含む、文字ストリング定数を指定します。

注

SQLSTATE 値: RESIGNAL ステートメントには、任意の有効な SQLSTATE 値を使用できますが、プログラマーは、アプリケーション用に予約されている範囲に基づいて、新規の SQLSTATE を定義することをお勧めします。そうすれば、使用する SQLSTATE 値が今後リリースされるデータベース・マネージャーで定義される値と重なってしまう可能性を防止できます。

SQLSTATE の詳細については、「SQL メッセージおよびコード」トピック集を参照してください。

割り当て: RESIGNAL ステートメントが実行されるとき、指定した各ストリング定数 および変数 の値は、対応する条件情報項目 に割り当てられます。ただし、ストリング定数 または変数 の長さが対応する条件情報項目 の最大長を超える場合は、警告せずに切り捨てられます。割り当て規則の詳細については、 113 ページの『割り当ておよび比較』を参照してください。特定の条件情報項目 の最大長については、 1489 ページの『GET DIAGNOSTICS』を参照してください。

RESIGNAL ステートメントの処理:

- RESIGNAL ステートメントを SQLSTATE 文節や *SQL-condition-name* なしで指定する場合、SQL 関数、SQL プロシージャー、または SQL トリガーは、ハンドラーを呼び出したのと同一の条件で再通知し、SQLCODE は変更されません。
- RESIGNAL ステートメントが発行され、 SQLSTATE または *SQL-condition-name* が指定されている場合は、 SQLCODE は次のように SQLSTATE 値に基づいて設定されます。

  - 指定された SQLSTATE クラスが '01' または '02' の場合、警告または NOT FOUND が通知され、SQLCODE は +438 に設定されます。
  - それ以外の場合、例外が戻され、SQLCODE は -438 に設定されます。

SQLSTATE または条件が、例外が通知されたこと (SQLSTATE クラスが '01' または '02' 以外) を示している場合は、次のようになります。

- その RESIGNAL ステートメントと同じ複合ステートメント内にハンドラーが存在し、しかも複合ステートメント の中に SQLEXCEPTION、あるいは指定の SQLSTATE または条件に対するハンドラーが含まれている場合は、例外は処理され、制御はそのハンドラーに渡されます。
- 複合ステートメント がネストされていて、外側の複合ステートメント の中に SQLEXCEPTION あるいは指定の SQLSTATE または条件に対するハンドラーが含まれている場合は、例外は処理され、制御はそのハンドラーに渡されます。
- それ以外の場合は、例外は処理されず、制御は即時に複合ステートメントの終了点に戻されます。

SQLSTATE または条件が、警告 (SQLSTATE クラス '01') または NOT FOUND (SQLSTATE クラス '02') が通知されたことを示している場合は、次のようになります。

- その RESIGNAL ステートメントと同じ複合ステートメント内にハンドラーが存在し、しかも複合ステートメント の中に SQLWARNING (SQLSTATE クラスが '01' の場合)、NOT FOUND (SQLSTATE クラスが '02' の場合)、または指定の SQLSTATE または条件に対するハンドラーが含まれている場合は、警告または NOT FOUND 条件は処理され、制御はそのハンドラーに渡されます。
- 複合ステートメント がネストされていて、外側のレベルの複合ステートメントの中に、SQLWARNING (SQLSTATE クラスが '01' の場合)、NOT FOUND (SQLSTATE クラスが '02' の場合)、または指定の SQLSTATE または条件に対するハンドラーが含まれている場合は、警告または NOT FOUND 条件は処理されず、次のステートメントから処理を継続します。
- それ以外の場合、警告は処理されず、次のステートメントから処理を継続します。

診断領域に関する考慮事項: RESIGNAL ステートメントは、現行の診断領域の内容を変更する場合があります。 SQLSTATE または *SQL-condition-name* が RESIGNAL ステートメントの一部として指定された場合、RESIGNAL ステートメントが診断領域のクリアを指定して開始し、指定された SQLSTATE または *SQL-condition-name* を反映するように RETURNED_SQLSTATE を設定します。通知情報 が指定されると、条件領域内の対応する項目に指定された値が割り当てられます。 DB2_RETURNED_SQLCODE は、指定された SQLSTATE または *SQL-condition-name* に対応して +438 または -438 に設定されます。

例

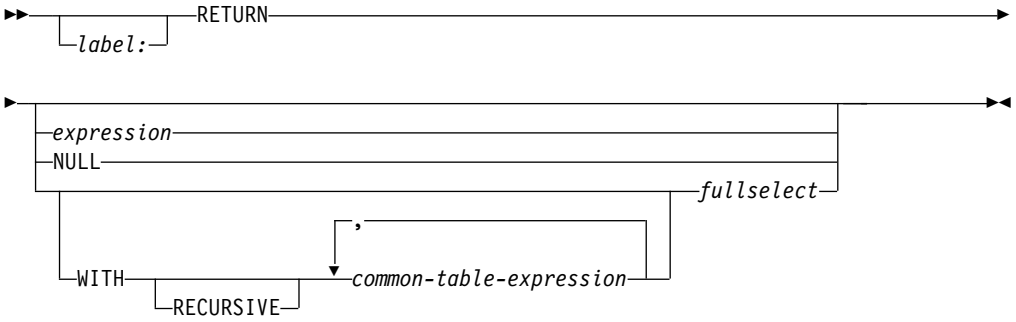
この例は、ゼロによる割り算 (ゼロ除算) エラーを検出します。 IF ステートメントは、SIGNAL ステートメントを使用して、オーバーフロー条件ハンドラーを呼び出します。オーバーフロー条件ハンドラーは、RESIGNAL ステートメントを使用して、異なる SQLSTATE 値をクライアント・アプリケーションに戻します。

```
CREATE PROCEDURE divide ( IN numerator INTEGER,
                          IN denominator INTEGER,
                           OUT divide_result INTEGER )
  LANGUAGE SQL
    BEGIN
       DECLARE overflow CONDITION FOR '22003';
       DECLARE CONTINUE HANDLER FOR overflow
           RESIGNAL SQLSTATE '22375';
       IF denominator = 0 THEN
           SIGNAL overflow;
       ELSE
           SET divide_result = numerator / denominator;
       END IF;
     END
```

戻り (RETURN) ステートメント

RETURN ステートメントは、ルーチンから戻るための処理を実行します。 SQL スカラー関数の場合は、関数の結果を戻します。SQL 非パイプライン表関数の場合は、関数の結果として表を戻します。 SQL パイプライン表関数の場合、結果表の終わりに達したことを示します。SQL プロシージャーの場合は、オプションで整数の状況値を戻します。

構文



説明

label
RETURN ステートメントのラベルを指定します。このラベル名は、ルーチン名または同じ有効範囲内の別のラベルと同じものにすることはできません。詳しくは、 1777 ページの『SQL ラベルの参照』を参照してください。

expression
ルーチンから戻される値を指定します。

- ルーチンがスカラー関数の場合、結果のデータ・タイプは、 113 ページの『割り当ておよび比較』で説明されている記憶域割り当て規則を使用して、関数結果に定義されたデータ・タイプに割り当て可能でなければなりません。集約関数、または集約関数をソースにしたユーザー定義関数は、SQL スカラー関数の RETURN ステートメントには指定できません。
- ルーチンが表関数の場合は、(スカラー全選択以外の) スカラー式を指定することはできません。
- ルーチンがプロシージャーである場合、*expression* のデータ・タイプは INTEGER であることが必要です。*expression* の評価が NULL 値の場合、値ゼロが戻されます。

NULL
ルーチンから NULL 値を戻すように指定します。

- ルーチンがスカラー関数の場合、NULL 値が戻されます。
- ルーチンが表関数の場合、NULL を指定することはできません。
- ルーチンがプロシージャーの場合、NULL を指定することはできません。

WITH *common-table-expression*
fullselect 内で 1 つ以上の *common-table-expression* を使用することを指定します。

fullselect
ルーチンに対して戻される行を指定します。

- ルーチンがスカラー関数の場合、全選択は 1 つの列と、最大で 1 つの行を返さなければなりません。結果列のデータ・タイプは、 113 ページの『割り当ておよび比較』で説明されている記憶域割り当て規則を使用して、関数結果に定義されたデータ・タイプに割り当て可能でなければなりません。
- ルーチンが表関数の場合は、全選択は 1 つ以上の列を持つゼロ以上の行を返すことができます。全選択内の列数は、関数の結果の列数に一致していなければなりません。また、全選択の結果表の列のデータ・タイプは、 113 ページの『割り当ておよび比較』で説明されている記憶域割り当て規則を使用して、関数結果に定義された列のデータ・タイプに割り当て可能でなければなりません。
- ルーチンがプロシージャーの場合、全選択を指定することはできません。

注

表関数からの戻り:

- *SQL-routine-body* で実行される最後のステートメントは RETURN ステートメントでなければなりません。
- 非パイプライン表関数の場合、RETURN ステートメントは戻される行を示します。必ず、厳密に 1 つの RETURN ステートメントを指定し、これは *fullselect* を含む必要があります。
- パイプライン表関数の場合、必ず、1 つ以上の RETURN ステートメントを指定し、これに戻り値が含まれていてはなりません。

プロシージャーからの戻り:

- プロシージャーから戻るのに戻り値が指定された RETURN ステートメントが使用される場合、SQLCA または診断領域内の SQLCODE、SQLSTATE、およびメッセージ長はゼロに初期化され、メッセージ・テキストはブランクに設定されます。エラーは呼び出し元に戻されません。
- プロシージャーから戻るのに RETURN ステートメントが使用されない場合、または RETURN ステートメントで値が指定されていない場合は、次のようになります。
  - プロシージャーがゼロ以上の SQLCODE で戻る場合、 GET DIAGNOSTICS ステートメント内にある DB2_RETURN_STATUS の指定されたターゲットは 0 の値に設定されます。
  - プロシージャーがゼロ未満の SQLCODE で戻る場合、 GET DIAGNOSTICS ステートメント内にある DB2_RETURN_STATUS の指定されたターゲットは -1 の値に設定されます。
- プロシージャーから値が戻された場合、呼び出し元は次の方法で値にアクセスできます。
  - SQL プロシージャーが別の SQL プロシージャーから呼び出された場合、GET DIAGNOSTICS ステートメントを使用して DB2_RETURN_STATUS を取り出します。

- ODBC または JDBC アプリケーションのエスケープ文節 CALL 構文 (?=CALL...) で、戻り値パラメーター・マーカーを宛先とするパラメーターを使用します。
- SQLCODE がゼロ未満ではない場合、SQL プロシージャーの CALL 処理によって戻された SQLCA から直接 sqlerrd[0] の値を取り出します。SQLCODE がゼロ未満の場合は、sqlerrd[0] の値は設定されず、アプリケーションは戻り状況値が -1 であるものと想定します。

RETURN の制約事項:

- RETURN は、SQL トリガーでは使用できません。
- 複合 (動的) ステートメントにおいて、RETURN で戻り値を指定することはできません。

例

例 1: RETURN ステートメントを使用して SQL プロシージャーから戻り、成功したときは状況値 0、失敗したときは状況値 -200 が返されます。

```
BEGIN
  ...
  GOTO fail;
  ...
  success: RETURN 0
  fail: RETURN -200
...
END
```

例 2: 既存のサイン (正弦) 関数とコサイン (余弦) 関数を使用して、値のタンジェント (正接) を戻すスカラー関数を定義します。

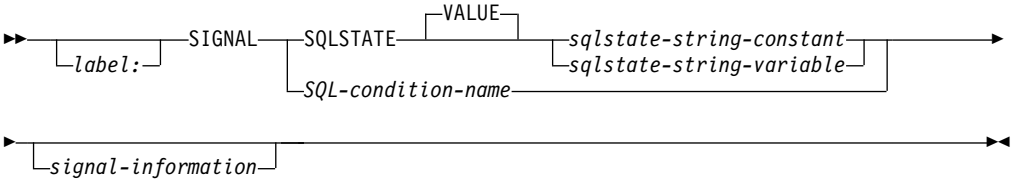
```
CREATE FUNCTION mytan (x DOUBLE)
  RETURNS DOUBLE
  LANGUAGE SQL
  CONTAINS SQL
  NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN SIN(x)/COS(x)
```

通知 (SIGNAL) ステートメント

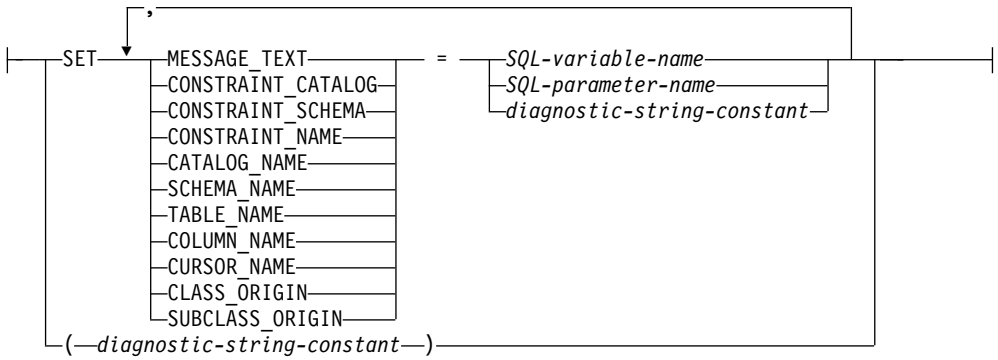
SIGNAL ステートメントは、エラー条件または警告条件を通知します。これは、指定の SQLSTATE とオプションの条件情報項目 を使用して、エラーまたは警告を戻します。 SQL 関数、SQL プロシージャー、または SQL トリガー内の SIGNAL の構文は、他のコンテキストで SIGNAL ステートメントとしてサポートされているものと似ています。

詳しくは、 1740 ページの『SIGNAL』を参照してください。

構文



signal-information:



説明

label

SIGNAL ステートメントのラベルを指定します。このラベル名は、ルーチン名または同じ有効範囲内の別のラベルと同じものにすることはできません。詳しくは、 1777 ページの『SQL ラベルの参照』を参照してください。

SQLSTATE VALUE

通知する SQLSTATE を指定します。指定値は NULL 不可で、次の SQLSTATE の規則に従わなければなりません。

- 各文字は、数字 ('0' から '9') またはアクセント記号なしの英大文字 ('A' から 'Z') でなければなりません。
- SQLSTATE クラス (最初の 2 文字) は、'00' であってはなりません (これは正常終了を表します)。

SQLSTATE がこの規則に従っていないと、エラーが戻されます。

SQLSTATE ストリング定数
SQLSTATE ストリング定数 は、厳密に 5 文字の文字ストリング定数でなければなりません。

SQLSTATE ストリング変数
SQLSTATE ストリング変数 は、文字または Unicode グラフィック変数でなければなりません。変数 の内容の実際の長さは、5 でなければなりません。

SQL-condition-name
通知される条件の名前を指定します。 *SQL-condition-name* は *compound-statement* の中で宣言する必要があります。

SET
条件情報項目 への値の割り当てを指定します。条件情報項目 値は、 GET DIAGNOSTICS ステートメントを使用してアクセスできます。 SQLCA 内でアクセス可能な条件情報項目 は、MESSAGE_TEXT だけです。

MESSAGE_TEXT
エラーまたは警告を説明するストリングを指定します。

SQLCA を使用する場合、

- このストリングは、SQLCA の SQLERRMC フィールドに戻されます。
- ストリングの実際の長さが 1000 バイトを超える場合、警告せずに切り捨てられます。

CONSTRAINT_CATALOG
通知されたエラーまたは警告に関連する制約を含む、データベースの名前を示すストリングを指定します。

CONSTRAINT_SCHEMA
通知されたエラーまたは警告に関連する制約を含む、スキーマの名前を示すストリングを指定します。

CONSTRAINT_NAME
通知されたエラーまたは警告に関連する制約の名前を示すストリングを指定します。

CATALOG_NAME
通知されたエラーまたは警告に関連する表またはビューを含む、データベースの名前を示すストリングを指定します。

SCHEMA_NAME
通知されたエラーまたは警告に関連する表またはビューを含む、スキーマの名前を示すストリングを指定します。

TABLE_NAME
通知されたエラーまたは警告に関連する表またはビューの名前を示すストリングを指定します。

COLUMN_NAME
通知されたエラーまたは警告に関連する表またはビューの列名を示すストリングを指定します。

CURSOR_NAME
通知されたエラーまたは警告に関連するカーソルの名前を示すストリングを指定します。

CLASS_ORIGIN
通知されたエラーまたは警告に関連する SQLSTATE クラスの起点を示すストリングを指定します。

SUBCLASS_ORIGIN
通知されたエラーまたは警告に関連する SQLSTATE サブクラスの起点を示すストリングを指定します。

SQL-variable-name
条件情報項目 に割り当てる値を含む、複合ステートメント 内で宣言される SQL 変数を示します。 SQL 変数は、CHAR、VARCHAR、Unicode GRAPHIC、または Unicode VARGRAPHIC 変数として定義する必要があります。

SQL-parameter-name
条件情報項目 に割り当てる値を含む、複合ステートメント 内で宣言される SQL パラメーターを示します。 SQL パラメーターは、CHAR、VARCHAR、Unicode GRAPHIC、または Unicode VARGRAPHIC 変数として定義する必要があります。

診断ストリング定数
条件情報項目 に割り当てる値を含む、文字ストリング定数を指定します。

*(*診断ストリング定数*)*
メッセージ・テキストを入れる文字ストリング定数を指定します。

この形式が使用できるのは、CREATE TRIGGER ステートメントのトリガー・アクションにおいてのみです。

ANS および ISO 規格に準拠するためには、この形式は使用してはなりません。これは、他のプロダクトとの互換性のために提供されています。

注

SQLSTATE 値: SIGNAL ステートメントには、任意の有効な SQLSTATE 値を使用できますが、プログラマーは、アプリケーション用に予約されている範囲に基づいて、新規の SQLSTATE を定義することをお勧めします。そうすれば、使用する SQLSTATE 値が今後リリースされるデータベース・マネージャーで定義される値と重なってしまう可能性を防止できます。

SQLSTATE の詳細については、「SQL メッセージおよびコード」トピック集を参照してください。

割り当て: SIGNAL ステートメントが実行されるとき、指定した各ストリング定数および変数 の値は、対応する条件情報項目 に割り当てられます。ただし、ストリング定数 または変数 の長さが対応する条件情報項目 の最大長を超える場合は、警告せずに切り捨てられます。割り当て規則の詳細については、 113 ページの『割り当ておよび比較』を参照してください。特定の条件情報項目 の最大長については、 1489 ページの『GET DIAGNOSTICS』を参照してください。

SIGNAL ステートメントの処理**:** SIGNAL ステートメントが実行された場合、SQLCA で戻される SQLCODE は、次のように SQLSTATE 値に基づいて設定されます。

- 指定された SQLSTATE クラスが '01' または '02' の場合、警告または NOT FOUND が通知され、SQLCODE は +438 に設定されます。
- それ以外の場合、例外が通知され、SQLCODE は -438 に設定されます。

SQLSTATE または条件が、例外 (SQLSTATE クラスが '01' または '02' 以外) が通知されたことを示している場合は、次のようになります。

- その SIGNAL ステートメントと同じ複合ステートメント内にハンドラーが存在し、しかもその複合ステートメントの中に SQLEXCEPTION あるいは指定の SQLSTATE または条件に対するハンドラーが含まれている場合、例外は処理され、制御はそのハンドラーに渡されます。
- 複合ステートメント がネストされていて、外側の複合ステートメント の中に SQLEXCEPTION あるいは指定の SQLSTATE または条件に対するハンドラーが含まれている場合は、例外は処理され、制御はそのハンドラーに渡されます。
- それ以外の場合は、例外は処理されず、制御は即時に複合ステートメントの終了点に戻されます。

SQLSTATE または条件が、警告 (SQLSTATE クラス '01') または NOT FOUND (SQLSTATE クラス '02') が通知されたことを示している場合は、次のようになります。

- その SIGNAL ステートメントと同じ複合ステートメント内にハンドラーが存在し、しかもその複合ステートメントの中に SQLWARNING (SQLSTATE クラスが '01'の場合)、NOT FOUND (SQLSTATE クラスが '02' の場合)、または指定の SQLSTATE または条件に対するハンドラーが含まれている場合、警告または NOT FOUND 条件は処理され、制御はそのハンドラーに渡されます。
- 複合ステートメント がネストされていて、外側のレベルの複合ステートメントの中に、SQLWARNING (SQLSTATE クラスが '01' の場合)、NOT FOUND (SQLSTATE クラスが '02' の場合)、または指定の SQLSTATE または条件に対するハンドラーが含まれている場合は、警告または NOT FOUND 条件は処理されず、次のステートメントから処理を継続します。
- それ以外の場合、警告は処理されず、次のステートメントから処理を継続します。

診断領域に関する考慮事項**:** SIGNAL ステートメントは診断領域のクリアを指定して開始し、指定された SQLSTATE または *SQL-condition-name* を反映するように RETURNED_SQLSTATE を設定します。通知情報 が指定されると、条件領域内の対応する項目に指定された値が割り当てられます。 DB2_RETURNED_SQLCODE は、指定された SQLSTATE または *SQL-condition-name* に対応して +438 または -438 に設定されます。

例

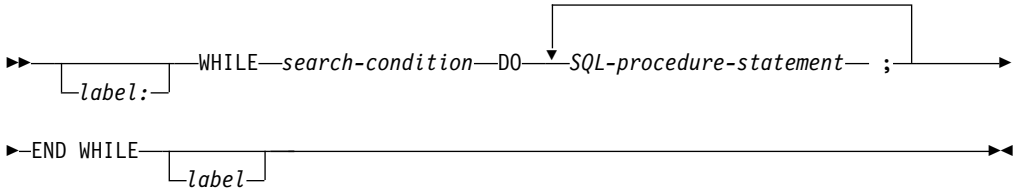
カスタマー番号がアプリケーションに知られていない場合にアプリケーション・エラーを通知するオーダー・システム用の SQL プロシージャー。 ORDERS 表には CUSTOMER 表への外部キーが含まれるので、オーダーが挿入可能になる前に CUSTNO が存在している必要があります。

```
CREATE PROCEDURE SUBMIT_ORDER
   (IN ONUM INTEGER, IN CNUM INTEGER,
    IN PNUM INTEGER, IN QNUM INTEGER)
    LANGUAGE SQL
    MODIFIES SQL DATA
   BEGIN
      DECLARE EXIT HANDLER FOR SQLSTATE VALUE '23503'
         SIGNAL SQLSTATE '75002'
            SET MESSAGE_TEXT = 'Customer number is not known';
      INSERT INTO ORDERS (ORDERNO, CUSTNO, PARTNO, QUANTITY)
         VALUES (ONUM, CNUM, PNUM, QNUM);
   END
```

WHILE ステートメント

WHILE ステートメントは、指定された条件が真である間、ステートメントの実行を繰り返します。

構文



説明

label
WHILE ステートメントのラベルを指定します。終了ラベルを指定する場合、開始ラベルと同じにしなければなりません。このラベル名は、ルーチン名または同じ有効範囲内の別のラベルと同じものにすることはできません。詳しくは、1777 ページの『SQL ラベルの参照』を参照してください。

search-condition
WHILE ループの実行の前に評価される条件を指定します。この条件が真であれば、WHILE ループ内の *SQL プロシージャー・ステートメント* が実行されます。

SQL-procedure-statement
WHILE ループで実行する 1 つ以上の SQL ステートメントを指定します。

注

診断領域に関する考慮事項**:** WHILE ステートメントの最初の繰り返しの開始時、および以降のすべての繰り返し時に、診断領域はクリアされます。

SQLSTATE および **SQLCODE SQL** に関する考慮事項**:** WHILE ステートメントの各繰り返しで、最初の *SQL-procedure-statement* を実行する際は、SQLSTATE および SQLCODE SQL 変数はその WHILE ステートメントの検索条件の評価結果を反映します。 GOTO、ITERATE、または LEAVE ステートメントによってループが終了された場合、SQLSTATE および SQLCODE の値はそのステートメントの正常終了を反映します。そうでない場合、WHILE ステートメントの END WHILE の完了後に、SQLSTATE および SQLCODE はその WHILE ステートメントの検索条件の評価結果を反映します。

例

この例では、WHILE ステートメントを使用して FETCH ステートメントおよび SET ステートメントを繰り返し実行します。 SQL 変数 *v_counter* の値が IN パラメーター *deptNumber* で識別される部門の従業員数の半分未満である間は、WHILE ステートメントは、FETCH および SET ステートメントを実行し続けます。条件が

真にならなくなった時点で、制御のフローは WHILE ステートメントから離れ、カーソルがクローズされます。

```
CREATE PROCEDURE dept_median
   (IN deptNumber SMALLINT,
    OUT medianSalary DECIMAL(7,2))
    LANGUAGE SQL
    BEGIN
       DECLARE v_numRecords INTEGER DEFAULT 1;
       DECLARE v_counter INTEGER DEFAULT 0;
       DECLARE c1 CURSOR FOR
          SELECT salary
          FROM staff
          WHERE dept = deptNumber
          ORDER BY salary;
       DECLARE EXIT HANDLER FOR NOT FOUND
          SET medianSalary = 6666;
       SET medianSalary = 0;
       SELECT COUNT(*) INTO v_numRecords
          FROM staff
          WHERE dept = deptNumber;
       OPEN c1;
       WHILE v_counter < (v_numRecords/2 + 1) DO
          FETCH c1 INTO medianSalary;
          SET v_counter = v_counter +1;
       END WHILE;
       CLOSE c1;
    END
```

付録 A. SQL の制約

以下の表に、Db2 for i のデータベース・マネージャーにより SQL およびデータベースに課せられる一定の制約を示します。

注:

- システム・ストレージの制限により、ここで指定される制限が無効になる場合があります。例えば、 1295 ページの『最大行サイズ 』を参照してください。
- ストレージ の制限とは、その制限が使用可能なストレージの量に依存していることを意味します。
- ステートメント の制限とは、その制限がステートメントの最大長制限に依存していることを意味します。

表 *119. ID* の長さに関する制約

制限される **ID**	**Db2 for i** の制限
権限名の最大長	10[131]
相関名の最大長	128
カーソル名の最大長	128
記述子名の最大長	128
外部プログラム名の最大長 (ストリング形式)	279[132]
外部プログラム名の最大長 (非修飾形式)	10
ホスト ID の最大長[133]	128
パッケージ・バージョン ID の最大長	64
パーティション名の最大長	10
セーブポイント名の最大長	128
スキーマ名の最大長	128
サーバー名の最大長	18
ステートメント名の最大長	128
SQL 条件名の最大長	128
SQL ラベルの最大長	128
非修飾の別名の最大長	128
非修飾の列名の最大長	128
非修飾の制約名の最大長	128
非修飾の特殊タイプ名の最大長	128
非修飾の関数名の最大長	128
非修飾のグローバル変数名の最大長	128
非修飾の索引名の最大長	128
非修飾のマスク名の最大長	128
非修飾のノード・グループ名の最大長	10
非修飾のパッケージ名の最大長	10
非修飾の許可名の最大長	128
非修飾のプロシージャー名の最大長	128
非修飾のシーケンス名の最大長	128
非修飾の特定名の最大長	128
非修飾の SQL パラメーター名の最大長	128
非修飾の SQL 変数名の最大長	128
非修飾のシステム列名の最大長	10
非修飾のシステム・オブジェクト名の最大長	10
非修飾のシステム・スキーマ名の最大長	10
非修飾の表名およびビュー名の最大長	128
非修飾のトリガー名の最大長	128
非修飾の XSR オブジェクト名の最大長	128
XMLELEMENT、XMLFOREST、XMLATTRIBUTES、XMLNAMESPACES、 XMLPI で指定する XML のエレメント名、属性名、接頭部名、処理命令名の最大長	128

表 *119. ID* の長さに関する制約 *(*続き*)*

制限される **ID**	**Db2 for i** の制限
XMTABLE で指定する XML パス名の最大長	128
解析する XML 文書の XML のエレメント名、属性名、接頭部名、処理命令名の最大長	1000
XML スキーマの Uniform Resource Identifier (URI) の最大長	1000
JSON パス名の最大長	128

131. アプリケーション・リクエスターとしては、Db2 for i は 255 バイトまでの権限名を送信できます。

132. REXX プロシージャーの場合、この制限は 33 です。

133. RPG、COBOL、または REXX プログラムの場合、この制限は 64 です。

表 120. 数値の制約

数値の制約	Db2 for i の制限
SMALLINT (短整数) の最小値	-32 768
SMALLINT (短整数) の最大値	+32 767
INTEGER (整数) の最小値	-2 147 483 648
INTEGER (整数) の最大値	+2 147 483 647
BIGINT (長整数) の最小値	-9 223 372 036 854 775 808
BIGINT (長整数) の最大値	+9 223 372 036 854 775 807
10 進数の精度の最大値	63
REAL 値の最大指数 (E^{max})	38
REAL (実数) の最小値[134]	$-3.4x10^{38}$
REAL (実数) の最大値[134]	$+3.4x10^{38}$
REAL 値の最小指数 (E^{min})	-38
REAL (実数) の正の最小値[134]	$+1.18x10^{-38}$
REAL (実数) の負の最大値[134]	$-1.18x10^{-38}$
DOUBLE 値の最大指数 (E^{max})	308
DOUBLE の最小値[134]	$-1.79x10^{308}$
DOUBLE の最大値[134]	$+1.79x10^{308}$
DOUBLE 値の最小指数 (E^{min})	-308
DOUBLE の正の最小値[134]	$+2.23x10^{-308}$
DOUBLE の負の最大値[134]	$-2.23x10^{-308}$
DECFLOAT(16) 値の最大指数 (E^{max})	384
DECFLOAT(16) の最小値[135]	$-9.999999999999999x10^{384}$
DECFLOAT(16) の最大値[135]	$9.999999999999999x10^{384}$
DECFLOAT(16) 値の最小指数 (E^{min})	-383
正の DECFLOAT(16) の最小値[135]	$1x10^{-383}$
負の DECFLOAT(16) の最大値[135]	$-1x10^{-383}$
DECFLOAT(34) 値の最大指数 (E^{max})	6144
DECFLOAT(34) の最小値[135]	$-9.999999999999999999999999999999999x10^{6144}$
DECFLOAT(34) の最大値[135]	$9.999999999999999999999999999999999x10^{6144}$
DECFLOAT(34) 値の最小指数 (E^{min})	-6143
正の DECFLOAT(34) の最小値[135]	$1x10^{-6143}$
負の DECFLOAT(34) の最大値[135]	$-1x10^{-6143}$

134. 記載した値は概算値です。

135. DECFLOAT の正規数には制約があります。 DECFLOAT には、NaN および無限大などの特殊値も含まれており、これらの特殊値も有効です。 DECFLOAT は、資料に記載された範囲から外れた、正常値に達していない数値もサポートします。

表 121. ストリングの制約

ストリングの制約	Db2 for i の制限
CHAR (文字) の最大長 (バイト数)	32765[136]
VARCHAR (可変長文字) の最大長 (バイト数)	32739[136]
CLOB の最大長 (バイト数)	2 147 483 647
GRAPHIC の最大長 (2 バイト文字数)	16382[136]
VARGRAPHIC の最大長 (2 バイト文字数)	16369[136]
DBCLOB の最大長 (2 バイト文字数)	1 073 741 823
BINARY の最大長 (バイト数)	32765[136]
VARBINARY の最大長 (バイト数)	32739[136]
BLOB の最大長 (バイト数)	2 147 483 647
直列化された XML の最大長 (バイト数)	2 147 483 647
文字定数の最大長	32740
グラフィック定数の最大長	16370
バイナリー定数の最大長	32740
連結した文字ストリングの最大長	2 147 483 647
連結したグラフィック・ストリングの最大長	1 073 741 823
連結したバイナリー・ストリングの最大長	2 147 483 647
16 進定数桁の最大数	32 762
カタログ・コメントの最大長	2000[137]
列ラベルの最大長 (バイト数)	60
SQL ルーチン・ラベルの最大長	128
表、パッケージ、または別名のラベルの最大長	50
C の NUL 終了ストリングの最大長	32739[136]
C の NUL 終了グラフィックの最大長	16369[136]

表 122. XML 制限

XML 制限	Db2 for i の制限
XML スキーマ文書の最大長 (バイト数)	2 147 483 647
解析する XML エンティティーの最大長	2 147 483 647
内部 XML ツリーの最大深さ	128

表 123. JSON の制限

JSON の制限	Db2 for i の制限
JSON 文書の最大長 (バイト数)	2 147 483 647

136. 列が NOT NULL の場合、最大値は 1 つ大きくなります。

137. シーケンスの場合は、制限は 500 です。

表 *124.* 日付/時刻の制約

日付/時刻の制約	**Db2 for i** の制限
DATE (日付) の最小値	0001-01-01
DATE (日付) の最大値	9999-12-31
TIME (時刻) の最小値	00:00:00
TIME (時刻) の最大値	24:00:00
TIMESTAMP (タイム・スタンプ) の最小値	0001-01-01-00.00.00.000000000000
TIMESTAMP (タイム・スタンプ) の最大値	9999-12-31-24.00.00.000000000000
タイム・スタンプの精度の最小値	0
タイム・スタンプの精度の最大値	12

表 *125.* データ・リンクの制約

データ・リンクの制約	**Db2 for i** の制限
DATALINK の最大長	32718
DATALINK コメントの最大長	254

表 126. データベース・マネージャーの制約

データベース・マネージャーの制約	Db2 for i の制限
リレーショナル・データベース	
スキーマの最大数	記憶域のサイズによる
リレーショナル・データベース内の表の最大数	記憶域のサイズによる
ノード・グループ内のノードの最大数	32
スキーマ	
スキーマ内のオブジェクトの最大数	約 1 000 000
表とビュー	
表の列の最大数	8000
ビュー内の列の最大数	8000
LOB のない行の最大長 (すべてのオーバーヘッドを含む)	32766
LOB がある行の最大長 (すべてのオーバーヘッドを含む)	3 758 096 383
非パーティション表内の行の最大数	4 294 967 288
データ・パーティション内の行の最大数	4 294 967 288
非パーティション表の最大サイズ	1.7 テラバイト
データ・パーティションの最大サイズ	1.7 テラバイト
1 つのパーティション表内のデータ・パーティションの最大数	256
表パーティション列の最大数	120
ビューまたはマテリアライズ照会表で参照する表の最大数	1000[138]
1 つの表またはビューの従属ビュー、マテリアライズ照会表、および索引の最大数	記憶域のサイズによる
制約	
1 つの表の制約の最大数	5000
UNIQUE 制約における列の最大数	120
UNIQUE 制約における列の結合後の最大長 (バイト数)	32767[136]
外部キーで参照される列の最大数	120
外部キーで参照される列の結合後の最大長 (バイト単位)	32767[136]
CHECK 制約の最大長 (バイト数)	ステートメント
トリガー	
表上のトリガーの最大数	300
トリガーのカスケード実行時の最大深さ	200
索引	
表の索引の最大数	約 15000
索引キーの列の最大数	120
索引キーの最大長	32767[136]
非パーティション化索引の最大サイズ	1.7 テラバイト
パーティション化索引のパーティションの最大サイズ	1.7 テラバイト
SQL	
SQL ステートメントの最大長 (バイト数)	2 097 152
1 つの SQL ステートメントで参照する表の最大数	1000[138]
SQL ステートメント内の変数および定数の最大数	32700[139]

表 126. データベース・マネージャーの制約 (続き)

データベース・マネージャーの制約	Db2 for i の制限
選択リスト内のエレメントの最大数	約 8000[140]
WHERE または HAVING 節の述部の最大数	ステートメント
1 つの GROUP BY 文節の列の最大数	GROUP BY の合計長
1 つの GROUP BY 文節の列の最大合計長	3.5 ギガバイト[141]
CUBE グループのエレメントの最大数	10
1 つの ORDER BY 文節の列の最大数	ORDER BY の合計長
1 つの ORDER BY 文節の列の最大合計長	3.5 ギガバイト[141]
階層照会の再帰レベルの最大数	250
副照会で許される最高レベル	256
挿入操作内の値の最大数	8000
単一の更新操作中の SET 節の最大数	8000
ルーチン	
プロシージャー内のパラメーターの最大数	2000[142]
関数内のパラメーターの最大数	2000[142]
表関数の戻り列の最大数	8000
ルーチンのネスト・レベルの最大数	記憶域のサイズによる
タイプ	
配列タイプの最大カーディナリティー。	2 147 483 647
アプリケーション	
プリコンパイル済みプログラム中のホスト変数宣言の最大数	記憶域のサイズによる[143]
ホスト変数値の最大長 (バイト数)	2 147 483 647
MQ メッセージ CLOB 値の最大長 (バイト数)	2M
MQ メッセージ可変長値の最大長 (バイト数)	32000
プログラムで宣言できるカーソルの最大数	記憶域のサイズによる
一度にオープンできるカーソルの最大数	記憶域のサイズによる[144]
作業単位あたりのロックされた行の最大数	500 000 000
作業単位あたりの DDL ステートメントの最大数	131 036
トランザクション内のロケーターの最大数	16 000 000 [145]
SQLDA の最大サイズ (バイト単位)	16 777 215
準備済みステートメントの最大数	記憶域のサイズによる
一度にアクティブにできるセーブポイントの最大数	記憶域のサイズによる
1 つのプロセス内で同時に割り振りできる CLI ハンドルの最大数	160 000 [146]
パッケージの最大サイズ	1008 メガバイト[147]
パスの最大長	8843
パス内のスキーマの最大数	268
パスワードの最大長	127
ヒントの最大長	32
プログラム、サービス・プログラムまたはモジュールに関連付けられるスペースの最大サイズ (バイト数)	16 777 216
診断領域の最大サイズ	90K

表 126. データベース・マネージャーの制約 (続き)

データベース・マネージャーの制約	Db2 for i の制限
配列変数の最大サイズ	4GB

付録 B. SQL ステートメントの特性

この付録では、SQL ステートメントの特性に関する情報をさまざまな使用箇所と関連付けて取り上げます。

- 1856 ページの『SQL ステートメントで許されるアクション』では、SQL ステートメントの実行が可能かどうか、対話式の準備または動的な準備が可能かどうか、ステートメントを処理するのがリクエスターなのか、サーバーなのか、プリコンパイラーなのかを示します。
- 1859 ページの『ルーチンで使用する SQL ステートメントのデータ・アクセスの種別』では、SQL ステートメントをルーチンの中で使用するために指定しなければならない SQL データ・アクセスのレベルを示します。
- 1862 ページの『分散リレーショナル・データベースの使用に関する考慮事項』では、アプリケーション・サーバーとアプリケーション・リクエスターが同じではない環境で SQL ステートメントを使用する場合の注意点を取り上げます。

---

138. 参照されるメンバー (およびパーティション) の最大数も 1000 です。DELETE および UPDATE ステートメントでは、最大数は 256 です。複合 SQL ステートメントで、参照できる表の数は、内部構造によって制限される場合があります。

139. この制限は内部構造によって制約され、ステートメントでの定数の使用方法に応じて、また使用するストリング定数またはストリング変数が非常に大きい場合は、制限値がもっと小さくなることがあります。

140. この制限は、解析対象の SQL ステートメントについて生成される内部構造のサイズに基づくものです。

141. CQE が選択ステートメントを処理した場合、限度は 32766 です。この限度は、ICU 照合順序または ALWCPYDTA(*NO) が使用されると低くなります。

142. SQL プロシージャーと SQL 関数では、最大 2000 個までパラメーターを指定できます。外部プロシージャーと外部関数のパラメーター数は、2000 を超えることはできず、その言語で許されるパラメーターの最大数によって制約されます。

143. RPG/400 および PL/I プログラムで従来のパラメーター引き渡し手法を使用する場合、制限は約 4000 になります。この制限は、プログラムで使用できるポインターの数によるものです。その他のすべての場合、制限はオペレーティング・システムの制約に基づきます。

144. 1 つのジョブ内で一度にオープンできるカーソルの最大数は、約 20 966 です。

145. SQL Server モードの 1 つのトランザクション内のロケーターの最大数は 209 000 です。

146. DRDA 接続あたりの割り振られるハンドルの最大数は 500 です。

147. DRDA パッケージの最大サイズは、QAQQINI オプションを使用して 500 メガバイトから 1 ギガバイトに増やすことができます。

SQL ステートメントで許されるアクション

ここでは、SQL ステートメントが実行可能かどうか、対話式または動的に準備できるかどうか、および、そのステートメントを処理するのがリクエスター、サーバー、またはプリコンパイラーのいずれであるかを示します。

以下の表は、特定の Db2 ステートメントが実行可能かどうか、対話式または動的に準備できるかどうか、またはそのステートメントを処理するのがリクエスター、サーバー、またはプリコンパイラーのいずれであるかを示します。文字 **Y** は *yes* (可能/該当) を意味します。

表 *127. SQL* ステートメントで許されるアクション

			処理の主体		
SQL ステートメント	実行可能	対話式または動的な準備	要求元システム	サーバー	プリコンパイラー
ALLOCATE CURSOR[5]	**Y**	**Y**	**Y**		
ALLOCATE DESCRIPTOR[4] [5]	**Y**			**Y**	
ALTER	**Y**	**Y**		**Y**	
ASSOCIATE LOCATORS[5]	**Y**	**Y**	**Y**		
BEGIN DECLARE SECTION[4] [5]					**Y**
CALL	**Y**	**Y**		**Y**	
CLOSE[4]	**Y**			**Y**	
COMMENT	**Y**	**Y**		**Y**	
COMMIT	**Y**	**Y**		**Y**	
コンパウンド (動的)	**Y**	**Y**		**Y**	
CONNECT (タイプ 1 とタイプ 2)[4] [5]	**Y**		**Y**		
CREATE	**Y**	**Y**		**Y**	
DEALLOCATE DESCRIPTOR[4] [5]	**Y**			**Y**	
DECLARE CURSOR[4]					**Y**
DECLARE GLOBAL TEMPORARY TABLE	**Y**	**Y**		**Y**	
DECLARE PROCEDURE[4] [5]					**Y**
DECLARE STATEMENT[4] [5]					**Y**
DECLARE VARIABLE[4] [5]					**Y**
DELETE	**Y**	**Y**		**Y**	
DESCRIBE[4]	**Y**			**Y**	
DESCRIBE CURSOR[4] [5]	**Y**		**Y**		
DESCRIBE INPUT[4] [5]	**Y**			**Y**	
DESCRIBE PROCEDURE[4] [5]	**Y**		**Y**		
DESCRIBE TABLE[4]	**Y**			**Y**	
DISCONNECT[4] [5]	**Y**		**Y**		
DROP	**Y**	**Y**		**Y**	
END DECLARE SECTION[4] [5]					**Y**
EXECUTE[4]	**Y**			**Y**	

表 127. SQL ステートメントで許されるアクション (続き)

SQL ステートメント	実行可能	対話式または動的な準備	処理の主体		
			要求元システム	サーバー	プリコンパイラー
EXECUTE IMMEDIATE[4]	Y			Y	
FETCH	Y			Y	
FREE LOCATOR[4] [5]	Y	Y		Y	
GET DESCRIPTOR[4] [5]	Y			Y	
GET DIAGNOSTICS[5]	Y			Y	
GRANT	Y	Y		Y	
HOLD LOCATOR[4] [5]	Y	Y		Y	
INCLUDE[4] [5]					Y
INSERT	Y	Y		Y	
LABEL	Y	Y		Y	
LOCK TABLE	Y	Y		Y	
MERGE	Y	Y		Y	
OPEN[4]	Y			Y	
PREPARE[4]	Y			Y	
REFRESH TABLE	Y	Y		Y	
RELEASE connection[4] [5]	Y		Y		
RELEASE SAVEPOINT	Y	Y		Y	
RENAME	Y	Y		Y	
REVOKE	Y	Y		Y	
ROLLBACK	Y	Y		Y	
SAVEPOINT	Y	Y		Y	
SELECT INTO [5]	Y			Y	
SET CONNECTION[4] [5]	Y		Y		
SET CURRENT DEBUG MODE	Y	Y		Y	
SET CURRENT DECFLOAT ROUNDING MODE	Y	Y		Y	
SET CURRENT DEGREE[5]	Y	Y		Y	
SET CURRENT IMPLICIT XMLPARSE OPTION	Y	Y		Y	
SET CURRENT TEMPORAL SYSTEM_TIME[5]	Y	Y		Y	
SET DESCRIPTOR[4] [5]	Y			Y	
SET ENCRYPTION PASSWORD	Y	Y		Y	
SET OPTION[4] [5]					Y
SET PATH	Y	Y		Y	
SET RESULT SETS[3] [5]	Y			Y	
SET SCHEMA	Y	Y		Y	
SET SESSION AUTHORIZATION[5]	Y	Y		Y	
SET TRANSACTION	Y	Y		Y	

表 127. SQL ステートメントで許されるアクション (続き)

			処理の主体		
SQL ステートメント	実行可能	対話式または動的な準備	要求元システム	サーバー	プリコンパイラー
SET 遷移変数[1]	**Y**			**Y**	
SET 変数	**Y**	**Y** [6]	**Y**		
SIGNAL[5]	**Y**			**Y**	
SQL 制御ステートメント[2]	**Y**			**Y**	
TRANSFER OWNERSHIP	**Y**	**Y**		**Y**	
TRUNCATE	**Y**	**Y**		**Y**	
UPDATE	**Y**	**Y**		**Y**	
VALUES[1]	**Y**			**Y**	
VALUES INTO [5]	**Y**	**Y**		**Y**	
WHENEVER[4] [5]					**Y**

注 :

1. このステートメントは、トリガーのトリガー・アクション内でのみ使用できます。
2. このステートメントは、SQL 関数、SQL プロシージャー、または SQL トリガー内でのみ使用できます。
3. このステートメントは、プロシージャー内でのみ使用できます。
4. このステートメントは、Java プログラムでは適用されません。
5. このステートメントは、REXX プログラムではサポートされません。
6. SET 変数ステートメントのターゲットはグローバル変数でなければなりません。

ルーチンで使用する SQL ステートメントのデータ・アクセスの種別

ここでは、ルーチン内で SQL ステートメントを使用する場合に指定しなければならない SQL データ・アクセスのレベルを示します。

以下の表では、最初の欄に示す SQL ステートメントを、SQL データ・アクセスのそれぞれの種別を指定した関数またはプロシージャーの中で実行できるかどうかを示しています。NO SQL と定義された関数またはプロシージャーの中で、実行可能 SQL ステートメントが検出された場合は、SQLSTATE 38001 が戻されます。その他の実行コンテキストの場合は、そのコンテキストでサポートされない SQL ステートメントがあれば、すべて SQLSTATE 38003 が戻されます。CONTAINS SQL コンテキスト内での使用を許可されていないその他の SQL ステートメントの場合は、SQLSTATE 38004 が戻され、READS SQL DATA コンテキストの場合は SQLSTATE 38002 が戻されます。 SQL 関数または SQL プロシージャーの作成時に、SQL データ・アクセスの種別に一致しないステートメントが検出されると、SQLSTATE 42895 が返されます。

表 *128. SQL* ステートメントと *SQL* データ・アクセスの種別

SQL ステートメント	**NO SQL**	**CONTAINS SQL**	**READS SQL DATA**	**MODIFIES SQL DATA**
ALLOCATE CURSOR			Y	Y
ALLOCATE DESCRIPTOR			Y	Y
ALTER ...				Y
ASSOCIATE LOCATORS			Y	Y
BEGIN DECLARE SECTION	Y[1]	Y	Y	Y
CALL		Y	Y	Y
CLOSE			Y	Y
COMMENT				Y
COMMIT[3]		Y	Y	Y
コンパウンド (動的)		Y[2]	Y[2]	Y
CONNECT (タイプ 1 およびタイプ 2)[3]				
CREATE ...				Y
DEALLOCATE DESCRIPTOR			Y	Y
DECLARE CURSOR	Y[1]	Y	Y	Y
DECLARE GLOBAL TEMPORARY TABLE				Y
DECLARE PROCEDURE	Y[1]	Y	Y	Y
DECLARE STATEMENT	Y[1]	Y	Y	Y
DECLARE VARIABLE	Y[1]	Y	Y	Y
DELETE				Y
DESCRIBE			Y	Y
DESCRIBE CURSOR			Y	Y
DESCRIBE INPUT			Y	Y

表 128. SQL ステートメントと SQL データ・アクセスの種別 (続き)

SQL ステートメント	NO SQL	CONTAINS SQL	READS SQL DATA	MODIFIES SQL DATA
DESCRIBE PROCEDURE			Y	Y
DESCRIBE TABLE			Y	Y
DISCONNECT[3]				
DROP ...				Y
END DECLARE SECTION	Y[1]	Y	Y	Y
EXECUTE		Y[2]	Y[2]	Y
EXECUTE IMMEDIATE		Y[2]	Y[2]	Y
FETCH			Y	Y
FREE LOCATOR		Y	Y	Y
GET DESCRIPTOR			Y	Y
GET DIAGNOSTICS		Y	Y	Y
GRANT ...				Y
HOLD LOCATOR		Y	Y	Y
INCLUDE	Y[1]	Y	Y	Y
INSERT				Y
LABEL				Y
LOCK TABLE		Y	Y	Y
MERGE				Y
OPEN			Y	Y
PREPARE		Y	Y	Y
REFRESH TABLE				Y
RELEASE CONNECTION[3]				
RELEASE SAVEPOINT				Y
RENAME				Y
REVOKE ...				Y
ROLLBACK[3]		Y	Y	Y
ROLLBACK TO SAVEPOINT				Y
SAVEPOINT				Y
SELECT INTO			Y	Y
SET CONNECTION[3]				
SET CURRENT DEBUG MODE			Y	Y
SET CURRENT DECFLOAT ROUNDING MODE		Y	Y	Y
SET CURRENT DEGREE			Y	Y
SET CURRENT IMPLICIT XMLPARSE OPTION		Y	Y	Y
SET CURRENT TEMPORAL SYSTEM_TIME		Y	Y	Y

表 128. SQL ステートメントと SQL データ・アクセスの種別 (続き)

SQL ステートメント	NO SQL	CONTAINS SQL	READS SQL DATA	MODIFIES SQL DATA
SET DESCRIPTOR			Y	Y
SET ENCRYPTION PASSWORD		Y	Y	Y
SET OPTION	Y[1]	Y	Y	Y
SET PATH		Y	Y	Y
SET RESULT SETS		Y	Y	Y
SET SCHEMA			Y	Y
SET SESSION AUTHORIZATION			Y	Y
SET TRANSACTION		Y	Y	Y
SET 変数		Y	Y	Y
SIGNAL		Y	Y	Y
TRANSFER OWNERSHIP				Y
TRUNCATE				Y
UPDATE				Y
VALUES				
VALUES INTO			Y	Y
WHENEVER	Y[1]	Y	Y	Y

注:

1. NO SQL オプションは SQL ステートメントを指定できないことを暗黙に示しますが、非実行ステートメントを制限するものではありません。
2. 実行されるステートメントによって決まります。 EXECUTE ステートメントに指定されるステートメントまたは複合 (動的) ステートメント内部で実行されるステートメントは、有効な特定の SQL アクセス・レベルのコンテキストで許可されるステートメントでなければなりません。 例えば、有効な SQL アクセス・レベルが READS SQL DATA の場合、ステートメントは、INSERT、UPDATE、または DELETE 以外でなければなりません。
3. 接続管理ステートメントおよびトランザクション・ステートメントは、リモート・サーバーで実行しているプロシージャーでは許可されません。 COMMIT および ROLLBACK は、ATOMIC SQL プロシージャー内で使用することはできません。

分散リレーショナル・データベースの使用に関する考慮事項

このセクションには、アプリケーション・リクエスターとは異なるプロダクトのアプリケーション・サーバーを使用するアプリケーションを開発するときに役立つ情報を収めてあります。

Db2 の製品はすべて、IBM SQL の拡張機能をサポートしています。このような拡張機能には製品固有の機能もありますが、既に複数の製品に共通する機能となっているものや、まだ一般に使用できませんがサポートが計画されているものも多くあります。

これらの大部分では、ステートメントおよび文節の一部をサポートしていないデータベース・マネージャーのアプリケーション・リクエスターを介して、アプリケーションが実行されている場合でも、現行サーバーのデータベース・マネージャーでサポートされているステートメントおよび文節であれば、そのアプリケーションで使用することができます。この一般的規則に対する制約事項は、アプリケーション・リクエスターによって識別されます。

- Db2 for z/OS Application Server アプリケーション・リクエスターについては、表 129 を参照してください。
- Db2 for i Application Server アプリケーション・リクエスターについては、1863 ページの表 130 を参照してください。
- Db2 LUW アプリケーション・リクエスターについては、 1863 ページの表 131 を参照してください。

表の中の 'R' は、この SQL 機能が指定された環境でサポートされていないことを示しています。同じ行のすべての欄に 'R' があるのは、その機能を使用できるのが、現行サーバーとリクエスターが同じプロダクトである場合だけに限られることを意味しています。それらが同じプロダクトではない場合、ステートメントはアプリケーション・リクエスターによってブロックされて、アプリケーション・サーバーで処理されません。

表 129. Db2 for z/OS アプリケーション・リクエスター

SQL のステートメントまたは関数	Db2 for z/OS Application Server	Db2 for i Application Server	Db2 LUW Application Server
COMMIT HOLD	R	R	R
DECLARE STATEMENT			
DECLARE TABLE			
DECLARE VARIABLE			
DESCRIBE TABLE			R
DESCRIBE、USING 文節付き			R
DISCONNECT	R	R	R
ROWID データ・タイプ			R
DATALINK データ・タイプ	R	R	R
BINARY データ・タイプと VARBINARY データ・タイプ			R
言語特定の付録にホスト宣言の記載なし		[148]	[148]
PREPARE、USING 文節付き			R

表 129. Db2 for z/OS アプリケーション・リクエスター (続き)

SQL のステートメントまたは関数	Db2 for z/OS Application Server	Db2 for i Application Server	Db2 LUW Application Server
ROLLBACK HOLD	R	R	R
SET CURRENT PACKAGESET			
SET 変数		R	R
SET TRANSACTION	R	R	R
スクロール可能カーソル・ステートメント	R	R	R
UPDATE カーソル - FOR UPDATE 文節が指定されていない			

表 130. Db2 for i アプリケーション・リクエスター

SQL のステートメントまたは関数	Db2 for z/OS Application Server	Db2 for i Application Server	Db2 LUW Application Server
COMMIT HOLD	R		R
DECLARE STATEMENT			
DECLARE TABLE			
DECLARE VARIABLE			
DESCRIBE TABLE			R
DESCRIBE、USING 文節付き			R
DISCONNECT			
ホスト変数 - コロンは任意指定	R	R	R
ROWID データ・タイプ			R
DATALINK データ・タイプ	R		R
BINARY データ・タイプと VARBINARY データ・タイプ			R
言語特定の付録にホスト宣言の記載なし	[148]		[148]
PREPARE、USING 文節付き			R
ROLLBACK HOLD	R		R
SET CURRENT PACKAGESET	R	R	R
SET 変数	R	R	R
SET TRANSACTION	R		R
スクロール可能カーソル・ステートメント	R		R
UPDATE カーソル - FOR UPDATE 文節が指定されていない	R		

表 131. Db2 LUW アプリケーション・リクエスター

SQL のステートメントまたは関数	Db2 for z/OS Application Server	Db2 for i Application Server	Db2 LUW Application Server
COMMIT HOLD	R	R	R
DECLARE STATEMENT	R	R	R

148. このステートメントがサポートされるのは、アプリケーション・リクエスターがこのステートメントを認識する場合です。

表 131. Db2 LUW アプリケーション・リクエスター (続き)

SQL のステートメントまたは関数	Db2 for z/OS Application Server	Db2 for i Application Server	Db2 LUW Application Server
DECLARE TABLE	R	R	R
DECLARE VARIABLE	R	R	R
DESCRIBE TABLE	R	R	R
DESCRIBE、USING 文節付き	R	R	R
DISCONNECT			
ホスト変数 - コロンは任意指定	R	R	R
ROWID データ・タイプ	[149]	[149]	R
DATALINK データ・タイプ	R	R	R
BINARY データ・タイプと VARBINARY データ・タイプ	R	R	R
言語特定の付録にホスト宣言の記載なし	[148]	[148]	
PREPARE、USING 文節付き	R	R	R
ROLLBACK HOLD	R	R	R
SET CURRENT PACKAGESET			
SET 変数	R	R	R
SET TRANSACTION	R	R	R
スクロール可能カーソル・ステートメント	R	R	R
UPDATE カーソル - FOR UPDATE 文節が指定されていない	R		

149. Db2 LUW アプリケーション・リクエスターは、互換性のある VARCHAR(40) FOR BIT DATA データ・タイプを使用して、アプリケーション・サーバーで ROWID データ・タイプを処理します。

CONNECT (タイプ 1) と CONNECT (タイプ 2) の相違点

CONNECT ステートメントには 2 つのタイプがあります。

それらは、構文は同じですが、意味が異なります。

- CONNECT (タイプ 1) は、リモート作業単位に対して使用されます。 50 ページの『リモート作業単位』を参照してください。
- CONNECT (タイプ 2) は、分散作業単位に対して使用されます。 1059 ページの『CONNECT (タイプ 2)』を参照してください。

次の表は、CONNECT (タイプ 1) と CONNECT (タイプ 2) の規則の相違点を要約しています。

表 *132. CONNECT (タイプ 1)* と *CONNECT (タイプ 2)* の相違点

タイプ **1** の規則	タイプ **2** の規則
CONNECT ステートメントは、活動化グループが接続可能状態である場合にのみ実行が可能です。同じ作業単位内では、CONNECT ステートメントを複数実行することはできません。	同じ作業単位内で複数の CONNECT ステートメントを実行することができます。接続可能状態に関する規則はありません。
該当のサーバー名がローカル・ディレクトリーにリストされていないことによって、CONNECT ステートメントが失敗した場合、その活動化グループの接続状態は変わりません。 該当の活動化グループが接続可能状態でないことによって、CONNECT ステートメントが失敗した場合、その活動化グループの SQL 接続状態は変わりません。 上記以外の理由で CONNECT ステートメントが失敗した場合、その活動化グループは未接続状態になります。	CONNECT ステートメントが失敗すると、現行 SQL 接続は変わらず、それ以後の SQL ステートメントはいずれも現行サーバーによって実行されます。
CONNECT は、その活動化グループの既存の接続をすべて終了させます。 したがって、CONNECT はまた、その活動化グループのオープン・カーソルをいずれもクローズします。	CONNECT は、接続の終了やカーソルのクローズを行いません。
現行サーバーに対する CONNECT は、他の CONNECT (タイプ 1) ステートメントと同じように実行されます。	現行サーバーに対する CONNECT はエラーの原因となります。

適用される CONNECT の規則の判別

プログラムによって行われる CONNECT のタイプの指定には、プログラム準備オプションが使用されます。プログラム準備オプションは、CRTSQLxxx コマンドの RDBCNNMTH パラメーターを使用して指定します。

リモート作業単位のみをサポートするサーバーへの接続

リモート作業単位のみをサポートするアプリケーション・サーバーへの CONNECT (タイプ 2) 接続は、読み取り専用の接続になる場合があります。

リモート作業単位のみをサポートするアプリケーション・サーバーに対して CONNECT (タイプ 2) を実行した場合:

- その接続の時点で更新を許す休止状態の接続が存在する場合、その接続では読み取り専用の操作が可能です。この場合、その接続では更新は許されません。
- これ以外の場合、その接続で更新が可能です。

分散作業単位をサポートするアプリケーション・サーバーに対して CONNECT (Type 2) を実行した場合:

- リモート作業単位のみをサポートするアプリケーション・サーバーに対して更新を許可する休止状態の接続がある場合には、その接続では読み取り専用操作が可能です。この場合、その休止状態の接続が終了するとただちにその接続での更新が可能になります。
- これ以外の場合、その接続で更新が可能です。

150. リモート作業単位のみをサポートするアプリケーション・サーバーの例としては、ネイティブ TCP/IP 用の初期 DRDA サポートを使用する Db2 for i があります。

付録 C. SQLCA (SQL 連絡域)

SQLCA は一組の変数で、各 SQL ステートメントの実行の終了時に更新されることがあります。 SQLCA が適用されない Java での場合を除いて、実行可能な SQL ステートメントが入っているプログラムは、1 つの SQLCA を用意することがありますが、それを複数用意することはありません (ただし、代わりに独立型の SQLCODE または独立型の SQLSTATE 変数を使用する場合を除く)。

SQLCA を使用する代わりに、すべての言語で GET DIAGNOSTICS ステートメントを使用して、戻りコードおよび直前の SQL ステートメントに関する他の情報を戻すことができます。詳しくは、 1489 ページの『GET DIAGNOSTICS』を参照してください。

Java、RPG、または REXX を除くすべてのホスト言語では、SQL INCLUDE ステートメントを使用して SQLCA を宣言することができます。 REXX プロシージャーでの SQLCA の使用方法については、「組み込み SQL プログラミング」トピック集を参照してください。 Java でエラーおよび警告についての情報にアクセスする方法については、「IBM Developer Kit for Java」トピック集を参照してください。

C、COBOL、および PL/I では、このストレージ域の名前は SQLCA でなければなりません。すべての SQL ステートメントは、必ず SQLCA の宣言の有効範囲内になければなりません。

プログラムで独立型の SQLCODE または SQLSTATE を指定している場合は、SQLCA を組み込んではなりません。詳しくは、 876 ページの『SQL 診断情報』を参照してください。

独立型の SQLCODE と独立型の SQLSTATE は、Java または REXX 言語では指定してはなりません。

フィールドの説明

以下の表に示している名前は、SQL の INCLUDE ステートメントによって指定されている名前です。

大部分については、C (および C++)、COBOL、および PL/I では同じ名前を使用します。RPG/400 では、名前が 6 文字までに制限されているため、RPG では異なる名前を使用します。 ILE RPG では、ロング・ネームと 6 文字の短い名前との両方がサポートされています。 PL/I の名前と COBOL の名前が異なっている 1 つの事例に注意してください。

表 133. SQL の INCLUDE ステートメントによって組み込まれる名前

C の名前 COBOL の名前 PL/I の名前	ILE RPG の名前 RPG/400 の名前	フィールド データ・タイプ	フィールドの値
SQLCAID sqlcaid	SQLCAID SQLAID	CHAR(8)	ストレージ・ダンプ用の「目印」として、「SQLCA」が入ります。
SQLCABC sqlcabc	SQLCABC SQLABC	INTEGER	SQLCA の長さ 136 が入ります。
SQLCODE sqlcode	SQLCODE SQLCOD	INTEGER	SQL 戻りコードが入ります。 コード 意味 **0** 正常に実行された。ただし、SQLWARN 標識がセットされている場合がある。 正の値 正常に実行された (ただし、警告条件があった)。 負の値 エラー状態。
SQLERRML[1] sqlerrml	SQLERRML SQLERL	SMALLINT	SQLERRMC の長さ標識 (0 から 70 までの範囲)。0 は SQLERRMC の値が無関係であることを示します。
SQLERRMC[1] sqlerrmc	SQLERRMC SQLERM	CHAR(70)	SQLCODE に関連するメッセージ置換テキストが入ります。 CONNECT および SET CONNECTION の場合、この SQLERRMC フィールドにはその接続に関する情報が入ります。置換テキストについての説明は、 1872 ページの表 136 を参照してください。
SQLERRP sqlerrp	SQLERRP SQLERP	CHAR(8)	エラーまたは警告を戻したプロダクトおよびモジュールの名前が入ります。最初の 3 文字は、次のようにプロダクトを識別します。 • ARI (Db2 UDB サーバー (VM および VSE 版) の場合) • DSN (Db2 for z/OS の場合) • QSQ (Db2 for i の場合) • 他のすべての Db2 プロダクトの場合は SQL 詳細については、 1053 ページの『CONNECT (タイプ 1)』、または 1059 ページの『CONNECT (タイプ 2)』を参照してください。
SQLERRD sqlerrd	SQLERRD SQLERR[2]	アレイ	診断情報が提供される 6 つの INTEGER 変数が入ります。診断情報の説明については、 1870 ページの表 135 を参照してください。
SQLWARN sqlwarn	SQLWARN SQLWRN[3]	CHAR(11)	11 個の CHAR(1) 警告標識です。それぞれにブランク、「W」、または「N」のいずれかが入ります。
SQLSTATE sqlstate	SQLSTATE SQLSTT	CHAR(5)	直前に実行された SQL ステートメントの結果を示す戻りコードです。

表 133. SQL の INCLUDE ステートメントによって組み込まれる名前 (続き)

C の名前 COBOL の名前 PL/I の名前	ILE RPG の名前 RPG/400 の名前	フィールド データ・タイプ	フィールドの値

注:

[1] COBOL では、SQLERRM には SQLERRML と SQLERRMC が含まれています。 PL/I では、可変長ストリング SQLERRM は、SQLERRMC にプレフィックス SQLERRML が付いたものと同じです。

[2] RPG/400 では、SQLERR は 24 文字 (配列ではなく) として定義されます。これらの文字は、SQLER1 から SQLER6 までのフィールドによって再定義されます。各フィールドは、フルワード 2 進数です。 ILE RPG の場合には、SQLERR は配列としても再定義されています。この名前の配列は SQLERRD です。

[3] RPG/400 では、SQLWRN は 11 文字 (配列ではなく) として定義されます。これらの文字は、SQLWN0 から SQLWNA までのフィールドによって再定義されます。各フィールドは、フルワード 2 進数です。 ILE RPG の場合には、SQLWRN は配列としても再定義されています。この名前の配列は SQLWARN です。

表 134. SQLWARN 診断情報

C の名前 COBOL の名前 PL/I の名前	ILE RPG の名前 RPG/400 の名前	フィールドの値
SQLWARN0 sqlwarn[0]	SQLWARN(1) SQLWN0	「W」または「N」が入っている他の標識が 1 つでもあれば、「W」が入ります。他の標識がすべてブランクの場合は、ブランクが入ります。
SQLWARN1 sqlwarn[1]	SQLWARN(2) SQLWN1	ホスト変数への割り当て時にストリング列の値が切り捨てられた場合は「W」になります。 *NOCNULRQD が CRTSQLCI または CRTSQLCPPI コマンド (または SET OPTION ステートメントの CNULRQD(*NO)) で指定され、ストリング列の値が C NUL 終了ホスト変数に割り当てられ、しかもそのホスト変数が結果を入れるには十分な大きさであるが、NUL 終了文字を入れるには十分な大きさでない場合は、ここに 'N' が入ります。
SQLWARN2 sqlwarn[2]	SQLWARN(3) SQLWN2	関数の引数から NULL 値が削除された場合は、「W」が入ります。ただし、MIN 関数または MAX 関数の場合は、必ずしも「W」に設定されるわけではありません。それらの関数の結果は、NULL 値の削除に左右されないからです。
SQLWARN3 sqlwarn[3]	SQLWARN(4) SQLWN3	列の数がホスト変数の数よりも多い場合には、「W」が入ります。
SQLWARN4 sqlwarn[4]	SQLWARN(5) SQLWN4	準備された UPDATE または DELETE ステートメントに WHERE 文節が指定されていない場合に、「W」が入ります。
SQLWARN5 sqlwarn[5]	SQLWARN(6) SQLWN5	OPEN ステートメントの後のカーソルの能力を示すための文字値 1 (読み取り専用)、2 (読み取りおよび削除)、または 4 (読み取り、削除、および更新) が入ります。
SQLWARN6 sqlwarn[6]	SQLWARN(7) SQLWN6	日付演算の結果、月の終わりの調整となった場合に、'W' が入ります。
SQLWARN7 sqlwarn[7]	SQLWARN(8) SQLWN7	予約済み
SQLWARN8 sqlwarn[8]	SQLWARN(9) SQLWN8	文字変換の結果に置換文字が含まれている場合に、'W' が入ります。

表 134. SQLWARN 診断情報 (続き)

C の名前 COBOL の名前 PL/I の名前	ILE RPG の名前 RPG/400 の名前	フィールドの値
SQLWARN9 sqlwarn[9]	SQLWARN(10) SQLWN9	予約済み
SQLWARNA sqlwarn[10]	SQLWARN(11) SQLWNA	予約済み

表 135. SQLERRD の診断情報

C の名前 COBOL の名前 PL/I の名前	ILE RPG の名前 RPG/400 の名前	フィールドの値
SQLERRD(1) sqlerrd[0]	SQLERRD(1) SQLER1	SQLCODE が 0 より小さい場合に、CPF エスケープ・メッセージの最後の 4 文字が入ります。例えば、メッセージが CPF5715 であれば、X'F5F7F1F5' が SQLERRD(1) に入れられます。[1] プロシージャーの呼び出しの場合、RETURN ステートメントで指定された戻り状況値が入ります。 RETURN ステートメントで戻り状況値が指定されていない場合、またはプロシージャーが外部プロシージャーの場合は、次のようになります。 • CALL ステートメントが成功した場合、0 が戻されます。 • CALL ステートメントが成功しなかった場合は、-1 が返されます。
SQLERRD(2) sqlerrd[1]	SQLERRD(2) SQLER2	SQL コードが 0 より小さい場合に、CPD 診断メッセージの最後の 4 文字が入ります。[1] CALL ステートメントの場合は、SQLERRD(2) には結果セットの数が入ります。 OPEN ステートメントでは、カーソルが変更に反応しない場合、SQLERRD(2) には結果セットに含まれる実際の行数が入ります。カーソルが変更に反応する場合、SQLERRD(2) には結果セットに含まれる行数の推定値が入ります。
SQLERRD(3) sqlerrd[2]	SQLERRD(3) SQLER3	状況ステートメントについての CONNECT の場合は、SQLERRD(3) には接続状況に関する情報が入ります。詳しくは、 1059 ページの『CONNECT (タイプ 2)』を参照してください。 INSERT、MERGE、UPDATE、REFRESH、DELETE の場合は、影響を受ける行の数が入ります。 TRUNCATE ステートメントの場合、値は -1 になる。 FETCH ステートメントの場合は、SQLERRD(3) には取り出された行の数が入ります。 PREPARE ステートメントの場合は、選択された行の見積数が入ります。行数が 2 147 483 647 より多い場合は、2 147 483 647 が戻されます。

表 135. SQLERRD の診断情報 (続き)

C の名前 COBOL の名前 PL/I の名前	ILE RPG の名前 RPG/400 の名前	フィールドの値
SQLERRD(4) sqlerrd[3]	SQLERRD(4) SQLER4	PREPARE ステートメントの場合は、すべての実行で必要なリソースの相対的な数の見積もりが入ります。この数は、索引、ファイル・サイズ、または CPU モデルなどの現在の可用性によって異なります。これは、Db2 for i Query Optimizer によって選択されたアクセス・プランの見積コストです。 CONNECT および SET CONNECTION ステートメントの場合は、SQLERRD(4) には会話のタイプが入り、コミット可能な更新を行うことができるかどうかが示されます。詳しくは、 1059 ページの『CONNECT (タイプ 2)』を参照してください。 CALL ステートメントの場合、SQLERRD(4) には、プロシージャーが正常に実行されなかった原因となった、エラーのメッセージ・キーが入ります。QMHRTVPM API は、そのメッセージ・キーのメッセージ記述を戻すために使用できます。 DELETE、INSERT、MERGE、UPDATE の各ステートメントでトリガー・エラーが発生すると、トリガー・プログラムから通知されたエラーのメッセージ・キーが SQLERRD(4) に入ります。QMHRTVPM API は、そのメッセージ・キーのメッセージ記述を戻すために使用できます。 FETCH ステートメントの場合は、結果行に LOB が含まれなければ、SQLERRD(4) には取り出された行の長さが入ります。 OPEN ステートメントの場合は、結果行に LOB が含まれなければ、SQLERRD(4) には結果行の長さが入ります。
SQLERRD(5) sqlerrd[4]	SQLERRD(5) SQLER5	CONNECT または SET CONNECTION ステートメントでは、SQLERRD(5) には次の値が入ります。 • 接続が未接続である場合には -1 • 接続がローカルである場合には 0 • 接続がリモートである場合には 1 DELETE、INSERT、MERGE、UPDATE の各ステートメントの場合は、参照制約とトリガーによって影響を受けた行の数が入ります。 EXECUTE IMMEDIATE または PREPARE ステートメントの場合、構文エラーの位置が入っていることがあります。 複数行の FETCH ステートメントでは、現在表にある最後の行が取り出された場合は、SQLERRD(5) には +100 が入ります。 PREPARE ステートメントの場合、SQLERRD(5) には、準備済みステートメント内のパラメーター・マーカーの数が入ります。

表 135. SQLERRD の診断情報 (続き)

C の名前 COBOL の名前 PL/I の名前	ILE RPG の名前 RPG/400 の名前	フィールドの値
SQLERRD(6) sqlerrd[5]	SQLERRD(6) SQLER6	SQLCODE が 0 の場合に、SQL 完了メッセージの ID が入ります。 それ以外の場合、この値は未定義になります。

注記:

[1] SQLERRD(1) および SQLERRD(2) が設定されるのは、設定の条件に該当する場合、および現行サーバーが Db2 for i である場合のみに限られます。

表 136. CONNECT および SET CONNECTION の場合の SQLERRMC の置換テキスト

説明	データ・タイプ
リレーショナル・データベース名	CHAR(18)
プロダクト識別 (SQLERRP と同じ)	CHAR(8)
サーバー・ジョブのユーザー ID	CHAR(10)
接続方式 (*DUW または *RUW)	CHAR(10)
DDM サーバー・クラス名	CHAR(10)

QAS Db2 for i

QDB2 Db2 for z/OS

QDB2/6000
Db2 for AIX

QDB2/HPUX
Db2 for HP-UX**

QDB2/LINUX
Db2 for Linux

QDB2/NT
Db2 for Windows** NT/2000/XP

QDB2/SUN
Db2 for SUN** Solaris**

QSQLDS/VM
Db2 サーバー (VM 版)

QSQLDS/VSE
Db2 サーバー (VSE 版)

接続タイプ (SQLERRD(4) と同じ)	SMALLINT

INCLUDE SQLCA の宣言

このセクションでは、C および C++、COBOL、PL/I、RPG/400、および ILE RPG について、同等の INCLUDE SQLCA 宣言を示します。

C および **C++** の場合、INCLUDE SQLCA 宣言は以下のステートメントと同等です。

```
#ifndef    SQLCODE
struct     sqlca
{
           unsigned char  sqlcaid[8];
           long           sqlcabc;
           long           sqlcode;
           short          sqlerrml;
           unsigned char  sqlerrmc[70];
           unsigned char  sqlerrp[8];
           long           sqlerrd[6];
           unsigned char  sqlwarn[11];
           unsigned char  sqlstate[5];
};
#define      SQLCODE   sqlca.sqlcode
#define      SQLWARN0  sqlca.sqlwarn[0]
#define      SQLWARN1  sqlca.sqlwarn[1]
#define      SQLWARN2  sqlca.sqlwarn[2]
#define      SQLWARN3  sqlca.sqlwarn[3]
#define      SQLWARN4  sqlca.sqlwarn[4]
#define      SQLWARN5  sqlca.sqlwarn[5]
#define      SQLWARN6  sqlca.sqlwarn[6]
#define      SQLWARN7  sqlca.sqlwarn[7]
#define      SQLWARN8  sqlca.sqlwarn[8]
#define      SQLWARN9  sqlca.sqlwarn[9]
#define      SQLWARNA  sqlca.sqlwarn[10]
#define      SQLSTATE  sqlca.sqlstate
#endif
struct sqlca sqlca;
```

COBOL の場合、INCLUDE SQLCA の宣言は、以下のステートメントと同等です。

```
01 SQLCA.
   05 SQLCAID      PIC X(8).
   05 SQLCABC      PIC S9(9) BINARY.
   05 SQLCODE      PIC S9(9) BINARY.
   05 SQLERRM.
      49 SQLERRML  PIC S9(4) BINARY.
      49 SQLERRMC  PIC X(70).
   05 SQLERRP      PIC X(8).
   05 SQLERRD      OCCURS 6 TIMES
                   PIC S9(9) BINARY.
   05 SQLWARN.
      10 SQLWARN0  PIC X(1).
      10 SQLWARN1  PIC X(1).
      10 SQLWARN2  PIC X(1).
      10 SQLWARN3  PIC X(1).
      10 SQLWARN4  PIC X(1).
      10 SQLWARN5  PIC X(1).
      10 SQLWARN6  PIC X(1).
      10 SQLWARN7  PIC X(1).
      10 SQLWARN8  PIC X(1).
      10 SQLWARN9  PIC X(1).
      10 SQLWARNA  PIC X(1).
   05 SQLSTATE     PIC X(5).
```

注: COBOL では、作業記憶域セクション (WORKING STORAGE SECTION) の外側で INCLUDE SQLCA を指定してはなりません。

PL/I の場合、INCLUDE SQLCA の宣言は、以下のステートメントと同等です。

```
DCL 1 SQLCA,
      2 SQLCAID      CHAR(8),
      2 SQLCABC      BIN FIXED(31),
      2 SQLCODE      BIN FIXED(31),
      2 SQLERRM      CHAR(70) VAR,
      2 SQLERRP      CHAR(8),
      2 SQLERRD(6)   BIN FIXED(31),
      2 SQLWARN,
        3 SQLWARN0   CHAR(1),
        3 SQLWARN1   CHAR(1),
        3 SQLWARN2   CHAR(1),
        3 SQLWARN3   CHAR(1),
        3 SQLWARN4   CHAR(1),
        3 SQLWARN5   CHAR(1),
        3 SQLWARN6   CHAR(1),
        3 SQLWARN7   CHAR(1),
        3 SQLWARN8   CHAR(1),
        3 SQLWARN9   CHAR(1),
        3 SQLWARNA   CHAR(1),
      2 SQLSTATE     CHAR(5);
```

RPG/400 の場合、SQLCA の宣言は、以下のステートメントと同等です。

```
ISQLCA          DS
I                                       1   8 SQLAID            SQL
I                                    B  9  120SQLABC            SQL
I                                    B 13  160SQLCOD            SQL
I                                    B 17  180SQLERL            SQL
I                                      19  88 SQLERM            SQL
I                                      89  96 SQLERP            SQL
I                                      97 120 SQLERR            SQL
I                                    B 97 1000SQLER1            SQL
I                                    B101 1040SQLER2            SQL
I                                    B105 1080SQLER3            SQL
I                                    B109 1120SQLER4            SQL
I                                    B113 1160SQLER5            SQL
I                                    B117 1200SQLER6            SQL
I                                      121 131 SQLWRN           SQL
I                                      121 121 SQLWN0           SQL
I                                      122 122 SQLWN1           SQL
I                                      123 123 SQLWN2           SQL
I                                      124 124 SQLWN3           SQL
I                                      125 125 SQLWN4           SQL
I                                      126 126 SQLWN5           SQL
I                                      127 127 SQLWN6           SQL
I                                      128 128 SQLWN7           SQL
I                                      129 129 SQLWN8           SQL
I                                      130 130 SQLWN9           SQL
I                                      131 131 SQLWNA           SQL
I                                      132 136 SQLSTT           SQL
```

ILE RPG の場合、SQLCA の宣言は、以下のステートメントと同等です。

```
 // SQL COMMUNICATION AREA
DCL-DS SQLCA;
  SQLCAID CHAR(8) INZ(X'0000000000000000');
   SQLAID CHAR(8) OVERLAY(SQLCAID);
  SQLCABC INT(10);
   SQLABC BINDEC(9) OVERLAY(SQLCABC);
  SQLCODE INT(10);
   SQLCOD BINDEC(9) OVERLAY(SQLCODE);
  SQLERRML INT(5);
```

```
   SQLERL BINDEC(4) OVERLAY(SQLERRML);
  SQLERRMC CHAR(70);
   SQLERM CHAR(70) OVERLAY(SQLERRMC);
  SQLERRP CHAR(8);
   SQLERP CHAR(8) OVERLAY(SQLERRP);
  SQLERR CHAR(24);
   SQLER1 BINDEC(9) OVERLAY(SQLERR:*NEXT);
   SQLER2 BINDEC(9) OVERLAY(SQLERR:*NEXT);
   SQLER3 BINDEC(9) OVERLAY(SQLERR:*NEXT);
   SQLER4 BINDEC(9) OVERLAY(SQLERR:*NEXT);
   SQLER5 BINDEC(9) OVERLAY(SQLERR:*NEXT);
   SQLER6 BINDEC(9) OVERLAY(SQLERR:*NEXT);
   SQLERRD INT(10) DIM(6) OVERLAY(SQLERR);
  SQLWRN CHAR(11);
   SQLWN0 CHAR(1) OVERLAY(SQLWRN:*NEXT);
   SQLWN1 CHAR(1) OVERLAY(SQLWRN:*NEXT);
   SQLWN2 CHAR(1) OVERLAY(SQLWRN:*NEXT);
   SQLWN3 CHAR(1) OVERLAY(SQLWRN:*NEXT);
   SQLWN4 CHAR(1) OVERLAY(SQLWRN:*NEXT);
   SQLWN5 CHAR(1) OVERLAY(SQLWRN:*NEXT);
   SQLWN6 CHAR(1) OVERLAY(SQLWRN:*NEXT);
   SQLWN7 CHAR(1) OVERLAY(SQLWRN:*NEXT);
   SQLWN8 CHAR(1) OVERLAY(SQLWRN:*NEXT);
   SQLWN9 CHAR(1) OVERLAY(SQLWRN:*NEXT);
   SQLWNA CHAR(1) OVERLAY(SQLWRN:*NEXT);
   SQLWARN CHAR(1) DIM(11) OVERLAY(SQLWRN);
  SQLSTATE CHAR(5);
   SQLSTT CHAR(5) OVERLAY(SQLSTATE);
END-DS SQLCA;
```

付録 D. SQLDA (SQL 記述子域)

SQLDA は、SQL DESCRIBE ステートメントの実行に使用される変数の集まりであり、 PREPARE、 OPEN、 CALL、 FETCH、および EXECUTE ステートメントで、必要に応じて使用することができます。

SQLDA は、DESCRIBE または PREPARE ステートメントで使用し、ストレージ域[151]のアドレスによって変更し、その後で FETCH ステートメントで再度使用することができます。

SQLDA はすべての言語でサポートされますが、事前定義宣言が用意されているのは、C (および C++)、 COBOL、ILE RPG、PL/I、および REXX の場合だけです。 REXX の場合、SQLDA は他の言語の場合と多少異なります。REXX での SQLDA の使用方法について詳しくは、「組み込み SQL プログラミング」トピック集を参照してください。

SQLDA の情報の意味は、その用途によって異なります。

- SQLDA を DESCRIBE または PREPARE ステートメントで使用すると、SQLDA によって準備済み選択ステートメントに関する情報がアプリケーション・プログラムに提供されます。 結果表の各列は、SQLVAR オカレンスまたは関連 SQLVAR オカレンスのセット内に記述されます。
- OPEN、EXECUTE、CALL、および FETCH で使用すると、SQLDA によって、入力データまたは出力データ用のストレージ域に関する情報がデータベース・マネージャーに提供されます。各ストレージ域は SQLVAR に記述されます。
  - CALL 以外のステートメントの OPEN および EXECUTE の場合、それぞれの SQLVAR オカレンスまたは関連した SQLVAR オカレンスのセットは、以前に準備された関連 SQL ステートメントにパラメーター・マーカーの代わりになる入力値を含めるのに使用するストレージ域を記述します。
  - FETCH の場合、それぞれの SQLVAR オカレンスまたは関連した SQLVAR オカレンスのセットは、結果表の行からの出力値を含めるのに使用するストレージ域を記述します。
  - 準備済み CALL ステートメントの CALL および EXECUTE の場合、それぞれの SQLVAR オカレンスまたは関連した SQLVAR オカレンスのセットで、プロシージャー用の引数リストの引数に対応する入力値または出力値 (またはその両方) を含めるのに使用するストレージ域を記述します。

SQLDA は、ヘッダー構造の中の 4 つの変数と、それに続く基本 *SQLVAR* の任意の数のオカレンスから構成されます。 SQLDA が LOB または特殊タイプを記述する場合、基本 SQLVAR の後に拡張 *SQLVAR* のオカレンスの同じ数が続きます。

基本 **SQLVAR** 項目

基本 SQLVAR 項目は、常に存在する項目です。この項目のフィールドに

151. ストレージ域は、プログラムで定義された変数 (ホスト変数のこともある) 用のストレージの場合と、アプリケーションによって明示的に割り振られたストレージの 1 領域の場合があります。

は、その列または変数に関する基本情報 (データ・タイプ・コード、長さ属性 (LOB の場合を除く)、列名 (またはラベル)、CCSID、変数アドレス、標識変数アドレスなど) が含まれます。

拡張 **SQLVAR** 項目

拡張 SQLVAR 項目は、結果に LOB または特殊タイプの列が含まれている場合に (各列ごとに) 必要となります。特殊タイプの場合、拡張 SQLVAR には特殊タイプ名が入ります。 LOB の場合、拡張 SQLVAR には、変数の長さ属性と、実際の長さを含むバッファーを指すポインターが入ります。ロケーターまたはファイル参照変数を使用して LOB を表す場合、拡張 SQLVAR は不要です。

拡張 SQLVAR 項目は、次の場合にも各列ごとに必要となります。

- USING BOTH が指定されている場合。これは、列名およびラベルが戻されることを示します。
- USING ALL が指定されている場合。これは、列名、ラベル、およびシステム列名が戻されることを示します。

LOB および特殊タイプ情報を戻す拡張 SQLVAR 内のフィールドはオーバーラップせず、LOB およびラベル情報を戻すフィールドもオーバーラップしません。ラベル、LOB、および特殊タイプの組み合わせによっては、情報を戻すのに列ごとに複数の拡張 SQLVAR 項目が必要になる場合があります。 1880 ページの『必要な SQLVAR オカレンスの数の決定』を参照してください。

SQLDA ヘッダーのフィールドの説明

SQLDA は、ヘッダー構造の中の 4 つの変数と、それに続く一連の 5 つの変数からなる任意の数のオカレンス (これらは一括して SQLVAR という名前が付けられている) から構成されます。 OPEN、CALL、FETCH、および EXECUTE では、SQLVAR の各オカレンスで、それぞれ変数を 1 つずつ記述します。 PREPARE および DESCRIBE では、SQLVAR の各オカレンスで、結果表の列を記述します。

SQL INCLUDE ステートメントを使用することにより、以下のフィールド名が組み込まれます。

表 *137. SQLDA* ヘッダーのフィールドの説明

C の名前[152] **PL/I** の名前 **COBOL** の名前	フィールド データ・タイプ	**DESCRIBE** および **PREPARE** で使用する場合 **(SQLN** 以外は、データベース・マネージャーによってセットされる**)**	**FETCH**、**OPEN**、**CALL**、または **EXECUTE** で使用する場合 **(**ユーザーがステートメントを実行する前にセットする**)**
sqldaid SQLDAID	CHAR(8)	ストレージ・ダンプのための「目印」として、「SQLDA」が入ります。 SQLDAID の 7 番目のバイトは、各列に複数の SQLVAR 項目が必要かどうかを判断するために使用できます。詳しくは、 1880 ページの『必要な SQLVAR オカレンスの数の決定』を参照してください。	7 番目のバイトの '2' は、各列に 2 つの SQLVAR 項目が割り振られたことを示します。 7 番目のバイトの '3' は、各列に 3 つの SQLVAR 項目が割り振られたことを示します。 7 番目のバイトの '4' は、各列に 4 つの SQLVAR 項目が割り振られたことを示します。
sqldabc SQLDABC	INTEGER	SQLDA の長さ。	SQLDA として割り振られた記憶域のサイズ (バイト数)。 SQLN に指定されているオカレンスが入るだけの記憶域を割り振る必要があります。SQLDABC には、16+SQLN*(80) 以上の値をセットする必要があります (80 は SQLVAR のオカレンスの長さ)。 LOB または特殊タイプが指定された場合には、各パラメーター・マーカーごとに 2 つの SQLVAR 項目が必要です。
sqln SQLN	SMALLINT	データベース・マネージャーでは変更しません。 PREPARE または DESCRIBE ステートメントを実行する前に、ユーザー側でゼロ以上の値をセットしなければなりません。この値は、結果内の列の数と同じか、それより大きい値にセットするか、複数の SQLVAR 項目セットが必要な場合には、結果内の列の数の倍数にセットする必要があります。 SQLVAR 配列のオカレンス数を指示します。	SQLDA に用意する SQLVAR 配列の合計オカレンス数。 SQLN には、ゼロ以上の値をセットしなければなりません。 LOB または特殊タイプが指定された場合には、各パラメーター・マーカーごとに 2 つの SQLVAR 項目が必要であり、SQLN はパラメーター・マーカー数の 2 倍にセットしなければなりません。

