IBM

IBM i
データベース
DB2 マルチシステム

7.1

IBM

IBM i
データベース
DB2 マルチシステム

7.1

ご注意!

本書および本書で紹介する製品をご使用になる前に、 65 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM i 7.1 (製品番号 5770-SS1)、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。このバージョンは、すべての RISC モデルで稼働するとは限りません。また CISC モデルでは稼働しません。

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

原典： IBM i
Database
DB2 Multisystem
7.1

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2010.4

目次

DB2 マルチシステム

DB2® マルチシステムの基本概念には、分散リレーショナル・データベース・ファイル、ノード・グループ、および区分化が含まれます。複数の System i® システム間で区分化されるデータベース・ファイルの作成と使用に必要な情報が記載されています。

システムの構成方法、ファイルの作成方法、およびアプリケーションにおけるファイルの使用方法について説明してあります。このトピックには、表の区分化に関する情報も記載されています。表の区分化は、単一システム上で区分化された表であり、マルチシステム区分化とは異なります。

注: コード例を使用する場合には、 64 ページの『コードに関するライセンス情報および特記事項』の条件に同意するものとします。

DB2 マルチシステムの PDF ファイル

この情報の PDF ファイルを表示または印刷できます。

本書の PDF 版を表示またはダウンロードするには、DB2 マルチシステムを選択します。

PDF ファイルの保存

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

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

Adobe Reader のダウンロード

これらの PDF を表示または印刷するには、Adobe® Reader がご使用のシステムにインストールされている必要があります。このアプリケーションは、 Adobe Web サイト

(www.adobe.com/products/acrobat/readstep.html) から無償でダウンロードできます。

関連資料

63 ページの『DB2 マルチシステムの関連情報』
他の Information Center のトピック・コレクションには、DB2 マルチシステムのトピック・コレクションに関連した情報が記載されています。

DB2 マルチシステムの概要

DB2 マルチシステム は、より大きなスケーラビリティーをデータベースに提供する並列処理技法です。

DB2 マルチシステムを使用すると、複数の System i モデル (最大 32 のシステム) を何も共用しないクラスターに同時に接続することができます。(*何も共用しない*とは、結合されたネットワーク内の各システムが、独自のメイン・メモリーとディスク装置を所有し、管理するということです。) システムが結合されるとすぐに、接続された各システム上の記憶装置にデータベース・ファイルを分散することができます。データベース・ファイルは、システムの集合間で区分化 (分散化) されたデータを持つことができます。また、

各システムはファイル内のすべてのデータにアクセスすることができます。しかしこのファイルは、ユーザーに対して、そのシステム上にあるローカル・ファイルのように振る舞います。ユーザーから見れば、データベースは単一のデータベースのように見えるため、ユーザーは、ネットワーク内のすべてのシステム間で並列的に QUERY を実行して、ファイル内のデータにリアルタイムでアクセスすることができます。