152. この欄では、小文字の名前は C の名前を示し、大文字の名前は COBOL、PL/I、または RPG の名前を示しています。

表 137. SQLDA ヘッダーのフィールドの説明 (続き)

C の名前[152] PL/I の名前 COBOL の名前	フィールド データ・タイプ	DESCRIBE および PREPARE で使用する場合 (SQLN 以外は、データベース・マネージャーによってセットされる)	FETCH、OPEN、CALL、または EXECUTE で使用する場合 (ユーザーがステートメントを実行する前にセットする)
sqld SQLD	SMALLINT	SQLVAR 配列の各オカレンスによって記述する列の数 (記述するステートメントが選択ステートメント、CALL、または VALUES INTO 以外の場合は、ゼロ)。	ステートメント実行中に使用される SQLVAR 項目の SQLDA 内オカレンス数。 SQLD には、ゼロ以上で SQLN 以下の値をセットしなければなりません。

必要な SQLVAR オカレンスの数の決定

必要な SQLVAR オカレンス数は、SQLDA に提供されたステートメントと、記述されている列またはパラメーターのデータ・タイプによって決まります。 詳細については、上記の表を参照してください。

SQLDAID の 7 番目のバイトは常に、結果セットに LOB または UDT が入っている場合に必要な SQLVAR のセット数に設定されます。

SQLD が十分な数の SQLVAR オカレンスにセットされない場合、

- LOB および UDT が結果セットに入っていない場合は、SQLD は、すべてのセットに必要な SQLVAR オカレンスの合計数に設定されます。 LOB または UDT が結果セットに入っている場合は、SQLD は結果表内の列の数に設定され、SQLDAID の 7 番目のバイトは必要な SQLVAR 項目セットの数を表します。必要な SQLVAR 項目の数は、SQLDAID の 7 番目のバイトの値を SQLD に掛けることによって常に計算できます。
- 少なくとも基本 SQLVAR 項目用に十分な数の SQLVAR が指定されている場合、警告 (SQLSTATE 01594) が戻されます。この場合、基本 SQLVAR 項目は戻されますが、拡張 SQLVAR は戻されません。
- 基本 SQLVAR 項目用にさえも十分な数の SQLVAR が指定されていない場合は、警告 (SQLSTATE 01005) が戻されます。 SQLVAR 項目は返されません。

153

1881 ページの表 138、 1881 ページの表 139、および 1881 ページの表 140 は、基本および拡張 SQLVAR 項目をマップする方法を示しています。 基本と拡張の両方の SQLVAR 項目を含む SQLDA の場合、基本 SQLVAR 項目は最初のブロック内にあり、その後に拡張 SQLVAR 項目のブロックが続き、さらに必要であれば、その後に第 2、第 3 の拡張 SQLVAR 項目のブロックが続きます。 各ブロックでの SQLVAR 項目のオカレンス数は、多数の拡張 SQLVAR 項目が未使用である可能性があっても、SQLD 内の値と同じになります。

153. LOB または UDT が結果セットに入っていない場合、この警告が戻されるのは、標準オプションが指定されている場合のみです。 標準オプションについては、 xi ページの『標準への準拠』を参照してください。

表 *138. USING NAMES*、*USING SYSTEM NAMES*、*USING LABELS* または *USING ANY* の *SQLVAR* 配列の内容

LOB	DISTINCT タイプ	SQLDAID の 7 番目のバイト	SQLN 最小値	最初のセット (基本)	2 番目のセット (拡張)	3 番目のセット (拡張)	4 番目のセット (拡張)
なし	なし	ブランク	n	列名、システム列名、またはラベル	使用されません	使用されません	使用されません
あり	なし	2	2n	列名、システム列名、またはラベル	LOB	使用されません	使用されません
なし	あり	2	2n	列名、システム列名、またはラベル	特殊タイプ	使用されません	使用されません
あり	あり	2	2n	列名、システム列名、またはラベル	LOB および特殊タイプ	使用されません	使用されません

表 *139. USING BOTH* の *SQLVAR* 配列の内容

LOB	DISTINCT タイプ	SQLDAID の 7 番目のバイト	SQLN 最小値	最初のセット (基本)	2 番目のセット (拡張)	3 番目のセット (拡張)	4 番目のセット (拡張)
なし	なし	2	2n	列名	ラベル	使用されません	使用されません
あり	なし	2	2n	列名	LOB およびラベル	使用されません	使用されません
なし	あり	3	3n	列名	特殊タイプ	ラベル	使用されません
あり	あり	3	3n	列名	LOB および特殊タイプ	ラベル	使用されません

表 *140. USING ALL* の *SQLVAR* 配列の内容

LOB	DISTINCT タイプ	SQLDAID の 7 番目のバイト	SQLN 最小値	最初のセット (基本)	2 番目のセット (拡張)	3 番目のセット (拡張)	4 番目のセット (拡張)
なし	なし	3	3n	システム列名	ラベル	列名	使用されません
あり	なし	3	3n	システム列名	LOB およびラベル	列名	使用されません
なし	あり	4	4n	システム列名	特殊タイプ	ラベル	列名
あり	あり	4	4n	システム列名	LOB および特殊タイプ	ラベル	列名

SQLVAR のオカレンスのフィールドの説明

このセクションでは、基本および 2 次 SQLVAR のオカレンス内のフィールドについて説明します。

基本 SQLVAR のオカレンス内のフィールド

表 *141. SQLVAR* のフィールドの説明

C の名前[154] **COBOL** の名前 **PL/I** の名前 **RPG** の名前	フィールド データ・タイプ	**DESCRIBE** および **PREPARE** で使用する場合 **(**データベース・マネージャーがセットする**)**	**FETCH**、**OPEN**、**CALL**、および **EXECUTE** で使用する場合 **(**ユーザーがステートメントを実行する前にセットする**)**
sqltype SQLTYPE	SMALLINT	列のデータ・タイプと、その列に NULL を入れられるかどうかを指示します。データ・タイプを示すコードについては、 1886 ページの表 143 を参照してください。 特殊タイプの場合、特殊タイプの基本となっているデータ・タイプがこのフィールドに置かれます。基本 SQLVAR には、これが特殊タイプの記述の一部であることを示すものは含まれません。	ホスト変数のデータ・タイプと、その変数に標識変数が指定されているかどうかを指示します。データ・タイプを示すコードについては、 1886 ページの表 143 を参照してください。
sqllen SQLLEN	SMALLINT	列の長さ属性です。 日付/時刻の列の場合、その値のストリング表現の長さ。 1886 ページの表 143を参照してください。 LOB の場合、その LOB の長さ属性に関係なく、この値は 0 になります。 XML の場合、値は 0。拡張 SQLVAR 項目のフィールド SQLLONGLEN には、LOB または XML の長さの属性が入ります。	ホスト変数の長さ属性です。 1886 ページの表 143を参照してください。 LOB の場合、その LOB の長さ属性に関係なく、この値は 0 になります。拡張 SQLVAR 項目内のフィールド SQLLONGLEN には、LOB の長さ属性が入ります。 XML AS BLOB、CLOB、または DBCLOB の場合、値は 0 です。
sqlres SQLRES	CHAR(12)	予約済み。 SQLDATA の境界合わせ用。	予約済み。 SQLDATA の境界合わせ用。

154. この欄の小文字の名前は C の名前を示し、大文字の名前は PL/I、COBOL、および RPG の名前を示しています。

表 *141. SQLVAR* のフィールドの説明 *(*続き*)*

C の名前[154] **COBOL** の名前 **PL/I** の名前 **RPG** の名前	フィールド・データ・タイプ	**DESCRIBE** および **PREPARE** で使用する場合 **(**データベース・マネージャーがセットする**)**	**FETCH**、**OPEN**、**CALL**、および **EXECUTE** で使用する場合 **(**ユーザーがステートメントを実行する前にセットする**)**
sqldata SQLDATA	ポインター	ストリング列または XML 列の CCSID (1889 ページの表 144 を参照してください)。	ホスト変数のアドレスが入ります。 LOB ホスト変数の場合、拡張 SQLVAR 内の SQLDATALEN フィールドが NULL であれば、4 バイトの LOB 長を指し、その直後に LOB データが続きます。 拡張 SQLVAR 内の SQLDATALEN フィールドが NULL でない場合は、LOB データを指し、SQLDATALEN フィールドは 4 バイトの LOB 長を指します。
sqlind SQLIND	ポインター	選択ステートメントの場合、列が、WITH ROW CHANGE COLUMNS 属性を使用した結果追加されたかどうかを以下のように示します。 • -1 ROW CHANGE TOKEN (特殊) • -2 ROW CHANGE TOKEN (特殊ではない) • -3 RID • -4 RID_BIT その他の場合には、予約済み。	標識変数のアドレスが入ります。標識変数がない (SQLTYPE の値が偶数である) 場合は、使用されません。
sqlname SQLNAME	VARCHAR (30)	修飾のない列名。列に名前がない場合、ストリングは式から構成されて、戻されます。 この名前には大文字小文字の区別があります。全体を区切り文字で囲んではなりません。	ホスト変数の CCSID (1889 ページの表 144 を参照) が入ります。 XML データの場合、XML サブタイプを示すために sqlname を以下のように設定できます。 sqlname の長さは 8 です。 バイト 1 および 2: X'0000' にする必要があります。 バイト 3 および 4: CCSID を含めることができます。 バイト 5 および 6: X'0100' XML ホスト変数 (XML AS CLOB、XML AS DBCLOB、XML AS BLOB、XML AS CLOB_FILE、XML AS DBCLOB_FILE、XML AS BLOB_FILE) バイト 7 および 8: X'0000' にする必要があります。

副次 SQLVAR のオカレンスのフィールド

表 142. 拡張 *SQLVAR* のフィールドの説明

C の名前[155] **COBOL** の名前 **PL/I** の名前 **RPG** の名前	フィールド データ・タイプ	**DESCRIBE** および **PREPARE** で使用する場合 **(**データベース・マネージャーがセットする**)**	**FETCH**、**OPEN**、**CALL**、および **EXECUTE** で使用する場合 **(**ユーザーがステートメントを実行する前にセットする**)**
len.sqllonglen SQLLONGL SQLLONGLEN	INTEGER	LOB 列の長さ属性。 XML の場合、値は 0。	LOB または XML ホスト変数の長さ属性。データベース・マネージャーは、これらのデータ・タイプの場合、基本 SQLVAR 内の SQLLEN フィールドは無視します。長さ属性は、BLOB の場合はバイト数、CLOB、DBCLOB または XML の場合は文字数を示します。
*	CHAR(12)	予約済み。 SQLDATALEN の境界合わせ用。	予約済み。 SQLDATALEN の境界合わせ用。
*	ポインター	予約済み。	予約済み。
sqldatalen SQLDATAL SQLDATALEN	ポインター	使用されません。	LOB ホスト変数にのみ使用されます。 このフィールドの値が NULL ではない場合は、このフィールドは、LOB の実際の長さ (バイト単位) を含む 4 バイトの長さのバッファーを指します (DBCLOB の場合でも)。そして、一致する基本 SQLVAR の SQLDATA フィールドは、LOB データを指します。 このフィールドの値が NULL の場合、LOB の実際の長さは、一致する基本 SQLVAR 内の SQLDATA フィールドが指す最初の 4 バイトに保管され、LOB データは 4 バイトの長さの直後に続きます。 実際の長さは、BLOB または CLOB のバイト数と、DBCLOB の 2 バイトの文字数を示します。 このフィールドが使用されるかどうかに関係なく、フィールド SQLLONGLEN はセットしなければなりません。

155. この欄の小文字の名前は C の名前を示し、最初の大文字の名前は PL/I および RPG の名前を示しています。 2 番目の大文字の名前は COBOL の名前を示しています。

表 142. 拡張 *SQLVAR* のフィールドの説明 (続き)

C の名前[155] **COBOL** の名前 **PL/I** の名前 **RPG** の名前	フィールド データ・タイプ	**DESCRIBE** および **PREPARE** で使用する場合 **(**データベース・マネージャーがセットする**)**	**FETCH**、**OPEN**、**CALL**、および **EXECUTE** で使用する場合 **(**ユーザーがステートメントを実行する前にセットする**)**
sqldatatype_name SQLTNAME SQLDATATYPE-NAME	VARCHAR (30)	拡張 SQLVAR の SQLTNAME フィールドは、次のいずれかにセットされます。 • 特殊タイプの列の場合、データベース・マネージャーはこれを完全修飾特殊タイプ名にセットします。この修飾名が 30 バイトより長い場合は、切り捨てられます。 • ラベルの場合、データベース・マネージャーは、これをラベルの最初の 20 バイトにセットします。 • 列名の場合、データベース・マネージャーはこれを列名にセットします。	使用されません。

SQLTYPE と SQLLEN

以下の表は、SQLDA の SQLTYPE および SQLLEN フィールドに入る値を示したものです。 PREPARE および DESCRIBE で使用した場合に、SQLTYPE の値が偶数ならば、その列には NULL が許されないことを示します。また、SQLTYPE の値が奇数ならば、その列に NULL が許されることを示します。

注**:** DESCRIBE または PREPARE ステートメントで使用される SQLDA において、1 つのオペランドがNULL 可能であるか、または式の結果が -2 のマッピング・エラー NULL 値になる場合、その式に奇数値が戻されます。

FETCH、OPEN、CALL、および EXECUTE で使用した場合、SQLTYPE の値が偶数ならば、標識変数が指定されていないことを意味し、奇数ならば、SQLIND に標識変数のアドレスが入っていることを意味します。

表 *143. PREPARE*、*DESCRIBE*、*FETCH*、*OPEN*、*CALL*、または *EXECUTE* の場合の *SQLTYPE* および *SQLLEN* の値

	PREPARE および **DESCRIBE** の場合		**FETCH**、**OPEN**、**CALL**、および **EXECUTE** の場合	
SQLTYPE	**COLUMN DATA TYPE**	**SQLLEN**	**HOST VARIABLE DATA TYPE**	**SQLLEN**
384/385	日付 [158]	10	日付の固定長文字ストリング表現	ホスト変数の長さ属性
388/389	時刻	8	時刻の固定長文字ストリング表現	ホスト変数の長さ属性
392/393	タイム・スタンプ	TIMESTAMP(0) の場合 19 それ以外で TIMESTAMP(*p*) の場合 20+*p*	タイム・スタンプの固定長文字ストリング表現	ホスト変数の長さ属性
396/397	データ・リンク	列の長さ属性	データ・リンク	ホスト変数の長さ属性
400/401	該当しない	該当しない	NUL で終了するグラフィック・ストリング	ホスト変数の長さ属性
404/405	BLOB	0 [157]	BLOB	使用されません。 [157]
408/409	CLOB	0 [157]	CLOB	使用されません。 [157]
412/413	DBCLOB	0 [157]	DBCLOB	使用されません。 [157]
448/449	可変長文字ストリング	列の長さ属性	可変長文字ストリング	ホスト変数の長さ属性
452/453	固定長文字ストリング	列の長さ属性	固定長文字ストリング	ホスト変数の長さ属性
456/457	長可変長文字ストリング	列の長さ属性	長可変長文字ストリング	ホスト変数の長さ属性
460/461	該当しない	該当しない	NUL で終了する文字ストリング	ホスト変数の長さ属性
464/465	可変長グラフィック・ストリング	列の長さ属性	可変長グラフィック・ストリング	ホスト変数の長さ属性
468/469	固定長グラフィック・ストリング	列の長さ属性	固定長グラフィック・ストリング	ホスト変数の長さ属性
472/473	長い可変長グラフィック・ストリング	列の長さ属性	長いグラフィック・ストリング	ホスト変数の長さ属性

表 143. *PREPARE*、*DESCRIBE*、*FETCH*、*OPEN*、*CALL*、または *EXECUTE* の場合の *SQLTYPE* および *SQLLEN* の値 (続き)

	PREPARE および **DESCRIBE** の場合		**FETCH**、**OPEN**、**CALL**、および **EXECUTE** の場合	
SQLTYPE	**COLUMN DATA TYPE**	**SQLLEN**	**HOST VARIABLE DATA TYPE**	**SQLLEN**
476/477	該当しない	該当しない	PASCAL の L- ストリング	ホスト変数の長さ属性
480/481	浮動小数点数	単精度では 4、倍精度では 8	浮動小数点数	単精度では 4、倍精度では 8
484/485	パック 10 進数	バイト 1 は精度、バイト 2 は位取り	パック 10 進数	バイト 1 は精度、バイト 2 は位取り
488/489	ゾーン 10 進数	バイト 1 は精度、バイト 2 は位取り	ゾーン 10 進数	バイト 1 は精度、バイト 2 は位取り
492/493	64 ビット整数	8 [156]	64 ビット整数	8
496/497	大整数	4 [156]	大整数	4
500/501	短整数	2 [156]	短整数	2
504/505	該当しない	該当しない	DISPLAY SIGN LEADING SEPARATE	バイト 1 は精度、バイト 2 は位取り
904/905	ROWID	40	ROWID	40
908/909	可変長バイナリー・ストリング	列の長さ属性	可変長バイナリー・ストリング	ホスト変数の長さ属性
912/913	固定長バイナリー・ストリング	列の長さ属性	固定長バイナリー・ストリング	ホスト変数の長さ属性
916/917	該当しない	該当しない	BLOB ファイル参照変数	267
920/921	該当しない	該当しない	CLOB ファイル参照変数	267
924/925	該当しない	該当しない	DBCLOB ファイル参照変数	267
960/961	該当しない	該当しない	BLOB ロケーター	4
964/965	該当しない	該当しない	CLOB ロケーター	4
968/969	該当しない	該当しない	DBCLOB ロケーター	4
972	該当しない	該当しない	結果セット・ロケーター	8
988/989	XML	0	該当しない。XML AS CLOB、 XML AS DBCLOB、または XML AS BLOB を使用します。	0
996/997	該当しない DECFLOAT(16) DECFLOAT(34)	該当しない 8 16	DECFLOAT(7) [159] DECFLOAT(16) DECFLOAT(34)	4 8 16
2452/2453	該当しない	該当しない	XML ロケーター	4

を参照してください。

156. SQLDA では、2 進数は長さ 2、4、または 8 で表される場合と、バイト 1 の精度とバイト 2 の位取りによって表される場合があります。最初のバイトが x'00' より大きい場合は、精度および位取りが入っていることを示します。

157. 拡張 SQLVAR 内のフィールド SQLLONGLEN には、列の長さ属性が含まれます。

158. *JUL、*YMD、*DMY、および *MDY フォーマットの場合はもっと短くなります。詳しくは、 96 ページの表 8

159. Db2 は DECFLOAT(7) の数値を内部に格納しませんが、アプリケーションから渡された DECFLOAT(7) の数値はサポートします。SQL ステートメントで参照されている DECFLOAT(7) 変数は、DECFLOAT(16) に変換されます。

SQLDATA または SQLNAME 内の CCSID の値

OPEN、FETCH、CALL、および EXECUTE ステートメントでは、SQLVAR エレメントの SQLNAME フィールドを使用して、ストリング・ホスト変数の CCSID を指定することができます。 SQLNAME フィールドによって CCSID を指定する場合は、SQLNAME の長さを 8 にセットしなければなりません。さらに、SQLNAME の最初の 4 バイトは次の表に従ってセットする必要があります。CCSID の指定がない場合、ジョブの CCSID が使用されます。

DESCRIBE、DESCRIBE TABLE、および PREPARE ステートメントにおいて、結果表の列がストリング列の場合、SQLVAR のエレメントの SQLDATA フィールドにその列の CCSID が入ります。その CCSID は、表 144 に示すようにバイト 3 とバイト 4 に入ります。

表 *144. SQLDATA* または *SQLNAME* に示される *CCSID* の値

データ・タイプ	エンコード スキーム	バイト **1** と **2**	バイト **3** と **4**
文字	SBCS データ	X'0000'	ccsid
文字	MIXED (混合) データ	X'0000'	ccsid
文字	ビット・データ	X'0000'	65535
グラフィック	該当しない	X'0000'	ccsid
上記以外のデータ・タイプ	該当しない	該当しない	該当しない

認識されずサポートされない SQLTYPES

SQLDA の SQLTYPE フィールドに表示される値は、データの送信側および受信側の双方で使用可能なデータ・タイプ・サポートのレベルに依存しています。 これは、新しいデータ・タイプをプロダクトに追加する際に特に重要です。

データの送信側または受信側が、新しいデータ・タイプをサポートしている場合とサポートしていない場合があり、認識している場合と認識さえしていない場合があります。状況に応じて、新しいデータ・タイプが戻される場合、データの送信側および受信側の両者が合意した互換データ・タイプが戻される場合、あるいはエラーが発生する場合があります。

以下の表は、送信側および受信側の両者が互換データ・タイプを使用することを合意した場合に、発生する可能性があるマッピングを示しています。このマッピングが発生するのは、送信側または受信側の少なくとも一方が提供されたデータ・タイプをサポートしていない場合です。 非サポート・データ・タイプは、アプリケーションまたはデータベース・マネージャーのどちらかによって提供されます。

表 *145.* サポートされないデータ・タイプの互換データ・タイプ

データ・タイプ	互換データ・タイプ
BIGINT	DECIMAL(19,0)
ROWID	VARCHAR(40) ビット・データ用
TIMESTAMP(n)	TIMESTAMP(6)

INCLUDE SQLDA の宣言

このセクションでは、C および C++、COBOL、ILE COBOL、PL/I、および ILE RPG について、同等の INCLUDE SQLDA 宣言を示します。

C および C++ の場合

C および C++ の場合、INCLUDE SQLDA 宣言は以下と同等です。

```
#ifndef SQLDASIZE
struct sqlda
{
       unsigned char  sqldaid[8];
       long           sqldabc;
       short          sqln;
       short          sqld;
       struct sqlvar
       {
              short          sqltype;
              short          sqllen;
              union {
                unsigned char *sqldata;
                long long sqld_result_set_locator;  };
              union {
                short *sqlind;
                long sqld_row_change;
                long sqld_result_set_rows;   };
              struct sqlname
              {
                     short          length;
                     unsigned char  data[30];
              } sqlname;
       } sqlvar[1];
};

struct sqlvar2
       { struct
                {  long           sqllonglen;
                   char           reserve1[28];
                }  len;
                char *sqldatalen;
                struct sqldistinct_type
                       {  short          length;
                          unsigned char  data[30];
                       } sqldatatype_name;
       };

#define  SQLDASIZE(n) (sizeof(struct sqlda)+(n-1) * sizeof(struct sqlvar))
#endif
```

図 *11. C* および *C++* の場合の *INCLUDE SQLDA* 宣言

```
                      /*********************************************************************/
                      /* Macros for using the sqlvar2 fields.                              */
                      /*********************************************************************/

                      /*********************************************************************/
                      /*   '2' in the 7th byte of sqldaid indicates a doubled number of    */
                      /*       sqlvar entries.                                             */
                      /*   '3' in the 7th byte of sqldaid indicates a tripled number of    */
                      /*       sqlvar entries.                                             */
                      /*   '4' in the 7th byte of sqldaid indicates a quadrupled number of */
                      /*       sqlvar entries.                                             */
                      /*********************************************************************/
```

```
#define SQLDOUBLED '2'
#define SQLSINGLED ' '

/*************************************************************************/
/* GETSQLDOUBLED(daptr) returns 1 if the SQLDA pointed to by         */
/* daptr has been doubled, or 0 if it has not been doubled.          */
/*************************************************************************/
#define GETSQLDOUBLED(daptr) (((daptr)->sqldaid[6]==  \
 (char) SQLDOUBLED)  ?                                \
    (1)           :      \
    (0)             )

/*************************************************************************/
/* SETSQLDOUBLED(daptr, SQLDOUBLED) sets the 7th byte of sqldaid     */
/* to '2'.                                                           */
/* SETSQLDOUBLED(daptr, SQLSINGLED) sets the 7th byte of sqldaid     */
/* to be a ' '.                                                      */
/*************************************************************************/
#define SETSQLDOUBLED(daptr, newvalue)               \
  (((daptr)->sqldaid[6] =(newvalue)))

/*************************************************************************/
/* GETSQLDALONGLEN(daptr,n) returns the data length of the nth       */
/* entry in the sqlda pointed to by daptr. Use this only if the      */
/* sqlda was doubled or tripled and the nth SQLVAR entry has a       */
/* LOB datatype.                                                     */
/*************************************************************************/
#define GETSQLDALONGLEN(daptr,n) ((long) (((struct sqlvar2 *) \
&((daptr)->sqlvar[(n) +((daptr)->sqld)])) ->len.sqllonglen))

/*************************************************************************/
/* SETSQLDALONGLEN(daptr,n,len) sets the sqllonglen field of the     */
/* sqlda pointed to by daptr to len for the nth entry. Use this only */
/* if the sqlda was doubled or tripled and the nth SQLVAR entry has  */
/* a LOB datatype.                                                   */
/*************************************************************************/
#define SETSQLDALONGLEN(daptr,n,length) {            \
 struct sqlvar2 *var2ptr;                            \
 var2ptr = (struct sqlvar2 *) &((daptr)->sqlvar[(n)+  \
   ((daptr)->sqld)]);                                \
 var2ptr->len.sqllonglen = (long) (length);              \
}

/*************************************************************************/
/* SETSQLDALENPTR(daptr,n,ptr) sets a pointer to the data length for */
/* the nth entry in the sqlda pointed to by daptr.                   */
/* Use this only if the sqlda has been doubled or tripled.           */
/*************************************************************************/
#define SETSQLDALENPTR(daptr,n,ptr) {                \
 struct sqlvar2 *var2ptr;                            \
 var2ptr = (struct sqlvar2 *) &((daptr)->sqlvar[(n)+  \
   ((daptr)->sqld)]);                                \
 var2ptr->sqldatalen = (char *) ptr;                 \
 }

/*************************************************************************/
/* GETSQLDALENPTR(daptr,n) returns a pointer to the data length for  */
/* the nth entry in the sqlda pointed to by daptr. Unlike the inline */
/* value (union sql8bytelen len), which is 8 bytes, the sqldatalen   */
/* pointer field returns a pointer to a long (4 byte) integer.       */
/* If the SQLDATALEN pointer is zero, a NULL pointer is be returned. */
/*                                                                   */
/* NOTE: Use this only if the sqlda has been doubled or tripled.     */
/*************************************************************************/
#define GETSQLDALENPTR(daptr,n) (                    \
  (((struct sqlvar2 *) &(daptr)->sqlvar[(n) +        \
```

```
   (daptr)->sqld])->sqldatalen == NULL) ?           \
  ((long *) NULL ) : ((long *) ((struct sqlvar2 *)   \
 &(daptr)->sqlvar[(n) + (daptr) ->sqld])->sqldatalen))
```

COBOL と ILE COBOL の場合

COBOL と ILE COBOL の場合、INCLUDE SQLDA 宣言は、以下のステートメントと同等です。

```
1 SQLDA.
  05 SQLDAID     PIC X(8).
  05 SQLDABC     PIC S9(9) BINARY.
  05 SQLN        PIC S9(4) BINARY.
  05 SQLD        PIC S9(4) BINARY.
  05 SQLVAR OCCURS 0 TO 409 TIMES DEPENDING ON SQLD.
     10 SQLVAR1.
        15 SQLTYPE   PIC S9(4) BINARY.
        15 SQLLEN    PIC S9(4) BINARY.
        15 FILLER  REDEFINES SQLLEN.
           20 SQLPRECISION PIC X.
           20 SQLSCALE     PIC X.
        15 SQLRES    PIC X(12).
        15 SQLDATA   POINTER.
        15 SQL-RESULT-SET-LOCATOR-R REDEFINES SQLDATA.
           20 SQL-RESULT-SET-LOCATOR PIC S9(18) BINARY.
        15 SQLIND   POINTER.
        15 SQL-ROW-CHANGE-SQL-R REDEFINES SQLIND.
           20 SQLD-ROW-CHANGE FIC S9(9) BINARY.
        15 SQL-RESULT-SET-ROWS-R PIC REDEFINES SQLIND.
           20 SQLD-RESULT-SET-ROWS PIC S9(9) BINARY.
        15 SQLNAME.
           49 SQLNAMEL PIC S9(4) BINARY.
           49 SQLNAMEC PIC X(30).
     10 SQLVAR2 REDEFINES SQLVAR1.
        15 SQLVAR2-RESERVED-1 PIC S9(9) BINARY.
        15 SQLLONGLEN REDEFINEDS SQLVAR2-RESERVED-1
                                PIC S9(9) BINARY.
        15 SQLVAR2-RESERVED-2 PIC X(28).
        15 SQLDATALEN POINTER.
        15 SQLDATATYPE-NAME.
           49 SQLDATATYPE_NAMEL PIC S9(4) BINARY.
           49 SQLDATATYPE_NAMEC PIC X(30).
```

図 *12. INCLUDE SQLDA* の宣言 *(COBOL* の場合*)*

PL/I の場合

PL/I の場合、INCLUDE SQLDA の宣言は、以下のステートメントと同等です。

```
DCL 1 SQLDA BASED(SQLDAPTR),
      2 SQLDAID     CHAR(8),
      2 SQLDABC     BIN FIXED(31),
      2 SQLN        BIN FIXED,
      2 SQLD        BIN FIXED,
      2 SQLVAR      (99),
        3 SQLTYPE   BIN FIXED,
        3 SQLLEN    BIN FIXED,
        3 SQLRES    CHAR(12),
        3 SQLDATA   PTR,
        3 SQLIND    PTR,
        3 SQLNAME   CHAR(30) VAR,

    1 SQLDA2 BASED(SQLDAPTR),
      2 SQLDAID2    CHAR(8),
      2 SQLDABC2    FIXED(31) BINARY,
      2 SQLN2       FIXED(15) BINARY,
      2 SQLD2       FIXED(15) BINARY,
      2 SQLVAR2     (99),
        3 SQLBIGLEN,
          4 SQLLONGL FIXED(31) BINARY,
          4 SQLRSVDL FIXED(31) BINARY,
        3 SQLDATAL  POINTER,
        3 SQLTNAME  CHAR(30) VAR;

    DECLARE SQLSIZE    FIXED(15) BINARY;
    DECLARE SQLDAPTR   PTR;
    DECLARE SQLDOUBLED CHAR(1)   INITIAL('2') STATIC;
    DECLARE SQLSINGLED CHAR(1)   INITIAL(' ') STATIC;
```

図 *13. INCLUDE SQLDA* の宣言 *(PL/I* の場合*)*

ILE RPG の場合

ILE RPG の場合、INCLUDE SQLDA の宣言は、以下のステートメントと同等です。

```
// SQL DESCRIPTOR AREA
DCL-DS SQLDA;
  SQLDAID CHAR(8);
  SQLDABC INT(10);
  SQLN INT(5);
  SQLD INT(5);
  SQL_VAR CHAR(80) DIM(SQL_NUM);
   *N POINTER OVERLAY(SQL_VAR:17);
   *N POINTER OVERLAY(SQL_VAR:33);
END-DS SQLDA;
DCL-DS SQLVAR;
  SQLTYPE INT(5);
  SQLLEN INT(5);
  SQLRES CHAR(12);
  SQLINFO1 CHAR(16);
   SQLDATA POINTER OVERLAY(SQLINFO1);
   SQL_RESULT_SET_LOCATOR INT(20) OVERLAY(SQLINFO1);
  SQLINFO2 CHAR(16);
   SQLIND POINTER OVERLAY(SQLINFO2);
   SQL_ROW_CHANGE INT(10) OVERLAY(SQLINFO2);
   SQL_RESULT_SET_ROWS INT(10) OVERLAY(SQLINFO2);
  SQLNAMELEN INT(5);
  SQLNAME CHAR(30);
END-DS SQLVAR;
// EXTENDED SQLDA
DCL-DS SQLVAR2;
  SQLLONGL INT(10);
   SQLRSVDL CHAR(28);
   SQLDATAL POINTER;
   SQLTNAMELN INT(5);
   SQLTNAME CHAR(30);
END-DS SQLVAR2;
```

図 *14. INCLUDE SQLDA* の宣言 *(ILE RPG* の場合*)*

ユーザーは、SQL_NUM の定義を行わなければなりません。 SQL_NUM は、SQL_VAR に必要な次元を持つ数値定数として定義する必要があります。

RPG は配列内の構造をサポートしていないので、SQLDA は 3 つのデータ構造として生成されます。2 番目および 3 番目のデータ構造を使用すると、フィールド記述が入っている SQLDA の部分をセットアップして参照できます。

SQLDA のフィールド記述をセットするには、プログラムは SQLVAR (または SQLVAR2) のサブフィールドにフィールド記述をセットアップしてから、SQLVAR (または SQLVAR2) の MOVEA を SQL_VAR,n に対して実行します。ここで、n は SQLDA の中のフィールドの数を表しています。 この動作は、すべてのフィールド記述がセットされるまで繰り返されます。

SQLDA フィールド記述を参照するときに、ユーザーは SQL_VAR,n の MOVEA を SQLVAR (または SQLVAR2) に対して実行します。ここで、n は処理されるフィールド記述の数を表しています。

付録 E. CCSID の値

このセクションに示す表では、IBM リレーショナル・データベース・プロダクトで提供される CCSID と変換について説明します。

詳しくは、 39 ページの『文字変換』を参照してください。

次のリストは、以下の表の Db2 プロダクトの欄で使用されている記号を定義しています。

X
該当の CCSID へ、または該当の CCSID からの変換を行う変換表が存在することを示しています。これは、この CCSID をローカル・データのタグ付けに使用できることも意味しています。

C
該当の CCSID から他の CCSID へ変換する変換表が存在することを示しています。これは、該当の CCSID が外部コード化体系であるため、その CCSID をローカル・データのタグ付けには使用できないことも意味しています (例えば、850 などのような PC データの CCSID は、Db2 for i のローカル・データのタグ付けには使用できません)。

ブランク
特定の製品が CCSID をサポートしていないことを示しています。特定の製品との相互運用性が必要でない限り、このような CCSID を使用しないでください。

リストされている CCSID に関するこの情報は、本書の発行日の時点において最新のものです。発行日以降に CCSID が追加された可能性もあり、その場合それらは下のリストにはありません。

表 146. 汎用文字セット (UTF-8、UTF-16、および UCS-2)

CCSID	説明	z/OS	i	AIX	HP	Sun	NT	SCO	SGI	Linux
1200	UTF-16	X	X	X	X	X	X	X	X	X
1208	UTF-8 レベル 3	X	X	X	X	X	X	X	X	X
13488	UCS-2 レベル 1	C	X	C *	C *	C *	C *	C *	C *	C *

注: * Db2LUW では、13488 が使用されるのは、eucJP および eucTW データベースの GRAPHIC 列をタグ付けする場合のみです。

表 *147. EBCDIC グループ 1 (ラテン-1)* の国または地域用の *CCSID*

CCSID	説明	z/OS	i	AIX	HP	Sun	NT	SCO	SGI	Linux
37	米国、カナダ、オランダ、ポルトガル、ブラジル、オーストラリア、ニュージーランド	X	X	C	C	C	C	C	C	C
256	ワード・プロセッシング、オランダ	X	X							
273	オーストリア、ドイツ	X	X	C	C	C	C	C	C	C
274	ベルギー	X		C	C	C	C	C	C	C
277	デンマーク、ノルウェー	X	X	C	C	C	C	C	C	C
278	フィンランド、スウェーデン	X	X	C	C	C	C	C	C	C
280	イタリア	X	X	C	C	C	C	C	C	C
284	スペイン、ラテン・アメリカ (スペイン語圏)	X	X	C	C	C	C	C	C	C
285	英国	X	X	C	C	C	C	C	C	C
297	フランス	X	X	C	C	C	C	C	C	C
500	ベルギー、カナダ、スイス、国際ラテン -1	X	X	C	C	C	C	C	C	C
871	アイスランド	X	X	C	C	C	C	C	C	C
924	ラテン 0	X	X							
1047	ラテン -0 (ユーロ対応)	X	X							
1140	米国、カナダ、オランダ、ポルトガル、ブラジル、オーストラリア、ニュージーランド	X	X	C	C	C	C	C	C	C
1141	オーストリア、ドイツ	X	X	C	C	C	C	C	C	C
1142	デンマーク、ノルウェー	X	X	C	C	C	C	C	C	C
1143	フィンランド、スウェーデン	X	X	C	C	C	C	C	C	C
1144	イタリア	X	X	C	C	C	C	C	C	C
1145	スペイン、ラテン・アメリカ (スペイン語圏)	X	X	C	C	C	C	C	C	C
1146	英国	X	X	C	C	C	C	C	C	C
1147	フランス	X	X	C	C	C	C	C	C	C
1148	ベルギー、カナダ、スイス、国際ラテン -1	X	X	C	C	C	C	C	C	C
1149	アイスランド	X	X	C	C	C	C	C	C	C

表 148. PC データおよび ISO グループ 1 (ラテン -1) の国または地域用の CCSID

CCSID	説明	z/OS	i	AIX	HP	Sun	NT	SCO	SGI	Linux
437		X	C	C	C	C	C	C	C	C
819	ラテン -1 の国または地域 (ISO 8859-1)	X	C	X	X	X	C	X	X	X
850	ラテン・アルファベット番号 1; ラテン -1 の国または地域	X	C	X	C	C	C	C	C	C
858	ラテン・アルファベット番号 1; ラテン -1 の国または地域 (ユーロ対応)	X	C							
860	ポルトガル (850 のサブセット)	X	C	C	C	C	C	C	C	C
861	アイスランド	X	C							
863	カナダ (850 のサブセット)	X	C	C	C	C	C	C	C	C
865	デンマーク、ノルウェー、フィンランド、スウェーデン	X	C							
923	ラテン 0	X	C	X	X	X	C	C	C	X
1009	IRV 7 ビット	X	C							
1010	フランス 7 ビット	X	C							
1011	ドイツ 7 ビット	X	C							
1012	イタリア 7 ビット	X	C							
1013	英国 7 ビット	X	C							
1014	スペイン 7 ビット	X	C							
1015	ポルトガル 7 ビット	X	C							
1016	ノルウェー 7 ビット	X	C							
1017	デンマーク 7 ビット	X	C							
1018	フィンランドおよびスウェーデン 7 ビット	X	C							
1019	ベルギーおよびオランダ 7 ビット	X	C							
1051	HP エミュレーション	X	C	C	X	C	C	C	C	C
1252	Windows ** ラテン-1	X	C	C	C	C	X	C	C	C
1275	Macintosh ** ラテン 1	X	C							
5348	Windows ラテン -1 (ユーロ対応)	X	C							

表 149. EBCDIC グループ 1a (非ラテン-1 SBCS) の国または地域用の CCSID

CCSID	説明	z/OS	i	AIX	HP	Sun	NT	SCO	SGI	Linux
420	アラビア語 (タイプ 4)	X	X	C	C	C	C	C	C	C
423	ギリシャ語	X	X	C	C	C	C	C	C	C
424	ヘブライ語(タイプ 4)	X	X	C	C	C	C	C	C	C
425	アラビア語(タイプ 5)		X	C	C	C	C	C	C	C
870	ラテン -2 マルチリンガル	X	X	C	C	C	C	C	C	C
875	ギリシャ語	X	X	C	C	C	C	C	C	C
880	キリル文字 マルチリンガル	X	X							
905	トルコ・ラテン -3 マルチリンガル	X	X							
918	ウルドゥー語	X	X							
1025	キリル文字 マルチリンガル	X	X	C	C	C	C	C	C	C
1026	トルコ・ラテン -5	X	X	C	C	C	C	C	C	C
1097	ペルシア語	X	X							
1112	バルト語 マルチリンガル	X	X	C	C	C	C	C	C	C
1122	エストニア語	X	X	C	C	C	C	C	C	C
1123	ウクライナ語	X	X	C	C	C	C	C	C	C
1137	デーバナーガリー文字	X	X	C	C	C	C	C	C	C
1153	ラテン -2 (ユーロ対応)	X	X	C	C	C	C	C	C	C
1154	キリル文字 (ユーロ対応)	X	X	C	C	C	C	C	C	C
1155	トルコ・ラテン -5 (ユーロ対応)	X	X	C	C	C	C	C	C	C
1156	バルト語 (ユーロ対応)	X	X	C	C	C	C	C	C	C
1157	エストニア語 (ユーロ対応)	X	X	C	C	C	C	C	C	C
1158	ウクライナ語 (ユーロ対応)	X	X	C	C	C	C	C	C	C
1166	キリル文字マルチリンガル (ユーロ対応)		X							
1175	トルコ語 Latin-5 (トルコ・リラ対応)		X							
4971	ギリシャ語 (ユーロ対応)	X	X							
5233	デーバナーガリー文字 (インド・ルピー対応)		X							
8612	アラビア語(タイプ 5)	X	X							
12708	アラビア語 (タイプ 7)		X							
62211	ヘブライ語 (タイプ 5)		X	C	C	C	C	C	C	C
62224	アラビア語 (タイプ 6)		X	C	C	C	C	C	C	C
62229	ヘブライ語 (タイプ 8)			C	C	C	C	C	C	C
62233	アラビア語 (タイプ 8)			C	C	C	C	C	C	C
62234	アラビア語 (タイプ 9)			C	C	C	C	C	C	C
62235	ヘブライ語 (タイプ 6)		X	C	C	C	C	C	C	C
62240	ヘブライ語 (タイプ 11)			C	C	C	C	C	C	C
62245	ヘブライ語 (タイプ 10)		X	C	C	C	C	C	C	C

表 149. EBCDIC グループ 1a (非ラテン-1 SBCS) の国または地域用の CCSID (続き)

CCSID	説明	z/OS	i	AIX	HP	Sun	NT	SCO	SGI	Linux
62250	アラビア語 (タイプ 12)			C	C	C	C	C	C	C
62251	アラビア語 (タイプ 6)		X	C	C	C	C	C	C	C

ストリング・タイプ:

4	表示 / 左から右 / 成形 / 対称スワッピング・オフ
5	暗黙 / 左から右 / 非成形 / 対称スワッピング・オン
6	暗黙 / 右から左 / 非成形 / 対称スワッピング・オン
7	表示 / コンテキスト / 非成形 / 対称スワッピング・オフ
8	表示 / 右から左 / 成形 / 対称スワッピング・オフ
9	表示 / 右から左 / 成形 / 対称スワッピング・オン
10	暗黙 / コンテキスト依存-左 / 非成形 / 対称スワッピング・オン
11	暗黙 / コンテキスト依存-右 / 非成形 / 対称スワッピング・オン
12	暗黙 / 右から左 / 成形 / 対称スワッピング・オン

表 *150. PC* データおよび *ISO* グループ *1a (*非ラテン*-1 SBCS)* の国または地域用の *CCSID*

CCSID	説明	z/OS	i	AIX	HP	Sun	NT	SCO	SGI	Linux
720	アラビア語 (MS-DOS)	X	C							
737	ギリシャ語 (MS-DOS)	X	C	C	C	C	X	C	C	C
775	バルト語 (MS-DOS)	X	C							
808	キリル文字 (ユーロ対応)	X								
813	ギリシャ/ラテン (ISO 8859-7)	X	C	X	X	C	C	X	C	X
848	ウクライナ語 (ユーロ対応)	X								
849	ベラルーシ (ユーロ対応)	X								
851	ギリシャ語	X	C							
852	ラテン -2 マルチリンガル	X	C	C	C	C	C	C	C	C
855	キリル文字 マルチリンガル	X	C	C	C	C	C	C	C	C
856	アラビア語(タイプ 5)	X	C	X	C	C	C	C	C	C
857	トルコ・ラテン -5	X	C	C	C	C	C	C	C	C
862	ヘブライ語(タイプ 4)	X	C	C	C	C	C	C	C	C
864	アラビア語(タイプ 5)	X	C	C	C	C	C	C	C	C
866	キリル文字	X	C	C	C	C	C	C	C	C
867	ヘブライ語 (ユーロ対応) (タイプ 10)	X								
868	ウルドゥー語	X	C							
869	ギリシャ語	X	C	C	C	C	C	C	C	C
872	キリル文字マルチリンガル (ユーロ対応)	X								
878	ロシア語インターネット	X	C							
901	バルト語 8 ビット (ユーロ対応)	X	C							
902	エストニア語 8 ビット (ユーロ対応)	X	C							
912	ラテン -2 (ISO 8859-2)	X	C	X	X	C	C	X	C	X
914	ラテン -4 (ISO 8859-4)	X	C							
915	キリル文字 マルチリンガル (ISO 8859-5)	X	C	X	X	C	C	X	C	X
916	ヘブライ語/ラテン (ISO 8859-8) (タイプ 5)	X	C	X	C	C	C	C	C	X
920	トルコ・ラテン-5 (ISO 8859-9)	X	C	X	X	C	C	X	C	X
921	バルト語 8 ビット (ISO 8859-13)	X	C	X	C	C	C	C	C	C
922	エストニア語 8 ビット	X	C	X	C	C	C	C	C	C
1008	アラビア語 8 ビット ISO	X	C							
1046	アラビア語(タイプ 5)	X	C	X	C	C	C	C	C	C
1089	アラビア語 (ISO 8859-6) (タイプ 5)	X	C	X	X	C	C	C	C	C

表 150. PC データおよび ISO グループ 1a (非ラテン-1 SBCS) の国または地域用の CCSID (続き)

CCSID	説明	z/OS	i	AIX	HP	Sun	NT	SCO	SGI	Linux
1098	ペルシア語	X	C							
1124	ウクライナ語 8 ビット ISO	X	C	X	C	C	C	C	C	C
1125	ウクライナ語	X	C	C	C	C	C	C	C	C
1131	ベラルーシ語	X	C	C	C	C	C	C	C	C
1250	Windows ラテン -2	X	C	C	C	C	X	C	C	C
1251	Windows キリル文字	X	C	C	C	C	X	C	C	C
1253	Windows ギリシャ語	X	C	C	C	C	X	C	C	C
1254	Windows トルコ語	X	C	C	C	C	X	C	C	C
1255	Windows ヘブライ語 (タイプ 5)	X	C	C	C	C	X	C	C	C
1256	Windows アラビア語 (タイプ 5)	X	C	C	C	C	X	C	C	C
1257	Windows バルト語	X	C	C	C	C	X	C	C	C
1280	Macintosh** ギリシャ語	X	C							
1281	Macintosh** トルコ語	X	C							
1282	Macintosh** ラテン -2	X	C							
1283	Macintosh** キリル文字	X	C							
4909	ISO 8859-7 ギリシャ語/ラテン (ユーロ対応)	X	C							
4948	ラテン -2 マルチリンガル	X	C							
4951	キリル文字 マルチリンガル	X	C							
4952	ヘブライ語	X	C							
4953	トルコ・ラテン -5	X	C							
4960	アラビア語	X	C							
4965	ギリシャ語		C							
5346	Windows ラテン -2 (ユーロ対応)	X								
5347	Windows キリル文字 (ユーロ対応)	X								
5349	Windows ギリシャ語 (ユーロ対応)	X								
5350	Windows トルコ語 (ユーロ対応)	X								
5351	Windows ヘブライ語 (ユーロ対応)	X								
5352	Windows アラビア語 (ユーロ対応)	X								
5353	Windows バルト語 Rim (ユーロ対応)	X								
9056	アラビア語 (記憶域交換)	X	C							
62208	ヘブライ語(タイプ 4)			X	X	X	X	X	X	X
62209	ヘブライ語 (タイプ 10)		C	C	C	C	C	C	C	C

表 150. PC データおよび ISO グループ 1a (非ラテン-1 SBCS) の国または地域用の CCSID (続き)

CCSID	説明	z/OS	i	AIX	HP	Sun	NT	SCO	SGI	Linux
62210	ヘブライ語/ラテン (ISO 8859-8) (タイプ 4)		C	X	X	C	C	C	C	C
62213	ヘブライ語 (タイプ 5)		C	C	C	C	C	C	C	C
62215	Windows ヘブライ語 (タイプ 4)		C	C	C	C	X	C	C	C
62218	アラビア語 (タイプ 4)		C	C	C	C	C	C	C	C
62220	ヘブライ語 (タイプ 6)			X	X	X	X	X	C	C
62221	ヘブライ語 (タイプ 6)		C	C	C	C	C	C	C	C
62222	ヘブライ語/ラテン (ISO 8859-8) (タイプ 6)		C	X	X	C	C	C	C	C
62223	Windows ヘブライ語 (タイプ 6)		C	C	C	C	X	C	C	C
62225	アラビア語 (タイプ 6)			C	C	C	C	C	C	C
62226	アラビア語 (タイプ 6)			X	C	C	C	C	C	C
62227	アラビア語 (ISO 8859-6) (タイプ 6)			X	X	C	C	C	C	C
62228	Windows アラビア語 (タイプ 6)		C	C	C	C	X	C	C	C
62230	ヘブライ語 (タイプ 8)			X	X	X	X	X	C	C
62231	ヘブライ語 (タイプ 8)			C	C	C	C	C	C	C
62232	ヘブライ語/ラテン (ISO 8859-8) (タイプ 8)			X	X	C	C	C	C	C
62236	ヘブライ語 (タイプ 10)			X	X	X	X	X	X	X
62238	ISO 8859-8 ヘブライ語/ラテン (タイプ 10)		C	C	C	C	X	C	C	C
62239	Windows ヘブライ語 (タイプ 10)		C	C	C	C	X	C	C	C
62241	ヘブライ語 (タイプ 11)			X	X	X	X	X	X	X
62242	ヘブライ語 (タイプ 11)			C	C	C	C	C	C	C
62243	ヘブライ語/ラテン (ISO 8859-8) (タイプ 11)			X	X	C	C	C	C	C
62244	Windows ヘブライ語 (タイプ 11)			C	C	C	X	C	C	C
62248	アラビア語 (タイプ 4)		C							

表 *150. PC* データおよび *ISO* グループ *1a (*非ラテン*-1 SBCS)* の国または地域用の *CCSID (*続き*)*

CCSID	説明	z/OS	i	AIX	HP	Sun	NT	SCO	SGI	Linux
ストリング・タイプ**:**										
4	表示 / 左から右 / 成形 / 対称スワッピング・オフ									
5	暗黙 / 左から右 / 非成形 / 対称スワッピング・オン									
6	暗黙 / 右から左 / 非成形 / 対称スワッピング・オン									
7	表示 / コンテキスト / 非成形 / 対称スワッピング・オフ									
8	表示 / 右から左 / 成形 / 対称スワッピング・オフ									
9	表示 / 右から左 / 成形 / 対称スワッピング・オン									
10	暗黙 / コンテキスト依存-左 / 非成形 / 対称スワッピング・オン									
11	暗黙 / コンテキスト依存-右 / 非成形 / 対称スワッピング・オン									
12	暗黙 / 右から左 / 成形 / 対称スワッピング・オン									

表 *151. EBCDIC グループ 2 (DBCS)* の国または地域用の *SBCS CCSID*

CCSID	説明	z/OS	i	AIX	HP	Sun	NT	SCO	SGI	Linux
290	日本カタカナ (拡張)	X	X	C	C	C	C	C	C	C
833	韓国 (拡張)	X	X	C	C	C	C	C	C	C
836	中国語 (簡体字) (拡張)	X	X	C	C	C	C	C	C	C
838	タイ (拡張)	X	X	C	C	C	C	C	C	C
1027	日本ローマ字 (拡張)	X	X	C	C	C	C	C	C	C
1130	ベトナム	X	X	C	C	C	C	C	C	C
1132	ラオ語	X	X							
1159	中国語 (繁体字) (ユーロ対応に拡張)			C	C	C	C	C	C	C
1160	タイ語 (ユーロ対応)	X	X	C	C	C	C	C	C	C
1164	ベトナム (ユーロ対応)	X	X	C	C	C	C	C	C	C
5123	日本ローマ字 (ユーロ対応)	X	X							
8482	日本カタカナ (ユーロ対応に拡張)	X		C	C	C	C	C	C	C
9030	タイ (拡張)	X	X							
13121	韓国、Windows	X	X							
13124	中国語 (繁体字)	X	X							
28709	中国語 (繁体字) (拡張)	X	X	C	C	C	C	C	C	C

表 152. PC データ・グループ 2 (DBCS) の国または地域用の SBCS CCSID

CCSID	説明	z/OS	i	AIX	HP	Sun	NT	SCO	SGI	Linux
367	韓国語および中国語 (簡体字) EUC	X	C	X			C			
874	タイ (拡張)	X	C	X	X		X			
891	韓国 (非拡張)	C	C							
895	日本 EUC - JISX201 ローマ字セット	C	C							
896	日本 EUC - JISX201 カタカナ・セット	C								
897	日本 (非拡張)	C	C							
903	中国語 (簡体字) (非拡張)	C	C							
904	中国語 (繁体字) (非拡張)	X	C							
1040	韓国 (拡張)	C	C							
1041	日本 (拡張)	X	C							
1042	中国語 (簡体字) (拡張)	C	C							
1043	中国語 (繁体字) (拡張)	X	C							
1088	韓国 (KS コード 5601-89)	X	C							
1114	中国語 (繁体字) (Big-5)	X	C							
1115	中国語 (簡体字) GB コード	X	C							
1126	韓国、Windows	X	C							
1129	ベトナム	X	C	X						
1133	ラオ語 ISO	X	C							
1162	タイ語 (拡張) (180 char) (ユーロ対応)	X								
1163	ISO ベトナム (ユーロ対応)	X								
1258	ベトナム	X	C			X				
4970	タイ (拡張)	X	C							
5210	中国語 (繁体字)	X	C							
9066	タイ (拡張)	X	C							

表 153. EBCDIC グループ 2 (DBCS) の国または地域用の DBCS CCSID

CCSID	説明	z/OS	i	AIX	HP	Sun	NT	SCO	SGI	Linux
300	日本 - 4370 のユーザー定義文字 (UDC) を含む	X	X	C	C	C	C	C	C	C
834	韓国 - 1880 UDC を含む	X	X	C	C	C	C	C	C	C
835	中国語 (繁体字) - 6204 UDC を含む	X	X	C	C	C	C	C	C	C
837	中国語 (簡体字) - 1880 UDC を含む	X	X	C	C	C	C	C	C	C
4396	日本 - 1880 UDC を含む	X	X	C	C	C	C	C	C	C
4930	韓国、Windows	X	X	C	C	C	C	C	C	C
4933	中国語 (簡体字)	X	X	C	C	C	C	C	C	C
9027	中国語 (繁体字) (ユーロ対応) - 6204 UDC を含む	C		C	C	C	C	C	C	C
16684	日本 (ユーロ対応)	X	X	C	C	C	C	C	C	C

表 154. PC データ・グループ 2 (DBCS) の国または地域用の DBCS CCSID

CCSID	説明	z/OS	i	AIX	HP	Sun	NT	SCO	SGI	Linux
301	日本 - 1880 UDC を含む	X	C	X	C	C	C	C	C	C
926	韓国 - 1880 UDC を含む	C	C							
927	中国語 (繁体字) - 6204 UDC を含む	X	C	C	C	C	C	C	C	C
928	中国語 (簡体字) - 1880 UDC を含む	C	C							
941	日本、Windows	X	C	C	C	C	X	C	C	C
947	中国語 (繁体字) (Big-5)	X	C	X	C	C	X	C	C	C
951	韓国 (KS コード 5601-89) - 1880 UDC を含む	X	C	C	C	C	X	C	C	C
952	日本 (EUC) X208-1990 セット	C								
953	日本 (EUC) X212-1990 セット	C								
971	韓国 (EUC) - 188 UDC を含む	X	C	X	X	X	C	C	C	C
1351	日本 HP-UX (J15)	X		C	X	C	C	C	C	C
1362	韓国、Windows	X	C	C	C	C	X	C	C	C
1380	中国語 (簡体字) (GB コード) - 1880 UDC を含む	X	C	C	C	C	X	X	C	C
1382	中国語 (簡体字) (EUC) - 1360 UDC を含む	X	C	X	X	X	C	X	C	C
1385	中国語 (繁体字)	X	C	C	C	C	X	C	C	C

表 *155. EBCDIC グループ 2 (DBCS)* の国または地域用の混合 *CCSID*

CCSID	説明	z/OS	i	AIX	HP	Sun	NT	SCO	SGI	Linux
930	日本カタカナ/漢字 (拡張) - 4370 UDC を含む	X	X	C	C	C	C	C	C	C
933	韓国 (拡張) - 1880 UDC を含む	X	X	C	C	C	C	C	C	C
935	中国語 (簡体字) (拡張) - 1880 UDC を含む	X	X	C	C	C	C	C	C	C
937	中国語 (繁体字) (拡張) - 4370 UDC を含む	X	X	C	C	C	C	C	C	C
939	日本ローマ字/漢字 (拡張) - 4370 UDC を含む	X	X	C	C	C	C	C	C	C
1364	韓国 (拡張)	X	X	C	C	C	C	C	C	C
1371	中国語 (繁体字) (ユーロ対応に拡張) - 4370 UDC を含む			C	C	C	C	C	C	C
1388	中国語 (簡体字)	X	X	C	C	C	C	C	C	C
1390	日本カタカナ/漢字 (ユーロ対応に拡張) - 4370 UDC を含む	X		C	C	C	C	C	C	C
1399	日本ローマ字 (ユーロ対応)	X	X						C	C
5026	日本カタカナ/漢字 (拡張) - 1880 UDC を含む	X	X	C	C	C	C	C	C	C
5035	日本ローマ字/漢字 (拡張) - 1880 UDC を含む	X	X	C	C	C	C	C	C	C

表 156. PC データ・グループ 2 (DBCS) の国または地域用の混合 CCSID

CCSID	説明	z/OS	i	AIX	HP	Sun	NT	SCO	SGI	Linux
932	日本 (非拡張) - 1880 UDC を含む	X	C	X	C	C	C	C	C	C
934	韓国 (非拡張) - 1880 UDC を含む		C							
936	中国語 (簡体字) (非拡張) - 1880 UDC を含む		C							
938	中国語 (繁体字) (非拡張) - 6204 UDC を含む	X	C	C	C	C	C	C	C	C
942	日本 (拡張) - 1880 UDC を含む	X	C	C	C	C	C	C	C	C
943	日本 NT	X	C	C	C	X	X	C	C	C
944	韓国 (拡張) - 1880 UDC を含む		C							
946	中国語 (簡体字) (拡張) - 1880 UDC を含む		C							
948	中国語 (繁体字) (拡張) - 6204 UDC を含む	X	C	C	C	C	C	C	C	C
949	韓国 (KS コード 5601-89) - 1880 UDC を含む	X	C	C	C	C	C	C	C	C
950	中国語 (繁体字) (Big-5)	X	C	X	X	X	X	C	C	X
954	日本 (EUC)		C	X	X	X	C	X	C	X
956	日本 2022 TCP		C							
957	日本 2022 TCP		C							
958	日本 2022 TCP		C							
959	日本 2022 TCP		C							
964	中国語 (繁体字) (EUC)		C	X	X	X	C	C	C	C
965	中国語 (繁体字) 2022 TCP		C							
970	韓国 EUC	X	C	X	X	X	C	C	X	X
1363	韓国、Windows	X	C	C	C	C	X	C	C	C
1381	中国語 (簡体字) GB コード	X	C	C	C	C	X	C	C	C
1383	中国語 (簡体字) EUC	X	C	X	X	X	C	X	C	X
1386	中国語 (簡体字)	X	C	X	C	C	X	C	C	C
1392	中国語 (簡体字) GB18030		C							
5039	日本 HP-UX (J15)	X		C	X	C	C	C	C	C
5050	日本 (EUC)		C							
5052	日本 2022 TCP		C							
5053	日本 2022 TCP		C							
5054	日本 2022 TCP		C							
5055	日本 2022 TCP		C							
5307	日本 HP-UX (J15) HISTORICAL	X								
17354	韓国 2022 TCP		C							

表 *156. PC* データ・グループ *2 (DBCS)* の国または地域用の混合 *CCSID (*続き*)*

CCSID	説明	z/OS	i	AIX	HP	Sun	NT	SCO	SGI	Linux
25546	韓国 2022 TCP		C							
33722	日本 EUC		C							

付録 F. Db2 for i のカタログ・ビュー

この付録では、Db2 for i のカタログに入っているビューについて説明します。

データベース・マネージャーは、それぞれのリレーショナル・データベース中のデータに関する情報が入っている一組の表を維持管理しています。これらの表をまとめてカタログ と呼びます。カタログ表 には、Db2 for i でサポートされる表、ユーザー定義関数、特殊タイプ、パラメーター、プロシージャー、パッケージ、ビュー、索引、別名、シーケンス、変数、制約、トリガー、XSR オブジェクト、および言語に関する情報が含まれています。カタログには、このシステムからアクセス可能なすべてのリレーショナル・データベースに関する情報も含まれています。

カタログ・ビューには次の 3 つのクラスがあります。

- IBM i のカタログ表およびビュー

  IBM i のカタログ表およびビューは、ANS および ISO のカタログ・ビューをモデルにしていますが、ANS および ISO のカタログ・ビューとまったく同じというわけではありません。iSeries のカタログ表およびビューは、Db2 for i の旧リリースと互換性があります。

  これらの表およびビューは、スキーマ QSYS および QSYS2 の中に入っています。

  カタログ表およびビューには、リレーショナル・データベース全体にわたるすべての表、パラメーター、プロシージャー、関数、特殊タイプ、パッケージ、XSR オブジェクト、ビュー、索引、別名、シーケンス、変数、トリガー、および制約に関する情報が含まれています。SQL スキーマの作成時に、そのスキーマにある表、パッケージ、ビュー、索引、および制約に関する情報のみが含まれているビューのサブセットが追加作成され、スキーマに組み込まれます。
- ODBC および JDBC のカタログ・ビュー

  ODBC および JDBC のカタログ・ビューは、ODBC および JDBC のメタデータ API 要求を満たすように設計されています (例えば、SQLCOLUMNS)。これらのビューは、Db2 LUW バージョン 8 のビューと互換性があります。ODBC または JDBC がそのメタデータ API を拡張または変更すると、それに応じてこれらのビューも変更されます。

  これらのビューはスキーマ SYSIBM の中に入っています。
- ANS および ISO のカタログ・ビュー

  ANS および ISO のカタログ・ビューは、ANS および ISO の SQL 標準 (情報スキーマ (Information Schema) カタログ・ビュー) に準拠するよう設計されています。これらのビューは、ANS および ISO 標準が拡張または変更されると、それに応じて変更されます。

  ビューには、将来、標準が拡張された場合に備えて予約されている列がいくつか含まれています。

これらのビューには、次の 2 つのバージョンがあります。

– これらのビューの最初のバージョンは、スキーマ INFORMATION_SCHEMA[160]の中に入っています。ユーザーが特定の特権を持っているオブジェクトに関連した行のみがビューに含まれます。このバージョンは、ANS および ISO SQL 標準と互換性があります。

  このカタログ・ビューのセットを使用して、ユーザーが特権を持っていないオブジェクトに関する情報を参照できないようにするには、他のカタログ・ビューに対する特権をユーザーおよび PUBLIC から取り消す必要があります。

– これらのビューの 2 番目のバージョンはスキーマ SYSIBM の中に入っています。ユーザーがビュー内の行に関連したオブジェクトに対する特権を持っているかどうかにかかわらず、すべての行がこれらのビューに含まれます。これらのビューは Db2 LUW バージョン 8 と互換性があり、一般に QSYS2 内の ANS および ISO ビューよりパフォーマンスが良好です。

例えば、ユーザーが QSYS2.TABLES および SYSIBM.TABLES カタログ・ビューに対する SELECT 特権を持っており、WORK.EMPLOYEE という表に対する特権は持っていないものと想定します。以下の SQL ステートメントは結果行を戻しません。

```
SELECT *
  FROM QSYS2.TABLES
  WHERE TABLE_SCHEMA = 'WORK' AND TABLE_NAME = 'EMPLOYEE'
```

ただし、以下の SQL ステートメントは結果行を戻します。

```
SELECT *
  FROM SYSIBM.TABLES
  WHERE TABLE_SCHEMA = 'WORK' AND TABLE_NAME = 'EMPLOYEE'
```

注: これらのビューのいくつかでは、ビュー定義の一部として、特殊なカタログ関数を使用します。これらの関数は SYSIBM および QSYS2 に存在します。これらの関数は、将来のリリースまたはフィックスパックで変更されて非互換になる可能性があるため、アプリケーション内で直接使用する場合は注意が必要です。

注

カタログ内の名前: 一般に、カタログ表の列に格納されるすべての名前は、区切り文字なしで、大文字小文字の区別があります。例えば、次の表が作成されたとします。

```
CREATE TABLE "colname"/"long_table_name"
                        ("long_column_name" CHAR(10),
                         INTCOL INTEGER)
```

SQL 名とシステム名の間のマッピングに関する情報を戻すには、次の選択ステートメントを使用できます。

160. INFORMATION_SCHEMA は、カタログ・ビューが含まれる ANS および ISO SQL 標準スキーマ名です。これは QSYS2 の同義語です。

```
SELECT TABLE_NAME, SYSTEM_TABLE_NAME, COLUMN_NAME, SYSTEM_COLUMN_NAME
    FROM QSYS2/SYSCOLUMNS
    WHERE TABLE_NAME = 'long_table_name' AND
          TABLE_SCHEMA = 'colname'
```

次の行が戻されます。

TABLE_NAME	SYSTEM_TABLE_NAME	COLUMN_NAME	SYSTEM_COLUMN_NAME
long_table_name	"long0001"	long_column_name	LONG_00001
long_table_name	"long0001"	INTCOL	INTCOL

カタログ内のシステム名**:** 通常は、短いシステム列名より、長い SQL 列名を使用してください。 IBM i のカタログ表およびビュー用の短いシステム列名は、旧リリースおよび他の Db2 プロダクトとの互換性を確保するために明示指定できるように維持されているものです。ODBC と JDBC のカタログ・ビュー、および ANS と ISO のカタログ・ビュー用の短いシステム列名は、明示的には維持されず、リリース間で変わる可能性があります。

カタログ内の **NULL** 値**:** 列の情報が適用されない場合は、NULL 値が戻されます。上記で作成された表を使用すると、次の選択ステートメントで NUMERIC_SCALE および CHARACTER_MAXIMUM_LENGTH を照会し、データが列のデータ・タイプに適用されなかった場合は、NULL 値が戻されます。

```
SELECT COLUMN_NAME, NUMERIC_SCALE, CHARACTER_MAXIMUM_LENGTH
  FROM QSYS2/SYSCOLUMNS
  WHERE TABLE_NAME = 'long_table_name'    AND
        TABLE_SCHEMA = 'colname'
```

次の行が戻されます。

COLUMN_NAME	NUMERIC_SCALE	CHARACTER_MAXIMUM_LENGTH
long_column_name	?	10
INTCOL	0	?

数値の位取りは文字の列では無効であるため、"long_column_name" の列の NUMERIC_SCALE には NULL 値が戻されます。文字長は数値の列では無効であるため、INTCOL の列の CHARACTER_MAXIMUM_LENGTH には NULL 値が戻されます。

インストールとバックアップに関する考慮事項**:** ある種のカタログ表、およびカタログ表とビューに関して作成されたビューは、定期的に保管してください。

- カタログ表 QSYS.QADBXRDBD には、リレーショナル・データベース情報が入っています。この表は定期的に保管する必要があります。
- ILE の外部関数またはプロシージャー、または SQL の関数またはプロシージャーを復元すると、これらのカタログ表に情報が自動的に挿入されます。ただし、非 ILE 外部関数およびプロシージャーの場合には、情報の自動挿入は行われません。非 ILE 外部関数またはプロシージャーの定義をバックアップするには、カタログ表 SYSROUTINES および SYSPARMS を確実に保管するか、またはこれらの関数およびプロシージャーを作成するために使用した SQL ソース・ステートメントのバックアップを必ずとっておくようにしてください。

- スキーマ QSYS2 または SYSIBM 内のカタログ・ビューは、すべてシステム・オブジェクトです。つまり、これらのカタログ・ビューに関して作成されたユーザー・ビューは、オペレーティング・システムのインストール時にすべて削除されます。従属オブジェクトも、すべて削除されます。この削除要件を避けるために、インストールの前にビューを保管しておき、後で復元することができます。
- QSYS ライブラリー内のカタログ表もシステム・オブジェクトです。しかし、QSYS ライブラリー内のカタログ表は、インストール時には削除されません。したがって、これらの表に関して作成されたビューはすべて、インストール・プロセスの終了後も保存されています。

カタログ・ビューへの特権の認可**:** カタログの表およびビューは、他のデータベース表およびデータベース・ビューと類似しています。権限を持っているユーザーであれば、他の表からデータを検索するときと同じように、SQL ステートメントを使用してカタログ・ビューのデータを見ることができます。カタログの表およびビューでは、出荷時に、PUBLIC に SELECT 特権が認可されています。この特権を取り消して、個々のユーザーに SELECT 特権を認可することができます。

QSYS カタログ表**:** ほとんどのカタログ・ビューは、QSYS ライブラリー (データベース相互参照ファイルとも言う) の中の次の表に基づいています。これらの表は、出荷時に SELECT 特権は PUBLIC には認可されていません。また、これらの表を直接使用してはなりません。

QADBCCST	QADBPKG	QADBXSFLD
QADBFDEP	QADBXCTLS	QADBXTRIGB
QADBFCST	QADBXCTLSD	QADBXTRIGC
QADBIFLD	QADBXRDBD	QADBXTRIGD
QADBKFLD	QADBXREF	

SELECT * の使用**:** 新規の機能がインプリメントされ、 ISO/ANSI 標準が変化するにつれて、新しい列がカタログ内の表およびビューに追加されます。そのため、ご使用のアプリケーションがこれらの新規の列を許容できる場合を除いて、カタログ表およびカタログ・ビューにアクセスする際に SELECT * を使用しないことをお勧めします。

IBM i のカタログ表およびビュー

IBM i カタログには、このセクションで示す QSYS2 スキーマのビューと表が含まれています。

Db2 for i の名前	対応する **ANSI/ISO** の名前	説明
1919 ページの『SYSCATALOGS』	CATALOGS	リレーショナル・データベースについての情報
1920 ページの『SYSCHKCST』	CHECK_CONSTRAINTS	検査制約についての情報
1921 ページの『SYSCOLAUTH』	COLUMN_PRIVILEGES	列特権についての情報
1922 ページの『SYSCOLUMNS』	COLUMNS	列属性についての情報
1930 ページの『SYSCOLUMNS2』		列属性についての情報
1941 ページの『SYSCOLUMNSTAT』		列統計についての情報
1944 ページの『SYSCONTROLS』		行の許可および列マスクについての情報
1946 ページの『SYSCONTROLSDEP』		行の許可および列マスクの従属関係についての情報
1947 ページの『SYSCST』	TABLE_CONSTRAINTS	すべての制約についての情報
1949 ページの『SYSCSTCOL』	CONSTRAINT_COLUMN_USAGE	制約で参照される列についての情報
1950 ページの『SYSCSTDEP』	CONSTRAINT_TABLE_USAGE	表に関する制約従属関係についての情報
1951 ページの『SYSFIELDS』		フィールド・プロシージャーについての情報
1956 ページの『SYSFUNCS』	ROUTINES	ユーザー定義関数についての情報
1962 ページの『SYSHISTORYTABLES』		履歴表についての情報
1963 ページの『SYSINDEXES』		索引についての情報
1965 ページの『SYSINDEXSTAT』		索引統計についての情報
1971 ページの『SYSJARCONTENTS』		Java ルーチンの jar についての情報
1972 ページの『SYSJAROBJECTS』		Java ルーチンの jar についての情報
1973 ページの『SYSKEYCST』	KEY_COLUMN_USAGE	固有、基本、および外部キーについての情報
1974 ページの『SYSKEYS』		索引キーについての情報
1975 ページの『SYSMQTSTAT』		マテリアライズ照会表統計についての情報
1979 ページの『SYSPACKAGE』		パッケージについての情報
1981 ページの『SYSPACKAGEAUTH』		パッケージ特権についての情報
1982 ページの『SYSPACKAGESTAT』		パッケージ統計についての情報
1989 ページの『SYSPACKAGESTMTSTAT』		パッケージ内の SQL ステートメントについての情報
1991 ページの『SYSPARMS』	PARAMETERS	ルーチン・パラメーターについての情報
1995 ページの『SYSPARTITIONDISK』		パーティションのディスク使用についての情報
1997 ページの『SYSPARTITIONINDEXDISK』		索引のディスク使用についての情報
2000 ページの『SYSPARTITIONINDEXES』		パーティション索引についての情報
2007 ページの『SYSPARTITIONINDEXSTAT』		パーティション索引統計についての情報
2013 ページの『SYSPARTITIONMQTS』		パーティション・マテリアライズ照会表についての情報
2017 ページの『SYSPARTITIONSTAT』		パーティション統計についての情報
2021 ページの『SYSPERIODS』		期間についての情報
2023 ページの『SYSPROCS』	ROUTINES	プロシージャーについての情報
2027 ページの『SYSPROGRAMSTAT』		SQL ステートメントを含むプログラム、サービス・プログラム、およびモジュールについての情報

Db2 for i の名前	対応する ANSI/ISO の名前	説明
2038 ページの『SYSPROGRAMSTMTSTAT』		プログラム、サービス・プログラム、およびモジュールに組み込まれた SQL ステートメントについての情報
2041 ページの『SYSREFCST』	REFERENTIAL_CONSTRAINTS	参照制約についての情報
2042 ページの『SYSROUTINEAUTH』	ROUTINE_PRIVILEGES	ルーチン特権についての情報
2043 ページの『SYSROUTINEDEP』	ROUTINE_TABLE_USAGE	関数およびプロシージャーの従属関係についての情報
2045 ページの『SYSROUTINES』	ROUTINES	関数およびプロシージャーについての情報
2052 ページの『SYSSCHEMAAUTH』		スキーマ特権についての情報
2053 ページの『SYSSCHEMAS』		スキーマについての情報
2054 ページの『SYSSEQUENCEAUTH』	USAGE_PRIVILEGES	シーケンス特権についての情報
2055 ページの『SYSSEQUENCES』		シーケンスについての情報
2058 ページの『SYSTABLEDEP』		マテリアライズ照会表の従属関係についての情報
2059 ページの『SYSTABLEINDEXSTAT』		表索引統計についての情報
2057 ページの『SYSTABAUTH』	TABLE_PRIVILEGES	表特権についての情報
2065 ページの『SYSTABLES』	TABLES	表およびビューについての情報
2069 ページの『SYSTABLESTAT』		表統計についての情報
2072 ページの『SYSTRIGCOL』	TRIGGER_COLUMN_USAGE	トリガーで使用される列についての情報
2073 ページの『SYSTRIGDEP』	TRIGGER_TABLE_USAGE	トリガーで使用されるオブジェクトについての情報
2074 ページの『SYSTRIGGERS』	TRIGGERS	トリガーについての情報
2078 ページの『SYSTRIGUPD』	TRIGGERED_UPDATE_COLUMNS	トリガーの WHEN 文節内の列についての情報
2079 ページの『SYSTYPES』	USER_DEFINED_TYPES	組み込みのデータ・タイプおよび特殊タイプについての情報
2085 ページの『SYSUDTAUTH』	UDT_PRIVILEGES	タイプ特権についての情報
2086 ページの『SYSVARIABLEAUTH』		グローバル変数特権についての情報
2087 ページの『SYSVARIABLEDEP』		グローバル変数で使用されるオブジェクトについての情報
2088 ページの『SYSVARIABLES』		グローバル変数についての情報
2094 ページの『SYSVIEWDEP』	VIEW_TABLE_USAGE	表のビューの従属関係についての情報
2096 ページの『SYSVIEWS』	VIEWS	ビューの定義についての情報
2098 ページの『SYSXSROBJECTAUTH』	USAGE_PRIVILEGES	XML スキーマ特権についての情報
2099 ページの『XSRANNOTATIONINFO』		注釈についての情報
2100 ページの『XSROBJECTCOMPONENTS』		XML スキーマのコンポーネントについての情報
2101 ページの『XSROBJECTHIERARCHIES』		XML スキーマの文書階層関係についての情報
2102 ページの『XSROBJECTS』		XML スキーマについての情報

SYSCATALOGS

SYSCATALOGS ビューには、ユーザーが接続できる各リレーショナル・データベースごとに、行が 1 つずつ入ります。次の表は、SYSCATALOGS ビューの列について説明しています。

表 *157. SYSCATALOGS* ビュー

列名	システム列名	データ・タイプ	説明
CATALOG_NAME	LOCATION	VARCHAR(18)	リレーショナル・データベース名。
CATALOG_STATUS	RDBASPSTAT	CHAR(10)	リレーショナル・データベースの状況。 **ACTIVE** リレーショナル・データベースは、アクティブな独立補助記憶域プール (IASP) に関連付けられていますが、まだ使用可能ではありません。 **AVAILABLE** リレーショナル・データベースは使用可能です。 **VARYOFF** リレーショナル・データベースは、オフに変更された独立補助記憶域プール (IASP) に関連付けられています。 **VARYON** リレーショナル・データベースは、オンに変更された独立補助記憶域プール (IASP) に関連付けられていますが、まだ使用可能ではありません。 **UNKNOWN** リレーショナル・データベースの状況は不明です。リモート・リレーショナル・データベースの状況は、常に不明です。
CATALOG_TYPE	RDBTYPE	CHAR(7)	リレーショナル・データベースのタイプ。 **LOCAL** リレーショナル・データベースは、このシステムにとってローカルのデータベースです。 **MIRROR** リレーショナル・データベースは、Db2 ミラー関係に定義されています。 **REMOTE** リレーショナル・データベースは、リモート・システム上にあります。
CATALOG_ASPGRP	RDBASPGRP	VARCHAR(10) NULL 可能	独立補助記憶域プール (IASP) の名前。 リレーショナル・データベースの状況が UNKNOWN の場合は、NULL 値が入ります。
CATALOG_ASPNUM	RDBASPNUM	VARCHAR(10) NULL 可能	独立補助記憶域プール (IASP) の番号。 リレーショナル・データベースの状況が UNKNOWN の場合は、NULL 値が入ります。
CATALOG_TEXT	RDBTEXT	VARGRAPHIC(50) CCSID 1200 NULL 可能	リレーショナル・データベースのテキスト記述。

SYSCHKCST

SYSCHKCST ビューには、SQL のスキーマにある各検査制約ごとに、行が 1 つずつ入ります。 次の表は、SYSCHKCST ビューの列について説明しています。

表 *158. SYSCHKCST* ビュー

列名	システム列名	データ・タイプ	説明
CONSTRAINT_SCHEMA	DBNAME	VARCHAR(128)	該当の制約が入っているスキーマの名前。
CONSTRAINT_NAME	RELNAME	VARCHAR(128)	制約の名前
CHECK_CLAUSE	CHECK	VARGRAPHIC(2000) CCSID 1200 NULL 可能	検査制約文節のテキスト 切り捨てなければ検査文節を表示できない場合は、NULL 値が入ります。
ROUNDING_MODE	DECFLTRND	CHAR(1) NULL 可能	検査制約用の丸めモードは以下のとおりです。 **C** ROUND_CEILING **D** ROUND_DOWN **F** ROUND_FLOOR **G** ROUND_HALF_DOWN **E** ROUND_HALF_EVEN **H** ROUND_HALF_UP **U** ROUND_UP
SYSTEM_CONSTRAINT_SCHEMA	SYS_CDNAME	CHAR(10)	制約が入っているシステム・スキーマの名前。
INSERT_ACTION	INS_ACTION	VARGRAPHIC(500) CCSID 1200 NULL 可能	挿入違反処置が定義されている場合、ON INSERT 節のテキストが入ります。指定されていない場合は NULL 値が入ります。
UPDATE_ACTION	UPD_ACTION	VARGRAPHIC(500) CCSID 1200 NULL 可能	更新違反処置が定義されている場合、ON UPDATE 節のテキストが入ります。指定されていない場合は NULL 値が入ります。

SYSCOLAUTH

SYSCOLAUTH ビューには、列に対して認可された各特権ごとに、行が 1 つずつ入ります。このカタログ・ビューを使用して、特定ユーザーが特定の列に対する権限を持っているかどうかを判別することはできないので、注意してください。なぜなら、列を使用するための特権は、グループ・ユーザー・プロファイルまたは特殊権限 (*ALLOBJ など) を通じて獲得できるからです。さらに、列を使用するための特権は、表に関して付与された特権によっても獲得されます。

次の表は、SYSCOLAUTH ビューの列について説明しています。

表 *159. SYSCOLAUTH* ビュー

列名	システム列名	データ・タイプ	説明
GRANTOR	GRANTOR	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
GRANTEE	GRANTEE	VARCHAR(128)	特権を認可する対象のユーザー・プロファイル。
TABLE_SCHEMA	DBNAME	VARCHAR(128)	スキーマの名前
TABLE_NAME	TBNAME	VARCHAR(128)	表の名前。
COLUMN_NAME	NAME	VARCHAR(128)	列の名前。
PRIVILEGE_TYPE	PRIVTYPE	VARCHAR(10)	認可される特権 : **UPDATE** 列を更新する特権。 **REFERENCES** 参照制約モードで列を参照する特権。
IS_GRANTABLE	GRANTABLE	VARCHAR(3)	特権を他のユーザーに認可できるかどうかを示します。 **NO** 特権は認可できません。 **YES** 特権を認可できます。
AUTHORIZATION_LIST	AUTL	VARCHAR(10) NULL 可能	NULL 値が入ります。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	スキーマのシステム名
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	表またはビューのシステム名
SYSTEM_COLUMN_NAME	SYS_CNAME	CHAR(10)	列のシステム名

SYSCOLUMNS

SYSCOLUMNS ビューには、SQL スキーマの中の各表または各ビューの各列 (SQL カタログの列を含む) ごとに行が 1 つずつ入ります。

次の表は、SYSCOLUMNS ビューの列について説明しています。

表 *160. SYSCOLUMNS* ビュー

列名	システム列名	データ・タイプ	説明
COLUMN_NAME	NAME	VARCHAR(128)	列の名前。 SQL の列名が存在する場合は、その SQL の列名になります。存在しない場合は、システムの列名になります。
TABLE_NAME	TBNAME	VARCHAR(128)	該当の列を含む表またはビューの名前。 SQL の表名またはビュー名が存在する場合は、その SQL の表名またはビュー名です。存在しない場合は、システムの表名またはビュー名になります。
TABLE_OWNER	TBCREATOR	VARCHAR(128)	表またはビューの所有者。
ORDINAL_POSITION	COLNO	INTEGER	表またはビューにおける該当の列の数値位置 (左から右への順序)。

表 *160. SYSCOLUMNS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
DATA_TYPE	COLTYPE	VARCHAR(8)	列のタイプ: **BIGINT** 大整数 **INTEGER** 長整数 **SMALLINT** 短整数 **DECIMAL** パック 10 進数 **NUMERIC** ゾーン 10 進数 **FLOAT** 浮動小数点数; FLOAT、REAL、または DOUBLE PRECISION **DECFLOAT** 10 進浮動小数点数 **CHAR** 固定長文字ストリング **VARCHAR** 可変長文字ストリング **CLOB** 文字ラージ・オブジェクト・ストリング **GRAPHIC** 固定長グラフィック・ストリング **VARG** 可変長グラフィック・ストリング **DBCLOB** 2 バイト文字ラージ・オブジェクト・ストリング **BINARY** 固定長バイナリー・ストリング **VARBIN** 可変長バイナリー・ストリング **BLOB** バイナリー・ラージ・オブジェクト・ストリング **DATE** 日付 **TIME** 時刻 **TIMESTMP** タイム・スタンプ **DATALINK** データ・リンク **ROWID** 行 ID **XML** XML **DISTINCT** 特殊タイプ

表 *160. SYSCOLUMNS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
LENGTH	LENGTH	INTEGER	列の長さ属性。ただし、10 進数、数値、または非ゼロ精度 2 進数の列の場合は、その精度: **8** バイト BIGINT **4** バイト INTEGER **2** バイト SMALLINT 数の精度 DECIMAL 数の精度 NUMERIC **8** バイト FLOAT、FLOAT(n) (ここで n = 25 から 53)、または DOUBLE PRECISION **4** バイト FLOAT(n) (ここで n = 1 から 24)、または REAL **8** バイト DECFLOAT(16) **16** バイト DECFLOAT(34) ストリングの長さ CHAR ストリングの最大長 VARCHAR または CLOB グラフィック・ストリングの長さ GRAPHIC グラフィック・ストリングの最大長 VARGRAPHIC または DBCLOB ストリングの長さ BINARY **2** 進ストリングの最大長 VARBIN または BLOB **4** バイト DATE **3** バイト TIME **((*p*+1)/2) + 7 (*p* はタイム・スタンプの精度)** の整数部分 TIMESTAMP データ・リンク **URL** およびコメントの最大長 DATALINK **40** バイト ROWID **2147483647** バイト XML ソース・タイプと同じ値 DISTINCT
NUMERIC_SCALE	SCALE	INTEGER NULL 可能	数値データの位取り。 列が 10 進数、数値、または 2 進数でない場合は、NULL 値が入ります。
IS_NULLABLE	NULLS	CHAR(1)	列に NULL 値を入れることが可能かどうか: **N** 不可 **Y** 可能

表 *160. SYSCOLUMNS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
IS_UPDATABLE	UPDATES	CHAR(1)	列が更新可能かどうか: **N** 不可 **Y** 可能
LONG_COMMENT	REMARKS	VARGRAPHIC(2000) CCSID 1200	COMMENT ステートメントで指定された文字ストリング。
		NULL 可能	詳細コメントがない場合は、NULL 値が入ります。
HAS_DEFAULT	DEFAULT	CHAR(1)	列がデフォルト値を持つ (DEFAULT 文節または NULL 使用可能) かどうか: **N** 不可 **Y** 可能 **A** 列は、ROWID データ・タイプおよび GENERATED ALWAYS 属性を持ちます。 **D** 列は、ROWID データ・タイプおよび GENERATED BY DEFAULT 属性を持ちます。 **E** 列は、FOR EACH ROW ON UPDATE 属性および GENERATED ALWAYS 属性により定義されます。 **F** 列は、FOR EACH ROW ON UPDATE 属性および GENERATED BY DEFAULT 属性により定義されます。 **I** 列は、AS IDENTITY 属性および GENERATED ALWAYS 属性により定義されます。 **J** 列は、AS IDENTITY 属性および GENERATED 属性により定義されます。 **Q** 列は GENERATED AS ROW BEGIN 属性を使って定義される。 **R** 列は GENERATED AS ROW END 属性を使って定義される。 **X** 列は GENERATED AS TRANSACTION START ID 属性を使って定義される。 **a** 列は、特殊レジスターを使用して生成された式として定義されます。 **c** 列は、グローバル変数を使用して生成された式として定義されます。 **d** 列は DATA CHANGE OPERATION を使用して生成された式として定義されます。 列がビューに関するものである場合、N が戻されます。
COLUMN_HEADING	LABEL	VARGRAPHIC(60) CCSID 1200	LABEL ステートメント (列見出し) で指定された文字ストリング。
		NULL 可能	列見出しがない場合は、NULL 値が入ります。

表 *160. SYSCOLUMNS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
STORAGE	STORAGE	INTEGER	列の記憶域所要量: **8** バイト　BIGINT **4** バイト　INTEGER **2** バイト　SMALLINT **(**精度**/2) + 1** DECIMAL 数の精度　NUMERIC **8** バイト　FLOAT、FLOAT(n) (ここで n = 25 から 53)、または DOUBLE PRECISION **4** バイト　FLOAT(n) (ここで n = 1 から 24)、または REAL **8** バイト　DECFLOAT(16) **16** バイト DECFLOAT(34) ストリングの長さ CHAR または BINARY ストリングの最大長 **+ 2** VARCHAR または VARBIN ストリングの最大長 **+ 29** CLOB または BLOB ストリングの長さ *** 2** GRAPHIC ストリングの最大長 *** 2 + 2** VARGRAPHIC ストリングの最大長 *** 2 + 29** DBCLOB **4** バイト　DATE **3** バイト　TIME **((***p***+1)/2) + 7 (***p* はタイム・スタンプの精度**)** の整数部分　TIMESTAMP データ・リンク **URL** およびコメントの最大長 **+ 24**　DATALINK **42** バイト ROWID **2147483647** バイト **+ 29** XML ソース・タイプと同じ値 DISTINCT **注:** この列には、すべてのデータ・タイプの記憶域所要量があります。
NUMERIC_PRECISION	PRECISION	INTEGER NULL 可能	数値の列すべての精度。 **注:** この列では、すべての数値データ・タイプ (10 進浮動小数点数、単精度および倍精度の浮動小数点数を含む) の精度を指定します。NUMERIC_PRECISION_RADIX 列は、この列の値が 2 進数であるか、または 10 進数であるかを示します。 列が数値の列でない場合は、NULL 値が入ります。

表 *160. SYSCOLUMNS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
CCSID	CCSID	INTEGER NULL 可能	CHAR、VARCHAR、CLOB、DATE、TIME、TIMESTAMP、GRAPHIC、VARGRAPHIC、DBCLOB、XML、および DATALINK 列の CCSID の値。 列が BINARY、VARBIN、BLOB または ROWID の場合は、65535 が入ります。 列が数値データ・タイプの場合は、NULL 値が入ります。
TABLE_SCHEMA	DBNAME	VARCHAR(128)	該当の表またはビューが入っている SQL スキーマの名前。
COLUMN_DEFAULT	DFTVALUE	VARGRAPHIC(2000) CCSID 1200 NULL 可能	列のデフォルト値が存在する場合は、そのデフォルト値。列のデフォルト値が、切り捨てなければ表示できない場合は、その列の値はストリング 'TRUNCATED' になります。デフォルト値は文字形式で保管されます。以下の特殊値も存在します。 **CURRENT_DATE** デフォルト値は、現在の日付です。 **CURRENT_TIME** デフォルト値は、現在の時刻です。 **CURRENT_TIMESTAMP** デフォルト値は、現在のタイム・スタンプです。 **NULL** デフォルト値は NULL 値になり、DEFAULT NULL が明示的に指定されています。 **USER** デフォルト値は、現在のジョブ・ユーザーです。 *special-register* 列 HAS_DEFAULT に値 'a' が含まれている場合、特殊レジスターの名前。 *global-variable* 列 HAS_DEFAULT に値 'c' が含まれている場合、グローバル変数の修飾名。 **DATA CHANGE OPERATION** 列 HAS_DEFAULT に値 'd' が含まれている場合。 以下の場合、NULL 値が入ります。 • 列にデフォルト値がない場合 (例えば、列に IDENTITY 属性が指定されている場合、列が行 ID である場合、または列が行変更タイム・スタンプ列、行開始列、行終了列、またはトランザクション開始 ID 列である場合)、または • DEFAULT 値が明示的に指定されていない場合。
CHARACTER_MAXIMUM_LENGTH	CHARLEN	INTEGER NULL 可能	データ・タイプが 2 進ストリング、文字ストリング、グラフィック・ストリング、および XML の場合は、ストリングの最大長。 列がストリングでない場合は、NULL 値が入ります。

表 *160. SYSCOLUMNS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
CHARACTER_OCTET_LENGTH	CHARBYTE	INTEGER NULL 可能	データ・タイプが 2 進ストリング、文字ストリング、グラフィック・ストリング、および XML の場合は、バイト数。 列がストリングでない場合は、NULL 値が入ります。
NUMERIC_PRECISION_RADIX	RADIX	INTEGER NULL 可能	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 列が数値の列でない場合は、NULL 値が入ります。
DATETIME_PRECISION	DATPRC	INTEGER NULL 可能	日付、時刻、またはタイム・スタンプの小数部分。 **0** データ・タイプが DATE および TIME の場合 **0-12** データ・タイプが TIMESTAMP の場合 (小数秒) 列が日付、時刻、またはタイム・スタンプの列でない場合は、NULL 値が入ります。
COLUMN_TEXT	LABELTEXT	VARGRAPHIC(50) CCSID 1200 NULL 可能	LABEL ステートメント (列テキスト) で指定された文字ストリング。 列テキストがない場合は、NULL 値が入ります。
SYSTEM_COLUMN_NAME	SYS_CNAME	CHAR(10)	列のシステム名
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	表またはビューのシステム名
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	スキーマのシステム名
USER_DEFINED_TYPE_SCHEMA	TYPESCHEMA	VARCHAR(128) NULL 可能	特殊タイプの場合は、スキーマの名前。 列が特殊タイプでない場合は、NULL 値が入ります。
USER_DEFINED_TYPE_NAME	TYPENAME	VARCHAR(128) NULL 可能	特殊タイプの名前。 列が特殊タイプでない場合は、NULL 値が入ります。
IS_IDENTITY	IDENTITY	VARCHAR(3)	この列は、列が識別列かどうかを指定します。 **NO** 列は識別列ではありません。 **YES** 列は識別列です。
IDENTITY_GENERATION	GENERATED	VARCHAR(10) NULL 可能	この列は、列が GENERATED ALWAYS か GENERATED BY DEFAULT かを識別します。 **ALWAYS** 列の値は常に生成されます。 **BY DEFAULT** 列の値はデフォルトにより生成されます。 この列が、ROWID 列、ID 列、行変更タイム・スタンプ列、行開始列、行終了列、トランザクション開始 ID 列、および生成式列でない場合、NULL 値が入ります。

表 *160. SYSCOLUMNS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
IDENTITY_START	START	DECIMAL(31,0)	識別列の開始値。
		NULL 可能	列が IDENTITY 列でない場合は、NULL 値が入ります。
IDENTITY_INCREMENT	INCREMENT	DECIMAL(31,0)	識別列の増分値。
		NULL 可能	列が IDENTITY 列でない場合は、NULL 値が入ります。
IDENTITY_MINIMUM	MINVALUE	DECIMAL(31,0)	識別列の最小値。
		NULL 可能	列が IDENTITY 列でない場合は、NULL 値が入ります。
IDENTITY_MAXIMUM	MAXVALUE	DECIMAL(31,0)	識別列の最大値。
		NULL 可能	列が IDENTITY 列でない場合は、NULL 値が入ります。
IDENTITY_CYCLE	CYCLE	VARCHAR(3)	この列は、識別列の値が最小値または最大値に達した後も、値の生成を続けるかどうかを識別します。
		NULL 可能	**NO** 値の生成は継続されません。 **YES** 値の生成は継続されます。 列が IDENTITY 列でない場合は、NULL 値が入ります。
IDENTITY_CACHE	CACHE	INTEGER	アクセスを高速化するために事前割り振りが可能な識別値の数を指定します。ゼロは、値が事前割り振りされないことを示します。
		NULL 可能	列が IDENTITY 列でない場合は、NULL 値が入ります。
IDENTITY_ORDER	ORDER	VARCHAR(3)	識別値を要求された順序で生成しなければならないかどうかを指定します。
		NULL 可能	**NO** 値は、要求された順序で生成する必要はありません。 **YES** 値は、要求された順序で生成する必要があります。 列が IDENTITY 列でない場合は、NULL 値が入ります。
COLUMN_EXPRESSION	EXPRESSION	DBCLOB(2097152)	列が式の場合は、その式が入ります。
		CCSID 1200 NULL 可能	列が式でない場合は、NULL 値が入ります。
HIDDEN	HIDDEN	CHAR(1)	列が暗黙の列リストに含まれるかどうかを指定します。
			P 部分的に隠されている。 **N** 隠されていない。
HAS_FLDPROC	FLDPROC	CHAR(1)	列にフィールド・プロシージャーがあるかどうかを示します。
			N 列にフィールド・プロシージャーがありません。 **Y** 列にフィールド・プロシージャーがあります。

SYSCOLUMNS2

SYSCOLUMNS2 ビューには、SQL スキーマの中の各表または各ビューの各列 (SQL カタログの列を含む) ごとに行が 1 つずつ入ります。

単一の表またはビューについての情報を得るには、 SYSCOLUMNS2 を使用する照会のほうが、SYSCOLUMNS を照会するよりも良好な結果になるのが一般的です。また、SYSCOLUMNS2 には SYSCOLUMNS よりも少し多くの属性が含まれます。

以下の表では、SYSCOLUMNS2 ビューの列について説明します。

表 *161. SYSCOLUMNS2* ビュー

列名	システム列名	データ・タイプ	説明
COLUMN_NAME	NAME	VARCHAR(128)	列の名前。 SQL の列名が存在する場合は、その SQL の列名になります。存在しない場合は、システムの列名になります。
TABLE_NAME	TBNAME	VARCHAR(128)	該当の列を含む表またはビューの名前。 SQL の表名またはビュー名が存在する場合は、その SQL の表名またはビュー名です。存在しない場合は、システムの表名またはビュー名になります。
TABLE_OWNER	TBCREATOR	VARCHAR(128)	表またはビューの所有者。
ORDINAL_POSITION	COLNO	INTEGER	表またはビューにおける該当の列の数値位置 (左から右への順序)。

表 *161. SYSCOLUMNS2* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
DATA_TYPE	COLTYPE	VARCHAR(8)	列のタイプ: **BIGINT** 大整数 **INTEGER** 長整数 **SMALLINT** 短整数 **DECIMAL** パック 10 進数 **NUMERIC** ゾーン 10 進数 **FLOAT** 浮動小数点数; FLOAT、REAL、または DOUBLE PRECISION **DECFLOAT** 10 進浮動小数点数 **CHAR** 固定長文字ストリング **VARCHAR** 可変長文字ストリング **CLOB** 文字ラージ・オブジェクト・ストリング **GRAPHIC** 固定長グラフィック・ストリング **VARG** 可変長グラフィック・ストリング **DBCLOB** 2 バイト文字ラージ・オブジェクト・ストリング **BINARY** 固定長バイナリー・ストリング **VARBIN** 可変長バイナリー・ストリング **BLOB** バイナリー・ラージ・オブジェクト・ストリング **DATE** 日付 **TIME** 時刻 **TIMESTMP** タイム・スタンプ **DATALINK** データ・リンク **ROWID** 行 ID **XML** XML **DISTINCT** 特殊タイプ

表 *161. SYSCOLUMNS2* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
LENGTH	LENGTH	INTEGER	列の長さ属性。ただし、10 進数、数値、または非ゼロ精度 2 進数の列の場合は、その精度: **8** バイト BIGINT **4** バイト INTEGER **2** バイト SMALLINT 数の精度 DECIMAL 数の精度 NUMERIC **8** バイト FLOAT、FLOAT(n) (ここで n = 25 から 53)、または DOUBLE PRECISION **4** バイト FLOAT(n) (ここで n = 1 から 24)、または REAL **8** バイト DECFLOAT(16) **16** バイト DECFLOAT(34) ストリングの長さ CHAR ストリングの最大長 VARCHAR または CLOB グラフィック・ストリングの長さ GRAPHIC グラフィック・ストリングの最大長 VARGRAPHIC または DBCLOB ストリングの長さ BINARY **2** 進ストリングの最大長 VARBIN または BLOB **4** バイト DATE **3** バイト TIME **((*p*+1)/2) + 7 (*p* はタイム・スタンプの精度)** の整数部分 TIMESTAMP データ・リンク **URL** およびコメントの最大長 DATALINK **40** バイト ROWID **2147483647** バイト XML ソース・タイプと同じ値 DISTINCT
NUMERIC_SCALE	SCALE	INTEGER	数値データの位取り。
		NULL 可能	列が 10 進数、数値、または 2 進数でない場合は、NULL 値が入ります。
IS_NULLABLE	NULLS	CHAR(1)	列に NULL 値を入れることが可能かどうか: **N** いいえ **Y** はい

表 *161. SYSCOLUMNS2* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
IS_UPDATABLE	UPDATES	CHAR(1)	列が更新可能かどうか: **N** いいえ **Y** はい
LONG_COMMENT	REMARKS	VARGRAPHIC(2000) CCSID 1200	COMMENT ステートメントで指定された文字ストリング。
		NULL 可能	詳細コメントがない場合は、NULL 値が入ります。
HAS_DEFAULT	DEFAULT	CHAR(1)	列がデフォルト値を持つ (DEFAULT 文節または NULL 使用可能) かどうか: **N** いいえ **Y** はい **A** 列は、ROWID データ・タイプおよび GENERATED ALWAYS 属性を持ちます。 **D** 列は、ROWID データ・タイプおよび GENERATED BY DEFAULT 属性を持ちます。 **E** 列は、FOR EACH ROW ON UPDATE 属性および GENERATED ALWAYS 属性により定義されます。 **F** 列は、FOR EACH ROW ON UPDATE 属性および GENERATED BY DEFAULT 属性により定義されます。 **I** 列は、AS IDENTITY 属性および GENERATED ALWAYS 属性により定義されます。 **J** 列は、AS IDENTITY 属性および GENERATED 属性により定義されます。 **Q** 列は GENERATED AS ROW BEGIN 属性を使って定義される。 **R** 列は GENERATED AS ROW END 属性を使って定義される。 **X** 列は GENERATED AS TRANSACTION START ID 属性を使って定義される。 **a** 列は、特殊レジスターを使用して生成された式として定義されます。 **c** 列は、グローバル変数を使用して生成された式として定義されます。 **d** 列は DATA CHANGE OPERATION を使用して生成された式として定義されます。 列がビューに関するものである場合、N が戻されます。
COLUMN_HEADING	LABEL	VARGRAPHIC(60) CCSID 1200	LABEL ステートメント (列見出し) で指定された文字ストリング。
		NULL 可能	列見出しがない場合は、NULL 値が入ります。

表 *161. SYSCOLUMNS2* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
STORAGE	STORAGE	INTEGER	列の記憶域所要量: **8** バイト　BIGINT **4** バイト　INTEGER **2** バイト　SMALLINT **(**精度**/2) + 1** DECIMAL 数の精度　NUMERIC **8** バイト　FLOAT、FLOAT(n) (ここで n = 25 から 53)、または DOUBLE PRECISION **4** バイト　FLOAT(n) (ここで n = 1 から 24)、または REAL **8** バイト　DECFLOAT(16) **16** バイト DECFLOAT(34) ストリングの長さ CHAR または BINARY ストリングの最大長 **+ 2** VARCHAR または VARBIN ストリングの最大長 **+ 29** CLOB または BLOB ストリングの長さ *** 2** GRAPHIC ストリングの最大長 *** 2 + 2** VARGRAPHIC ストリングの最大長 *** 2 + 29** DBCLOB **4** バイト　DATE **3** バイト　TIME **((***p***+1)/2) + 7 (***p* はタイム・スタンプの精度**)** の整数部分　TIMESTAMP データ・リンク **URL** およびコメントの最大長 **+ 24**　DATALINK **42** バイト ROWID **2147483647** バイト **+ 29** XML ソース・タイプと同じ値 DISTINCT **注:** この列には、すべてのデータ・タイプの記憶域所要量があります。

表 *161. SYSCOLUMNS2* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
NUMERIC_PRECISION	PRECISION	INTEGER NULL 可能	数値の列すべての精度。 **注:** この列では、すべての数値データ・タイプ (10 進浮動小数点数、単精度および倍精度の浮動小数点数を含む) の精度を指定します。 NUMERIC_PRECISION_RADIX 列は、この列の値が 2 進数であるか、または 10 進数であるかを示します。 列が数値の列でない場合は、NULL 値が入ります。
CCSID	CCSID	INTEGER NULL 可能	CHAR、VARCHAR、CLOB、DATE、TIME、TIMESTAMP、GRAPHIC、VARGRAPHIC、DBCLOB、XML、および DATALINK 列の CCSID の値。 列が BINARY、VARBIN、BLOB または ROWID の場合は、65535 が入ります。 列が数値データ・タイプの場合は、NULL 値が入ります。
TABLE_SCHEMA	DBNAME	VARCHAR(128)	該当の表またはビューが入っている SQL スキーマの名前。

表 *161. SYSCOLUMNS2* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
COLUMN_DEFAULT	DFTVALUE	VARGRAPHIC(2000) CCSID 1200 NULL 可能	列のデフォルト値が存在する場合は、そのデフォルト値。列のデフォルト値が、切り捨てなければ表示できない場合は、その列の値はストリング 'TRUNCATED' になります。デフォルト値は文字形式で保管されます。以下の特殊値も存在します。 **CURRENT_DATE** デフォルト値は、現在の日付です。 **CURRENT_TIME** デフォルト値は、現在の時刻です。 **CURRENT_TIMESTAMP** デフォルト値は、現在のタイム・スタンプです。 **NULL** デフォルト値は NULL 値になり、DEFAULT NULL が明示的に指定されています。 **USER** デフォルト値は、現在のジョブ・ユーザーです。 *special-register* 列 HAS_DEFAULT に値 'a' が含まれている場合、特殊レジスターの名前。 *global-variable* 列 HAS_DEFAULT に値 'c' が含まれている場合、グローバル変数の修飾名。 **DATA CHANGE OPERATION** 列 HAS_DEFAULT に値 'd' が含まれている場合。 以下の場合、NULL 値が入ります。 • 列にデフォルト値がない場合 (例えば、列に IDENTITY 属性が指定されている場合、列が行 ID である場合、または列が行変更タイム・スタンプ列、行開始列、行終了列、またはトランザクション開始 ID 列である場合)、または • DEFAULT 値が明示的に指定されていない場合。
CHARACTER_MAXIMUM_LENGTH	CHARLEN	INTEGER NULL 可能	データ・タイプが 2 進ストリング、文字ストリング、グラフィック・ストリング、および XML の場合は、ストリングの最大長。 列がストリングでない場合は、NULL 値が入ります。
CHARACTER_OCTET_LENGTH	CHARBYTE	INTEGER NULL 可能	データ・タイプが 2 進ストリング、文字ストリング、グラフィック・ストリング、および XML の場合は、バイト数。 列がストリングでない場合は、NULL 値が入ります。

表 *161. SYSCOLUMNS2* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
NUMERIC_PRECISION_RADIX	RADIX	INTEGER NULL 可能	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 列が数値の列でない場合は、NULL 値が入ります。
DATETIME_PRECISION	DATPRC	INTEGER NULL 可能	日付、時刻、またはタイム・スタンプの小数部分。 **0** データ・タイプが DATE および TIME の場合 **0-12** データ・タイプが TIMESTAMP の場合 (小数秒) 列が日付、時刻、またはタイム・スタンプの列でない場合は、NULL 値が入ります。
COLUMN_TEXT	LABELTEXT	VARGRAPHIC(50) CCSID 1200 NULL 可能	LABEL ステートメント (列テキスト) で指定された文字ストリング。 列テキストがない場合は、NULL 値が入ります。
SYSTEM_COLUMN_NAME	SYS_CNAME	CHAR(10)	列のシステム名
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	表またはビューのシステム名
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	スキーマのシステム名
USER_DEFINED_TYPE_SCHEMA	TYPESCHEMA	VARCHAR(128) NULL 可能	特殊タイプの場合は、スキーマの名前。 列が特殊タイプでない場合は、NULL 値が入ります。
USER_DEFINED_TYPE_NAME	TYPENAME	VARCHAR(128) NULL 可能	特殊タイプの名前。 列が特殊タイプでない場合は、NULL 値が入ります。
IS_IDENTITY	IDENTITY	VARCHAR(3)	この列は、列が識別列かどうかを指定します。 **NO** 列は識別列ではありません。 **YES** 列は識別列です。
IDENTITY_GENERATION	GENERATED	VARCHAR(10) NULL 可能	この列は、列が GENERATED ALWAYS か GENERATED BY DEFAULT かを識別します。 **ALWAYS** 列の値は常に生成されます。 **BY DEFAULT** 列の値はデフォルトにより生成されます。 この列が、ROWID 列、ID 列、行変更タイム・スタンプ列、行開始列、行終了列、トランザクション開始 ID 列、および生成式列でない場合、NULL 値が入ります。
IDENTITY_START	START	DECIMAL(31,0) NULL 可能	識別列の開始値。 列が IDENTITY 列でない場合は、NULL 値が入ります。
IDENTITY_INCREMENT	INCREMENT	DECIMAL(31,0) NULL 可能	識別列の増分値。 列が IDENTITY 列でない場合は、NULL 値が入ります。

表 *161. SYSCOLUMNS2* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
IDENTITY_MINIMUM	MINVALUE	DECIMAL(31,0)	識別列の最小値。
		NULL 可能	列が IDENTITY 列でない場合は、NULL 値が入ります。
IDENTITY_MAXIMUM	MAXVALUE	DECIMAL(31,0)	識別列の最大値。
		NULL 可能	列が IDENTITY 列でない場合は、NULL 値が入ります。
IDENTITY_CYCLE	CYCLE	VARCHAR(3)	この列は、識別列の値が最小値または最大値に達した後も、値の生成を続けるかどうかを識別します。
		NULL 可能	**NO** 値の生成は継続されません。 **YES** 値の生成は継続されます。 列が IDENTITY 列でない場合は、NULL 値が入ります。
IDENTITY_CACHE	CACHE	INTEGER	アクセスを高速化するために事前割り振りが可能な識別値の数を指定します。ゼロは、値が事前割り振りされないことを示します。
		NULL 可能	列が IDENTITY 列でない場合は、NULL 値が入ります。
IDENTITY_ORDER	ORDER	VARCHAR(3)	識別値を要求された順序で生成しなければならないかどうかを指定します。
		NULL 可能	**NO** 値は、要求された順序で生成する必要はありません。 **YES** 値は、要求された順序で生成する必要があります。 列が IDENTITY 列でない場合は、NULL 値が入ります。
COLUMN_EXPRESSION	EXPRESSION	DBCLOB(2097152)	列が式の場合は、その式が入ります。
		CCSID 1200 NULL 可能	列が式でない場合は、NULL 値が入ります。
HIDDEN	HIDDEN	CHAR(1)	列が暗黙の列リストに含まれるかどうかを指定します。
			P 部分的に隠されている。 **N** 隠されていない。
HAS_FLDPROC	FLDPROC	CHAR(1)	列にフィールド・プロシージャーがあるかどうかを示します。
			N 列にフィールド・プロシージャーがありません。 **Y** 列にフィールド・プロシージャーがあります。
INLINE_LENGTH	ALLOCATE	INTEGER	可変長の列に割り振られた長さ (ALLOCATE) を指定します。
		NULL 可能	列が可変長でない場合は、NULL 値が入ります。
NORMALIZE	NORMALIZE	CHAR(1)	アプリケーションから渡されるときに列データの正規化が必要かどうかを指定します。
		NULL 可能	**0** 列は正規化されません。 **1** 列は正規化されます。 列に Unicode データが含まれていない場合は、NULL 値が入ります。

表 *161. SYSCOLUMNS2* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
DATALINK_LINK_CONTROL	DL_LINKC	CHAR(1) NULL 可能	DATALINK 列のリンクされたファイルが存在するかどうかを判別するための検査が実行されるかどうかを指定します。 **0** 検査は実行されません。 **1** 検査が実行されます。 列のデータ・タイプが DATALINK でない場合は、NULL 値が入ります。
DATALINK_INTEGRITY	DL_INTEG	CHAR(1) NULL 可能	DATALINK 値とリンクされたファイルとの間のリンクの整合性レベルを指定します。 **0** ALL 列のデータ・タイプが DATALINK でないか、データ・リンクに NO LINK CONTROL が指定されている場合は、NULL 値が入ります。
DATALINK_READ_PERMISSION	DL_READP	CHAR(1) NULL 可能	DATALINK 値に指定されたファイルの読み取り許可を決定する方法を指定します。 **0** FS **1** DB 列のデータ・タイプが DATALINK でないか、データ・リンクに NO LINK CONTROL が指定されている場合は、NULL 値が入ります。
DATALINK_WRITE_PERMISSION	DL_WRITEP	CHAR(1) NULL 可能	DATALINK 値に指定されたファイルの書き込み許可を決定する方法を指定します。 **0** FS **1** BLOCKED 列のデータ・タイプが DATALINK でないか、データ・リンクに NO LINK CONTROL が指定されている場合は、NULL 値が入ります。
DATALINK_RECOVERY	DL_RECOVER	CHAR(1) NULL 可能	DATALINK 列のリンクされたファイルのポイント・イン・タイム・リカバリーがサポートされるかどうかを指定します。 **0** NO 列のデータ・タイプが DATALINK でないか、データ・リンクに NO LINK CONTROL が指定されている場合は、NULL 値が入ります。
DATALINK_UNLINK_CONTROL	DL_UNLINKC	CHAR(1) NULL 可能	ファイルがリンク解除されるときに DataLink File Manager が実行する処置を指定します。 **0** RESTORE **1** DELETE 列のデータ・タイプが DATALINK でないか、データ・リンクに NO LINK CONTROL が指定されている場合は、NULL 値が入ります。
DDS_TYPE	DDS_TYPE	CHAR(1) NULL 可能	列のデータ記述仕様 (DDS) データ・タイプを指定します。

表 *161. SYSCOLUMNS2* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
SECURE	SECURE	CHAR(1) NULL 可能	データベース・モニターまたはプラン・キャッシュ内で保護する必要があるデータを列が含んでいるかどうかを指定します。 **0** 列には、データベース・モニターまたはプラン・キャッシュ内で保護する必要があるデータは含まれていません。 **1** 列には、データベース・モニターまたはプラン・キャッシュ内で保護する必要があるデータが含まれています。

SYSCOLUMNSTAT

SYSCOLUMNSTAT ビューには、列統計コレクションの任意の表パーティションまたは表メンバー内の列ごとに 1 つの行が含まれます。表が分散表の場合は、他のデータベース・ノードにあるパーティションは、このカタログ・ビューには含まれません。

これらは該当の他のデータベース・ノードのカタログ・ビューに含まれます。以下の表では、SYSCOLUMNSTAT ビューの列について説明します。

表 *162. SYSCOLUMNSTAT* ビュー

列名	システム列名	データ・タイプ	説明
TABLE_SCHEMA	TABSCHEMA	VARCHAR(128)	該当の表が入っている SQL のスキーマの名前。
TABLE_NAME	TABNAME	VARCHAR(128)	表の名前。
TABLE_PARTITION	TABPART	VARCHAR(128)	表パーティションまたはメンバーの名前。
PARTITION_TYPE	PARTTYPE	CHAR(1)	表パーティショニングのタイプ ブランク 表はパーティション化されません。 **H** これはデータ・ハッシュ・パーティションです。 **R** これはデータ範囲パーティションです。 **D** これは分散データベース・ハッシュ・パーティションです。
PARTITION_NUMBER	PARTNBR	INTEGER NULL 可能	このパーティションのパーティション番号。表が分散表の場合には NULL が入ります。
NUMBER_DISTRIBUTED_PARTITIONS	DSTPARTS	INTEGER NULL 可能	表が分散表の場合にはパーティションの合計数が入ります。表が分散表でない場合には NULL が入ります。
NUMBER_COLUMN_NAMES	NBRCOLS	INTEGER	このコレクション内の列名の数。個別の列統計のみが必要な場合は、NUMBER_COLUMN_NAMES が 1 の行のみを選択してください。 現在は、1 つの名前のみが返されます。
COLUMN_NAME	COLNAME	VARCHAR(1280)	列の名前。最大 10 列が返されます。 現在は、1 つの名前のみが返されます。
NUMBER_DISTINCT_VALUES	COLCARD	BIGINT	列内の特殊値の数。統計が収集されない場合は -1 が入ります。
HIGH2KEY	HIGH2KEY	VARCHAR(254)	Db2 for i には適用されません。空ストリングが入ります。
LOW2KEY	LOW2KEY	VARCHAR(254)	Db2 for i には適用されません。空ストリングが入ります。
AVERAGE_COLUMN_LENGTH	AVGCOLLEN	INTEGER	Db2 for i には適用されません。常に -1 になります。
NUMBER_NULLS	NUMNULLS	BIGINT	NULL 値の見積数。統計が収集されない場合は -1。
SUB_COUNT	SUB_COUNT	SMALLINT	Db2 for i には適用されません。常に -1 になります。
SUB_DELIM_LENGTH	SUBDLENGTH	SMALLINT	Db2 for i には適用されません。常に -1 になります。
NUMBER_HISTOGRAM_RANGES	NQUANTILES	INTEGER	この統計コレクションに使用可能なヒストグラム範囲の数。実際のヒストグラム範囲値は、統計コレクション詳細リスト (QDBSTLDS、QdbstListDetailStatistics) API を使用して入手できます。統計が収集されない場合は -1 が入ります。

表 *162. SYSCOLUMNSTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
NUMBER_MOST_FREQUENT_VALUES	NMOSTFREQ	INTEGER	使用可能な最高頻度値の数。実際の最高頻度値は、統計コレクション詳細リスト (QDBSTLDS、QdbstListDetailStatistics) API を使用して入手できます。統計が収集されない場合は -1 が入ります。
AVGDISTINCTPERPAGE	AVGDSTPAGE	DOUBLE NULL 可能	Db2 for i には適用されません。常に NULL になります。
PAGEVARIANCERATIO	PVARRATIO	DOUBLE NULL 可能	Db2 for i には適用されません。常に NULL になります。
STATISTICS_NAME	STATNAME	VARCHAR(128) NULL 可能	この表パーティションのこの統計コレクションの固有の名前。統計が収集されない場合は NULL が入ります。
INTERNAL_STATISTICS_ID	STATID	VARCHAR(16) FOR BIT DATA NULL 可能	この表パーティションのこの統計コレクションの内部統計 ID。統計が収集されない場合は NULL が入ります。
STATISTIC_CREATED	STATCREATE	TIMESTAMP NULL 可能	統計コレクションが作成されたときのタイム・スタンプ。統計が収集されない場合は NULL が入ります。
STATISTIC_CREATOR	STATCUSER	VARCHAR(128) NULL 可能	統計コレクションを作成したユーザー。システム作成の統計コレクションの場合は *SYS が入ります。統計が収集されない場合は NULL が入ります。
STATISTIC_LAST_UPDATED	UPDATEDTS	TIMESTAMP NULL 可能	統計コレクションの最終更新時のタイム・スタンプ。統計が収集されない場合は NULL が入ります。
STATISTIC_UPDATER	STATUUSER	VARCHAR(128) NULL 可能	統計コレクションの最終更新をしたユーザー。統計コレクションがシステムにより自動更新された場合は、*SYS が入ります。統計が収集されない場合は NULL が入ります。
STATISTICS_SIZE	STATSIZE	BIGINT NULL 可能	この表パーティションの統計コレクションのサイズ。統計が収集されない場合は NULL が入ります。
AGING_MODE	AGING_MODE	VARCHAR(10)	この表パーティションの統計コレクションのエージングまたは除去をシステムが自動的に行うことができるかどうかを指定します。 ***SYS** 統計コレクションは、必要な時点でシステムが自動的にリフレッシュまたは除去します。 ***USER** ユーザーが明示的に要求した場合のみ、統計コレクションがリフレッシュまたは除去されます。
AGING_STATUS	AGING_STS	CHAR(1) NULL 可能	この表パーティションの統計コレクションがどの程度最新のものであるか示します。 **0** 統計データのリフレッシュが必要なことを示す標識はありません。 **1** 統計データのリフレッシュが必要なことを示す標識があります。 統計が収集されない場合は NULL が入ります。
BLOCK_OPTION	BLKOPTION	CHAR(1)	この表パーティションについて自動統計コレクション作成要求が許されるかどうかを示します。 **0** システム開始の自動統計コレクションはブロックされません。 **1** システム開始の自動統計コレクションはブロックされます。

表 *162. SYSCOLUMNSTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
CURRENT_LAST_CHANGE	UPDATED	TIMESTAMP NULL 可能	表パーティション内のデータの最終変更日時を示すタイム・スタンプ。統計が収集されない場合は NULL が入ります。
CURRENT_ROWS	CURROWS	BIGINT NULL 可能	表パーティションの現在の有効行数。統計が収集されない場合は NULL が入ります。
CURRENT_DELETED_ROWS	CURDELROWS	BIGINT NULL 可能	表パーティションの現在の削除済み行数。 統計が収集されない場合は NULL が入ります。
CURRENT_DATA_CHANGES	CURDATACHG	BIGINT NULL 可能	この表パーティションに対して実行された挿入、更新、および削除の数。統計が収集されない場合は NULL が入ります。
STATISTICS_ROWS	STATROWS	BIGINT NULL 可能	統計が収集された時点での、表パーティション内の有効行数。統計が収集されない場合は NULL が入ります。
STATISTICS_DELETED_ROWS	STATDELROW	BIGINT NULL 可能	統計が収集された時点での、表パーティション内の削除済み行数。統計が収集されない場合は NULL が入ります。
STATISTICS_DATA_CHANGES	STATDATCHG	BIGINT NULL 可能	統計が収集された時点での、表パーティションに対して実行された挿入、更新、および削除の数。統計が収集されない場合は NULL が入ります。
TRANSLATION_ATTRIBUTES	TRANSATRS	VARCHAR(10) NULL 可能	統計が収集された時点での、データ値に対して使用された変換のタイプを示します。 **0** 固有の重み変換。 **1** 共用重み変換。 **9** 変換なし。 このコレクションで複数の列が使用されている場合は、複数の変換が行われる場合もあります。 現在は、1 つの変換のみが返されます。
TRANSLATION_TABLES	TRANSTBLS	VARCHAR(210) NULL 可能	統計コレクションで変換テーブルが使用された場合に、変換テーブルの修飾名。 変換テーブルが使用されなかった場合は、空ストリングが返されます。統計が収集されない場合は NULL が入ります。 このコレクションで複数の列が使用されている場合は、複数の変換テーブルが使用されることがあります。 現在は、1 つの変換テーブルのみが返されます。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	システムのスキーマ名。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	システム表名。
SYSTEM_TABLE_MEMBER	SYS_MNAME	CHAR(10)	システム・メンバー名。
SYSTEM_COLUMN_NAME	SYS_CNAME	VARCHAR(100)	システム列名。最大 10 個の名前で構成される配列の場合があります。 現在は、1 つの名前のみが返されます。

SYSCONTROLS

SYSCONTROLS ビューには、 CREATE PERMISSION または CREATE MASK ステートメントで定義された行の許可または列マスクごとに 1 つの行が含まれます。

次の表は、SYSCONTROLS ビューの列について説明しています。

表 *163. SYSCONTROLS* ビュー

列名	システム列名	データ・タイプ	説明
RCAC_SCHEMA	SCHEMA	VARCHAR(128)	行権限または列マスクのスキーマ名。
RCAC_NAME	NAME	VARCHAR(128)	行の許可または列マスクの名前。
RCAC_OWNER	CREATOR	VARCHAR(128)	行の許可または列マスクの作成者。
TABLE_SCHEMA	TBSCHEMA	VARCHAR(128)	行の許可または列マスクが定義されている表のスキーマ名。
TABLE_NAME	TBNAME	VARCHAR(128)	行の許可または列マスクが定義されている表の名前。
TBCORRELATION	TBCORRNAME	VARCHAR(128)	行の許可または列マスクが定義されている表の相関名。指定されていない場合、値が生成されます。
COLUMN_NAME	COLNAME	VARCHAR(128)	列マスクが定義されている列の名前。行の許可の場合はブランクが入ります。
SYSTEM_COLUMN_NAME	SYS_CNAME	CHAR(10)	列のシステム名。行の許可の場合はブランクが入ります
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	表のシステム名。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	スキーマのシステム名。
CONTROL_TYPE	CONTROLTYP	CHAR(1)	アクセス制御のタイプを示します。 **R** 行の許可 **M** 列マスク
ENFORCED	ENFORCED	CHAR(1)	行の許可によって施行されるアクセスのタイプを示します。列マスクは常に「A」です。 **A** すべてのアクセス
IMPLICIT	IMPLICIT	CHAR(1)	行の許可が暗黙的に作成されたかどうかを示します。 **N** 行の許可は暗黙的に作成されなかったか、または、これは列マスクである。 **Y** 行の許可は暗黙的に作成された。
ENABLE	ENABLE	CHAR(1)	アクセス制御のために行の許可または列マスクが使用可能であるかどうかを示します。 **N** 使用可能ではない **Y** 使用可能
CREATE_TIME	CREATEDTS	TIMESTAMP	行の許可または列マスクが作成された時点のタイム・スタンプを指定します。
LAST_ALTERED	ALTEREDTS	TIMESTAMP NULL 可能	行の許可または列マスクが最後に変更された時点のタイム・スタンプを指定します。 一度も変更されていない場合は、NULL 値が入ります。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。
LABEL	LABEL	VARGRAPHIC(50) CCSID 1200 NULL 可能	LABEL ステートメントで指定された文字ストリング。ラベルが存在しない場合は NULL 値が入ります。

表 *163. SYSCONTROLS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
LONG_COMMENT	REMARKS	VARGRAPHIC(2000) CCSID 1200 NULL 可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、NULL 値が入ります。
RULETEXT	RULETEXT	DBCLOB(2M) CCSID 1200	CREATE PERMISSION ステートメントまたは CREATE MASK ステートメントの検索条件または式の部分のソース・テキスト。

SYSCONTROLSDEP

SYSCONTROLSDEP ビューは、行の許可および列マスクの従属関係を記録します。

次の表は、SYSCONTROLSDEP ビューの列について説明しています。

表 *164. SYSCONTROLSDEP* ビュー

列名	システム列名	データ・タイプ	説明
RCAC_SCHEMA	SCHEMA	VARCHAR(128)	行権限または列マスクのスキーマ名。
RCAC_NAME	NAME	VARCHAR(128)	行の許可または列マスクの名前。
CONTROL_TYPE	CONTROLTYP	CHAR(1)	アクセス制御のタイプを示します。 **R** 行の許可 **M** 列マスク
OBJECT_SCHEMA	BSCHEMA	VARCHAR(128)	該当のオブジェクトが入っているスキーマの名前。
OBJECT_NAME	BNAME	VARCHAR(128)	行の許可または列マスクが従属しているオブジェクトの名前。
OBJECT_TYPE	BTYPE	CHAR(24)	行の許可または列マスク内で参照されているオブジェクトのオブジェクト・タイプを示します。 **COLUMN** オブジェクトは表内の列です。 **FUNCTION** オブジェクトは関数です。 **MATERIALIZED QUERY TABLE** オブジェクトはマテリアライズ照会表です。 **SEQUENCE** オブジェクトはシーケンスです。 **TABLE** オブジェクトは表です。 **TYPE** オブジェクトは特殊タイプです。 **VARIABLE** オブジェクトは変数です。 **VIEW** オブジェクトはビューです。
COLUMN_NAME	BCOLNAME	VARCHAR(128)	OBJECT_TYPE の値が「COLUMN」の場合、オブジェクトの列名。この場合、OBJECT_NAME には列の表名が含まれます。 オブジェクトが列でない場合はブランクが入ります。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	行の許可または列マスクが定義されている表のシステム名。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	行の許可または列マスクが定義されている表のスキーマのシステム名。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。
PARM_SIGNATURE	SIGNATURE	BLOB(3M) NULL 可能	この列はルーチン・シグニチャーを識別します。 オブジェクトがルーチンでない場合は、NULL 値が入ります。

SYSCST

SYSCST ビューには、SQL のスキーマにある各制約ごとに、行が 1 つずつ入ります。

次の表は、SYSCST ビューの列について説明しています。

表 *165. SYSCST* ビュー

列名	システム列名	データ・タイプ	説明
CONSTRAINT_SCHEMA	CDBNAME	VARCHAR(128)	該当の制約が入っているスキーマの名前。
CONSTRAINT_NAME	RELNAME	VARCHAR(128)	制約の名前。
CONSTRAINT_TYPE	TYPE	VARCHAR(11)	制約のタイプ • CHECK • UNIQUE • PRIMARY KEY • FOREIGN KEY
TABLE_SCHEMA	TDBNAME	VARCHAR(128)	該当の表が入っているスキーマの名前。
TABLE_NAME	TBNAME	VARCHAR(128)	該当の制約が作成される表の名前。 SQL の表名が存在する場合は、その SQL の表名になります。存在しない場合は、システムの表名になります。
IS_DEFERRABLE	ISDEFER	VARCHAR(3)	制約の検査が据え置きできるかどうかを示します。常に 'NO' になります。
INITIALLY_DEFERRED	INITDEFER	VARCHAR(3)	制約が初期据え置きとして定義されたかどうかを示します。 常に 'NO' になります。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	表のシステム名。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	該当の表が入っているスキーマのシステム名。
CONSTRAINT_KEYS	COLCOUNT	SMALLINT NULL 可能	これが UNIQUE、PRIMARY KEY、または FOREIGN KEY 制約の場合は、キー列の数を指定します。 制約が CHECK 制約の場合は、NULL 値が入ります。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。
CONSTRAINT_STATE	CST_STATE	VARCHAR(11)	制約が確立または定義されているかどうかを示します。 **ESTABLISHED** 参照制約が確立されています。親表が存在します。 **DEFINED** 参照制約が定義されています。親表が存在しません。
ENABLED	ENABLED	VARCHAR(3) NULL 可能	制約が使用可能かどうかを示します。 **NO** 制約は使用不可です。 **YES** 制約は使用可能です。 制約が定義されているかまたはユニーク制約の場合は、NULL 値が入ります。
CHECK_PENDING	CHECKFLAG	VARCHAR(3) NULL 可能	制約がチェック・ペンディング状態にあるかどうかを示します。 **NO** 制約はチェック・ペンディング中ではありません。 **YES** 制約はチェック・ペンディング中です。 制約が定義されているか使用不可の場合、またはユニーク制約の場合は、NULL 値が入ります。

表 *165. SYSCST* ビュー (続き)

列名	システム列名	データ・タイプ	説明
CONSTRAINT_TEXT	LABEL	VARGRAPHIC(50) CCSID 1200	LABEL ステートメントで指定された文字ストリング。
		NULL 可能	ラベルがない場合は、NULL 値が入ります。
LONG_COMMENT	REMARKS	VARGRAPHIC(2000) CCSID 1200	COMMENT ステートメントで指定された文字ストリング。
		NULL 可能	詳細コメントがない場合は、NULL 値が入ります。
SYSTEM_CONSTRAINT_SCHEMA	SYS_CDNAME	CHAR(10)	制約が入っているシステム・スキーマの名前。

SYSCSTCOL

ビュー SYSCSTCOL は、制約が定義される対象となる列を記録します。固有キー制約、基本キー制約および表検査制約の列ごとに、および参照制約の参照列に対して、行が 1 つずつあります。

次の表は、SYSCSTCOL ビューの列について説明しています。

表 *166. SYSCSTCOL* ビュー

列名	システム列名	データ・タイプ	説明
TABLE_SCHEMA	TDBNAME	VARCHAR(128)	該当の制約が従属している表が入っている SQL スキーマの名前。
TABLE_NAME	TBNAME	VARCHAR(128)	該当の制約が従属している表の名前。 SQL の表名が存在する場合は、その SQL の表名です。存在しない場合は、システムの表名になります。
COLUMN_NAME	COLUMN	VARCHAR(128)	該当の制約が作成された列。 SQL の列名が存在する場合は、その SQL の列名です。存在しない場合は、システムの列名になります。
CONSTRAINT_SCHEMA	CDBNAME	VARCHAR(128)	該当の制約のスキーマの名前。
CONSTRAINT_NAME	RELNAME	VARCHAR(128)	制約の名前。
SYSTEM_COLUMN_NAME	SYS_CNAME	CHAR(10)	列のシステム名。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	表のシステム名。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	該当の表が入っているスキーマのシステム名。
SYSTEM_CONSTRAINT_SCHEMA	SYS_CDNAME	CHAR(10)	制約が入っているシステム・スキーマの名前。

SYSCSTDEP

ビュー SYSCSTDEP は、制約が定義される対象となる表を記録します。

次の表は、SYSCSTDEP ビューの列について説明しています。

表 *167. SYSCSTDEP* ビュー

列名	システム列名	データ・タイプ	説明
TABLE_SCHEMA	TDBNAME	VARCHAR(128)	該当の制約が従属している表が入っている SQL スキーマの名前。
TABLE_NAME	TBNAME	VARCHAR(128)	該当の制約が従属している表の名前。 SQL の表名が存在する場合は、その SQL の表名です。存在しない場合は、システムの表名になります。
CONSTRAINT_SCHEMA	CDBNAME	VARCHAR(128)	該当の制約のスキーマの名前。
CONSTRAINT_NAME	RELNAME	VARCHAR(128)	制約の名前。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	表のシステム名。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	該当の表が入っているスキーマのシステム名。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。
SYSTEM_CONSTRAINT_SCHEMA	SYS_CDNAME	CHAR(10)	制約が入っているシステム・スキーマの名前。

SYSFIELDS

SYSFIELDS ビューには、フィールド・プロシージャーを持つ各列ごとに、行が 1 つずつ入ります。

SYSFIELDS 内の列属性は、フィールド・プロシージャーによって定義された内部列属性を記述します。 次の表は、SYSFIELDS ビューの列について説明しています。

表 *168. SYSFIELDS* ビュー

列名	システム列名	データ・タイプ	説明
TABLE_SCHEMA	DBNAME	VARCHAR(128)	表が入っている SQL スキーマの名前。
TABLE_NAME	TBNAME	VARCHAR(128)	列が入っている表の名前。 SQL の表名が存在する場合は、その SQL の表名になります。存在しない場合は、システムの表名になります。
COLUMN_NAME	NAME	VARCHAR(128)	列の名前。 SQL の列名が存在する場合は、その SQL の列名になります。存在しない場合は、システムの列名になります。
ORDINAL_POSITION	COLNO	INTEGER	表における該当の列の数値位置 (左から右への順序)。

表 *168. SYSFIELDS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
DATA_TYPE	COLTYPE	VARCHAR(8)	列のタイプ: **BIGINT** 大整数 **INTEGER** 長整数 **SMALLINT** 短整数 **DECIMAL** パック 10 進数 **NUMERIC** ゾーン 10 進数 **FLOAT** 浮動小数点数; FLOAT、REAL、または DOUBLE PRECISION **DECFLOAT** 10 進浮動小数点数 **CHAR** 固定長文字ストリング **VARCHAR** 可変長文字ストリング **CLOB** 文字ラージ・オブジェクト・ストリング **GRAPHIC** 固定長グラフィック・ストリング **VARG** 可変長グラフィック・ストリング **DBCLOB** 2 バイト文字ラージ・オブジェクト・ストリング **BINARY** 固定長バイナリー・ストリング **VARBIN** 可変長バイナリー・ストリング **BLOB** バイナリー・ラージ・オブジェクト・ストリング **DATE** 日付 **TIME** 時刻 **TIMESTMP** タイム・スタンプ **DATALINK** データ・リンク **ROWID** 行 ID **XML** XML **DISTINCT** 特殊タイプ

表 *168. SYSFIELDS* ビュー (続き)

列名	システム列名	データ・タイプ	説明
LENGTH	LENGTH	INTEGER	列の長さ属性。ただし、10 進数、数値、または非ゼロ精度 2 進数の列の場合は、その精度: **8** バイト BIGINT **4** バイト INTEGER **2** バイト SMALLINT 数の精度 DECIMAL 数の精度 NUMERIC **8** バイト FLOAT、FLOAT(n) (ここで n = 25 から 53)、または DOUBLE PRECISION **4** バイト FLOAT(n) (ここで n = 1 から 24)、または REAL **8** バイト DECFLOAT(16) **16** バイト DECFLOAT(34) ストリングの長さ CHAR ストリングの最大長 VARCHAR または CLOB グラフィック・ストリングの長さ GRAPHIC グラフィック・ストリングの最大長 VARGRAPHIC または DBCLOB ストリングの長さ BINARY **2** 進ストリングの最大長 VARBIN または BLOB **4** バイト DATE **3** バイト TIME **((*p*+1)/2) + 7 (*p* はタイム・スタンプの精度)** の整数部分 TIMESTAMP データ・リンク **URL** およびコメントの最大長 DATALINK **40** バイト ROWID **2147483647** バイト XML ソース・タイプと同じ値 DISTINCT
CHARACTER_MAXIMUM_LENGTH	CHARLEN	INTEGER NULL 可能	データ・タイプが 2 進ストリング、文字ストリング、グラフィック・ストリング、および XML の場合は、ストリングの最大長。 列がストリングでない場合は、NULL 値が入ります。

表 *168. SYSFIELDS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
CHARACTER_OCTET_LENGTH	CHARBYTE	INTEGER NULL 可能	データ・タイプが 2 進ストリング、文字ストリング、グラフィック・ストリング、および XML の場合は、バイト数。 列がストリングでない場合は、NULL 値が入ります。
NUMERIC_SCALE	SCALE	INTEGER NULL 可能	数値データの位取り。 列が 10 進数、数値、または 2 進数でない場合は、NULL 値が入ります。
NUMERIC_PRECISION	PRECISION	INTEGER NULL 可能	数値の列すべての精度。 **注:** この列では、すべての数値データ・タイプ (10 進浮動小数点数、単精度および倍精度の浮動小数点数を含む) の精度を指定します。NUMERIC_PRECISION_RADIX 列は、この列の値が 2 進数であるか、または 10 進数であるかを示します。 列が数値の列でない場合は、NULL 値が入ります。
NUMERIC_PRECISION_RADIX	RADIX	INTEGER NULL 可能	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 列が数値の列でない場合は、NULL 値が入ります。
CCSID	CCSID	INTEGER NULL 可能	CHAR、VARCHAR、CLOB、DATE、TIME、TIMESTAMP、GRAPHIC、VARGRAPHIC、DBCLOB、XML、および DATALINK 列の CCSID の値。 列が BINARY、VARBIN、BLOB または ROWID の場合は、65535 が入ります。 列が数値データ・タイプの場合は、NULL 値が入ります。
DATETIME_PRECISION	DATPRC	INTEGER NULL 可能	日付、時刻、またはタイム・スタンプの小数部分。 **0** データ・タイプが DATE および TIME の場合 **0-12** データ・タイプが TIMESTAMP の場合 (小数秒) 列が日付、時刻、またはタイム・スタンプの列でない場合は、NULL 値が入ります。
FIELD_PROC	FLDPROC	VARCHAR(279) NULL 可能	プロシージャーの名前。
PARMLIST	PARMLIST	DBCLOB(1M) CCSID 1200 NULL 可能	フィールド・プロシージャーを定義したステートメントの FIELDPROC の後ろにあるパラメーター・リスト (意味のないブランクは削除)。
EXITPARM	EXITPARM	BLOB(1M) NULL 可能	フィールド・プロシージャーのパラメーター値ブロック。フィールド・プロシージャーの呼び出し時にフィールド・プロシージャーに渡す制御ブロックです。

表 *168. SYSFIELDS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
SYSTEM_COLUMN_NAME	SYS_CNAME	CHAR(10)	列のシステム名
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	表のシステム名
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	スキーマのシステム名

SYSFUNCS

SYSFUNCS ビューには、CREATE FUNCTION ステートメントによって作成された各関数ごとに行が 1 つずつ入ります。

次の表は、SYSFUNCS ビューの列について説明しています。

表 *169. SYSFUNCS* ビュー

列名	システム列名	データ・タイプ	説明
SPECIFIC_SCHEMA	SPECSCHEMA	VARCHAR(128)	ルーチン (関数) インスタンスのスキーマ名。
SPECIFIC_NAME	SPECNAME	VARCHAR(128)	ルーチン・インスタンスの特定名。
ROUTINE_SCHEMA	FUNCSCHEMA	VARCHAR(128)	該当のルーチンが入っている SQL スキーマの名前。
ROUTINE_NAME	FUNCNAME	VARCHAR(128)	ルーチンの名前。
ROUTINE_CREATED	RTNCREATE	TIMESTAMP	ルーチンが作成されたときのタイム・スタンプを識別します。
ROUTINE_DEFINER	DEFINER	VARCHAR(128)	該当のルーチンを定義したユーザーの名前。
ROUTINE_BODY	BODY	VARCHAR(8)	ルーチン本体のタイプ: **EXTERNAL** これは外部ルーチンです。 **SQL** これは SQL ルーチンです。
EXTERNAL_NAME	EXTNAME	VARCHAR(279) NULL 可能	この列は外部プログラム名を識別します。 • SQL 関数または ILE サービス・プログラムの場合、外部プログラム名はスキーマ名/サービス・プログラム名 (入り口名) です。 • Java プログラムの場合、外部プログラム名はオプションの jar-id の後に完全修飾クラス名!メソッド名 または完全修飾クラス名.メソッド名 が続きます。 • その他のすべての言語では、外部プログラム名は、スキーマ名/プログラム名 です。 これがシステム生成関数でない場合は、NULL 値が入ります。

表 *169. SYSFUNCS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
EXTERNAL_LANGUAGE	LANGUAGE	VARCHAR(8) NULL 可能	これが外部ルーチンである場合は、この列は外部プログラム名を識別します。 **C** 外部プログラムは C で作成されます。 **C++** 外部プログラムは C++ で作成されます。 **CL** 外部プログラムは CL で作成されます。 **COBOL** 外部プログラムは COBOL で作成されます。 **COBOLLE** 外部プログラムは ILE COBOL で作成されます。 **JAVA** 外部プログラムは JAVA で作成されます。 **PLI** 外部プログラムは PL/I で作成されます。 **RPG** 外部プログラムは RPG で作成されます。 **RPGLE** 外部プログラムは ILE RPG で作成されます。 これが外部ルーチンでない場合は、NULL 値が入ります。
PARAMETER_STYLE	PARM_STYLE	VARCHAR(7) NULL 可能	これが外部ルーチンである場合は、この列はパラメーターのスタイル (呼び出し規則) を識別します。 **DB2SQL** これは DB2SQL 呼び出し規則です。 **DB2GNRL** これは DB2GENERAL 呼び出し規則です。 **GENERAL** これは GENERAL 呼び出し規則です。 **JAVA** これは JAVA 呼び出し規則です。 **NULLS** これは GENERAL WITH NULLS 呼び出し規則です。 **SQL** これは SQL 標準呼び出し規則です。 これが外部ルーチンでない場合は、NULL 値が入ります。
IS_DETERMINISTIC	DETERMINE	VARCHAR(3)	この列はルーチンが deterministic であるかどうかを識別します。 つまり、同じ引数のルーチンに対する呼び出しが、常に同じ結果を戻すかどうかを識別します。 **NO** ルーチンは deterministic ではありません。 **YES** ルーチンはグローバル deterministic です。 **STM** ルーチンはステートメント deterministic です。

表 *169. SYSFUNCS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
SQL_DATA_ACCESS	DATAACCESS	VARCHAR(8) NULL 可能	この列は、ルーチンに SQL が含まれているか、およびルーチンがデータの読み取りまたは変更を行うかを識別します。 **NONE** ルーチンは SQL ステートメントを含みません。 **CONTAINS** ルーチンは SQL ステートメントを含みます。 **READS** ルーチンは、おそらく表またはビューからデータを読み取ります。 **MODIFIES** ルーチンは、おそらく表またはビュー内のデータを変更するか、SQL DDL ステートメントを発行します。
SQL_PATH	SQL_PATH	VARCHAR(3483) NULL 可能	これが SQL ルーチンの場合、この列はパスを識別します。 これが外部ルーチンの場合は、NULL 値が入ります。
PARM_SIGNATURE	SIGNATURE	VARCHAR(16000)	この列はルーチン・シグニチャーを識別します。
NUMBER_OF_RESULTS	NUMRESULTS	SMALLINT NULL 可能	結果の数を識別します。
IN_PARMS	IN_PARMS	SMALLINT	入力パラメーターの数を識別します。 0 は入力パラメーターがないことを示します。
LONG_COMMENT	REMARKS	VARGRAPHIC(2000) CCSID 1200 NULL 可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、NULL 値が入ります。
ROUTINE_DEFINITION	ROUTINEDEF	DBCLOB(2M) CCSID 13488 NULL 可能	これが SQL ルーチンの場合、この列は SQL ルーチン本体を含みます。 これが難読化されたルーチンの場合、テキストは WRAPPED キーワードで始まり、その後に、エンコードされた形式のステートメント・テキストが続きます。 これが SQL ルーチンでない場合は、NULL 値が入ります。
FUNCTION_ORIGIN	ORIGIN	CHAR(1)	関数のタイプを識別します。これがプロシージャーの場合、この列にはブランクが入ります。 **B** これは組み込み関数 (Db2 for iによって定義された) です。 **E** これはユーザー定義関数です。 **U** これは、他の関数に基づいているユーザー定義関数です。 **S** これはシステム生成関数です。
FUNCTION_TYPE	TYPE	CHAR(1)	関数の形式を識別します。これがプロシージャーの場合、この列にはブランクが入ります。 **S** これはスカラー関数です。 **C** これは列関数です。 **T** これは表関数です。

表 *169. SYSFUNCS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
EXTERNAL_ACTION	EXT_ACTION	CHAR(1) NULL 可能	関数の呼び出しに外部的な作用があるかどうかを識別します。 **E** この関数には、外部的な副次作用があります。 **N** この関数には、外部的な副次作用はありません。
IS_NULL_CALL	NULL_CALL	VARCHAR(3) NULL 可能	入力パラメーターが NULL 値である場合に、関数を呼び出す必要があるかどうかを識別します。 **NO** この関数は、入力パラメーターが NULL 値の場合に呼び出す必要はありません。これがスカラー関数の場合は、いずれかのオペランドが NULL であれば、この関数の結果は暗黙的に NULL になります。これが表関数の場合は、いずれかのオペランドが NULL 値であれば、この関数の結果は空の表になります。 **YES** この関数は、入力オペランドが NULL でも呼び出す必要があります。
SCRATCH_PAD	SCRATCHPAD	INTEGER NULL 可能	静的メモリー域 (スクラッチパッド) のアドレスが関数に渡されるかどうかを識別します。 **0** 関数にはスクラッチパッドはありません。 整数 関数に渡されるスクラッチパッドのサイズを示します。
FINAL_CALL	FINAL_CALL	VARCHAR(3) NULL 可能	関数がその作業域 (スクラッチパッド) の終結処理を行えるようにするために、関数への最終呼び出しを行う必要があるかどうかを示します。 **NO** 最終呼び出しは行いません。 **YES** ステートメントが完了したときに関数への最終呼び出しを行います。
PARALLELIZABLE	PARALLEL	VARCHAR(3) NULL 可能	関数が並行して実行できるかどうかを識別します。 **NO** 関数は同期させなければなりません。 **YES** 関数は並行して実行できます。
DBINFO	DBINFO	VARCHAR(3) NULL 可能	データベースに関する情報を関数に渡すかどうかを識別します。 **NO** 関数にデータベース情報を渡しません。 **YES** 関数にデータベースに関する情報を渡します。
SOURCE_ SPECIFIC_SCHEMA	SRCSCHEMA	VARCHAR(128) NULL 可能	これがソース化関数であり、ソースがユーザー定義の場合、この列にはソース・スキーマが入ります。これがソース化関数であり、ソースが組み込みである場合、この列には 'QSYS2' が入ります。 これがソース化関数でない場合は、NULL 値が入ります。
SOURCE_SPECIFIC_NAME	SRCNAME	VARCHAR(128) NULL 可能	これがソース化関数であり、ソースがユーザー定義である場合、この列にはソース関数名の特定名が入ります。 これがソース化関数でない場合は、NULL 値が入ります。

表 *169. SYSFUNCS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
IS_USER_DEFINED_CAST	CAST_FUNC	VARCHAR(3) NULL 可能	この関数が、特殊タイプの作成時に作成されたキャスト関数であるかどうかを識別します。 **NO** この関数はキャスト関数ではありません。 **YES** この関数はキャスト関数です。
CARDINALITY	CARD	BIGINT NULL 可能	表関数の基数を指定します。 この関数が表関数でない場合、または基数が指定されていなかった場合は、NULL 値が入ります。
FENCED	FENCED	VARCHAR(3) NULL 可能	関数を隔離するかどうかを指定します。 **NO** 関数を隔離しません。 **YES** 関数を隔離します。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。
ROUNDING_MODE	DECFLTRND	CHAR(1) NULL 可能	これが SQL 関数の場合は、DECFLOAT 丸めモードを識別します。 **C** ROUND_CEILING **D** ROUND_DOWN **F** ROUND_FLOOR **G** ROUND_HALF_DOWN **E** ROUND_HALF_EVEN **H** ROUND_HALF_UP **U** ROUND_UP 関数が SQL 関数でない場合は、NULL 値が入ります。
INLINE	INLINE	VARCHAR(3) NULL 可能	関数をインラインで記述できるかどうかを示します。 **NO** 関数をインラインで記述できません。 **YES** 関数をインラインで記述できます。 関数が SQL 関数でない場合は、NULL 値が入ります。
ROUTINE_TEXT	LABEL	VARGRAPHIC(50) CCSID 1200 NULL 可能	ルーチンのラベルが入ります。ラベルが存在しない場合は NULL 値が入ります。
SECURE	SECURE	CHAR(1)	関数が行アクセス制御と列アクセス制御においてセキュアであると見なされるかどうかを示します。 **N** 関数が行アクセス制御と列アクセス制御においてセキュアであると見なされません。 **Y** 関数が行アクセス制御と列アクセス制御においてセキュアであると見なされます。

表 *169. SYSFUNCS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
PIPELINED	PIPELINED	VARCHAR(3) NULL 可能	表関数がパイプライン化されているかどうかを示します。 **NO** 表関数は、パイプライン表関数ではありません。 **YES** 表関数は、パイプライン関数です。 ルーチンが表関数でない場合は、NULL 値が入ります。

SYSHISTORYTABLES

SYSHISTORYTABLES ビューには、確立されたバージョン管理関係に履歴表が含まれているかどうかに関係なく、すべての各履歴表ごとに、行が 1 つずつ入ります。

次の表は、SYSHISTORYTABLES ビューの列について説明しています。

表 *170. SYSHISTORYTABLES* ビュー

列名	システム列名	データ・タイプ	説明
HISTORY_TABLE_SCHEMA	HSTDBNAME	VARCHAR(128)	履歴表のスキーマ名。
HISTORY_TABLE_NAME	HSTTBNAME	VARCHAR(128)	履歴表の名前。
VERSIONING_STATUS	VERSIONSTS	CHAR(1)	バージョン管理の状況 **E** システム期間テンポラル表と履歴表とのバージョン管理関係が確立されています。システム期間テンポラル表は、変更された行の以前のバージョンをこの履歴表に保管しています。 **D** この履歴表とシステム期間テンポラル表のバージョン管理関係が定義されていますが、確立はされていません。
PERIOD_NAME	PERIODNAME	VARCHAR(128)	期間の名前。
TABLE_SCHEMA	DBNAME	VARCHAR(128) NULL 可能	該当のテンポラル表が入っている SQL のスキーマの名前。 システム期間テンポラル表が存在しない場合は NULL 値が入ります。
TABLE_NAME	TBNAME	VARCHAR(128) NULL 可能	テンポラル表の名前。 システム期間テンポラル表が存在しない場合は NULL 値が入ります。
SYSTEM_HISTORY_SCHEMA	SYSHSTLIB	CHAR(10)	履歴表のシステム・スキーマ名。
SYSTEM_HISTORY_TABLE_NAME	SYSHSTNAME	CHAR(10)	履歴表のシステム名。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10) NULL 可能	テンポラル表のシステム・スキーマ名。 システム期間テンポラル表が存在しない場合は NULL 値が入ります。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10) NULL 可能	テンポラル表のシステム名。 システム期間テンポラル表が存在しない場合は NULL 値が入ります。

SYSINDEXES

SYSINDEXES ビューには、SQL CREATE INDEX ステートメントを使用して作成された SQL スキーマにある各索引 (SQL カタログに関する索引を含む) ごとに、行が 1 つずつ入ります。

次の表は、SYSINDEXES ビューの列について説明しています。

表 *171. SYSINDEXES* ビュー

列名	システム列名	データ・タイプ	説明
INDEX_NAME	NAME	VARCHAR(128)	索引の名前。 SQL の索引名が存在する場合は、その SQL の索引名になります。存在しない場合は、システムの索引名になります。
INDEX_OWNER	CREATOR	VARCHAR(128)	索引の所有者
TABLE_NAME	TBNAME	VARCHAR(128)	該当の索引が定義される表の名前。 SQL の表名が存在する場合は、その SQL の表名になります。存在しない場合は、システムの表名になります。
TABLE_OWNER	TBCREATOR	VARCHAR(128)	表の所有者
TABLE_SCHEMA	TBDBNAME	VARCHAR(128)	該当の索引が定義される表が入っている SQL スキーマの名前
IS_UNIQUE	UNIQUERULE	CHAR(1)	索引が固有索引の場合: **D** NO (重複が許されます) **V** YES (重複 NULL 値が許されます) **U** YES **E** コード化ベクトル索引
COLUMN_COUNT	COLCOUNT	INTEGER	キーの中の列の数
INDEX_SCHEMA	DBNAME	VARCHAR(128)	該当の索引が入っている SQL スキーマの名前
SYSTEM_INDEX_NAME	SYS_IXNAME	CHAR(10)	システムの索引名
SYSTEM_INDEX_SCHEMA	SYS_IDNAME	CHAR(10)	システムの索引スキーマ名
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	システム表名
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	システムの表スキーマ名
LONG_COMMENT	REMARKS	VARGRAPHIC(2000) CCSID 1200 NULL 可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、NULL 値が入ります。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。
INDEX_TEXT	LABEL	VARGRAPHIC(50) CCSID 1200	LABEL ステートメントで指定された文字ストリング。
IS_SPANNING_INDEX	SPANNING	VARCHAR(3) NULL 可能	索引が複数のパーティションまたはメンバーで作成されているかどうかを示す標識。 **DISTRIBUTED** 索引は分散表で作成されています。 **NO** 索引は複数のパーティションまたはメンバーで作成されていません。 **YES** 索引は複数のパーティションまたはメンバーで作成されています。 基本表がパーティション化された表でない場合は、NULL 値が入ります。
INDEX_DEFINER	DEFINER	VARCHAR(128) NULL 可能	該当の索引を定義したユーザーの名前。

表 171. SYSINDEXES ビュー (続き)

列名	システム列名	データ・タイプ	説明
ROUNDING_MODE	DECFLTRND	CHAR(1) NULL 可能	索引の DECFLOAT 丸めモードを示します。 **C** ROUND_CEILING **D** ROUND_DOWN **F** ROUND_FLOOR **G** ROUND_HALF_DOWN **E** ROUND_HALF_EVEN **H** ROUND_HALF_UP **U** ROUND_UP DECFLOAT 列、関数、または定数を参照する式が索引にない場合は、NULL 値が入ります。
INDEX_HAS_SEARCH_CONDITION	IXHASWHERE	CHAR(1)	索引に検索条件があるかどうかを示します。 **N** 索引には検索条件はありません。 **Y** 索引には検索条件があります。
SEARCH_CONDITION_HAS_UDF	IXWHEREUDF	CHAR(1)	索引検索条件にユーザー定義関数が含まれているかどうかを示します。 **N** 索引は疎索引でないか、またはユーザー定義関数を含んでいません。 **Y** 索引は疎索引で、検索条件に UDF が含まれています。
SEARCH_CONDITION	IXWHERECON	DBCLOB(2M) CCSID 1200 NULL 可能	疎索引の場合は、検索条件が入ります。 疎索引でない場合は、NULL 値が入ります。
INDEX_HAS_INCLUDE_EXPRESSION	IXHASINCEX	CHAR(1)	索引に INCLUDE 文節が含まれているかどうかを示す標識。 **N** 索引には INCLUDE 文節が含まれていません。 **Y** 索引には INCLUDE 文節が含まれています。
INCLUDE_EXPRESSION	IXINCEXPR	DBCLOB(2M) CCSID 1200 NULL 可能	索引に INCLUDE 文節が含まれている場合は、INCLUDE 式のリストが入ります。 INCLUDE 文節がない場合は、NULL 値が入ります。
CREATED	CREATED	TIMESTAMP	SQL 索引が作成されたときのタイム・スタンプ。

SYSINDEXSTAT

SYSINDEXSTAT ビューには、各 SQL 索引パーティションごとに行が 1 つずつ入ります。

このビューを使用すると、特定の SQL 索引または一連の SQL 索引についての情報を表示することができます。この情報は、System i ナビゲーターで「索引の表示」を選択したときに返される情報と同様のものです。

以下の表では、SYSINDEXSTAT ビューの列について説明します。

表 *172. SYSINDEXSTAT* ビュー

列名	システム列名	データ・タイプ	説明
INDEX_SCHEMA	INDSCHEMA	VARCHAR(128)	該当の SQL 索引が入っている SQL スキーマの名前。
INDEX_NAME	INDNAME	VARCHAR(128)	SQL 索引の名前。
INDEX_PARTITION	INDPART	VARCHAR(128)	SQL 索引のパーティション名またはメンバー名。
INDEX_OWNER	INDOWNER	VARCHAR(128)	SQL 索引の所有者。
INDEX_TEXT	LABEL	VARGRAPHIC(50) CCSID 1200 NULL 可能	SQL 索引のテキスト。該当の SQL 索引にテキストが存在しない場合は、NULL が入ります。
TABLE_SCHEMA	TABSCHEMA	VARCHAR(128)	該当の表が入っている SQL のスキーマの名前。
TABLE_NAME	TABNAME	VARCHAR(128)	表の名前。
TABLE_PARTITION	TABPART	VARCHAR(128)	表パーティションまたはメンバーの名前。
INDEX_VALID	VALID	VARCHAR(3)	SQL 索引が無効で再作成が必要であるかどうかを示す指示。 **NO** SQL 索引は無効です。 **YES** SQL 索引は有効です。
INDEX_CREATE_TIMESTAMP	INDCREATED	TIMESTAMP	SQL 索引が作成されたときのタイム・スタンプ。
CREATE_TIMESTAMP	CREATED	TIMESTAMP	SQL 索引パーティションが作成されたときのタイム・スタンプ。
LAST_BUILD_TIMESTAMP	LASTBUILD	TIMESTAMP NULL 可能	SQL 索引が最後に再作成されたときのタイム・スタンプ。 SQL 索引が一度も再作成されていない場合は、NULL が入ります。
LAST_QUERY_USE	LASTQRYUSE	TIMESTAMP NULL 可能	使用状況統計が前回リセットされた後で、該当の SQL 索引が最後に照会で使用されたときのタイム・スタンプ。使用状況統計が前回リセットされた後でその SQL 索引が照会で一度も使用されていない場合には、NULL が入ります。
LAST_STATISTICS_USE	LASTSTUSE	TIMESTAMP NULL 可能	使用状況統計が前回リセットされた後で、該当の SQL 索引が最後にオプティマイザーで統計用に使用されたときのタイム・スタンプ。使用状況統計が前回リセットされた後でその SQL 索引が一度も統計用に使用されていない場合には、NULL が入ります。
QUERY_USE_COUNT	QRYUSECNT	BIGINT	使用状況統計が前回リセットされた後で、該当の SQL 索引が 1 つの照会の中で使用された回数。使用状況統計が前回リセットされた後でその SQL 索引が照会で一度も使用されていない場合には、0 が入ります。
QUERY_STATISTICS_COUNT	QRYSTCNT	BIGINT	使用状況統計が前回リセットされた後で、該当の SQL 索引がオプティマイザーで統計用に使用された回数。使用状況統計が前回リセットされた後でその SQL 索引が一度も統計用に使用されていない場合には、0 が入ります。

表 172. SYSINDEXSTAT ビュー (続き)

列名	システム列名	データ・タイプ	説明
LAST_USED_TIMESTAMP	LASTUSED	TIMESTAMP NULL 可能	アプリケーションによってネイティブ・レコード入出力操作または SQL 操作に該当の SQL 索引が最後に直接使用されたときのタイム・スタンプ。その SQL 索引が一度も使用されていない場合には、NULL が入ります。
DAYS_USED_COUNT	DAYSUSED	INTEGER	使用状況統計が前回リセットされた後で、アプリケーションによってネイティブ・レコード入出力操作または SQL 操作に該当の SQL 索引が直接使用された日数。使用状況統計が前回リセットされた後でその SQL 索引が一度も使用されていない場合には、0 が入ります。
LAST_RESET_TIMESTAMP	LASTRESET	TIMESTAMP NULL 可能	該当の SQL 索引について使用状況統計が前回リセットされたときのタイム・スタンプ。詳しくは、オブジェクト記述変更 (CHGOBJD) コマンドを参照してください。 SQL 索引の最終使用時のタイム・スタンプが一度もリセットされていない場合は、NULL が入ります。
NUMBER_KEY_COLUMNS	INDKEYS	BIGINT	SQL 索引キーを定義する列の数。
COLUMN_NAMES	COLNAMES	VARCHAR(1024)	SQL 索引キーを定義する列名のコンマ区切りリスト。すべての列名の長さが 1024 を超える場合は、列値の最後に「...」が返されます。
NUMBER_KEYS	NUMRIDS	BIGINT	SQL 索引内のキーの数。SQL 索引が無効な場合は、-1 が返されます。
INDEX_SIZE	SIZE	BIGINT	SQL 索引の二分木またはコード化ベクトル索引のサイズ (バイト数)。
NUMBER_PAGES	PAGES	BIGINT NULL 可能	SQL 索引内のページの数。 SQL 索引が無効な場合、またはコード化ベクトル索引である場合は、NULL が入ります。
LOGICAL_PAGE_SIZE	PAGE_SIZE	INTEGER NULL 可能	索引の論理ページ・サイズ。索引がコード化ベクトル索引である場合は、NULL が入ります。
UNIQUE	UNIQUE	VARCHAR(21)	SQL 索引がユニークであるかどうかを示します。 **UNIQUE** SQL 索引は UNIQUE 索引です。 **UNIQUE WHERE NOT NULL** SQL 索引は UNIQUE WHERE NOT NULL 索引です。 **FIFO** SQL 索引は非ユニーク先入れ先出し (FIFO) 索引です。 **LIFO** SQL 索引は非ユニーク後入れ後出し (LIFO) 索引です。 **FCFO** SQL 索引は非ユニーク先変更先出し (FCFO) 索引です。
MAXIMUM_KEY_LENGTH	KEY_LENGTH	INTEGER NULL 可能	SQL 索引のキーの最大長。 SQL 索引がコード化ベクトル索引の場合は、NULL が入ります。
UNIQUE_PARTIAL_KEY_VALUES	KEYCARDS	VARCHAR(96) NULL 可能	SQL 索引のユニーク部分キー値。コード化ベクトル索引の場合は、最初のユニーク部分キー値が索引キー全体の固有値の総数になります。返されるその他のユニーク部分キー値は、適用外です。
OVERFLOW_VALUES	OVERFLOW	INTEGER NULL 可能	コード化ベクトル索引からオーバーフローした特殊キー値の数。 SQL 索引がコード化ベクトル索引でない場合は、NULL が入ります。
EVI_CODE_SIZE	CODE_SIZE	INTEGER NULL 可能	コード化ベクトル索引のバイト・コードのサイズ。SQL 索引がコード化ベクトル索引でない場合は、NULL が入ります。

表 *172. SYSINDEXSTAT* ビュー (続き)

列名	システム列名	データ・タイプ	説明
SPARSE	SPARSE	VARCHAR(3)	SQL 索引に従属表のすべての行のキーが含まれるかどうかを示します。 **NO** 索引には、従属表のすべての行のキーが含まれます。 **YES** SQL 索引には WHERE 文節が組み込まれます。従属表のすべての行のキーは含まれません。
DERIVED_KEY	DERIVED	VARCHAR(3)	SQL 索引のキー列が式であるかどうかを示します。 **NO** SQL 索引のキー列は、式ではありません。 **YES** 少なくとも 1 つのキー列が式です。
PARTITIONED	PARTITION	VARCHAR(3)	SQL 索引がパーティション化されているか、パーティション化されていないかを示します。 **DISTRIBUTED** 索引は分散表で作成されています。 **NO** SQL 索引はパーティション化されていません (複数のパーティションにまたがっています)。 **YES** パーティション表で SQL 索引が作成されていないか、パーティション表で SQL 索引が作成されていて、パーティション化されているかのどちらかです (複数のパーティションにまたがっていません)。 基本表がパーティション化された表でない場合は、NULL 値が入ります。
ACCPTH_TYPE	ACCPTHTYPE	VARCHAR(4)	SQL 索引のタイプを示します。 **1 TB** SQL 索引は、最大 1 テラバイト (*MAX1TB) の 2 進基数索引です。 **4 GB** SQL 索引は、最大 4 ギガバイト (*MAX4GB) の 2 進基数索引です。 **EVI** SQL 索引は、コード化ベクトル索引です。
SORT_SEQUENCE	SRTSEQ	VARCHAR(12)	SQL 索引で照合順序を使用するかどうかを示します。 **BY HEX VALUE** SQL 索引では照合テーブルを使用しません。 ***LANGIDSHR** SQL 索引では共用重み照合順序 (SRTSEQ) を使用します。 ***LANGIDUNQ** SQL 索引では固有重み照合順序 (SRTSEQ) を使用します。 **ALTSEQ** SQL 索引では代替照合順序 (ALTSEQ) を使用します。
LANGUAGE_IDENTIFIER	LANGID	CHAR(3) NULL 可能	SQL 索引の言語 ID。照合順序が 16 進数の場合は NULL が入ります。

表 172. SYSINDEXSTAT ビュー (続き)

列名	システム列名	データ・タイプ	説明
SORT_SEQUENCE_SCHEMA	SRTSEQSCH	CHAR(10) NULL 可能	使用するソート順序のスキーマ名。ソート順序のスキーマ名がない場合は、NULL が入ります。
SORT_SEQUENCE_NAME	SRTSEQNAM	CHAR(10) NULL 可能	使用するソート順序の名前。ソート順序の名前がない場合は、NULL が入ります。
ESTIMATED_BUILD_TIME	ESTBLDTIME	INTEGER	SQL 索引の再作成に必要な概算時間 (秒)。
LAST_BUILD_TIME	LSTBLDTIME	INTEGER NULL 可能	索引が最後に作成されたときからの経過時間 (秒単位)。最新の作成情報が使用できない場合には、NULL が含まれます。
LAST_BUILD_KEYS	LSTBLDKEYS	BIGINT NULL 可能	索引が最後に作成されたときのキーの数。最新の作成情報が使用できない場合には、NULL が含まれます。
LAST_BUILD_DEGREE	LSTBLDDEG	SMALLINT NULL 可能	索引が最後に作成されたときの並列度。最新の作成情報が使用できない場合には、NULL が含まれます。
LAST_BUILD_TYPE	LSTBLDTYPE	CHAR(1) NULL 可能	最後の索引作成が、全体作成であったか、遅延保守キーによる作成であったかを示します。 **0** 索引の最後の再作成は、遅延保守キーによるものでした。 **1** 索引の最後の作成または再作成は、表内の行による全体作成でした。 その索引が一度も作成されていない場合には、NULL が入ります。
LAST_INVALIDATION_TIMESTAMP	LSTINVAL	TIMESTAMP NULL 可能	索引が最後にいつ無効化されたかを示します。その索引が一度も無効化されていない場合には、NULL が入ります。
INDEX_HELD	HELD	VARCHAR(3)	SQL 索引の再作成保留について、ユーザーが現在それを保留しているかどうかを示します。 **NO** SQL 索引の再作成は保留中でも、保留されてもいません。 **YES** SQL 索引の再作成保留はユーザーが保留しています。
MAINTENANCE	MAINT	VARCHAR(11) NULL 可能	SQL 索引の保守: **REBUILD** SQL 索引は保守されておらず、オープン時に再作成されます。 **DELAYED** SQL 索引の保守は、索引のオープン時まで延期されます。 **DO NOT WAIT** SQL 索引はすぐに保守されます。 SQL 索引がコード化ベクトル索引の場合は、NULL が入ります。
DELAYED_MAINT_KEYS	DLYKEYS	INTEGER NULL 可能	遅延保守索引の二分木に挿入する必要のあるキーの数。 SQL 索引が遅延保守索引でない場合は、NULL が入ります。

表 *172. SYSINDEXSTAT* ビュー (続き)

列名	システム列名	データ・タイプ	説明
RECOVERY	RECOVERY	VARCHAR(10) NULL 可能	SQL 索引のリカバリー属性: **DURING IPL** 必要であれば、IPL 時に SQL 索引がリカバリーされます。 **AFTER IPL** 必要であれば、IPL 後に SQL 索引がリカバリーされます。 **NEXT OPEN** 必要であれば、次回のオープン時に SQL 索引がリカバリーされます。 SQL 索引がコード化ベクトル索引の場合は、NULL が入ります。
ROUNDING_MODE	DECFLTRND	VARCHAR(8) NULL 可能	索引の DECFLOAT 丸めモードを示します。 **CEILING** ROUND_CEILING **DOWN** ROUND_DOWN **FLOOR** ROUND_FLOOR **HALFDOWN** ROUND_HALF_DOWN **HALFEVEN** ROUND_HALF_EVEN **HALFUP** ROUND_HALF_UP **UP** ROUND_UP DECFLOAT 列、関数、または定数を参照する式が索引にない場合は、NULL 値が入ります。
DECFLOAT_WARNING	DECFLTWRN	VARCHAR(3) NULL 可能	DECFLOAT 警告が返されるかどうかを指定します。 **NO** DECFLOAT 警告は返されません。 **YES** DECFLOAT 警告が返されます。 DECFLOAT 列、関数、または定数を参照する式が索引にない場合は、NULL 値が入ります。
LOGICAL_READS	LGLREADS	BIGINT	最後の IPL の後で SQL 索引に対して実行された論理読み取り操作の数。
SEQUENTIAL_READS	SEQREADS	BIGINT	最後の IPL の後で索引に対して実行された順次読み取り操作の数。
RANDOM_READS	RANREADS	BIGINT	最後の IPL の後で索引に対して実行されたランダム読み取り操作の数。
SEARCH_CONDITION	IXWHERECON	VARGRAPHIC(1024) CCSID 1200 NULL 可能	疎索引の場合はその索引の検索条件を示します。検索条件の長さが 1024 を超えた場合は、列値の最後に「...」が返されます。疎索引でない場合は、NULL が入ります。
SEARCH_CONDITION_HAS_UDF	IXWHEREUDF	VARCHAR(3) NULL 可能	疎索引の場合は、索引の検索条件にユーザー定義関数が含まれるかどうかを示します。疎索引でない場合は、NULL が入ります。 **NO** 索引検索条件には UDF は含まれません。 **YES** 索引検索条件に UDF が含まれます。

表 172. SYSINDEXSTAT ビュー (続き)

列名	システム列名	データ・タイプ	説明
KEEP_IN_MEMORY	KEEPINMEM	VARCHAR(3)	索引をメモリー内に保持するかどうかを示します。 **NO** メモリー設定がありません。 **YES** 可能な場合、索引をメモリーに保持します。
MEDIA_PREFERENCE	MEDIAPREF	VARCHAR(3)	索引のメディア設定を示します。 **ANY** メディア設定がありません。 **SSD** 可能な場合、索引を ソリッド・ステート・ディスク (SSD) に割り振ります。
INCLUDE_EXPRESSION	IXINCEXPR	VARGRAPHIC(1024) CCSID 1200 NULL 可能	索引の INCLUDE 式。索引に INCLUDE 式がない場合は、NULL が入ります。
OWNING_INDEX_SCHEMA	OWNINDSCH	VARCHAR(128)	索引を所有しているオブジェクトのスキーマの名前。
OWNING_INDEX_NAME	OWNINDNAME	VARCHAR(128)	索引を所有しているオブジェクトの名前。
OWNING_INDEX_TYPE	OWNINDTYPE	VARCHAR(11)	索引を所有しているオブジェクトのタイプ: **INDEX** 所有者は SQL 索引です。 **LOGICAL** 所有者は論理ファイルの一部です。 **PHYSICAL** 所有者はキー付き物理ファイルの一部です。 **PRIMARY KEY** 所有者は基本キー制約です。 **UNIQUE** 所有者はユニーク制約です。 **FOREIGN KEY** 所有者は外部キー制約です。
OWNING_INDEX_OWNER	OWNINDOWN	VARCHAR(128)	索引を所有しているオブジェクトの所有者。
OWNING_SYSTEM_INDEX_SCHEMA	OWNSYS_IXD	CHAR(10) NULL 可能	索引の所有者のシステム索引スキーマ名。所有者が制約の場合は、NULL が入ります。
OWNING_SYSTEM_INDEX_NAME	OWNSYS_IXN	CHAR(10) NULL 可能	索引の所有者のシステム名。所有者が制約の場合は、NULL が入ります。
OWNING_INDEX_TEXT	OWNLABEL	VARGRAPHIC(50) CCSID 1200 NULL 可能	索引を所有しているオブジェクトのテキスト。オブジェクトのテキストが存在しない場合は、NULL が入ります。
OWNING_INDEX_PARTITION	OWNINDMMBR	VARCHAR(128) NULL 可能	索引を所有しているオブジェクトのパーティション名またはメンバー名。所有者が制約の場合は、NULL が入ります。
SYSTEM_INDEX_SCHEMA	SYS_IXDNAM	CHAR(10)	システム索引スキーマ名。
SYSTEM_INDEX_NAME	SYS_IXNAME	CHAR(10)	システム索引名。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	システム表スキーマ名。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	システム表名。
SYSTEM_TABLE_MEMBER	SYS_MNAME	CHAR(10)	システム・メンバー名。

SYSJARCONTENTS

SYSJARCONTENTS 表には、SQL スキーマの jarid によって定義された各クラスごとに、行が 1 つずつ入ります。

次の表は、SYSJARCONTENTS 表の列について説明しています。

表 *173. SYSJARCONTENTS* 表

列名	システム列名	データ・タイプ	説明
JARSCHEMA	JARSCHEMA	VARCHAR(128)	jar_id が入っているスキーマの名前。
JAR_ID	JAR_ID	VARCHAR(128)	jar_id の名前。
CLASS	CLASS	VARCHAR(128)	クラスの名前。
CLASS_SOURCE	CLASSSRC	DBCLOB(10485760) CCSID 13488 NULL 可能	予約済み。 NULL 値が入ります。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。

SYSJAROBJECTS

SYSJAROBJECTS 表には、SQL スキーマの各 jarid ごとに、行が 1 つずつ入ります。

次の表は、SYSJAROBJECTS 表の列について説明しています。

表 *174. SYSJAROBJECTS* 表

列名	システム列名	データ・タイプ	説明
JARSCHEMA	JARSCHEMA	VARCHAR(128)	jar_id が入っているスキーマの名前。
JAR_ID	JAR_ID	VARCHAR(128)	jar_id の名前。
DEFINER	DEFINER	VARCHAR(128)	jarid の所有者の名前。
JAR_DATA	JAR_DATA	BLOB(104857600) NULL 可能	jar のバイト・コード。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。
JAR_CREATED	CREATEDTS	TIMESTAMP	Jar 作成のタイム・スタンプ。
LAST_ALTERED	ALTEREDTS	TIMESTAMP NULL 可能	予約済み。 NULL 値が入ります。
DEBUG_MODE	DEBUG_MODE	CHAR(1)	ルーチンがデバッグ可能かどうかを識別します。 **0** ルーチンはデバッグ不可能です。 **1** ルーチンは Unified Debugger でデバッグ可能です。 **2** ルーチンはシステム・デバッガーでデバッグ可能です。 **N** ルーチンは Unified Debugger によるデバッグは使用不可です。
DEBUG_DATA	DEBUG_DATA	CLOB(1048576) NULL 可能	予約済み。 NULL 値が入ります。