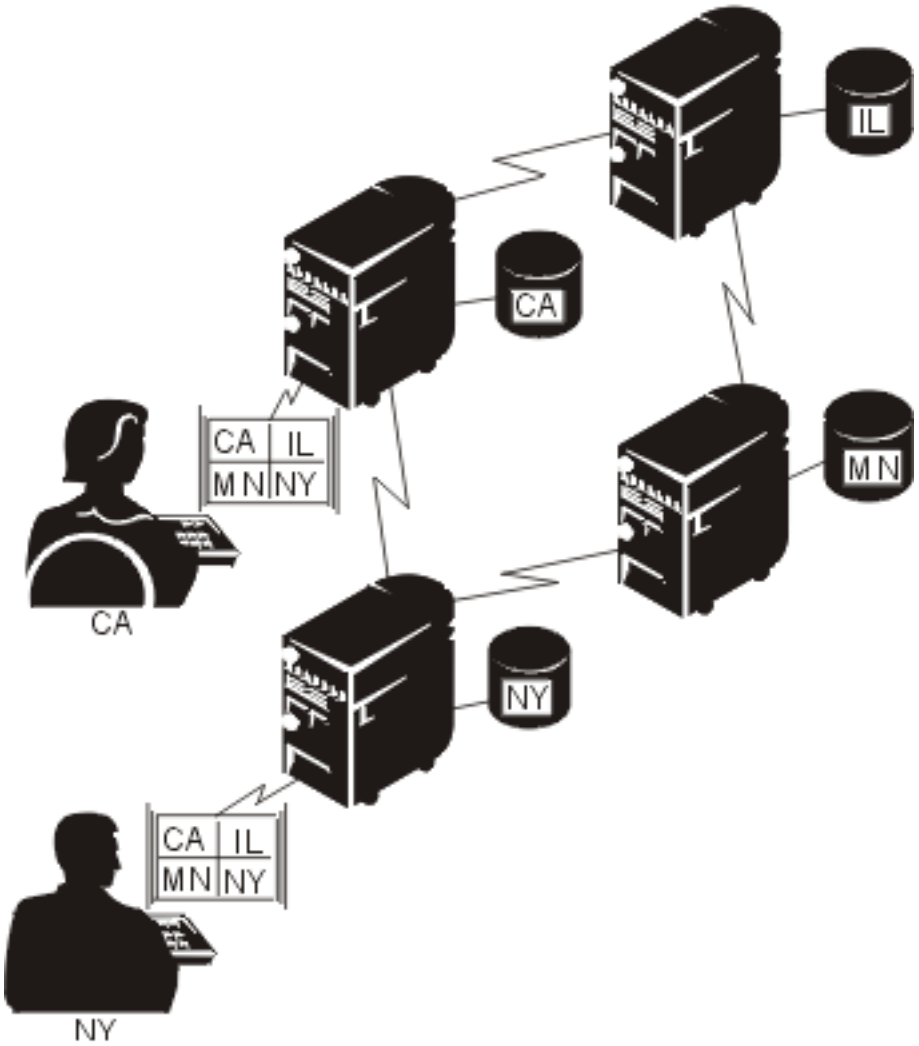


図 1. 複数システムでのデータベース・ファイルの分散

この並列処理技法を使用すると、あるシステムの使用頻度が高くても、ネットワーク内で接続されたその他のシステムのパフォーマンスが低下することはありません。大量のデータがあって、QUERY を実行する必要がある場合は、DB2 マルチシステムを使用すると、使用可能な最も効率的な方式の 1 つによって、これらの QUERY を実行することができます。ほとんどの場合、QUERY のパフォーマンスが向上します。QUERY がローカル・ファイルに対して実行されるのではなく、いくつかのシステム間で並列的に実行されるからです。

まだ DB2 マルチシステムを導入していない場合は、「i5/OS® および関連ソフトウェアのインストール、アップグレード、または削除」に記載されている、追加のライセンス・プログラムの導入に関する情報を参照してください。DB2 マルチシステムを導入するには、オペレーティング・システムの導入可能オプションのリストにあるオプション 27 を使用してください。

DB2 マルチシステム使用時の利点

DB2 マルチシステムを使用すると、QUERY のパフォーマンスの改善、データ複製の削減、データベース容量の増加などの利点があります。

DB2 マルチシステムを使用すると、次のような利点があります。

- QUERY を並列的に実行する (QUERY の各部分を、異なるシステム上で同時に実行する) ことによって、QUERY のパフォーマンスが向上する。
- すべてのシステムがすべてのデータにアクセスできるため、データ複製の必要性が少なくなる。
- より大きなデータベース・ファイルを収容することができる。
- アプリケーションで、リモート・データの位置を考える必要がなくなる。
- 規模を拡大する必要がある場合、さらに多くのシステム間でファイルを分配し直し、新しいシステム上でアプリケーションを変更せずに実行することができる。