SYSKEYCST

SYSKEYCST ビューには、SQL スキーマにある各 UNIQUE KEY、PRIMARY KEY、または FOREIGN KEY ごとに、1 つ以上の行が入ります。固有キー制約または基本キー制約の列ごとに、および参照制約の参照列に対して、行が 1 つずつあります。

次の表は、SYSKEYCST ビューの列について説明しています。

表 *175. SYSKEYCST* ビュー

列名	システム列名	データ・タイプ	説明
CONSTRAINT_SCHEMA	CDBNAME	VARCHAR(128)	該当の制約が入っているスキーマの名前。
CONSTRAINT_NAME	RELNAME	VARCHAR(128)	制約の名前。
TABLE_SCHEMA	TDBNAME	VARCHAR(128)	該当の表が入っているスキーマの名前。
TABLE_NAME	TBNAME	VARCHAR(128)	表の名前。
COLUMN_NAME	COLNAME	VARCHAR(128)	列の名前。
ORDINAL_POSITION	COLSEQ	INTEGER	キー内における列の位置。
COLUMN_POSITION	COLNO	INTEGER	行内における列の位置。
TABLE_OWNER	CREATOR	VARCHAR(128)	表の所有者。
SYSTEM_COLUMN_NAME	SYS_CNAME	CHAR(10)	列のシステム名。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	表のシステム名。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	該当のスキーマ表が入っているスキーマのシステム名。
SYSTEM_CONSTRAINT_SCHEMA	SYS_CDNAME	CHAR(10)	制約が入っているシステム・スキーマの名前。

SYSKEYS

SYSKEYS ビューには、SQL スキーマにある索引 (SQL カタログの索引のキーを含む) の各列ごとに、行が 1 つずつ入ります。

次の表は、SYSKEYS ビューの列について説明しています。

表 *176. SYSKEYS* ビュー

列名	システム列名	データ・タイプ	説明
INDEX_NAME	IXNAME	VARCHAR(128)	索引の名前。 SQL の索引名が存在する場合は、その SQL の索引名になります。存在しない場合は、システムの索引名になります。
INDEX_OWNER	IXCREATOR	VARCHAR(128)	索引の所有者
COLUMN_NAME	COLNAME	VARCHAR(128)	該当のキーの列の名前。 SQL の列名が存在する場合は、その SQL の列名になります。存在しない場合は、システムの列名になります。
COLUMN_POSITION	COLNO	INTEGER	行内での列の位置を示す数値。
		NULL 可能	キー列が式である場合は、NULL 値が入ります。
ORDINAL_POSITION	COLSEQ	INTEGER	キー内における列の数値位置
ORDERING	ORDERING	CHAR(1)	キー内における列の順序: **A** 昇順 **D** 降順
INDEX_SCHEMA	IXDBNAME	VARCHAR(128)	該当の索引が入っているスキーマの名前
SYSTEM_COLUMN_NAME	SYS_CNAME	CHAR(10)	列のシステム名
SYSTEM_INDEX_NAME	SYS_IXNAME	CHAR(10)	索引のシステム名
SYSTEM_INDEX_SCHEMA	SYS_IDNAME	CHAR(10)	該当の索引が入っているスキーマのシステム名
COLUMN_IS_EXPRESSION	IXISEXP	CHAR(1)	キー列が式かどうかを示します。 **Y** キー列は式ではありません。 **N** キー列は式です。
EXPRESSION_HAS_UDF	IXEXPUDF	CHAR(1) NULL 可能	キー列が式かどうか、およびその式にユーザー定義関数が含まれるかどうかを示します。 **N** キー列は式ではないか、または式にユーザー定義関数が含まれません。 **Y** キー列は式で、その式に UDF が含まれます。
KEY_EXPRESSION	IXKEYEXP	DBCLOB(2097152) CCSID 1200 NULL 可能	キー列が式の場合は、その式が入ります。 キー列が式でない場合は、列名が入ります。

SYSMQTSTAT

SYSMQTSTAT ビューには、各マテリアライズ表パーティションごとに行が 1 つずつ入ります。

このビューを使用すると、指定した特定のマテリアライズ照会表または一連のマテリアライズ照会表についての情報を表示することができます。この情報は、System i ナビゲーターで「マテリアライズ照会表の表示」を選択したときに返される情報と同様のものです。

以下の表では、SYSMQTSTAT ビューの列について説明します。

表 *177. SYSMQTSTAT* ビュー

列名	システム列名	データ・タイプ	説明
MQT_SCHEMA	MQTSCHEMA	VARCHAR(128)	該当のマテリアライズ照会表が入っている SQL スキーマの名前。
MQT_NAME	MQTNAME	VARCHAR(128)	マテリアライズ照会表の名前。
MQT_PARTITION	MQTMEMBER	VARCHAR(128)	マテリアライズ照会表のパーティション名またはメンバー名。
MQT_OWNER	MQTOWNER	VARCHAR(128)	マテリアライズ照会表の所有者。
MQT_TEXT	LABEL	VARGRAPHIC(50) CCSID 1200 NULL 可能	マテリアライズ照会表のテキスト。マテリアライズ照会表にテキストが存在しない場合は、NULL が入ります。
ENABLED	ENABLED	VARCHAR(3)	マテリアライズ照会表を使用可能にするかどうかを示します。 **NO** マテリアライズ照会表を使用可能にしません。 **YES** マテリアライズ照会表を使用可能にしてデータベース・マネージャーが使用できるようにします。
CREATE_TIMESTAMP	CREATED	TIMESTAMP	マテリアライズ照会表が作成されたときのタイム・スタンプ。
REFRESH_TIME	REFRESHDTS	TIMESTAMP NULL 可能	マテリアライズ照会表が最後にリフレッシュされたときのタイム・スタンプ。マテリアライズ照会表が一度もリフレッシュされていない場合は、NULL が入ります。
LAST_QUERY_USE	LASTQRYUSE	TIMESTAMP NULL 可能	使用状況統計が前回リセットされた後で、該当のマテリアライズ照会表が最後に照会で使用されたときのタイム・スタンプ。使用状況統計が前回リセットされた後でそのマテリアライズ照会表が照会で一度も使用されていない場合には、NULL が入ります。
LAST_STATISTICS_USE	LASTSTUSE	TIMESTAMP NULL 可能	使用状況統計が前回リセットされた後で、該当のマテリアライズ照会表が最後にオプティマイザーで統計用に使用されたときのタイム・スタンプ。使用状況統計が前回リセットされた後でそのマテリアライズ照会表が統計用に一度も使用されていない場合には、NULL が入ります。
QUERY_USE_COUNT	QRYUSECNT	BIGINT	使用状況統計が前回リセットされた後で、該当のマテリアライズ照会表が 1 つの照会の中で使用された回数。使用状況統計が前回リセットされた後でそのマテリアライズ照会表が照会で一度も使用されていない場合には、0 が入ります。
QUERY_STATISTICS_COUNT	QRYSTCNT	BIGINT	使用状況統計が前回リセットされた後で、該当のマテリアライズ照会表がオプティマイザーで統計用に使用された回数。使用状況統計が前回リセットされた後でそのマテリアライズ照会表が統計用に一度も使用されていない場合には、0 が入ります。

表 177. SYSMQTSTAT ビュー (続き)

列名	システム列名	データ・タイプ	説明
LAST_USED_TIMESTAMP	LASTUSED	TIMESTAMP NULL 可能	アプリケーションによってネイティブ・レコード入出力操作または SQL 操作に該当のマテリアライズ照会表が最後に直接使用されたときのタイム・スタンプ。そのマテリアライズ照会表が一度も使用されていない場合には、NULL が入ります。
DAYS_USED_COUNT	DAYSUSED	INTEGER	使用状況統計が前回リセットされた後で、アプリケーションによってネイティブ・レコード入出力操作または SQL 操作に該当のマテリアライズ照会表が直接使用された日数。使用状況統計が前回リセットされた後でそのマテリアライズ照会表が一度も使用されていない場合には、0 が入ります。
LAST_RESET_TIMESTAMP	LASTRESET	TIMESTAMP NULL 可能	該当のマテリアライズ照会表について使用状況統計が前回リセットされたときのタイム・スタンプ。詳しくは、オブジェクト記述変更 (CHGOBJD) コマンドを参照してください。マテリアライズ照会表の最終使用時のタイム・スタンプが一度もリセットされていない場合は、NULL が入ります。
NUMBER_ROWS	CARD	BIGINT	マテリアライズ照会表の行の数。
MQT_SIZE	SIZE	BIGINT	マテリアライズ照会表のサイズ (バイト数)。
LAST_CHANGE_TIMESTAMP	LASTCHG	TIMESTAMP NULL 可能	マテリアライズ照会表が最後に変更されたときのタイム・スタンプ。使用状況統計が前回リセットされた後でそのマテリアライズ照会表が一度も変更されていない場合には、NULL が入ります。
MAINTENANCE	MAINTAIN	VARCHAR(6)	マテリアライズ照会表の保守方法を示します。 **SYSTEM** マテリアライズ照会表はシステムが保守します。 **USER** マテリアライズ照会表はユーザーが保守します。
INITIAL_DATA	INITIAL	VARCHAR(19)	マテリアライズ照会表の初期データを示します。 **INITIALLY DEFERRED** マテリアライズ照会表の作成時に、そこにデータを挿入しません。 **INITIALLY IMMEDIATE** マテリアライズ照会表の作成時に、そこにデータを挿入します。
REFRESH	REFRESH	VARCHAR(9)	マテリアライズ照会表内のデータをリフレッシュできる時期を示します。 **DEFERRED** マテリアライズ照会表内のデータは、REFRESH TABLE ステートメントを使用していつでもリフレッシュできます。 **IMMEDIATE** マテリアライズ照会表内のデータは即時にリフレッシュされます。

表 177. SYSMQTSTAT ビュー (続き)

列名	システム列名	データ・タイプ	説明
ISOLATION	ISOLATION	VARCHAR(27)	マテリアライズ照会表のリフレッシュ時に使用される分離レベルを示します。 **NO COMMIT** 分離レベルは NO COMMIT です。 **UNCOMMITTED READ** 分離レベルは UNCOMMITTED READ です。 **CURSOR STABILITY** 分離レベルは CURSOR STABILITY です。 **CURSOR STABILITY KEEP LOCKS** 分離レベルは CURSOR STABILITY KEEP LOCKS です。 **READ STABILITY** 分離レベルは READ STABILITY です。 **REPEATABLE READ** 分離レベルは REPEATABLE READ です。
SORT_SEQUENCE	SRTSEQ	VARCHAR(12)	マテリアライズ照会表で照合順序を使用するかどうかを示します。 **BY HEX VALUE** マテリアライズ照会表では照合テーブルを使用しません。 ***LANGIDSHR** マテリアライズ照会表では共用重みソート順序 (SRTSEQ) を使用します。 ***LANGIDUNQ** マテリアライズ照会表では固有重みソート順序 (SRTSEQ) を使用します。 **ALTSEQ** マテリアライズ照会表では代替照合順序 (ALTSEQ) を使用します。
LANGUAGE_IDENTIFIER	LANGID	CHAR(3) NULL 可能	マテリアライズ照会表の言語 ID。ソート順序が 16 進数の場合は NULL が入ります。
SORT_SEQUENCE_SCHEMA	SRTSEQSCH	CHAR(10) NULL 可能	ソート順序表のシステム・スキーマ。 ソート順序が 16 進数の場合は NULL が入ります。
SORT_SEQUENCE_NAME	SRTSEQNAME	CHAR(10) NULL 可能	ソート順序表の名前。ソート順序が 16 進数の場合は NULL が入ります。
MQT_RESTORE_DEFERRED	MQTRSTDFR	VARCHAR(3)	MQT の復元がその従属表の 1 つの復元まで据え置かれるかどうかを示します。 **NO** MQT の復元は、その従属表の 1 つの復元まで据え置かれることはありません。 **YES** MQT の復元は、その従属表の 1 つの復元まで据え置かれます。

表 177. SYSMQTSTAT ビュー (続き)

列名	システム列名	データ・タイプ	説明
ROUNDING_MODE	DECFLTRND	VARCHAR(8) NULL 可能	マテリアライズ照会表の DECFLOAT 丸めモードを示します。 **CEILING** ROUND_CEILING **DOWN** ROUND_DOWN **FLOOR** ROUND_FLOOR **HALFDOWN** ROUND_HALF_DOWN **HALFEVEN** ROUND_HALF_EVEN **HALFUP** ROUND_HALF_UP **UP** ROUND_UP DECFLOAT 列、関数、または定数を参照する式がマテリアライズ照会表にない場合は、NULL 値が入ります。
DECFLOAT_WARNING	DECFLTWRN	VARCHAR(3) NULL 可能	DECFLOAT 警告が返されるかどうかを指定します。 **NO** DECFLOAT 警告は返されません。 **YES** DECFLOAT 警告が返されます。 DECFLOAT 列、関数、または定数を参照する式がマテリアライズ照会表にない場合は、NULL 値が入ります。
MQT_DEFINITION	MQTDEF	VARGRAPHIC(5000) CCSID 1200	マテリアライズ照会表の照会。照会の長さが 5000 を超えた場合は、列値の最後に「...」が返されます。
SYSTEM_MQT_SCHEMA	SYS_MQDNAM	CHAR(10)	システム・マテリアライズ照会表のスキーマ名。
SYSTEM_MQT_NAME	SYS_MQNAME	CHAR(10)	システム・マテリアライズ照会表名。

SYSPACKAGE

SYSPACKAGE ビューには、SQL のスキーマにある各 SQL パッケージごとに、行が 1 つずつ入ります。

次の表は、SYSPACKAGE ビューの列について説明しています。

表 *178. SYSPACKAGE* ビュー

列名	システム列名	データ・タイプ	説明
PACKAGE_CATALOG	LOCATION	VARCHAR(128)	SQL パッケージのリレーショナル・データベース名 (RDBNAME)
PACKAGE_SCHEMA	COLLID	VARCHAR(128)	スキーマの名前
PACKAGE_NAME	NAME	VARCHAR(128)	SQL パッケージの名前
PACKAGE_OWNER	OWNER	VARCHAR(128)	SQL パッケージの所有者
PACKAGE_CREATOR	CREATOR	VARCHAR(128)	SQL パッケージの作成者
CREATION_TIMESTAMP	TIMESTAMP	CHAR(26)	SQL パッケージ作成時のタイム・スタンプ
DEFAULT_SCHEMA	QUALIFIER	VARCHAR(128)	修飾されていない表、ビュー、および索引の暗黙の名前
PROGRAM_NAME	PROGNAME	VARCHAR(128)	パッケージ作成の元になったプログラムの名前
PROGRAM_SCHEMA	LIBRARY	VARCHAR(128)	該当のプログラムが入っているスキーマの名前
PROGRAM_CATALOG	RDB	VARCHAR(128)	該当のプログラムが常駐するリレーショナル・データベースの名前
ISOLATION	ISOLATION	CHAR(2)	分離オプションの指定: RR 反復可能読み取り (*RR) RS 読み取り固定 (*ALL) CS カーソル固定 (*CS) UR 非コミット読み取り (*CHG) NO なし (*NONE)
QUOTE	QUOTE	CHAR(1)	エスケープ文字の指定 (Y/N): Y = 引用符 N = アポストロフィ
COMMA	COMMA	CHAR(1)	コンマ・オプションの指定 (Y/N): Y = コンマ N = ピリオド
PACKAGE_TEXT	LABEL	VARCHAR(50)	ユーザーが LABEL ステートメントで指定する文字ストリング。
LONG_COMMENT	REMARKS	VARCHAR(2000)	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、NULL 値が入ります。
CONSISTENCY_TOKEN	CONTOKEN	CHAR(8) ビット・データ用	パッケージの整合性トークン
SYSTEM_PACKAGE_NAME	SYS_NAME	CHAR(10)	パッケージのシステム名
SYSTEM_PACKAGE_SCHEMA	SYS_DNAME	CHAR(10)	該当のパッケージが入っているスキーマのシステム名
SYSTEM_DEFAULT_SCHEMA	SYS_DDNAME	CHAR(10)	修飾されていない表、ビュー、索引、およびパッケージの暗黙の修飾子のシステム名。
SYSTEM_PROGRAM_NAME	SYS_PNAME	CHAR(10)	プログラムのシステム名。
SYSTEM_PROGRAM_SCHEMA	SYS_PDNAME	CHAR(10)	該当のプログラムが入っているスキーマのシステム名。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。

表 *178. SYSPACKAGE* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
ROUNDING_MODE	DECFLTRND	CHAR(1)	パッケージの丸めモード。 **C** ROUND_CEILING **D** ROUND_DOWN **F** ROUND_FLOOR **G** ROUND_HALF_DOWN **E** ROUND_HALF_EVEN **H** ROUND_HALF_UP **U** ROUND_UP
CONCURRENTACCESSRESOLUTION	CONCURRENT	CHAR(1)	並行アクセスの解決方法を指定します。 ブランク 指定しない **W** WAIT FOR OUTCOME **U** USE CURRENTLY COMMITTED

SYSPACKAGEAUTH

SYSPACKAGEAUTH ビューには、パッケージに対して認可された各特権ごとに、行が 1 つずつ入ります。このカタログ・ビューを使用して、特定ユーザーが特定のパッケージに対する権限を持っているかどうかを判別することはできないので、注意してください。なぜなら、パッケージを使用するための特権は、グループ・ユーザー・プロファイルまたは特殊権限 (*ALLOBJ など) を通じて獲得できるからです。

以下の表では、SYSPACKAGEAUTH ビューの列について説明します。

表 *179. SYSPACKAGEAUTH* ビュー

列名	システム列名	データ・タイプ	説明
GRANTOR	GRANTOR	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
GRANTEE	GRANTEE	VARCHAR(128)	特権を認可する対象のユーザー・プロファイル。
PACKAGE_SCHEMA	COLLID	VARCHAR(128)	スキーマの名前
PACKAGE_NAME	NAME	VARCHAR(128)	SQL パッケージの名前
PRIVILEGE_TYPE	PRIVTYPE	VARCHAR(10)	認可される特権 : **ALTER** パッケージを変更する特権。 **EXECUTE** パッケージを実行する特権。
IS_GRANTABLE	GRANTABLE	VARCHAR(3)	特権を他のユーザーに認可できるかどうかを示します。 **NO** 特権は認可できません。 **YES** 特権を認可できます。
AUTHORIZATION_LIST	AUTL	VARCHAR(10) NULL 可能	特権が権限を介して認可されている場合、権限リストの名前が入ります。 特権が権限リストを介して認可されるのではない場合は、NULL 値が入ります。
SYSTEM_PACKAGE_SCHEMA	SYS_DNAME	CHAR(10)	該当のパッケージが入っているスキーマのシステム名
SYSTEM_PACKAGE_NAME	SYS_NAME	CHAR(10)	パッケージのシステム名

SYSPACKAGESTAT

SYSPACKAGESTAT ビューには、SQL スキーマにある各 SQL パッケージごとに、行が 1 つずつ入ります。

以下の表では、SYSPACKAGESTAT ビュー内の列について説明します。

表 *180. SYSPACKAGESTAT* ビュー

列名	システム列名	データ・タイプ	説明
PACKAGE_SCHEMA	COLLID	VARCHAR(128)	スキーマの名前
PACKAGE_NAME	NAME	VARCHAR(128)	SQL パッケージの名前
PACKAGE_OWNER	OWNER	VARCHAR(128)	SQL パッケージの所有者
PACKAGE_CREATOR	CREATOR	VARCHAR(128)	SQL パッケージの作成者
CREATION_TIMESTAMP	TIMESTAMP	CHAR(26)	SQL パッケージ作成時のタイム・スタンプ
DEFAULT_SCHEMA	QUALIFIER	VARCHAR(128)	修飾されていない表、ビュー、および索引の暗黙の名前
ISOLATION	ISOLATION	CHAR(2)	分離オプションの指定: **RR** 反復可能読み取り (*RR) **RS** 読み取り固定 (*ALL) **CS** カーソル固定 (*CS) **UR** 非コミット読み取り (*CHG) **NC** コミットなし (*NONE)
CONCURRENTACCESSRESOLUTION	CONCURRENT	CHAR(1)	並行アクセスの解決方法を指定します。 ブランク 指定しない **W** WAIT FOR OUTCOME **U** USE CURRENTLY COMMITTED
SECONDARY_SPACE_COUNT	PKG_SPACES	INTEGER	パッケージ内のスペースの数。
PENDING_FULL	PENDFULL	VARCHAR(3) NULL 可能	パッケージがフル保留状態であるかどうかを示します。 **NO** パッケージはフル保留状態ではありません。 **YES** パッケージはフル保留状態です。 DRDA パッケージの場合は、NULL が入ります。
PACKAGE_TYPE	PKG_TYPE	VARCHAR(16)	パッケージのタイプを示します。 **EXTENDED DYNAMIC** パッケージは拡張動的パッケージです。 **DRDA** パッケージは DRDA パッケージです。
NUMBER_STATEMENTS	NBRSTMTS	INTEGER	パッケージ内の SQL ステートメントの数。
PACKAGE_USED_SIZE	PKSIZE	INTEGER	パッケージ内の SQL ステートメントおよびアクセス・プランに使用されるバイトの数。
NUMBER_DUMMIES	NBRDUMMIES	INTEGER NULL 可能	パッケージ内のダミー・ステートメントの数。 DRDA パッケージの場合は、NULL が入ります。
NUMBER_COMPRESSIONS	PGM_CMP	INTEGER NULL 可能	パッケージが圧縮された回数。 DRDA パッケージの場合は、NULL が入ります。
STATEMENT_CONTENTION_COUNT	CONTENTION	BIGINT	新規アクセス・プランを保管しようとしたときに競合が発生した回数。

表 *180. SYSPACKAGESTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
EXTENDED_INDICATOR	EXTIND	VARCHAR(9)	EXTIND 属性を示します。 ***EXTIND** 拡張標識サポートは有効です。 ***NOEXTIND** 拡張標識サポートは有効ではありません。
C_NUL_REQUIRED	CNULRQD	VARCHAR(10)	CNULRQD 属性を示します。 ***CNULRQD** C NUL は必須です。 ***NOCNULRQD** C NUL は必須ではありません。
NAMING	NAMING	VARCHAR(4)	NAMING 属性を示します。 ***SYS** これはシステムによる命名です。 ***SQL** これは SQL による命名です。
TARGET_RELEASE	TGTRLS	VARCHAR(6)	パッケージのターゲット・リリースを示します (VxRxMx)。
EARLIEST_POSSIBLE_RELEASE	MINRLS	VARCHAR(6) NULL 可能	パッケージ内のすべての SQL ステートメントをサポートする最も古い IBM i リリースを示します (VxRxMx)。 **ANY** ステートメントは、サポートされている任意の IBM i リリースで有効です。 **VxRxMx** ステートメントは、IBM i VxRxMx リリースまたはそれ以降で有効です。 最も古いリリースがまだ判別されていない場合、NULL が入ります。
RDB	RDB	VARCHAR(18)	パッケージについて指定された RDB を示します。 **rdb-name** リレーショナル・データベースの名前。 ***NONE** リレーショナル・データベースは指定されていません。
CONSISTENCY_TOKEN	CONTOKEN	VARBINARY(8) NULL 可能	パッケージの整合性トークンを示します。 パッケージが DRDA パッケージでない場合は、NULL が入ります。
ALLOW_COPY_DATA	ALWCPYDTA	VARCHAR(9)	ALWCPYDTA 属性を示します。 ***NO** データのコピーを使用することはできません。 ***OPTIMIZE** 結果としてパフォーマンスがよくなる可能性がある場合は常に、データのコピーが許可されます。 ***YES** 必要な場合に限り、データのコピーが許可されます。

表 *180. SYSPACKAGESTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
LOB_FETCH_OPTIMIZATION	OPTLOB	VARCHAR(9)	LOB 最適化属性を示します。 ***OPTLOB** カーソルの最初の FETCH によって、それ以降のすべての FETCH においてそのカーソルが LOB および XML 結果列にどのように使用されるかが決定されます。 ***NOOPTLOB** どの FETCH も LOB または XML 結果列を取り出してロケーターまたは変数のいずれかに入れることができます。
DECIMAL_POINT	DECPNT	VARCHAR(7)	SQL ステートメントで使用される数値定数の小数点を示します。 ***PERIOD** 小数点にピリオドを使用します。 ***COMMA** 小数点にコンマを使用します。
SQL_STRING_DELIMITER	STRDLM	VARCHAR(9)	SQL ステートメントでストリング区切り文字として使用される文字を示します。 ***APOSTSQL** ストリング区切り文字はアポストロフィ (') です。 ***QUOTESQL** ストリング区切り文字は引用符 (") です。
DATE_FORMAT	DATFMT	VARCHAR(4)	DATFMT 属性を示します。 ***JOB** 実行時にジョブで指定される日付形式を使用します。 ***USA** 日付形式は *USA です。 ***ISO** 日付形式は *ISO です。 ***EUR** 日付形式は *EUR です。 ***JIS** 日付形式は *JIS です。 ***MDY** 日付形式は *MDY です。 ***DMY** 日付形式は *DMY です。 ***YMD** 日付形式は *YMD です。 ***JUL** 日付形式は *JUL です。
DATE_SEPARATOR	DATSEP	CHAR(1)	日付区切り記号を示します。
TIME_FORMAT	TIMFMT	VARCHAR(4)	TIMFMT 属性を示します。 ***JOB** 実行時にジョブで指定される時刻形式を使用します。 ***USA** 時刻形式は *USA です。 ***ISO** 時刻形式は *ISO です。 ***EUR** 時刻形式は *EUR です。 ***JIS** 時刻形式は *JIS です。 ***HMS** 時刻形式は *HMS です。
TIME_SEPARATOR	TIMSEP	CHAR(1)	時刻区切り記号を示します。

表 *180. SYSPACKAGESTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
DYNAMIC_DEFAULT_SCHEMA	DYNDFTCOL	VARCHAR(4) NULL 可能	動的 SQL ステートメントの暗黙の修飾に DFTRDBCOL の値を使用するかどうかを指定します。 ***NO** DFTDRBCOL で指定されているスキーマは、動的 SQL ステートメント用に使用されません。 ***YES** DFTDRBCOL で指定されているスキーマは、動的 SQL ステートメント用に使用されます。 デフォルトのスキーマ (DFTRDBCOL) が指定されていなかった場合は、NULL が入ります。
CURRENT_RULES	SQLCURRULE	VARCHAR(4)	SQLCURRULE 属性を示します。 ***DB2** すべての SQL ステートメントの意味体系は、デフォルトにより、Db2 用に設定された規則に従います。 ***STD** すべての SQL ステートメントの意味体系は、デフォルトにより、 ISO および ANSI SQL の規格用に設定された規則に従います。
ALLOW_BLOCK	ALWBLK	VARCHAR(8)	ALWBLK 属性を示します。 ***ALLREAD** 読み取り専用カーソルの場合は行がブロックされます。 ***NONE** カーソルに関するデータの検索のために、行はブロックされません。 ***READ** 以下の場合は、カーソルに関するデータの読み取り専用検索で、レコードがブロックされます。 • コミットメント制御 (COMMIT) パラメーターに *NONE が指定されているとき。 • FOR READ ONLY 文節によってカーソルが宣言されたとき、またはカーソルに関して位置指定 UPDATE ステートメントまたは DELETE ステートメントを実行できる動的ステートメントがないとき。
DELAY_PREPARE	DLYPRP	VARCHAR(4)	DLYPRP 属性を示します。 ***NO** 動的ステートメントの妥当性検査は、動的ステートメントの作成時に実行されます。 ***YES** 動的ステートメントの妥当性検査は、その動的ステートメントが使用されるときまで遅延されます。

表 *180. SYSPACKAGESTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
USER_PROFILE	USRPRF	VARCHAR(7)	権限検査に使用するユーザー・プロファイルを指定します。 ***USER** パッケージ内のステートメントを実行するユーザーのプロファイルが使用されます。 ***OWNER** パッケージの所有者とパッケージ内のステートメントを実行するユーザーの両方のプロファイルが使用されます。 ***NAMING** 命名規則が *SQL の場合、*OWNER が使用されます。命名規則が *SYS の場合、*USER が使用されます。
DYNAMIC_USER_PROFILE	DYNUSRPRF	VARCHAR(6)	動的 SQL ステートメントに使用するユーザー・プロファイルを指定します。 ***USER** ローカル動的 SQL ステートメントは、ジョブまたはスレッドのプロファイルのもとで実行されます。分散動的 SQL ステートメントは、アプリケーション・サーバー・ジョブのプロファイルのもとで実行されます。 ***OWNER** ローカル動的 SQL ステートメントは、パッケージの所有者のプロファイルのもとで実行されます。分散動的 SQL ステートメントは、SQL パッケージの所有者のプロファイルのもとで実行されます。
SORT_SEQUENCE	SRTSEQ	VARCHAR(12)	パッケージで照合順序を使用するかどうかを示します。 **BY HEX VALUE** パッケージでは照合テーブルを使用しません。 ***LANGIDSHR** パッケージでは共用重みソート順序 (SRTSEQ) を使用します。 ***LANGIDUNQ** パッケージでは固有重みソート順序 (SRTSEQ) を使用します。 **ALTSEQ** パッケージでは、代替照合順序 (ALTSEQ) を使用します。
LANGUAGE_IDENTIFIER	LANGID	CHAR(3) NULL 可能	言語 ID ソート順序。 ソート順序が *LANGIDSHR または *LANGIDUNQ ではない場合は、NULL が入ります。
SORT_SEQUENCE_SCHEMA	SRTSEQSCH	CHAR(10) NULL 可能	ソート順序表のシステム・スキーマ。 ソート順序が 16 進数の場合は NULL が入ります。
SORT_SEQUENCE_NAME	SRTSEQNAME	CHAR(10) NULL 可能	ソート順序表の名前。ソート順序が 16 進数の場合は NULL が入ります。

表 *180. SYSPACKAGESTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
RDB_CONNECTION_METHOD	RDBCNNMTH	VARCHAR(4)	CONNECT ステートメントに使用する意味体系を指定します。 ***RUW** CONNECT (タイプ 1) の意味体系は、リモート作業単位をサポートするのに使用されます。 ***DUW** CONNECT (タイプ 2) の意味体系は、分散作業単位をサポートするのに使用されます。
DECRESULT_MAXIMUM_PRECISION	DECMAXPRC	SMALLINT	最大精度を指定します。 **31** 最大精度は 31 です。 **63** 最大精度は 63 です。
DECRESULT_MAXIMUM_SCALE	DECMAXSCL	SMALLINT	結果データ・タイプについて返される最大位取り (小数点の右側にある小数部の桁数)。
DECRESULT_MINIMUM_DIVIDE_SCALE	DECMINDIV	SMALLINT	中間および結果の両方のデータ・タイプについて返される最小除算位取り (小数点の右側にある小数部の桁数)。
DECFLOAT_ROUNDING_MODE	DECFLTRND	VARCHAR(8) NULL 可能	DECFLOAT 丸めモードを示します。 **CEILING** ROUND_CEILING **DOWN** ROUND_DOWN **FLOOR** ROUND_FLOOR **HALFDOWN** ROUND_HALF_DOWN **HALFEVEN** ROUND_HALF_EVEN **HALFUP** ROUND_HALF_UP **UP** ROUND_UP
DECFLOAT_WARNING	DECFLTWRN	VARCHAR(3)	DECFLOAT 警告が返されるかどうかを指定します。 **NO** DECFLOAT 警告は返されません。 **YES** DECFLOAT 警告が返されます。
SQLPATH	SQLPATH	VARCHAR(3483) NULL 可能	SQL パスを示します。 SQL パスが指定されていない場合は、NULL 値が入ります。
LAST_USED_TIMESTAMP	LASTUSED	TIMESTAMP NULL 可能	パッケージが最後に使用されたときのタイム・スタンプ。そのパッケージが一度も使用されていない場合には、NULL が入ります。
DAYS_USED_COUNT	DAYSUSED	INTEGER	使用状況統計が最後にリセットされたとき以降に、パッケージが使用された日数。使用状況統計が前回リセットされた後でそのパッケージが一度も使用されていない場合には、0 が入ります。
LAST_RESET_TIMESTAMP	LASTRESET	TIMESTAMP NULL 可能	使用状況統計が前回リセットされたときのタイム・スタンプ。統計が一度もリセットされていない場合は、NULL が入ります。
SYSTEM_PACKAGE_NAME	SYS_NAME	CHAR(10)	パッケージのシステム名
SYSTEM_PACKAGE_SCHEMA	SYS_DNAME	CHAR(10)	該当のパッケージが入っているスキーマのシステム名
IASP_NUMBER	IASPNUMBER	INTEGER	独立補助記憶域プール (IASP) 番号を指定します。

表 *180. SYSPACKAGESTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
SYSTEM_TIME_SENSITIVE	SYSTIME	VARCHAR(3) NULL 可能	CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターがパッケージ内の静的および動的 SQL ステートメントに影響するかどうかを指定します。 **YES** システム期間テンポラル表への参照は、CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターの値の影響を受けます。 **NO** システム期間テンポラル表への参照は、CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターの値の影響を受けません。 パッケージが 7.3 より前のバージョンで作成されたものである場合は、NULL が入り、これは NO と同じように扱われます。

SYSPACKAGESTMTSTAT

SYSPACKAGESTMTSTAT ビューには、すべての SQL パッケージ内の各 SQL ステートメントごとに、行が 1 つずつ入ります。

次の表は、SYSPACKAGESTMTSTAT ビューの列について説明しています。

表 *181. SYSPACKAGESTMTSTAT* ビュー

列名	システム列名	データ・タイプ	説明
PACKAGE_SCHEMA	COLLID	VARCHAR(128)	スキーマの名前。
PACKAGE_NAME	NAME	VARCHAR(128)	SQL パッケージの名前。
SPACE_NAME	SPCNAME	VARCHAR(10) NULL 可能	拡張動的パッケージのパッケージ内の生成スペース名。内部スペースが 1 つだけの DRDA パッケージの場合、NULL 値が入ります。
STATEMENT_NUMBER	STMTNBR	INTEGER	スペースにおけるこのステートメントの番号。
STATEMENT_NAME	STMTNAME	VARCHAR(128) NULL 可能	ステートメントの名前。 DRDA パッケージの場合、NULL 値が入ります。
NUMBER_TIMES_PREPARED	NBRPREP	INTEGER	このステートメントが準備された回数。
NUMBER_TIMES_EXECUTED	NBREXEC	INTEGER	このステートメントが実行された回数。 CALL、SET、および VALUES INTO のステートメントでこの値は保守されません。
ROWS_AFFECTED	ROWCNT	INTEGER	このステートメントのすべての実行でフェッチ、更新、挿入、または削除された合計行数。ステートメントが FETCH、SET、VALUES INTO、UPDATE、INSERT、DELETE、および MERGE でない場合は、0 が入ります。カーソルを使用して実装されていない SET または VALUES INTO ステートメントの場合、0 が入ります。
COMPRESSES_SINCE_LAST_USED	CMPLU	INTEGER	このステートメントの最後の実行以降にパッケージでストレージが圧縮された回数。この値は、SQL_STMT_COMPRESS_MAX QAQQINI オプションを処理する際に、ステートメントの QDT に使用されるスペースを削除する時期を決定するために使用されます。 DRDA パッケージの場合、常に 0 になります。
PARAMETER_MARKER_CONVERTED	PMCNVT	CHAR(3)	動的ステートメントの場合に、ステートメントを再使用するためにリテラルがパラメーター・マーカーに変換されたかどうかを示します。 **YES** リテラルはパラメーター・マーカーに変換されました。 **NO** 変換が行われなかったか、ステートメントが動的ではありません。
WITH_HOLD	WITHHOLD	CHAR(3) NULL 可能	ステートメントに WITH HOLD オプションを指定します。 **YES** WITH HOLD 文節が指定されました。 文節が指定されていない場合は、NULL 値が入ります。
FETCH_ONLY	FETCHONLY	CHAR(3) NULL 可能	ステートメントに FOR READ ONLY オプションを指定します。 **YES** FOR READ ONLY 文節が指定されました。 文節が指定されていない場合は、NULL 値が入ります。

表 181. SYSPACKAGESTMTSTAT ビュー (続き)

列名	システム列名	データ・タイプ	説明
CONCURRENTACCESSRESOLUTION	CONCURRENT	CHAR(1) NULL 可能	ステートメントに並行アクセスの解決方法を指定します。 **W** WAIT FOR OUTCOME **U** USE CURRENTLY COMMITTED **S** SKIP LOCKED DATA ステートメント・レベルで並行アクセスが指定されなかった場合は、NULL 値が入ります。
NUMBER_REBUILDS	NBRREBLD	INTEGER	QDT またはアクセス・プランが再作成された回数。
ISOLATION	ISOLATION	CHAR(2) NULL 可能	分離オプションの指定: **RR** 反復可能読み取り (*RR) **RS** 読み取り固定 (*ALL) **CS** カーソル固定 (*CS) **UR** 非コミット読み取り (*CHG) **NC** コミットなし (*NONE) ステートメント・レベルで分離レベルが指定されなかった場合は、NULL 値が入ります。
NUMBER_ROWS_TO_OPTIMIZE	OPTROWS	INTEGER NULL 可能	OPTIMIZE FOR *n* ROWS 文節で指定された行数。 -1 は、値 *ALL が指定されたことを意味します。 文節が指定されていない場合は、NULL 値が入ります。
NUMBER_ROWS_TO_FETCH	FETCHROWS	INTEGER NULL 可能	FETCH FIRST *n* ROWS 文節で指定された行数。 文節が指定されていない場合は、NULL 値が入ります。
LAST_QDT_REBUILD_REASON	QDTRBLD	CHAR(2) NULL 可能	最後の QDT 再作成の理由コード。 QQRID = 1000 の場合、これは STRDBMON 出力ファイルの列 QVC22 に対応します。 ステートメントが QDT を使用しない場合、またはステートメントで QDT が一度も再作成されていない場合は、NULL 値が入ります。
STATEMENT_TEXT	STMTTEXT	DBCLOB(2M) CCSID(1200)	SQL ステートメントのテキスト。
SYSTEM_PACKAGE_NAME	SYS_NAME	CHAR(10)	パッケージのシステム名
SYSTEM_PACKAGE_SCHEMA	SYS_DNAME	CHAR(10)	該当のパッケージが入っているスキーマのシステム名
ACCESS_PLAN_LENGTH	AP_LENGTH	INTEGER	ステートメントの QDT とアクセス・プランに使用されるバイトの数。

SYSPARMS

SYSPARMS 表には、CREATE PROCEDURE ステートメントによって作成されたプロシージャー、または CREATE FUNCTION ステートメントによって作成された関数の、各パラメーターごとに行が 1 つずつ入ります。 スカラー関数の結果および表関数の結果列も返されます。

次の表は、SYSPARMS 表の列について説明しています。

表 *182. SYSPARMS* 表

列名	システム列名	データ・タイプ	説明
SPECIFIC_SCHEMA	SPECSCHEMA	VARCHAR(128)	ルーチン (関数) インスタンスのスキーマ名。
SPECIFIC_NAME	SPECNAME	VARCHAR(128)	ルーチン・インスタンスの特定名。
ORDINAL_POSITION	PARMNO	INTEGER	パラメーター・リスト内の該当のパラメーターの位置を示す数値で、左から右への順に、1 (左端のパラメーター) から *n* (*n* 番目のパラメーター) までです。 スカラー関数の場合、関数の結果は *n*+1 の値になります。 表関数の場合は、結果列には *n*+1 (左端の結果列) から *n*+*m* (*m* 番目の結果列) までの番号が付きます。
PARAMETER_MODE	PARMMODE	VARCHAR(5)	パラメーターのタイプ: **IN** これは入力パラメーターです。 **OUT** これは出力パラメーターです。 **INOUT** これは入出力パラメーターです。
PARAMETER_NAME	PARMNAME	VARCHAR(128) NULL 可能	パラメーターの名前。 パラメーターに名前がない場合は、NULL 値が入ります。

表 *182. SYSPARMS* 表 *(*続き*)*

列名	システム列名	データ・タイプ	説明
DATA_TYPE	DATA_TYPE	VARCHAR(128)	パラメーターのタイプ: **BIGINT** 大整数 **INTEGER** 長整数 **SMALLINT** 短整数 **DECIMAL** パック 10 進数 **NUMERIC** ゾーン 10 進数 **DOUBLE PRECISION** 浮動小数点数; DOUBLE PRECISION **REAL** 浮動小数点数; REAL **DECFLOAT** 10 進浮動小数点数 **CHARACTER** 固定長文字ストリング **CHARACTER VARYING** 可変長文字ストリング **CHARACTER LARGE OBJECT** 文字ラージ・オブジェクト・ストリング **GRAPHIC** 固定長グラフィック・ストリング **GRAPHIC VARYING** 可変長グラフィック・ストリング **DOUBLE-BYTE CHARACTER LARGE OBJECT** 2 バイト文字ラージ・オブジェクト・ストリング **BINARY** 固定長バイナリー・ストリング **BINARY VARYING** 可変長バイナリー・ストリング **BINARY LARGE OBJECT** バイナリー・ラージ・オブジェクト・ストリング **DATE** 日付 **TIME** 時刻 **TIMESTAMP** タイム・スタンプ **DATALINK** データ・リンク **ROWID** 行 ID **XML** XML **DISTINCT** 特殊タイプ **ARRAY** 配列タイプ

表 *182. SYSPARMS* 表 (続き)

列名	システム列名	データ・タイプ	説明
NUMERIC_SCALE	SCALE	INTEGER NULL 可能	数値データの位取り。 パラメーターが 10 進数、数値、または 2 進数でない場合は、NULL 値が入ります。
NUMERIC_PRECISION	PRECISION	INTEGER NULL 可能	数値パラメーターすべての精度。 **注:** この列では、すべての数値データ・タイプ (10 進浮動小数点数、単精度および倍精度の浮動小数点数を含む) の精度を指定します。NUMERIC_PRECISION_RADIX 列は、この列の値が 2 進数であるか、または 10 進数であるかを示します。 パラメーターが数値パラメーターでない場合は、NULL 値が入ります。
CCSID	CCSID	INTEGER NULL 可能	CHAR、VARCHAR、CLOB、DATE、TIME、TIMESTAMP、GRAPHIC、VARGRAPHIC、DBCLOB、および DATALINK パラメーターの CCSID の値。 CCSID が 0 であるということは、実行時にジョブの CCSID が使用されることを示しています。 XML パラメーターは、QAQQINI ファイルの SQL_XML_DATA_CCSID の値を使用します。 パラメーターが数値パラメーターの場合は、NULL 値が入ります。
CHARACTER_MAXIMUM_LENGTH	CHARLEN	INTEGER NULL 可能	データ・タイプが 2 進ストリング、文字ストリング、グラフィック・ストリング、および XML の場合は、ストリングの最大長。 パラメーターがストリングでない場合は、NULL 値が入ります。
CHARACTER_OCTET_LENGTH	CHARBYTE	INTEGER NULL 可能	データ・タイプが 2 進ストリング、文字ストリング、グラフィック・ストリング、および XML の場合は、バイト数。 パラメーターがストリングでない場合は、NULL 値が入ります。
NUMERIC_PRECISION_RADIX	RADIX	INTEGER NULL 可能	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 パラメーターが数値パラメーターでない場合は、NULL 値が入ります。
DATETIME_PRECISION	DATPRC	INTEGER NULL 可能	日付、時刻、またはタイム・スタンプの小数部分。 **0** データ・タイプが DATE および TIME の場合 **0-12** データ・タイプが TIMESTAMP の場合 (小数秒) パラメーター日付、時刻、またはタイム・スタンプのパラメーターでない場合は、NULL 値が入ります。

表 182. SYSPARMS 表 (続き)

列名	システム列名	データ・タイプ	説明
IS_NULLABLE	NULLS	VARCHAR(3)	パラメーターがNULL 可能かどうかを示します。 **NO** パラメーターに NULL は許されません。 **YES** パラメーターに NULL が許されます。
LONG_COMMENT	REMARKS	VARGRAPHIC(2000) CCSID 1200 NULL 可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、NULL 値が入ります。
ROW_TYPE	ROWTYPE	CHAR(1) NULL 可能	行のタイプを識別します。 **P** パラメーター。 **R** この関数が表関数の場合は、結果列を示します。それ以外の場合は、キャスト前の結果です。 **C** キャスト後の結果。
DATA_TYPE_SCHEMA	TYPESCHEMA	VARCHAR(128) NULL 可能	特殊タイプの場合は、データ・タイプのスキーマ。 パラメーターが特殊タイプでない場合は、NULL 値が入ります。
DATA_TYPE_NAME	TYPENAME	VARCHAR(128) NULL 可能	特殊タイプの場合は、データ・タイプの名前。 パラメーターが特殊タイプでない場合は、NULL 値が入ります。
AS_LOCATOR	ASLOCATOR	VARCHAR(3)	パラメーターがロケーターとして指定されたかどうかを識別します。 **NO** パラメーターはロケーターとして指定されませんでした。 **YES** パラメーターはロケーターとして指定されました。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。
NORMALIZE_DATA	NORMALIZE	VARCHAR(3) NULL 可能	パラメーター値を正規化するかどうかを示します。この属性は UTF-8 および UTF-16 のデータのみに適用されます。 **NO** 値は正規化されません。 **YES** 値は正規化されます。
DEFAULT	DEFAULT	DBCLOB(64K) CCSID 1200 NULL 可能	パラメーターのデフォルト値が存在する場合、パラメーターのデフォルト値を計算するために使用される式ストリング。デフォルト値がヌル値の場合、式ストリングはキーワード NULL です。パラメーターにデフォルトがない場合は、NULL 値が入ります。

SYSPARTITIONDISK

SYSPARTITIONDISK ビューには、各表パーティションまたは表メンバーのデータを保管するために使用されるディスク装置ごとに 1 つの行が含まれます。 表が分散表の場合は、他のデータベース・ノードにあるパーティションは、このカタログ・ビューには含まれません。これらは該当の他のデータベース・ノードのカタログ・ビューに含まれます。

次の表は、SYSPARTITIONDISK ビューの列について説明しています。

表 *183. SYSPARTITIONDISK* ビュー

列名	システム列名	データ・タイプ	説明
TABLE_SCHEMA	TABSCHEMA	VARCHAR(128)	該当の表が入っている SQL のスキーマの名前。
TABLE_NAME	TABNAME	VARCHAR(128)	表の名前。
TABLE_PARTITION	TABPART	VARCHAR(128)	表パーティションまたはメンバーの名前。
ASP_NUMBER	ASP_NUMBER	SMALLINT	パーティションを含んでいる補助記憶域プール (ASP)。
DISK_TYPE	DISK_TYPE	VARCHAR(4)	ディスクのディスク・タイプ番号。
DISK_MODEL	DISK_MODEL	VARCHAR(4)	ディスクのモデル番号。
UNIT_NUMBER	UNITNBR	SMALLINT	ディスクの装置番号。
LOGICAL_MIRRORED_PAIR_STATUS	MIRRORPS	CHAR(1) NULL 可能	ミラー保護されたディスクのペアの状況を示します。 **0** ミラー保護されたペアの片方の装置が活動状態ではないことを示します。 **1** ミラー保護されたペアの両方の装置が活動状態であることを示します。 装置がミラー保護されていない場合は、NULL が入ります。
MIRRORED_UNIT_STATUS	MIRRORUS	CHAR(1) NULL 可能	ミラー保護された装置の状況を示します。 **1** ミラー保護されたペアのこのミラー保護された装置は活動状態である (現行データでオンラインである) ことを示します。 **2** このミラー保護された装置は同期化中であることを示します。 **3** このミラー保護された装置は中断状態であることを示します。 装置がミラー保護されていない場合は、NULL が入ります。
UNIT_MEDIA_CAPACITY	UNITMCAP	BIGINT	装置のストレージ容量 (バイト単位)。
UNIT_SPACE_AVAILABLE	UNITSPACE	BIGINT	装置上の使用可能なスペース (バイト単位)。
UNIT_SPACE_RESERVED_FOR_SYSTEM	UNITSRES	BIGINT	システムでの使用のために予約済みの装置上のスペース (バイト単位)。
UNIT_SPACE_USED	UNITSUSED	BIGINT	パーティション用に使用されている装置上のスペース (バイト単位)。
UNIT_TOTAL_ACCESS_TIME	UNITTATIME	INTEGER	パーティション用の装置にある固定長データを順次読み取りするのに必要な見積もり時間 (ミリ秒単位)。この時間は、装置上のデータ量と装置の平均アクセス時間に基づきます。見積もりでは、ディスクには他のスレッドとの競合がないと想定されています。

表 *183. SYSPARTITIONDISK* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
UNIT_TYPE	UNIT_TYPE	SMALLINT	ディスク装置のタイプを示します。 **0** ソリッド・ステート・ディスク (SSD) ではありません。 **1** ソリッド・ステート・ディスク (SSD)。
DATA_SEGMENT_TYPE	SEGMENT	SMALLINT	ディスクのセグメント・タイプ。 **0** このセグメントが固定長データ用であることを示します。 **1** このセグメントが可変長データ用であることを示します。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	システムのスキーマ名。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	システム表名。
SYSTEM_TABLE_MEMBER	SYS_MNAME	CHAR(10)	システム・メンバー名。

SYSPARTITIONINDEXDISK

SYSPARTITIONINDEXDISK ビューには、各表パーティションまたは表メンバーの索引データを保管するために使用されるディスク装置ごとに 1 つの行が含まれます。索引が分散索引の場合は、他のデータベース・ノードにあるパーティションは、このカタログ・ビューには含まれません。これらは該当の他のデータベース・ノードのカタログ・ビューに含まれます。

次の表は、SYSPARTITIONINDEXDISK ビューの列について説明しています。

表 *184. SYSPARTITIONINDEXDISK* ビュー

列名	システム列名	データ・タイプ	説明
TABLE_SCHEMA	TABSCHEMA	VARCHAR(128)	該当の表が入っている SQL のスキーマの名前。
TABLE_NAME	TABNAME	VARCHAR(128)	表の名前。
TABLE_PARTITION	TABPART	VARCHAR(128)	表パーティションまたはメンバーの名前。
PARTITION_TYPE	PARTTYPE	CHAR(1)	表パーティショニングのタイプ ブランク 表はパーティション化されません。 **H** これはデータ・ハッシュ・パーティションです。 **R** これはデータ範囲パーティションです。 **D** これは分散データベース・ハッシュ・パーティションです。
PARTITION_NUMBER	PARTNBR	INTEGER NULL 可能	このパーティションのパーティション番号。表が分散表の場合には NULL が入ります。
NUMBER_DISTRIBUTED_PARTITIONS	DSTPARTS	INTEGER NULL 可能	表が分散表の場合にはパーティションの合計数が入ります。表が分散表でない場合には NULL が入ります。
INDEX_SCHEMA	INDSCHEMA	VARCHAR(128)	索引、論理ファイル、または制約を含む SQL スキーマの名前。
INDEX_NAME	INDNAME	VARCHAR(128)	索引、論理ファイル、または制約の名前。
INDEX_MEMBER	INDMEMBER	VARCHAR(128) NULL 可能	索引または論理ファイルのメンバーの名前。索引のタイプが制約である場合、メンバー名は NULL です。
INDEX_TYPE	INDTYPE	VARCHAR(11)	索引のタイプ: **INDEX** 索引は SQL 索引です。 **LOGICAL** 索引は論理ファイルの一部です。 **PHYSICAL** 索引はキー付き物理ファイルの一部です。 **PRIMARY KEY** 索引は基本キー制約です。 **UNIQUE** 索引はユニーク制約です。 **REFERENTIAL** 索引は外部キー制約です。

表 184. SYSPARTITIONINDEXDISK ビュー (続き)

列名	システム列名	データ・タイプ	説明
PARTITIONED	PARTITION	CHAR(1)	索引がパーティション化されているか、パーティション化されていないかを示します。 **0** SQL 索引はパーティション化されていません (複数のパーティションにまたがっています)。 **1** パーティション表で索引が作成されていないか、パーティション表で索引が作成されていて、パーティション化されているかのどちらかです (複数のパーティションまたはメンバーにまたがっていません)。 **2** 索引は、複数のパーティションまたはメンバー上に構築された論理ファイルです。
ASP_NUMBER	ASP_NUMBER	SMALLINT	索引を含んでいる補助記憶域プール (ASP)。
DISK_TYPE	DISK_TYPE	VARCHAR(4)	ディスクのディスク・タイプ番号。
DISK_MODEL	DISK_MODEL	VARCHAR(4)	ディスクのモデル番号。
UNIT_NUMBER	UNITNBR	SMALLINT	ディスクの装置番号。
LOGICAL_MIRRORED_PAIR_STATUS	MIRRORPS	CHAR(1) NULL 可能	ミラー保護されたディスクのペアの状況を示します。 **0** ミラー保護されたペアの片方の装置が活動状態ではないことを示します。 **1** ミラー保護されたペアの両方の装置が活動状態であることを示します。 装置がミラー保護されていない場合は、NULL が入ります。
MIRRORED_UNIT_STATUS	MIRRORUS	CHAR(1) NULL 可能	ミラー保護された装置の状況を示します。 **1** ミラー保護されたペアのこのミラー保護された装置は活動状態である (現行データでオンラインである) ことを示します。 **2** このミラー保護された装置は同期化中であることを示します。 **3** このミラー保護された装置は中断状態であることを示します。 装置がミラー保護されていない場合は、NULL が入ります。
UNIT_MEDIA_CAPACITY	UNITMCAP	BIGINT	装置のストレージ容量 (バイト単位)。
UNIT_SPACE_AVAILABLE	UNITSPACE	BIGINT	装置上の使用可能なスペース (バイト単位)。
UNIT_SPACE_RESERVED_FOR_SYSTEM	UNITSRES	BIGINT	システムでの使用のために予約済みの装置上のスペース (バイト単位)。
UNIT_SPACE_USED	UNITSUSED	BIGINT	索引用に使用されている装置上のスペース (バイト単位)。
UNIT_TYPE	UNIT_TYPE	SMALLINT	ディスク装置のタイプを示します。 **0** ソリッド・ステート・ディスク (SSD) ではありません。 **1** ソリッド・ステート・ディスク (SSD)。
SYSTEM_INDEX_SCHEMA	SYS_IXDNAME	CHAR(10)	システム索引スキーマ名。
SYSTEM_INDEX_NAME	SYS_IXNAME	CHAR(10)	システム索引名。

表 *184. SYSPARTITIONINDEXDISK* ビュー (続き)

列名	システム列名	データ・タイプ	説明
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	システムのスキーマ名。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	システム表名。
SYSTEM_TABLE_MEMBER	SYS_MNAME	CHAR(10)	システム・メンバー名。

SYSPARTITIONINDEXES

SYSPARTITIONINDEXES ビューには、表パーティションまたは表メンバーをもとに作成された各索引ごとに、行が 1 つずつ入ります。表が分散表の場合は、他のデータベース・ノードにあるパーティションに基づく索引はこのカタログ・ビューには含まれません。これらは該当の他のデータベース・ノードのカタログ・ビューに含まれます。

このビューを使用すると、指定した特定の表または一連の表に基づき作成された索引についての索引情報を表示することができます。この情報は、System i ナビゲーターで「索引の表示」を選択したときに返される情報と同様のものです。

次の表は、SYSPARTITIONINDEXES ビューの列について説明しています。

表 185. SYSPARTITIONINDEXES ビュー

列名	システム列名	データ・タイプ	説明
TABLE_SCHEMA	TABSCHEMA	VARCHAR(128)	該当の表が入っている SQL のスキーマの名前。
TABLE_NAME	TABNAME	VARCHAR(128)	表の名前。
TABLE_PARTITION	TABPART	VARCHAR(128)	表パーティションまたはメンバーの名前。
INDEX_NAME	INDNAME	VARCHAR(128)	索引、論理ファイル、または制約の名前。索引のタイプが 1 つ以上の一時索引を示す場合、INDEX_NAME には、表に現在存在している保守されている一時索引数の後に、ストリング「MAINTAINED TEMPORARY INDEXES」が続いたものが含まれます。
INDEX_TYPE	INDTYPE	VARCHAR(11)	索引のタイプ: **INDEX** 索引は SQL 索引です。 **LOGICAL** 索引は論理ファイルの一部です。 **PHYSICAL** 索引はキー付き物理ファイルの一部です。 **PRIMARY KEY** 索引は基本キー制約です。 **UNIQUE** 索引はユニーク制約です。 **FOREIGN KEY** 索引は外部キー制約です。 **TEMPORARY** 1 つ以上の一時索引が表に存在することを示します。
INDEX_SCHEMA	INDSCHEMA	VARCHAR(128) NULL 可能	索引、論理ファイル、または制約を含む SQL スキーマの名前。 1 つ以上の保守されている一時索引を行が示している場合は、NULL が入ります。
INDEX_OWNER	INDOWNER	VARCHAR(128) NULL 可能	索引の所有者。1 つ以上の保守されている一時索引を行が示している場合は、NULL が入ります。
SYSTEM_INDEX_SCHEMA	SYS_IXDNAM	CHAR(10) NULL 可能	システム索引スキーマ名。索引タイプが INDEX または LOGICAL である場合を除き、NULL が入ります。
SYSTEM_INDEX_NAME	SYS_IXNAME	CHAR(10) NULL 可能	システム索引名。索引タイプが INDEX または LOGICAL である場合を除き、NULL が入ります。

表 *185. SYSPARTITIONINDEXES* ビュー (続き)

列名	システム列名	データ・タイプ	説明
INDEX_TEXT	LABEL	VARGRAPHIC(50) CCSID 1200 NULL 可能	索引、論理ファイル、または制約のテキスト。索引にテキストが存在しない場合は、NULL が入ります。
INDEX_PARTITION	INDMEMBER	VARCHAR(128) NULL 可能	索引のパーティション名またはメンバー名。1 つ以上の保守されている一時索引を行が示している場合は、NULL が入ります。
INDEX_VALID	VALID	VARCHAR(3)	索引が無効で、再作成が必要であるかどうかを示す指示。 **NO** 索引は無効です。 **YES** 索引は有効です。
CREATE_TIMESTAMP	CREATED	TIMESTAMP NULL 可能	索引が作成されたときのタイム・スタンプ。1 つ以上の保守されている一時索引を行が示している場合は、NULL が入ります。
LAST_BUILD_TIMESTAMP	LASTBUILD	TIMESTAMP NULL 可能	索引が最後に再作成されたときのタイム・スタンプ。1 つ以上の保守されている一時索引を行が示している場合は、NULL が入ります。
LAST_QUERY_USE	LASTQRYUSE	TIMESTAMP NULL 可能	使用状況統計が前回リセットされた後で、該当の索引が最後に照会で使用されたときのタイム・スタンプ。使用状況統計が前回リセットされた後でその索引が照会で一度も使用されていない場合、または 1 つ以上の保守されている一時索引を行が示す場合には、NULL が入ります。
LAST_STATISTICS_USE	LASTSTUSE	TIMESTAMP NULL 可能	使用状況統計が前回リセットされた後で、該当の索引が最後にオプティマイザーで統計用に使用されたときのタイム・スタンプ。使用状況統計が前回リセットされた後でその索引が統計用に一度も使用されていない場合、または 1 つ以上の保守されている一時索引を行が示す場合には、NULL が入ります。
QUERY_USE_COUNT	QRYUSECNT	BIGINT	使用状況統計が前回リセットされた後で、該当の索引が 1 つの照会の中で使用された回数。使用状況統計が前回リセットされた後でその索引が照会で一度も使用されていない場合には、0 が入ります。
QUERY_STATISTICS_COUNT	QRYSTCNT	BIGINT	使用状況統計が前回リセットされた後で、該当の索引がオプティマイザーで統計用に使用された回数。使用状況統計が前回リセットされた後でその索引が一度も統計用に使用されていない場合には、0 が入ります。
LAST_USED_TIMESTAMP		TIMESTAMP NULL 可能	アプリケーションによってネイティブ・レコード入出力操作または SQL 操作に該当の索引が最後に直接使用されたときのタイム・スタンプ。索引が一度も使用されていない場合、または 1 つ以上の保守されている一時索引を行が示す場合には、NULL が入ります。
DAYS_USED_COUNT	DAYSUSED	INTEGER	使用状況統計が前回リセットされた後で、アプリケーションによってネイティブ・レコード入出力操作または SQL 操作に該当の索引が直接使用された日数。使用状況統計が前回リセットされた後でその索引が一度も使用されていない場合には、0 が入ります。
LAST_RESET_TIMESTAMP	LASTRESET	TIMESTAMP NULL 可能	該当の索引について使用状況統計が前回リセットされたときのタイム・スタンプ。詳しくは、オブジェクト記述変更 (CHGOBJD) コマンドを参照してください。索引の最終使用時のタイム・スタンプが一度もリセットされていない場合は、NULL が入ります。

表 185. SYSPARTITIONINDEXES ビュー (続き)

列名	システム列名	データ・タイプ	説明
NUMBER_KEY_COLUMNS	INDKEYS	BIGINT NULL 可能	索引キーを定義する列の数。1 つ以上の保守されている一時索引を行が示している場合は、NULL が入ります。
COLUMN_NAMES	COLNAMES	VARCHAR(1024) NULL 可能	索引キーを定義する列名のコンマ区切りリスト。すべての列名の長さが 1024 を超える場合は、列値の最後に「...」が返されます。1 つ以上の保守されている一時索引を行が示している場合は、NULL が入ります。
NUMBER_KEYS	NUMRIDS	BIGINT NULL 可能	索引内のキーの数。索引が無効な場合、またはコード化ベクトル索引である場合は、-1 が返されます。1 つ以上の保守されている一時索引を行が示している場合は、NULL が入ります。
INDEX_SIZE	SIZE	BIGINT	索引の二分木またはコード化ベクトル索引のサイズ (バイト数)。
NUMBER_PAGES	PAGES	BIGINT NULL 可能	索引内のページの数。索引が無効な場合、またはコード化ベクトル索引である場合は、NULL が入ります。
LOGICAL_PAGE_SIZE	PAGE_SIZE	INTEGER NULL 可能	索引の論理ページ・サイズ。索引がコード化ベクトル索引である場合、または 1 つ以上の保守されている一時索引を行が示す場合には、NULL が入ります。
UNIQUE	UNIQUE	VARCHAR(21) NULL 可能	索引がユニークであるかどうかを示します。 **UNIQUE** 索引は UNIQUE 索引です。 **UNIQUE WHERE NOT NULL** 索引は UNIQUE WHERE NOT NULL 索引です。 **FIFO** 索引は非ユニーク先入れ先出し (FIFO) 索引です。 **LIFO** 索引は非ユニーク後入れ後出し (LIFO) 索引です。 **FCFO** 索引は非ユニーク先変更先出し (FCFO) 索引です。 1 つ以上の保守されている一時索引を行が示している場合は、NULL が入ります。
MAXIMUM_KEY_LENGTH	KEY_LENGTH	INTEGER NULL 可能	索引のキーの最大長。索引がコード化ベクトル索引である場合は、NULL が入ります。
UNIQUE_PARTIAL_KEY_VALUES	KEYCARDS	VARCHAR(96) NULL 可能	索引のユニーク部分キー値。コード化ベクトル索引の場合は、最初のユニーク部分キー値が索引キー全体の固有値の総数になります。返されるその他のユニーク部分キー値は、適用外です。1 つ以上の保守されている一時索引の場合は、NULL が入ります。
OVERFLOW_VALUES	OVERFLOW	INTEGER NULL 可能	コード化ベクトル索引からオーバーフローした特殊キー値の数。索引がコード化ベクトル索引でない場合は、NULL が入ります。
EVI_CODE_SIZE	CODE_SIZE	INTEGER NULL 可能	コード化ベクトル索引のバイト・コードのサイズ。索引がコード化ベクトル索引でない場合は、NULL が入ります。

表 *185. SYSPARTITIONINDEXES* ビュー (続き)

列名	システム列名	データ・タイプ	説明
SPARSE	SPARSE	VARCHAR(3) NULL 可能	索引に従属表のすべての行のキーが含まれるかどうかを示します。 **NO** 索引には、従属表のすべての行のキーが含まれます。 **YES** 索引は選択/除外論理ファイルであるか、WHERE 文節がある SQL 索引であり、従属表のすべての行のキーは含まれません。 1 つ以上の保守されている一時索引を行が示している場合は、NULL が入ります。
DERIVED_KEY	DERIVED	VARCHAR(3) NULL 可能	索引のキー列が式であるかどうかを示します。 **NO** 索引のキー列は、式ではありません。 **YES** 少なくとも 1 つのキー列が式です。 1 つ以上の保守されている一時索引を行が示している場合は、NULL が入ります。
PARTITIONED	PARTITION	VARCHAR(20) NULL 可能	索引がパーティション化されているか、パーティション化されていないかを示します。 **NO** SQL 索引はパーティション化されていません (複数のパーティションにまたがっています)。 **YES** パーティション表で索引が作成されていないか、パーティション表で索引が作成されていて、パーティション化されているかのどちらかです (複数のパーティションまたはメンバーにまたがっていません)。 **MULTI-MEMBER LOGICAL** 索引は、複数のパーティションまたはメンバー上に構築された論理ファイルです。 1 つ以上の保守されている一時索引を行が示している場合は、NULL が入ります。
ACCPTH_TYPE	ACCPTHTYPE	VARCHAR(4) NULL 可能	索引のタイプを示します。 **1 TB** 索引は、最大 1 テラバイト (*MAX1TB) の 2 進基数索引です。 **4 GB** 索引は、最大 4 ギガバイト (*MAX4GB) の 2 進基数索引です。 **EVI** 索引は、コード化ベクトル索引です。 1 つ以上の保守されている一時索引を行が示している場合は、NULL が入ります。

表 185. SYSPARTITIONINDEXES ビュー (続き)

列名	システム列名	データ・タイプ	説明
SORT_SEQUENCE	SRTSEQ	VARCHAR(12) NULL 可能	索引で照合順序を使用するかどうかを示します。 **BY HEX VALUE** 索引では照合テーブルを使用しません。 ***LANGIDSHR** 索引では共用重みソート順序 (SRTSEQ) を使用します。 ***LANGIDUNQ** 索引では固有重みソート順序 (SRTSEQ) を使用します。 **ALTSEQ** 索引では代替照合順序 (ALTSEQ) を使用します。 1 つ以上の保守されている一時索引を行が示している場合は、NULL が入ります。
LANGUAGE_IDENTIFIER	LANGID	CHAR(3) NULL 可能	索引の言語 ID。ソート順序が 16 進数である場合、または 1 つ以上の保守されている一時索引を行が示している場合は、NULL が入ります。
SORT_SEQUENCE_SCHEMA	SRTSEQSCH	CHAR(10) NULL 可能	使用するソート順序のスキーマ名。スキーマ名がない場合は、NULL が入ります。
SORT_SEQUENCE_NAME	SRTSEQNAME	CHAR(10) NULL 可能	使用するソート順序の名前。ソート順序の名前がない場合は、NULL が入ります。
ESTIMATED_BUILD_TIME	ESTBLDTIME	INTEGER NULL 可能	索引の再作成に必要な概算時間 (秒)。1 つ以上の保守されている一時索引を行が示している場合は、NULL が入ります。
LAST_BUILD_TIME	LSTBLDTIME	INTEGER NULL 可能	索引が最後に作成されたときからの経過時間 (秒単位)。最新の作成情報が使用できない場合には、NULL が含まれます。
LAST_BUILD_KEYS	LSTBLDKEYS	BIGINT NULL 可能	索引が最後に作成されたときのキーの数。最新の作成情報が使用できない場合には、NULL が含まれます。
LAST_BUILD_DEGREE	LSTBLDDEG	SMALLINT NULL 可能	索引が最後に作成されたときの並列度。最新の作成情報が使用できない場合には、NULL が含まれます。
LAST_BUILD_TYPE	LSTBLDTYPE	CHAR(1) NULL 可能	最後の索引作成が、全体作成であったか、遅延保守キーによる作成であったかを示します。 **0** 索引の最後の再作成は、遅延保守キーによるものでした。 **1** 索引の最後の作成または再作成は、表内の行による全体作成でした。 その索引が一度も作成されていない場合には、NULL が入ります。
LAST_INVALIDATION_TIMESTAMP	LSTINVAL	TIMESTAMP NULL 可能	索引が最後にいつ無効化されたかを示します。その索引が一度も無効化されていない場合には、NULL が入ります。
INDEX_HELD	HELD	VARCHAR(3)	索引の再作成保留について、ユーザーが現在それを保留しているかどうかを示します。 **NO** 索引の再作成は保留中でも、保留されてもいません。 **YES** 索引の再作成保留はユーザーが保留しています。

表 *185. SYSPARTITIONINDEXES* ビュー (続き)

列名	システム列名	データ・タイプ	説明
MAINTENANCE	MAINT	VARCHAR(11) NULL 可能	索引の保守: **REBUILD** 索引は保守されておらず、オープン時に再作成されます。 **DELAYED** 索引の保守は、索引のオープン時まで延期されます。 **DO NOT WAIT** 索引はすぐに保守されます。 索引がコード化ベクトル索引である場合、または 1 つ以上の保守されている一時索引を行が示す場合には、NULL が入ります。
DELAYED_MAINT_KEYS	DLYKEYS	INTEGER NULL 可能	遅延保守索引の二分木に挿入する必要のあるキーの数。索引が遅延保守索引でない場合は、NULL が入ります。
RECOVERY	RECOVERY	VARCHAR(10) NULL 可能	索引のリカバリー属性: **DURING IPL** 必要であれば、IPL 時に索引がリカバリーされます。 **AFTER IPL** 必要であれば、IPL 後に索引がリカバリーされます。 **NEXT OPEN** 必要であれば、次回のオープン時に索引がリカバリーされます。 索引がコード化ベクトル索引である場合、または 1 つ以上の保守されている一時索引を行が示す場合には、NULL が入ります。
ROUNDING_MODE	DECFLTRND	VARCHAR(8) NULL 可能	索引の DECFLOAT 丸めモードを示します。 **CEILING** ROUND_CEILING **DOWN** ROUND_DOWN **FLOOR** ROUND_FLOOR **HALFDOWN** ROUND_HALF_DOWN **HALFEVEN** ROUND_HALF_EVEN **HALFUP** ROUND_HALF_UP **UP** ROUND_UP DECFLOAT 列、関数、または定数を参照する式が索引にない場合、または 1 つ以上の保守されている一時索引を行が示している場合は、NULL 値が入ります。

表 185. SYSPARTITIONINDEXES ビュー (続き)

列名	システム列名	データ・タイプ	説明
DECFLOAT_WARNING	DECFLTWRN	VARCHAR(3) NULL 可能	DECFLOAT 警告が返されるかどうかを指定します。 **NO** DECFLOAT 警告は返されません。 **YES** DECFLOAT 警告が返されます。 DECFLOAT 列、関数、または定数を参照する式が索引にない場合、または 1 つ以上の保守されている一時索引を行が示している場合は、NULL 値が入ります。
LOGICAL_READS	LGLREADS	BIGINT NULL 可能	最後の IPL の後で索引に対して実行された論理読み取り操作の数。1 つ以上の保守されている一時索引を行が示している場合は、NULL が入ります。
SEQUENTIAL_READS	SEQREADS	BIGINT	最後の IPL の後で索引に対して実行された順次読み取り操作の数。
RANDOM_READS	RANREADS	BIGINT	最後の IPL の後で索引に対して実行されたランダム読み取り操作の数。
SEARCH_CONDITION	IXWHERECON	VARGRAPHIC(1024) CCSID 1200 NULL 可能	疎索引の場合はその索引の検索条件を示します。検索条件の長さが 1024 を超えた場合は、列値の最後に「...」が返されます。疎索引でない場合は、NULL が入ります。
SEARCH_CONDITION_HAS_UDF	IXWHEREUDF	VARCHAR(3) NULL 可能	疎索引の場合は、索引の検索条件にユーザー定義関数が含まれるかどうかを示します。疎索引でない場合は、NULL が入ります。 **NO** 索引検索条件には UDF は含まれません。 **YES** 索引検索条件に UDF が含まれます。
KEEP_IN_MEMORY	KEEPINMEM	VARCHAR(3)	索引をメモリー内に保持するかどうかを示します。 **NO** メモリー設定がありません。 **YES** 可能な場合、索引をメモリーに保持します。
MEDIA_PREFERENCE	MEDIAPREF	VARCHAR(3)	索引のメディア設定を示します。 **ANY** メディア設定がありません。 **SSD** 可能な場合、索引を ソリッド・ステート・ディスク (SSD) に割り振ります。
INCLUDE_EXPRESSION	IXINCEXPR	VARGRAPHIC(1024) CCSID 1200 NULL 可能	索引の INCLUDE 式。索引に INCLUDE 式がない場合は、NULL が入ります。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	システムのスキーマ名。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	システム表名。
SYSTEM_TABLE_MEMBER	SYS_MNAME	CHAR(10)	システム・メンバー名。

SYSPARTITIONINDEXSTAT

SYSPARTITIONINDEXSTAT ビューには、表パーティションまたは表メンバーをもとに作成された各索引ごとに、行が 1 つずつ入ります。別の索引の二分木を共有している索引は含まれません。 表が分散表の場合は、他のデータベース・ノードにあるパーティションに基づく索引はこのカタログ・ビューには含まれません。これらは該当の他のデータベース・ノードのカタログ・ビューに含まれます。

次の表は、SYSPARTITIONINDEXSTAT ビューの列について説明しています。

表 *186. SYSPARTITIONINDEXSTAT* ビュー

列名	システム列名	データ・タイプ	説明
TABLE_SCHEMA	TABSCHEMA	VARCHAR(128)	該当の表が入っている SQL のスキーマの名前。
TABLE_NAME	TABNAME	VARCHAR(128)	表の名前。
TABLE_PARTITION	TABPART	VARCHAR(128)	表パーティションまたはメンバーの名前。
PARTITION_TYPE	PARTTYPE	CHAR(1)	表パーティショニングのタイプ ブランク 表はパーティション化されません。 **H** これはデータ・ハッシュ・パーティションです。 **R** これはデータ範囲パーティションです。 **D** これは分散データベース・ハッシュ・パーティションです。
PARTITION_NUMBER	PARTNBR	INTEGER NULL 可能	このパーティションのパーティション番号。表が分散表の場合には NULL が入ります。
NUMBER_DISTRIBUTED_PARTITIONS	DSTPARTS	INTEGER NULL 可能	表が分散表の場合にはパーティションの合計数が入ります。表が分散表でない場合には NULL が入ります。
INDEX_SCHEMA	INDSCHEMA	VARCHAR(128)	索引、論理ファイル、または制約を含む SQL スキーマの名前。
INDEX_NAME	INDNAME	VARCHAR(128)	索引、論理ファイル、または制約の名前。
INDEX_MEMBER	INDMEMBER	VARCHAR(128) NULL 可能	索引または論理ファイルのメンバーの名前。索引のタイプが制約である場合、メンバー名は NULL です。
INDEX_TYPE	INDTYPE	VARCHAR(11)	索引のタイプ: **INDEX** 索引は SQL 索引です。 **LOGICAL** 索引は論理ファイルの一部です。 **PHYSICAL** 索引はキー付き物理ファイルの一部です。 **PRIMARY KEY** 索引は基本キー制約です。 **UNIQUE** 索引はユニーク制約です。 **REFERENTIAL** 索引は外部キー制約です。
NUMBER_KEY_COLUMNS	INDKEYS	BIGINT	索引キーを定義する列の数。
COLUMN_NAMES	COLNAMES	VARCHAR(1024)	索引キーを定義する列名のコンマ区切りリスト。すべての列名の長さが 1024 を超える場合は、列値の最後に「...」が返されます。
NUMBER_LEAF_PAGES	NLEAF	BIGINT	Db2 for i には適用されません。常に -1 になります。

表 186. SYSPARTITIONINDEXSTAT ビュー (続き)

列名	システム列名	データ・タイプ	説明
NUMBER_LEVELS	NLEVELS	SMALLINT	Db2 for i には適用されません。常に -1 になります。
FIRSTKEYCARD	KEYCARD1	BIGINT	最初のキーの値の種類数。コード化ベクトル索引の場合は、索引キー全体の固有値の総数です。
FIRST2KEYCARD	KEYCARD2	BIGINT	索引の最初の 2 つの列を使用するキーの種類数。索引がコード化ベクトル索引である場合は、-1 が返されます。
FIRST3KEYCARD	KEYCARD3	BIGINT	索引の最初の 3 つの列を使用するキーの種類数。索引がコード化ベクトル索引である場合は、-1 が返されます。
FIRST4KEYCARD	KEYCARD4	BIGINT	索引の最初の 4 つの列を使用するキーの種類数。索引がコード化ベクトル索引である場合は、-1 が返されます。
FULLKEYCARD	KEYCARDF	BIGINT	特殊なフル・キー値の数。索引に 4 つを超えるキー列がある場合、または索引がコード化ベクトル索引である場合は、-1 が返されます。
CLUSTERRATIO	CLSRATIO	SMALLINT	Db2 for i には適用されません。常に -1 になります。
CLUSTERFACTOR	CLSFACTOR	DOUBLE	Db2 for i には適用されません。常に -1 になります。
SEQUENTIAL_PAGES	SEQPAGES	BIGINT	Db2 for i には適用されません。常に -1 になります。
DENSITY	DENSITY	INTEGER	Db2 for i には適用されません。常に -1 になります。
PAGE_FETCH_PAIRS	FETCHPAIRS	VARCHAR(520)	Db2 for i には適用されません。常に空ストリングになります。
NUMBER_KEYS	NUMRIDS	BIGINT	索引内のキーの数。索引が無効な場合、またはコード化ベクトル索引である場合は、-1 が返されます。
NUMRIDS_DELETED	NUMRIDSDLT	BIGINT	Db2 for i には適用されません。常に 0 になります。
NUM_EMPTY_LEAFS	EMPTYLEAFS	BIGINT	Db2 for i には適用されません。常に 0 になります。
AVERAGE_RANDOM_FETCH_PAGES	AVGRNDFTCH	DOUBLE	Db2 for i には適用されません。常に -1 になります。
AVERAGE_RANDOM_PAGES	AVGRNDPAGE	DOUBLE	Db2 for i には適用されません。常に -1 になります。
AVERAGE_SEQUENCE_GAP	AVGSEQGAP	DOUBLE	Db2 for i には適用されません。常に -1 になります。
AVERAGE_SEQUENCE_FETCH_GAP	AVGSEQFGAP	DOUBLE	Db2 for i には適用されません。常に -1 になります。
AVERAGE_SEQUENCE_PAGES	AVGSEQPAGE	DOUBLE	Db2 for i には適用されません。常に -1 になります。
AVERAGE_SEQUENCE_FETCH_PAGES	AVGSEQFPAG	DOUBLE	Db2 for i には適用されません。常に -1 になります。
AVGPARTITION_CLUSTERRATIO	PCLSRATIO	SMALLINT	Db2 for i には適用されません。常に -1 になります。
AVGPARTITION_CLUSTERFACTOR	PCLSFACTOR	DOUBLE	Db2 for i には適用されません。常に -1 になります。
AVGPARTITION_PAGE_FETCH_PAIRS	PFETCHPAIR	VARCHAR(520)	Db2 for i には適用されません。常に空ストリングになります。
DATAPARTITION_CLUSTERFACTOR	DCLSFACTOR	DOUBLE	データ・パーティションに関しての索引キーの「クラスタリング」を測定する統計。これは 0 から 1 の間の数値で、1 は完全なクラスタリングを表し、0 はクラスタリングがないことを表します。

表 *186. SYSPARTITIONINDEXSTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
INDCARD	INDCARD	BIGINT	索引内のキーの数。索引が無効な場合、またはコード化ベクトル索引である場合は、-1 が返されます。
INDEX_VALID	VALID	CHAR(1)	索引が無効で、再作成が必要であるかどうかを示す標識。 **0** 索引は無効です。 **1** 索引は有効です。 **2** 索引は、STG(*FREE) で保管されました。
INDEX_HELD	HELD	CHAR(1)	索引の再作成保留について、ユーザーが現在それを保留しているかどうかを示します。 **0** 索引の再作成は保留中でも、保留されてもいません。 **1** 索引の再作成保留はユーザーが保留しています。
CREATE_TIMESTAMP	CREATED	TIMESTAMP	索引が作成されたときのタイム・スタンプ。
LAST_BUILD_TIMESTAMP	LASTBUILD	TIMESTAMP	索引が最後に再作成されたときのタイム・スタンプ。
LAST_QUERY_USE	LASTQRYUSE	TIMESTAMP NULL 可能	使用状況統計が前回リセットされた後で、該当の索引が最後に照会で使用されたときのタイム・スタンプ。使用状況統計が前回リセットされた後でその索引が照会で一度も使用されていない場合には、NULL が入ります。
LAST_STATISTICS_USE	LASTSTUSE	TIMESTAMP NULL 可能	使用状況統計が前回リセットされた後で、該当の索引が最後にオプティマイザーで統計用に使用されたときのタイム・スタンプ。使用状況統計が前回リセットされた後でその索引が一度も統計用に使用されていない場合には、NULL が入ります。
QUERY_USE_COUNT	QRYUSECNT	BIGINT	使用状況統計が前回リセットされた後で、該当の索引が 1 つの照会の中で使用された回数。使用状況統計が前回リセットされた後でその索引が照会で一度も使用されていない場合には、0 が入ります。
QUERY_STATISTICS_COUNT	QRYSTCNT	BIGINT	使用状況統計が前回リセットされた後で、該当の索引がオプティマイザーで統計用に使用された回数。使用状況統計が前回リセットされた後でその索引が一度も統計用に使用されていない場合には、0 が入ります。
LAST_USED_TIMESTAMP	LASTUSED	TIMESTAMP NULL 可能	アプリケーションによってネイティブ・レコード入出力操作または SQL 操作に該当の索引が最後に直接使用されたときのタイム・スタンプ。その索引が一度も使用されていない場合には、NULL が入ります。
DAYS_USED_COUNT	DAYSUSED	INTEGER	使用状況統計が前回リセットされた後で、アプリケーションによってネイティブ・レコード入出力操作または SQL 操作に該当の索引が直接使用された日数。使用状況統計が前回リセットされた後でその索引が一度も使用されていない場合には、0 が入ります。
LAST_RESET_TIMESTAMP	LASTRESET	TIMESTAMP NULL 可能	該当の索引について使用状況統計が前回リセットされたときのタイム・スタンプ。詳しくは、オブジェクト記述変更 (CHGOBJD) コマンドを参照してください。索引の最終使用時のタイム・スタンプが一度もリセットされていない場合は、NULL が入ります。
INDEX_SIZE	SIZE	BIGINT	索引の二分木またはコード化ベクトル索引のサイズ (バイト数)。
ESTIMATED_BUILD_TIME	ESTBLDTIME	INTEGER	索引の再作成に必要な概算時間 (秒)。

表 186. SYSPARTITIONINDEXSTAT ビュー (続き)

列名	システム列名	データ・タイプ	説明
LAST_BUILD_TIME	LSTBLDTIME	INTEGER NULL 可能	索引が最後に作成されたときからの経過時間 (秒単位)。最新の作成情報が使用できない場合には、NULL が含まれます。
LAST_BUILD_KEYS	LSTBLDKEYS	BIGINT NULL 可能	索引が最後に作成されたときのキーの数。最新の作成情報が使用できない場合には、NULL が含まれます。
LAST_BUILD_DEGREE	LSTBLDDEG	SMALLINT NULL 可能	索引が最後に作成されたときの並列度。最新の作成情報が使用できない場合には、NULL が含まれます。
LAST_BUILD_TYPE	LSTBLDTYPE	CHAR(1) NULL 可能	最後の索引作成が、全体作成であったか、遅延保守キーによる作成であったかを示します。 **0** 索引の最後の再作成は、遅延保守キーによるものでした。 **1** 索引の最後の作成または再作成は、表内の行による全体作成でした。 その索引が一度も作成されていない場合には、NULL が入ります。
LAST_INVALIDATION_TIMESTAMP	LSTINVAL	TIMESTAMP NULL 可能	索引が最後にいつ無効化されたかを示します。その索引が一度も無効化されていない場合には、NULL が入ります。
DELAYED_MAINT_KEYS	DLYKEYS	INTEGER NULL 可能	遅延保守索引の二分木に挿入する必要のあるキーの数。索引が遅延保守索引でない場合は、NULL が入ります。
SPARSE	SPARSE	CHAR(1)	索引に従属表のすべての行のキーが含まれるかどうかを示します。 **0** 索引には、従属表のすべての行のキーが含まれます。 **1** 索引は選択/除外論理ファイルであるか、WHERE 文節がある SQL 索引であり、従属表のすべての行のキーは含まれません。
DERIVED_KEY	DERIVED	CHAR(1)	索引のキー列が式であるかどうかを示します。 **0** 索引のキー列は、式ではありません。 **1** 少なくとも 1 つのキー列が式です。現在、DDS で作成された論理ファイルまたは一時索引でのみ可能です。
PARTITIONED	PARTITION	CHAR(1)	索引がパーティション化されているか、パーティション化されていないかを示します。 **0** SQL 索引はパーティション化されていません (複数のパーティションにまたがっています)。 **1** パーティション表で索引が作成されていないか、パーティション表で索引が作成されていて、パーティション化されているかのどちらかです (複数のパーティションまたはメンバーにまたがっていません)。 **2** 索引は、複数のパーティションまたはメンバー上に構築された論理ファイルです。

表 *186. SYSPARTITIONINDEXSTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
ACCPTH_TYPE	ACCPTHTYPE	CHAR(1)	索引のタイプを示します。 **0** 索引は、最大 1 テラバイト (*MAX1TB) の 2 進基数索引です。 **1** 索引は、最大 4 ギガバイト (*MAX4GB) の 2 進基数索引です。 **2** 索引は、コード化ベクトル索引です。
UNIQUE	UNIQUE	CHAR(1)	索引がユニークであるかどうかを示します。 **0** 索引は UNIQUE 索引です。 **1** 索引は UNIQUE WHERE NOT NULL 索引です。 **2** 索引は非ユニーク先入れ先出し (FIFO) 索引です。 **3** 索引は非ユニーク後入れ後出し (LIFO) 索引です。 **4** 索引は非ユニーク先変更先出し (FCFO) 索引です。
SRTSEQ_TYPE	SRTSEQ	CHAR(1)	索引で照合順序を使用するかどうかを示します。 **0** 索引では照合テーブルを使用しません。 **1** 索引では代替照合順序 (ALTSEQ) を使用します。 **2** 索引ではソート順序 (SRTSEQ) を使用します。
LOGICAL_PAGE_SIZE	PAGE_SIZE	INTEGER NULL 可能	索引の論理ページ・サイズ。索引がコード化ベクトル索引である場合は、NULL が入ります。
OVERFLOW_VALUES	OVERFLOW	INTEGER NULL 可能	コード化ベクトル索引からオーバーフローした特殊キー値の数。索引がコード化ベクトル索引でない場合は、NULL が入ります。
EVI_CODE_SIZE	CODE_SIZE	INTEGER NULL 可能	コード化ベクトル索引のバイト・コードのサイズ。索引がコード化ベクトル索引でない場合は、NULL が入ります。
LOGICAL_READS	LGLREADS	BIGINT	最後の IPL の後で索引に対して実行された論理読み取り操作の数。
PHYSICAL_READS	PHYREADS	BIGINT	Db2 for i には適用されません。常に 0 になります。
SEQUENTIAL_READS	SEQREADS	BIGINT	最後の IPL の後で索引に対して実行された順次読み取り操作の数。
RANDOM_READS	RANREADS	BIGINT	最後の IPL の後で索引に対して実行されたランダム読み取り操作の数。
SEARCH_CONDITION	IXWHERECON	VARGRAPHIC(1024) CCSID 1200	疎索引の場合はその索引の検索条件を示します。検索条件の長さが 1024 を超えた場合は、列値の最後に「...」が返されます。
KEEP_IN_MEMORY	KEEPINMEM	CHAR(1)	索引をメモリー内に保持するかどうかを示します。 **0** メモリー設定がありません。 **1** 可能な場合、索引をメモリーに保持します。
MEDIA_PREFERENCE	MEDIAPREF	SMALLINT	索引のメディア設定を示します。 **0** メディア設定がありません。 **255** 可能な場合、索引を ソリッド・ステート・ディスク (SSD) に割り振ります。