DB2 マルチシステムでは、以前使用していたものと同じ入出力 (I/O) 方式 (GET、PUT、および UPDATE) またはファイル・アクセス方式を使用することができます。追加の、あるいは異なる入出力方式やファイル・アクセス方式は必要ありません。

アプリケーションを変更する必要はありません。どのような接続方式を使用していても、OptiConnect を使用していない限り、アプリケーションは、作成されたすべての分散ファイルで作動します。OptiConnect では、OptiConnect 制御装置記述を使用する必要があります。

関連情報

OptiConnect

DB2 マルチシステム: 基本的な用語と概念

分散ファイル とは、複数の System i モデルに分散されたデータベース・ファイルのことをいいます。以下に、DB2 マルチシステムによる分散ファイルの作成と使用に関する主な概念の一部を示します。

分散ファイルの一部を含む各システムは、*ノード* と呼ばれます。各システムは、リレーショナル・データベース・ディレクトリーで定義された名前によって識別されます。

1 つまたは複数の分散ファイルを含むシステムのグループは、*ノード・グループ* と呼ばれます。*ノード・グループ* とは、データが配布されるノードのリストが含まれているシステム・オブジェクトのことをいいます。システムは、複数のノード・グループにおける 1 つのノードの場合もあります。

以下の図は 2 つのノード・グループを示します。ノード・グループ 1 はシステム A、B、および C を含みます。ノード・グループ 2 はシステム A、B、および D を含みます。ある 1 つのシステムが複数のグループのノードとなることが可能なので、ノード・グループ 1 および 2 はシステム A および B を共有します。

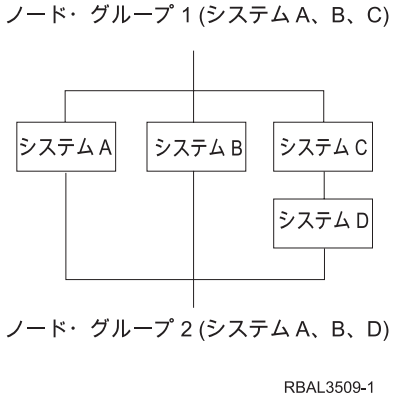


図 2. ノード・グループ

ファイルは、*区分化* によってノード・グループ内のすべてのシステム間に分散されます。 (区分化表で詳しく説明される) *表の区分化* は、単一システム上で区分化される表のことです。

区画番号 は、0 から 1023 までの数字です。各区画番号は、ノード・グループ内の 1 つのノードに割り当てられます。各ノードには、多数の区画番号を割り当てることができます。ノードと区画番号の相関は、*区画マップ* に格納されます。区画マップは、ノード・グループ・オブジェクトの一部としても格納されます。区画マップは、ノード・グループを作成するときに用意することができます。そうでない場合は、システムによりデフォルトのマップが作成されます。

区分化ファイルを使用することによって、区画マップを定義します。*区分化ファイル* とは、各区画番号のノード番号を定義する物理ファイルのことをいいます。

区分化キー は、分散されているファイル内の 1 つまたは複数のフィールドからなります。区分化キーは、ノード・グループ内のどのノードが、特定の値を持つ行を物理的に含むようにするかを判別するために使用されます。この判別は、*ハッシュ* を使用して行われます。ハッシュとは、オペレーティング・システムの機能の 1 つで、レコードの区分化キーの値をとって、それを区画番号にマップするものです。その区画番号に対応するノードが、レコードの格納に使用されます。

次の例は、2 つのシステムの分散表で、区画番号とノードがどのように表示されるかを示しています。この表には、LASTNAME という区分化キーがあります。

表 1. 区画マップ

区画番号	**ノード**
0	SYSA
1	SYSB
2	SYSA
3	SYSB

区画マップでは、区画番号 0 には SYSA が、区画番号 1 には SYSB が、区画番号 2 には SYSA が、区画番号 3 には SYSB が入ります。このパターンが繰り返されます。

区分化キーのハッシュでは、区画番号に対応する番号が判別されます。たとえば、`Andrews` という値を持つレコードは区画番号 1 にハッシュされ、`Anderson` という値を持つレコードは区画番号 2 にハッシュされたとします。表 1 に示された区画マップを参照すると、区画番号 1 のレコードは SYSB に格納されているのに対して、区画番号 2 のレコードは SYSA に格納されています。

関連概念

25 ページの『区分化表』
DB2 for i は、SQL を使用して区分化表をサポートします。

DB2 マルチシステムでのノード・グループ: 概要

一連の System i モデルからデータベース・ファイルを見ることができるようにするためには、このファイルを置きたいシステムのグループ (ノード・グループ) をまず定義する必要があります。

ノード・グループには、2 つから 32 のノードを定義することができます。ノード・グループに定義されるノードの数によって、データベース・ファイルが作成されるシステムの数が決まります。ローカル・システムは、ノード・グループに指定されたシステムのうちの 1 つでなければなりません。システムがノード・グループを作成する場合、各ノードには、1 から始まる番号が割り当てられます。

DB2 マルチシステムでのノード・グループの仕組み

ノード・グループ とは、作成されたシステム上に格納されたシステム・オブジェクト (*NODGRP) のことをいいます。

ノード・グループは、分散オブジェクトではありません。*NODGRP システム・オブジェクトには、データ・ファイル内のデータがどのように区分化 (分配) されるかに関する情報とともに、グループ内のシステムに関するすべての情報が含まれます。デフォルトの区分化では、各システム (ノード) がデータの等しい共用分を受け取ります。

区分化は、ハッシュ・アルゴリズムを使用して処理されます。ノード・グループが作成されると、0 から 1023 の範囲内の区画番号が、そのノード・グループに対応付けられます。デフォルトの区分化では、等しい数の区画が、ノード・グループ内の各ノードに割り当てられます。 データがファイルに追加されると、区分化キー内のデータがハッシュされ、区画番号が作成されます。データのレコード全体はハッシュされません。区分化キー内のデータだけが、ハッシュ・アルゴリズムを通してハッシュされます。結果の区画番号に対応するノードは、データのレコードが物理的に存在する場所です。したがって、デフォルトの区分化では、十分なデータ・レコードと広範囲の値がある場合、各ノードはデータの等しい共用分を格納します。

各ノードでデータの等しい共用分を受け取りたくない場合、またはデータの特定部分を所有するシステムを指定したい場合は、ノード・グループの作成 (CRTNODGRP) コマンドにファイルの区分化 (PTNFILE) パラメーターを使用してカスタム区分化構造を指定するか、またはノード・グループ属性の変更 (CHGNODGRPA) コマンドを使用して区分化構造をあとで変更することによって、データの区分化の方法を変更することができます。PTNFILE パラメーターを使用すると、ノード・グループ内の各区画にノード番号を設定することができます。すなわち、PTNFILE パラメーターを使用すると、ノード・グループ内のシステムでのデータの区分化方法を調整することができます。(PTNFILE パラメーターは、ノード・グループの作成コマンドのトピックの例で使用しています。)

ノード・グループはシステム・オブジェクトであるため、オブジェクトの保管 (SAVOBJ) コマンドとオブジェクトの復元 (RSTOBJ) コマンドを使用して保管することができます。ノード・グループ・オブジェクトは、作成されたシステムか、またはノード・グループ内のいずれかのシステムに復元することができます。ノード・グループ・オブジェクトがノード・グループ内にないシステムに復元された場合、オブジェクトは使用できません。

関連概念

『ノード・グループの作成コマンド』
2 つの CL コマンドの例で、ノード・グループの作成 (CRTNODGRP) コマンドによってノード・グループを作成する方法を示します。