表 *186. SYSPARTITIONINDEXSTAT* ビュー (続き)

列名	システム列名	データ・タイプ	説明
INCLUDE_EXPRESSION	IXINCEXPR	VARGRAPHIC(1024) CCSID 1200 NULL 可能	索引の INCLUDE 式。索引に INCLUDE 式がない場合は、NULL が入ります。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	システムのスキーマ名。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	システム表名。
SYSTEM_TABLE_MEMBER	SYS_MNAME	CHAR(10)	システム・メンバー名。

SYSPARTITIONMQTS

SYSPARTITIONMQTS ビューには、表パーティションまたは表メンバーをもとに作成された各マテリアライズ表ごとに、行が 1 つずつ入ります。表が分散表の場合は、他のデータベース・ノードにあるパーティションに基づくマテリアライズ表はこのカタログ・ビューには含まれません。これらは該当の他のデータベース・ノードのカタログ・ビューに含まれます。

このビューを使用すると、指定した特定の表または一連の表に基づき作成されたマテリアライズ表についてのマテリアライズ照会表情報を表示することができます。この情報は、System i ナビゲーターで「マテリアライズ照会表の表示」を選択したときに返される情報と同様のものです。

次の表は、SYSPARTITIONMQTS ビューの列について説明しています。

表 *187. SYSPARTITIONMQTS* ビュー

列名	システム列名	データ・タイプ	説明
TABLE_SCHEMA	TABSCHEMA	VARCHAR(128)	該当の表が入っている SQL のスキーマの名前。
TABLE_NAME	TABNAME	VARCHAR(128)	表の名前。
TABLE_PARTITION	TABPART	VARCHAR(128)	表パーティションまたはメンバーの名前。
MQT_NAME	MQTNAME	VARCHAR(128)	マテリアライズ照会表の名前。
MQT_SCHEMA	MQTSCHEMA	VARCHAR(128)	該当のマテリアライズ照会表が入っている SQL スキーマの名前。
MQT_PARTITION	MQTMEMBER	VARCHAR(128)	マテリアライズ照会表のパーティション名またはメンバー名。
MQT_OWNER	MQTOWNER	VARCHAR(128)	マテリアライズ照会表の所有者。
SYSTEM_MQT_SCHEMA	SYS_MQDNAM	CHAR(10)	システム・マテリアライズ照会表のスキーマ名。
SYSTEM_MQT_NAME	SYS_MQNAME	CHAR(10)	システム・マテリアライズ照会表名。
ENABLED	ENABLED	VARCHAR(3)	マテリアライズ照会表を使用可能にするかどうかを示します。 **NO** マテリアライズ照会表を使用可能にしません。 **YES** マテリアライズ照会表を使用可能にしてデータベース・マネージャーが使用できるようにします。
CREATE_TIMESTAMP	CREATED	TIMESTAMP	マテリアライズ照会表が作成されたときのタイム・スタンプ。
REFRESH_TIME	REFRESHDTS	TIMESTAMP NULL 可能	マテリアライズ照会表が最後にリフレッシュされたときのタイム・スタンプ。マテリアライズ照会表が一度もリフレッシュされていない場合は、NULL が入ります。
LAST_QUERY_USE	LASTQRYUSE	TIMESTAMP NULL 可能	使用状況統計が前回リセットされた後で、該当のマテリアライズ照会表が最後に照会で使用されたときのタイム・スタンプ。使用状況統計が前回リセットされた後でそのマテリアライズ照会表が照会で一度も使用されていない場合には、NULL が入ります。
LAST_STATISTICS_USE	LASTSTUSE	TIMESTAMP NULL 可能	使用状況統計が前回リセットされた後で、該当のマテリアライズ照会表が最後にオプティマイザーで統計用に使用されたときのタイム・スタンプ。使用状況統計が前回リセットされた後でそのマテリアライズ照会表が統計用に一度も使用されていない場合には、NULL が入ります。

表 *187. SYSPARTITIONMQTS* ビュー (続き)

列名	システム列名	データ・タイプ	説明
QUERY_USE_COUNT	QRYUSECNT	BIGINT	使用状況統計が前回リセットされた後で、該当のマテリアライズ照会表が 1 つの照会の中で使用された回数。使用状況統計が前回リセットされた後でそのマテリアライズ照会表が照会で一度も使用されていない場合には、0 が入ります。
QUERY_STATISTICS_COUNT	QRYSTCNT	BIGINT	使用状況統計が前回リセットされた後で、該当のマテリアライズ照会表がオプティマイザーで統計用に使用された回数。使用状況統計が前回リセットされた後でそのマテリアライズ照会表が統計用に一度も使用されていない場合には、0 が入ります。
LAST_USED_TIMESTAMP	LASTUSED	TIMESTAMP NULL 可能	アプリケーションによってネイティブ・レコード入出力操作または SQL 操作に該当のマテリアライズ照会表が最後に直接使用されたときのタイム・スタンプ。そのマテリアライズ照会表が一度も使用されていない場合には、NULL が入ります。
DAYS_USED_COUNT	DAYSUSED	INTEGER	使用状況統計が前回リセットされた後で、アプリケーションによってネイティブ・レコード入出力操作または SQL 操作に該当のマテリアライズ照会表が直接使用された日数。使用状況統計が前回リセットされた後でそのマテリアライズ照会表が一度も使用されていない場合には、0 が入ります。
LAST_RESET_TIMESTAMP	LASTRESET	TIMESTAMP NULL 可能	該当のマテリアライズ照会表について使用状況統計が前回リセットされたときのタイム・スタンプ。詳しくは、オブジェクト記述変更 (CHGOBJD) コマンドを参照してください。マテリアライズ照会表の最終使用時のタイム・スタンプが一度もリセットされていない場合は、NULL が入ります。
NUMBER_ROWS	CARD	BIGINT	マテリアライズ照会表の行の数。
MQT_SIZE	SIZE	BIGINT	マテリアライズ照会表のサイズ (バイト数)。
LAST_CHANGE_TIMESTAMP	LASTCHG	TIMESTAMP NULL 可能	マテリアライズ照会表が最後に変更されたときのタイム・スタンプ。使用状況統計が前回リセットされた後でそのマテリアライズ照会表が一度も変更されていない場合には、NULL が入ります。
MAINTENANCE	MAINTAIN	VARCHAR(6)	マテリアライズ照会表の保守方法を示します。 **SYSTEM** マテリアライズ照会表はシステムが保守します。 **USER** マテリアライズ照会表はユーザーが保守します。
INITIAL_DATA	INITIAL	VARCHAR(19)	マテリアライズ照会表の初期データを示します。 **INITIALLY DEFERRED** マテリアライズ照会表の作成時に、そこにデータを挿入しません。 **INITIALLY IMMEDIATE** マテリアライズ照会表の作成時に、そこにデータを挿入します。
REFRESH	REFRESH	VARCHAR(9)	マテリアライズ照会表内のデータをリフレッシュできる時期を示します。 **DEFERRED** マテリアライズ照会表内のデータは、REFRESH TABLE ステートメントを使用していつでもリフレッシュできます。 **IMMEDIATE** マテリアライズ照会表内のデータは即時にリフレッシュされます。

表 *187. SYSPARTITIONMQTS* ビュー (続き)

列名	システム列名	データ・タイプ	説明
ISOLATION	ISOLATION	VARCHAR(27)	マテリアライズ照会表のリフレッシュ時に使用される分離レベルを示します。 **NO COMMIT** 分離レベルは NO COMMIT です。 **UNCOMMITTED READ** 分離レベルは UNCOMMITTED READ です。 **CURSOR STABILITY** 分離レベルは CURSOR STABILITY です。 **CURSOR STABILITY KEEP LOCKS** 分離レベルは CURSOR STABILITY KEEP LOCKS です。 **READ STABILITY** 分離レベルは READ STABILITY です。 **REPEATABLE READ** 分離レベルは REPEATABLE READ です。
SORT_SEQUENCE	SRTSEQ	VARCHAR(12)	マテリアライズ照会表で照合順序を使用するかどうかを示します。 **BY HEX VALUE** マテリアライズ照会表では照合テーブルを使用しません。 ***LANGIDSHR** マテリアライズ照会表では共用重みソート順序 (SRTSEQ) を使用します。 ***LANGIDUNQ** マテリアライズ照会表では固有重みソート順序 (SRTSEQ) を使用します。 **ALTSEQ** マテリアライズ照会表では代替照合順序 (ALTSEQ) を使用します。
LANGUAGE_IDENTIFIER	LANGID	CHAR(3) NULL 可能	マテリアライズ照会表の言語 ID。ソート順序が 16 進数の場合は NULL が入ります。
SORT_SEQUENCE_SCHEMA	SRTSEQSCH	CHAR(10) NULL 可能	使用するソート順序のスキーマ名。スキーマ名がない場合は、NULL が入ります。
SORT_SEQUENCE_NAME	SRTSEQNAM	CHAR(10) NULL 可能	使用するソート順序の名前。ソート順序の名前がない場合は、NULL が入ります。
MQT_RESTORE_DEFERRED	MQTRSTDFR	VARCHAR(3)	MQT の復元がその従属表の 1 つの復元まで据え置かれるかどうかを示します。 **NO** MQT の復元は、その従属表の 1 つの復元まで据え置かれることはありません。 **YES** MQT の復元は、その従属表の 1 つの復元まで据え置かれます。

表 *187. SYSPARTITIONMQTS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
ROUNDING_MODE	DECFLTRND	VARCHAR(8) NULL 可能	マテリアライズ照会表の DECFLOAT 丸めモードを示します。 **CEILING** ROUND_CEILING **DOWN** ROUND_DOWN **FLOOR** ROUND_FLOOR **HALFDOWN** ROUND_HALF_DOWN **HALFEVEN** ROUND_HALF_EVEN **HALFUP** ROUND_HALF_UP **UP** ROUND_UP DECFLOAT 列、関数、または定数を参照する式がマテリアライズ照会表にない場合は、NULL 値が入ります。
DECFLOAT_WARNING	DECFLTWRN	VARCHAR(3) NULL 可能	DECFLOAT 警告が返されるかどうかを指定します。 **NO** DECFLOAT 警告は返されません。 **YES** DECFLOAT 警告が返されます。 DECFLOAT 列、関数、または定数を参照する式がマテリアライズ照会表にない場合は、NULL 値が入ります。
MQT_DEFINITION	MQTDEF	VARGRAPHIC(5000) CCSID 1200	マテリアライズ照会表の照会。照会の長さが 5000 を超えた場合は、列値の最後に「...」が返されます。
MQT_TEXT	LABEL	VARGRAPHIC(50) CCSID 1200 NULL 可能	マテリアライズ照会表のテキスト。マテリアライズ照会表にテキストが存在しない場合は、NULL が入ります。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	システムのスキーマ名。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	システム表名。
SYSTEM_TABLE_MEMBER	SYS_MNAME	CHAR(10)	システム・メンバー名。

SYSPARTITIONSTAT

SYSPARTITIONSTAT ビューには、各表パーティションまたは表メンバーごとに行が 1 つずつ入ります。表が分散表の場合は、他のデータベース・ノードにあるパーティションは、このカタログ・ビューには含まれません。これらは該当の他のデータベース・ノードのカタログ・ビューに含まれます。

次の表は、SYSPARTITIONSTAT ビューの列について説明しています。

表 *188. SYSPARTITIONSTAT* ビュー

列名	システム列名	データ・タイプ	説明
TABLE_SCHEMA	TABSCHEMA	VARCHAR(128)	該当の表が入っている SQL のスキーマの名前。
TABLE_NAME	TABNAME	VARCHAR(128)	表の名前。
TABLE_PARTITION	TABPART	VARCHAR(128)	表パーティションまたはメンバーの名前。
PARTITION_TYPE	PARTTYPE	CHAR(1)	表パーティショニングのタイプ ブランク　表はパーティション化されません。 **H**　これはデータ・ハッシュ・パーティションです。 **R**　これはデータ範囲パーティションです。 **D**　これは分散データベース・ハッシュ・パーティションです。
PARTITION_NUMBER	PARTNBR	INTEGER NULL 可能	このパーティションのパーティション番号。表が分散表の場合には NULL が入ります。
NUMBER_DISTRIBUTED_PARTITIONS	DSTPARTS	INTEGER NULL 可能	表が分散表の場合にはパーティションの合計数が入ります。表が分散表でない場合には NULL が入ります。
NUMBER_ROWS	CARD	BIGINT	表パーティションまたはメンバー内の有効行数
NUMBER_ROW_PAGES	NPAGES	BIGINT	パーティションのデータ内の 64K ページの数
NUMBER_PAGES	FPAGES	BIGINT	NUMBER_ROW_PAGES と同じ
OVERFLOW	OVERFLOW	BIGINT	可変長セグメントにオーバーフローした見積もり行数。表に可変長または LOB 列が含まれていない場合、0 が入ります。
CLUSTERED	CLUSTERED	CHAR(1) NULL 可能	Db2 for i には適用されません。常に NULL になります。
ACTIVE_BLOCKS	ACTBLOCKS	BIGINT	Db2 for i には適用されません。常に -1 になります。
AVGCOMPRESSEDROWSIZE	ACROWSIZE	BIGINT	Db2 for i には適用されません。常に -1 になります。
AVGROWCOMPRESSIONRATIO	ACROWRATIO	REAL	Db2 for i には適用されません。常に -1 になります。
AVGROWSIZE	AVGROWSIZE	BIGINT	この表内の行の平均の長さ (バイト数)。表に可変長または LOB 列がある場合、-1 が入ります。
PCTROWSCOMPRESSED	PCTCROWS	REAL	Db2 for i には適用されません。常に -1 になります。
PCTPAGESSAVED	PCTPGSAVED	SMALLINT	Db2 for i には適用されません。常に -1 になります。
NUMBER_DELETED_ROWS	削除	BIGINT	表パーティションまたはメンバー内の削除された行の数
DATA_SIZE	SIZE	BIGINT	パーティションまたはメンバー内のデータ・スペースの合計サイズ (バイト数)
VARIABLE_LENGTH_SIZE	VLSIZE	BIGINT	パーティションまたはメンバー内の可変長データ・スペース・セグメントのサイズ (バイト数)

表 188. SYSPARTITIONSTAT ビュー (続き)

列名	システム列名	データ・タイプ	説明
FIXED_LENGTH_EXTENTS	FLEXTENTS	BIGINT	Db2 for i には適用されません。常に -1 になります。
VARIABLE_LENGTH_EXTENTS	VLEXTENTS	BIGINT	Db2 for i には適用されません。常に -1 になります。
COLUMN_STATS_SIZE	CSTATSSIZE	BIGINT	パーティションまたはメンバー内の列統計のサイズ (バイト数)
MAINTAINED_TEMPORARY_INDEX_SIZE	MTISIZE	BIGINT	パーティションまたはメンバー上で保守されているすべての一時索引のサイズ (バイト数)
NUMBER_DISTINCT_INDEXES	DISTINCTIX	INTEGER	パーティションまたはメンバー上に構築される特殊索引の数。これには、保守されている一時索引は含まれません。
OPEN_OPERATIONS	OPENS	BIGINT	最後の IPL 以降のパーティションまたはメンバーのフル・オープンの数
CLOSE_OPERATIONS	CLOSES	BIGINT	最後の IPL 以降のパーティションまたはメンバーのフル・クローズの数
INSERT_OPERATIONS	INSERTS	BIGINT	最後の IPL 以降のパーティションまたはメンバーに対する挿入操作の数
UPDATE_OPERATIONS	UPDATES	BIGINT	最後の IPL 以降のパーティションまたはメンバーに対する更新操作の数
DELETE_OPERATIONS	DELETES	BIGINT	最後の IPL 以降のパーティションまたはメンバーに対する削除操作の数
CLEAR_OPERATIONS	DSCLEARS	BIGINT	最後の IPL 以降のパーティションまたはメンバーに対するクリア操作 (CLRPFM 操作) の数
COPY_OPERATIONS	DSCOPIES	BIGINT	最後の IPL 以降のパーティションまたはメンバーに対するデータ・スペース・コピー操作 (特定の CPYxxx 操作) の数
REORGANIZE_OPERATIONS	DSREORGS	BIGINT	最後の IPL 以降のパーティションまたはメンバーに対するデータ・スペース再編成操作 (中断不可能な RGZPFM 操作) の数
INDEX_BUILDS	DSINXBLDS	BIGINT	最後の IPL 以降の、パーティションまたはメンバーを参照する索引の作成または再ビルドの数これには、保守されている一時索引は含まれません。
LOGICAL_READS	LGLREADS	BIGINT	最後の IPL 以降のパーティションまたはメンバーに対する論理読み取り操作の数
PHYSICAL_READS	PHYREADS	BIGINT	最後の IPL 以降のパーティションまたはメンバーに対する物理読み取り操作の数
SEQUENTIAL_READS	SEQREADS	BIGINT	最後の IPL 以降のパーティションまたはメンバーに対する順次読み取り操作の数
RANDOM_READS	RANREADS	BIGINT	最後の IPL 以降のパーティションまたはメンバーに対するランダム読み取り操作の数
CREATE_TIMESTAMP	CREATED	TIMESTAMP	パーティションまたはメンバーの作成タイム・スタンプ。
LAST_CHANGE_TIMESTAMP	LASTCHG	TIMESTAMP	パーティションまたはメンバーに対して行われた最終変更のタイム・スタンプ
LAST_SAVE_TIMESTAMP	LASTSAVE	TIMESTAMP NULL 可能	パーティションまたはメンバーの最終保管のタイム・スタンプ。そのパーティションまたはメンバーが保管されていない場合は、NULL が入ります。
LAST_RESTORE_TIMESTAMP	LASTRST	TIMESTAMP NULL 可能	パーティションまたはメンバーの最終復元のタイム・スタンプ。そのパーティションまたはメンバーが復元されていない場合は、NULL が入ります。
LAST_USED_TIMESTAMP	LASTUSED	TIMESTAMP NULL 可能	アプリケーションによってネイティブ・レコード入出力操作または SQL 操作にパーティションまたはメンバーが最後に直接使用されたときのタイム・スタンプ。そのパーティションまたはメンバーが使用されていない場合は、NULL が入ります。

表 *188. SYSPARTITIONSTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
DAYS_USED_COUNT	DAYSUSED	INTEGER	使用状況統計が前回リセットされた後で、アプリケーションによってネイティブ・レコード入出力操作または SQL 操作にパーティションまたはメンバーが直接使用された日数。使用状況統計が前回リセットされた後でそのパーティションまたはメンバーが一度も使用されていない場合には、0 が入ります。
LAST_RESET_TIMESTAMP	LASTRESET	TIMESTAMP NULL 可能	該当の表について使用状況統計が前回リセットされたときのタイム・スタンプ。詳しくは、オブジェクト記述変更 (CHGOBJD) コマンドを参照してください。パーティションまたはメンバーの最終使用時のタイム・スタンプが一度もリセットされていない場合は、NULL が入ります。
NEXT_IDENTITY_VALUE	NEXTVALUE	DECIMAL(31,0) NULL 可能	次の IDENTITY 値。推定値の場合もあります。表に IDENTITY 値がない場合は、NULL が入ります。
LOWINCLUSIVE	LOWINCL	CHAR(1) NULL 可能	パーティションの下限キー値が範囲の中に含まれているかどうかを示す標識。 **N** 下限キー値は範囲の中に含まれていません。 **Y** 下限キー値は範囲の中に含まれています。 表が範囲でパーティション化されていない場合は、NULL が入ります。
LOWVALUE	LOWVALUE	VARGRAPHIC(1024) CCSID 1200 NULL 可能	範囲パーティションの下限キー値のストリング表記。表が範囲でパーティション化されていない場合は、NULL が入ります。
HIGHINCLUSIVE	HIGHINCL	CHAR(1) NULL 可能	パーティションの上限キー値が範囲の中に含まれているかどうかを示す標識。 **N** 上限キー値は範囲の中に含まれていません。 **Y** 上限キー値は範囲の中に含まれています。 表が範囲でパーティション化されていない場合は、NULL が入ります。
HIGHVALUE	HIGHVALUE	VARGRAPHIC(1024) CCSID 1200 NULL 可能	範囲パーティションの上限キー値のストリング表記。表が範囲でパーティション化されていない場合は、NULL が入ります。
NUMBER_PARTITIONING_KEYS	NBRPKEYS	INTEGER NULL 可能	パーティション・キーの数。表がパーティション化されていない場合には NULL が入ります。
PARTITIONING_KEYS	PARTKEYS	VARCHAR(2880) NULL 可能	パーティション・キーのリスト。表がパーティション化されていない場合には NULL が入ります。
KEEP_IN_MEMORY	KEEPINMEM	CHAR(1)	パーティションをメモリー内に保持するかどうかを示します。 **0** メモリー設定がありません。 **1** 可能な場合、パーティションをメモリーに保持します。

表 *188. SYSPARTITIONSTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
MEDIA_PREFERENCE	MEDIAPREF	SMALLINT	パーティションのメディア設定を示します。 **0** メディア設定がありません。 **255** 可能な場合、パーティションを ソリッド・ステート・ディスク (SSD) に割り振ります。
LAST_SOURCE_UPDATE_TIMESTAMP	LASRSRCUPD	TIMESTAMP NULL 可能	ソース・メンバーに対する最後のソース変更のタイム・スタンプ。表がソース・ファイルでない場合には NULL が入ります。
SOURCE_TYPE	SRCTYPE	VARCHAR(10) NULL 可能	ソース・メンバーのソース・タイプ。表がソース・ファイルでない場合には NULL が入ります。
VOLATILE	VOLATILE	CHAR(1)	表が揮発性かどうかを示します。 **0** 表は揮発性ではありません。 **1** 表は揮発性です。
PARTITION_TEXT	LABEL	VARGRAPHIC(50) CCSID 1200 NULL 可能	パーティションのテキスト。パーティションにテキストが存在しない場合は、NULL が入ります。
PARTIAL_TRANSACTION	PARTIALTX	CHAR(1)	パーティションに部分トランザクションが含まれるかどうかを示します。 **N** パーティションに部分トランザクションは含まれません。 **Y** パーティションは保管されましたが、部分トランザクションでアクティブでした。 パーティションの後続の復元には、部分トランザクションが含まれます。トランザクションを完了させるために、ユーザーがジャーナルから変更を適用する必要があります。 **R** ロールバックが、完了前に異常終了しました。 これにより、パーティションに、ロールバック済みの行が部分的に残っています。
APPLY_STARTING_RECEIVER _LIBRARY	APYRCVLIB	VARCHAR(10) NULL 可能	開始ジャーナル・レシーバーを含むライブラリー APPLY_STARTING_RECEIVER が NULL の場合は、NULL が入ります。
APPLY_STARTING_RECEIVER	APYRCVNAME	VARCHAR(10) NULL 可能	パーティションが保管され、その後、復元されたことを示します。パーティションの保管時に表がジャーナルに記録された場合、開始ジャーナル・レシーバー名は、 APYJRNCHG がその後に使用された場合に開始するジャーナル・レシーバーを示します。 PARTIAL_TRANSACTION の値が R の場合、NULL を含みます。APYJRNCHG が実行されると、適用情報はクリアされます。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	システムのスキーマ名。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	システム表名。
SYSTEM_TABLE_MEMBER	SYS_MNAME	CHAR(10)	システム・メンバー名。

SYSPERIODS

SYSPERIODS ビューには、SQL スキーマ内のテンポラル表に定義されたすべての各期間ごとに、行が 1 つずつ入ります。

次の表は、SYSPERIODS ビューの列について説明しています。

表 *189. SYSPERIODS* ビュー

列名	システム列名	データ・タイプ	説明
PERIOD_NAME	PERIODNAME	VARCHAR(128)	期間の名前。
TABLE_SCHEMA	DBNAME	VARCHAR(128)	該当の表が入っている SQL のスキーマの名前。
TABLE_NAME	TBNAME	VARCHAR(128)	テンポラル表の名前。
BEGIN_COLUMN_NAME	BGNCOLNAME	VARCHAR(128)	期間開始列の名前。
END_COLUMN_NAME	ENDCOLNAME	VARCHAR(128)	期間終了列の名前
PERIOD_TYPE	PERIODTYPE	CHAR(1)	この行の期間のタイプ。 **S** システム期間
HISTORY_TABLE_SCHEMA	HSTDBNAME	VARCHAR(128) NULL 可能	履歴表のスキーマ名。 バージョン管理がシステム期間テンポラル表に追加されていない場合は、NULL 値が入ります。
HISTORY_TABLE_NAME	HSTTBNAME	VARCHAR(128) NULL 可能	履歴表の名前。 バージョン管理がシステム期間テンポラル表に追加されていない場合は、NULL 値が入ります。
ON_DELETE_ADD_EXTRA_ROW	ADD_ROW	VARCHAR(3) NULL 可能	ON DELETE ADD EXTRA ROW で定義されたバージョン管理。 **YES** ON DELETE ADD EXTRA ROW が指定されました。 **NO** ON DELETE ADD EXTRA ROW が指定されませんでした。 バージョン管理がシステム期間テンポラル表に追加されていない場合は、NULL 値が入ります。
VERSIONING_STATUS	VERSIONSTS	CHAR(1) NULL 可能	バージョン管理の状況 **E** システム期間テンポラル表と履歴表とのバージョン管理関係が確立されています。システム期間テンポラル表の変更された行の、以前のバージョンを保管するために、履歴表が使用されます。 **D** システム期間テンポラル表と履歴表とのバージョン管理関係は、定義されましたが、確立されていません。バージョン管理関係が確立されるか、システム期間テンポラル表のバージョン管理を削除するまで、システム期間テンポラル表に対する操作は阻止されます。 バージョン管理がシステム期間テンポラル表に追加されていない場合は、NULL 値が入ります。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	テンポラル表のシステム・スキーマ名。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	テンポラル表のシステム名。
SYSTEM_HISTORY_TABLE_SCHEMA	SYSHSTLIB	CHAR(10) NULL 可能	履歴表のシステム・スキーマ名。 バージョン管理がシステム期間テンポラル表に追加されていない場合は、NULL 値が入ります。

表 *189. SYSPERIODS* ビュー (続き)

列名	システム列名	データ・タイプ	説明
SYSTEM_HISTORY_TABLE_NAME	SYSHSTNAME	CHAR(10)	履歴表のシステム名。
		NULL 可能	バージョン管理がシステム期間テンポラル表に追加されていない場合は、NULL 値が入ります。
SYSTEM_BEGIN_COLUMN_NAME	SYSBGNCOL	CHAR(10)	期間開始システム列の名前。
SYSTEM_END_COLUMN_NAME	SYSENDCOL	CHAR(10)	期間終了システム列の名前

SYSPROCS

SYSPROCS ビューには、CREATE PROCEDURE ステートメントで作成された各プロシージャーごとに、行が 1 つずつ入ります。

次の表は、SYSPROCS ビューの列について説明しています。

表 *190. SYSPROCS* ビュー

列名	システム列名	データ・タイプ	説明
SPECIFIC_SCHEMA	SPECSCHEMA	VARCHAR(128)	ルーチン (プロシージャー) インスタンスのスキーマ名。
SPECIFIC_NAME	SPECNAME	VARCHAR(128)	ルーチン・インスタンスの特定名。
ROUTINE_SCHEMA	PROCSCHEMA	VARCHAR(128)	該当のルーチンが入っている SQL スキーマの名前。
ROUTINE_NAME	PROCNAME	VARCHAR(128)	ルーチンの名前。
ROUTINE_CREATED	RTNCREATE	TIMESTAMP	ルーチンが作成されたときのタイム・スタンプを識別します。
ROUTINE_DEFINER	DEFINER	VARCHAR(128)	該当のルーチンを定義したユーザーの名前。
ROUTINE_BODY	BODY	VARCHAR(8)	ルーチン本体のタイプ: **EXTERNAL** これは外部ルーチンです。 **SQL** これは SQL ルーチンです。
EXTERNAL_NAME	EXTNAME	VARCHAR(279) NULL 可能	この列は外部プログラム名を識別します。 • ILE サービス・プログラムの場合、外部プログラム名はスキーマ名/サービス・プログラム名 (入り口名) です。 • REXX の場合は、外部プログラム名は、スキーマ名/ソース・ファイル名 (メンバー名) です。 • Java プログラムの場合、外部プログラム名はオプションの jar-id の後に完全修飾クラス名!メソッド名 または完全修飾クラス名.メソッド名 が続きます。 • その他のすべての言語では、外部プログラム名は、スキーマ名/プログラム名 です。

表 *190. SYSPROCS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
EXTERNAL_LANGUAGE	LANGUAGE	VARCHAR(8) NULL 可能	これが外部ルーチンである場合は、この列は外部プログラム名を識別します。 **C** 外部プログラムは C で作成されます。 **C++** 外部プログラムは C++ で作成されます。 **CL** 外部プログラムは CL で作成されます。 **COBOL** 外部プログラムは COBOL で作成されます。 **COBOLLE** 外部プログラムは ILE COBOL で作成されます。 **FORTRAN** 外部プログラムは FORTRAN で作成されます。 **JAVA** 外部プログラムは JAVA で作成されます。 **PLI** 外部プログラムは PL/I で作成されます。 **REXX** 外部プログラムは REXX プロシージャーです。 **RPG** 外部プログラムは RPG で作成されます。 **RPGLE** 外部プログラムは ILE RPG で作成されます。 これが外部ルーチンでない場合は、NULL 値が入ります。
PARAMETER_STYLE	PARM_STYLE	VARCHAR(7) NULL 可能	これが外部ルーチンである場合は、この列はパラメーターのスタイル (呼び出し規則) を識別します。 **DB2GNRL** これは DB2GENERAL 呼び出し規則です。 **DB2SQL** これは DB2SQL 呼び出し規則です。 **GENERAL** これは GENERAL 呼び出し規則です。 **JAVA** これは JAVA 呼び出し規則です。 **NULLS** これは GENERAL WITH NULLS 呼び出し規則です。 **SQL** これは SQL 標準呼び出し規則です。 これが外部ルーチンでない場合は、NULL 値が入ります。
IS_DETERMINISTIC	DETERMINE	VARCHAR(3)	この列はルーチンが deterministic であるかどうかを識別します。 つまり、同じ引数のルーチンに対する呼び出しが、常に同じ結果を戻すかどうかを識別します。 **NO** ルーチンは deterministic ではありません。 **YES** ルーチンは deterministic です。

表 *190. SYSPROCS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
SQL_DATA_ACCESS	DATAACCESS	VARCHAR(8)	この列は、ルーチンに SQL が含まれているか、およびルーチンがデータの読み取りまたは変更を行うかを識別します。 **NONE** ルーチンは SQL ステートメントを含みません。 **CONTAINS** ルーチンは SQL ステートメントを含みます。 **READS** ルーチンは、おそらく表またはビューからデータを読み取ります。 **MODIFIES** ルーチンは、おそらく表またはビュー内のデータを変更するか、SQL DDL ステートメントを発行します。
SQL_PATH	SQL_PATH	VARCHAR(3483) NULL 可能	これが SQL ルーチンの場合、この列はパスを識別します。 これが SQL ルーチンでない場合は、NULL 値が入ります。
PARM_SIGNATURE	SIGNATURE	VARCHAR(16000)	この列はルーチン・シグニチャーを識別します。
RESULT_SETS	RESULTS	SMALLINT	戻される結果セットの最大数を識別します。 0 は結果セットがないことを示します。
IN_PARMS	IN_PARMS	SMALLINT	入力パラメーターの数を識別します。 0 は入力パラメーターがないことを示します。
OUT_PARMS	OUT_PARMS	SMALLINT	出力パラメーターの数を識別します。 0 は出力パラメーターがないことを示します。
INOUT_PARMS	INOUT_PARM	SMALLINT	入出力パラメーターの数を識別します。 0 は入出力パラメーターがないことを示します。
LONG_COMMENT	REMARKS	VARGRAPHIC(2000) CCSID 1200 NULL 可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、NULL 値が入ります。
ROUTINE_DEFINITION	ROUTINEDEF	DBCLOB(2M) CCSID 13488 NULL 可能	これが SQL ルーチンの場合、この列は SQL ルーチン本体を含みます。 これが難読化されたルーチンの場合、テキストは WRAPPED キーワードで始まり、その後に、エンコードされた形式のステートメント・テキストが続きます。 これが SQL ルーチンでない場合は、NULL 値が入ります。
DBINFO	DBINFO	VARCHAR(3) NULL 可能	データベースに関する情報をプロシージャーに渡すかどうかを識別します。 **NO** データベースに関する情報をプロシージャーに渡しません。 **YES** データベースに関する情報をプロシージャーに渡します。

表 *190. SYSPROCS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
COMMIT_ON_RETURN	CMTONRET	VARCHAR(3) NULL 可能	この列は、プロシージャーから正常に戻った時点でそのプロシージャーをコミットするかどうかを識別します。 **NO** プロシージャーから正常に戻ったときに、コミットは行われません。 **YES** プロシージャーから正常に戻ったときに、コミットが行われます。 **AUT** プロシージャーは自律的にコミットまたはロールバックします。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。
NEW_SAVEPOINT_LEVEL	NEWSAVEPTL	VARCHAR(3) NULL 可能	この列は、ルーチンが新しいセーブポイント・レベルを開始するかどうかを識別します。 **NO** 新しいセーブポイント・レベルを開始しません。 **YES** 新しいセーブポイント・レベルを開始します。
ROUNDING_MODE	DECFLTRND	CHAR(1) NULL 可能	これが SQL プロシージャーの場合は、DECFLOAT 丸めモードを識別します。 **C** ROUND_CEILING **D** ROUND_DOWN **F** ROUND_FLOOR **G** ROUND_HALF_DOWN **E** ROUND_HALF_EVEN **H** ROUND_HALF_UP **U** ROUND_UP プロシージャーが SQL プロシージャーでない場合は、NULL 値が入ります。
ROUTINE_TEXT	LABEL	VARGRAPHIC(50) CCSID 1200 NULL 可能	ルーチンのラベルが入ります。ラベルが存在しない場合は NULL 値が入ります。
AUTONOMOUS	AUTONOMOUS	VARCHAR(3)	この列は、プロシージャーが自律型かどうかを示します。 **NO** プロシージャーは自律型ではありません。 **YES** プロシージャーは自律型です。

SYSPROGRAMSTAT

SYSPROGRAMSTAT ビューには、SQL ステートメントを含むプログラム、サービス・プログラム、およびモジュールごとに 1 行ずつ含まれます。

以下の表では、SYSPROGRAMSTAT ビューの列について説明します。

表 *191. SYSPROGRAMSTAT* ビュー

列名	システム列名	データ・タイプ	説明
PROGRAM_SCHEMA	COLLID	VARCHAR(128)	スキーマの名前。
PROGRAM_NAME	NAME	VARCHAR(128)	プログラム、サービス・プログラム、またはモジュールの名前。SQL プロシージャー、関数、またはトリガーの場合、これは SQL オブジェクト名です。
PROGRAM_TYPE	PGMTYPE	VARCHAR(128)	オブジェクトのタイプ ***PGM** オブジェクトはプログラムです。 ***MODULE** オブジェクトはモジュールです。 ***SRVPGM** オブジェクトはサービス・プログラムです。
MODULE_NAME	MODNAME	VARCHAR(10) NULL 可能	ILE プログラムまたはサービス・プログラムのモジュール名。 これが ILE プログラムでもサービス・プログラムでもない場合は、NULL 値が入ります。
PROGRAM_OWNER	OWNER	VARCHAR(128)	プログラム、サービス・プログラム、またはモジュールの所有者
PROGRAM_CREATOR	CREATOR	VARCHAR(128)	プログラム、サービス・プログラム、またはモジュールの作成者
CREATION_TIMESTAMP	TIMESTAMP	TIMESTAMP	プログラム、サービス・プログラム、またはモジュールが作成されたときのタイム・スタンプ
DEFAULT_SCHEMA	QUALIFIER	VARCHAR(128) NULL 可能	修飾されていない表、ビュー、および索引の暗黙の名前。 デフォルトのスキーマ (DFTRDBCOL) が指定されなかった場合、または、プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
ISOLATION	ISOLATION	CHAR(2) NULL 可能	分離オプションの指定: **RR** 反復可能読み取り (*RR) **RS** 読み取り固定 (*ALL) **CS** カーソル固定 (*CS) **UR** 非コミット読み取り (*CHG) **NC** コミットなし (*NONE) プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
CONCURRENTACCESSRESOLUTION	CONCURRENT	CHAR(1) NULL 可能	並行アクセスの解決方法を指定します。 ブランク 指定しない **W** WAIT FOR OUTCOME **U** USE CURRENTLY COMMITTED プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。

表 *191. SYSPROGRAMSTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
NUMBER_STATEMENTS	NBRSTMTS	INTEGER	プログラム、サービス・プログラム、またはモジュール内の SQL ステートメントの数
PROGRAM_USED_SIZE	PGMSIZE	INTEGER	プログラム、サービス・プログラム、またはモジュール内の SQL ステートメントおよびアクセス・プランに使用されるバイトの数
NUMBER_COMPRESSIONS	PGM_CMP	INTEGER NULL 可能	プログラムまたはサービス・プログラムが圧縮された回数。 モジュールの場合、または、プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
STATEMENT_CONTENTION_COUNT	CONTENTION	BIGINT NULL 可能	新規アクセス・プランを保管しようとしたときに競合が発生した回数。 モジュールの場合、または、プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
ORIGINAL_SOURCE_FILE	SOURCE	VARCHAR(128) NULL 可能	プログラムまたはモジュールの作成に使用された完全修飾ソース・ファイルおよびメンバー。 SQL ルーチンの場合、または、プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
ORIGINAL_SOURCE_FILE_CCSID	SRC_CCSID	INTEGER NULL 可能	プログラムまたはモジュールの作成時に使用されたソース・ファイルの CCSID。 SQL ルーチンの場合、または、プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
ROUTINE_TYPE	RTNTYPE	VARCHAR(9) NULL 可能	ルーチンのタイプ。 **PROCEDURE** これはプロシージャーです。 **FUNCTION** これは関数です。 **TRIGGER** これはトリガーです。 モジュールの場合、もしくはプログラムまたはサービス・プログラムがプロシージャー、関数、またはトリガーではない場合は、NULL が入ります NUMBER_EXTERNAL_ROUTINES がゼロより大きい場合を除いて、外部プロシージャーは PROCEDURE として識別されません。
ROUTINE_BODY	BODY	VARCHAR(8) NULL 可能	ルーチン本体のタイプ: **EXTERNAL** これは外部ルーチンです。 **SQL** これは SQL ルーチンです。 モジュールの場合、もしくはプログラムまたはサービス・プログラムがプロシージャーまたは関数ではない場合は、NULL が入ります

表 *191. SYSPROGRAMSTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
FUNCTION_ORIGIN	ORIGIN	CHAR(1) NULL 可能	関数のタイプを識別します。これがプロシージャーの場合、この列にはブランクが入ります。 **B** これは組み込み関数 (Db2 for iによって定義された) です。 **E** これはユーザー定義関数です。 **U** これは、他の関数をソースとしているユーザー定義関数です。 **S** これはシステム生成関数です。 モジュールの場合、もしくはプログラムまたはサービス・プログラムがプロシージャーまたは関数ではない場合は、NULL が入ります
FUNCTION_TYPE	TYPE	CHAR(1) NULL 可能	関数の形式を識別します。これがプロシージャーの場合、この列にはブランクが入ります。 **S** これはスカラー関数です。 **C** これは列関数です。 **T** これは表関数です。 モジュールの場合、もしくはプログラムまたはサービス・プログラムがプロシージャー、関数、またはトリガーではない場合は、NULL が入ります
NUMBER_EXTERNAL_ROUTINES	NBREXTRTN	SMALLINT NULL 可能	プログラムまたはサービス・プログラムに保管されたプロシージャーおよび関数定義の数を示します。 モジュール、トリガー、または SQL ルーチンの場合は NULL が入ります。
EXTENDED_INDICATOR	EXTIND	VARCHAR(9) NULL 可能	EXTIND 属性を示します。 ***EXTIND** 拡張標識サポートは有効です。 ***NOEXTIND** 拡張標識サポートは有効ではありません。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
C_NUL_REQUIRED	CNULRQD	VARCHAR(10) NULL 可能	CNULRQD 属性を示します。 ***CNULRQD** C NUL は必須です。 ***NOCNULRQD** C NUL は必須ではありません。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
NAMING	NAMING	VARCHAR(4) NULL 可能	NAMING 属性を示します。 ***SYS** これはシステムによる命名です。 ***SQL** これは SQL による命名です。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。

表 *191. SYSPROGRAMSTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
TARGET_RELEASE	TGTRLS	VARCHAR(6) NULL 可能	プログラム、サービス・プログラム、またはモジュールのターゲット・リリースを示します (VxRxMx)。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
EARLIEST_POSSIBLE_RELEASE	MINRLS	VARCHAR(6) NULL 可能	プログラム、サービス・プログラム、またはモジュール内のすべての SQL ステートメントをサポートする最も古い IBM i リリースを示します (VxRxMx)。 **ANY** ステートメントは、サポートされている任意の IBM i リリースで有効です。 **VxRxMx** ステートメントは、IBM i VxRxMx リリースまたはそれ以降で有効です。 最も古いリリースがまだ判別されていない場合、または、プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
RDB	RDB	VARCHAR(18) NULL 可能	プログラム、サービス・プログラム、またはモジュールに対して指定された RDB を示します。 **rdb-name** リレーショナル・データベースの名前。 ***NONE** リレーショナル・データベースは指定されていません。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
CONSISTENCY_TOKEN	CONTOKEN	VARBINARY(8) NULL 可能	プログラムの整合性トークンを示します。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
ALLOW_COPY_DATA	ALWCPYDTA	VARCHAR(9) NULL 可能	ALWCPYDTA 属性を示します。 ***NO** データのコピーを使用することはできません。 ***OPTIMIZE** 結果としてパフォーマンスがよくなる可能性がある場合は常に、データのコピーが許可されます。 ***YES** 必要な場合に限り、データのコピーが許可されます。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。

表 *191. SYSPROGRAMSTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
CLOSE_SQL_CURSOR	CLOSQLCSR	VARCHAR(10) NULL 可能	CLOSQLCSR 属性を示します。 ***ENDACTGRP** 活動化グループが終了すると、SQL カーソルはクローズし、SQL 準備済みステートメントは暗黙的に廃棄され、LOCK TABLE ロックは解除されます。 ***ENDJOB** ジョブが終了すると、SQL カーソルはクローズされ、SQL 準備済みステートメントは暗黙的に廃棄され、LOCK TABLE ロックは解除されます。 ***ENDMOD** モジュールが終了すると、SQL カーソルはクローズし、SQL 準備済みステートメントは暗黙的に廃棄されます。LOCK TABLE ロックは、呼び出しスタックの最初の SQL プログラムが終了すると解除されます。 ***ENDPGM** プログラムが終了すると、SQL カーソルはクローズし、SQL 準備済みステートメントは暗黙的に廃棄されます。LOCK TABLE ロックは、呼び出しスタックの最初の SQL プログラムが終了すると解除されます。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
LOB_FETCH_OPTIMIZATION	OPTLOB	VARCHAR(9)	LOB 最適化属性を示します。 ***OPTLOB** カーソルの最初の FETCH によって、それ以降のすべての FETCH においてそのカーソルが LOB および XML 結果列にどのように使用されるかが決定されます。 ***NOOPTLOB** どの FETCH も LOB または XML 結果列を取り出してロケーターまたは変数のいずれかに入れることができます。
DECIMAL_POINT	DECPNT	VARCHAR(7)	SQL ステートメントで使用される数値定数の小数点を示します。 ***PERIOD** 小数点にピリオドを使用します。 ***COMMA** 小数点にコンマを使用します。
SQL_STRING_DELIMITER	STRDLM	VARCHAR(9)	SQL ステートメントでストリング区切り文字として使用される文字を示します。 ***APOSTSQL** ストリング区切り文字はアポストロフィ (') です。 ***QUOTESQL** ストリング区切り文字は引用符 (") です。

表 *191. SYSPROGRAMSTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
DATE_FORMAT	DATFMT	VARCHAR(4) NULL 可能	DATFMT 属性を示します。 ***JOB** 実行時にジョブで指定される日付形式を使用します。 ***USA** 日付形式は *USA です。 ***ISO** 日付形式は *ISO です。 ***EUR** 日付形式は *EUR です。 ***JIS** 日付形式は *JIS です。 ***MDY** 日付形式は *MDY です。 ***DMY** 日付形式は *DMY です。 ***YMD** 日付形式は *YMD です。 ***JUL** 日付形式は *JUL です。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
DATE_SEPARATOR	DATSEP	CHAR(1) NULL 可能	日付区切り記号を示します。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
TIME_FORMAT	TIMFMT	VARCHAR(4) NULL 可能	TIMFMT 属性を示します。 ***JOB** 実行時にジョブで指定される時刻形式を使用します。 ***USA** 時刻形式は *USA です。 ***ISO** 時刻形式は *ISO です。 ***EUR** 時刻形式は *EUR です。 ***JIS** 時刻形式は *JIS です。 ***HMS** 時刻形式は *HMS です。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
TIME_SEPARATOR	TIMSEP	CHAR(1) NULL 可能	時刻区切り記号を示します。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
DYNAMIC_DEFAULT_SCHEMA	DYNDFTCOL	VARCHAR(4) NULL 可能	動的 SQL ステートメントの暗黙の修飾に DFTRDBCOL の値を使用するかどうかを指定します。 ***NO** DFTDRBCOL で指定されているスキーマは、動的 SQL ステートメント用に使用されません。 ***YES** DFTDRBCOL で指定されているスキーマは、動的 SQL ステートメント用に使用されます。 デフォルトのスキーマ (DFTRDBCOL) が指定されなかった場合、または、プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。

表 *191. SYSPROGRAMSTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
CURRENT_RULES	SQLCURRULE	VARCHAR(4) NULL 可能	SQLCURRULE 属性を示します。 ***DB2** すべての SQL ステートメントの意味体系は、デフォルトにより、Db2 用に設定された規則に従います。 ***STD** すべての SQL ステートメントの意味体系は、デフォルトにより、 ISO および ANSI SQL の規格用に設定された規則に従います。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
ALLOW_BLOCK	ALWBLK	VARCHAR(8) NULL 可能	ALWBLK 属性を示します。 ***ALLREAD** 読み取り専用カーソルの場合は行がブロックされます。 ***NONE** カーソルに関するデータの検索のために、行はブロックされません。 ***READ** 以下の場合は、カーソルに関するデータの読み取り専用検索で、レコードがブロックされます。 • コミットメント制御 (COMMIT) パラメーターに *NONE が指定されているとき。 • FOR READ ONLY 文節によってカーソルが宣言されたとき、またはカーソルに関して位置指定 UPDATE ステートメントまたは DELETE ステートメントを実行できる動的ステートメントがないとき。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
DELAY_PREPARE	DLYPRP	VARCHAR(4) NULL 可能	DLYPRP 属性を示します。 ***NO** 動的ステートメントの妥当性検査は、動的ステートメントの作成時に実行されます。 ***YES** 動的ステートメントの妥当性検査は、その動的ステートメントが使用されるときまで遅延されます。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。

表 *191. SYSPROGRAMSTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
USER_PROFILE	USRPRF	VARCHAR(7) NULL 可能	権限検査に使用するユーザー・プロファイルを指定します。 ***USER** プログラムを実行しているユーザーのプロファイルが使用されます。 ***OWNER** プログラムの所有者とプログラムを実行するユーザーの両方のプロファイルが使用されます。 ***NAMING** 命名規則が *SQL の場合、*OWNER が使用されます。命名規則が *SYS の場合、*USER が使用されます。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
DYNAMIC_USER_PROFILE	DYNUSRPRF	VARCHAR(6) NULL 可能	動的 SQL ステートメントに使用するユーザー・プロファイルを指定します。 ***USER** ローカル動的 SQL ステートメントは、ジョブまたはスレッドのプロファイルのもとで実行されます。分散動的 SQL ステートメントは、アプリケーション・サーバー・ジョブのプロファイルのもとで実行されます。 ***OWNER** ローカル動的 SQL ステートメントが、プログラムの所有者のプロファイルのもとで実行されます。分散動的 SQL ステートメントは、SQL パッケージの所有者のプロファイルのもとで実行されます。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
SORT_SEQUENCE	SRTSEQ	VARCHAR(12) NULL 可能	プログラム、サービス・プログラム、またはモジュールが照合順序を使用するかどうかを示します。 **BY HEX VALUE** SQL 索引では照合テーブルを使用しません。 ***LANGIDSHR** SQL 索引では共用重みソート順序 (SRTSEQ) を使用します。 ***LANGIDUNQ** SQL 索引では固有重みソート順序 (SRTSEQ) を使用します。 **ALTSEQ** SQL 索引では代替照合順序 (ALTSEQ) を使用します。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
LANGUAGE_IDENTIFIER	LANGID	CHAR(3) NULL 可能	言語 ID ソート順序。 ソート順序が *LANGIDSHR でも *LANGIDUNQ でもない場合、または、プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。

表 *191. SYSPROGRAMSTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
SORT_SEQUENCE_SCHEMA	SRTSEQSCH	CHAR(10)	ソート順序表のシステム・スキーマ。
		NULL 可能	ソート順序が 16 進である場合、または、プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
SORT_SEQUENCE_NAME	SRTSEQNAME	CHAR(10)	ソート順序表の名前。
		NULL 可能	ソート順序が 16 進である場合、または、プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
RDB_CONNECTION_METHOD	RDBCNNMTH	VARCHAR(4)	CONNECT ステートメントに使用する意味体系を指定します。
		NULL 可能	***RUW** CONNECT (タイプ 1) の意味体系は、リモート作業単位をサポートするのに使用されます。 ***DUW** CONNECT (タイプ 2) の意味体系は、分散作業単位をサポートするのに使用されます。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
DECRESULT_MAXIMUM_PRECISION	DECMAXPRC	SMALLINT	最大精度を指定します。
		NULL 可能	**31** 最大精度は 31 です。 **63** 最大精度は 63 です。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
DECRESULT_MAXIMUM_SCALE	DECMAXSCL	SMALLINT	結果データ・タイプについて返される最大位取り (小数点の右側にある小数部の桁数)。
		NULL 可能	プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
DECRESULT_MINIMUM_DIVIDE_SCALE	DECMINDIV	SMALLINT	中間および結果の両方のデータ・タイプについて返される最小除算位取り (小数点の右側にある小数部の桁数)。
		NULL 可能	プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
DECFLOAT_ROUNDING_MODE	DECFLTRND	VARCHAR(8)	DECFLOAT 丸めモードを示します。
		NULL 可能	**CEILING** ROUND_CEILING **DOWN** ROUND_DOWN **FLOOR** ROUND_FLOOR **HALFDOWN** ROUND_HALF_DOWN **HALFEVEN** ROUND_HALF_EVEN **HALFUP** ROUND_HALF_UP **UP** ROUND_UP プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。

表 *191. SYSPROGRAMSTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
DECFLOAT_WARNING	DECFLTWRN	VARCHAR(3) NULL 可能	DECFLOAT 警告が返されるかどうかを指定します。 **NO** DECFLOAT 警告は返されません。 **YES** DECFLOAT 警告が返されます。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
SQLPATH	SQLPATH	VARCHAR(3483) NULL 可能	SQL パスを示します。 SQL パスが指定されていない場合、または、プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL 値が入ります。
DBGVIEW	DBGVIEW	VARCHAR(9) NULL 可能	ソース・デバッグ情報のタイプを指定します。 ***NONE** デバッグがありません。 ***SOURCE** デバッグ・ビューには、ソースと SQL INCLUDE ステートメントが含まれます。 ***STMT** デバッグ・ビューには、プリコンパイラー生成ステートメントが含まれます。 ***LIST** デバッグ・ビューには、コンパイルされたリストが含まれます。 ***LSTDBG** デバッグ・ビューには、OPM プログラムのコンパイルされたリストが含まれます。 **ALLOW** ソース・デバッグは Unified Debugger によって許可されます。 **DISALLOW** ソース・デバッグは Unified Debugger によって許可されません。 **DISABLE** ソース・デバッグは Unified Debugger によって許可されず、DEBUG MODE は変更できません。 プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。
DBGKEY	DBGKEY	VARCHAR(3) NULL 可能	ソース・デバッグ情報のタイプを指定します。 **NO** デバッグ暗号鍵 (DBGENCKEY) パラメーターで暗号鍵が指定されていませんでした。 **YES** デバッグ暗号鍵 (DBGENCKEY) パラメーターで鍵が指定されていました。 DBGENCKEY がサポートされていない場合、または、プログラムが SQL ステートメントのない外部プロシージャーである場合、NULL が入ります。

表 *191. SYSPROGRAMSTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
LAST_USED_TIMESTAMP	LASTUSED	TIMESTAMP NULL 可能	プログラム、サービス・プログラム、モジュールのいずれかが最後に使用されたときのタイム・スタンプ。 プログラム、サービス・プログラム、またはモジュールが一度も使用されていない場合、NULL が入ります。
DAYS_USED_COUNT	DAYSUSED	INTEGER	使用状況統計が最後にリセットされたとき以降に、プログラム、サービス・プログラム、モジュールのいずれかが使用された日数。使用状況統計が前回リセットされた後でそのプログラム、サービス・プログラム、またはモジュールが一度も使用されていない場合には、0 が入ります。
LAST_RESET_TIMESTAMP	LASTRESET	TIMESTAMP NULL 可能	使用状況統計が前回リセットされたときのタイム・スタンプ。 統計が一度もリセットされていない場合、NULL が入ります。
SYSTEM_PROGRAM_NAME	SYS_NAME	CHAR(10)	プログラム、サービス・プログラム、またはモジュールのシステム名
SYSTEM_PROGRAM_SCHEMA	SYS_DNAME	CHAR(10)	プログラム、サービス・プログラム、またはモジュールを含むスキーマのシステム名
IASP_NUMBER	IASPNUMBER	INTEGER	独立補助記憶域プール (IASP) 番号を指定します。
SYSTEM_TIME_SENSITIVE	SYSTIME	VARCHAR(3) NULL 可能	CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターがプログラム内の静的および動的 SQL ステートメントに影響するかどうかを指定します。 **YES** システム期間テンポラル表への参照は、CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターの値の影響を受けます。 **NO** システム期間テンポラル表への参照は、CURRENT TEMPORAL SYSTEM_TIME 特殊レジスターの値の影響を受けません。 プログラムが 7.3 より前のバージョンでコンパイルされたものである場合は、NULL が入り、これは NO と同じように扱われます。

SYSPROGRAMSTMTSTAT

SYSPROGRAMSTMTSTAT ビューには、プログラム、モジュール、またはサービス・プログラム内の各組み込み SQL ステートメントごとに、行が 1 つずつ入ります。

次の表は、SYSPROGRAMSTMTSTAT ビューの列について説明しています。

表 *192. SYSPROGRAMSTMTSTAT* ビュー

列名	システム列名	データ・タイプ	説明
PROGRAM_SCHEMA	COLLID	VARCHAR(128)	スキーマの名前。
PROGRAM_NAME	NAME	VARCHAR(128)	プログラム、サービス・プログラム、またはモジュールの名前。SQL プロシージャー、関数、またはトリガーの場合、これは SQL オブジェクト名です。
PROGRAM_TYPE	PGMTYPE	VARCHAR(128)	オブジェクトのタイプ: ***PGM** オブジェクトはプログラムです。 ***MODULE** オブジェクトはモジュールです。 ***SRVPGM** オブジェクトはサービス・プログラムです。
MODULE_NAME	MODNAME	VARCHAR(10) NULL 可能	ILE プログラムまたはサービス・プログラムのモジュール名。 これが ILE プログラムでもサービス・プログラムでもない場合は、NULL 値が入ります。
STATEMENT_NUMBER	STMTNBR	INTEGER	プログラム内のステートメント番号。
NUMBER_TIMES_EXECUTED	NBREXEC	INTEGER	このステートメントが実行された回数。CALL、SET、および VALUES INTO のステートメントでこの値は保守されません。
ROWS_AFFECTED	ROWCNT	INTEGER	このステートメントのすべての実行でフェッチ、更新、挿入、または削除された合計行数。ステートメントが FETCH、SET、VALUES INTO、UPDATE、INSERT、DELETE、および MERGE でない場合は、0 が入ります。カーソルを使用して実装されていない SET または VALUES INTO ステートメントの場合、0 が入ります。
NUMBER_HOST_VARIABLES	NBRHV	INTEGER	ステートメントに指定されたホスト変数の合計数。これには、入力ホスト変数と出力ホスト変数が含まれます。
NUMBER_INPUT_HOST_VARIABLES	NBRIHV	INTEGER	ステートメントに指定された入力ホスト変数の数。
WITH_HOLD	WITHHOLD	CHAR(3) NULL 可能	ステートメントに WITH HOLD オプションを指定します。 **YES** WITH HOLD 文節が指定されました。 文節が指定されていない場合は、NULL 値が入ります。
FETCH_ONLY	FETCHONLY	CHAR(3) NULL 可能	ステートメントに FOR READ ONLY オプションを指定します。 **YES** FOR READ ONLY 文節が指定されました。 文節が指定されていない場合は、NULL 値が入ります。

表 *192. SYSPROGRAMSTMTSTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
CONCURRENTACCESSRESOLUTION	CONCURRENT	CHAR(1) NULL 可能	ステートメントに並行アクセスの解決方法を指定します。 **W** WAIT FOR OUTCOME **U** USE CURRENTLY COMMITTED **S** SKIP LOCKED DATA ステートメント・レベルで並行アクセスが指定されなかった場合は、NULL 値が入ります。
NUMBER_REBUILDS	NBRREBLD	INTEGER	QDT またはアクセス・プランが再作成された回数。
ISOLATION	ISOLATION	CHAR(2) NULL 可能	分離オプションの指定: **RR** 反復可能読み取り (*RR) **RS** 読み取り固定 (*ALL) **CS** カーソル固定 (*CS) **UR** 非コミット読み取り (*CHG) **NC** コミットなし (*NONE) ステートメント・レベルで分離レベルが指定されなかった場合は、NULL 値が入ります。
NUMBER_ROWS_TO_OPTIMIZE	OPTROWS	INTEGER NULL 可能	OPTIMIZE FOR *n* ROWS 文節で指定された行数。 -1 は、値 *ALL が指定されたことを意味します。 文節が指定されていない場合は、NULL 値が入ります。
NUMBER_ROWS_TO_FETCH	FETCHROWS	INTEGER NULL 可能	FETCH FIRST *n* ROWS 文節で指定された行数。 文節が指定されていない場合は、NULL 値が入ります。
LAST_QDT_REBUILD_REASON	QDTRBLD	CHAR(2) NULL 可能	最後の QDT 再作成の理由コード。 QQRID = 1000 の場合、これは STRDBMON 出力ファイルの列 QVC22 に対応します。 ステートメントが QDT を使用しない場合、またはステートメントで QDT が一度も再作成されていない場合は、NULL 値が入ります。
STATEMENT_TEXT	STMTTEXT	DBCLOB(2M) CCSID(1200)	SQL ステートメントのテキスト。
SYSTEM_PROGRAM_NAME	SYS_NAME	CHAR(10)	プログラムのシステム名。
SYSTEM_PROGRAM_SCHEMA	SYS_DNAME	CHAR(10)	プログラムが入っているスキーマのシステム名。
ACCESS_PLAN_LENGTH	AP_LENGTH	INTEGER	ステートメントの QDT とアクセス・プランに使用されるバイトの数。
EARLIEST_POSSIBLE_RELEASE	MINRLS	VARCHAR(6)	この SQL ステートメントをサポートする最も古い IBM i リリース (VxRxMx)。 **ANY** ステートメントは、サポートされている任意の IBM i リリースで有効です。 **VxRxMx** ステートメントは、IBM i VxRxMx リリースまたはそれ以降で有効です。 最も古いリリースが不明の場合は、NULL 値が入ります。

表 *192. SYSPROGRAMSTMTSTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
SQL_DB2_GROUP_LEVEL	SQL_LEVEL	INTEGER NULL 可能	このステートメントで使用される SQL 言語構文が依存する最新の Db2 PTF グループ・レベル。 Db2 PTF グループに依存すると識別された SQL 構文がこのステートメントになかった場合は、NULL 値が入ります。
SERVICES_DB2_GROUP_LEVEL	SERV_LEVEL	INTEGER NULL 可能	このステートメントで参照されるサービスが依存する可能性がある最新の Db2 PTF グループ・レベル。 SQL ステートメントが IBM i 提供サービス、または Db2 for i 組み込み関数または組み込みグローバル変数を参照する場合、この列は値を含みます。 IBM i 提供サービスに名前が一致するオブジェクトに対する非修飾参照が SQL ステートメントに含まれる場合、この列は、その非修飾名を IBM i サービスと見なします。 Db2 PTF グループに依存すると識別された組み込み関数、組み込みグローバル変数、および IBM i 提供サービスがこのステートメントになかった場合は、NULL 値が入ります。

SYSREFCST

SYSREFCST ビューには、SQL のスキーマにある各外部キーごとに、行が 1 つずつ入ります。

次の表は、SYSREFCST ビューの列について説明しています。

表 *193. SYSREFCST* ビュー

列名	システム列名	データ・タイプ	説明
CONSTRAINT_SCHEMA	CDBNAME	VARCHAR(128)	該当の制約が入っているスキーマの名前。
CONSTRAINT_NAME	RELNAME	VARCHAR(128)	制約の名前。
UNIQUE_CONSTRAINT_SCHEMA	UNQDBNAME	VARCHAR(128) NULL 可能	参照制約によって参照された固有制約が入っている SQL のスキーマの名前。 固有制約が存在しない場合は NULL 値が入ります。一般的には、親表なしで表を復元すると、この状態になります。
UNIQUE_CONSTRAINT_NAME	UNQNAME	VARCHAR(128) NULL 可能	参照制約によって参照された固有制約の名前。 固有制約が存在しない場合は NULL 値が入ります。一般的には、親表なしで表を復元すると、この状態になります。
MATCH_OPTION	MATCH	VARCHAR(7)	突き合わせオプション。 常に NONE になります。
UPDATE_RULE	UPDATE	VARCHAR(11)	UPDATE の規則。 • NO ACTION • RESTRICT
DELETE_RULE	DELETE	VARCHAR(11)	DELETE の規則。 • NO ACTION • CASCADE • SET NULL • SET DEFAULT • RESTRICT
COLUMN_COUNT	COLCOUNT	INTEGER	外部キーの中の列の数。
SYSTEM_CONSTRAINT_SCHEMA	SYS_CDNAME	CHAR(10)	制約が入っているシステム・スキーマの名前。
SYSTEM_UNIQUE_CONSTRAINT_SCHEMA	SYS_UDNAME	CHAR(10) NULL 可能	制約が入っているシステム・スキーマの名前。 固有制約が存在しない場合は NULL 値が入ります。一般的には、親表なしで表を復元すると、この状態になります。

SYSROUTINEAUTH

SYSROUTINEAUTH ビューには、ルーチンに対して認可された各特権ごとに、行が 1 つずつ入ります。このカタログ・ビューを使用して、特定ユーザーが特定のルーチンに対する権限を持っているかどうかを判別することはできないので、注意してください。なぜなら、ルーチンを使用するための特権は、グループ・ユーザー・プロファイルまたは特殊権限 (*ALLOBJ など) を通じて獲得できるからです。

次の表は、SYSROUTINEAUTH ビューの列について説明しています。

表 *194. SYSROUTINEAUTH* ビュー

列名	システム列名	データ・タイプ	説明
GRANTOR	GRANTOR	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
GRANTEE	GRANTEE	VARCHAR(128)	特権を認可する対象のユーザー・プロファイル。
ROUTINE_SCHEMA	RTNSCHEMA	VARCHAR(128)	スキーマの名前
ROUTINE_NAME	RTNNAME	VARCHAR(128)	ルーチンの名前
SPECIFIC_SCHEMA	SPECSCHEMA	VARCHAR(128)	スキーマの特定名。
SPECIFIC_NAME	SPECNAME	VARCHAR(128)	ルーチンの特定名
PRIVILEGE_TYPE	PRIVTYPE	VARCHAR(10)	認可される特権 : **ALTER** ルーチンを変更する特権。 **EXECUTE** ルーチンを実行する特権。
IS_GRANTABLE	GRANTABLE	VARCHAR(3)	特権を他のユーザーに認可できるかどうかを示します。 **NO** 特権は認可できません。 **YES** 特権を認可できます。
AUTHORIZATION_LIST	AUTL	VARCHAR(10) NULL 可能	特権が権限を介して認可されている場合、権限リストの名前が入ります。 特権が権限リストを介して認可されるのではない場合は、NULL 値が入ります。

SYSROUTINEDEP

SYSROUTINEDEP ビューは、ルーチンの従属関係を記録します。

次の表は、SYSROUTINEDEP ビューの列について説明しています。

表 *195. SYSROUTINEDEP* ビュー

列名	システム列名	データ・タイプ	説明
SPECIFIC_SCHEMA	SPECSCHEMA	VARCHAR(128)	ルーチン (関数) インスタンスのスキーマ名。
SPECIFIC_NAME	SPECNAME	VARCHAR(128)	ルーチン・インスタンスの特定名。
OBJECT_CATALOG	BCATALOG	VARCHAR(128) NULL 可能 リレーショナル・データベース名が指定されていない場合は、NULL 値が入ります。	該当のオブジェクトが入っているリレーショナル・データベースの名前。
OBJECT_SCHEMA	BSCHEMA	VARCHAR(128)	該当のオブジェクトが入っている SQL スキーマの名前。
OBJECT_NAME	BNAME	VARCHAR(128)	該当のルーチンが従属しているオブジェクトの名前。
OBJECT_TYPE	BTYPE	CHAR(24)	ルーチンで参照されたオブジェクトのオブジェクト・タイプを示します。 **ALIAS** オブジェクトは別名です。 **FUNCTION** オブジェクトは関数です。 **INDEX** オブジェクトは索引です。 **MATERIALIZED QUERY TABLE** オブジェクトはマテリアライズ照会表です。 **PROCEDURE** オブジェクトはプロシージャーです。 **SCHEMA** オブジェクトはスキーマです。 **SEQUENCE** オブジェクトはシーケンスです。 **TABLE** オブジェクトは表です。 実行時に使用される実際のオブジェクトが別名、マテリアライズ照会表、またはビューであるとしても、オブジェクトがルーチンの作成時に存在しない場合、または OBJECT_SCHEMA が *LIBL である場合、 TABLE が戻される可能性があります。 **TYPE** オブジェクトは特殊タイプです。 **VARIABLE** オブジェクトは変数です。 **VIEW** オブジェクトはビューです。
PARM_SIGNATURE	SIGNATURE	BLOB(3M) NULL 可能	この列はルーチン・シグニチャーを識別します。 オブジェクトがルーチンでない場合は、NULL 値が入ります。
IASP_NUMBER	IASPNUMBER	SMALLINT	オブジェクトの独立補助記憶域プール (IASP) 番号を指定します。

表 *195. SYSROUTINEDEP* ビュー (続き)

列名	システム列名	データ・タイプ	説明
NUMBER_OF_PARMS	NUMPARMS	SMALLINT	パラメーターの数を識別します。
		NULL 可能	オブジェクトがルーチンでない場合は、NULL 値が入ります。

SYSROUTINES

SYSROUTINES ビューには、CREATE PROCEDURE ステートメントによって作成された各プロシージャーごとに、および CREATE FUNCTION ステートメントによって作成された各関数ごとに、行が 1 つずつ入ります。

次の表は、SYSROUTINES 表の列について説明しています。

表 *196. SYSROUTINES* 表

列名	システム列名	データ・タイプ	説明
SPECIFIC_SCHEMA	SPECSCHEMA	VARCHAR(128)	ルーチン (関数) インスタンスのスキーマ名。
SPECIFIC_NAME	SPECNAME	VARCHAR(128)	ルーチン・インスタンスの特定名。
ROUTINE_SCHEMA	RTNSCHEMA	VARCHAR(128)	該当のルーチンが入っている SQL スキーマの名前。
ROUTINE_NAME	RTNNAME	VARCHAR(128)	ルーチンの名前。
ROUTINE_TYPE	RTNTYPE	VARCHAR(9)	ルーチンのタイプ。 **PROCEDURE** これはプロシージャーです。 **FUNCTION** これは関数です。
ROUTINE_CREATED	RTNCREATE	TIMESTAMP	ルーチンが作成されたときのタイム・スタンプを識別します。
ROUTINE_DEFINER	DEFINER	VARCHAR(128)	該当のルーチンを定義したユーザーの名前。
ROUTINE_BODY	BODY	VARCHAR(8)	ルーチン本体のタイプ: **EXTERNAL** これは外部ルーチンです。 **SQL** これは SQL ルーチンです。
EXTERNAL_NAME	EXTNAME	VARCHAR(279) NULL 可能	この列は外部プログラム名を識別します。 • SQL 関数または ILE サービス・プログラムの場合、外部プログラム名はスキーマ名/サービス・プログラム名 (入り口名) です。 • REXX の場合は、外部プログラム名は、スキーマ名/ソース・ファイル名 (メンバー名) です。 • Java プログラムの場合、外部プログラム名はオプションの jar-id の後に完全修飾クラス名!メソッド名 または完全修飾クラス名.メソッド名 が続きます。 • その他のすべての言語では、外部プログラム名は、スキーマ名/プログラム名 です。 これがシステム生成関数でない場合は、NULL 値が入ります。

表 196. SYSROUTINES 表 (続き)

列名	システム列名	データ・タイプ	説明
EXTERNAL_LANGUAGE	LANGUAGE	VARCHAR(8) NULL 可能	これが外部ルーチンである場合は、この列は外部プログラム名を識別します。 **C** 外部プログラムは C で作成されます。 **C++** 外部プログラムは C++ で作成されます。 **CL** 外部プログラムは CL で作成されます。 **COBOL** 外部プログラムは COBOL で作成されます。 **COBOLLE** 外部プログラムは ILE COBOL で作成されます。 **FORTRAN** 外部プログラムは FORTRAN で作成されます。 **JAVA** 外部プログラムは JAVA で作成されます。 **PLI** 外部プログラムは PL/I で作成されます。 **REXX** 外部プログラムは REXX プロシージャーです。 **RPG** 外部プログラムは RPG で作成されます。 **RPGLE** 外部プログラムは ILE RPG で作成されます。 これが外部ルーチンでない場合は、NULL 値が入ります。
PARAMETER_STYLE	PARM_STYLE	VARCHAR(7) NULL 可能	これが外部ルーチンである場合は、この列はパラメーターのスタイル (呼び出し規則) を識別します。 **DB2GNRL** これは DB2GENERAL 呼び出し規則です。 **DB2SQL** これは DB2SQL 呼び出し規則です。 **GENERAL** これは GENERAL 呼び出し規則です。 **JAVA** これは JAVA 呼び出し規則です。 **NULLS** これは GENERAL WITH NULLS 呼び出し規則です。 **SQL** これは SQL 標準呼び出し規則です。 これが外部ルーチンでない場合は、NULL 値が入ります。

表 196. SYSROUTINES 表 (続き)

列名	システム列名	データ・タイプ	説明
IS_DETERMINISTIC	DETERMINE	VARCHAR(3)	この列はルーチンが deterministic であるかどうかを識別します。 つまり、同じ引数のルーチンに対する呼び出しが、常に同じ結果を戻すかどうかを識別します。 **NO** ルーチンは deterministic ではありません。 **YES** ルーチンはグローバル deterministic です。 **STM** ルーチンはステートメント deterministic です。
SQL_DATA_ACCESS	DATAACCESS	VARCHAR(8) NULL 可能	この列は、ルーチンに SQL が含まれているか、およびルーチンがデータの読み取りまたは変更を行うかを識別します。 **NONE** ルーチンは SQL ステートメントを含みません。 **CONTAINS** ルーチンは SQL ステートメントを含みます。 **READS** ルーチンは、おそらく表またはビューからデータを読み取ります。 **MODIFIES** ルーチンは、おそらく表またはビュー内のデータを変更するか、SQL DDL ステートメントを発行します。
SQL_PATH	SQL_PATH	VARCHAR(3483) NULL 可能	これが SQL ルーチンの場合、この列はパスを識別します。 これが SQL ルーチンでない場合は、NULL 値が入ります。
PARM_SIGNATURE	SIGNATURE	VARCHAR(16000)	この列はルーチン・シグニチャーを識別します。
NUMBER_OF_RESULTS	NUMRESULTS	SMALLINT	結果の数を識別します。
MAX_DYNAMIC_RESULT_SETS	RESULTS	SMALLINT	戻される結果セットの最大数を識別します。 0 は結果セットがないことを示します。
IN_PARMS	IN_PARMS	SMALLINT	入力パラメーターの数を識別します。 0 は入力パラメーターがないことを示します。
OUT_PARMS	OUT_PARMS	SMALLINT	出力パラメーターの数を識別します。 0 は出力パラメーターがないことを示します。
INOUT_PARMS	INOUT_PARM	SMALLINT	入出力パラメーターの数を識別します。 0 は入出力パラメーターがないことを示します。
PARSE_TREE	PARSE_TREE	VARCHAR(1024) ビット・データ用	CREATE FUNCTION ステートメントまたは CREATE PROCEDURE ステートメントの解析ツリーを識別します。これは内部的にしか使用されません。
PARM_ARRAY	PARM_ARRAY	BLOB(4M)	これが外部ルーチンである場合、この列は CREATE FUNCTION ステートメントまたは CREATE PROCEDURE ステートメントから構築されたパラメーター配列を識別します。これは内部的にしか使用されません。
LONG_COMMENT	REMARKS	VARGRAPHIC(2000) CCSID 1200 NULL 可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、NULL 値が入ります。

表 196. SYSROUTINES 表 (続き)

列名	システム列名	データ・タイプ	説明
ROUTINE_DEFINITION	ROUTINEDEF	DBCLOB(2M) CCSID 13488 NULL 可能	これが SQL ルーチンの場合、この列は SQL ルーチン本体を含みます。 これが難読化されたルーチンの場合、テキストは WRAPPED キーワードで始まり、その後に、エンコードされた形式のステートメント・テキストが続きます。 これが SQL ルーチンでない場合は、NULL 値が入ります。
FUNCTION_ORIGIN	ORIGIN	CHAR(1)	関数のタイプを識別します。これがプロシージャーの場合、この列にはブランクが入ります。 **B** これは組み込み関数 (Db2 for iによって定義された) です。 **E** これはユーザー定義関数です。 **U** これは、他の関数をソースとしているユーザー定義関数です。 **S** これはシステム生成関数です。
FUNCTION_TYPE	TYPE	CHAR(1)	関数の形式を識別します。これがプロシージャーの場合、この列にはブランクが入ります。 **S** これはスカラー関数です。 **C** これは列関数です。 **T** これは表関数です。
EXTERNAL_ACTION	EXTACTION	CHAR(1) NULL 可能	関数の呼び出しに外部的な作用があるかどうかを識別します。 **E** この関数には、外部的な副次作用があります。 **N** この関数には、外部的な副次作用はありません。 ルーチンがプロシージャーの場合は、NULL 値が入ります。
IS_NULL_CALL	NULL_CALL	VARCHAR(3) NULL 可能	入力パラメーターが NULL 値である場合に、関数を呼び出す必要があるかどうかを識別します。 **NO** この関数は、入力パラメーターが NULL 値の場合に呼び出す必要はありません。これがスカラー関数の場合は、いずれかのオペランドが NULL であれば、この関数の結果は暗黙的に NULL になります。これが表関数の場合は、いずれかのオペランドが NULL 値であれば、この関数の結果は空の表になります。 **YES** この関数は、入力オペランドが NULL でも呼び出す必要があります。 ルーチンがプロシージャーの場合は、NULL 値が入ります。

表 196. SYSROUTINES 表 (続き)

列名	システム列名	データ・タイプ	説明
SCRATCH_PAD	SCRATCHPAD	INTEGER NULL 可能	静的メモリー域 (スクラッチパッド) のアドレスが関数に渡されるかどうかを識別します。 **0** 関数にはスクラッチパッドはありません。 整数 関数に渡されるスクラッチパッドのサイズを示します。 ルーチンがプロシージャーの場合は、NULL 値が入ります。
FINAL_CALL	FINAL_CALL	VARCHAR(3) NULL 可能	関数がその作業域 (スクラッチパッド) の終結処理を行えるようにするために、関数への最終呼び出しを行う必要があるかどうかを示します。 **NO** 最終呼び出しは行いません。 **YES** ステートメントが完了したときに関数への最終呼び出しを行います。 ルーチンがプロシージャーの場合は、NULL 値が入ります。
PARALLELIZABLE	PARALLEL	VARCHAR(3) NULL 可能	関数が並行して実行できるかどうかを識別します。 **NO** 関数は同期させなければなりません。 **YES** 関数は並行して実行できます。 ルーチンがプロシージャーの場合は、NULL 値が入ります。
DBINFO	DBINFO	VARCHAR(3) NULL 可能	データベースに関する情報をルーチンに渡すかどうかを識別します。 **NO** データベース情報をルーチンに渡しません。 **YES** データベースに関する情報をルーチンに渡します。 ルーチンがプロシージャーの場合は、NULL 値が入ります。
SOURCE_SPECIFIC_SCHEMA	SRCSCHEMA	VARCHAR(128) NULL 可能	これがソース化関数であり、ソースがユーザー定義の場合、この列にはソース・スキーマが入ります。これがソース化関数であり、ソースが組み込みである場合、この列には 'QSYS2' が入ります。 ルーチンがソース化関数でない場合は、NULL 値が入ります。
SOURCE_SPECIFIC_NAME	SRCNAME	VARCHAR(128) NULL 可能	これがソース化関数であり、ソースがユーザー定義である場合、この列にはソース関数名の特定名が入ります。 ルーチンがソース化関数でない場合は、NULL 値が入ります。
IS_USER_DEFINED_CAST	CAST_FUNC	VARCHAR(3) NULL 可能	この関数が、特殊タイプの作成時に作成されたキャスト関数であるかどうかを識別します。 **NO** この関数はキャスト関数ではありません。 **YES** この関数はキャスト関数です。 ルーチンがプロシージャーの場合は、NULL 値が入ります。

表 196. SYSROUTINES 表 (続き)

列名	システム列名	データ・タイプ	説明
CARDINALITY	CARD	BIGINT	表関数の基数を指定します。
		NULL 可能	この関数が表関数でない場合、または基数が指定されていなかった場合は、NULL 値が入ります。
FENCED	FENCED	VARCHAR(3)	関数を隔離するかどうかを指定します。
		NULL 可能	**NO** 関数を隔離しません。 **YES** 関数を隔離します。 ルーチンがプロシージャーの場合は、NULL 値が入ります。
COMMIT_ON_RETURN	CMTONRET	VARCHAR(3)	この列は、プロシージャーから正常に戻った時点でそのプロシージャーをコミットするかどうかを識別します。
		NULL 可能	**NO** プロシージャーから正常に戻ったときに、コミットは行われません。 **YES** プロシージャーから正常に戻ったときに、コミットが行われます。 **AUT** プロシージャーは自律的にコミットまたはロールバックします。 ルーチンが関数の場合は、NULL 値が入ります。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。
NEW_SAVEPOINT_LEVEL	NEWSAVEPTL	VARCHAR(3)	この列は、ルーチンが新しいセーブポイント・レベルを開始するかどうかを識別します。
		NULL 可能	**NO** 新しいセーブポイント・レベルを開始しません。 **YES** 新しいセーブポイント・レベルを開始します。 ルーチンが関数の場合は、NULL 値が入ります。
LAST_ALTERED	ALTEREDTS	TIMESTAMP	ルーチンが最後に変更されたときのタイム・スタンプ。ルーチンが一度も変更されていない場合には、NULL が入ります。
		NULL 可能	
DEBUG_MODE	DEBUG_MODE	CHAR(1)	ルーチンがデバッグ可能かどうかを識別します。
			0 ルーチンはデバッグ不可能です。 **1** ルーチンは Unified Debugger でデバッグ可能です。 **2** ルーチンはシステム・デバッガーでデバッグ可能です。 **N** ルーチンは Unified Debugger によるデバッグは使用不可です。
DEBUG_DATA	DEBUG_DATA	CLOB(1M)	予約済み。 NULL 値が入ります。
		NULL 可能	

表 *196. SYSROUTINES* 表 *(*続き*)*

列名	システム列名	データ・タイプ	説明
ROUNDING_MODE	DECFLTRND	CHAR(1) NULL 可能	これが SQL ルーチンの場合は、DECFLOAT 丸めモードを識別します。 **C** ROUND_CEILING **D** ROUND_DOWN **F** ROUND_FLOOR **G** ROUND_HALF_DOWN **E** ROUND_HALF_EVEN **H** ROUND_HALF_UP **U** ROUND_UP このルーチンが SQL ルーチンでない場合は、NULL 値が入ります。
ROUTINE_TEXT	LABEL	VARGRAPHIC(50) CCSID 1200 NULL 可能	ルーチンのラベルが入ります。ラベルが存在しない場合は NULL 値が入ります。
SECURE	SECURE	CHAR(1)	ルーチンが行アクセス制御と列アクセス制御においてセキュアであると見なされるかどうかを示します。 **N** ルーチンが行アクセス制御と列アクセス制御においてセキュアであると見なされません。 **Y** ルーチンが行アクセス制御と列アクセス制御においてセキュアであると見なされます。
ROUTINE_ENVIRONMENT	RTN_ENV	BLOB(16M) NULL 可能	デフォルト式と共に定義されたルーチンの内部環境情報を含みます。これがデフォルト式のないプロシージャーまたは関数の場合、NULL 値が入ります。
ROUTINE_DEFAULT_QDT	RTNDFTQDT	BLOB(1M) NULL 可能	デフォルト式と共に定義されたルーチンの内部構造を含みます。これがデフォルト式のないプロシージャーまたは関数の場合、NULL 値が入ります。
ROUTINE_BIND_OPTION	BINDOPT	DBCLOB(5000) CCSID 1200 NULL 可能	このルーチンの作成時に使用された追加のバインド・オプション。明示的なバインド・オプションが指定されなかった場合は、NULL 値が入ります。

SYSSCHEMAAUTH

SYSSCHEMAAUTH ビューには、スキーマに対して認可された各特権ごとに、行が 1 つずつ入ります。このカタログ・ビューを使用して、特定ユーザーが特定のスキーマに対する権限を持っているかどうかを判別することはできないので、注意してください。なぜなら、スキーマを使用するための特権は、グループ・ユーザー・プロファイルまたは特殊権限 (*ALLOBJ など) を通じて獲得できるからです。

次の表は、SYSSCHEMAAUTH ビューの列について説明しています。

表 *197. SYSSCHEMAAUTH* ビュー

列名	システム列名	データ・タイプ	説明
GRANTOR	GRANTOR	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
GRANTEE	GRANTEE	VARCHAR(128)	特権を認可する対象のユーザー・プロファイル。
SCHEMA_NAME	NAME	VARCHAR(128)	スキーマの名前
PRIVILEGE_TYPE	PRIVTYPE	VARCHAR(10)	認可される特権 : **CREATEIN** スキーマにオブジェクトを作成する特権。 **USAGE** スキーマを使用する特権。
IS_GRANTABLE	GRANTABLE	VARCHAR(3)	特権を他のユーザーに認可できるかどうかを示します。 **NO** 特権は認可できません。 **YES** 特権を認可できます。
AUTHORIZATION_LIST	AUTL	VARCHAR(10) NULL 可能	特権が権限を介して認可されている場合、権限リストの名前が入ります。 特権が権限リストを介して認可されるのではない場合は、NULL 値が入ります。
SYSTEM_SCHEMA_NAME	SYS_NAME	CHAR(10)	スキーマのシステム名

SYSSCHEMAS

SYSSCHEMAS ビューには、リレーショナル・データベースにある各スキーマごとに、行が 1 つずつ入ります。

単一のスキーマについての情報を得るには、表関数 OBJECT_STATISTICS を使用する照会のほうが、SYSSCHEMAS を照会するよりもはるかに良好な結果になります。例を以下に示します。

```
SELECT *
   FROM TABLE (QSYS2.OBJECT_STATISTICS('MJATST    ','LIB    ')) AS A
```

次の表は、SYSSCHEMAS ビューの列について説明しています。

表 *198. SYSSCHEMAS* ビュー

列名	システム列名	データ・タイプ	説明
SCHEMA_NAME	NAME	VARCHAR(128)	SQL スキーマの名前。
SCHEMA_OWNER	OWNER	VARCHAR(128)	スキーマの所有者。
SCHEMA_CREATOR	CREATOR	VARCHAR(128)	該当のスキーマを作成したユーザーの名前。
CREATION_TIMESTAMP	TIMESTAMP	TIMESTAMP	スキーマ作成されたときのタイム・スタンプ。
SCHEMA_SIZE	SIZE	DECIMAL(15,0)	スキーマのサイズ (バイト数)。
SCHEMA_TEXT	LABEL	VARGRAPHIC(50) CCSID 1200 NULL 可能	LABEL ステートメントで指定された文字ストリング。 スキーマにテキストがない場合は、NULL 値が入ります。
SYSTEM_SCHEMA_NAME	SYS_NAME	CHAR(10)	システムのスキーマ名。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。

SYSSEQUENCEAUTH

SYSSEQUENCEAUTH ビューには、シーケンスに対して認可された各特権ごとに、行が 1 つずつ入ります。このカタログ・ビューを使用して、特定ユーザーが特定のシーケンスに対する権限を持っているかどうかを判別することはできないので、注意してください。なぜなら、シーケンスを使用するための特権は、グループ・ユーザー・プロファイルまたは特殊権限 (*ALLOBJ など) を通じて獲得できるからです。

次の表は、SYSSEQUENCEAUTH ビューの列について説明しています。

表 199. SYSSEQUENCEAUTH ビュー

列名	システム列名	データ・タイプ	説明
GRANTOR	GRANTOR	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
GRANTEE	GRANTEE	VARCHAR(128)	特権を認可する対象のユーザー・プロファイル。
SEQUENCE_SCHEMA	SEQSCHEMA	VARCHAR(128)	スキーマの名前
SEQUENCE_NAME	SEQNAME	VARCHAR(128)	シーケンスの名前
PRIVILEGE_TYPE	PRIVTYPE	VARCHAR(10)	認可される特権 : **ALTER** シーケンスを変更する特権。 **USAGE** シーケンスを使用する特権。
IS_GRANTABLE	GRANTABLE	VARCHAR(3)	特権を他のユーザーに認可できるかどうかを示します。 **NO** 特権は認可できません。 **YES** 特権を認可できます。
AUTHORIZATION_LIST	AUTL	VARCHAR(10) NULL 可能	特権が権限を介して認可されている場合、権限リストの名前が入ります。 特権が権限リストを介して認可されるのではない場合は、NULL 値が入ります。
SYSTEM_SEQ_SCHEMA	SYSSSCHEMA	CHAR(10)	スキーマのシステム名
SYSTEM_SEQ_NAME	SYSSNAME	CHAR(10)	シーケンスのシステム名

SYSSEQUENCES

SYSSEQUENCES ビューには、SQL のスキーマにある各シーケンス・オブジェクトごとに、行が 1 つずつ入ります。

次の表は、SYSSEQUENCES ビューの列について説明しています。

表 *200. SYSSEQUENCES* ビュー

列名	システム列名	データ・タイプ	説明
SEQUENCE_SCHEMA	SEQSCHEMA	VARCHAR(128)	シーケンスが入っている SQL スキーマの名前。
SEQUENCE_NAME	SEQNAME	VARCHAR(128)	シーケンスの名前。
MAXIMUM_VALUE	MAXVALUE	DECIMAL(63,0)	シーケンスの最大値。
MINIMUM_VALUE	MINVALUE	DECIMAL(63,0)	シーケンスの最小値。
INCREMENT	INCREMENT	INTEGER	シーケンスの増分値。
CYCLE_OPTION	CYCLE	VARCHAR(3)	シーケンス値が最小値または最大値に達した後も、値の生成を続けるかどうかを識別します。 **NO** 値の生成は継続されません。 **YES** 値の生成は継続されます。
CACHE	CACHE	INTEGER	アクセスを高速化するために事前割り振りが可能なシーケンス値の数を指定します。ゼロは、値が事前割り振りされないことを示します。
ORDER	ORDER	VARCHAR(3)	要求された順序でシーケンス値を生成するかどうかを指定します。 **NO** 値は、要求された順序で生成する必要はありません。 **YES** 値は、要求された順序で生成する必要があります。
DATA_TYPE	DATA_TYPE	VARCHAR(128)	シーケンスのタイプ: **BIGINT** 大整数 **INTEGER** 長整数 **SMALLINT** 短整数 **DECIMAL** パック 10 進数 **NUMERIC** ゾーン 10 進数 **DISTINCT** 特殊タイプ
NUMERIC_PRECISION	PRECISION	INTEGER	数値の列すべての精度。
USER_DEFINED_TYPE_SCHEMA	TYPESCHEMA	VARCHAR(128)	特殊タイプの場合は、スキーマの名前。
		NULL 可能	シーケンスが特殊タイプでない場合は、NULL 値が入ります。
USER_DEFINED_TYPE_NAME	TYPENAME	VARCHAR(128)	特殊タイプの名前。
		NULL 可能	シーケンスが特殊タイプでない場合は、NULL 値が入ります。
START	START	DECIMAL(63,0)	シーケンスの開始値。

表 *200. SYSSEQUENCES* ビュー (続き)

列名	システム列名	データ・タイプ	説明
MAXASSIGNEDVAL	MAXASNVAL	DECIMAL(63,0) NULL 可能	最後に割り当てられる可能性のあるシーケンス値。この値には、キャッシュされたものの使用されていない値がすべて含まれます。 シーケンスが作成されている場合は、NULL 値が入ります。最初の値が割り当てられると、NULL ではなくなります。
SEQUENCE_DEFINER	DEFINER	VARCHAR(128)	シーケンスが作成された権限 ID。
SEQUENCE_CREATED	CREATEDTS	TIMESTAMP	シーケンスが作成されたときのタイム・スタンプ。
LAST_ALTERED_TIMESTAMP	ALTEREDTS	TIMESTAMP	シーケンスが最後に変更されたときのタイム・スタンプ。
SEQUENCE_TEXT	LABEL	VARGRAPHIC(50) CCSID 1200 NULL 可能	LABEL ステートメント (シーケンス・テキスト) で指定された文字ストリング。 シーケンスにシーケンス・テキストがない場合は、NULL 値が入ります。
LONG_COMMENT	REMARKS	VARGRAPHIC(2000) CCSID 1200 NULL 可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、NULL 値が入ります。
SYSTEM_SEQ_SCHEMA	SYSSSCHEMA	CHAR(10)	スキーマのシステム名
SYSTEM_SEQ_NAME	SYSSNAME	CHAR(10)	シーケンスのシステム名
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。

SYSTABAUTH

SYSTABAUTH ビューには、表またはビューに対して認可された各特権ごとに、行が 1 つずつ入ります。このカタログ・ビューを使用して、特定ユーザーが特定の表またはビューに対する権限を持っているかどうかを判別することはできないので、注意してください。なぜなら、表またはビューを使用するための特権は、グループ・ユーザー・プロファイルまたは特殊権限 (*ALLOBJ など) を通じて獲得できるからです。

次の表は、SYSTABAUTH ビューの列について説明しています。

表 *201. SYSTABAUTH* ビュー

列名	システム列名	データ・タイプ	説明
GRANTOR	GRANTOR	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
GRANTEE	GRANTEE	VARCHAR(128)	特権を認可する対象のユーザー・プロファイル。
TABLE_SCHEMA	DBNAME	VARCHAR(128)	スキーマの名前
TABLE_NAME	NAME	VARCHAR(128)	表の名前。
PRIVILEGE_TYPE	PRIVTYPE	VARCHAR(10)	認可される特権 : **ALTER** 表を変更する特権。 **DELETE** 表から行を削除する特権。 **INDEX** 表の索引を作成する特権。 **INSERT** 表に行を挿入する特権。 **REFERENCES** 参照制約の中で表を参照する特権。 **SELECT** 表から行を選択する特権。 **UPDATE** 表を更新する特権。
IS_GRANTABLE	GRANTABLE	VARCHAR(3)	特権を他のユーザーに認可できるかどうかを示します。 **NO** 特権は認可できません。 **YES** 特権を認可できます。
AUTHORIZATION_LIST	AUTL	VARCHAR(10) NULL 可能	特権が権限を介して認可されている場合、権限リストの名前が入ります。 特権が権限リストを介して認可されるのではない場合は、NULL 値が入ります。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	スキーマのシステム名
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	表またはビューのシステム名

SYSTABLEDEP

SYSTABLEDEP ビューは、マテリアライズ照会表の従属関係を記録します。

次の表は、SYSTABLEDEP ビューの列について説明しています。

表 *202. SYSTABLEDEP* ビュー

列名	システム列名	データ・タイプ	説明
TABLE_SCHEMA	TABSCHEMA	VARCHAR(128)	該当の表、ビュー、または別名を含む SQL スキーマの名前。
TABLE_NAME	TABNAME	VARCHAR(128)	その表、ビュー、または別名の名前。 SQL の表名、ビュー名、または別名が存在する場合は、その SQL の表名、ビュー名または別名です。存在しない場合は、システムの表名、ビュー名、または別名です。
OBJECT_SCHEMA	BSCHEMA	VARCHAR(128)	該当のオブジェクトが入っている SQL スキーマの名前。
OBJECT_NAME	BNAME	VARCHAR(128)	該当のマテリアライズ照会表が従属しているオブジェクトの名前。
OBJECT_TYPE	BTYPE	CHAR(24)	マテリアライズ照会表で参照されたオブジェクトのオブジェクト・タイプを示します。 **FUNCTION** オブジェクトは関数です。 **TABLE** オブジェクトは表です。 **TYPE** オブジェクトは特殊タイプです。 **VIEW** オブジェクトはビューです。
IASP_NUMBER	IASPNUMBER	SMALLINT	オブジェクトの独立補助記憶域プール (IASP) 番号を指定します。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	システムのスキーマ名
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	システム表名。
PARM_SIGNATURE	SIGNATURE	BLOB(3M) NULL 可能	この列はルーチン・シグニチャーを識別します。 オブジェクトがルーチンでない場合は、NULL 値が入ります。

SYSTABLEINDEXSTAT

SYSTABLEINDEXSTAT ビューには、表をもとに作成された少なくとも 1 つのパーティションまたはメンバーがある各索引ごとに、行が 1 つずつ入ります。索引が複数のパーティションまたはメンバーをもとにしている場合、統計には、それらのパーティションとメンバーのすべてが含まれます。表が分散表の場合は、他のデータベース・ノードにあるパーティションは含まれません。これらは該当の他のデータベース・ノードのカタログ・ビューに含まれます。

次の表は、SYSTABLEINDEXSTAT ビューの列について説明しています。

表 *203. SYSTABLEINDEXSTAT* ビュー

列名	システム列名	データ・タイプ	説明
TABLE_SCHEMA	TABSCHEMA	VARCHAR(128)	該当の表が入っている SQL のスキーマの名前。
TABLE_NAME	TABNAME	VARCHAR(128)	表の名前。
PARTITION_TYPE	PARTTYPE	CHAR(1)	表パーティショニングのタイプ ブランク 表はパーティション化されません。 **H** これはデータ・ハッシュ・パーティションです。 **R** これはデータ範囲パーティションです。 **D** これは分散データベース・ハッシュ・パーティションです。
NUMBER_PARTITIONS	NBRPARTS	INTEGER	表のパーティションまたはメンバーの数
NUMBER_DISTRIBUTED_PARTITIONS	DSTPARTS	INTEGER NULL 可能	表が分散表の場合にはパーティションの合計数が入ります。表が分散表でない場合には NULL が入ります。
INDEX_SCHEMA	INDSCHEMA	VARCHAR(128)	索引、論理ファイル、または制約を含む SQL スキーマの名前。
INDEX_NAME	INDNAME	VARCHAR(128)	索引、論理ファイル、または制約の名前。
INDEX_TYPE	INDTYPE	VARCHAR(11)	索引のタイプ: **INDEX** 索引は SQL 索引です。 **LOGICAL** 索引は論理ファイルの一部です。 **PRIMARY KEY** 索引は基本キー制約です。 **UNIQUE** 索引はユニーク制約です。 **REFERENTIAL** 索引は外部キー制約です。
NUMBER_KEY_COLUMNS	INDKEYS	BIGINT	索引キーを定義する列の数。
COLUMN_NAMES	COLNAMES	VARCHAR(1024)	索引キーを定義する列名のコンマ区切りリスト。すべての列名の長さが 1024 を超える場合は、列値の最後に「...」が返されます。
NUMBER_LEAF_PAGES	NLEAF	BIGINT	Db2 for i には適用されません。常に -1 になります。
NUMBER_LEVELS	NLEVELS	SMALLINT	Db2 for i には適用されません。常に -1 になります。
FIRSTKEYCARD	KEYCARD1	BIGINT	すべての索引パーティションについて最初のキー値の合計種類数。コード化ベクトル索引の場合は、索引キー全体の固有値の総数です。
FIRST2KEYCARD	KEYCARD2	BIGINT	すべての索引パーティションについて最初の 2 つの列を使用するキーの合計種類数。索引がコード化ベクトル索引である場合は、-1 が返されます。

表 *203. SYSTABLEINDEXSTAT* ビュー (続き)

列名	システム列名	データ・タイプ	説明
FIRST3KEYCARD	KEYCARD3	BIGINT	すべての索引パーティションについて最初の 3 つの列を使用するキーの合計種類数。索引がコード化ベクトル索引である場合は、-1 が返されます。
FIRST4KEYCARD	KEYCARD4	BIGINT	すべての索引パーティションについて最初の 4 つの列を使用するキーの合計種類数。索引がコード化ベクトル索引である場合は、-1 が返されます。
FULLKEYCARD	KEYCARDF	BIGINT	すべての索引パーティションについて特殊なフル・キー値の合計数。索引に 4 つを超えるキー列がある場合、または索引がコード化ベクトル索引である場合は、-1 が返されます。
CLUSTERRATIO	CLSRATIO	SMALLINT	Db2 for i には適用されません。常に -1 になります。
CLUSTERFACTOR	CLSFACTOR	DOUBLE	Db2 for i には適用されません。常に -1 になります。
SEQUENTIAL_PAGES	SEQPAGES	BIGINT	Db2 for i には適用されません。常に -1 になります。
DENSITY	DENSITY	INTEGER	Db2 for i には適用されません。常に -1 になります。
PAGE_FETCH_PAIRS	FETCHPAIRS	VARCHAR(520)	Db2 for i には適用されません。常に -1 になります。
NUMBER_KEYS	NUMRIDS	BIGINT	すべての索引パーティションの合計キー数。索引が無効な場合、またはコード化ベクトル索引である場合は、-1 が返されます。
NUMRIDS_DELETED	NUMRIDSDLT	BIGINT	Db2 for i には適用されません。常に 0 になります。
NUM_EMPTY_LEAFS	EMPTYLEAFS	BIGINT	Db2 for i には適用されません。常に 0 になります。
AVERAGE_RANDOM_FETCH_PAGES	AVGRNDFTCH	DOUBLE	Db2 for i には適用されません。常に -1 になります。
AVERAGE_RANDOM_PAGES	AVGRNDPAGE	DOUBLE	Db2 for i には適用されません。常に -1 になります。
AVERAGE_SEQUENCE_GAP	AVGSEQGAP	DOUBLE	Db2 for i には適用されません。常に -1 になります。
AVERAGE_SEQUENCE_FETCH_GAP	AVGSEQFGAP	DOUBLE	Db2 for i には適用されません。常に -1 になります。
AVERAGE_SEQUENCE_PAGES	AVGSEQPAGE	DOUBLE	Db2 for i には適用されません。常に -1 になります。
AVERAGE_SEQUENCE_FETCH_PAGES	AVGSEQFPAG	DOUBLE	Db2 for i には適用されません。常に -1 になります。
AVGPARTITION_CLUSTERRATIO	PCLSRATIO	SMALLINT	Db2 for i には適用されません。常に -1 になります。
AVGPARTITION_CLUSTERFACTOR	PCLSFACTOR	DOUBLE	Db2 for i には適用されません。常に -1 になります。
AVGPARTITION_PAGE_FETCH_PAIRS	PFETCHPAIR	VARCHAR(520)	Db2 for i には適用されません。常に空ストリングになります。
DATAPARTITION_CLUSTERFACTOR	DCLSFACTOR	DOUBLE	データ・パーティションに関しての索引キーの「クラスタリング」を測定する統計。これは 0 から 1 の間の数値で、1 は完全なクラスタリングを表し、0 はクラスタリングがないことを表します。
INDCARD	INDCARD	BIGINT	索引内のキーの数。索引が無効な場合、またはコード化ベクトル索引である場合は、-1 が返されます。

表 *203. SYSTABLEINDEXSTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
INDEX_VALID	VALID	CHAR(1)	いずれかの索引が無効で、再作成が必要であるかどうかを示す標識。 **0** 索引の少なくとも 1 つのパーティションまたはメンバーが無効です。 **1** 索引のすべてのパーティションまたはメンバーが有効です。 **2** 索引は、STG(*FREE) で保管されました。
INDEX_HELD	HELD	CHAR(1)	索引の再作成保留について、ユーザーが現在それを保留しているかどうかを示します。 **0** 索引のパーティションまたはメンバーについて保留中であるか、保留されている再作成はありません。 **1** 索引の少なくとも 1 つのパーティションまたはメンバーに対する再作成保留は、ユーザーが保留しています。
CREATE_TIMESTAMP	CREATED	TIMESTAMP	索引のパーティションまたはメンバーが作成されたときの最大タイム・スタンプ
LAST_BUILD_TIMESTAMP	LASTBUILD	TIMESTAMP	索引のパーティションまたはメンバーが最後に再作成されたときの最大タイム・スタンプ
LAST_QUERY_USE	LASTQRYUSE	TIMESTAMP NULL 可能	使用状況統計が前回リセットされた後で、索引のいずれかのパーティションまたはメンバーが最後に照会で使用されたときの最大タイム・スタンプ。使用状況統計が前回リセットされた後でその索引のパーティションまたはメンバーが照会で一度も使用されていない場合には、NULL が入ります。
LAST_STATISTICS_USE	LASTSTUSE	TIMESTAMP NULL 可能	使用状況統計が前回リセットされた後で、索引のいずれかのパーティションまたはメンバーが最後にオプティマイザーによって統計用に使用されたときの最大タイム・スタンプ。使用状況統計が前回リセットされた後でその索引のパーティションまたはメンバーが統計用に一度も使用されていない場合には、NULL が入ります。
QUERY_USE_COUNT	QRYUSECNT	BIGINT	使用状況統計が前回リセットされた後で、索引のいずれかのパーティションまたはメンバーが照会で使用された合計回数。使用状況統計が前回リセットされた後でその索引のパーティションまたはメンバーが照会で一度も使用されていない場合には、0 が入ります。
QUERY_STATISTICS_COUNT	QRYSTCNT	BIGINT	使用状況統計が前回リセットされた後で、索引のいずれかのパーティションまたはメンバーがオプティマイザーによって統計用に使用された合計回数。使用状況統計が前回リセットされた後でその索引のパーティションまたはメンバーが統計用に一度も使用されていない場合には、0 が入ります。
LAST_USED_TIMESTAMP	LASTUSED	TIMESTAMP NULL 可能	アプリケーションによってネイティブ・レコード入出力操作または SQL 操作に索引のいずれかのパーティションまたはメンバーが最後に直接使用されたときの最大タイム・スタンプ。その索引のパーティションまたはメンバーが一度も使用されていない場合には、NULL が入ります。
DAYS_USED_COUNT	DAYSUSED	INTEGER	使用状況統計が前回リセットされた後で、アプリケーションによってネイティブ・レコード入出力操作または SQL 操作に索引のいずれかのパーティションまたはメンバーが直接使用された最大日数。使用状況統計が前回リセットされた後でその索引のパーティションまたはメンバーが一度も使用されていない場合には、0 が入ります。

表 *203. SYSTABLEINDEXSTAT* ビュー (続き)

列名	システム列名	データ・タイプ	説明
LAST_RESET_TIMESTAMP	LASTRESET	TIMESTAMP NULL 可能	該当の索引について使用状況統計が前回リセットされたときの最大タイム・スタンプ。詳しくは、オブジェクト記述変更 (CHGOBJD) コマンドを参照してください。索引の最終使用時のタイム・スタンプが一度もリセットされていない場合は、NULL が入ります。
INDEX_SIZE	SIZE	BIGINT	索引のすべてのパーティションまたはメンバーの二分木またはコード化ベクトル索引の合計サイズ (バイト数)。
ESTIMATED_BUILD_TIME	ESTBLDTIME	INTEGER	索引のいずれかのパーティションまたはメンバーの再作成に必要な最大概算時間 (秒)。
LAST_BUILD_TIME	LSTBLDTIME	INTEGER NULL 可能	索引が最後に作成されたときからの経過時間 (秒単位)。最新の作成情報が使用できない場合には、NULL が含まれます。
LAST_BUILD_KEYS	LSTBLDKEYS	BIGINT NULL 可能	索引が最後に作成されたときのキーの数。最新の作成情報が使用できない場合には、NULL が含まれます。
LAST_BUILD_DEGREE	LSTBLDDEG	SMALLINT NULL 可能	索引が最後に作成されたときの並列度。最新の作成情報が使用できない場合には、NULL が含まれます。
DELAYED_MAINT_KEYS	DLYKEYS	INTEGER NULL 可能	遅延保守索引のいずれかのパーティションまたはメンバーの二分木に挿入する必要のあるキーの最大数。索引が遅延保守索引でない場合は、NULL が入ります。
SPARSE	SPARSE	CHAR(1)	索引に従属表のすべての行のキーが含まれるかどうかを示します。 **0** 索引には、従属表のすべての行のキーが含まれます。 **1** 索引は選択/除外論理ファイルであり、従属表のすべての行のキーは含まれません。
DERIVED_KEY	DERIVED	CHAR(1)	索引のキー列が式であるかどうかを示します。 **0** 索引のキー列は、式ではありません。 **1** 少なくとも 1 つのキー列が式です。現在、DDS で作成された論理ファイルまたは一時索引でのみ可能です。
PARTITIONED	PARTITION	CHAR(1)	索引がパーティション化されているか、パーティション化されていないかを示します。 **0** SQL 索引はパーティション化されていません (複数のパーティションにまたがっています)。 **1** パーティション表で索引が作成されていないか、パーティション表で索引が作成されていて、パーティション化されているかのどちらかです (複数のパーティションまたはメンバーにまたがっていません)。 **2** 索引は、複数のパーティションまたはメンバー上に構築された論理ファイルです。

表 *203. SYSTABLEINDEXSTAT* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
ACCPTH_TYPE	ACCPTHTYPE	CHAR(1)	索引のタイプを示します。 **0** 索引は、最大 1 テラバイト (*MAX1TB) の 2 進基数索引です。 **1** 索引は、最大 4 ギガバイト (*MAX4GB) の 2 進基数索引です。 **2** 索引は、コード化ベクトル索引です。
UNIQUE	UNIQUE	CHAR(1)	索引がユニークであるかどうかを示します。 **0** 索引は UNIQUE 索引です。 **1** 索引は UNIQUE WHERE NOT NULL 索引です。 **2** 索引は非ユニーク先入れ先出し (FIFO) 索引です。 **3** 索引は非ユニーク後入れ後出し (LIFO) 索引です。 **4** 索引は非ユニーク先変更先出し (FCFO) 索引です。
SRTSEQ_TYPE	SRTSEQ	CHAR(1)	索引で照合順序を使用するかどうかを示します。 **0** 索引では照合テーブルを使用しません。 **1** 索引では代替照合順序 (ALTSEQ) を使用します。 **2** 索引ではソート順序 (SRTSEQ) を使用します。
LOGICAL_PAGE_SIZE	PAGE_SIZE	INTEGER NULL 可能	索引の論理ページ・サイズ。索引がコード化ベクトル索引である場合は、NULL が入ります。
OVERFLOW_VALUES	OVERFLOW	INTEGER NULL 可能	コード化ベクトル索引のいずれかのパーティションまたはメンバーからオーバーフローした特殊キー値の最大数。索引がコード化ベクトル索引でない場合は、NULL が入ります。
EVI_CODE_SIZE	CODE_SIZE	INTEGER NULL 可能	コード化ベクトル索引のバイト・コードのサイズ。索引がコード化ベクトル索引でない場合は、NULL が入ります。
LOGICAL_READS	LGLREADS	BIGINT	最後の IPL 以降の、索引のいずれかのパーティションまたはメンバーに対する論理読み取り操作の合計数
PHYSICAL_READS	PHYREADS	BIGINT	Db2 for i には適用されません。常に 0 になります。
SEQUENTIAL_READS	SEQREADS	BIGINT	最後の IPL の後で索引に対して実行された順次読み取り操作の数。
RANDOM_READS	RANREADS	BIGINT	最後の IPL の後で索引に対して実行されたランダム読み取り操作の数。
SEARCH_CONDITION	IXWHERECON	VARGRAPHIC(1024) CCSID 1200	疎索引の場合はその索引の検索条件を示します。検索条件の長さが 1024 を超えた場合は、列値の最後に「...」が返されます。
KEEP_IN_MEMORY	KEEPINMEM	CHAR(1)	索引をメモリー内に保持するかどうかを示します。 **0** メモリー設定がありません。 **1** 可能な場合、索引をメモリーに保持します。

表 *203. SYSTABLEINDEXSTAT* ビュー (続き)

列名	システム列名	データ・タイプ	説明
MEDIA_PREFERENCE	MEDIAPREF	SMALLINT	索引のメディア設定を示します。 **0** メディア設定がありません。 **255** 可能な場合、索引を ソリッド・ステート・ディスク (SSD) に割り振ります。
INCLUDE_EXPRESSION	IXINCEXPR	VARGRAPHIC(1024) CCSID 1200 NULL 可能	索引の INCLUDE 式。索引に INCLUDE 式がない場合は、NULL が入ります。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	システムのスキーマ名。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	システム表名。

SYSTABLES

SYSTABLES ビューには、SQL スキーマの中にある各表、ビューまたは別名 (SQL カタログの表およびビューを含む) ごとに行が 1 つずつ入ります。

次の表は、SYSTABLES ビューの列について説明しています。

表 *204. SYSTABLES* ビュー

列名	システム列名	データ・タイプ	説明
TABLE_NAME	NAME	VARCHAR(128)	その表、ビュー、または別名の名前。 SQL の表名、ビュー名、または別名が存在する場合は、その SQL の表名、ビュー名または別名です。存在しない場合は、システムの表名、ビュー名、または別名です。
TABLE_OWNER	CREATOR	VARCHAR(128)	表、ビュー、または別名の所有者。
TABLE_TYPE	TYPE	CHAR(1)	表、ビュー、または別名を記述する行の場合： **A** 別名 **L** 論理ファイル **M** マテリアライズ照会表 **P** 物理ファイル **T** 表 **V** ビュー
COLUMN_COUNT	COLCOUNT	INTEGER	表またはビューの列の数。別名の場合はゼロです。
ROW_LENGTH	RECLENGTH[161]	INTEGER	表にあるレコードの最大長。別名の場合はゼロです。
TABLE_TEXT	LABEL	VARGRAPHIC(50) CCSID 1200	LABEL ステートメントで指定された文字ストリング。
LONG_COMMENT	REMARKS	VARGRAPHIC(2000) CCSID 1200 NULL 可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、NULL 値が入ります。
TABLE_SCHEMA	DBNAME	VARCHAR(128)	該当の表、ビュー、または別名を含む SQL スキーマの名前。
LAST_ALTERED_TIMESTAMP	ALTEREDTS	TIMESTAMP	表が最後に変更または作成されたときのタイム・スタンプ。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	システム表名。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	システムのスキーマ名
FILE_TYPE	FILETYPE	CHAR(1)	ファイル・タイプ **D** データ・ファイルまたは別名 **S** ソース・ファイル
BASE_TABLE_CATALOG	LOCATION	VARCHAR(18) NULL 可能	別名の場合、これは、その別名の基になっている表またはビューを含むリレーショナル・データベースの名前です。 表が別名でない場合は、NULL 値が入ります。
BASE_TABLE_SCHEMA	TBDBNAME	VARCHAR(128) NULL 可能	別名の場合、これは、その別名の基になっている表またはビューを含む SQL スキーマの名前です。 表が別名でない場合は、NULL 値が入ります。
BASE_TABLE_NAME	TBNAME	VARCHAR(128) NULL 可能	別名の場合、これは、その別名の基になっている表またはビューの名前です。 表が別名でない場合は、NULL 値が入ります。

表 *204. SYSTABLES* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
BASE_TABLE_MEMBER	TBMEMBER	VARCHAR(10) NULL 可能	別名の場合、これは、その別名の基になっているファイル・メンバーの名前です。これが別名であって、メンバー名が指定されていなかった場合は、*FIRST が入ります。 表が別名でない場合は、NULL 値が入ります。
SYSTEM_TABLE	SYSTABLE	CHAR(1)	システム表 **N** 表は、システム表ではありません。 **Y** 表は、システム表です。
SELECT_OMIT	SELECTOMIT	CHAR(1)	選択/除外論理ファイル **D** 表は動的選択/除外論理ファイルです。 **N** 表は、選択/除外論理ファイルではありません。 **Y** 表は、選択/除外論理ファイルです。
IS_INSERTABLE_INTO	INSERTABLE	VARCHAR(3)	表で INSERT を使用できるかどうかを識別します。 **NO** この表では INSERT は使用できません。 **YES** この表では INSERT を使用できます。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。
ENABLED	ENABLED	VARCHAR(3) NULL 可能	マテリアライズ照会表を最適化用に使用可能にするかどうかを指定します。 **NO** マテリアライズ照会表を最適化用に使用可能にしません。 **YES** マテリアライズ照会表を最適化用に使用可能にします。 表がマテリアライズ照会表でない場合は、NULL 値が入ります。
MAINTENANCE	MAINTAIN	VARCHAR(6) NULL 可能	マテリアライズ照会表をユーザーまたはシステムのどちらで保守するかを指定します。 **USER** マテリアライズ照会表はユーザーが保守します。 表がマテリアライズ照会表でない場合は、NULL 値が入ります。
REFRESH	REFRESH	VARCHAR(9) NULL 可能	マテリアライズ照会表 REFRESH オプションを示します。 **DEFERRED** マテリアライズ照会表は REFRESH DEFERRED です。 表がマテリアライズ照会表でない場合は、NULL 値が入ります。
REFRESH_TIME	REFRESHDTS	TIMESTAMP NULL 可能	最後のマテリアライズ照会表 REFRESH のタイム・スタンプを示します。 表がマテリアライズ照会表でない場合、またはこの表が一度もリフレッシュされていない場合は、NULL 値が入ります。
MQT_DEFINITION	MQTDEF	DBCLOB(2M) CCSID 13488 NULL 可能	マテリアライズ照会表の照会式を示します。 表がマテリアライズ照会表でない場合は、NULL 値が入ります。

表 *204. SYSTABLES* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
ISOLATION	ISOLATION	CHAR(2) NULL 可能	マテリアライズ照会表のリフレッシュ時に選択ステートメント に使用される分離レベルを示します。 **RR** 反復可能読み取り (*RR) **RS** 読み取り固定 (*ALL) **CS** カーソル固定 (*CS) **UR** 非コミット読み取り (*CHG) **NC** コミットなし (*NONE) 表がマテリアライズ照会表でない場合は、NULL 値が入ります。
PARTITION_TABLE	PART_TABLE	VARCHAR(11)	表がパーティション化された表かどうかを示します。 **DISTRIBUTED** 表は、分散表です。 **NO** 表は、パーティション化された表ではありません。 **YES** 表は、パーティション化された表です。
TABLE_DEFINER	DEFINER	VARCHAR(128)	該当の表を定義したユーザーの名前。
MQT_RESTORE_DEFERRED	MQTRSTDFR	CHAR(1)	表がマテリアライズ照会表の場合: **Y** MQT は復元の結果として据え置かれます。 **N** MQT は据え置かれません。 表がマテリアライズ照会表でない場合は、NULL 値が入ります。
ROUNDING_MODE	DECFLTRND	CHAR(1)	マテリアライズ照会表またはビューの DECFLOAT 丸めモードを示します。 **C** ROUND_CEILING **D** ROUND_DOWN **F** ROUND_FLOOR **G** ROUND_HALF_DOWN **E** ROUND_HALF_EVEN **H** ROUND_HALF_UP **U** ROUND_UP 表がビューまたはマテリアライズ照会表 (MQT) でない場合、または DECFLOAT 列、関数、または定数を参照する式が MQT またはビューにない場合は、NULL 値が入ります。
CONTROL	CONTROL	CHAR(1)	行アクセス制御または列アクセス制御の表が適用されているかどうかを示します。 ブランク アクセス制御の適用なし。 **R** 行アクセス制御を表に適用。 **C** 列アクセス制御を表に適用。 **B** 行アクセス制御と列アクセス制御の両方を表に適用。

表 *204. SYSTABLES* ビュー (続き)

列名	システム列名	データ・タイプ	説明
TEMPORAL_TYPE	TEMPORALTY	CHAR(1)	テンポラル表のタイプを示します。 **H** システム期間テンポラル表に関する履歴表 **N** テンポラル表ではない **S** システム期間テンポラル表

161. 長さはデータベース・バッファーで渡されたバイトの数であり、内部記憶長さではありません。

SYSTABLESTAT

SYSTABLESTAT ビューには、少なくとも 1 つのパーティションまたはメンバーがある各表ごとに、行が 1 つずつ入ります。表に複数のパーティションまたはメンバーがある場合、統計にはすべてのパーティションとメンバーが含まれます。表が分散表の場合は、他のデータベース・ノードにあるパーティションは含まれません。これらは該当の他のデータベース・ノードのカタログ・ビューに含まれます。

以下の表では、SYSTABLESTAT ビューの列について説明します。

表 *205. SYSTABLESTAT* ビュー

列名	システム列名	データ・タイプ	説明
TABLE_SCHEMA	TABSCHEMA	VARCHAR(128)	該当の表が入っている SQL のスキーマの名前。
TABLE_NAME	TABNAME	VARCHAR(128)	表の名前。
PARTITION_TYPE	PARTTYPE	CHAR(1)	表パーティショニングのタイプ ブランク　表はパーティション化されません。 **H**　これはデータ・ハッシュ・パーティションです。 **R**　これはデータ範囲パーティションです。 **D**　これは分散データベース・ハッシュ・パーティションです。
NUMBER_PARTITIONS	NBRPARTS	INTEGER	表のパーティションまたはメンバーの数
NUMBER_DISTRIBUTED_PARTITIONS	DSTPARTS	INTEGER NULL 可能	表が分散表の場合にはパーティションの合計数が入ります。表が分散表でない場合には NULL が入ります。
NUMBER_ROWS	CARD	BIGINT	表のすべてのパーティションまたはメンバー内の有効行数
NUMBER_ROW_PAGES	NPAGES	BIGINT	表のすべてのパーティションまたはメンバー内の 64K ページの数
NUMBER_PAGES	FPAGES	BIGINT	NUMBER_ROW_PAGES と同じ
OVERFLOW	OVERFLOW	BIGINT	可変長セグメントにオーバーフローした見積もり行数。表に可変長または LOB 列が含まれていない場合、0 が入ります。
CLUSTERED	CLUSTERED	CHAR(1) NULL 可能	Db2 for i には適用されません。常に NULL になります。
ACTIVE_BLOCKS	ACTBLOCKS	BIGINT	Db2 for i には適用されません。常に -1 になります。
AVGCOMPRESSEDROWSIZE	ACROWSIZE	BIGINT	Db2 for i には適用されません。常に -1 になります。
AVGROWCOMPRESSIONRATIO	ACROWRATIO	REAL	Db2 for i には適用されません。常に -1 になります。
AVGROWSIZE	AVGROWSIZE	BIGINT	この表内の行の平均の長さ (バイト数)。表に可変長または LOB 列がある場合、-1 が入ります。
PCTROWSCOMPRESSED	PCTCROWS	REAL	Db2 for i には適用されません。常に -1 になります。
PCTPAGESSAVED	PCTPGSAVED	SMALLINT	Db2 for i には適用されません。常に -1 になります。
NUMBER_DELETED_ROWS	DELETED	BIGINT	表のすべてのパーティションまたはメンバー内で削除された行の数
DATA_SIZE	SIZE	BIGINT	表のすべてのパーティションまたはメンバー内のデータ・スペースの合計サイズ (バイト数)
VARIABLE_LENGTH_SIZE	VLSIZE	BIGINT	表のすべてのパーティションまたはメンバー内の可変長データ・スペース・セグメントのサイズ (バイト数)

表 205. SYSTABLESTAT ビュー (続き)

列名	システム列名	データ・タイプ	説明
FIXED_LENGTH_EXTENTS	FLEXTENTS	BIGINT	Db2 for i には適用されません。常に -1 になります。
VARIABLE_LENGTH_EXTENTS	VLEXTENTS	BIGINT	Db2 for i には適用されません。常に -1 になります。
COLUMN_STATS_SIZE	CSTATSSIZE	BIGINT	表のすべてのパーティションまたはメンバー内の列統計のサイズ (バイト数)
MAINTAINED_TEMPORARY_INDEX_SIZE	MTISIZE	BIGINT	表のいずれかのパーティションまたはメンバー上で保守されているすべての一時索引のサイズ (バイト数)
NUMBER_DISTINCT_INDEXES	DISTINCTIX	INTEGER	表のいずれかのパーティションまたはメンバー上に構築される特殊索引の数。これには、保守されている一時索引は含まれません。
OPEN_OPERATIONS	OPENS	BIGINT	最後の IPL 以降の、表のすべてのパーティションまたはメンバーのフル・オープンの数
CLOSE_OPERATIONS	CLOSES	BIGINT	最後の IPL 以降の、表のすべてのパーティションまたはメンバーのフル・クローズの数
INSERT_OPERATIONS	INSERTS	BIGINT	最後の IPL 以降の、表のすべてのパーティションまたはメンバーの挿入操作の数
UPDATE_OPERATIONS	UPDATES	BIGINT	最後の IPL 以降の、表のすべてのパーティションまたはメンバーの更新操作の数
DELETE_OPERATIONS	DELETES	BIGINT	最後の IPL 以降の、表のすべてのパーティションまたはメンバーの削除操作の数
CLEAR_OPERATIONS	DSCLEARS	BIGINT	最後の IPL 以降の、表のすべてのパーティションまたはメンバーのクリア操作 (CLRPFM 操作) の数
COPY_OPERATIONS	DSCOPIES	BIGINT	最後の IPL 以降の、表のすべてのパーティションまたはメンバーのデータ・スペース・コピー操作 (特定の CPYxxx 操作) の数
REORGANIZE_OPERATIONS	DSREORGS	BIGINT	最後の IPL 以降の、表のすべてのパーティションまたはメンバーのデータ・スペース再編成操作 (中断不可能な RGZPFM 操作) の数
INDEX_BUILDS	DSINXBLDS	BIGINT	最後の IPL 以降の、表のいずれかのパーティションまたはメンバーを参照する索引の作成または再ビルドの数これには、保守されている一時索引は含まれません。
LOGICAL_READS	LGLREADS	BIGINT	最後の IPL 以降の、表のすべてのパーティションまたはメンバーの論理読み取り操作の数
PHYSICAL_READS	PHYREADS	BIGINT	最後の IPL 以降の、表のすべてのパーティションまたはメンバーの物理読み取り操作の数
SEQUENTIAL_READS	SEQREADS	BIGINT	最後の IPL 以降の、表のすべてのパーティションまたはメンバーの順次読み取り操作の数
RANDOM_READS	RANREADS	BIGINT	最後の IPL 以降の、表のすべてのパーティションまたはメンバーのランダム読み取り操作の数
LAST_CHANGE_TIMESTAMP	LASTCHG	TIMESTAMP	表のいずれかのパーティションまたはメンバーに対して行われた最終変更の最大タイム・スタンプ
LAST_SAVE_TIMESTAMP	LASTSAVE	TIMESTAMP NULL 可能	表のいずれかのパーティションまたはメンバーの最終保管の最小タイム・スタンプ。パーティションまたはメンバーが保管されていない場合は、NULL が入ります。
LAST_RESTORE_TIMESTAMP	LASTRST	TIMESTAMP NULL 可能	表のいずれかのパーティションまたはメンバーの最終復元の最大タイム・スタンプ。パーティションまたはメンバーが復元されていない場合は、NULL が入ります。
LAST_USED_TIMESTAMP	LASTUSED	TIMESTAMP NULL 可能	アプリケーションによってネイティブ・レコード入出力操作または SQL 操作にいずれかのパーティションまたはメンバーが最後に直接使用されたときの最大タイム・スタンプ。パーティションまたはメンバーが使用されていない場合は、NULL が入ります。

表 *205. SYSTABLESTAT* ビュー (続き)

列名	システム列名	データ・タイプ	説明
DAYS_USED_COUNT	DAYSUSED	INTEGER	使用状況統計が前回リセットされた後で、アプリケーションによってネイティブ・レコード入出力操作または SQL 操作にいずれかのパーティションまたはメンバーが直接使用された最大日数。使用状況統計が前回リセットされた後でパーティションまたはメンバーが使用されていない場合には、0 が入ります。
LAST_RESET_TIMESTAMP	LASTRESET	TIMESTAMP NULL 可能	該当の表について使用状況統計が前回リセットされたときの最大タイム・スタンプ。詳しくは、オブジェクト記述変更 (CHGOBJD) コマンドを参照してください。パーティションまたはメンバーの最終使用時のタイム・スタンプが一度もリセットされていない場合は、NULL が入ります。
NUMBER_PARTITIONING_KEYS	NBRPKEYS	INTEGER NULL 可能	パーティション・キーの数。表がパーティション化されていない場合には NULL が入ります。
PARTITIONING_KEYS	PARTKEYS	VARCHAR(2880) NULL 可能	パーティション・キーのリスト。表がパーティション化されていない場合には NULL が入ります。
VOLATILE	VOLATILE	CHAR(1)	表が揮発性かどうかを示します。 **0** 表は揮発性ではありません。 **1** 表は揮発性です。
HIGH_CONTENTION	CONTENTION	VARCHAR(3)	表がミラーリングで高競合と見なされるかどうかを示します。 **NO** 表は、高競合と見なされません。 **YES** 表は、高競合と見なされます。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10)	システムのスキーマ名。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10)	システム表名。

SYSTRIGCOL

SYSTRIGCOL ビューには、WHEN 文節またはトリガーの起動された SQL ステートメントによって暗黙的または明示的に参照された各列ごとに行が 1 つずつ入ります。

次の表は、SYSTRIGCOL ビューの列について説明しています。

表 *206. SYSTRIGCOL* ビュー

列名	システム列名	データ・タイプ	説明
TRIGGER_SCHEMA	TRIGSCHEMA	VARCHAR(128)	トリガーが入っているスキーマの名前。
TRIGGER_NAME	TRIGNAME	VARCHAR(128)	トリガーの名前。
TABLE_SCHEMA	TABSCHEMA	VARCHAR(128)	トリガーで参照された列を含む表またはビューが入っているスキーマの名前。
TABLE_NAME	TABNAME	VARCHAR(128)	トリガーで参照された列を含む表またはビューの名前。
COLUMN_NAME	TABCOLUMN	VARCHAR(128)	トリガーで参照された列の名前。
OBJECT_TYPE	BTYPE	CHAR(24)	トリガーで参照された列が入っているオブジェクトのオブジェクト・タイプを示します。 **FUNCTION** オブジェクトは関数です。 **MATERIALIZED QUERY TABLE** オブジェクトはマテリアライズ照会表です。 **TABLE** オブジェクトは表です。 **VIEW** オブジェクトはビューです。
SYSTEM_TRIGGER_SCHEMA	SYS_TDNAME	CHAR(10)	システムのスキーマ名。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10) NULL 可能	トリガーで参照された列を含む表またはビューのシステム・スキーマ名。 参照している表が存在しない場合、または参照しているオブジェクトが表関数である場合は、NULL 値が入ります。
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10) NULL 可能	トリガーで参照された列を含む表またはビューのシステム表名。 参照している表が存在しない場合、または参照しているオブジェクトが表関数である場合は、NULL 値が入ります。

SYSTRIGDEP

SYSTRIGDEP ビューには、WHEN 文節またはトリガーの起動された SQL ステートメントによって参照された各列ごとに行が 1 つずつ入ります。

次の表は、SYSTRIGDEP ビューの列について説明しています。

表 *207. SYSTRIGDEP* ビュー

列名	システム列名	データ・タイプ	説明
TRIGGER_SCHEMA	TRIGSCHEMA	VARCHAR(128)	トリガーが入っているスキーマの名前。
TRIGGER_NAME	TRIGNAME	VARCHAR(128)	トリガーの名前。
OBJECT_SCHEMA	BSCHEMA	VARCHAR(128)	トリガーで参照されたオブジェクトが入っているスキーマの名前。
OBJECT_NAME	BNAME	VARCHAR(128)	トリガーで参照されたオブジェクトの名前。
OBJECT_TYPE	BTYPE	CHAR(24)	トリガーで参照されたオブジェクトのオブジェクト・タイプを示します。 **ALIAS** オブジェクトは別名です。 **FUNCTION** オブジェクトは関数です。 **INDEX** オブジェクトは索引です。 **MATERIALIZED QUERY TABLE** オブジェクトはマテリアライズ照会表です。 **PACKAGE** オブジェクトはパッケージです。 **PROCEDURE** オブジェクトはプロシージャーです。 **SCHEMA** オブジェクトはスキーマです。 **SEQUENCE** オブジェクトはシーケンスです。 **TABLE** オブジェクトは表です。 **TYPE** オブジェクトは特殊タイプです。 **VARIABLE** オブジェクトは変数です。 **VIEW** オブジェクトはビューです。
PARM_SIGNATURE	SIGNATURE	BLOB(3M) NULL 可能	この列はルーチン・シグニチャーを識別します。 オブジェクトがルーチンでない場合は、NULL 値が入ります。
SYSTEM_TRIGGER_SCHEMA	SYS_TDNAME	CHAR(10)	システムのスキーマ名。
SYSTEM_OBJECT_SCHEMA	SYS_DNAME	CHAR(10) NULL 可能	トリガーで参照されたオブジェクトのシステム・スキーマ名。 参照しているオブジェクトが存在しない場合、または参照しているオブジェクトが別名、索引、マテリアライズ照会表、表、またはビューでない場合は、NULL 値が入ります。
SYSTEM_OBJECT_NAME	SYS_ONAME	CHAR(10) NULL 可能	トリガーで参照されたオブジェクトのシステム・オブジェクト名。 参照しているオブジェクトが存在しない場合、または参照しているオブジェクトが別名、索引、マテリアライズ照会表、表、またはビューでない場合は、NULL 値が入ります。

SYSTRIGGERS

SYSTRIGGERS ビューには、SQL スキーマにある各トリガーごとに、行が 1 つずつ入ります。

次の表は、SYSTRIGGERS ビューの列について説明しています。

表 *208. SYSTRIGGERS* ビュー

列名	システム列名	データ・タイプ	説明
TRIGGER_SCHEMA	TRIGSCHEMA	VARCHAR(128)	トリガーが入っているスキーマの名前。
TRIGGER_NAME	TRIGNAME	VARCHAR(128)	トリガーの名前。
EVENT_MANIPULATION	TRIGEVENT	VARCHAR(6)	トリガーを起動するイベントを示します。 **DELETE** DELETE 時にトリガーが起動されます。 **INSERT** INSERT 時にトリガーが起動されます。 **UPDATE** UPDATE 時にトリガーが起動されます。 **READ** 行の読み取り時にトリガーが起動されます。 これは、ADDPFTRG コマンドによって作成されたトリガーに対してのみ有効です。 **MULTI** トリガーは複数のイベントに対して起動されます。 EVENTUPDATE、EVENTDELETE、および EVENTINSERT 列がイベントを定義します。
EVENT_OBJECT_SCHEMA	TABSCHEMA	VARCHAR(128)	トリガーの対象表またはビューが入っているスキーマの名前。
EVENT_OBJECT_TABLE	TABNAME	VARCHAR(128)	トリガーの対象表またはビューの名前。
ACTION_ORDER	ORDERSEQNO	INTEGER	表またはビューのトリガー・リスト内のこのトリガーの順位。これはトリガーが起動される順序を示します。
ACTION_CONDITION	CONDITION	DBCLOB(2097152) CCSID 13488 NULL 可能	トリガーの WHEN 文節のテキスト。 WHEN 節がない場合、またはこれが難読化されたトリガーである場合、NULL が入ります。
ACTION_STATEMENT	TEXT	DBCLOB(2097152) CCSID 13488 NULL 可能	トリガー・アクション内の SQL ステートメントのテキスト。 これが難読化されたトリガーの場合、テキストは WRAPPED キーワードで始まり、その後に、エンコードされた形式のステートメント・テキストが続きます。 これが ADDPFTRG コマンドによって作成されたトリガーの場合は、NULL 値が入ります。
ACTION_ORIENTATION	GRANULAR	VARCHAR(9)	これが行トリガーであるか、ステートメント・トリガーであるかを示します。 **ROW** トリガーは各行ごとに起動されます。 **STATEMENT** トリガーは各ステートメントごとに起動されます。

表 *208. SYSTRIGGERS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
ACTION_TIMING	TRIGTIME	VARCHAR(7)	これが前トリガー、後トリガー、代用トリガーのいずれかであるかを示します。 **BEFORE** トリガーはトリガー・イベントの前に起動されます。 **AFTER** トリガーはトリガー・イベントの後に起動されます。 **INSTEAD** トリガーはトリガー・イベントの代わりに起動されます。
TRIGGER_MODE	TRIGMODE	VARCHAR(6)	トリガーの起動モードを示します。 **DB2SQL** トリガー・モードは DB2SQL です。 **DB2ROW** トリガー・モードは DB2ROW です。
ACTION_REFERENCE_OLD_ROW	OLD_ROW	VARCHAR(128)	OLD ROW 相関名。
		NULL 可能	OLD ROW 相関名が指定されていない場合は、NULL 値が入ります。
ACTION_REFERENCE_NEW_ROW	NEW_ROW	VARCHAR(128)	NEW ROW 相関名。
		NULL 可能	NEW ROW 相関名が指定されていない場合は、NULL 値が入ります。
ACTION_REFERENCE_OLD_TABLE	OLD_TABLE	VARCHAR(128)	OLD TABLE 相関名。
		NULL 可能	OLD TABLE 相関名が指定されていない場合は、NULL 値が入ります。
ACTION_REFERENCE_NEW_TABLE	NEW_TABLE	VARCHAR(128)	NEW TABLE 相関名。
		NULL 可能	NEW TABLE 相関名が指定されていない場合は、NULL 値が入ります。
SQL_PATH	SQL_PATH	VARCHAR(3483)	トリガーを作成するときに使用された SQL パス。
		NULL 可能	このトリガーが ADDPFTRG コマンドによって作成された場合は、NULL 値が入ります。
CREATED	CREATE_DTS	TIMESTAMP	トリガーが作成されたときのタイム・スタンプ。
TRIGGER_PROGRAM_NAME	TRIGPGM	VARCHAR(128)	トリガー・プログラムの名前。
TRIGGER_PROGRAM_LIBRARY	TRIGPGMLIB	VARCHAR(128)	トリガー・プログラムが入っているスキーマのシステム名。
OPERATIVE	OPERATIVE	VARCHAR(1)	トリガーが作動可能かどうかを示します。 表またはビューが持つトリガーがトリガー・アクション にその表またはビューの参照を含む場合、その表またはビューは自己参照表またはビューです。自己参照トリガーが、別のライブラリーに複写された場合、別のライブラリーに復元された場合、別のライブラリーに移された場合、または名前が変更された場合、トリガーは作動不能とマークされます。これは、トリガー・アクション 内の表参照は変わっていないために、依然として元のスキーマ名と表名が参照されるからです。 **Y** トリガーが作動可能です。 **N** トリガーは作動不能です。
ENABLED	ENABLED	VARCHAR(1)	トリガーが使用可能かどうかを示します。 **Y** トリガーは使用可能です。 **N** トリガーは使用不可です。

表 208. SYSTRIGGERS ビュー (続き)

列名	システム列名	データ・タイプ	説明
THREADSAFE	THDSAFE	VARCHAR(8)	トリガーがスレッド・セーフかどうかを示します。 **YES** トリガーがスレッド・セーフです。 **NO** トリガーはスレッド・セーフではありません。 **UNKNOWN** トリガーのスレッド・セーフティーは不明です。
MULTITHREADED_JOB_ACTION	MLTTHDACN	VARCHAR(8)	マルチスレッド・ジョブでトリガー・プログラムが呼び出されたときに取るアクションを示します。 **SYSVAL** QMLTTHDACN システム値を使用して、取るアクションを判別します。 **MSG** マルチスレッド・ジョブでトリガー・プログラムを実行しますが、診断メッセージを送信します。 **NORUN** マルチスレッド・ジョブでトリガー・プログラムを実行しません。 **RUN** マルチスレッド・ジョブでトリガー・プログラムを実行します。
ALLOW_REPEATED_CHANGE	ALWREPCHG	VARCHAR(8)	更新イベントがトリガーを起動する条件を示します。 **YES** トリガーは同じ行に対する反復変更を許します。 **NO** トリガーは同じ行に対する反復変更を許しません。
TRIGGER_UPDATE_CONDITION	TRGUPDCND	CHAR(8) NULL 可能	UPDATE トリガーは、更新イベント時には常に起動するのか、列値が実際に変更されているときにだけ起動するのかを示します。 **ALWAYS** トリガーは更新イベント時に常に起動されます。 **CHANGE** トリガーは、更新イベント時に列値が実際に変更されているときだけ起動します。 トリガーが UPDATE トリガーでない場合は、NULL 値が入ります。
TRIGGER_DEFINER	DEFINER	VARCHAR(128)	該当のトリガーを定義したユーザーの名前。
TRIGGER_TEXT	LABEL	VARGRAPHIC(50) CCSID 1200 NULL 可能	LABEL ステートメントで指定された文字ストリング。 ラベルがない場合は、NULL 値が入ります。
LONG_COMMENT	REMARKS	VARGRAPHIC(2000) CCSID 13488 NULL 可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、NULL 値が入ります。

表 *208. SYSTRIGGERS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
ROUNDING_MODE	DECFLTRND	CHAR(1) NULL 可能	トリガーの丸めモード。 **C** ROUND_CEILING **D** ROUND_DOWN **F** ROUND_FLOOR **G** ROUND_HALF_DOWN **E** ROUND_HALF_EVEN **H** ROUND_HALF_UP **U** ROUND_UP このトリガーが ADDPFTRG コマンドによって作成された場合は、NULL 値が入ります。
SYSTEM_TRIGGER_SCHEMA	SYS_TDNAME	CHAR(10)	システムのスキーマ名。
SYSTEM_EVENT_OBJECT_SCHEMA	SYS_DNAME	CHAR(10)	トリガーの対象表またはビューを含むスキーマのシステム・スキーマ名。
SYSTEM_EVENT_OBJECT_TABLE	SYS_TNAME	CHAR(10)	トリガーの対象表またはビューを含む表またはビューのシステム表名。
SECURE	SECURE	CHAR(1)	トリガーが行アクセス制御と列アクセス制御においてセキュアであると見なされるかどうかを示します。 **N** トリガーが行アクセス制御と列アクセス制御においてセキュアであると見なされません。 **Y** トリガーが行アクセス制御と列アクセス制御においてセキュアであると見なされます。
LAST_ALTERED	ALTEREDTS	TIMESTAMP NULL 可能	トリガーが最後に変更されたときのタイム・スタンプ。トリガーが一度も変更されていない場合、NULL 値が入ります。
EVENTUPDATE	EVENT_U	CHAR(1)	更新イベントによってこのトリガーが起動されるかどうかを示します。 **Y** はい **N** いいえ
EVENTINSERT	EVENT_I	CHAR(1)	挿入イベントによってこのトリガーが起動されるかどうかを示します。 **Y** はい **N** いいえ
EVENTDELETE	EVENT_D	CHAR(1)	削除イベントによってこのトリガーが起動されるかどうかを示します。 **Y** はい **N** いいえ
TRIGGER_BIND_OPTION	BINDOPT	DBCLOB(5000) CCSID 1200 NULL 可能	このトリガーの作成時に使用された追加のバインド・オプション。明示的なバインド・オプションが指定されなかった場合は、NULL 値が入ります。

SYSTRIGUPD

SYSTRIGUPD ビューには、UPDATE 列リスト (存在する場合) に識別された各列ごとに行が 1 つずつ入ります。

次の表は、SYSTRIGUPD ビューの列について説明しています。

表 *209. SYSTRIGUPD* ビュー

列名	システム列名	データ・タイプ	説明
TRIGGER_SCHEMA	TRIGSCHEMA	VARCHAR(128)	トリガーが入っているスキーマの名前。
TRIGGER_NAME	TRIGNAME	VARCHAR(128)	トリガーの名前。
EVENT_OBJECT_SCHEMA	TABSCHEMA	VARCHAR(128)	トリガーの対象表が入っているスキーマの名前。
EVENT_OBJECT_TABLE	TABNAME	VARCHAR(128)	トリガーの対象表の名前。
TRIGGERED_UPDATE_COLUMNS	TABCOLUMN	VARCHAR(128)	トリガーの UPDATE 列リストに指定された列の名前。
SYSTEM_TRIGGER_SCHEMA	SYS_TDNAME	CHAR(10)	システムのスキーマ名。
SYSTEM_EVENT_OBJECT_SCHEMA	SYS_DNAME	CHAR(10)	トリガーの対象表またはビューを含むスキーマのシステム・スキーマ名。
SYSTEM_EVENT_OBJECT_TABLE	SYS_TNAME	CHAR(10)	トリガーの対象表またはビューを含む表またはビューのシステム表名。

SYSTYPES

SYSTYPES 表には、CREATE TYPE ステートメントによって作成された各組み込みデータ・タイプおよび各特殊タイプや配列タイプごとに、行が 1 つずつ入ります。

次の表は、SYSTYPES 表の列について説明しています。

表 *210. SYSTYPES* 表

列名	システム列名	データ・タイプ	説明
USER_DEFINED_TYPE_SCHEMA	TYPESCHEMA	VARCHAR(128)	データ・タイプのスキーマ名。
USER_DEFINED_TYPE_NAME	TYPENAME	VARCHAR(128)	データ・タイプの名前。
USER_DEFINED_TYPE_DEFINER	DEFINER	VARCHAR(128)	該当のデータ・タイプを作成したユーザーの名前。
SOURCE_SCHEMA	SRCSCHEMA	VARCHAR(128) NULL 可能	このデータ・タイプのソース・データ・タイプのスキーマ。 これが組み込みデータ・タイプである場合は、NULL 値が入ります。
SOURCE_TYPE	SRCTYPE	VARCHAR(128) NULL 可能	このデータ・タイプのソース・データ・タイプの名前。 これが組み込みデータ・タイプである場合は、NULL 値が入ります。
SYSTEM_TYPE_SCHEMA	SYSTSCHEMA	CHAR(10)	データ・タイプのシステム・スキーマ名。
SYSTEM_TYPE_NAME	SYSTNAME	CHAR(10)	データ・タイプのシステム名。
METATYPE	METATYPE	CHAR(1)	データ・タイプのタイプを識別します。 **A** 配列データ・タイプ。 **S** システム事前定義データ・タイプ。 **T** ユーザー定義特殊タイプ。
TYPERULES	TYPERULES	CHAR(1)	ユーザー定義タイプの型規則を指定します。 **S** 強い型定義。 **W** 弱い型定義。 ブランク 適用されない。

表 *210. SYSTYPES* 表 *(*続き*)*

列名	システム列名	データ・タイプ	説明
LENGTH	LENGTH	INTEGER	データ・タイプの長さ属性。ただし、10 進数、数値、または非ゼロ精度 2 進数の列の場合は、その精度。配列データ・タイプの場合は、1 つの配列エレメントの長さになります。 **8** バイト　BIGINT **4** バイト　INTEGER **2** バイト　SMALLINT 数の精度　DECIMAL 数の精度　NUMERIC **8** バイト　FLOAT、FLOAT(n) (ここで n = 25 から 53)、または DOUBLE PRECISION **4** バイト　FLOAT(n) (ここで n = 1 から 24)、または REAL **8** バイト　DECFLOAT(16) **16** バイト DECFLOAT(34) ストリングの長さ CHARACTER ストリングの最大長 VARCHAR または CLOB グラフィック・ストリングの長さ GRAPHIC グラフィック・ストリングの最大長 VARGRAPHIC または DBCLOB **2** 進ストリングの長さ BINARY **2** 進ストリングの最大長 VARBINARY または BLOB **4** バイト　DATE **3** バイト　TIME **((*p*+1)/2) + 7 (***p* はタイム・スタンプの精度**)** の整数部分　TIMESTAMP データ・リンク **URL** およびコメントの最大長 DATALINK **40** バイト ROWID **2147483647** バイト XML ソース・タイプと同じ値 DISTINCT
NUMERIC_SCALE	SCALE	SMALLINT	数値データの位取り。
		NULL 可能	データ・タイプが 10 進数、数値、または 2 進数でない場合は、NULL 値が入ります。

表 *210. SYSTYPES* 表 (続き)

列名	システム列名	データ・タイプ	説明
CCSID	CCSID	INTEGER NULL 可能	CHAR、VARCHAR、CLOB、DATE、TIME、TIMESTAMP、GRAPHIC、VARGRAPHIC、DBCLOB、XML、および DATALINK データ・タイプの CCSID の値。 データ・タイプが数値の場合は、NULL 値が入ります。

表 *210. SYSTYPES* 表 *(*続き*)*

列名	システム列名	データ・タイプ	説明
STORAGE	STORAGE	INTEGER	データ・タイプの記憶域所要量: **8** バイト　BIGINT **4** バイト　INTEGER **2** バイト　SMALLINT **(**精度**/2) + 1** DECIMAL 数の精度　NUMERIC **8** バイト　FLOAT、FLOAT(n) (ここで n = 25 から 53)、または DOUBLE PRECISION **4** バイト　FLOAT(n) (ここで n = 1 から 24)、または REAL **8** バイト　DECFLOAT(16) **16** バイト DECFLOAT(34) ストリングの長さ CHAR ストリングの最大長 **+ 2** VARCHAR ストリングの最大長 **+ 29** CLOB ストリングの長さ *** 2** GRAPHIC ストリングの最大長 *** 2 + 2** VARGRAPHIC ストリングの最大長 *** 2 + 29** DBCLOB **2** 進ストリングの長さ BINARY **2** 進ストリングの最大長 **+ 2** VARBINARY ストリングの最大長 **+ 29** BLOB **4** バイト　DATE **3** バイト　TIME **((***p***+1)/2) + 7 (***p* はタイム・スタンプの精度**)** の整数部分　TIMESTAMP データ・リンク **URL** およびコメントの最大長 **+ 24**　DATALINK **42** バイト ROWID **2147483647** バイト **+ 29** XML ソース・タイプと同じ値 DISTINCT **注:** この列には、すべてのデータ・タイプの記憶域所要量があります。

表 *210. SYSTYPES* 表 (続き)

列名	システム列名	データ・タイプ	説明
NUMERIC_PRECISION	PRECISION	INTEGER NULL 可能	すべての数値データ・タイプの精度。 **注:** この列では、すべての数値データ・タイプ (10 進浮動小数点数、単精度および倍精度の浮動小数点数を含む) の精度を指定します。 NUMERIC_PRECISION_RADIX 列は、この列の値が 2 進数であるか、または 10 進数であるかを示します。 データ・タイプが数値でない場合は、NULL 値が入ります。
CHARACTER_MAXIMUM_LENGTH	CHARLEN	INTEGER NULL 可能	データ・タイプが 2 進ストリング、文字ストリング、グラフィック・ストリング、および XML の場合は、ストリングの最大長。 データ・タイプがストリングでない場合は、NULL 値が入ります。
CHARACTER_OCTET_LENGTH	CHARBYTE	INTEGER NULL 可能	データ・タイプが 2 進ストリング、文字ストリング、グラフィック・ストリング、および XML の場合は、バイト数。 データ・タイプがストリングでない場合は、NULL 値が入ります。
ALLOCATE	ALLOCATE	INTEGER NULL 可能	データ・タイプが 2 進数、可変長文字、可変長グラフィック、および XML データ・タイプの場合は、ストリングの割り振り済みの長さ。 データ・タイプが数値または固定長、あるいは配列データ・タイプの場合は、NULL 値が入ります。
NUMERIC_PRECISION_RADIX	RADIX	INTEGER NULL 可能	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 データ・タイプが数値でない場合は、NULL 値が入ります。
DATETIME_PRECISION	DATPRC	INTEGER NULL 可能	日付、時刻、またはタイム・スタンプの小数部分。 **0** データ・タイプが DATE および TIME の場合 **0-12** データ・タイプが TIMESTAMP の場合 (小数秒) データ・タイプが日付、時刻、またはタイム・スタンプでない場合は、NULL 値が入ります。
CREATE_TIME	CRTTIME	TIMESTAMP NULL 可能	データ・タイプが作成されたときのタイム・スタンプを識別します。
LONG_COMMENT	REMARKS	VARGRAPHIC(2000) CCSID 1200 NULL 可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、NULL 値が入ります。
IASP_NUMBER	IASPNUMBER	SMALLINT	データ・タイプの独立補助記憶域プール (IASP) 番号を指定します。
LAST_ ALTERED	ALTEREDTS	TIMESTAMP NULL 可能	予約済み。 NULL 値が入ります。

表 *210. SYSTYPES* 表 *(*続き*)*

列名	システム列名	データ・タイプ	説明
NORMALIZE_DATA	NORMALIZE	VARCHAR(3) NULL 可能	パラメーター値を正規化するかどうかを示します。この属性は UTF-8 および UTF-16 のデータのみに適用されます。 **NO** 値は正規化されません。 **YES** 値は正規化されます。
TYPE_TEXT	LABEL	VARGRAPHIC(50) CCSID 1200 NULL 可能	LABEL ステートメント (タイプ・テキスト) で指定された文字ストリング。 タイプにテキストがない場合は、NULL 値が入ります。
MAXIMUM_CARDINALITY	MAXCARD	BIGINT NULL 可能	配列データ・タイプの最大カーディナリティー。 タイプが配列タイプでない場合は、NULL 値が入ります。

SYSUDTAUTH

SYSUDTAUTH ビューには、タイプに対して認可された各特権ごとに、行が 1 つずつ入ります。このカタログ・ビューを使用して、特定ユーザーが特定のタイプに対する権限を持っているかどうかを判別することはできないので、注意してください。なぜなら、タイプを使用するための特権は、グループ・ユーザー・プロファイルまたは特殊権限 (*ALLOBJ など) を通じて獲得できるからです。

次の表は、SYSUDTAUTH ビューの列について説明しています。

表 *211. SYSUDTAUTH* ビュー

列名	システム列名	データ・タイプ	説明
GRANTOR	GRANTOR	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
GRANTEE	GRANTEE	VARCHAR(128)	特権を認可する対象のユーザー・プロファイル。
USER_DEFINED_TYPE_SCHEMA	UDT_SCHEMA	VARCHAR(128)	スキーマの名前
USER_DEFINED_TYPE_NAME	UDT_NAME	VARCHAR(128)	タイプの名前
PRIVILEGE_TYPE	PRIVTYPE	VARCHAR(10)	認可される特権： **ALTER** タイプを変更する特権。 **USAGE** タイプを使用する特権。
IS_GRANTABLE	GRANTABLE	VARCHAR(3)	特権を他のユーザーに認可できるかどうかを示します。 **NO** 特権は認可できません。 **YES** 特権を認可できます。
AUTHORIZATION_LIST	AUTL	VARCHAR(10) NULL 可能	特権が権限を介して認可されている場合、権限リストの名前が入ります。 特権が権限リストを介して認可されるのではない場合は、NULL 値が入ります。
SYSTEM_TYPE_SCHEMA	SYSTSCHEMA	CHAR(10)	スキーマのシステム名
SYSTEM_TYPE_NAME	SYSTNAME	CHAR(10)	タイプのシステム名

SYSVARIABLEAUTH

SYSVARIABLEAUTH ビューには、グローバル変数に対して認可された各特権ごとに、行が 1 つずつ入ります。このカタログ・ビューを使用して、特定ユーザーが特定のグローバル変数に対する権限を持っているかどうかを判別することはできないので、注意してください。なぜなら、グローバル変数を使用するための特権は、グループ・ユーザー・プロファイルまたは特殊権限 (*ALLOBJ など) を通じて獲得できるからです。

次の表は、SYSVARIABLEAUTH ビューの列について説明しています。

表 *212. SYSVARIABLEAUTH* ビュー

列名	システム列名	データ・タイプ	説明
GRANTOR	GRANTOR	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
GRANTEE	GRANTEE	VARCHAR(128)	特権を認可する対象のユーザー・プロファイル。
VARIABLE_SCHEMA	VARSCHEMA	VARCHAR(128)	スキーマの名前
VARIABLE_NAME	VARNAME	VARCHAR(128)	グローバル変数の名前
PRIVILEGE_TYPE	PRIVTYPE	VARCHAR(10)	認可される特権 : **ALTER** グローバル変数を変更する特権。 **READ** グローバル変数の値を読み取る特権。 **WRITE** グローバル変数に値を割り当てる特権。
IS_GRANTABLE	GRANTABLE	VARCHAR(3)	特権を他のユーザーに認可できるかどうかを示します。 **NO** 特権は認可できません。 **YES** 特権を認可できます。
AUTHORIZATION_LIST	AUTL	VARCHAR(10) NULL 可能	特権が権限を介して認可されている場合、権限リストの名前が入ります。 特権が権限リストを介して認可されるのではない場合は、NULL 値が入ります。
SYSTEM_VAR_SCHEMA	SYSVSCHEMA	CHAR(10)	スキーマのシステム名
SYSTEM_VAR_NAME	SYSVNAME	CHAR(10)	グローバル変数のシステム名

SYSVARIABLEDEP

SYSVARIABLEDEP 表では、変数の従属関係を記録します。

次の表は、SYSVARIABLEDEP 表の列について説明しています。

表 *213. SYSVARIABLEDEP* 表

列名	システム列名	データ・タイプ	説明
VARIABLE_SCHEMA	VARSCHEMA	VARCHAR(128)	変数のスキーマ名。
VARIABLE_NAME	VARNAME	VARCHAR(128)	変数の名前。
OBJECT_SCHEMA	BSCHEMA	VARCHAR(128)	該当のオブジェクトが入っている SQL スキーマの名前。
OBJECT_NAME	BNAME	VARCHAR(128)	該当の変数が従属しているオブジェクトの名前。
OBJECT_TYPE	BTYPE	CHAR(24)	変数で参照されたオブジェクトのオブジェクト・タイプを示します。 **ALIAS** オブジェクトは別名です。 **FUNCTION** オブジェクトは関数です。 **MATERIALIZED QUERY TABLE** オブジェクトはマテリアライズ照会表です。 **SCHEMA** オブジェクトはスキーマです。 **SEQUENCE** オブジェクトはシーケンスです。 **TABLE** オブジェクトは表です。 **TYPE** オブジェクトは特殊タイプです。 **VARIABLE** オブジェクトは変数です。 **VIEW** オブジェクトはビューです。
IASP_NUMBER	IASPNUMBER	SMALLINT	オブジェクトの独立補助記憶域プール (IASP) 番号を指定します。
PARM_SIGNATURE	SIGNATURE	BLOB(3M)	この列はルーチン・シグニチャーを識別します。
		NULL 可能	オブジェクトがルーチンでない場合は、NULL 値が入ります。

SYSVARIABLES

SYSVARIABLES 表には、各グローバル変数ごとに行が 1 つずつ入ります。

次の表は、SYSVARIABLES 表の列について説明しています。

表 *214. SYSVARIABLES* 表

列名	システム列名	データ・タイプ	説明
VARIABLE_SCHEMA	VARSCHEMA	VARCHAR(128)	変数のスキーマ名。
VARIABLE_NAME	VARNAME	VARCHAR(128)	変数の名前。
SYSTEM_VAR_SCHEMA	SYSVSCHEMA	CHAR(10)	システムのスキーマ名。
SYSTEM_VAR_NAME	SYSVNAME	CHAR(10)	システム変数名。
VARIABLE_OWNER	OWNER	VARCHAR(128)	グローバル変数の所有者の許可 ID。
VARIABLE_DEFINER	DEFINER	VARCHAR(128)	該当の変数を作成したユーザーの名前。
VARIABLE_CREATED	CREATDTS	TIMESTAMP	変数が作成されたときのタイム・スタンプを識別します。
SCOPE	SCOPE	CHAR(1)	変数の有効範囲を示します。 **S** セッション
OWNER_TYPE	OWNER_TYPE	CHAR(1)	変数の所有者を示します。 **U** 所有者は個々のユーザーです。 **S** 変数はシステム所有です。
USER_DEFINED_TYPE_SCHEMA	TYPESCHEMA	VARCHAR(128) NULL 可能	特殊タイプの場合は、データ・タイプのスキーマ。 変数が特殊タイプでない場合は、NULL 値が入ります。
USER_DEFINED_TYPE_NAME	TYPENAME	VARCHAR(128) NULL 可能	特殊タイプの場合は、データ・タイプの名前。 変数が特殊タイプでない場合は、NULL 値が入ります。

表 214. SYSVARIABLES 表 (続き)

列名	システム列名	データ・タイプ	説明
DATA_TYPE	DATA_TYPE	VARCHAR(128)	変数のタイプ: **BIGINT** 大整数 **INTEGER** 長整数 **SMALLINT** 短整数 **DECIMAL** パック 10 進数 **NUMERIC** ゾーン 10 進数 **FLOAT** 浮動小数点数; FLOAT、REAL、または DOUBLE PRECISION **DECFLOAT** 10 進浮動小数点数 **CHARACTER** 固定長文字ストリング **VARCHAR** 可変長文字ストリング **CLOB** 文字ラージ・オブジェクト・ストリング **GRAPHIC** 固定長グラフィック・ストリング **VARGRAPHIC** 可変長グラフィック・ストリング **DBCLOB** 2 バイト文字ラージ・オブジェクト・ストリング **BINARY** 固定長バイナリー・ストリング **VARBINARY** 可変長バイナリー・ストリング **BLOB** バイナリー・ラージ・オブジェクト・ストリング **DATE** 日付 **TIME** 時刻 **TIMESTAMP** タイム・スタンプ **XML** XML

表 214. SYSVARIABLES 表 (続き)

列名	システム列名	データ・タイプ	説明
LENGTH	LENGTH	INTEGER	データ・タイプの長さ属性。ただし、10 進数、数値、または非ゼロ精度 2 進数の変数の場合は、その精度。 **8** バイト BIGINT **4** バイト INTEGER **2** バイト SMALLINT 数の精度 DECIMAL 数の精度 NUMERIC **8** バイト FLOAT、FLOAT(n) (ここで n = 25 から 53)、または DOUBLE PRECISION **4** バイト FLOAT(n) (ここで n = 1 から 24)、または REAL **8** バイト DECFLOAT(16) **16** バイト DECFLOAT(34) ストリングの長さ CHARACTER ストリングの最大長 VARCHAR または CLOB グラフィック・ストリングの長さ GRAPHIC グラフィック・ストリングの最大長 VARGRAPHIC または DBCLOB **2** 進ストリングの長さ BINARY **2** 進ストリングの最大長 VARBINARY または BLOB **4** バイト DATE **3** バイト TIME **((*p*+1)/2) + 7 (*p*** はタイム・スタンプの精度**)** の整数部分 TIMESTAMP **2147483647** バイト XML ソース・タイプと同じ値 DISTINCT
NUMERIC_SCALE	SCALE	INTEGER	数値データの位取り。
		NULL 可能	データ・タイプが 10 進数、数値、または 2 進数でない場合は、NULL 値が入ります。

表 *214. SYSVARIABLES* 表 (続き)

列名	システム列名	データ・タイプ	説明
STORAGE	STORAGE	INTEGER	変数の記憶域所要量: **8** バイト BIGINT **4** バイト INTEGER **2** バイト SMALLINT **(**精度**/2) + 1** DECIMAL 数の精度 NUMERIC **8** バイト FLOAT、FLOAT(n) (ここで n = 25 から 53)、または DOUBLE PRECISION **4** バイト FLOAT(n) (ここで n = 1 から 24)、または REAL **8** バイト DECFLOAT(16) **16** バイト DECFLOAT(34) ストリングの長さ CHAR ストリングの最大長 **+ 2** VARCHAR ストリングの最大長 **+ 29** CLOB ストリングの長さ *** 2** GRAPHIC ストリングの最大長 *** 2 + 2** VARGRAPHIC ストリングの最大長 *** 2 + 29** DBCLOB **2** 進ストリングの長さ BINARY **2** 進ストリングの最大長 **+ 2** VARBINARY ストリングの最大長 **+ 29** BLOB **4** バイト DATE **3** バイト TIME **((***p***+1)/2) + 7 (***p* はタイム・スタンプの精度**)** の整数部分 TIMESTAMP **2147483647 + 29** バイト XML ソース・タイプと同じ値 DISTINCT **注:** この列には、すべてのデータ・タイプの記憶域所要量があります。

表 214. SYSVARIABLES 表 (続き)

列名	システム列名	データ・タイプ	説明
NUMERIC_PRECISION	PRECISION	INTEGER NULL 可能	すべての数値データ・タイプの精度。 **注:** この列では、すべての数値データ・タイプ (10 進浮動小数点数、単精度および倍精度の浮動小数点数を含む) の精度を指定します。NUMERIC_PRECISION_RADIX 列は、この列の値が 2 進数であるか、または 10 進数であるかを示します。 データ・タイプが数値でない場合は、NULL 値が入ります。
CCSID	CCSID	INTEGER NULL 可能	CHAR、VARCHAR、CLOB、DATE、TIME、TIMESTAMP、GRAPHIC、VARGRAPHIC、DBCLOB、XML、および DATALINK データ・タイプの CCSID の値。 データ・タイプが数値の場合は、NULL 値が入ります。
CHARACTER_MAXIMUM_LENGTH	CHARLEN	INTEGER NULL 可能	バイナリー・ストリング、文字ストリング、グラフィック・ストリングの各データ・タイプと XML データ・タイプの最大長。 データ・タイプがストリングまたは XML でない場合は、NULL 値が入ります。
CHARACTER_OCTET_LENGTH	CHARBYTE	INTEGER NULL 可能	バイナリー・ストリング、文字ストリング、グラフィック・ストリングの各データ・タイプと XML データ・タイプのバイト数。 データ・タイプがストリングまたは XML でない場合は、NULL 値が入ります。
NUMERIC_PRECISION_RADIX	RADIX	INTEGER NULL 可能	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 データ・タイプが数値でない場合は、NULL 値が入ります。
DATETIME_PRECISION	DATPRC	INTEGER NULL 可能	日付、時刻、またはタイム・スタンプの小数部分。 **0** データ・タイプが DATE および TIME の場合 **0-12** データ・タイプが TIMESTAMP の場合 (小数秒) データ・タイプが日付、時刻、またはタイム・スタンプでない場合は、NULL 値が入ります。
DEFAULT	DEFAULT	DBCLOB(2M) CCSID 1200	グローバル変数が最初に参照されるときにグローバル変数の初期値を計算するための式。
IASP_NUMBER	IASPNUMBER	SMALLINT	変数の独立補助記憶域プール (IASP) 番号を指定します。
VARIABLE_TEXT	LABEL	VARGRAPHIC(50) CCSID 1200 NULL 可能	LABEL ステートメント (タイプ・テキスト) で指定された文字ストリング。 タイプにテキストがない場合は、NULL 値が入ります。

表 214. *SYSVARIABLES* 表 (続き)

列名	システム列名	データ・タイプ	説明
LONG_COMMENT	REMARKS	VARGRAPHIC(2000) CCSID 1200 NULL 可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、NULL 値が入ります。
DEFAULT_SCHEMA	QUALIFIER	VARCHAR(128) NULL 可能	非修飾の表とビューの修飾子。
SQL_PATH	SQL_PATH	VARCHAR(3483)	パスを示します。
ROUNDING_MODE	DECFLTRND	CHAR(1) NULL 可能	DECFLOAT 丸めモードを示します。 **C** ROUND_CEILING **D** ROUND_DOWN **F** ROUND_FLOOR **G** ROUND_HALF_DOWN **E** ROUND_HALF_EVEN **H** ROUND_HALF_UP **U** ROUND_UP
READONLY	READONLY	CHAR(1) NULL 可能	変数が変更可能なのか、読み取り専用なのかを示します。 **N** 読み取り専用ではない **S** グローバル変数はデータベース・マネージャーによって保守されるため、読み取り専用。

SYSVIEWDEP

SYSVIEWDEP ビューは、表に対するビューの従属関係 (SQL カタログのビューを含む) を記録します。

次の表は、SYSVIEWDEP ビューの列について説明しています。

表 *215. SYSVIEWDEP* ビュー

列名	システム列名	データ・タイプ	説明
VIEW_NAME	DNAME	VARCHAR(128)	ビューの名前。 SQL ビュー名が存在する場合は、その SQL ビュー名です。存在しない場合は、システム・ビュー名です。
VIEW_OWNER	DCREATOR	VARCHAR(128)	ビューの所有者
OBJECT_NAME	ONAME	VARCHAR(128)	該当のビューが従属しているオブジェクトの名前。
OBJECT_SCHEMA	OSCHEMA	VARCHAR(128)	該当のビューが従属しているオブジェクトが入っている SQL スキーマの名前。
OBJECT_TYPE	OTYPE	CHAR(24)	該当のビューの基となったオブジェクトのタイプ: **FUNCTION** 関数 **MATERIALIZED QUERY TABLE** オブジェクトはマテリアライズ照会表です。 **TABLE** 表 **TYPE** 特殊タイプ **VARIABLE** オブジェクトは変数です。 **VIEW** ビュー
VIEW_SCHEMA	DDBNAME	VARCHAR(128)	ビューのスキーマ名
SYSTEM_VIEW_NAME	SYS_VNAME	CHAR(10)	システム・ビュー名
SYSTEM_VIEW_SCHEMA	SYS_VDNAME	CHAR(10)	システム・ビュー・スキーマ
SYSTEM_TABLE_NAME	SYS_TNAME	CHAR(10) NULL 可能	システム表名。 オブジェクトが関数または特殊タイプである場合は、NULL 値が入ります。
SYSTEM_TABLE_SCHEMA	SYS_DNAME	CHAR(10) NULL 可能	システム表スキーマ。 オブジェクトが関数または特殊タイプである場合は、NULL 値が入ります。
TABLE_NAME	BNAME	VARCHAR(128) NULL 可能	該当のビューが従属している表またはビューの名前。 SQL ビュー名が存在する場合は、その SQL ビュー名です。存在しない場合は、システム・ビュー名です。 オブジェクトが関数または特殊タイプである場合は、NULL 値が入ります。
TABLE_OWNER	BCREATOR	VARCHAR(128) NULL 可能	該当のビューが従属している表またはビューの所有者。 オブジェクトが関数または特殊タイプである場合は、NULL 値が入ります。
TABLE_SCHEMA	BDBNAME	VARCHAR(128) NULL 可能	該当のビューが従属している表、またはビューが入っている SQL のスキーマの名前。 オブジェクトが関数または特殊タイプである場合は、NULL 値が入ります。

表 *215. SYSVIEWDEP* ビュー (続き)

列名	システム列名	データ・タイプ	説明
TABLE_TYPE	BTYPE	CHAR(1) NULL 可能	該当のビューの基となったオブジェクトのタイプ: **T** 表 **P** 物理ファイル **M** マテリアライズ照会表 **V** ビュー **L** 論理ファイル オブジェクトが関数または特殊タイプである場合は、NULL 値が入ります。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。
PARM_SIGNATURE	SIGNATURE	BLOB(3M) NULL 可能	この列はルーチン・シグニチャーを識別します。 オブジェクトがルーチンでない場合は、NULL 値が入ります。

SYSVIEWS

SYSVIEWS ビューには、SQL のスキーマにある各ビュー (SQL カタログのビューを含む) ごとに、行が 1 つずつ入ります。

次の表は、SYSVIEWS ビューの列について説明しています。

表 *216. SYSVIEWS* ビュー

列名	システム列名	データ・タイプ	説明
TABLE_NAME	NAME	VARCHAR(128)	ビューの名前。 SQL ビュー名が存在する場合は、その SQL ビュー名です。存在しない場合は、システム・ビュー名です。
VIEW_OWNER	CREATOR	VARCHAR(128)	ビューの所有者
SEQNO	SEQNO	INTEGER	該当の行の順序番号。常に 1 になります。
CHECK_OPTION	CHECK	CHAR(1)	該当のビューに対して使用された検査オプション **N** 検査オプションは指定されませんでした **Y** ローカル・オプションが指定されました **C** カスケード・オプションが指定されました
VIEW_DEFINITION	TEXT	VARGRAPHIC(5000) CCSID 1200 NULL 可能	CREATE VIEW ステートメントの QUERY 式の部分。 切り捨てなければビュー定義を列に収容できない場合は、NULL 値が入ります。
IS_UPDATABLE	UPDATES	CHAR(1)	ビューが更新可能かどうかを指定します。 **Y** 更新可能なビューです。 **N** 更新不能のビューです。
TABLE_SCHEMA	DBNAME	VARCHAR(128)	該当のビューが入っている SQL のスキーマの名前。
SYSTEM_VIEW_NAME	SYS_VNAME	CHAR(10)	システム・ビュー名
SYSTEM_VIEW_SCHEMA	SYS_VDNAME	CHAR(10)	システム・ビュー・スキーマ名
IS_INSERTABLE_INTO	INSERTABLE	VARCHAR(3)	ビューで INSERT を使用できるかどうかを識別します。 **NO** このビューでは INSERT は使用できません。 **YES** このビューでは INSERT を使用できます。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。
IS_DELETABLE	DELETES	CHAR(1) NULL 可能	ビューが削除可能かどうかを指定します。 **Y** 削除可能なビューです。 **N** 読み取り専用のビューです。
VIEW_DEFINER	DEFINER	VARCHAR(128)	該当のビューを定義したユーザーの名前。

表 *216. SYSVIEWS* ビュー *(*続き*)*

列名	システム列名	データ・タイプ	説明
ROUNDING_MODE	DECFLTRND	CHAR(1) NULL 可能	ビューの DECFLOAT 丸めモードを示します。 **C** ROUND_CEILING **D** ROUND_DOWN **F** ROUND_FLOOR **G** ROUND_HALF_DOWN **E** ROUND_HALF_EVEN **H** ROUND_HALF_UP **U** ROUND_UP DECFLOAT 列、関数、または定数を参照する式がビューにない場合は、NULL 値が入ります。

SYSXSROBJECTAUTH

SYSXSROBJECTAUTH ビューには、XML スキーマに対して認可された各特権ごとに、行が 1 つずつ入ります。このカタログ・ビューを使用して、特定ユーザーが特定の XML スキーマに対する権限を持っているかどうかを判別することはできないので、注意してください。なぜなら、XML スキーマを使用するための特権は、グループ・ユーザー・プロファイルまたは特殊権限 (*ALLOBJ など) を通じて獲得できるからです。

次の表は、SYSXSROBJECTAUTH ビューの列について説明しています。

表 *217. SYSXSROBJECTAUTH* ビュー

列名	システム列名	データ・タイプ	説明
GRANTOR	GRANTOR	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
GRANTEE	GRANTEE	VARCHAR(128)	特権を認可する対象のユーザー・プロファイル。
XSROBJECTSCHEMA	XSRSCHEMA	VARCHAR(128)	スキーマの名前
XSROBJECTNAME	XSRNAME	VARCHAR(128)	XML スキーマの名前。
PRIVILEGE_TYPE	PRIVTYPE	VARCHAR(10)	認可される特権 : **ALTER** XML スキーマを変更する特権。 **USAGE** XML スキーマを使用する特権。
IS_GRANTABLE	GRANTABLE	VARCHAR(3)	特権を他のユーザーに認可できるかどうかを示します。 **NO** 特権は認可できません。 **YES** 特権を認可できます。
AUTHORIZATION_LIST	AUTL	VARCHAR(10) NULL 可能	特権が権限を介して認可されている場合、権限リストの名前が入ります。 特権が権限リストを介して認可されるのではない場合は、NULL 値が入ります。
SYSTEM_XSR_SCHEMA	SYSXSCHEMA	CHAR(10)	スキーマのシステム名
SYSTEM_XSR_NAME	SYSXNAME	CHAR(10)	XML スキーマのシステム名

XSRANNOTATIONINFO

XSRANNOTATIONINFO 表の各行は、XML スキーマのそれぞれの注釈に対応しており、注釈に関する表と列の情報を各行で記録しています。

次の表は、XSRANNOTATIONINFO 表の列について説明しています。

表 *218. XSRANNOTATIONINFO* 表

列名	システム列名	データ・タイプ	説明
XSROBJECTID	OBJECTID	BIGINT	XML スキーマの内部 ID。
ANNOTATION_ID	ANNOID	INTEGER	XML スキーマの注釈の内部 ID。
TABLE_SCHEMA	TABSCHEMA	VARCHAR(128)	データを挿入する表のスキーマ。
TABLE_NAME	TABNAME	VARCHAR(128)	データを挿入する表。
ROWSET	ROWSET	VARCHAR(1000)	この注釈の行セットの名前。
COLUMN_NAME	COLNAME	VARCHAR(128)	この注釈の列の名前。
DATA_TYPE	COLTYPE	INTEGER	この注釈の列のデータ・タイプ。
INSTANCE_TYPE	INSTTYPE	INTEGER	パーサーが分解時に用意するデータのタイプ。 **2** 10 進数 **4** 長整数 **5** 整数 **6** 短整数 **16** ストリング **30** 日付/時刻 **41** float **42** double
TRUNCATE	TRUNCATE	INTEGER	データの切り捨てが可能かどうかを示す標識。 **0** データを切り捨てることはできません。 **1** データを切り捨てることができます。
EXPRESSION	EXPRESSION	VARCHAR(1024) NULL 可能	Db2 が挿入時にデータに適用する式。
CONDITION	CONDITION	VARCHAR(1024) NULL 可能	Db2 がデータを挿入する前に適用する条件。
CAST_EXPRESSION	CASTEXP	VARCHAR(20) NULL 可能	Db2 が分解時に列データを挿入するときに適用するキャスト式。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。
ROWSET_ORDER	ORDER	INTEGER	RowSet オーダー・シーケンス。デフォルトは 1 です。