21 ページの『DB2 マルチシステムでの区分化』
区分化 とは、ノード・グループ内でノード間にファイルを配布するプロセスのことをいいます。

DB2 マルチシステムでノード・グループ・コマンドを使用する前に完了しておくべきタスク

ノード・グループの作成 (CRTNODGRP) コマンドまたはノード・グループ・コマンドのいずれかを使用するにあたっては、使用している分散リレーショナル・データベース・ネットワークが正しく設定されていることを確認する必要があります。

これが新しい分散リレーショナル・データベース・ネットワークである場合は、分散データベース・プログラミングの情報を使用すると、ネットワークの確立に役立ちます。

ネットワーク内の 1 つのシステムが、ローカル (*LOCAL) システムとして定義されていることを確認する必要があります。RDB (リレーショナル・データベース) ディレクトリー項目の処理 (WRKRDBDIRE) コマンドを使用して、項目に関する詳細を表示してください。ローカル・システムが定義されていない場合は、RDB ディレクトリー項目の追加 (ADDRDBDIRE) コマンドのリモート・ロケーション名 (RMTLOCNAME) パラメーターに *LOCAL を指定することによって、定義することができます。次に例を示します。

```
ADDRDBDIRE  RDB(MP000) RMTLOCNAME(*LOCAL) TEXT ('New York')
```

New York の MP000 という名前のシステムは、リレーショナル・データベース・ディレクトリーにローカル・システムとして定義されます。ネットワーク構成内のシステムのシステム名またはローカル・ロケーション名として定義できるローカル・リレーショナル・データベースは 1 つだけです。これは、特にネットワークが複雑な場合、データベース名を識別して、それを分散リレーショナル・データベース内の特定のシステムに関連付けるのに役立ちます。

DB2 マルチシステムが、定義したノード・グループ内のシステムにファイルを正しく分配するためには、リモート・データベース (RDB) 名を、そのノード・グループ内のすべてのノード (システム) で整合性のあるものにする必要があります。

たとえば、ノード・グループに 3 つのシステムを用意する場合、各システムは少なくとも 3 つの項目を RDB ディレクトリーに持っている必要があります。各システムでは、3 つの名前がすべて同じでなければなりません。3 つのシステムのそれぞれにおいて、ローカル・システムを示す 1 つの項目が存在します。これは、*LOCAL と識別されます。他の 2 つの項目には、適切なリモート・ロケーション情報が含まれます。

関連概念

分散データベース・プログラミング

ノード・グループの作成コマンド

2 つの CL コマンドの例で、ノード・グループの作成 (CRTNODGRP) コマンドによってノード・グループを作成する方法を示します。

次の例では、デフォルトの区分化 (システムでの等しい区分化) を使用するノード・グループを作成しています。

```
CRTNODGRP NODGRP(LIB1/GROUP1) RDB(SYSTEMA SYSTEMB SYSTEMC SYSTEMD)
          TEXT('Node group for test files')
```

この例では、コマンドは 4 つのノードを含むノード・グループを作成しています。各ノードには、RDB 項目 (ADDRDBDIRE コマンドを使用してリレーショナル・データベース・ディレクトリーに前に追加されたもの) を定義する必要があり、1 つのノードをローカル (*LOCAL) として定義する必要があることに注意してください。

区分化属性はデフォルト解釈として、各ノード番号に区画の 4 分の 1 が割り当てられることになります。このノード・グループを、物理ファイルの作成 (CRTPF) コマンドの NODGRP パラメーターで使用すれば、分散ファイルを作成することができます。

次の例では、指定の区分化を使用するノード・グループが、ファイルの区分化 (PTNFILE) パラメーターを使用して作成されます。

```
CRTNODGRP NODGRP(LIB1/GROUP2) RDB(SYSTEMA SYSTEMB SYSTEMC)
          PTNFILE(LIB1/PTN1)
          TEXT('Partition most of the data to SYSTEMA')
```