XSROBJECTCOMPONENTS

XSROBJECTCOMPONENTS 表には、XML スキーマの各コンポーネント (文書) ごとに行が 1 つずつ入ります。

次の表は、XSROBJECTCOMPONENTS 表の列について説明しています。

表 *219. XSROBJECTCOMPONENTS* 表

列名	システム列名	データ・タイプ	説明
XSRCOMPONENTID	COMPID	BIGINT	XML スキーマ文書の内部 ID。
TARGETNAMESPACE	TGTNAMESPC	VARCHAR(1000) NULL 可能	ターゲット名前空間のストリング ID。
SCHEMALOCATION	SCHEMALOC	VARCHAR(1000) NULL 可能	スキーマ・ロケーションのストリング ID。
COMPONENT	COMPONENT	BLOB(30M) NULL 可能	XML スキーマ文書の内容。
PROPERTIES	PROPERTIES	BLOB(5M) NULL 可能	XML スキーマ文書の追加の文書プロパティー情報。
CREATED	CREATEDTS	TIMESTAMP	XML スキーマに対して XSR_REGISTER ストアード・プロシージャーが実行された時刻。
STATUS	STATUS	CHAR(1)	XML スキーマの登録状況。 **C** 完了 **I** 未完了
IASP_NUMBER	IASPNUMBER	SMALLINT	カタログ表が配置されている IASP を示します。

XSROBJECTHIERARCHIES

XSROBJECTHIERARCHIES 表には、XML スキーマの各コンポーネント (文書) ごとに行が 1 つずつ入り、XML スキーマ文書の階層関係を記録します。

次の表は、XSROBJECTHIERARCHIES 表の列について説明しています。

表 *220. XSROBJECTHIERARCHIES* 表

列名	システム列名	データ・タイプ	説明
XSROBJECTID	OBJECTID	BIGINT	XML スキーマの内部 ID。
XSRCOMPONENTID	COMPID	BIGINT	XML スキーマ文書の内部 ID。
HTYPE	HTYPE	CHAR(1)	階層タイプ: **D** 文書 **P** 1 次文書
TARGETNAMESPACE	TGTNAMESPC	VARCHAR(1000) NULL 可能	ターゲット名前空間のストリング ID。
SCHEMALOCATION	SCHEMALOC	VARCHAR(1000) NULL 可能	スキーマ・ロケーションのストリング ID。
IASP_NUMBER	IASPNUMBER	SMALLINT	独立補助記憶域プール (IASP) 番号を指定します。

XSROBJECTS

XSROBJECTS 表には、登録済みの各 XML スキーマごとに行が 1 つずつ入ります。

次の表は、XSROBJECTS 表の列について説明しています。

表 *221. XSROBJECTS* 表

列名	システム列名	データ・タイプ	説明
XSROBJECTID	OBJECTID	BIGINT	XML スキーマの内部 ID。
XSROBJECTSCHEMA	XSRSCHEMA	VARCHAR(128)	XML スキーマ名の修飾子。
XSROBJECTNAME	XSRNAME	VARCHAR(128)	XML スキーマの名前。
SYSTEM_XSR_SCHEMA	SYSXSCHEMA	CHAR(10)	XML スキーマの修飾子に対応するシステム・ライブラリー名。
SYSTEM_XSR_NAME	SYSXNAME	CHAR(10)	XML スキーマとの間に相関関係が存在する XSROBJECT のシステム名。
TARGETNAMESPACE	TGTNAMESPC	VARCHAR(1000) NULL 可能	ターゲット名前空間のストリング ID。
SCHEMALOCATION	SCHEMALOC	VARCHAR(1000) NULL 可能	スキーマ・ロケーションのストリング ID。
XSR_OBJECT_TEXT	LABEL	VARGRAPHIC(50) CCSID 1200 NULL 可能	LABEL ステートメントで指定された文字ストリング。 ラベルがない場合は、NULL 値が入ります。
GRAMMAR	GRAMMAR	BLOB(250M) NULL 可能	XML_COMPLETE の結果として得られる XML スキーマの内部バイナリー表記。
PROPERTIES	PROPERTIES	BLOB(5M) NULL 可能	XML スキーマ文書の追加の文書プロパティー情報。
XSR_SCHEMA_DEFINER	DEFINER	VARCHAR(128)	XML スキーマが作成された際の許可 ID。
XSR_SCHEMA_OWNER	OWNER	VARCHAR(128)	XML スキーマを所有する権限 ID。
XSR_SCHEMA_CREATED	CREATEDTS	TIMESTAMP	XML スキーマ文書が登録された時刻。
STATUS	STATUS	CHAR(1)	XML スキーマの登録状況。 **C** 完了 **I** 未完了
DECOMPOSITION	DECOMP	CHAR(1) NULL 可能	XSR オブジェクトの分解状況を示します。 **Y** 使用可能 **N** 使用可能ではない
VERSION	VERSION	VARCHAR(128) NULL 可能	分解時に使用されたバージョンを示します。
IASP_NUMBER	IASPNUMBER	SMALLINT	カタログ表が配置されている IASP を示します。
LONG_COMMENT	REMARKS	VARGRAPHIC(2000) CCSID 1200 NULL 可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、NULL 値が入ります。

ODBC および JDBC のカタログ・ビュー

カタログには、SYSIBM ライブラリー内にあるビューおよび表が含まれます。

ビュー名	説明
2104 ページの『SQLCOLPRIVILEGES』	列に対して認可された特権についての情報
2105 ページの『SQLCOLUMNS』	列属性についての情報
2112 ページの『SQLFOREIGNKEYS』	外部キーについての情報
2113 ページの『SQLFUNCTIONCOLS』	関数仮パラメーターについての情報
2118 ページの『SQLFUNCTIONS』	関数についての情報
2119 ページの『SQLPRIMARYKEYS』	基本キーについての情報
2120 ページの『SQLPROCEDURECOLS』	プロシージャー・パラメーターについての情報
2127 ページの『SQLPROCEDURES』	プロシージャーについての情報
2128 ページの『SQLSCHEMAS』	スキーマについての情報
2129 ページの『SQLSPECIALCOLUMNS』	行を一意的に識別するために使用できる表の列についての情報
2133 ページの『SQLSTATISTICS』	表についての統計情報
2134 ページの『SQLTABLEPRIVILEGES』	表に認可された特権についての情報
2135 ページの『SQLTABLES』	表についての情報
2136 ページの『SQLTYPEINFO』	表のタイプについての情報
2143 ページの『SQLUDTS』	組み込みのデータ・タイプおよび特殊タイプについての情報

SQLCOLPRIVILEGES

SQLCOLPRIVILEGES ビューには、列に対して認可された特権、または列の表に対して認可された特権について、それぞれ 1 つの行が含まれます。このカタログ・ビューを使用して、特定ユーザーが特定の列に対する権限を持っているかどうかを判別することはできないので、注意してください。なぜなら、列を使用するための特権は、グループ・ユーザー・プロファイルまたは特殊権限 (*ALLOBJ など) を通じて獲得できるからです。

次の表は、ビューの列について説明しています。

表 222. *SQLCOLPRIVILEGES* ビュー

列名	データ・タイプ	説明
TABLE_CAT	VARCHAR(128)	リレーショナル・データベース名。
TABLE_SCHEM	VARCHAR(128)	該当の表が入っている SQL のスキーマの名前。
TABLE_NAME	VARCHAR(128)	表名
COLUMN_NAME	VARCHAR(128)	列名
GRANTOR	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
GRANTEE	VARCHAR(128)	特権を認可する対象のユーザー・プロファイル。
PRIVILEGE	VARCHAR(10)	認可される特権 : **UPDATE** 列を更新する特権。 **REFERENCES** 参照制約モードで列を参照する特権。
IS_GRANTABLE	VARCHAR(3)	特権を他のユーザーに認可できるかどうかを示します。 **NO** 特権は認可できません。 **YES** 特権を認可できます。
DBNAME	VARCHAR(8) NULL 可能	予約済み。列には NULL 値が入ります。

SQLCOLUMNS

SQLCOLUMNS ビューには、表、ビュー、または別名の中の各列ごとに、行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *223. SQLCOLUMNS* ビュー

列名	データ・タイプ	説明
TABLE_CAT	VARCHAR(128)	リレーショナル・データベース名。
TABLE_SCHEM	VARCHAR(128)	該当の表が入っている SQL のスキーマの名前。
TABLE_NAME	VARCHAR(128)	表名
COLUMN_NAME	VARCHAR(128)	列名
DATA_TYPE	SMALLINT	列のデータ・タイプ。 **–5** BIGINT **4** INTEGER **5** SMALLINT **3** DECIMAL **2** NUMERIC **8** DOUBLE PRECISION **7** REAL **-360** DECFLOAT **1** CHARACTER **–2** CHARACTER FOR BIT DATA または BINARY **12** VARCHAR **–3** VARCHAR FOR BIT DATA または VARBINARY **-99** CLOB **-95** GRAPHIC **–96** VARGRAPHIC **–350** DBCLOB **–8** NCHAR **–9** NVARCHAR **–10** NCLOB **-98** BLOB **91** DATE **92** TIME **93** TIMESTAMP **70** DATALINK **–100** ROWID **-370** XML **17** DISTINCT

表 223. SQLCOLUMNS ビュー (続き)

列名	データ・タイプ	説明
TYPE_NAME	VARCHAR(261)	列のデータ・タイプの名前。 **BIGINT** BIGINT **INTeger** INTEGER **SMALLINT** SMALLINT **DECIMAL** DECIMAL **NUMERIC** NUMERIC **FLOAT** DOUBLE PRECISION **REAL** REAL **DECFLOAT** DECFLOAT **CHARacter** CHARACTER **CHARacter FOR BIT DATA** CHARACTER FOR BIT DATA **VARCHAR** VARCHAR **VARCHAR FOR BIT DATA** VARCHAR FOR BIT DATA **CLOB** CLOB **GRAPHIC** GRAPHIC **VARGRAPHIC** VARGRAPHIC **DBCLOB** DBCLOB **NCHAR** NCHAR **NVARCHAR** NVARCHAR **NCLOB** NCLOB **BINARY** BINARY **VARBINARY** VARBINARY **BLOB** BLOB **DATE** DATE **TIME** TIME **TIMESTAMP** TIMESTAMP **DATALINK** DATALINK **ROWID** ROWID **XML** XML 修飾タイプ名 DISTINCT

表 223. *SQLCOLUMNS* ビュー (続き)

列名	データ・タイプ	説明
COLUMN_SIZE	INTEGER	列の長さです。
BUFFER_LENGTH	INTEGER	バッファー内の列の長さを示します。
DECIMAL_DIGITS	SMALLINT	数値列の桁数を示します。
	NULL 可能	オブジェクトが数値またはタイム・スタンプでない場合は、NULL 値が入ります。
NUM_PREC_RADIX	SMALLINT	数値列の基数を示します。
	NULL 可能	オブジェクトが数値でない場合は、NULL 値が入ります。
NULLABLE	SMALLINT	列に NULL 値を入れることができるかどうかを示します。 **0** この列では NULL は許されません。 **1** この列では NULL が許されます。
REMARKS	VARCHAR(2000)	COMMENT ステートメントで指定された文字ストリング。
	NULL 可能	詳細コメントがない場合は、NULL 値が入ります。
COLUMN_DEF	VARCHAR(2000)	列のデフォルト値。
	NULL 可能	デフォルト値がない場合は、NULL 値が入ります。

表 223. *SQLCOLUMNS* ビュー (続き)

列名	データ・タイプ	説明
SQL_DATA_TYPE	SMALLINT	列の SQL データ・タイプを示します。 **–5** BIGINT **4** INTEGER **5** SMALLINT **3** DECIMAL **2** NUMERIC **8** DOUBLE PRECISION **7** REAL **-360** DECFLOAT **1** CHARACTER **–2** CHARACTER FOR BIT DATA または BINARY **12** VARCHAR **–3** VARCHAR FOR BIT DATA または VARBINARY **-99** CLOB **-95** GRAPHIC **–96** VARGRAPHIC **–350** DBCLOB **–8** NCHAR **–9** NVARCHAR **–10** NCLOB **-98** BLOB **9** DATE **9** TIME **9** TIMESTAMP **70** DATALINK **–100** ROWID **-370** XML **17** DISTINCT
SQL_DATETIME_SUB	SMALLINT NULL 可能	データ・タイプの日時サブタイプ: **1** DATE **2** TIME **3** TIMESTAMP 列が日時データ・タイプでない場合は、NULL 値が入ります。
CHAR_OCTET_LENGTH	INTEGER NULL 可能	列の長さをバイト数で示します。 列がストリングでない場合は、NULL 値が入ります。
ORDINAL_POSITION	INTEGER	表内の列の順序位置を示します。
IS_NULLABLE	VARCHAR(3)	列に NULL 値を入れることができるかどうかを示します。 **NO** 列はNULL 可能ではありません。 **YES** 列はNULL 可能です。

表 223. *SQLCOLUMNS* ビュー (続き)

列名	データ・タイプ	説明
JDBC_DATA_TYPE	SMALLINT	列の JDBC データ・タイプを示します。 **–5** BIGINT **4** INTEGER **5** SMALLINT **3** DECIMAL **2** NUMERIC **8** DOUBLE PRECISION **7** REAL **1111** DECFLOAT **1** CHARACTER または GRAPHIC **-15** NCHAR **–2** CHARACTER FOR BIT DATA または BINARY **12** VARCHAR または VARGRAPHIC **-9** NVARCHAR **–3** VARCHAR FOR BIT DATA または VARBINARY **2005** CLOB または DBCLOB **2011** NCLOB **2004** BLOB **91** DATE **92** TIME **93** TIMESTAMP **70** DATALINK **-8** ROWID **2009** XML **2001** DISTINCT
SCOPE_CATALOG	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
SCOPE_SCHEMA	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
SCOPE_TABLE	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
SOURCE_DATA_TYPE	SMALLINT NULL 可能	列のデータ・タイプが特殊データ・タイプである場合は、ソース・データ・タイプ。値については、JDBC_DATA_TYPE を参照してください。 データ・タイプが特殊タイプでない場合は、NULL 値が入ります。
DBNAME	VARCHAR(8) NULL 可能	予約済み。 NULL 値が入ります。

表 223. *SQLCOLUMNS* ビュー (続き)

列名	データ・タイプ	説明
PSEUDO_COLUMN	SMALLINT	これが、ROWID 列、ID 列、行変更タイム・スタンプ列、行開始列、行終了列、トランザクション開始 ID 列、または生成式列であるかどうかを示します。 **1** この列は、ROWID 列、ID 列、行変更タイム・スタンプ列、行開始列、行終了列、トランザクション開始 ID 列、および生成式列ではありません。 **2** この列は、ROWID 列、ID 列、行変更タイム・スタンプ列、行開始列、行終了列、トランザクション開始 ID 列、または生成式列です。
COLUMN_TEXT	VARCHAR(50)	列のテキスト。
	NULL 可能	列テキストがない場合は、NULL 値が入ります。
SYSTEM_COLUMN_NAME	CHAR(10)	列のシステム名。
I_DATA_TYPE	SMALLINT	列の IBM i CLI データ・タイプを示します。 **19** BIGINT **4** INTEGER **5** SMALLINT **3** DECIMAL **2** NUMERIC **8** DOUBLE PRECISION **7** REAL **-360** DECFLOAT **1** CHARACTER **–2** CHARACTER FOR BIT DATA または BINARY **12** VARCHAR **–3** VARCHAR FOR BIT DATA または VARBINARY **14** CLOB **95** GRAPHIC または NCHAR **96** VARGRAPHIC または NVARCHAR **15** DBCLOB または NCLOB **13** BLOB **91** DATE **92** TIME **93** TIMESTAMP **16** DATALINK **1111** ROWID **-370** XML **2001** DISTINCT
HIDDEN	CHAR(1)	列が暗黙の列リストに含まれるかどうかを指定します。 **P** 部分的に隠されている。 **N** 隠されていない。

表 223. *SQLCOLUMNS* ビュー (続き)

列名	データ・タイプ	説明
HAS_DEFAULT	CHAR(1)	列がデフォルト値を持つ (DEFAULT 文節または NULL 使用可能) かどうか: **N** いいえ **Y** はい **A** 列は、ROWID データ・タイプおよび GENERATED ALWAYS 属性を持ちます。 **D** 列は、ROWID データ・タイプおよび GENERATED BY DEFAULT 属性を持ちます。 **E** 列は、FOR EACH ROW ON UPDATE 属性および GENERATED ALWAYS 属性により定義されます。 **F** 列は、FOR EACH ROW ON UPDATE 属性および GENERATED BY DEFAULT 属性により定義されます。 **I** 列は、AS IDENTITY 属性および GENERATED ALWAYS 属性により定義されます。 **J** 列は、AS IDENTITY 属性および GENERATED 属性により定義されます。 **Q** 列は GENERATED AS ROW BEGIN 属性を使って定義される。 **R** 列は GENERATED AS ROW END 属性を使って定義される。 **X** 列は GENERATED AS TRANSACTION START ID 属性を使って定義される。 **a** 列は、特殊レジスターを使用して生成された式として定義されます。 **c** 列は、グローバル変数を使用して生成された式として定義されます。 **d** 列は DATA CHANGE OPERATION を使用して生成された式として定義されます。 列がビューに関するものである場合、N が戻されます。
SOURCE_TYPE_NAME	VARCHAR(128) NULL 可能	列データ・タイプがユーザー定義タイプである場合、そのソース・タイプの組み込みデータ・タイプ名。 列のデータ・タイプがユーザー定義タイプでない場合は、NULL 値が入ります。
SOURCE_SQL_DATA_TYPE	SMALLINT NULL 可能	列データ・タイプがユーザー定義タイプである場合、そのソース・タイプの組み込み SQL_DATA_TYPE。値については、SQL_DATA_TYPE を参照してください。 列のデータ・タイプがユーザー定義タイプでない場合は、NULL 値が入ります。
SOURCE_JDBC_DATA_TYPE	SMALLINT NULL 可能	列データ・タイプがユーザー定義タイプである場合、そのソース・タイプの組み込み JDBC_DATA_TYPE。値については、JDBC_DATA_TYPE を参照してください。 列のデータ・タイプがユーザー定義タイプでない場合は、NULL 値が入ります。

SQLFOREIGNKEYS

SQLFOREIGNKEYS ビューには、表の各参照制約キーごとに、行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *224. SQLFOREIGNKEYS* ビュー

列名	データ・タイプ	説明
PKTABLE_CAT	VARCHAR(128)	リレーショナル・データベース名
PKTABLE_SCHEM	VARCHAR(128)	親表が入っている SQL スキーマの名前。
PKTABLE_NAME	VARCHAR(128)	親表の名前。
PKCOLUMN_NAME	VARCHAR(128)	親キーの列名。
FKTABLE_CAT	VARCHAR(128)	リレーショナル・データベース名
FKTABLE_SCHEM	VARCHAR(128)	参照制約の従属表が入っている SQL スキーマの名前。
FKTABLE_NAME	VARCHAR(128)	参照制約の従属表名。
FKCOLUMN_NAME	VARCHAR(128)	従属キー名。
KEY_SEQ	SMALLINT	キー内における列の位置。
UPDATE_RULE	SMALLINT	UPDATE の規則。 **1** RESTRICT **3** NO ACTION
DELETE_RULE	SMALLINT	削除規則: **0** CASCADE **1** RESTRICT **2** SET NULL **3** NO ACTION **4** SET DEFAULT
FK_NAME	VARCHAR(128)	参照制約の名前。
PK_NAME	VARCHAR(128)	固有制約の名前。
DEFERRABILITY	SMALLINT	制約の検査が据え置きできるかどうかを示します。 常に 7 になります。
UNIQUE_OR_PRIMARY	CHAR(7)	親制約のタイプを示します。 **PRIMARY** 親制約は基本キーです。 **UNIQUE** 親制約はユニーク制約です。

SQLFUNCTIONCOLS

SQLFUNCTIONCOLS ビューには、関数の各パラメーターごとに、行が 1 つずつ入ります。スカラー関数の結果および表関数の結果列も返されます。

次の表は、ビューの列について説明しています。

表 *225. SQLFUNCTIONCOLS* ビュー

列名	データ・タイプ	説明
FUNCTION_CAT	VARCHAR(128)	リレーショナル・データベース名
FUNCTION_SCHEM	VARCHAR(128)	関数インスタンスのスキーマ名。
FUNCTION_NAME	VARCHAR(128)	関数インスタンスの名前。
COLUMN_NAME	VARCHAR(128)	関数仮パラメーターの名前。
	NULL 可能	パラメーターに名前がない場合は、NULL 値が入ります。
COLUMN_TYPE	SMALLINT	パラメーターのタイプ:
		1 IN
		4 RETURN 値
		5 表関数の結果列
DATA_TYPE	SMALLINT	パラメーターのデータ・タイプ。
		–5 BIGINT
		4 INTEGER
		5 SMALLINT
		3 DECIMAL
		2 NUMERIC
		8 DOUBLE PRECISION
		7 REAL
		-360 DECFLOAT
		1 CHARACTER
		–2 CHARACTER FOR BIT DATA または BINARY
		12 VARCHAR
		–3 VARCHAR FOR BIT DATA または VARBINARY
		-99 CLOB
		-95 GRAPHIC または
		–96 VARGRAPHIC
		–350 DBCLOB
		–8 NCHAR
		–9 NVARCHAR
		–10 NCLOB
		-98 BLOB
		91 DATE
		92 TIME
		93 TIMESTAMP
		70 DATALINK
		–100 ROWID
		-370 XML
		17 DISTINCT
		50 ARRAY

表 225. SQLFUNCTIONCOLS ビュー (続き)

列名	データ・タイプ	説明
TYPE_NAME	VARCHAR(261)	パラメーターのデータ・タイプの名前。 **BIGINT** BIGINT **INTeger** INTEGER **SMALLINT** SMALLINT **DECIMAL** DECIMAL **NUMERIC** NUMERIC **FLOAT** DOUBLE PRECISION **REAL** REAL **DECFLOAT** DECFLOAT **CHARacter** CHARACTER **CHARacter FOR BIT DATA** CHARACTER FOR BIT DATA **VARCHAR** VARCHAR **VARCHAR FOR BIT DATA** VARCHAR FOR BIT DATA **CLOB** CLOB **GRAPHIC** GRAPHIC **VARGRAPHIC** VARGRAPHIC **DBCLOB** DBCLOB **NCHAR** NCHAR **NVARCHAR** NVARCHAR **NCLOB** NCLOB **BINARY** BINARY **VARBINARY** VARBINARY **BLOB** BLOB **DATE** DATE **TIME** TIME **TIMESTAMP** TIMESTAMP **DATALINK** DATALINK **ROWID** ROWID **XML** XML 修飾タイプ名 DISTINCT 配列タイプ名 ARRAY
COLUMN_SIZE	INTEGER	パラメーターの長さ。
BUFFER_LENGTH	INTEGER	バッファー内のパラメーターの長さを示します。
DECIMAL_DIGITS	SMALLINT	数値データまたは日時データの位取り。
	NULL 可能	パラメーターが 10 進数、数値、2 進数、時刻、またはタイム・スタンプでない場合は、NULL 値が入ります。
NUM_PREC_RADIX	SMALLINT	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。
	NULL 可能	**2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 パラメーターが数値パラメーターでない場合は、NULL 値が入ります。
NULLABLE	SMALLINT	パラメーターがNULL 可能かどうかを示します。 **0** パラメーターに NULL は許されません。 **1** パラメーターに NULL が許されます。

表 225. *SQLFUNCTIONCOLS* ビュー (続き)

列名	データ・タイプ	説明
REMARKS	VARGRAPHIC(2000) CCSID 1200 NULL 可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、NULL 値が入ります。
COLUMN_DEF	DBCLOB(64K) CCSID 1200 NULL 可能	パラメーターのデフォルト値が存在する場合、パラメーターのデフォルト値を計算するために使用される式ストリング。デフォルト値がヌル値の場合、式ストリングはキーワード NULL です。パラメーターにデフォルトがない場合は、NULL 値が入ります。
SQL_DATA_TYPE	SMALLINT	パラメーターの SQL データ・タイプ。 **–5** BIGINT **4** INTEGER **5** SMALLINT **3** DECIMAL **2** NUMERIC **8** DOUBLE PRECISION **7** REAL **-360** DECFLOAT **1** CHARACTER **–2** CHARACTER FOR BIT DATA または BINARY **12** VARCHAR **–3** VARCHAR FOR BIT DATA または VARBINARY **-99** CLOB **-95** GRAPHIC **–96** VARGRAPHIC **–350** DBCLOB **–8** NCHAR **–9** NVARCHAR **–10** NCLOB **-98** BLOB **9** DATE **9** TIME **9** TIMESTAMP **70** DATALINK **–100** ROWID **-370** XML **17** DISTINCT **50** ARRAY
SQL_DATETIME_SUB	SMALLINT NULL 可能	パラメーターの日時サブタイプ。 **1** DATE **2** TIME **3** TIMESTAMP データ・タイプが日時データ・タイプでない場合は、NULL 値が入ります。
CHAR_OCTET_LENGTH	INTEGER NULL 可能	パラメーターの長さを文字数で示します。 列がストリングでない場合は、NULL 値が入ります。
ORDINAL_POSITION	INTEGER	パラメーター・リストにおける該当のパラメーターの数値位置 (左から右への順序)。 スカラー関数の場合、関数の結果は値 0 になります。 表関数の場合は、結果列には 1 (左端の結果列) から *n* (*n* 番目の結果列) までの番号が付きます。
IS_NULLABLE	VARCHAR(3)	パラメーターがNULL 可能かどうかを示します。 **NO** パラメーターに NULL は許されません。 **YES** パラメーターに NULL が許されます。
SPECIFIC_NAME	VARCHAR(128)	関数インスタンスの特定名。

表 225. SQLFUNCTIONCOLS ビュー (続き)

列名	データ・タイプ	説明	
JDBC_DATA_TYPE	INTEGER	パラメーターの JDBC データ・タイプ。	
		–5	BIGINT
		4	INTEGER
		5	SMALLINT
		3	DECIMAL
		2	NUMERIC
		8	DOUBLE PRECISION
		7	REAL
		1111	DECFLOAT
		1	CHARACTER または GRAPHIC
		-15	NCHAR
		–2	CHARACTER FOR BIT DATA または BINARY
		12	VARCHAR または VARGRAPHIC
		-9	NVARCHAR
		–3	VARCHAR FOR BIT DATA または VARBINARY
		2005	CLOB または DBCLOB
		2011	NCLOB
		2004	BLOB
		91	DATE
		92	TIME
		93	TIMESTAMP
		70	DATALINK
		-8	ROWID
		2009	XML
		2001	DISTINCT
		2003	ARRAY

表 225. *SQLFUNCTIONCOLS* ビュー (続き)

列名	データ・タイプ	説明
I_DATA_TYPE	INTEGER	パラメーターの IBM i CLI データ・タイプを示します。 **19** BIGINT **4** INTEGER **5** SMALLINT **3** DECIMAL **2** NUMERIC **8** DOUBLE PRECISION **7** REAL **-360** DECFLOAT **1** CHARACTER **–2** CHARACTER FOR BIT DATA または BINARY **12** VARCHAR **–3** VARCHAR FOR BIT DATA または VARBINARY **14** CLOB **95** GRAPHIC または NCHAR **96** VARGRAPHIC または NVARCHAR **15** DBCLOB または NCLOB **13** BLOB **91** DATE **92** TIME **93** TIMESTAMP **16** DATALINK **1111** ROWID **-370** XML **2001** DISTINCT **50** ARRAY
SOURCE_DATA_TYPE	SMALLINT	パラメーターのデータ・タイプが特殊データ・タイプである場合は、ソース・データ・タイプ。値については、JDBC_DATA_TYPE を参照してください。
	NULL 可能	データ・タイプが特殊タイプでない場合は、NULL 値が入ります。
SOURCE_TYPE_NAME	VARCHAR(128)	パラメーターのデータ・タイプがユーザー定義タイプである場合、そのソース・タイプの組み込みデータ・タイプ名。
	NULL 可能	パラメーター・データ・タイプがユーザー定義タイプでない場合は、NULL 値が入ります。
SOURCE_SQL_DATA_TYPE	SMALLINT	パラメーターのデータ・タイプがユーザー定義タイプである場合、そのソース・タイプの組み込み SQL_DATA_TYPE。値については、SQL_DATA_TYPE を参照してください。
	NULL 可能	パラメーター・データ・タイプがユーザー定義タイプでない場合は、NULL 値が入ります。
SOURCE_JDBC_DATA_TYPE	SMALLINT	パラメーターのデータ・タイプがユーザー定義タイプである場合、そのソース・タイプの組み込み JDBC_DATA_TYPE。値については、JDBC_DATA_TYPE を参照してください。
	NULL 可能	パラメーター・データ・タイプがユーザー定義タイプでない場合は、NULL 値が入ります。
MAXIMUM_CARDINALITY	BIGINT	配列データ・タイプの最大カーディナリティー。
	NULL 可能	タイプが配列タイプでない場合は、NULL 値が入ります。

SQLFUNCTIONS

SQLFUNCTIONS ビューには、各関数ごとに行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *226. SQLFUNCTIONS* ビュー

列名	データ・タイプ	説明
FUNCTION_CAT	VARCHAR(128)	リレーショナル・データベース名
FUNCTION_SCHEM	VARCHAR(128)	関数インスタンスのスキーマ名
FUNCTION_NAME	VARCHAR(128)	関数の名前
REMARKS	VARGRAPHIC(2000) CCSID 1200 NULL 可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、NULL 値が入ります。
FUNCTION_TYPE	SMALLINT	関数のタイプ: **1** 列関数またはスカラー関数 **2** 表関数
SPECIFIC_NAME	VARCHAR(128)	関数インスタンスの特定名

SQLPRIMARYKEYS

SQLPRIMARYKEYS ビューには、表の各基本制約キーごとに、行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *227. SQLPRIMARYKEYS* ビュー

列名	データ・タイプ	説明
TABLE_CAT	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEM	VARCHAR(128)	基本キーを持つ表が入っているスキーマの名前。
TABLE_NAME	VARCHAR(128)	基本キーを持つ表の名前。
COLUMN_NAME	VARCHAR(128)	基本キー列の名前。
KEY_SEQ	SMALLINT	キー内における列の位置。
PK_NAME	VARCHAR(128)	基本キー制約の名前。

SQLPROCEDURECOLS

SQLPROCEDURECOLS ビューには、プロシージャーの各パラメーターごとに、行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *228. SQLPROCEDURECOLS* ビュー

列名	データ・タイプ	説明
PROCEDURE_CAT	VARCHAR(128)	リレーショナル・データベース名
PROCEDURE_SCHEM	VARCHAR(128)	プロシージャー・インスタンスのスキーマ名。
PROCEDURE_NAME	VARCHAR(128)	プロシージャー・インスタンスの名前。
COLUMN_NAME	VARCHAR(128)	プロシージャー・パラメーターの名前。
	NULL 可能	パラメーターに名前がない場合は、NULL 値が入ります。
COLUMN_TYPE	SMALLINT	パラメーターのタイプ: **1** IN **2** INOUT **4** OUT

表 228. *SQLPROCEDURECOLS* ビュー (続き)

列名	データ・タイプ	説明	
DATA_TYPE	SMALLINT	パラメーターのデータ・タイプ。	
		–5	BIGINT
		4	INTEGER
		5	SMALLINT
		3	DECIMAL
		2	NUMERIC
		8	DOUBLE PRECISION
		7	REAL
		-360	DECFLOAT
		1	CHARACTER
		–2	CHARACTER FOR BIT DATA または BINARY
		12	VARCHAR
		–3	VARCHAR FOR BIT DATA または VARBINARY
		-99	CLOB
		-95	GRAPHIC
		–96	VARGRAPHIC
		–350	DBCLOB
		–8	NCHAR
		–9	NVARCHAR
		–10	NCLOB
		-98	BLOB
		91	DATE
		92	TIME
		93	TIMESTAMP
		70	DATALINK
		–100	ROWID
		-370	XML
		17	DISTINCT
		50	ARRAY

表 228. SQLPROCEDURECOLS ビュー (続き)

列名	データ・タイプ	説明
TYPE_NAME	VARCHAR(261)	パラメーターのデータ・タイプの名前。 **BIGINT** BIGINT **INTeger** INTEGER **SMALLINT** SMALLINT **DECIMAL** DECIMAL **NUMERIC** NUMERIC **FLOAT** DOUBLE PRECISION **REAL** REAL **DECFLOAT** DECFLOAT **CHARacter** CHARACTER **CHARacter FOR BIT DATA** CHARACTER FOR BIT DATA **VARCHAR** VARCHAR **VARCHAR FOR BIT DATA** VARCHAR FOR BIT DATA **CLOB** CLOB **GRAPHIC** GRAPHIC **VARGRAPHIC** VARGRAPHIC **DBCLOB** DBCLOB **NCHAR** NCHAR **NVARCHAR** NVARCHAR **NCLOB** NCLOB **BINARY** BINARY **VARBINARY** VARBINARY **BLOB** BLOB **DATE** DATE **TIME** TIME **TIMESTAMP** TIMESTAMP **DATALINK** DATALINK **ROWID** ROWID **XML** XML 修飾タイプ名 DISTINCT 配列タイプ名 ARRAY

表 228. *SQLPROCEDURECOLS* ビュー (続き)

列名	データ・タイプ	説明
COLUMN_SIZE	INTEGER	パラメーターの長さ。
BUFFER_LENGTH	INTEGER	バッファー内のパラメーターの長さを示します。
DECIMAL_DIGITS	SMALLINT NULL 可能	数値データまたは日時データの位取り。 パラメーターが 10 進数、数値、2 進数、時刻、またはタイム・スタンプでない場合は、NULL 値が入ります。
NUM_PREC_RADIX	SMALLINT NULL 可能	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 パラメーターが数値パラメーターでない場合は、NULL 値が入ります。
NULLABLE	SMALLINT	パラメーターがNULL 可能かどうかを示します。 **0** パラメーターに NULL は許されません。 **1** パラメーターに NULL が許されます。
REMARKS	VARGRAPHIC(2000) CCSID 1200 NULL 可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、NULL 値が入ります。
COLUMN_DEF	DBCLOB(64K) CCSID(1200) NULL 可能	パラメーターのデフォルト値が存在する場合、パラメーターのデフォルト値を計算するために使用される式ストリング。デフォルト値がヌル値の場合、式ストリングはキーワード NULL です。パラメーターにデフォルトがない場合は、NULL 値が入ります。

表 228. SQLPROCEDURECOLS ビュー (続き)

列名	データ・タイプ	説明
SQL_DATA_TYPE	SMALLINT	パラメーターの SQL データ・タイプ。
		–5 BIGINT
		4 INTEGER
		5 SMALLINT
		3 DECIMAL
		2 NUMERIC
		8 DOUBLE PRECISION
		7 REAL
		-360 DECFLOAT
		1 CHARACTER
		–2 CHARACTER FOR BIT DATA または BINARY
		12 VARCHAR
		–3 VARCHAR FOR BIT DATA または VARBINARY
		–99 CLOB
		-95 GRAPHIC
		–96 VARGRAPHIC
		–350 DBCLOB
		–8 NCHAR
		–9 NVARCHAR
		–10 NCLOB
		–98 BLOB
		9 DATE
		9 TIME
		9 TIMESTAMP
		70 DATALINK
		–100 ROWID
		-370 XML
		17 DISTINCT
		50 ARRAY
SQL_DATETIME_SUB	SMALLINT	パラメーターの日時サブタイプ。
	NULL 可能	**1** DATE
		2 TIME
		3 TIMESTAMP
		データ・タイプが日時データ・タイプでない場合は、NULL 値が入ります。
CHAR_OCTET_LENGTH	INTEGER	パラメーターの長さを文字数で示します。
	NULL 可能	列がストリングでない場合は、NULL 値が入ります。
ORDINAL_POSITION	INTEGER	パラメーター・リストにおける該当のパラメーターの数値位置 (左から右への順序)。
IS_NULLABLE	VARCHAR(3)	パラメーターがNULL 可能かどうかを示します。
		NO パラメーターに NULL は許されません。
		YES パラメーターに NULL が許されます。

表 228. *SQLPROCEDURECOLS* ビュー (続き)

列名	データ・タイプ	説明
SPECIFIC_NAME	VARCHAR(128)	プロシージャー・インスタンスの特定名。
JDBC_DATA_TYPE	INTEGER	パラメーターの JDBC データ・タイプ。 **–5** BIGINT **4** INTEGER **5** SMALLINT **3** DECIMAL **2** NUMERIC **8** DOUBLE PRECISION **7** REAL **1111** DECFLOAT **1** CHARACTER または GRAPHIC **-15** NCHAR **–2** CHARACTER FOR BIT DATA または BINARY **12** VARCHAR または VARGRAPHIC **-9** NVARCHAR **–3** VARCHAR FOR BIT DATA または VARBINARY **2005** CLOB または DBCLOB **2011** NCLOB **2004** BLOB **91** DATE **92** TIME **93** TIMESTAMP **70** DATALINK **-8** ROWID **2009** XML **2001** DISTINCT **2003** ARRAY

表 228. SQLPROCEDURECOLS ビュー (続き)

列名	データ・タイプ	説明
I_DATA_TYPE	INTEGER	パラメーターの IBM i CLI データ・タイプを示します。 **19** BIGINT **4** INTEGER **5** SMALLINT **3** DECIMAL **2** NUMERIC **8** DOUBLE PRECISION **7** REAL **-360** DECFLOAT **1** CHARACTER **–2** CHARACTER FOR BIT DATA または BINARY **12** VARCHAR **–3** VARCHAR FOR BIT DATA または VARBINARY **14** CLOB **95** GRAPHIC または NCHAR **96** VARGRAPHIC または NVARCHAR **15** DBCLOB または NCLOB **13** BLOB **91** DATE **92** TIME **93** TIMESTAMP **16** DATALINK **1111** ROWID **-370** XML **2001** DISTINCT **50** ARRAY
SOURCE_TYPE_NAME	VARCHAR(128) NULL 可能	パラメーターのデータ・タイプがユーザー定義タイプである場合、そのソース・タイプの組み込みデータ・タイプ名。 パラメーター・データ・タイプがユーザー定義タイプでない場合は、NULL 値が入ります。
SOURCE_SQL_DATA_TYPE	SMALLINT NULL 可能	パラメーターのデータ・タイプがユーザー定義タイプである場合、そのソース・タイプの組み込み SQL_DATA_TYPE。値については、SQL_DATA_TYPE を参照してください。 パラメーター・データ・タイプがユーザー定義タイプでない場合は、NULL 値が入ります。
SOURCE_JDBC_DATA_TYPE	SMALLINT NULL 可能	パラメーターのデータ・タイプがユーザー定義タイプである場合、そのソース・タイプの組み込み JDBC_DATA_TYPE。値については、JDBC_DATA_TYPE を参照してください。 パラメーター・データ・タイプがユーザー定義タイプでない場合は、NULL 値が入ります。
MAXIMUM_CARDINALITY	BIGINT NULL 可能	配列データ・タイプの最大カーディナリティー。 タイプが配列タイプでない場合は、NULL 値が入ります。

SQLPROCEDURES

SQLPROCEDURES ビューには、各プロシージャーごとに行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *229. SQLPROCEDURES* ビュー

列名	データ・タイプ	説明
PROCEDURE_CAT	VARCHAR(128)	リレーショナル・データベース名
PROCEDURE_SCHEM	VARCHAR(128)	プロシージャー・インスタンスのスキーマ名
PROCEDURE_NAME	VARCHAR(128)	プロシージャーの名前。
NUM_INPUT_PARAMS	INTEGER	入力パラメーターの数を識別します。 0 は入力パラメーターがないことを示します。
NUM_OUTPUT_PARAMS	INTEGER	出力パラメーターの数を識別します。 0 は出力パラメーターがないことを示します。
NUM_RESULT_SETS	SMALLINT	戻される結果セットの最大数を識別します。 0 は結果セットがないことを示します。
REMARKS	VARGRAPHIC(2000) CCSID 1200 NULL 可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、NULL 値が入ります。
PROCEDURE_TYPE	SMALLINT	予約済み。 0 が入ります。
NUM_INOUT_PARAMS	INTEGER	入出力パラメーターの数を識別します。 0 は入出力パラメーターがないことを示します。
SPECIFIC_NAME	VARCHAR(128)	プロシージャー・インスタンスの特定名。

SQLSCHEMAS

SQLSCHEMAS ビューには、各スキーマごとの行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *230. SQLSCHEMAS* ビュー

列名	データ・タイプ	説明
TABLE_CAT	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEM	VARCHAR(128)	スキーマの名前。
TABLE_NAME	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
TABLE_TYPE	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
REMARKS	VARGRAPHIC(2000) CCSID 1200 NULL 可能	予約済み。 NULL 値が入ります。
TYPE_CAT	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
TYPE_SCHEM	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
TYPE_NAME	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
SELF_REFERENCING_COL_NAME	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
REF_GENERATION	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
DBNAME	VARCHAR(8) NULL 可能	予約済み。 NULL 値が入ります。
SCHEMA_TEXT	VARGRAPHIC(50) CCSID 1200 NULL 可能	スキーマを記述する文字ストリング。 テキストがない場合は、空ストリングが入ります。
SYSTEM_TABLE_SCHEMA	CHAR(10)	システムのスキーマ名。

SQLSPECIALCOLUMNS

SQLSPECIALCOLUMNS ビューには、表の 1 行を識別できる基本キー、固有制約、または固有索引の各列ごとに、行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *231. SQLSPECIALCOLUMNS* ビュー

列名	データ・タイプ	説明
SCOPE	SMALLINT	予約済み。 0 が入ります。
COLUMN_NAME	VARCHAR(128)	列名
DATA_TYPE	SMALLINT	列のデータ・タイプ。
		–5 BIGINT
		4 INTEGER
		5 SMALLINT
		3 DECIMAL
		2 NUMERIC
		8 DOUBLE PRECISION
		7 REAL
		-360 DECFLOAT
		1 CHARACTER
		–2 CHARACTER FOR BIT DATA
		12 VARCHAR
		–3 VARCHAR FOR BIT DATA
		40 CLOB
		-95 GRAPHIC
		–96 VARGRAPHIC
		–350 DBCLOB
		–8 NCHAR
		–9 NVARCHAR
		–10 NCLOB
		–2 BINARY
		–3 VARBINARY
		30 BLOB
		91 DATE
		92 TIME
		93 TIMESTAMP
		70 DATALINK
		–100 ROWID
		17 DISTINCT

表 231. SQLSPECIALCOLUMNS ビュー (続き)

列名	データ・タイプ	説明
TYPE_NAME	VARCHAR(260)	パラメーターのデータ・タイプの名前。 **BIGINT** BIGINT **INTeger** INTEGER **SMALLINT** SMALLINT **DECIMAL** DECIMAL **NUMERIC** NUMERIC **FLOAT** DOUBLE PRECISION **REAL** REAL **DECFLOAT** DECFLOAT **CHARacter** CHARACTER **CHARacter FOR BIT DATA** CHARACTER FOR BIT DATA **VARCHAR** VARCHAR **VARCHAR FOR BIT DATA** VARCHAR FOR BIT DATA **CLOB** CLOB **GRAPHIC** GRAPHIC **VARGRAPHIC** VARGRAPHIC **DBCLOB** DBCLOB **NCHAR** NCHAR **NVARCHAR** NVARCHAR **NCLOB** NCLOB **BINARY** BINARY **VARBINARY** VARBINARY **BLOB** BLOB **DATE** DATE **TIME** TIME **TIMESTAMP** TIMESTAMP **DATALINK** DATALINK **ROWID** ROWID **XML** XML 修飾タイプ名 DISTINCT

表 *231. SQLSPECIALCOLUMNS* ビュー (続き)

列名	データ・タイプ	説明
COLUMN_SIZE	INTEGER	列の長さです。
BUFFER_LENGTH	INTEGER	バッファー内の列の長さを示します。
DECIMAL_DIGITS	SMALLINT	数値列の桁数を示します。
	NULL 可能	列が数値の列でない場合は、NULL 値が入ります。
PSEUDO_COLUMN	SMALLINT	これが、ROWID 列、ID 列、行変更タイム・スタンプ列、行開始列、行終了列、トランザクション開始 ID 列、または生成式列であるかどうかを示します。 **1** この列は、ROWID 列、ID 列、行変更タイム・スタンプ列、行開始列、行終了列、トランザクション開始 ID 列、および生成式列ではありません。 **2** この列は、ROWID 列、ID 列、行変更タイム・スタンプ列、行開始列、行終了列、トランザクション開始 ID 列、または生成式列です。
TABLE_CAT	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEM	VARCHAR(128)	該当の表が入っている SQL のスキーマの名前。
TABLE_NAME	VARCHAR(128)	表の名前。
NULLABLE	SMALLINT	列に NULL 値を入れることができるかどうかを示します。 **0** 列はNULL 可能ではありません。 **1** 列はNULL 可能です。
JDBC_DATA_TYPE	SMALLINT	列の JDBC データ・タイプを示します。 **–5** BIGINT **4** INTEGER **5** SMALLINT **3** DECIMAL **2** NUMERIC **8** DOUBLE PRECISION **7** REAL **1111** DECFLOAT **1** CHARACTER または GRAPHIC **-15** NCHAR **–2** CHARACTER FOR BIT DATA または BINARY **12** VARCHAR または VARGRAPHIC **-9** NVARCHAR **–3** VARCHAR FOR BIT DATA または VARBINARY **2005** CLOB または DBCLOB **2011** NCLOB **2004** BLOB **91** DATE **92** TIME **93** TIMESTAMP **70** DATALINK **-8** ROWID **2009** XML **2001** DISTINCT

表 *231. SQLSPECIALCOLUMNS* ビュー *(*続き*)*

列名	データ・タイプ	説明
I_DATA_TYPE	SMALLINT	列の IBM i CLI データ・タイプを示します。 **19** BIGINT **4** INTEGER **5** SMALLINT **3** DECIMAL **2** NUMERIC **8** DOUBLE PRECISION **7** REAL **-360** DECFLOAT **1** CHARACTER **–2** CHARACTER FOR BIT DATA または BINARY **12** VARCHAR **–3** VARCHAR FOR BIT DATA または VARBINARY **14** CLOB **95** GRAPHIC または NCHAR **96** VARGRAPHIC または NVARCHAR **15** DBCLOB または NCLOB **13** BLOB **91** DATE **92** TIME **93** TIMESTAMP **16** DATALINK **1111** ROWID **-370** XML **2001** DISTINCT
SOURCE_TYPE_NAME	VARCHAR(128) NULL 可能	列データ・タイプがユーザー定義タイプである場合、そのソース・タイプの組み込みデータ・タイプ名。 列のデータ・タイプがユーザー定義タイプでない場合は、NULL 値が入ります。
SOURCE_SQL_DATA_TYPE	SMALLINT NULL 可能	列データ・タイプがユーザー定義タイプである場合、そのソース・タイプの組み込み SQL_DATA_TYPE。値については、SQL_DATA_TYPE を参照してください。 列のデータ・タイプがユーザー定義タイプでない場合は、NULL 値が入ります。
SOURCE_JDBC_DATA_TYPE	SMALLINT NULL 可能	列データ・タイプがユーザー定義タイプである場合、そのソース・タイプの組み込み JDBC_DATA_TYPE。値については、JDBC_DATA_TYPE を参照してください。 列のデータ・タイプがユーザー定義タイプでない場合は、NULL 値が入ります。

SQLSTATISTICS

SQLSTATISTICS ビューには、表についての統計情報が入ります。

次の表は、ビューの列について説明しています。

表 *232. SQLSTATISTICS* ビュー

列名	データ・タイプ	説明
TABLE_CAT	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEM	VARCHAR(128)	該当の表が入っている SQL スキーマの名前。
TABLE_NAME	VARCHAR(128)	表の名前。
NON_UNIQUE	SMALLINT	同一索引の重複キーを表で禁止するかどうかを示します。
	NULL 可能	TYPE が 0 の場合は、NULL 値が入ります。
INDEX_QUALIFIER	VARCHAR(128)	索引のスキーマ名。
	NULL 可能	TYPE が 0 の場合は、NULL 値が入ります。
INDEX_NAME	VARCHAR(128)	索引の名前。
	NULL 可能	TYPE が 0 の場合は、NULL 値が入ります。
TYPE	SMALLINT	戻される情報のタイプを示します。 **0** 表の行数。 **3** 表の索引。
ORDINAL_POSITION	SMALLINT	索引内のキーの順序位置を示します。
	NULL 可能	TYPE が 0 の場合は、NULL 値が入ります。
COLUMN_NAME	DBCLOB(2097152) CCSID 1200	キー列が式の場合は、その式が入ります。キー列が式でない場合は、列名が入ります。
	NULL 可能	TYPE が 0 の場合は、NULL 値が入ります。
ASC_OR_DESC	CHAR(1)	キー内における列の順序:
	NULL 可能	**A** 昇順 **D** 降順 TYPE が 0 の場合は、NULL 値が入ります。
CARDINALITY	BIGINT	表のすべてのパーティションまたはメンバー内の有効行数。
	NULL 可能	TYPE が 3 の場合は、NULL 値が入ります。
PAGES	BIGINT	表のすべてのパーティションまたはメンバー内の 64K ページの数。
	NULL 可能	TYPE が 3 の場合は、NULL 値が入ります。
FILTER_CONDITION	DBCLOB(2097152) CCSID 1200	索引が疎索引かどうかを示します。
	NULL 可能	検索条件 これは、WHERE 節のある SQL 索引です。 空ストリング これは、DDS で作成された選択/除外索引です。 TYPE が 0 の場合、またはこれが疎索引でない場合は、NULL 値が入ります。
I_INDEXTYPE	INTEGER	索引のタイプが入ります。 **0** TYPE は 0 です。 **1** 索引は PRIMARY KEY またはキー付き物理ファイルです。 **2** 索引は SQL 索引です。 **3** 索引はキー付き論理ファイルです。 **4** 索引は UNIQUE 制約または REFERENTIAL 制約です。

SQLTABLEPRIVILEGES

SQLTABLEPRIVILEGES ビューには、表に対して認可された各特権ごとに、行が 1 つずつ入ります。このカタログ・ビューを使用して、特定ユーザーが特定の表またはビューに対する権限を持っているかどうかを判別することはできないので、注意してください。なぜなら、表またはビューを使用するための特権は、グループ・ユーザー・プロファイルまたは特殊権限 (*ALLOBJ など) を通じて獲得できるからです。

次の表は、ビューの列について説明しています。

表 *233. SQLTABLEPRIVILEGES* ビュー

列名	データ・タイプ	説明
TABLE_CAT	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEM	VARCHAR(128)	該当の表が入っている SQL スキーマの名前。
TABLE_NAME	VARCHAR(128)	表の名前。
GRANTOR	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
GRANTEE	VARCHAR(128)	特権を認可する対象のユーザー・プロファイル。
PRIVILEGE	VARCHAR(10)	認可される特権 : **ALTER** 表を変更する特権。 **DELETE** 表から行を削除する特権。 **INDEX** 表の索引を作成する特権。 **INSERT** 表に行を挿入する特権。 **REFERENCES** 参照制約の中で表を参照する特権。 **SELECT** 表から行を選択する特権。 **UPDATE** 表を更新する特権。
IS_GRANTABLE	VARCHAR(3)	特権を他のユーザーに認可できるかどうかを示します。 **NO** 特権は認可できません。 **YES** 特権を認可できます。
DBNAME	VARCHAR(8) NULL 可能	予約済み。 NULL 値が入ります。

SQLTABLES

SQLTABLES ビューには、各表、ビュー、および別名ごとに、行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *234. SQLTABLES* ビュー

列名	データ・タイプ	説明
TABLE_CAT	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEM	VARCHAR(128)	該当の表が入っているスキーマの名前。
TABLE_NAME	VARCHAR(128)	表の名前。
TABLE_TYPE	VARCHAR(24)	表のタイプを識別します。 **ALIAS** 表は別名です。 **MATERIALIZED QUERY TABLE** オブジェクトはマテリアライズ照会表です。 **SYSTEM TABLE** 表は、システム表です。 **TABLE** 表は、SQL 表または物理ファイルです。 **VIEW** 表は、SQL ビューまたは論理ファイルです。
REMARKS	VARGRAPHIC(2000) CCSID 1200 NULL 可能	COMMENT ステートメントで指定された文字ストリング。 詳細コメントがない場合は、NULL 値が入ります。
TYPE_CAT	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
TYPE_SCHEM	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
TYPE_NAME	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
SELF_REF_COL_NAME	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
REF_GENERATION	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
DBNAME	VARCHAR(8) NULL 可能	予約済み。 NULL 値が入ります。
TABLE_TEXT	VARGRAPHIC(50) CCSID 1200	LABEL ステートメントで指定された文字ストリング。

SQLTYPEINFO

SQLTYPEINFO 表には、各組み込みデータ・タイプごとに、行が 1 つずつ入ります。

次の表は、この表の列について説明しています。

表 235. *SQLTYPEINFO* 表

列名	データ・タイプ	説明
TYPE_NAME	VARCHAR(128)	組み込みデータ・タイプの名前: **BIGINT** BIGINT **INTeger** INTEGER **SMALLINT** SMALLINT **DECIMAL** DECIMAL **NUMERIC** NUMERIC **FLOAT** DOUBLE PRECISION **REAL** REAL **DECFLOAT** DECFLOAT **CHARacter** CHARACTER **CHARacter FOR BIT DATA** CHARACTER FOR BIT DATA **VARCHAR** VARCHAR **VARCHAR FOR BIT DATA** VARCHAR FOR BIT DATA **CLOB** CLOB **GRAPHIC** GRAPHIC **VARGRAPHIC** VARGRAPHIC **DBCLOB** DBCLOB **NCHAR** NCHAR **NVARCHAR** NVARCHAR **NCLOB** NCLOB **BINARY** BINARY **VARBINARY** VARBINARY **BLOB** BLOB **DATE** DATE **TIME** TIME **TIMESTAMP** TIMESTAMP **DATALINK** DATALINK **ROWID** ROWID **XML** XML

表 235. SQLTYPEINFO 表 (続き)

列名	データ・タイプ	説明
DATA_TYPE	SMALLINT	組み込みデータ・タイプのデータ・タイプ:
		–5 BIGINT
		4 INTEGER
		5 SMALLINT
		3 DECIMAL
		2 NUMERIC
		8 DOUBLE PRECISION
		7 REAL
		-360 DECFLOAT
		1 CHARACTER
		–2 CHARACTER FOR BIT DATA または BINARY
		12 VARCHAR
		–3 VARCHAR FOR BIT DATA または VARBINARY
		-99 CLOB
		-95 GRAPHIC
		–96 VARGRAPHIC
		–350 DBCLOB
		–8 NCHAR
		–9 NVARCHAR
		–10 NCLOB
		-99 BLOB
		91 DATE
		92 TIME
		93 TIMESTAMP
		70 DATALINK
		–100 ROWID
		-370 XML
		2001 DISTINCT
COLUMN_SIZE	INTEGER	データ・タイプの最大長。
	NULL 可能	
LITERAL_PREFIX	VARCHAR(128)	ストリング・リテラルのプレフィックスを示します。
	NULL 可能	データ・タイプがストリングでない場合は、NULL 値が入ります。
LITERAL_SUFFIX	VARCHAR(128)	ストリング・リテラルのサフィックスを示します。
	NULL 可能	データ・タイプがストリングでない場合は、NULL 値が入ります。

表 235. *SQLTYPEINFO* 表 (続き)

列名	データ・タイプ	説明
CREATE_PARAMS	VARCHAR(128) NULL 可能	データ・タイプでサポートされるパラメーターを示します。 **length** パラメーターは、長さです。すべてのストリング・データ・タイプおよび DATALINK に対して戻されます。 **precision,scale** パラメーターには、精度および位取りが含まれます。すべての DECIMAL および NUMERIC データ・タイプに対して戻されます。 精度 パラメーターには、精度が含まれます。すべての DECFLOAT および TIMESTAMP データ・タイプに対して戻されます。 その他のすべてのデータ・タイプの場合は、NULL 値が入ります。
NULLABLE	SMALLINT NULL 可能	データ・タイプがNULL 可能かどうかを示します。 **0** このデータ・タイプでは NULL は許されません。 **1** このデータ・タイプでは NULL が許されます。
CASE_SENSITIVE	SMALLINT NULL 可能	データ・タイプで、大文字小文字が区別されるかどうかを示します。 **0** このデータ・タイプでは大文字小文字は区別されません。 **1** このデータ・タイプでは、大文字小文字が区別されます。
SEARCHABLE	SMALLINT NULL 可能	データ・タイプを述部で使用できるかどうかを示します。 **0** このデータ・タイプは述部では使用できません。 **2** このデータ・タイプは LIKE 述部以外のすべての述部で使用できます。 **3** このデータ・タイプは、LIKE 述部を含め、すべての述部で使用できます。
UNSIGNED_ATTRIBUTE	SMALLINT NULL 可能	数値データ・タイプが符号付きか符号なしかを示します。 **0** データ・タイプは符号付きです。 **1** データ・タイプは符号なしです。 データ・タイプが数値でない場合は、NULL 値が入ります。
FIXED_PREC_SCALE	SMALLINT NULL 可能	データ・タイプに固定した精度および位取りがあるかどうかを示します。 **0** データ・タイプには、固定した精度および位取りはありません。 **1** データ・タイプには、固定した精度および位取りがあります。 データ・タイプが数値でない場合は、NULL 値が入ります。
AUTO_UNIQUE_VALUE	SMALLINT NULL 可能	数値データ・タイプが自動増分かどうかを示します。 **0** データ・タイプは自動増分ではありません。 **1** データ・タイプは自動増分です。 データ・タイプが数値でない場合は、NULL 値が入ります。
LOCAL_TYPE_NAME	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
MINIMUM_SCALE	SMALLINT NULL 可能	数値データ・タイプの最小位取りを示します。 データ・タイプが数値でない場合は、NULL 値が入ります。
MAXIMUM_SCALE	SMALLINT NULL 可能	数値データ・タイプの最大位取りを示します。 データ・タイプが数値でない場合は、NULL 値が入ります。

表 235. SQLTYPEINFO 表 (続き)

列名	データ・タイプ	説明
SQL_DATA_TYPE	SMALLINT NULL 可能	データ・タイプの SQL データ・タイプ値を示します。 **–5** BIGINT **4** INTEGER **5** SMALLINT **3** DECIMAL **2** NUMERIC **8** DOUBLE PRECISION **7** REAL **-360** DECFLOAT **1** CHARACTER **–2** CHARACTER FOR BIT DATA または BINARY **12** VARCHAR **–3** VARCHAR FOR BIT DATA または VARBINARY **-99** CLOB **-95** GRAPHIC **–96** VARGRAPHIC **–350** DBCLOB **–8** NCHAR **–9** NVARCHAR **–10** NCLOB **-98** BLOB **9** DATE **9** TIME **9** TIMESTAMP **70** DATALINK **–100** ROWID **-370** XML **17** DISTINCT
SQL_DATETIME_SUB	SMALLINT NULL 可能	データ・タイプの日時サブタイプ: **1** DATE **2** TIME **3** TIMESTAMP データ・タイプが日時データ・タイプでない場合は、NULL 値が入ります。
NUM_PREC_RADIX	INTEGER NULL 可能	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 パラメーターが数値パラメーターでない場合は、NULL 値が入ります。
INTERVAL_PRECISION	SMALLINT NULL 可能	予約済み。 NULL 値が入ります。

表 *235. SQLTYPEINFO* 表 (続き)

列名	データ・タイプ	説明	
JDBC_DATA_TYPE	SMALLINT	データ・タイプの JDBC データ・タイプ値:	
		–5	BIGINT
		4	INTEGER
		5	SMALLINT
		3	DECIMAL
		2	NUMERIC
		8	DOUBLE PRECISION
		7	REAL
		1111	DECFLOAT
		1	CHARACTER または GRAPHIC
		-15	NCHAR
		–2	CHARACTER FOR BIT DATA または BINARY
		12	VARCHAR または VARGRAPHIC
		-9	NVARCHAR
		–3	VARCHAR FOR BIT DATA または VARBINARY
		2005	CLOB または DBCLOB
		2011	NCLOB
		2004	BLOB
		91	DATE
		92	TIME
		93	TIMESTAMP
		70	DATALINK
		-8	ROWID
		2009	XML
		2001	DISTINCT

表 235. SQLTYPEINFO 表 (続き)

列名	データ・タイプ	説明	
I_DATA_TYPE	SMALLINT	データ・タイプの IBM i CLI データ・タイプを示します。	
		19	BIGINT
		4	INTEGER
		5	SMALLINT
		3	DECIMAL
		2	NUMERIC
		8	DOUBLE PRECISION
		7	REAL
		-360	DECFLOAT
		1	CHARACTER
		–2	CHARACTER FOR BIT DATA または BINARY
		12	VARCHAR
		–3	VARCHAR FOR BIT DATA または VARBINARY
		14	CLOB
		95	GRAPHIC または NCHAR
		96	VARGRAPHIC または NVARCHAR
		15	DBCLOB または NCLOB
		13	BLOB
		91	DATE
		92	TIME
		93	TIMESTAMP
		16	DATALINK
		1111	ROWID
		-370	XML
		2001	DISTINCT

SQLUDTS

SQLUDTS ビューには、各特殊タイプごとに、行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *236. SQLUDTS* ビュー

列名	データ・タイプ	説明
TYPE_CAT	VARCHAR(128)	リレーショナル・データベース名
TYPE_SCHEM	VARCHAR(128)	ユーザー定義タイプが入っているスキーマの名前。
TYPE_NAME	VARCHAR(128)	ユーザー定義タイプの名前。

表 *236. SQLUDTS* ビュー *(*続き*)*

列名	データ・タイプ	説明
CLASS_NAME	VARCHAR(20)	ユーザー定義タイプの Java クラス名。 **java.math.BigInteger** BIGINT **java.lang.Integer** INTEGER **java.lang.Short** SMALLINT **java.math.BigDecimal** DECIMAL **java.sql.BigDecimal** NUMERIC **java.lang.Double** DOUBLE PRECISION **java.lang.Float** REAL **java.math.BigDecimal** DECFLOAT **java.lang.String** CHARACTER バイト[] CHARACTER FOR BIT DATA **java.lang.String** VARCHAR バイト[] VARCHAR FOR BIT DATA **java.sql.Clob** CLOB **java.lang.String** GRAPHIC **java.lang.String** VARGRAPHIC **java.sql.Clob** DBCLOB バイト[] BINARY バイト[] VARBINARY **java.sql.Blob** BLOB **java.sql.Date** DATE **java.sql.Time** TIME **java.sql.Timestamp** TIMESTAMP **java.net.URL** DATALINK バイト[] ROWID **java.sql.SQLXML** XML
DATA_TYPE	SMALLINT	予約済み。 2001 が入ります。

表 *236. SQLUDTS* ビュー (続き)

列名	データ・タイプ	説明
BASE_TYPE	SMALLINT	ユーザー定義のデータ・タイプのソース・データ・タイプ: **–5** BIGINT **4** INTEGER **5** SMALLINT **3** DECIMAL **2** NUMERIC **8** DOUBLE PRECISION **7** REAL **1111** DECFLOAT **1** CHARACTER または GRAPHIC **-15** NCHAR **–2** CHARACTER FOR BIT DATA または BINARY **12** VARCHAR または VARGRAPHIC **-9** NVARCHAR **–3** VARCHAR FOR BIT DATA または VARBINARY **2005** CLOB または DBCLOB **2011** NCLOB **2004** BLOB **91** DATE **92** TIME **93** TIMESTAMP **70** DATALINK **1111** ROWID **2009** XML
REMARKS	VARGRAPHIC(2000) CCSID 1200 NULL 可能	COMMENT ステートメントで指定された文字ストリング。 コメントがない場合は、NULL 値が入ります。

ANS および ISO のカタログ・ビュー

一部の ANS および ISO のカタログ・ビューには、2 つのバージョンがあります。本書に記載してあるバージョンは、正規セットの ANS および ISO のビューです。2 番目のセットのビューは、18 文字以下に名前が制限されているもので、本書にはビュー名のみを示してあります。

ANS および ISO カタログには、QSYS2 ライブラリー内にある以下の表が含まれます。

ビュー名	短いビュー名	説明
2173 ページの『SQL_FEATURES』		データベース・マネージャーでサポートされている機能についての情報
2174 ページの『SQL_LANGUAGES』	SQL_LANGUAGES_S	サポートされている言語についての情報
2175 ページの『SQL_SIZING』		データベース・マネージャーでサポートされている限度についての情報

ANS および ISO カタログには、SYSIBM および QSYS2 ライブラリー内にある以下のビューおよび表が含まれます。

ビュー名	短いビュー名	説明
2147 ページの『権限』		権限 ID についての情報
2148 ページの『CHARACTER_SETS』	CHARACTER_SETS_S	サポートされている CCSID についての情報
2149 ページの『CHECK_CONSTRAINTS』		検査制約についての情報
2150 ページの『COLUMN_PRIVILEGES』		列特権についての情報
2151 ページの『COLUMNS』	COLUMNS_S	列についての情報
2156 ページの『INFORMATION_SCHEMA_CATALOG_NAME』	CATALOG_NAME	リレーショナル・データベースについての情報
2157 ページの『PARAMETERS』	PARAMETERS_S	プロシージャー・パラメーターについての情報
2161 ページの『REFERENTIAL_CONSTRAINTS』	REF_CONSTRAINTS	参照制約についての情報
2170 ページの『ROUTINE_PRIVILEGES』	RTNPRIV	ルーチン特権についての情報
2162 ページの『ROUTINES』	ROUTINES_S	ルーチンについての情報
2171 ページの『SCHEMATA』	SCHEMATA_S	スキーマについての統計情報
2172 ページの『SEQUENCES』		シーケンスについての情報
2176 ページの『TABLE_CONSTRAINTS』		制約についての情報
2177 ページの『TABLE_PRIVILEGES』		表特権についての情報
2178 ページの『TABLES』	TABLES_S	表についての情報
2179 ページの『UDT_PRIVILEGES』	UDTPRIV	タイプ特権についての情報
2180 ページの『USAGE_PRIVILEGES』	USAGEPRIV	シーケンス特権および XML スキーマ特権についての情報
2181 ページの『USER_DEFINED_TYPES』	UDT_S	タイプについての情報
2185 ページの『VARIABLE_PRIVILEGES』	VARPRIV	グローバル変数特権についての情報
2186 ページの『VIEWS』		ビューについての情報

権限

AUTHORIZATIONS ビューには、各権限 ID ごとに行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *237. AUTHORIZATIONS* ビュー

列名	データ・タイプ	説明
AUTHORIZATION_NAME	VARCHAR(128)	権限 ID 名
AUTHORIZATION_TYPE	VARCHAR(4)	許可 ID のタイプ。'USER' が入ります。
AUTHORIZATION_ATTR	VARCHAR(5)	ユーザー・プロファイルのタイプ。「USER」または「GROUP」が入ります。
AUTHORIZATION_TEXT	VARCHAR(50)	プロファイルのテキスト記述。

CHARACTER_SETS

CHARACTER_SETS ビューには、サポートされている各 CCSID ごとに、行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *238. CHARACTER_SETS* ビュー

列名	データ・タイプ	説明
CHARACTER_SET_CATALOG	VARCHAR(128)	リレーショナル・データベース名
CHARACTER_SET_SCHEMA	VARCHAR(128)	文字セットのスキーマ名。 'SYSIBM' が入ります。
CHARACTER_SET_NAME	VARCHAR(128)	文字セット名。
FORM_OF_USE	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
NUMBER_OF_CHARACTERS	INTEGER NULL 可能	予約済み。 NULL 値が入ります。
DEFAULT_COLLATE_CATALOG	VARCHAR(128)	予約済み。
DEFAULT_COLLATE_SCHEMA	VARCHAR(128)	予約済み。
DEFAULT_COLLATE_NAME	VARCHAR(128)	予約済み。

CHECK_CONSTRAINTS

CHECK_CONSTRAINTS ビューには、各検査制約ごとに、行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *239. CHECK_CONSTRAINTS* ビュー

列名	データ・タイプ	説明
CONSTRAINT_CATALOG	VARCHAR(128)	リレーショナル・データベース名
CONSTRAINT_SCHEMA	VARCHAR(128)	該当の制約が入っているスキーマの名前
CONSTRAINT_NAME	VARCHAR(128)	制約の名前
CHECK_CLAUSE	VARGRAPHIC(2000) CCSID 1200 NULL 可能	検査制約文節のテキスト 切り捨てなければ検査文節を列に収容できない場合は、NULL 値が入ります。

COLUMN_PRIVILEGES

COLUMN_PRIVILEGES ビューには、列に対して認可された各特権ごとに、行が 1 つずつ入ります。このカタログ・ビューを使用して、特定ユーザーが特定の列に対する権限を持っているかどうかを判別することはできないので、注意してください。なぜなら、列を使用するための特権は、グループ・ユーザー・プロファイルまたは特殊権限 (*ALLOBJ など) を通じて獲得できるからです。さらに、列を使用するための特権は、表に関して付与された特権によっても獲得されます。

次の表は、ビューの列について説明しています。

表 *240. COLUMN_PRIVILEGES* ビュー

列名	データ・タイプ	説明
GRANTOR	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
GRANTEE	VARCHAR(128)	特権を認可する対象のユーザー・プロファイル。
TABLE_CATALOG	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEMA	VARCHAR(128)	該当の表が入っている SQL のスキーマの名前。
TABLE_NAME	VARCHAR(128)	表の名前。
COLUMN_NAME	VARCHAR(128)	列の名前。
PRIVILEGE_TYPE	VARCHAR(10)	認可される特権 : **UPDATE** 列を更新する特権。 **REFERENCES** 参照制約モードで列を参照する特権。
IS_GRANTABLE	VARCHAR(3)	特権を他のユーザーに認可できるかどうかを示します。 **NO** 特権は認可できません。 **YES** 特権を認可できます。

COLUMNS

COLUMNS ビューには、各列ごとに行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *241. COLUMNS* ビュー

列名	データ・タイプ	説明
TABLE_CATALOG	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEMA	VARCHAR(128)	該当の表またはビューが入っている SQL スキーマの名前。
TABLE_NAME	VARCHAR(128)	該当の列を含む表またはビューの名前。
COLUMN_NAME	VARCHAR(128)	列の名前。
ORDINAL_POSITION	INTEGER	表またはビューにおける該当の列の数値位置 (左から右への順序)。
COLUMN_DEFAULT	VARGRAPHIC(2000) CCSID 1200 NULL 可能	列のデフォルト値が存在する場合は、そのデフォルト値。列のデフォルト値が、切り捨てなければ表示できない場合は、その列の値はストリング 'TRUNCATED' になります。デフォルト値は文字形式で保管されます。以下の特殊値も存在します。 **CURRENT_DATE** デフォルト値は、現在の日付です。 **CURRENT_TIME** デフォルト値は、現在の時刻です。 **CURRENT_TIMESTAMP** デフォルト値は、現在のタイム・スタンプです。 **NULL** デフォルト値は NULL 値になり、DEFAULT NULL が明示的に指定されています。 **USER** デフォルト値は、現在のジョブ・ユーザーです。 *special-register* 列 HAS_DEFAULT に値 'a' が含まれている場合、特殊レジスターの名前。 *global-variable* 列 HAS_DEFAULT に値 'c' が含まれている場合、グローバル変数の修飾名。 **DATA CHANGE OPERATION** 列 HAS_DEFAULT に値 'd' が含まれている場合。 以下の場合、NULL 値が入ります。 • 列にデフォルト値がない場合 (例えば、列に IDENTITY 属性が指定されている場合、列が行 ID である場合、または列が行変更タイム・スタンプ列、行開始列、行終了列、またはトランザクション開始 ID 列である場合)、または • DEFAULT 値が明示的に指定されていない場合。
IS_NULLABLE	VARCHAR(3)	列に NULL 値を入れることができるかどうかを示します。 **NO** 列には NULL 値を入れることはできません。 **YES** 列には NULL 値を入れることができます。

表 241. COLUMNS ビュー (続き)

列名	データ・タイプ	説明
DATA_TYPE	VARCHAR(128)	列のタイプ: **BIGINT** 大整数 **INTEGER** 長整数 **SMALLINT** 短整数 **DECIMAL** パック 10 進数 **NUMERIC** ゾーン 10 進数 **DOUBLE PRECISION** 倍精度浮動小数点数 **REAL** 単精度浮動小数点数 **DECFLOAT** 10 進浮動小数点数 **CHARACTER** 固定長文字ストリング **CHARACTER VARYING** 可変長文字ストリング **CHARACTER LARGE OBJECT** 文字ラージ・オブジェクト・ストリング **GRAPHIC** 固定長グラフィック・ストリング **GRAPHIC VARYING** 可変長グラフィック・ストリング **DOUBLE-BYTE CHARACTER LARGE OBJECT** 2 バイト文字ラージ・オブジェクト・ストリング **NATIONAL CHARACTER** 国別文字 **NATIONAL CHARACTER VARYING** 可変長国別文字 **NATIONAL CHARACTER LARGE OBJECT** 国別文字ラージ・オブジェクト **BINARY** 固定長バイナリー・ストリング **BINARY VARYING** 可変長バイナリー・ストリング **BINARY LARGE OBJECT** バイナリー・ラージ・オブジェクト・ストリング **DATE** 日付 **TIME** 時刻 **TIMESTAMP** タイム・スタンプ **DATALINK** データ・リンク **ROWID** 行 ID **XML** XML **USER-DEFINED** 特殊タイプ

表 241. COLUMNS ビュー (続き)

列名	データ・タイプ	説明
CHARACTER_MAXIMUM_LENGTH	INTEGER NULL 可能	データ・タイプが 2 進ストリング、文字ストリング、グラフィック・ストリング、および XML の場合は、ストリングの最大長。 列がストリングでない場合は、NULL 値が入ります。
CHARACTER_OCTET_LENGTH	INTEGER NULL 可能	データ・タイプが 2 進ストリング、文字ストリング、グラフィック・ストリング、および XML の場合は、バイト数。 列がストリングでない場合は、NULL 値が入ります。
NUMERIC_PRECISION	INTEGER NULL 可能	数値の列すべての精度。 **注:** この列では、すべての数値データ・タイプ (単精度および倍精度の浮動小数点数を含む) の精度を指定します。NUMERIC_PRECISION_RADIX 列は、この列の値が 2 進数であるか、または 10 進数であるかを示します。 列が数値の列でない場合は、NULL 値が入ります。
NUMERIC_PRECISION_RADIX	INTEGER NULL 可能	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 列が数値の列でない場合は、NULL 値が入ります。
NUMERIC_SCALE	INTEGER NULL 可能	数値データの位取り。 列が 10 進数、数値、または 2 進数でない場合は、NULL 値が入ります。
DATETIME_PRECISION	INTEGER NULL 可能	日付、時刻、またはタイム・スタンプの小数部分。 **0** データ・タイプが DATE および TIME の場合 **0-12** データ・タイプが TIMESTAMP の場合 (小数秒) 列が日付、時刻、またはタイム・スタンプの列でない場合は、NULL 値が入ります。
INTERVAL_TYPE	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
INTERVAL_PRECISION	INTEGER NULL 可能	予約済み。 NULL 値が入ります。
CHARACTER_SET_CATALOG	VARCHAR(128) NULL 可能	リレーショナル・データベース名 列がストリングでない場合は、NULL 値が入ります。
CHARACTER_SET_SCHEMA	VARCHAR(128) NULL 可能	文字セットのスキーマ名。 SYSIBM が入ります。 列がストリングでない場合は、NULL 値が入ります。
CHARACTER_SET_NAME	VARCHAR(128) NULL 可能	文字セット名。 列がストリングでない場合は、NULL 値が入ります。
COLLATION_CATALOG	VARCHAR(128) NULL 可能	リレーショナル・データベース名 列がストリングでない場合は、NULL 値が入ります。
COLLATION_SCHEMA	VARCHAR(128) NULL 可能	照合のスキーマ。 SYSIBM が入ります。 列がストリングでない場合は、NULL 値が入ります。
COLLATION_NAME	VARCHAR(128) NULL 可能	照合名。 IBMBINARY が入ります。 列がストリングでない場合は、NULL 値が入ります。

表 241. COLUMNS ビュー (続き)

列名	データ・タイプ	説明
DOMAIN_CATALOG	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
DOMAIN_SCHEMA	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
DOMAIN_NAME	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
UDT_CATALOG	VARCHAR(128) NULL 可能	特殊タイプの場合は、リレーショナル・データベース名。 特殊タイプでない場合には、NULL 値が入ります。
UDT_SCHEMA	VARCHAR(128) NULL 可能	特殊タイプの場合は、スキーマの名前。 特殊タイプでない場合には、NULL 値が入ります。
UDT_NAME	VARCHAR(128) NULL 可能	特殊タイプの名前。 特殊タイプでない場合には、NULL 値が入ります。
SCOPE_CATALOG	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
SCOPE_SCHEMA	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
SCOPE_NAME	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
MAXIMUM_CARDINALITY	INTEGER NULL 可能	予約済み。 NULL 値が入ります。
DTD_IDENTIFIER	VARCHAR(128) NULL 可能	列の固有の内部 ID。
IS_SELF_REFERENCING	VARCHAR(3)	予約済み。 'NO' が入ります。
IS_IDENTITY	VARCHAR(3)	この列は、列が識別列かどうかを指定します。 **NO** 列は識別列ではありません。 **YES** 列は識別列です。
IDENTITY_GENERATION	VARCHAR(10) NULL 可能	この列は、列が GENERATED ALWAYS か GENERATED BY DEFAULT かを識別します。 **ALWAYS** 列の値は常に生成されます。 **BY DEFAULT** 列の値はデフォルトにより生成されます。 列が ROWID、IDENTITY、または行変更タイム・スタンプ列でない場合は、NULL 値が入ります。
IDENTITY_START	DECIMAL(31,0) NULL 可能	識別列の開始値。 列が IDENTITY 列でない場合は、NULL 値が入ります。
IDENTITY_INCREMENT	DECIMAL(31,0) NULL 可能	識別列の増分値。 列が IDENTITY 列でない場合は、NULL 値が入ります。
IDENTITY_MAXIMUM	DECIMAL(31,0) NULL 可能	識別列の最大値。 列が IDENTITY 列でない場合は、NULL 値が入ります。

表 241. *COLUMNS* ビュー (続き)

列名	データ・タイプ	説明
IDENTITY_MINIMUM	DECIMAL(31,0)	識別列の最小値。 列が IDENTITY 列でない場合は、NULL 値が入ります。
	NULL 可能	
IDENTITY_CYCLE	VARCHAR(3)	この列は、識別列の値が最小値または最大値に達した後も、値の生成を続けるかどうかを識別します。 **NO** 値の生成は継続されません。 **YES** 値の生成は継続されます。 列が IDENTITY 列でない場合は、NULL 値が入ります。
	NULL 可能	
IS_GENERATED	VARCHAR(5)	予約済み。 'NEVER' が入ります。
GENERATION_EXPRESSION	VARCHAR(128)	予約済み。 NULL 値が入ります。
	NULL 可能	
IS_SYSTEM_TIME_PERIOD_START	VARCHAR(3)	この列は、列が行開始列かどうかを指定します。 **NO** この列は、行開始列ではありません。 **YES** この列は、行開始列です。
IS_SYSTEM_TIME_PERIOD_END	VARCHAR(3)	この列は、列が行終了列かどうかを指定します。 **NO** この列は、行終了列ではありません。 **YES** この列は、行終了列です。
SYSTEM_TIME_PERIOD_TIMESTAMP_GENERATION	VARCHAR(6)	この列は、行開始または行終了の生成属性を識別します。
	NULL 可能	**ALWAYS** この列は、GENERATED ALWAYS の行開始列または行終了列です。 行開始列でも行終了列でもない場合には、NULL 値が入ります。
IS_UPDATABLE	VARCHAR(3)	この列は、列が更新可能かどうかを示します。 **NO** 列は更新できません。 **YES** 列は更新できます。
DECLARED_DATA_TYPE	VARCHAR(128)	予約済み。 NULL 値が入ります。
	NULL 可能	
DECLARED_NUMERIC_PRECISION	INTEGER	予約済み。 NULL 値が入ります。
	NULL 可能	
DECLARED_NUMERIC_SCALE	INTEGER	予約済み。 NULL 値が入ります。
	NULL 可能	

INFORMATION_SCHEMA_CATALOG_NAME

INFORMATION_SCHEMA_CATALOG_NAME ビューには、リレーショナル・データベースに対応する行が 1 つ入ります。

次の表は、ビューの列について説明しています。

表 242. *INFORMATION_SCHEMA_CATALOG_NAME* ビュー

列名	データ・タイプ	説明
CATALOG_NAME	VARCHAR(128)	リレーショナル・データベース名

PARAMETERS

PARAMETERS ビューには、リレーショナル・データベース内のルーチンの各パラメーターごとに、行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *243. PARAMETERS* ビュー

列名	データ・タイプ	説明
SPECIFIC_CATALOG	VARCHAR(128)	リレーショナル・データベース名
SPECIFIC_SCHEMA	VARCHAR(128)	ルーチン (関数) インスタンスのスキーマ名。
SPECIFIC_NAME	VARCHAR(128)	ルーチン・インスタンスの特定名。
ORDINAL_POSITION	INTEGER	パラメーター・リストにおける該当のパラメーターの数値位置 (左から右への順序)。
PARAMETER_MODE	VARCHAR(5)	パラメーターのタイプ: **IN** これは入力パラメーターです。 **OUT** これは出力パラメーターです。 **INOUT** これは入出力パラメーターです。
IS_RESULT	VARCHAR(3)	予約済み。 'NO' が入ります。
AS_LOCATOR	VARCHAR(3)	パラメーターがロケーターとして指定されたかどうかを識別します。 **NO** パラメーターはロケーターとして指定されませんでした。 **YES** パラメーターはロケーターとして指定されました。
PARAMETER_NAME	VARCHAR(128) NULL 可能	パラメーターの名前。 パラメーターに名前がない場合は、NULL 値が入ります。
FROM_SQL_SPECIFIC_CATALOG	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
FROM_SQL_SPECIFIC_SCHEMA	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
FROM_SQL_SPECIFIC_NAME	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
TO_SQL_SPECIFIC_CATALOG	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
TO_SQL_SPECIFIC_SCHEMA	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
TO_SQL_SPECIFIC_NAME	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。

PARAMETERS

表 243. *PARAMETERS* ビュー (続き)

列名	データ・タイプ	説明
DATA_TYPE	VARCHAR(128) NULL 可能	パラメーターのタイプ: **BIGINT** 大整数 **INTEGER** 長整数 **SMALLINT** 短整数 **DECIMAL** パック 10 進数 **NUMERIC** ゾーン 10 進数 **DOUBLE PRECISION** 浮動小数点数; DOUBLE PRECISION **REAL** 浮動小数点数; REAL **DECFLOAT** 10 進浮動小数点数 **CHARACTER** 固定長文字ストリング **CHARACTER VARYING** 可変長文字ストリング **CHARACTER LARGE OBJECT** 文字ラージ・オブジェクト・ストリング **GRAPHIC** 固定長グラフィック・ストリング **GRAPHIC VARYING** 可変長グラフィック・ストリング **DOUBLE-BYTE CHARACTER LARGE OBJECT** 2 バイト文字ラージ・オブジェクト・ストリング **NATIONAL CHARACTER** 国別文字 **NATIONAL CHARACTER VARYING** 可変長国別文字 **NATIONAL CHARACTER LARGE OBJECT** 国別文字ラージ・オブジェクト **BINARY** 固定長バイナリー・ストリング **BINARY VARYING** 可変長バイナリー・ストリング **BINARY LARGE OBJECT** バイナリー・ラージ・オブジェクト・ストリング **DATE** 日付 **TIME** 時刻 **TIMESTAMP** タイム・スタンプ **DATALINK** データ・リンク **ROWID** 行 ID **XML** XML **USER-DEFINED** 特殊タイプまたは配列タイプ

表 243. *PARAMETERS* ビュー (続き)

列名	データ・タイプ	説明
CHARACTER_MAXIMUM_LENGTH	INTEGER NULL 可能	データ・タイプが 2 進ストリング、文字ストリング、グラフィック・ストリング、および XML の場合は、ストリングの最大長。 パラメーターがストリングでない場合は、NULL 値が入ります。
CHARACTER_OCTET_LENGTH	INTEGER NULL 可能	データ・タイプが 2 進ストリング、文字ストリング、グラフィック・ストリング、および XML の場合は、バイト数。 パラメーターがストリングでない場合は、NULL 値が入ります。
CHARACTER_SET_CATALOG	VARCHAR(128) NULL 可能	リレーショナル・データベース名 列がストリングでない場合は、NULL 値が入ります。
CHARACTER_SET_SCHEMA	VARCHAR(128) NULL 可能	文字セットのスキーマ名。 'SYSIBM' が入ります。 列がストリングでない場合は、NULL 値が入ります。
CHARACTER_SET_NAME	VARCHAR(128) NULL 可能	文字セット名。 列がストリングでない場合は、NULL 値が入ります。
COLLATION_CATALOG	VARCHAR(128) NULL 可能	リレーショナル・データベース名 列がストリングでない場合は、NULL 値が入ります。
COLLATION_SCHEMA	VARCHAR(128) NULL 可能	照合のスキーマ。 SYSIBM が戻されます。 列がストリングでない場合は、NULL 値が入ります。
COLLATION_NAME	VARCHAR(128) NULL 可能	照合名。 IBMBINARY が戻されます。 列がストリングでない場合は、NULL 値が入ります。
NUMERIC_PRECISION	INTEGER NULL 可能	数値パラメーターすべての精度。 **注:** この列では、すべての数値データ・タイプ (単精度および倍精度の浮動小数点数を含む) の精度を指定します。NUMERIC_PRECISION_RADIX 列は、この列の値が 2 進数であるか、または 10 進数であるかを示します。 パラメーターが数値パラメーターでない場合は、NULL 値が入ります。
NUMERIC_PRECISION_RADIX	INTEGER NULL 可能	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 パラメーターが数値パラメーターでない場合は、NULL 値が入ります。
NUMERIC_SCALE	INTEGER NULL 可能	数値データの位取り。 10 進数、数値、または 2 進数のパラメーターでない場合は、NULL 値が入ります。
DATETIME_PRECISION	INTEGER NULL 可能	日付、時刻、またはタイム・スタンプの小数部分。 **0** データ・タイプが DATE および TIME の場合 **0-12** データ・タイプが TIMESTAMP の場合 (小数秒) パラメーターが日付、時刻、またはタイム・スタンプでない場合は、NULL 値が入ります。
INTERVAL_TYPE	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。

表 243. PARAMETERS ビュー (続き)

列名	データ・タイプ	説明
INTERVAL_PRECISION	INTEGER NULL 可能	予約済み。 NULL 値が入ります。
UDT_CATALOG	VARCHAR(128) NULL 可能	特殊タイプまたは配列タイプの場合は、リレーショナル・データベース名。 特殊タイプまたは配列タイプでない場合には、NULL 値が入ります。
UDT_SCHEMA	VARCHAR(128) NULL 可能	特殊タイプまたは配列タイプの場合は、スキーマの名前。 特殊タイプまたは配列タイプでない場合には、NULL 値が入ります。
UDT_NAME	VARCHAR(128) NULL 可能	特殊タイプまたは配列タイプの名前。 特殊タイプまたは配列タイプでない場合には、NULL 値が入ります。
SCOPE_CATALOG	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
SCOPE_SCHEMA	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
SCOPE_NAME	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
MAXIMUM_CARDINALITY	BIGINT NULL 可能	このパラメーターが配列タイプである場合は、その配列タイプの最大カーディナリティー。 このパラメーターが配列タイプでない場合は、NULL 値が入ります。
DTD_IDENTIFIER	VARCHAR(128) NULL 可能	パラメーターの固有の内部 ID。
DECLARED_DATA_TYPE	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
DECLARED_NUMERIC_PRECISION	INTEGER NULL 可能	予約済み。 NULL 値が入ります。
DECLARED_NUMERIC_SCALE	INTEGER NULL 可能	予約済み。 NULL 値が入ります。
PARAMETER_DEFAULT	DBCLOB(64K) CCSID 1200 NULL 可能	パラメーターのデフォルト値が存在する場合、パラメーターのデフォルト値を計算するために使用される式ストリング。デフォルト値がヌル値の場合、式ストリングはキーワード NULL です。 パラメーターにデフォルトがない場合は、NULL 値が入ります。