この例では、コマンドは 3 つのノード (SYSTEMA、SYSTEMB、および SYSTEMC) を含むノード・グループを作成しています。区分化属性は、PTN1 と呼ばれるファイルからとられています。このファイルを設定すると、さらに多い割合のレコードを特定のシステム上に置くことができます。

この例の PTN1 は区分化ファイルです。このファイルは分散ファイルではなく、通常のローカル物理ファイルであり、カスタム区分化構造を設定するために使用することができます。区分化ファイルには、2 進数フィールドが 1 つ必要です。区分化ファイルには 1024 のレコードが必要であり、各レコードには有効なノード番号が含まれます。

2 バイトの 2 進数フィールド

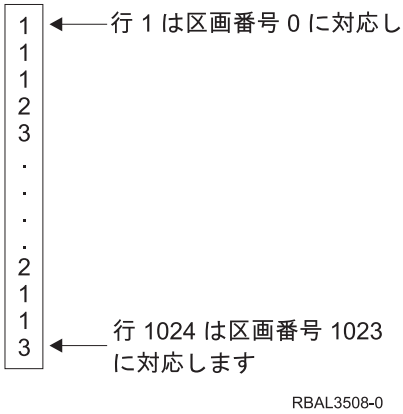


図 3. 区分化ファイル PTNFILE の内容の例

ノード・グループに 3 つのノードがある場合、区分化ファイルのレコードすべてに 1、2、または 3 という番号が必要です。ノード番号は、RDB 名がノード・グループの作成 (CRTNODGRP) コマンドに指定された順序で割り当てられます。区分化ファイルにそのノード番号を含むレコードをさらに多く持たせることによって、より多い割合のデータを特定ノードに与えることができます。これは、各システムに物理的に存在するデータの量に関して、区分化をカスタマイズする方式です。特定のノードに存在する特定の値に関して区分化をカスタマイズするには、ノード・グループ属性の変更 (CHGNODGRPA) コマンドを使用してください。

ノード・グループ情報は分散ファイルに格納されるため、ファイルは、ノード・グループ内の変更、またはノード・グループに含まれる RDB ディレクトリー項目の変更に対してすぐには対応しないことに注意する必要があります。ノード・グループおよび RDB ディレクトリー項目に変更を加えることはできますが、CHGPF コマンドを使用して変更されるノード・グループを指定するまで、ファイルの動作は変更されません。

他の概念として、*可視ノード* というものがあります。ノード・グループ内の可視ノードには、ファイル・オブジェクト (ファイルをいくつかのノードに分散できるようにするメカニズムの一部) が含まれますが、データは含まれません。可視ノードは、常に現行レベルのファイル・オブジェクトを保存します。可視ノードにはデータが格納されません。反対に、ノード (データ・ノード と呼ばれることもある) にはデータが含まれます。可視ノードをノード・グループで使用する例として、営業部長が使用している System i 製品をノード・グループの一部にします。これらの部長は、定期的には QUERY を実行しないで、特定の QUERY を実行したい場合がよくあります。それらのシステムから、部長は各自の QUERY を実行して、リアルタイム・データにアクセスし、その QUERY の結果を受け取ることができます。したがって、データがそれらのシステムにまったく格納されていない場合でも、それらのシステムは可視ノードであるため、部長は必要に応じていつでも QUERY を実行することができます。

可視ノードとしてノードを指定するには、PTNFILE パラメーターをノード・グループの作成 (CRTNODGRP) コマンドに使用する必要があります。区分化ファイルに特定のノード番号のレコードが含まれない場合、そのノードは可視ノードです。

関連概念

5 ページの『DB2 マルチシステムでのノード・グループの仕組み』
ノード・グループ とは、作成されたシステム上に格納されたシステム・オブジェクト (*NODGRP) のことをいいます。