REFERENTIAL_CONSTRAINTS

REFERENTIAL_CONSTRAINTS ビューには、各参照制約ごとに、行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *244. REFERENTIAL_CONSTRAINTS* ビュー

列名	データ・タイプ	説明
CONSTRAINT_CATALOG	VARCHAR(128)	リレーショナル・データベース名
CONSTRAINT_SCHEMA	VARCHAR(128)	該当の制約が入っているスキーマの名前。
CONSTRAINT_NAME	VARCHAR(128)	制約の名前。
UNIQUE_CONSTRAINT_CATALOG	VARCHAR(128)	参照制約によって参照された固有制約が入っているリレーショナル・データベースの名前。
UNIQUE_CONSTRAINT_SCHEMA	VARCHAR(128)	参照制約によって参照された固有制約が入っている SQL のスキーマの名前。
UNIQUE_CONSTRAINT_NAME	VARCHAR(128)	参照制約によって参照された固有制約の名前。
MATCH_OPTION	VARCHAR(7)	予約済み。 'NONE' が入ります。
UPDATE_RULE	VARCHAR(11)	UPDATE の規則。 • NO ACTION • RESTRICT
DELETE_RULE	VARCHAR(11)	DELETE の規則。 • NO ACTION • CASCADE • SET NULL • SET DEFAULT • RESTRICT
COLUMN_COUNT	INTEGER	制約の列の数。

ROUTINES

ROUTINES ビューには、各ルーチンごとに行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *245. ROUTINES* ビュー

列名	データ・タイプ	説明
SPECIFIC_CATALOG	VARCHAR(128)	リレーショナル・データベース名
SPECIFIC_SCHEMA	VARCHAR(128)	ルーチン (関数) インスタンスのスキーマ名。
SPECIFIC_NAME	VARCHAR(128)	ルーチンの特定名。
ROUTINE_CATALOG	VARCHAR(128)	リレーショナル・データベース名
ROUTINE_SCHEMA	VARCHAR(128)	該当のルーチンが入っている SQL スキーマの名前。
ROUTINE_NAME	VARCHAR(128)	ルーチンの名前。
ROUTINE_TYPE	VARCHAR(15)	ルーチンのタイプ。 **PROCEDURE** これはプロシージャーです。 **FUNCTION** これは関数です。 **INSTANCE METHOD** これは特殊タイプ用に作成された組み込みデータ・タイプ関数です。
MODULE_CATALOG	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
MODULE_SCHEMA	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
MODULE_NAME	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
UDT_CATALOG	VARCHAR(128) NULL 可能	リレーショナル・データベース名。 これが INSTANCE METHOD でない場合は、NULL 値が入ります。
UDT_SCHEMA	VARCHAR(128) NULL 可能	この関数に関連した特殊タイプが入っている SQL スキーマの名前。 これが INSTANCE METHOD でない場合は、NULL 値が入ります。
UDT_NAME	VARCHAR(128) NULL 可能	この関数に関連した特殊タイプの名前。 これが INSTANCE METHOD でない場合は、NULL 値が入ります。

表 245. ROUTINES ビュー (続き)

列名	データ・タイプ	説明
DATA_TYPE	VARCHAR(128)	関数の結果のタイプ。 NULL 可能 **BIGINT** 大整数 **INTEGER** 長整数 **SMALLINT** 短整数 **DECIMAL** パック 10 進数 **NUMERIC** ゾーン 10 進数 **DOUBLE PRECISION** 浮動小数点数; DOUBLE PRECISION **REAL** 浮動小数点数; REAL **DECFLOAT** 10 進浮動小数点数 **CHARACTER** 固定長文字ストリング **CHARACTER VARYING** 可変長文字ストリング **CHARACTER LARGE OBJECT** 文字ラージ・オブジェクト・ストリング **GRAPHIC** 固定長グラフィック・ストリング **GRAPHIC VARYING** 可変長グラフィック・ストリング **DOUBLE-BYTE CHARACTER LARGE OBJECT** 2 バイト文字ラージ・オブジェクト・ストリング **NATIONAL CHARACTER** 国別文字 **NATIONAL CHARACTER VARYING** 可変長国別文字 **NATIONAL CHARACTER LARGE OBJECT** 国別文字ラージ・オブジェクト **BINARY** 固定長バイナリー・ストリング **BINARY VARYING** 可変長バイナリー・ストリング **BINARY LARGE OBJECT** バイナリー・ラージ・オブジェクト・ストリング **DATE** 日付 **TIME** 時刻 **TIMESTAMP** タイム・スタンプ **DATALINK** データ・リンク **ROWID** 行 ID **XML** XML **USER-DEFINED** 特殊タイプまたは配列タイプ これがスカラー関数でない場合は、NULL 値が入ります。
CHARACTER_MAXIMUM_LENGTH	INTEGER	データ・タイプが 2 進数、文字、グラフィック・ストリング、および XML の場合は、関数の結果ストリングの最大長。 NULL 可能 これがスカラー関数でない場合、またはパラメーターがストリングでない場合は、NULL 値が入ります。
CHARACTER_OCTET_LENGTH	INTEGER	データ・タイプが 2 進数、文字、グラフィック・ストリング、および XML の場合は、関数の結果ストリングのバイト数。 NULL 可能 これがスカラー関数でない場合、またはパラメーターがストリングでない場合は、NULL 値が入ります。
CHARACTER_SET_CATALOG	VARCHAR(128)	関数の結果のリレーショナル・データベース名。 NULL 可能 これがスカラー関数でない場合、または結果がストリングでない場合は、NULL 値が入ります。
CHARACTER_SET_SCHEMA	VARCHAR(128)	関数の結果の文字セットのスキーマ名。 'SYSIBM' が入ります。 NULL 可能 これがスカラー関数でない場合、または結果がストリングでない場合は、NULL 値が入ります。

表 245. ROUTINES ビュー (続き)

列名	データ・タイプ	説明
CHARACTER_SET_NAME	VARCHAR(128)	関数の結果の文字セット名。
	NULL 可能	これがスカラー関数でない場合、または結果がストリングでない場合は、NULL 値が入ります。
COLLATION_CATALOG	VARCHAR(128)	関数の結果のリレーショナル・データベース名。
	NULL 可能	これがスカラー関数でない場合、または結果がストリングでない場合は、NULL 値が入ります。
COLLATION_SCHEMA	VARCHAR(128)	関数の結果の照合のスキーマ。 SYSIBM が戻されます。
	NULL 可能	これがスカラー関数でない場合、または結果がストリングでない場合は、NULL 値が入ります。
COLLATION_NAME	VARCHAR(128)	関数の結果の照合名。 IBMBINARY が戻されます。
	NULL 可能	これがスカラー関数でない場合、または結果がストリングでない場合は、NULL 値が入ります。
NUMERIC_PRECISION	INTEGER	関数の結果の精度。 **注:** この列では、すべての数値データ・タイプ (単精度および倍精度の浮動小数点数を含む) の精度を指定します。 NUMERIC_PRECISION_RADIX 列は、この列の値が 2 進数であるか、または 10 進数であるかを示します。
	NULL 可能	これがスカラー関数でない場合、または結果が数値でない場合は、NULL 値が入ります。
NUMERIC_PRECISION_RADIX	INTEGER	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。
	NULL 可能	**2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 これがスカラー関数でない場合、または結果が数値でない場合は、NULL 値が入ります。
NUMERIC_SCALE	INTEGER	関数の結果である数値の位取り。
	NULL 可能	これがスカラー関数でない場合、または結果が数値でない場合は、NULL 値が入ります。
DATETIME_PRECISION	INTEGER	関数の結果である日付、時刻、またはタイム・スタンプの小数部分。
	NULL 可能	**0** データ・タイプが DATE および TIME の場合 **0-12** データ・タイプが TIMESTAMP の場合 (小数秒) これがスカラー関数でない場合、または結果が日付、時刻、またはタイム・スタンプでない場合は、NULL 値が入ります。
INTERVAL_TYPE	VARCHAR(128)	予約済み。 NULL 値が入ります。
	NULL 可能	
INTERVAL_PRECISION	INTEGER	予約済み。 NULL 値が入ります。
	NULL 可能	
TYPE_UDT_CATALOG	VARCHAR(128)	関数の結果が特殊タイプまたは配列タイプである場合は、リレーショナル・データベース名。
	NULL 可能	これがスカラー関数でない場合、または結果が特殊タイプや配列タイプでない場合は、NULL 値が入ります。
TYPE_UDT_SCHEMA	VARCHAR(128)	関数の結果が特殊タイプまたは配列タイプである場合は、スキーマの名前。
	NULL 可能	これがスカラー関数でない場合、または結果が特殊タイプや配列タイプでない場合は、NULL 値が入ります。
TYPE_UDT_NAME	VARCHAR(128)	関数の結果が特殊タイプまたは配列タイプである場合は、その特殊タイプの名前。
	NULL 可能	これがスカラー関数でない場合、または結果が特殊タイプや配列タイプでない場合は、NULL 値が入ります。
SCOPE_CATALOG	VARCHAR(128)	予約済み。 NULL 値が入ります。
	NULL 可能	
SCOPE_SCHEMA	VARCHAR(128)	予約済み。 NULL 値が入ります。
	NULL 可能	
SCOPE_NAME	VARCHAR(128)	予約済み。 NULL 値が入ります。
	NULL 可能	
MAXIMUM_CARDINALITY	BIGINT	関数の結果が配列タイプである場合は、その配列タイプの最大カーディナリティー。
	NULL 可能	これが配列タイプでない場合は、NULL 値が入ります。
DTD_IDENTIFIER	VARCHAR(128)	関数の結果の固有の内部 ID。
	NULL 可能	
ROUTINE_BODY	VARCHAR(8)	ルーチン本体のタイプ: **EXTERNAL** これは外部ルーチンです。 **SQL** これは SQL ルーチンです。

表 245. ROUTINES ビュー (続き)

列名	データ・タイプ	説明
ROUTINE_DEFINITION	DBCLOB(2M) CCSID 13488 NULL 可能	これが SQL ルーチンの場合、この列は SQL ルーチン本体を含みます。 これが難読化されたルーチンの場合、テキストは WRAPPED キーワードで始まり、その後に、エンコードされた形式のステートメント・テキストが続きます。 これが SQL ルーチンでない場合、または切り捨てなければルーチン本体をこの列に収容できない場合は、NULL 値が入ります。
EXTERNAL_NAME	VARCHAR(279) NULL 可能	これが外部ルーチンである場合は、この列は外部プログラム名を識別します。 • REXX の場合は、外部プログラム名は、スキーマ名/ソース・ファイル名 (メンバー名) です。 • ILE サービス・プログラムの場合、外部プログラム名はスキーマ名/サービス・プログラム名 (入り口名) です。 • Java プログラムの場合、外部プログラム名はオプションの jar-id の後に完全修飾クラス名!メソッド名 または 完全修飾クラス名.メソッド名 が続きます。 • その他のすべての言語では、外部プログラム名は、スキーマ名/プログラム名 です。 これがシステム生成の関数であるか、組み込み関数をソースとする関数である場合は、NULL 値が入ります。
EXTERNAL_LANGUAGE	VARCHAR(8) NULL 可能	これが外部ルーチンである場合は、この列は外部プログラム名を識別します。 **C** 外部プログラムは C で作成されます。 **C++** 外部プログラムは C++ で作成されます。 **CL** 外部プログラムは CL で作成されます。 **COBOL** 外部プログラムは COBOL で作成されます。 **COBOLLE** 外部プログラムは ILE COBOL で作成されます。 **FORTRAN** 外部プログラムは FORTRAN で作成されます。 **JAVA** 外部プログラムは JAVA で作成されます。 **PLI** 外部プログラムは PL/I で作成されます。 **REXX** 外部プログラムは REXX プロシージャーです。 **RPG** 外部プログラムは RPG で作成されます。 **RPGLE** 外部プログラムは ILE RPG で作成されます。 これが外部ルーチンでない場合は、NULL 値が入ります。
PARAMETER_STYLE	VARCHAR(18) NULL 可能	これが外部ルーチンである場合は、この列はパラメーターのスタイル (呼び出し規則) を識別します。 **DB2GENERAL** これは DB2GENERAL 呼び出し規則です。 **DB2SQL** これは DB2SQL 呼び出し規則です。 **GENERAL** これは GENERAL 呼び出し規則です。 **JAVA** これは JAVA 呼び出し規則です。 **GENERAL WITH NULLS** これは GENERAL WITH NULLS 呼び出し規則です。 **SQL** これは SQL 標準呼び出し規則です。 これが外部ルーチンでない場合は、NULL 値が入ります。
IS_DETERMINISTIC	VARCHAR(3)	この列はルーチンが deterministic であるかどうかを識別します。 つまり、同じ引数のルーチンに対する呼び出しが、常に同じ結果を戻すかどうかを識別します。 **NO** ルーチンは deterministic ではありません。 **YES** ルーチンは deterministic です。
SQL_DATA_ACCESS	VARCHAR(17)	この列は、ルーチンに SQL が含まれているか、およびルーチンがデータの読み取りまたは変更を行うかを識別します。 **NO SQL** ルーチンは SQL ステートメントを含みません。 **CONTAINS SQL** ルーチンは SQL ステートメントを含みます。 **READS SQL DATA** ルーチンは、おそらく表またはビューからデータを読み取ります。 **MODIFIES SQL DATA** ルーチンは、おそらく表またはビュー内のデータを変更するか、SQL DDL ステートメントを発行します。

表 245. ROUTINES ビュー (続き)

列名	データ・タイプ	説明
IS_NULL_CALL	VARCHAR(3) NULL 可能	入力パラメーターが NULL 値である場合に、関数を呼び出す必要があるかどうかを識別します。 **NO** この関数は、入力パラメーターが NULL 値の場合に呼び出す必要はありません。これがスカラー関数の場合は、いずれかのオペランドが NULL であれば、この関数の結果は暗黙的に NULL になります。これが表関数の場合は、いずれかのオペランドが NULL 値であれば、この関数の結果は空の表になります。 **YES** この関数は、入力オペランドが NULL でも呼び出す必要があります。 これが関数でない場合は、NULL 値が入ります。
SQL_PATH	VARCHAR(3483) NULL 可能	これが SQL ルーチンの場合、この列はパスを識別します。 これが SQL ルーチンでない場合は、NULL 値が入ります。
SCHEMA_LEVEL_ROUTINE	VARCHAR(3)	予約済み。 'YES' が入ります。
MAX_DYNAMIC_RESULT_SETS	SMALLINT	戻される結果セットの最大数を識別します。 0 は結果セットがないことを示します。
IS_USER_DEFINED_CAST	VARCHAR(3) NULL 可能	この関数が、特殊タイプの作成時に作成されたキャスト関数であるかどうかを識別します。 **NO** この関数はキャスト関数ではありません。 **YES** この関数はキャスト関数です。 ルーチンが関数でない場合は、NULL 値が入ります。
IS_IMPLICITLY_INVOCABLE	VARCHAR(3) NULL 可能	この関数が、特殊タイプの作成時に作成されたキャスト関数であって、暗黙的に呼び出せるかどうかを判別します。 **NO** この関数はキャスト関数ではありません。 **YES** この関数はキャスト関数であって、暗黙的に呼び出すことができます。 ルーチンが関数でない場合は、NULL 値が入ります。
SECURITY_TYPE	VARCHAR(22) NULL 可能	予約済み。これが外部ルーチンである場合は、'IMPLEMENTATION DEFINED' が入ります。 ルーチンが外部ルーチンでない場合は、NULL 値が入ります。
TO_SQL_SPECIFIC_CATALOG	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
TO_SQL_SPECIFIC_SCHEMA	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
TO_SQL_SPECIFIC_NAME	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
AS_LOCATOR	VARCHAR(3) NULL 可能	結果がロケーターとして指定されたかどうかを識別します。 **NO** パラメーターはロケーターとして指定されませんでした。 **YES** パラメーターはロケーターとして指定されました。 これがスカラー関数でない場合は、NULL 値が入ります。
CREATED	TIMESTAMP	ルーチンが作成されたときのタイム・スタンプを識別します。
LAST_ALTERED	TIMESTAMP NULL 可能	ルーチンが最後に変更されたときのタイム・スタンプ。ルーチンが一度も変更されていない場合には、NULL が入ります。
NEW_SAVEPOINT_LEVEL	VARCHAR(3) NULL 可能	ルーチンが新しいセーブポイント・レベルを開始するかどうかを示します。 **NO** プロシージャーの呼び出し時に新しいセーブポイント・レベルを開始しません。 **YES** プロシージャーの呼び出し時に新しいセーブポイント・レベルを開始します。 これが関数でない場合は、NULL 値が入ります。
IS_UDT_DEPENDENT	VARCHAR(3)	ルーチンが UDT に依存しているかどうかを示します。 **NO** ルーチンは UDT に依存しません。 **YES** ルーチンは UDT に依存します。

表 245. ROUTINES ビュー (続き)

列名	データ・タイプ	説明
RESULT_CAST_FROM_DATA_TYPE	VARCHAR(128) NULL 可能	パラメーターのタイプ: **BIGINT** 大整数 **INTEGER** 長整数 **SMALLINT** 短整数 **DECIMAL** パック 10 進数 **NUMERIC** ゾーン 10 進数 **DOUBLE PRECISION** 浮動小数点数; DOUBLE PRECISION **REAL** 浮動小数点数; REAL **DECFLOAT** 10 進浮動小数点数 **CHARACTER** 固定長文字ストリング **CHARACTER VARYING** 可変長文字ストリング **CHARACTER LARGE OBJECT** 文字ラージ・オブジェクト・ストリング **GRAPHIC** 固定長グラフィック・ストリング **GRAPHIC VARYING** 可変長グラフィック・ストリング **DOUBLE-BYTE CHARACTER LARGE OBJECT** 2 バイト文字ラージ・オブジェクト・ストリング **NATIONAL CHARACTER** 国別文字 **NATIONAL CHARACTER VARYING** 可変長国別文字 **NATIONAL CHARACTER LARGE OBJECT** 国別文字ラージ・オブジェクト **BINARY** 固定長バイナリー・ストリング **BINARY VARYING** 可変長バイナリー・ストリング **BINARY LARGE OBJECT** バイナリー・ラージ・オブジェクト・ストリング **DATE** 日付 **TIME** 時刻 **TIMESTAMP** タイム・スタンプ **DATALINK** データ・リンク **ROWID** 行 ID **XML** XML **USER-DEFINED** 特殊タイプ
RESULT_CAST_AS_LOCATOR	VARCHAR(3) NULL 可能	結果をロケーターからキャストするかどうかを示します。 **NO** 結果をロケーターからキャストしません。 **YES** 結果をロケーターからキャストします。
RESULT_CAST_CHAR_MAX_LENGTH	INTEGER NULL 可能	データ・タイプが 2 進ストリング、文字ストリング、グラフィック・ストリング、および XML の場合は、ストリングの最大長。 パラメーターがストリングでない場合は、NULL 値が入ります。
RESULT_CAST_CHAR_OCTET_LENGTH	INTEGER NULL 可能	データ・タイプが 2 進ストリング、文字ストリング、グラフィック・ストリング、および XML の場合は、バイト数。 パラメーターがストリングでない場合は、NULL 値が入ります。
RESULT_CAST_CHAR_SET_CATALOG	VARCHAR(128) NULL 可能	リレーショナル・データベース名 列がストリングでない場合は、NULL 値が入ります。

表 245. ROUTINES ビュー (続き)

列名	データ・タイプ	説明
RESULT_CAST_CHAR_SET_SCHEMA	VARCHAR(128)	文字セットのスキーマ名。 'SYSIBM' が入ります。
	NULL 可能	列がストリングでない場合は、NULL 値が入ります。
RESULT_CAST_CHAR_SET_NAME	VARCHAR(128)	文字セット名。
	NULL 可能	列がストリングでない場合は、NULL 値が入ります。
RESULT_CAST_COLLATION_CATALOG	VARCHAR(128)	リレーショナル・データベース名
	NULL 可能	列がストリングでない場合は、NULL 値が入ります。
RESULT_CAST_COLLATION_SCHEMA	VARCHAR(128)	照合のスキーマ。 SYSIBM が戻されます。
	NULL 可能	列がストリングでない場合は、NULL 値が入ります。
RESULT_CAST_COLLATION_NAME	VARCHAR(128)	照合名。 IBMBINARY が戻されます。
	NULL 可能	列がストリングでない場合は、NULL 値が入ります。
RESULT_CAST_NUMERIC_PRECISION	INTEGER	数値パラメーターすべての精度。
	NULL 可能	**注:** この列では、すべての数値データ・タイプ (単精度および倍精度の浮動小数点数を含む) の精度を指定します。 NUMERIC_PRECISION_RADIX 列は、この列の値が 2 進数であるか、または 10 進数であるかを示します。
		パラメーターが数値パラメーターでない場合は、NULL 値が入ります。
RESULT_CAST_NUMERIC_RADIX	INTEGER	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。
	NULL 可能	**2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。
		10 10 進数: 他の数値タイプはすべて 10 進数で指定されます。
		パラメーターが数値パラメーターでない場合は、NULL 値が入ります。
RESULT_CAST_NUMERIC_SCALE	INTEGER	数値データの位取り。
	NULL 可能	10 進数、数値、または 2 進数のパラメーターでない場合は、NULL 値が入ります。
RESULT_CAST_DATETIME_PRECISION	INTEGER	日付、時刻、またはタイム・スタンプの小数部分。
	NULL 可能	**0** データ・タイプが DATE および TIME の場合
		0-12 データ・タイプが TIMESTAMP の場合 (小数秒)
		パラメーターが日付、時刻、またはタイム・スタンプでない場合は、NULL 値が入ります。
RESULT_CAST_INTERVAL_TYPE	VARCHAR(128)	予約済み。 NULL 値が入ります。
	NULL 可能	
RESULT_CAST_INTERVAL_PRECISION	INTEGER	予約済み。 NULL 値が入ります。
	NULL 可能	
RESULT_CAST_TYPE_UDT_CATALOG	VARCHAR(128)	特殊タイプの場合は、リレーショナル・データベース名。
	NULL 可能	特殊タイプでない場合には、NULL 値が入ります。
RESULT_CAST_TYPE_UDT_SCHEMA	VARCHAR(128)	特殊タイプの場合は、スキーマの名前。
	NULL 可能	特殊タイプでない場合には、NULL 値が入ります。
RESULT_CAST_TYPE_UDT_NAME	VARCHAR(128)	特殊タイプの名前。
	NULL 可能	特殊タイプでない場合には、NULL 値が入ります。
RESULT_CAST_SCOPE_CATALOG	VARCHAR(128)	予約済み。 NULL 値が入ります。
	NULL 可能	
RESULT_CAST_SCOPE_SCHEMA	VARCHAR(128)	予約済み。 NULL 値が入ります。
	NULL 可能	
RESULT_CAST_SCOPE_NAME	VARCHAR(128)	予約済み。 NULL 値が入ります。
	NULL 可能	
RESULT_CAST_MAX_CARDINALITY	INTEGER	予約済み。 NULL 値が入ります。
	NULL 可能	
RESULT_CAST_DTD_IDENTIFIER	VARCHAR(128)	パラメーターの固有の内部 ID。
	NULL 可能	
DECLARED_DATA_TYPE	VARCHAR(128)	予約済み。 NULL 値が入ります。
	NULL 可能	
DECLARED_NUMERIC_PRECISION	INTEGER	予約済み。 NULL 値が入ります。
	NULL 可能	

表 245. ROUTINES ビュー (続き)

列名	データ・タイプ	説明
DECLARED_NUMERIC_SCALE	INTEGER NULL 可能	予約済み。 NULL 値が入ります。
RESULT_CAST_FROM_DECLARED_DATA_TYPE	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
RESULT_CAST_DECLARED_NUMERIC_PRECISION	INTEGER NULL 可能	予約済み。 NULL 値が入ります。
RESULT_CAST_DECLARED_NUMERIC_SCALE	INTEGER NULL 可能	予約済み。 NULL 値が入ります。

ROUTINE_PRIVILEGES

ROUTINE_PRIVILEGES ビューには、ルーチンに対して認可された各特権ごとに、行が 1 つずつ入ります。このカタログ・ビューを使用して、特定ユーザーが特定のルーチンに対する権限を持っているかどうかを判別することはできないので、注意してください。なぜなら、ルーチンを使用するための特権は、グループ・ユーザー・プロファイルまたは特殊権限 (*ALLOBJ など) を通じて獲得できるからです。

次の表は、ビューの列について説明しています。

表 *246. ROUTINE_PRIVILEGES* ビュー

列名	データ・タイプ	説明
GRANTOR	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
GRANTEE	VARCHAR(128)	特権を認可する対象のユーザー・プロファイル。
SPECIFIC_CATALOG	VARCHAR(128)	リレーショナル・データベース名
SPECIFIC_SCHEMA	VARCHAR(128)	ルーチン (関数) インスタンスのスキーマ名。
SPECIFIC_NAME	VARCHAR(128)	ルーチンの特定名。
ROUTINE_CATALOG	VARCHAR(128)	リレーショナル・データベース名
ROUTINE_SCHEMA	VARCHAR(128)	該当のルーチンが入っている SQL スキーマの名前。
ROUTINE_NAME	VARCHAR(128)	ルーチンの名前。
PRIVILEGE_TYPE	VARCHAR(10)	認可される特権 : **ALTER** ルーチンを変更する特権。 **EXECUTE** ルーチンを実行する特権。
IS_GRANTABLE	VARCHAR(3)	特権を他のユーザーに認可できるかどうかを示します。 **NO** 特権は認可できません。 **YES** 特権を認可できます。

SCHEMATA

SCHEMATA ビューには、各スキーマごとに行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *247. SCHEMATA* ビュー

列名	データ・タイプ	説明
CATALOG_NAME	VARCHAR(128)	リレーショナル・データベース名
SCHEMA_NAME	VARCHAR(128)	スキーマの名前
SCHEMA_OWNER	VARCHAR(128)	スキーマの所有者
DEFAULT_CHARACTER_SET_CATALOG	VARCHAR(128)	リレーショナル・データベース名
DEFAULT_CHARACTER_SET_SCHEMA	VARCHAR(128)	デフォルト文字セットのスキーマ名。 'SYSIBM' が入ります。
DEFAULT_CHARACTER_SET_NAME	VARCHAR(128)	デフォルト文字セット名。
SQL_PATH	VARCHAR(4096)	予約済み。 NULL 値が入ります。

NULL 可能

SEQUENCES

SEQUENCES ビューには、各シーケンスごとに行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *248. SEQUENCES* ビュー

列名	データ・タイプ	説明
SEQUENCE_CATALOG	VARCHAR(128)	リレーショナル・データベース名
SEQUENCE_SCHEMA	VARCHAR(128)	シーケンスを含んでいる SQL スキーマ
SEQUENCE_NAME	VARCHAR(128)	シーケンスの名前
DATA_TYPE	VARCHAR(128)	シーケンスのタイプ
NUMERIC_PRECISION	INTEGER	シーケンス・タイプの精度
NUMERIC_PRECISION_RADIX	INTEGER	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **10** すべてのシーケンス・データ・タイプが 10 進数字で指定されます。
NUMERIC_SCALE	INTEGER	すべてのシーケンス・データ・タイプが位取り 0 を使用します。
START_VALUE	DECIMAL(63,0)	シーケンスの開始値
MINIMUM_VALUE	DECIMAL(63,0)	シーケンスの最小値
MAXIMUM_VALUE	DECIMAL(63,0)	シーケンスの最大値
INCREMENT	INTEGER	シーケンスの増分値
CYCLE_OPTION	VARCHAR(3)	シーケンス値が最小値または最大値に達した後も、値の生成を続けるかどうかを識別します。 **NO** 値の生成は継続されません **YES** 値の生成は継続されます
DECLARED_DATA_TYPE	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
DECLARED_NUMERIC_PRECISION	INTEGER NULL 可能	予約済み。 NULL 値が入ります。
DECLARED_NUMERIC_SCALE	INTEGER NULL 可能	予約済み。 NULL 値が入ります。

SQL_FEATURES

SQL_FEATURES 表には、データベース・マネージャーでサポートされている各フィーチャーごとに、行が 1 つずつ入ります。

次の表は、この表の列について説明しています。

表 *249. SQL_FEATURES* 表

列名	データ・タイプ	説明
FEATURE_ID	VARCHAR(7) NULL 可能	ANS および ISO のフィーチャー ID。
FEATURE_NAME	VARCHAR(128)	ANS および ISO のフィーチャーの名前。
SUB_FEATURE_ID	VARCHAR(7) NULL 可能	ANS および ISO のサブフィーチャー ID。
SUB_FEATURE_NAME	VARCHAR(256)	ANS および ISO のサブフィーチャーの名前。
IS_SUPPORTED	VARCHAR(3)	該当のフィーチャーがサポートされているかどうかを示します。 **YES** このフィーチャーはサポートされています。 **NO** このフィーチャーはサポートされていません。
IS_VERIFIED_BY	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
COMMENTS	VARCHAR(2000) NULL 可能	予約済み。 NULL 値が入ります。

SQL_LANGUAGES

SQL_LANGUAGES 表には、適合性が要求される SQL 言語バインディングおよびプログラム言語ごとに行が 1 つずつ入ります。

次の表は、SQL_LANGUAGES 表の列について説明しています。

表 *250. SQL_LANGUAGES* 表

列名	データ・タイプ	説明
SQL_LANGUAGE_SOURCE	VARCHAR(254)	標準の名前。
SQL_LANGUAGE_YEAR	VARCHAR(254)	標準が承認された年。
SQL_LANGUAGE_CONFORMANCE	VARCHAR(254) NULL 可能	適合性のレベル。 **2** 1987 年および 1989 年の標準の場合、レベル 2 の適合性が要求されることを示します。 **ENTRY** 1992 年の標準の場合、エントリー・レベルの適合性が要求されることを示します。 **CORE** 1999 年の標準の場合、コア・レベルの適合性が要求されることを示します。 適合性がまだ要求されていない場合は、NULL 値が入ります。
SQL_LANGUAGE_INTEGRITY	VARCHAR(254) NULL 可能	整合性機能のサポート。 **YES** 整合性に対して適合性が要求されます。 **NO** 整合性に対して適合性は要求されません。 標準に単独の整合性フィーチャーがない場合は、NULL 値が入ります。
SQL_LANGUAGE_IMPLEMENTATION	VARCHAR(254) NULL 可能	予約済み。 NULL 値が入ります。
SQL_LANGUAGE_BINDING_STYLE	VARCHAR(254)	SQL 言語のバインディングのスタイル。 **EMBEDDED** 以下の言語に関する組み込み SQL のサポート SQL_LANGUAGE_PROGRAMMING_LANG **DIRECT** DIRECT SQL がサポートされます (例えば、対話式 SQL)。 **CLI** 以下の言語に関する CLI のサポート SQL_LANGUAGE_PROGRAMMING_LANG
SQL_LANGUAGE_PROGRAMMING_LANG	VARCHAR(254) NULL 可能	EMBEDDED または CLI によってサポートされている言語。 **C** C 言語がサポートされます。 **COBOL** COBOL 言語がサポートされます。 **PLI** PL/I 言語がサポートされます。 SQL_LANGUAGE_BINDING_STYLE が DIRECT の場合は、NULL 値が入ります。

SQL_SIZING

SQL_SIZING 表には、データベース・マネージャーでサポートされている各限度ごとに、行が 1 つずつ入ります。

次の表は、この表の列について説明しています。

表 *251. SQL_SIZING* 表

列名	データ・タイプ	説明
SIZING_ID	INTEGER	ANS および ISO のサイジング ID。
SIZING_NAME	VARCHAR(128)	ANS および ISO のサイジングの名前。
SUPPORTED_VALUE	DECIMAL(21) NULL 可能	サイジング限度を示します。 サイジング限度が適用されない場合は、NULL 値が入ります。
COMMENTS	VARCHAR(2000) NULL 可能	予約済み。 NULL 値が入ります。

TABLE_CONSTRAINTS

TABLE_CONSTRAINTS ビューには、各制約ごとに行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *252. TABLE_CONSTRAINTS* ビュー

列名	データ・タイプ	説明
CONSTRAINT_CATALOG	VARCHAR(128)	リレーショナル・データベース名
CONSTRAINT_SCHEMA	VARCHAR(128)	該当の制約が入っているスキーマの名前。
CONSTRAINT_NAME	VARCHAR(128)	制約の名前。
TABLE_CATALOG	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEMA	VARCHAR(128)	該当の表が入っているスキーマの名前。
TABLE_NAME	VARCHAR(128)	該当の制約が作成される表の名前。
CONSTRAINT_TYPE	VARCHAR(11)	制約のタイプ • CHECK • UNIQUE • PRIMARY KEY • FOREIGN KEY
IS_DEFERRABLE	VARCHAR(3)	制約の検査が据え置きできるかどうかを示します。 'NO' が入ります。
INITIALLY_DEFERRED	VARCHAR(3)	制約が初期据え置きとして定義されたかどうかを示します。 'NO' が入ります。

TABLE_PRIVILEGES

TABLE_PRIVILEGES ビューには、表に対して認可された各特権ごとに、行が 1 つずつ入ります。このカタログ・ビューを使用して、特定ユーザーが特定の表に対する権限を持っているかどうかを判別することはできないので、注意してください。なぜなら、表を使用するための特権は、グループ・ユーザー・プロファイルまたは特殊権限 (*ALLOBJ など) を通じて獲得できるからです。

次の表は、ビューの列について説明しています。

表 *253. TABLE_PRIVILEGES* ビュー

列名	データ・タイプ	説明
GRANTOR	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
GRANTEE	VARCHAR(128)	特権を認可する対象のユーザー・プロファイル。
TABLE_CATALOG	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEMA	VARCHAR(128)	該当の表が入っている SQL のスキーマの名前。
TABLE_NAME	VARCHAR(128)	表の名前。
PRIVILEGE_TYPE	VARCHAR(10)	認可される特権 : **ALTER** 表を変更する特権。 **DELETE** 表から行を削除する特権。 **INDEX** 表の索引を作成する特権。 **INSERT** 表に行を挿入する特権。 **REFERENCES** 参照制約の中で表を参照する特権。 **SELECT** 表から行を選択する特権。 **UPDATE** 表を更新する特権。
IS_GRANTABLE	VARCHAR(3)	特権を他のユーザーに認可できるかどうかを示します。 **NO** 特権は認可できません。 **YES** 特権を認可できます。

TABLES

TABLES ビューには、各表、ビュー、および別名ごとに、行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *254. TABLES* ビュー

列名	データ・タイプ	説明
TABLE_CATALOG	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEMA	VARCHAR(128)	該当の表、ビュー、または別名を含む SQL スキーマの名前。
TABLE_NAME	VARCHAR(128)	その表、ビュー、または別名の名前。
TABLE_TYPE	VARCHAR(24)	表のタイプを識別します。 **ALIAS** 表は別名です。 **BASE TABLE** 表は、SQL 表または物理ファイルです。 **MATERIALIZED QUERY TABLE** オブジェクトはマテリアライズ照会表です。 **VIEW** 表は、SQL ビューまたは論理ファイルです。
SELF_REFERENCING_COLUMN_NAME	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
REFERENCE_GENERATION	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
USER_DEFINED_TYPE_CATALOG	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
USER_DEFINED_TYPE_SCHEMA	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
USER_DEFINED_TYPE_NAME	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
IS_INSERTABLE_INTO	VARCHAR(3)	表で INSERT を使用できるかどうかを識別します。 **NO** この表では INSERT は使用できません。 **YES** この表では INSERT を使用できます。

UDT_PRIVILEGES

UDT_PRIVILEGES ビューには、タイプに対して認可された各特権ごとに、行が 1 つずつ入ります。このカタログ・ビューを使用して、特定ユーザーが特定のタイプに対する権限を持っているかどうかを判別することはできないので、注意してください。なぜなら、タイプを使用するための特権は、グループ・ユーザー・プロファイルまたは特殊権限 (*ALLOBJ など) を通じて獲得できるからです。

次の表は、ビューの列について説明しています。

表 *255. UDT_PRIVILEGES* ビュー

列名	データ・タイプ	説明
GRANTOR	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
GRANTEE	VARCHAR(128)	特権を認可する対象のユーザー・プロファイル。
UDT_CATALOG	VARCHAR(128)	リレーショナル・データベース名
UDT_SCHEMA	VARCHAR(128)	該当のタイプが入っている SQL のスキーマの名前。
UDT_NAME	VARCHAR(128)	タイプの名前。
PRIVILEGE_TYPE	VARCHAR(10)	認可される特権 : **ALTER** タイプを変更する特権。 **USAGE** タイプを使用する特権。
IS_GRANTABLE	VARCHAR(3)	特権を他のユーザーに認可できるかどうかを示します。 **NO** 特権は認可できません。 **YES** 特権を認可できます。

USAGE_PRIVILEGES

USAGE_PRIVILEGES ビューには、シーケンスまたは XML スキーマに対して認可された各特権ごとに、行が 1 つずつ入ります。このカタログ・ビューを使用して、特定ユーザーが特定のシーケンスまたは XML スキーマに対する権限を持っているかどうかを判別することはできないので、注意してください。なぜなら、シーケンスまたは XML スキーマを使用するための特権は、グループ・ユーザー・プロファイルまたは特殊権限 (*ALLOBJ など) を通じて獲得できるからです。

次の表は、ビューの列について説明しています。

表 *256. USAGE_PRIVILEGES* ビュー

列名	データ・タイプ	説明
GRANTOR	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
GRANTEE	VARCHAR(128)	特権を認可する対象のユーザー・プロファイル。
OBJECT_CATALOG	VARCHAR(128)	リレーショナル・データベース名
OBJECT_SCHEMA	VARCHAR(128)	該当のオブジェクトが入っている SQL スキーマの名前。
OBJECT_NAME	VARCHAR(128)	オブジェクトの名前。
OBJECT_TYPE	VARCHAR(10)	オブジェクトのタイプ: **SEQUENCE** オブジェクトはシーケンスです。 **XML SCHEMA** オブジェクトは XML スキーマです。
PRIVILEGE_TYPE	VARCHAR(10)	認可される特権 : **ALTER** シーケンスまたは XML スキーマを変更する特権。 **USAGE** シーケンスまたは XML スキーマを使用する特権。
IS_GRANTABLE	VARCHAR(3)	特権を他のユーザーに認可できるかどうかを示します。 **NO** 特権は認可できません。 **YES** 特権を認可できます。

USER_DEFINED_TYPES

USER_DEFINED_TYPES ビューには、各特殊タイプごとに行が 1 つずつ入ります。

次の表では、このビューの列について説明します。[162]

表 *257. USER_DEFINED_TYPES* ビュー

列名	データ・タイプ	説明
USER_DEFINED_TYPE_CATALOG	VARCHAR(128)	リレーショナル・データベース名
USER_DEFINED_TYPE_SCHEMA	VARCHAR(128)	特殊タイプのスキーマ名。
USER_DEFINED_TYPE_NAME	VARCHAR(128)	該当の特殊タイプを作成したユーザーの名前。
USER_DEFINED_TYPE_CATEGORY	VARCHAR(128)	ユーザー定義タイプのタイプを示します。 'DISTINCT' が入ります。
IS_INSTANTIABLE	VARCHAR(3)	予約済み。 'YES' が入ります。
IS_FINAL	VARCHAR(3)	予約済み。 'YES' が入ります。
ORDERING_FORM	VARCHAR(4)	この特殊タイプが被比較数の場合に、許される述部の種類を示します。 **FULL** すべての述部が許されます。 **NONE** 述部は許されません。
ORDERING_CATEGORY	VARCHAR(8)	予約済み。 'MAP' が入ります。
ORDERING_ROUTINE_CATALOG	VARCHAR(128)	リレーショナル・データベース名
	NULL 可能	ORDERING_FORM が 'NONE' である場合は、NULL 値が入ります。
ORDERING_ROUTINE_SCHEMA	VARCHAR(128)	予約済み。 'SYSIBM' が入ります。
	NULL 可能	ORDERING_FORM が 'NONE' である場合は、NULL 値が入ります。
ORDERING_ROUTINE_NAME	VARCHAR(128)	予約済み。データ・タイプ名が入ります。
	NULL 可能	ORDERING_FORM が 'NONE' である場合は、NULL 値が入ります。
REFERENCE_TYPE	VARCHAR(16)	予約済み。 NULL 値が入ります。
	NULL 可能	

162. このビューには、組み込みデータ・タイプについての情報は含まれていません。

表 257. USER_DEFINED_TYPES ビュー (続き)

列名	データ・タイプ	説明
DATA_TYPE	VARCHAR(128) NULL 可能	特殊タイプのソース・データ・タイプ: **BIGINT** 大整数 **INTEGER** 長整数 **SMALLINT** 短整数 **DECIMAL** パック 10 進数 **NUMERIC** ゾーン 10 進数 **DOUBLE PRECISION** 浮動小数点数; DOUBLE PRECISION **REAL** 浮動小数点数; REAL **DECFLOAT** 10 進浮動小数点数 **CHARACTER** 固定長文字ストリング **CHARACTER VARYING** 可変長文字ストリング **CHARACTER LARGE OBJECT** 文字ラージ・オブジェクト・ストリング **GRAPHIC** 固定長グラフィック・ストリング **GRAPHIC VARYING** 可変長グラフィック・ストリング **DOUBLE-BYTE CHARACTER LARGE OBJECT** 2 バイト文字ラージ・オブジェクト・ストリング **NATIONAL CHARACTER** 国別文字 **NATIONAL CHARACTER VARYING** 可変長国別文字 **NATIONAL CHARACTER LARGE OBJECT** 国別文字ラージ・オブジェクト **BINARY** 固定長バイナリー・ストリング **BINARY VARYING** 可変長バイナリー・ストリング **BINARY LARGE OBJECT** バイナリー・ラージ・オブジェクト・ストリング **DATE** 日付 **TIME** 時刻 **TIMESTAMP** タイム・スタンプ **DATALINK** データ・リンク **ROWID** 行 ID **XML** XML **USER-DEFINED** 特殊タイプ

表 257. *USER_DEFINED_TYPES* ビュー (続き)

列名	データ・タイプ	説明
CHARACTER_MAXIMUM_LENGTH	INTEGER NULL 可能	データ・タイプが 2 進数、文字、グラフィック・ストリング、および XML の場合は、特殊タイプの最大長。 特殊タイプがストリングでない場合は、NULL 値が入ります。
CHARACTER_OCTET_LENGTH	INTEGER NULL 可能	データ・タイプが 2 進数、文字、グラフィック・ストリング、および XML の場合は、特殊タイプのバイト数。 特殊タイプがストリングでない場合は、NULL 値が入ります。
CHARACTER_SET_CATALOG	VARCHAR(128) NULL 可能	特殊タイプのリレーショナル・データベース名。 特殊タイプがストリングでない場合は、NULL 値が入ります。
CHARACTER_SET_SCHEMA	VARCHAR(128) NULL 可能	特殊タイプの文字セットのスキーマ名。 'SYSIBM' が入ります。 特殊タイプがストリングでない場合は、NULL 値が入ります。
CHARACTER_SET_NAME	VARCHAR(128) NULL 可能	特殊タイプの文字セット名。 特殊タイプがストリングでない場合は、NULL 値が入ります。
COLLATION_CATALOG	VARCHAR(128) NULL 可能	特殊タイプのリレーショナル・データベース名。 特殊タイプがストリングでない場合は、NULL 値が入ります。
COLLATION_SCHEMA	VARCHAR(128) NULL 可能	特殊タイプの照合のスキーマ。 SYSIBM が戻されます。 特殊タイプがストリングでない場合は、NULL 値が入ります。
COLLATION_NAME	VARCHAR(128) NULL 可能	特殊タイプの照合名。 IBMBINARY が戻されます。 特殊タイプがストリングでない場合は、NULL 値が入ります。
NUMERIC_PRECISION	INTEGER NULL 可能	特殊タイプの精度。 **注:** この列では、すべての数値データ・タイプ (単精度および倍精度の浮動小数点数を含む) の精度を指定します。NUMERIC_PRECISION_RADIX 列は、この列の値が 2 進数であるか、または 10 進数であるかを示します。 特殊タイプが数値でない場合は、NULL 値が入ります。
NUMERIC_PRECISION_RADIX	INTEGER NULL 可能	NUMERIC_PRECISION の列で指定される精度が、2 進数 と 10 進数のどちらの数値で指定されるかを指示します。 **2** 2 進数: 浮動小数点数の精度は 2 進数で指定されます。 **10** 10 進数: 他の数値タイプはすべて 10 進数で指定されます。 特殊タイプが数値でない場合は、NULL 値が入ります。
NUMERIC_SCALE	SMALLINT NULL 可能	数値特殊タイプの位取り 特殊タイプが 10 進数、数値、または 2 進数でない場合は、NULL 値が入ります。
DATETIME_PRECISION	INTEGER NULL 可能	日付、時刻、またはタイム・スタンプ特殊タイプの小数部分。 **0** データ・タイプが DATE および TIME の場合 **0-12** データ・タイプが TIMESTAMP の場合 (小数秒) 特殊タイプが日付、時刻、またはタイム・スタンプでない場合は、NULL 値が入ります。
INTERVAL_TYPE	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
INTERVAL_PRECISION	INTEGER NULL 可能	予約済み。 NULL 値が入ります。

表 257. *USER_DEFINED_TYPES* ビュー (続き)

列名	データ・タイプ	説明
SOURCE_DTD_IDENTIFIER	VARCHAR(128)	ソース・データ・タイプの固有の内部 ID。
	NULL 可能	この特殊タイプが他の特殊タイプをソースとしていない場合は、NULL 値が入ります。
REF_DTD_IDENTIFIER	VARCHAR(256)	予約済み。 NULL 値が入ります。
	NULL 可能	
DECLARED_DATA_TYPE	VARCHAR(128)	予約済み。 NULL 値が入ります。
	NULL 可能	
DECLARED_NUMERIC_PRECISION	INTEGER	予約済み。 NULL 値が入ります。
	NULL 可能	
DECLARED_NUMERIC_SCALE	INTEGER	予約済み。 NULL 値が入ります。
	NULL 可能	
MAXIMUM_CARDINALITY	BIGINT	配列データ・タイプの最大カーディナリティー。
	NULL 可能	タイプが配列タイプでない場合は、NULL 値が入ります。

VARIABLE_PRIVILEGES

VARIABLE_PRIVILEGES ビューには、グローバル変数に対して認可された各特権ごとに、行が 1 つずつ入ります。このカタログ・ビューを使用して、特定ユーザーが特定のグローバル変数に対する権限を持っているかどうかを判別することはできないので、注意してください。なぜなら、グローバル変数を使用するための特権は、グループ・ユーザー・プロファイルまたは特殊権限 (*ALLOBJ など) を通じて獲得できるからです。

次の表は、ビューの列について説明しています。

表 *258. VARIABLE_PRIVILEGESS* ビュー

列名	データ・タイプ	説明
GRANTOR	VARCHAR(128) NULL 可能	予約済み。 NULL 値が入ります。
GRANTEE	VARCHAR(128)	特権を認可する対象のユーザー・プロファイル。
VARIABLE_CATALOG	VARCHAR(128)	リレーショナル・データベース名
VARIABLE_SCHEMA	VARCHAR(128)	該当のグローバル変数が入っている SQL スキーマの名前。
VARIABLE_NAME	VARCHAR(128)	グローバル変数の名前。
PRIVILEGE_TYPE	VARCHAR(10)	認可される特権 : **ALTER** グローバル変数を変更する特権。 **READ** グローバル変数の値を読み取る特権。 **WRITE** グローバル変数に値を割り当てる特権。
IS_GRANTABLE	VARCHAR(3)	特権を他のユーザーに認可できるかどうかを示します。 **NO** 特権は認可できません。 **YES** 特権を認可できます。

VIEWS

VIEWS ビューには、各ビューごとに行が 1 つずつ入ります。

次の表は、ビューの列について説明しています。

表 *259. VIEWS* ビュー

列名	データ・タイプ	説明
TABLE_CATALOG	VARCHAR(128)	リレーショナル・データベース名
TABLE_SCHEMA	VARCHAR(128)	該当のビューが入っている SQL のスキーマの名前。
TABLE_NAME	VARCHAR(128)	ビューの名前。
VIEW_DEFINITION	DBCLOB(2M) CCSID 13488 NULL 可能	CREATE VIEW ステートメントの QUERY 式の部分。
CHECK_OPTION	VARCHAR(8)	該当のビューに対して使用された検査オプション **NONE** 検査オプションは指定されませんでした **LOCAL** ローカル・オプションが指定されました **CASCADED** カスケード・オプションが指定されました
IS_UPDATABLE	VARCHAR(3)	ビューが更新可能かどうかを指定します。 **YES** 更新可能なビューです。 **NO** 読み取り専用のビューです。

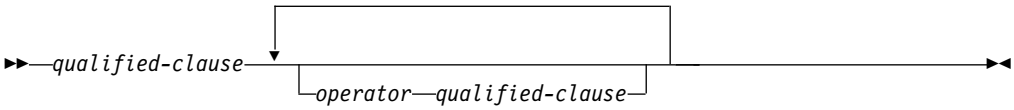
付録 G. テキスト検索引数の構文

テキスト検索引数は、テキスト文書で用語を検索する場合に指定します。検索パラメーターと、1 つ以上の検索語で構成されています。テキスト検索引数を使用する SQL スカラーのテキスト検索関数は CONTAINS と SCORE です。

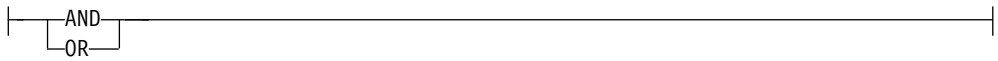
CONTAINS と SCORE については、 394 ページの『CONTAINS』 および 620 ページの『SCORE』 を参照してください。

テキスト検索について詳しくは、OmniFind Text Search Server for Db2 for i を参照してください。

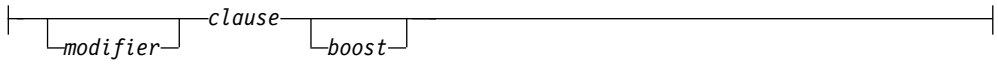
構文



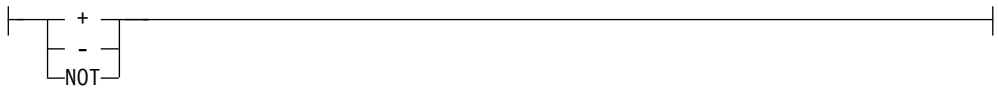
演算子:



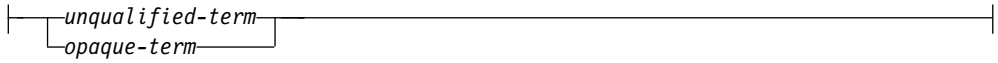
qualified-clause:



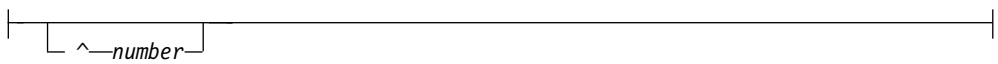
modifier:



clause:



boost:



説明

検索引数は 1 つの用語、または空白で区切られた一連の用語で、テキスト文書で検索を行う場合に指定します。これは、1 つ以上の検索語と、様々なオプションの検索パラメーターで構成されます。

簡単な検索を実行する場合、1 つ以上の用語を入力できます。検索エンジンは、これらの語、またはこれらの語の変化形がすべてが含まれる文書を戻します。例えば、*king* という語を検索すると、*king* が含まれる文書を戻します。デフォルトでは、検索エンジンは検索語の変化形も戻します。つまり、*kings* が含まれる文書も戻します。同様に、2 つの用語で検索を行う場合、検索エンジンは両方の用語が含まれる文書を戻します。それらの語と正確に一致する句を検索する場合には、引用符を追加するだけです。

使用する検索語が具体的であれば、それだけ結果も絞り込まれます。ただし、以下のオプションのうち 1 つ以上を使用しても、検索をより絞り込むことができます。

operator
検索において満たす必要があるのが、*operator* の両側の *qualified clause* であるのか、それともその片方だけであるのかを指定します。

AND
検索において満たす必要があるのは、*operator* の両側の *qualified clause* です。

OR 検索において満たす必要があるのは、*operator* の両側の *qualified clause* のうち、少なくとも片方だけです。

SQL における検索条件と同様、*qualified clauses* と*operators* のどちらを最初に評価するかを決定するために括弧を使用できます。括弧を指定しないと、AND が OR の前に適用されます。

modifier
それぞれの *clause* はオカレンス修飾子を指定できます。 *modifier* を *clause* に指定しないと、デフォルトの正符号 (+) になります。

+ 文書において *clause* が必須です。

- または **NOT**
文書では、*clause* を指定することはできません。

? 文書では、*clause* はオプションです。

clause
検索ストリングを指定します。検索ストリングでは、疑問符 (?) とアスタリスクに特別な意味があります。単一文字を示す場合には疑問符 (?) を使用し、ゼロ個以上の文字ストリングを表わす場合にはアスタリスク (*) を使用します。検索ストリングに複数の語が含まれており、句として扱う必要のある場合には、検索ストリングを引用符 (") で囲んでください。*clause* の中に含まれる空白は、無視されます。

unqualified-term
非修飾用語は用語または句を単に示します。用語は、king のような単語や、"king" のように正確に一致する単語、または任意の単一文字を表わすための疑問符 (?) が含まれる単語やゼロ個以上の文字ストリングを表わすた

めのアスタリスク (*) が含まれる単語 (例: king* または king?) などにすることができます。同様に、句は cabbages and kings などの一群の単語で構成されていたり、"The King and I" のように正確に一致する句であったり、"all the king's ho*ses" または "all the king's ?" などのようにワイルドカードが含まれる句にすることができます。

clause 内の文字が検索引数の構文で特別な意味を持ついずれかの文字の場合、エスケープ文字 (¥) を使用すると、それに続く文字は *clause* において正規文字として扱うべきであることを示します。

opaque-term
あいまいな照会条件と言われているのは、言語照会パーサーによって解析されないためです。あいまい条件は、独自の構文によって識別されます。XML マークアップが含まれている、テキスト検索照会で使用するあいまい条件とは @xmlxp で、@xmlxp:'/TagA/TagB[.contains("king")]' などとなります。

boost
boost は、*clause* ごとに指定できます。*boost* によって、*clause* のオカレンスに対して重要度の高または低を指定します。

number
ゼロより大きい小数または整数定数を指定します。 *boost* を *clause* に指定しないと、デフォルトのランキング調整値は 1 になります。

注

大/小文字の区別**:** 検索では大/小文字の区別がありません。そのため、スペイン語で用語 "DOS" に正確に一致する文書を検索すると、DOS または dos が含まれる文書が戻されます。

例**:** 単純なテキスト検索

CONTAINS 関数と SCORE 関数を使用すると、テキスト検索索引で 1 語または複数語に関して単純なテキスト検索を実行できます。

検索エンジンは、文字間の空白は無視します。検索引数が空白の場合、またはブランクしか含まれていない場合、何も一致しません。

以下の表には、単純なテキスト検索要求の例がいくつか示されています。

表 *260.* 単純なテキスト検索の例

検索語のタイプ	例	検索結果
1 語	`king`	`king` または `kings` という語が含まれるすべての文書が戻ります。検索では、大/小文字の区別はありません。
複数語	`king lear`	`king` と `lear` が含まれるすべての文書が戻ります。デフォルトの演算子は、論理演算子 AND です。

例: 単純なテキスト検索

すべてのテキスト検索で、演算子 AND および + (正符号) が暗黙的に指定されます。例えば、King Lear のテキスト検索が戻す結果は、King AND Lear または King + Lear の場合と同じです。

論理演算子 NOT、AND、および OR に関しては、すべて大文字で入力する必要があります。

拡張テキスト検索演算子

拡張テキスト検索演算子を使用して、CONTAINS 関数と SCORE 関数の検索結果をさらに洗練できます。

以下の表の第 1 列では、テキスト検索で使用できる演算子を列挙しています。(論理演算子の NOT、AND、OR はすべて大文字で入力しなければなりません。) 第 2 列では、入力できるサンプル・テキスト検索を示しています。第 3 列では、テキスト検索の例から得られる結果のタイプについて説明しています。

表 *261.* 拡張検索演算子と複雑なテキスト検索の例

演算子	例	検索結果
AND	"King Lear" AND "Othello" "King Lear" "Othello"	どちらのテキスト検索からも、「King Lear」と「Othello」の両方の語が含まれている文書が返されます。AND 演算子は、デフォルトの結合演算子です。2 つの語の間に論理演算子が存在しない場合は、AND 演算子が使用されます。例えば、「King Lear」というテキスト検索は、「King AND Lear」というテキスト検索と同じ意味になります。
OR	"King Lear" OR Lear	「King Lear」または「Lear」が含まれている文書が返されます。OR 演算子は、2 つの語をリンクして、どちらかの語が含まれている文書を一致項目として検出します。
NOT	"King Lear" NOT "Norman Lear"	「King Lear」が含まれていて、「Norman Lear」が含まれていない文書が返されます。NOT 演算子は、1 つの検索条件だけで使用することはできません。例えば、「NOT "King Lear"」という検索では、結果は返されません。
" " (完全一致)	第 1 のテキスト検索: "King Lear" 第 2 のテキスト検索: "king"	第 1 のテキスト検索からは、「King Lear」という語句の完全一致項目が返されます。 第 2 のテキスト検索からは、「king」という語だけが返され、「kings」や「kingly」などの語は返されません。
* (ワイルドカード文字)	test* te*t	「test」、「tests」、「tester」などと、「test」、「text」などの組み合わせが含まれている文書が一致項目として返されます。

表 *261.* 拡張検索演算子と複雑なテキスト検索の例 *(*続き*)*

演算子	例	検索結果
? (ワイルドカード文字)	test? te?t	「tests」などと、「test」、「text」などの組み合わせが含まれている文書が一致項目として返されます。
^ (スコア・ランキング調整係数) 語句^数値	第 1 のテキスト検索: "King Lear"^4 "Richard III" 第 2 のテキスト検索: title: (software download)^5 pdf viewer -shipping	第 1 のテキスト検索では、「King Lear」という語句が含まれている文書が検索結果リストの高い位置に強制的に表示されます。 第 2 のテキスト検索では、「software download」というタイトルの文書が結果リストの高い位置に強制的に表示されます。 ランキング調整係数は、正の数でなければなりませんが、1 未満の数 (0.2 など) も指定できます。ランキング調整係数に制限はありません。
+ (包含)	+Lear King	「Lear」と「King」が含まれているすべての文書が返されます。「Lear AND King」というテキスト検索と同じ意味になります。
- (除外)	"King Lear" -"Lear Jet"	「King Lear」が含まれていて、「Lear Jet」が含まれていない文書が返されます。
( )	(King OR Lear) AND plays	「King」または「Lear」のいずれかと「plays」が含まれている文書が返されます。この括弧を使用することによって、「plays」が存在し、さらに「King」または「Lear」のいずれかの語が存在する文書を検出できます。
\ (エスケープ文字)	\(1\+1\)\:2	「(1+1):2」が含まれている文書が返されます。テキスト検索構文の中で使用する特殊文字をクリアするには、¥ を使用します。特殊文字は、+ - && ‖ ！ ( ) { } [ ] ^ " * ? : ¥ です。特殊文字をクリアすると、特殊文字がテキスト検索の一部と見なされ、分析の対象になります。

例: CONTAINS 関数と SCORE 関数の使用

同じ照会で CONTAINS 関数と SCORE 関数の両方を使用できます。その照会は、テキスト検索索引を検索し、テキスト文書が検索引数の基準に一致するかどうか、一致する場合はその一致の頻度がどれほどかを示す値を返します。

以下の例では、基本表 BOOKS のデータを使用します。その基本表には、ISBN (VARCHAR(20))、ABSTRACT (VARCHAR(10000))、PRICE (INTEGER) の各列が含まれています。

例: CONTAINS 関数と SCORE 関数の使用

表 *262.* 基本表 *BOOKS*

ISBN	ABSTRACT	PRICE
i1	"a b c"	7
i2	"a b d"	10
i3	"a e a"	8

以下の照会を実行します。

```
SELECT ISBN, SCORE(ABSTRACT,'"b"')
FROM BOOKS
WHERE CONTAINS (ABSTRACT,'"b"') = 1
```

この照会からは、以下の 2 つの行が返されます。

```
i1, 0.3
i3, 0.4
```

スコア値は、テキスト列の内容によって異なる可能性があります。

XML テキスト検索

XML テキスト検索では、XPath 言語のサブセットをテキスト検索用の拡張機能と共に使用することにより、XML 文書に索引付けをして検索することができます。そのようにすると、構造的エレメントは、個別に使用することも、照会内のフリー・テキストと結合することもできます。

構造的エレメントとは、タグ名、属性名、および属性値のことです。

以下のリストは、XML 検索の主要な機能について取り上げています。

XML 構造的検索
: 特殊なあいまい条件を照会に含めることにより、XML 文書内で、構造的エレメント (タグ名、属性名、および属性値) を、およびそれらのエレメントによって範囲限定されるテキストを検索できます。

XML 照会のトークン化
: XML 照会用語のフリー・テキストは、非 XML 照会用語のテキストがトークン化されるのと同じ方法でトークン化されます。ただし、ネストされたあいまい条件はサポートされません。同義語、ワイルドカード文字、句、および見出し語化はサポートされます。

数値
: 属性値を、数値、日付、日時のデータ・タイプと比較する述部はサポートされます。

完全一致
: 述部にストリング引数がある = (等号) 演算子は、ストリング内のすべてのトークンと、指定されたテキスト・スパン内のすべてのトークンとの完全一致を要求します。順序は、重要ではありません。

UIMA アクセス不可
: Unstructured Information Management Architecture (UIMA) は、XML 検索内でのトークン化に使用されますが、ユーザー作成のアノテーターはサポートされません。

XML テキスト検索の構文

XPath 言語のサブセットは Extended Backus-Naur Form (EBNF) 構文によって定義されていて、XML 検索照会パーサーによってサポートされています。照会パーサーは、サポートされている構文に準拠していない照会は拒否し、例外をスローします。

EBNF 構文は、以下の方法で簡略化されています。

- 反復や範囲を指定すると、大規模な構造が排除されます。
- フィルター式は除去されます。
- 述部式で絶対パス名は使用できなくなっています。
- 1 つの軸 (タグ) だけを認識し、順方向に進むだけになっています。

以下の表は、EBNF 表記でサポートされる構文を示しています。

表 263. EBNF 表記でサポートされる照会構文

記号	実動

表 263. EBNF 表記でサポートされる照会構文 (続き)

XMLQuery ::=	QueryPrefix NameSpaceDeclaration QueryString | QueryPrefix QueryString
QueryPrefix ::=	@xmlxp:
QueryString ::=	"'" PathExpr "'"
PathExpr ::=	RelativePathExpr | "/" RelativePathExpr? | "//" RelativePathExpr
RelativePathExpr ::=	StepExpr (("/" | "//") StepExpr)*
StepExpr ::=	("." | AbbrevForwardStep) Predicate?
AbbrevForwardStep ::=	"@"? (QName | "*")
Predicate ::=	"[" PredicateExpr "]"
PredicateExpr ::=	Expr | PredicateExpr ("and" | "or") | "(" PredicateExpr ")"
Expr ::=	ComparisonExpr | ContainmentExpr
ComparisonExpr ::=	PathExpr ComparisonOp Literal
ComparisonOp ::=	"=" | "<" | ">" | "!=" | "<=" | ">="
Literal ::=	StringLiteral | NumericLiteral | DateLiteral
ContainmentExpr ::=	PathExpr "contains" "(" StringLiteral ")" | PathExpr "excludes" "(" StringLiteral ")"
StringLiteral ::=	"\"" [^"]* "\"" | "'" [^']* "'"
DateLiteral ::=	"xs:date(¥"" xmlDate "¥")" | "xs:dateTime(¥"" xmlDateTime "¥")"
xmlDate ::=	yyyy"-"mm"-"dd
xmlDateTime ::=	yyyy"-"mm"-"dd [T] hh":"mm":"ss"."uuuuuu
NameSpaceDeclaration ::=	defaultNameSpace (NameSpacePrefixDeclaration)*
defaultNameSpace ::=	"declare default element namespace " StringLiteral ";"
NameSpacePrefixDeclaration ::=	"declare namespace" NameSpacePrefix "=" StringLiteral ";"
NameSpacePrefix ::=	[^":]+

QName について詳しくは、http://www.w3.org/TR/REC-xml-names/#NT-QNameを参照してください。

EBNF 構文の表記では、XPath 表記を使用する以下の XML 検索照会に関しては明確にされていない場合があります。

- 正規化されていない名前: XML タグと属性の名前は、索引付けされる際に正規化されません。それらの名前はいずれかの方法で小文字に変換されたり、変更されたりしません。一致項目を取得するには、XML タグと属性の名前で大/小文字の区別が重要になります。そのため、照会で XML タグや属性名に使用するストリングが、ソース文書に含まれている名前と正確に一致しなければなりません。

- フリー・テキストの正規化: XML 文書のフリー・テキスト (タグ自体の内部ではなくタグ間のテキスト) と属性値は、索引付けの前に正規化されます。XML 検索照会内のテキスト (*contains* 演算子または *excludes* 演算子内、あるいは引用符によって囲まれたストリング内) も正規化されます。句、同義語、ワイルドカード文字、および見出し語などの機能もサポートされます。
- 演算子優先順位: XML 検索述部では、包含演算子と比較演算子が論理演算子よりも優先されます。すべての論理演算子の優先順位は同じです。包含演算子は、*contains* と *excludes* です。比較演算子は、 =、!=、<、>、<=、および >= です。論理演算子は、「and」と「or」です。優先順位を確実に指定するために括弧を使用することもできます。
- 意味体系: XML 検索述部では、比較演算子は属性値にのみ適用され、タグには適用されません。

例: XPath テキスト検索

XML パーサーに送信される XPath 照会がすべて有効であるためには、あいまい条件を使用した XPath 言語のサブセットで作成される必要があります。あいまい条件は、言語テキスト検索パーサーによって解析されません。

テキスト検索パーサーは、テキスト検索で使用される構文によってあいまい条件を識別します。例を以下に示します。

```
@xpath:'query'
```

ここで、*query* は以下の表の例に記されているテキストです。

表 *264.* 有効な *XPath* 照会の例

照会	説明
/sentences	最上位タグが *sentences* の文書。
//sentences	任意のレベルに *sentences* というタグがある文書。
/sentence/paragraph	最上位のタグが *sentence* で、直接の子タグが *paragraph* の文書。
/sentence/paragraph/	最上位のタグが *sentence* で、直接の子タグが *paragraph* の文書。
/book/@author	最上位が *book* タグで、属性 *author* の文書。
/book//@author	最上位が *book* タグで、任意のレベルで属性 *author* の下位タグを持つ文書。
/book[@author contains("barnes") and @title = "the lemon table"]	最上位が *book* タグで、*author* 属性には「barnes」(正規化) が含まれ、*title* 属性には「the」、「lemon」、「table」という語 (この順序で正規化) だけが含まれる文書。
/book[@author contains("barnes") and (@title contains("lemon") or @title contains("flaubert"))]	最上位が *book* タグで、指定された *author* 属性と、指定された 2 つの *title* 属性のどちらかを持つ文書。
/book[@publishDate > xs:date("2000-01-01")]	最上位タグが *book* で、*book* の *PublishDate* 属性に 2000-01-01 より後の日付が含まれます。

表 264. 有効な XPath 照会の例 (続き)

照会	説明
/book[purchaseTime > xs:dateTime("2009-05-20T13:00:00")]	最上位タグが *book* です。*book* には、2009-05-20T13:00:00 より後の日時式である直接の子 *purchaseTime* が含まれます。
/program[. contains("""hello, world.""")	最上位が *program* タグで、トークン *hello* と *world* (正規化) がこのままの順序で、なおかつ連続して含まれている文書。
/book[paragraph contains("foo")]//sentence	最上位が *book* タグで、直接の子タグ *paragraph* には「foo」が含まれ、*book* タグに対して任意のレベルに下位タグ *sentence* を持つ文書。
/auto[@price < 30000.]	最上位が *auto* タグで、属性 *price* に 30000 未満の数値が入っている文書。
//microbe[@size < 3.0e-06]	任意のレベルに *microbe* タグがあり、*size* 属性が .000003 未満の値の文書。

テキスト検索の言語オプション

テキスト検索の言語オプションは、テキスト検索の実行時に使用する言語規則を指定します。

QUERYLANGUAGE を指定しないと、CONTAINS 関数または SCORE 関数の呼び出し時に使用されるテキスト検索索引の言語値がデフォルトとなります。テキスト検索索引の言語値が AUTO の場合、QUERYLANGUAGE のデフォルト値は en_US です。以下の表は、QUERYLANGUAGE オプションで使用可能な言語コードを示しています。

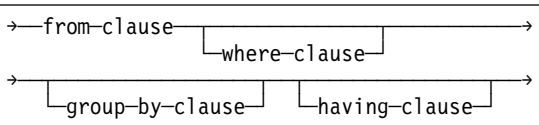
言語コード	言語
ar_AA	アラビア語
cs_CZ	チェコ語
da_DK	デンマーク語
de_CH	ドイツ語 (スイス)
de_DE	ドイツ語 (ドイツ)
el_GR	ギリシャ語
en_AU	英語 (オーストラリア)
en_GB	英語 (英国)
en_US	英語 (米国)
es_ES	スペイン語 (スペイン)
fi_FI	フィンランド語
fr_CA	フランス語 (カナダ)
fr_FR	フランス語 (フランス)
it_IT	イタリア語
ja_JP	日本語
ko_KR	韓国語
nb_NO	ノルウェー語ブークモール
nl_NL	オランダ語
nn_NO	ノルウェー語ニーノシュク
pl_PL	ポーランド語
pt_BR	ポルトガル語 (ブラジル)
pt_PT	ポルトガル語 (ポルトガル)
ru_RU	ロシア語
sv_SE	スウェーデン語
zh_CN	中国語 (簡体字)
zh_TW	中国語 (繁体字)

付録 H. 用語の差異

ANSI および ISO 標準で使用されている用語の中には、本書および他の製品で使用されている用語と異なるものがあります。

次の表は、SQL 2003 Core 標準の用語から Db2 SQL 用語への相互参照のためのものです。

表 *265. ANSI/ISO* 用語から *Db2 SQL* 用語への相互参照

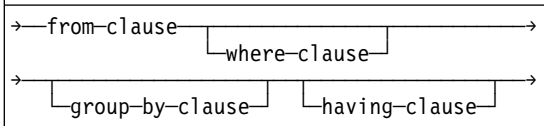
ANSI/ISO 用語	**Db2 SQL** 用語
リテラル	定数
比較述部	基本述部
比較述部副照会	基本述部での副照会
表/カーソルの度合い	選択リスト内の項目数
グループ化された表	GROUP BY 文節あるいは HAVING 文節によって作成された結果表
グループ化されたビュー	GROUP BY 文節あるいは HAVING 文節によって作成された結果ビュー
グループ化列	GROUP BY 文節内の列
外部参照	相関参照
照会式	全選択
照会仕様	副選択
結果仕様	結果
セット関数	集約関数
表式	
ターゲット仕様	標識変数が後に続くホスト変数
トランザクション	作業論理単位または作業単位
値式	算術式

次の表は、Db2 SQL 用語から SQL 2003 Core 標準の用語への相互参照のためのものです。

表 *266. Db2 SQL* 用語から *ANSI/ISO* 用語への相互参照

Db2 SQL 用語	**ANSI/ISO** 用語
集約関数	セット関数
算術式	値式
基本述部	比較述部
GROUP BY 文節内の列	グループ化列
相関参照	外部参照

表 *266. Db2 SQL* 用語から *ANSI/ISO* 用語への相互参照 *(*続き*)*

Db2 SQL 用語	**ANSI/ISO** 用語
	表式
全選択	照会式
標識変数が後に続くホスト変数	ターゲット仕様
作業論理単位または作業単位	トランザクション
対話式 SQL	ダイレクト SQL
選択リスト内の項目数	表/カーソルの度合い
結果	結果仕様
GROUP BY 文節あるいは HAVING 文節によって作成された結果表	グループ化された表
GROUP BY 文節あるいは HAVING 文節によって作成された結果ビュー	グループ化されたビュー
基本述部での副照会	比較述部副照会
副選択	照会仕様
括弧内の副選択または全選択	照会条件

付録 I. 予約済みスキーマ名と予約語

このトピックでは、データベース・マネージャーによって使用される特定の名前の制約事項について説明します。名前によっては、予約済みで、アプリケーション・プログラムで使用できない名前があります。また、データベース・マネージャーによって、その使用は禁止されてはいないものの、アプリケーション・プログラムによる使用をお勧めできない名前もあります。

予約済みスキーマ名

これは、予約済みスキーマ名のリストです。

次のスキーマ名が予約されます。

- QSYS2
- SYSCAT
- SYSFUN
- SYSIBM
- SYSIBMADM
- SYSPROC
- SYSPUBLIC
- SYSSTAT
- SYSTEM

さらに、Q および SYS は規則によりシステムで予約されている領域を示すのに使用されるので、 Q の接頭部または SYS の接頭部で始まるスキーマ名は使用しないようにしてください。

さらに、SESSION はスキーマ名としては使用しないようお勧めします。

予約語

次の表は、現時点での Db2 for i の予約語のリストを示しています。

新たな語が、必要に応じて追加されることがあります。 予約語として将来追加される可能性のある語のリストについては、「SQL Reference for Cross-Platform Development」(http://www.ibm.com/developerworks/data/library/techarticle/0206sqlref/0206sqlref.html) の IBM SQL および ANSI 予約語を参照してください。

表 *267. SQL* 予約語

ABSENT	COLLECT	DB2GENERAL	EXCLUSIVE
ACCORDING	COLLECTION	DB2GENRL	EXECUTE
ACCTNG	COLUMN	DB2SQL	EXISTS
ACTION	COMMENT	DEACTIVATE	EXIT
ACTIVATE	COMMIT	DEALLOCATE	EXTEND
ADD	COMPACT	DECLARE	EXTERNAL
ALIAS	COMPRESS	DEFAULT	EXTRACT
ALL	CONCAT	DEFAULTS	FENCED
ALLOCATE	CONCURRENT	DEFER	FETCH
ALLOW	CONDITION	DEFINE	FIELDPROC
ALTER	CONNECT	DEFINITION	FILE
AND	CONNECT_BY_ROOT	DELETE	FINAL
ANY	CONNECTION	DELETING	FIRST_VALUE
APPEND	CONSTANT	DENSERANK	FOR
APPLNAME	CONSTRAINT	DENSE_RANK	FOREIGN
ARRAY	CONTAINS	DESC	FORMAT
ARRAY_AGG	CONTENT	DESCRIBE	FREE
AS	CONTINUE	DESCRIPTOR	FREEPAGE
ASC	COPY	DETACH	FROM
ASENSITIVE	COUNT	DETERMINISTIC	FULL
ASSOCIATE	COUNT_BIG	DIAGNOSTICS	FUNCTION
ATOMIC	CREATE	DISABLE	GBPCACHE
ATTACH	CREATEIN	DISALLOW	GENERAL
ATTRIBUTES	CROSS	DISCONNECT	GENERATED
AUTHORIZATION	CUBE	DISTINCT	GET
AUTONOMOUS	CUME_DIST	DO	GLOBAL
BEFORE	CURRENT	DOCUMENT	GO
BEGIN	CURRENT_DATE	DOUBLE	GOTO
BETWEEN	CURRENT_PATH	DROP	GRANT
BINARY	CURRENT_SCHEMA	DYNAMIC	GRAPHIC
BIND	CURRENT_SERVER	EACH	GROUP
BIT	CURRENT_TIME	ELSE	HANDLER
BUFFERPOOL	CURRENT_TIMESTAMP	ELSEIF	HASH
BY	CURRENT_TIMEZONE	EMPTY	HASHED_VALUE
CACHE	CURRENT_USER	ENABLE	HAVING
CALL	CURSOR	ENCODING	HINT
CALLED	CYCLE	ENCRYPTION	HOLD
CARDINALITY	DATA	END	HOUR
CASE	DATABASE	ENDING	HOURS
CAST	DATAPARTITIONNAME	END-EXEC (COBOL only)	ID
CCSID	DATAPARTITIONNUM	ENFORCED	IDENTITY
CHAR	DATE	ERROR	IF
CHARACTER	DAY	ESCAPE	IGNORE
CHECK	DAYS	EVERY	IMMEDIATE
CL	DBINFO	EXCEPT	IMPLICITLY
CLOSE	DBPARTITIONNAME	EXCEPTION	IN
CLUSTER	DBPARTITIONNUM	EXCLUDING	INCLUDE

表 *268. SQL* 予約語 (続き)

INCLUDING	LOCK	OLD_TABLE	RCDFMT
INCLUSIVE	LOCKSIZE	OMIT	READ
INCREMENT	LOG	ON	READS
INDEX	LOGGED	ONLY	RECOVERY
INDEXBP	LONG	OPEN	REFERENCES
INDICATOR	LOOP	OPTIMIZE	REFERENCING
INF	MAINTAINED	OPTION	REFRESH
INFINITY	MASK	OR	REGEXP_LIKE
INHERIT	MATCHED	ORDER	RELEASE
INLINE	MATERIALIZED	ORDINALITY	RENAME
INNER	MAXVALUE	ORGANIZE	REPEAT
INOUT	MERGE	OUT	RESET
INSENSITIVE	MICROSECOND	OUTER	RESIGNAL
INSERT	MICROSECONDS	OVER	RESTART
INSERTING	MINPCTUSED	OVERLAY	RESULT
INTEGRITY	MINUTE	OVERRIDING	RESULT_SET_LOCATOR
INTERSECT	MINUTES	PACKAGE	RETURN
INTO	MINVALUE	PADDED	RETURNING
IS	MIXED	PAGE	RETURNS
ISOLATION	MODE	PAGESIZE	REVOKE
ITERATE	MODIFIES	PARAMETER	RID
JAVA	MONTH	PART	RIGHT
JOIN	MONTHS	PARTITION	ROLLBACK
JSON_ARRAY	NAMESPACE	PARTITIONED	ROLLUP
JSON_ARRAYAGG	NAN	PARTITIONING	ROUTINE
JSON_EXISTS	NATIONAL	PARTITIONS	ROW
JSON_OBJECT	NCHAR	PASSING	ROWNUMBER
JSON_OBJECTAGG	NCLOB	PASSWORD	ROW_NUMBER
JSON_QUERY	NESTED	PATH	ROWS
JSON_TABLE	NEW	PCTFREE	RRN
JSON_VALUE	NEW_TABLE	PERCENT_RANK	RUN
KEEP	NEXTVAL	PERCENTILE_CONT	SAVEPOINT
KEY	NO	PERCENTILE_DISC	SBCS
LABEL	NOCACHE	PERIOD	SCHEMA
LAG	NOCYCLE	PERMISSION	SCRATCHPAD
LANGUAGE	NODENAME	PIECESIZE	SCROLL
LAST_VALUE	NODENUMBER	PIPE	SEARCH
LATERAL	NOMAXVALUE	PLAN	SECOND
LEAD	NOMINVALUE	POSITION	SECONDS
LEAVE	NONE	PREPARE	SECQTY
LEFT	NOORDER	PREVVAL	SECURED
LEVEL2	NORMALIZED	PRIMARY	SELECT
LIKE	NOT	PRIOR	SENSITIVE
LIMIT	NTH_VALUE	PRIQTY	SEQUENCE
LINKTYPE	NTILE	PRIVILEGES	SESSION
LISTAGG	NULL	PROCEDURE	SESSION_USER
LOCAL	NULLS	PROGRAM	SET
LOCALDATE	NVARCHAR	PROGRAMID	SIGNAL
LOCALTIME	OBID	QUERY	SIMPLE
LOCALTIMESTAMP	OF	RANGE	SKIP
LOCATION	OFFSET	RANK	SNAN
LOCATOR	OLD	RATIO_TO_REPORT	SOME

表 *269. SQL* 予約語 *(*続き*)*

SOURCE	TRANSACTION	VARIABLE	XMLCONCAT
SPECIFIC	TRANSFER	VARIANT	XMLDOCUMENT
SQL	TRIGGER	VCAT	XMLELEMENT
SQLID	TRIM	VERSION	XMLFOREST
STACKED	TRIM_ARRAY	VERSIONING	XMLGROUP
START	TRUNCATE	VIEW	XMLNAMESPACES
STARTING	TYPE	VOLATILE	XMLPARSE
STATEMENT	UNDO	WAIT	XMLPI
STATIC	UNION	WHEN	XMLROW
STOGROUP	UNIQUE	WHENEVER	XMLSERIALIZE
SUBSTRING	UNIT	WHERE	XMLTABLE
SUMMARY	UNNEST	WHILE	XMLTEXT
SYNONYM	UNTIL	WITH	XMLVALIDATE
SYSTEM_TIME	UPDATE	WITHIN	XSLTRANSFORM
SYSTEM_USER	UPDATING	WITHOUT	XSROBJECT
TABLE	URI	WRAPPED	YEAR
TABLESPACE	USAGE	WRAPPER	YEARS
TABLESPACES	USE	WRITE	YES
THEN	USER	WRKSTNNAME	ZONE
THREADSAFE	USERID	XMLAGG	
TIME	USING	XMLATTRIBUTES	
TIMESTAMP	VALUE	XMLCAST	
TO	VALUES	XMLCOMMENT	

付録 J. 関連情報

ここにリストした資料には、本書で説明した事項や参照したトピックに関する追加情報が収録されています。

いずれの資料も、それぞれの正式表題と資料番号を付けて示してあります。本書で参照している資料の場合、略称を使用しています。

- バックアップおよび回復

  バックアップおよびリカバリーの計画、保管および復元手順のために使用できる各種のメディア、およびディスク・リカバリー手順に関する情報が収録されています。バックアップからシステムを再インストールする方法も紹介されています。

- ILE COBOL プログラマーの手引き 

  この資料には、IBM i プロダクトで COBOL プログラムの設計、作成、テスト、および保守を行う上で必要な情報が収められています。

- ILE RPG プログラマーの手引き 

  この資料には、IBM i プロダクトで ILE RPG プログラムの設計、作成、テスト、および保守を行う上で必要な情報が収められています。

- REXX/400 Programmer's Guide 

  この資料には、IBM i プロダクト上で REXX/400 プログラムの設計、作成、テスト、および保守を行う上で必要な情報を収めてあります。

- CL プログラミング

  この資料には、プログラミングに関するトピックを広範囲にわたって論じています。その主なものを挙げると、オブジェクトとライブラリーの全般的な説明、CL プログラミング、プログラム相互間の流れと通信の制御、CL プログラムのオブジェクトの扱い方、CL プログラムの作成方法などがあります。その他に、事前定義のメッセージと即時メッセージ、ユーザーが定義するコマンドとメニューの取り扱い、定義、および作成の方法、デバッグ・モード、停止点、トレースなどを含むアプリケーションのテスト、および表示機能についても説明しています。

- データベース・ファイル管理

  アプリケーション・プログラムにおけるファイルの使い方を説明しています。

- データベース・プログラミング

  この資料では、IBM i データベース編成を詳しく説明し、システム上でデータベース・ファイルを作成、記述、および更新する方法についても示します。

- 分散データベース・プログラミング

この資料では、分散リレーショナル・データベース・アーキテクチャー (DRDA) を使用して IBM i プロダクトを分散リレーショナル・データベースで準備および管理する方法について説明しています。類似のシステム環境における複数の IBM i プロダクト上で分散リレーショナル・データベースを計画、セットアップ、プログラミング、管理、および操作する方法について説明します。

- 機密保護解説書

  本書には、システム・セキュリティーの概念、セキュリティーのための計画方法、およびシステムにおけるセキュリティーのセットアップ方法に関する情報が記載されています。また、正当な権限を持たないユーザーの使用からシステムやデータを保護する方法、故意または過失による損傷や破壊からデータを保護する方法、セキュリティーを最新に保つ方法、システム上にセキュリティーを設定する方法についても説明しています。

- SQL プログラミング

  この資料では、Db2 for i ステートメントの設計、作成、実行、およびテストの方法について概説しています。また、対話式構造化照会言語 (SQL) についても説明しています。

- SQL XML プログラミング

  この資料は、 SQL での XML データ・タイプの使用法を説明しています。XML の生成や XML 文書のすべてまたは一部の取り出しを行う関数およびプロシージャーの使用例も含まれています。

- 組み込み SQL プログラミング

  この資料には、ILE C、ILE C++、COBOL、ILE COBOL、RPG、ILE RPG、REXX、および PL/I プログラムで SQL ステートメントを作成する方法の例が収められています。

- データベース・パフォーマンスおよび Query 最適化

  この資料には、使用可能なツールおよび技法を使用して照会のパフォーマンスを最適化するための情報が収められています。

- IDDU Use 

  この資料は、IBM i の対話式データ定義ユーティリティー (IDDU) を使用して、データ・ディクショナリー、ファイル、およびレコードをシステムに対して記述する方法を説明しています。

- SQL 呼び出しレベル・インターフェース (ODBC)

  この資料では、X/Open SQL 呼び出しレベル・インターフェースを使用し、Db2 for i で用意されているサービス・プログラムに対するプロシージャー呼び出しを直接介して、SQL 関数にアクセスする方法を説明しています。

- IBM i Information Center の IBM i Access カテゴリー

  この資料では、IBM i Access ODBC を使用して、クライアント上で ODBC アプリケーションをセットアップし実行する方法を説明しています。この資料には、パフォーマンス、例、および IBM i Access ODBC によって実行する特定のアプリケーションの構成に関する章が含まれています。

- IBM Toolbox for Java

  この資料では、IBM Toolbox for Java を使用して、クライアントで JDBC アプリケーションをセットアップし、実行する方法を説明しています。この資料には、パフォーマンス、例、および IBM Toolbox for Java によって実行する特定のアプリケーションの構成に関する章が含まれています。

- IBM Developer Kit for Java

  この資料には、IBM i プロダクトで Java プログラムの設計、作成、テスト、および保守を行う上で必要な情報が収められています。この資料には、IBM Developer Kit for Java JDBC ドライバーに関する情報も含まれています。

- Db2 マルチシステム

  この資料は、分散リレーショナル・データベース・ファイル、ノード・グループ、および区分化についての、基本的な概念を説明しています。これには、複数のシステムにわたって区分化されるデータベース・ファイルを作成し、使用するために必要な情報が収録されています。システムの構成方法、ファイルの作成方法、およびアプリケーションにおけるファイルの使用方法について説明してあります。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒103-8510
東京都中央区日本橋箱崎町19番21号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書はプランニング目的としてのみ記述されています。記述内容は製品が使用可能になる前に変更になる場合があります。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式

においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。これらのサンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。 IBM は、お客様の当該サンプル・プログラムの使用から生ずるいかなる損害に対しても一切の責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。

© Copyright IBM Corp. _年を入れる_.

プログラミング・インターフェース情報

本書「Db2 for i SQL 解説書」には、プログラムを作成するユーザーが IBM i のサービスを使用するためのプログラミング・インターフェースが記述されています。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、『www.ibm.com/legal/copytrade.shtml』 をご覧ください。

Adobe、Adobe ロゴ、PostScript、PostScript ロゴは、Adobe Systems Incorporated の米国およびその他の国における登録商標または商標です。

Linux は、Linus Torvalds の米国およびその他の国における登録商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。

使用条件

これらの資料は、以下の条件に同意していただける場合に限りご使用いただけます。

個人使用**:** これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。た

だし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

商業的使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。 ただし、IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態で提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

[ワ行]

[数字]

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

［特殊文字］

プログラム番号: 5770-SS1

Printed in Japan

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21