11 ページの『DB2 マルチシステムでの分散ファイル』
分散ファイル とは、複数の System i モデルに分散されたデータベース・ファイルのことをいいます。

10 ページの『ノード・グループ属性の変更コマンド』
ノード・グループ属性の変更 (CHGNODGRPA) コマンドは、ノード・グループのデータ区分化属性を変更します。

ノード・グループの表示コマンド

ノード・グループの表示 (DSPNODGRP) コマンドは、ノード・グループ内のノード (システム) を表示します。

また、ノード・グループの区分化構造も表示されます (区分化については、『DB2 マルチシステムでの区分化』で説明します)。

次の例は、GROUP1 という名前のノード・グループを、そのノード・グループに関連する区分化構造とともに表示する方法を示しています。この情報は、ワークステーションに表示されます。Information Center のトピック「制御言語 (CL)」に、DSPNODGRP コマンドに関する詳細情報があります。

```
DSPNODGRP NODGRP(LIB1/GROUP1)
```

ノード・グループ名を指定して DSPNODGRP コマンドを実行すると、ノード・グループの表示画面が示されます。この画面には、システム名 (リレーショナル・データベースの列に示されたもの) と、そのシステムに割り当てられたノード番号が示されます。これは、どのシステムがどのノード番号を持つかを決めるための直接的な方式です。

```
                          ノード・グループの表示

ノード・グループ :   GROUP1          ライブラリー :   LIB1

 リレーショナル・データベース       ノード番号
      SYSTEMA                          1
      SYSTEMB                          2
      SYSTEMC                          3

                                                                終わり
F3= 終了    F11= 区画データ    F12= 取消し    F17= 最上部    F18= 最下部
```

図 4. ノード・グループの表示: データベースとノード番号の相関

各区画番号に割り当てられたノード番号を見るには、 F11 (区画設定データ) をノード・グループの表示画面で使用します。次の画面は、各区画番号に割り当てられたノード番号を示しています。システムとノード番号間での (またはノード番号からシステムへの) マッピングは、DSPNODGRP コマンドを使用して簡単に実行することができます。

```
                          ノード・グループの表示

ノード・グループ :   GROUP1          ライブラリー :   LIB1

 区画番号        ノード番号
       0               1
       1               2
       2               3
       3               1
       4               2
       5               3
       6               1
       7               2
       8               3
       9               1
      10               2
      11               3
      12               1
      13               2
      14               3
      15               1
                                                          続く ...
F3= 終了    F11= ノード・データ    F12= 取消し    F17= 最上部    F18= 最下部
```

図 5. ノード・グループの表示: 区画番号とノード番号の相関

次の例は、GROUP2 というノード・グループ内のシステムのリストを、関連する区分化構造とともに印刷するものです。

```
DSPNODGRP NODGRP(LIB1/GROUP2) OUTPUT(*PRINT)
```

関連概念

21 ページの『DB2 マルチシステムでの区分化』
区分化 とは、ノード・グループ内でノード間にファイルを配布するプロセスのことをいいます。

関連資料

制御言語

ノード・グループ属性の変更コマンド

ノード・グループ属性の変更 (CHGNODGRPA) コマンドは、ノード・グループのデータ区分化属性を変更します。

ノード・グループには、1024 の区画を持つ表が含まれます。各区画にはノード番号があります。ノード番号は、ノード・グループの作成時に割り当てられ、ノード・グループの作成 (CRTNODGRP) コマンドの RDB パラメーターに指定されたリレーショナル・データベースに対応します。有効なノード番号値と、ノード番号とリレーショナル・データベース名の相関を見るには、ノード・グループの表示 (DSPNODGRP) コマンドを使用します。

CHGNODGRPA コマンドは、指定のノード・グループを使用して作成された既存の分散ファイルに影響を与えません。変更されるノード・グループを使用するには、その変更されるノード・グループを、新しいファイルの作成時、または物理ファイルの変更 (CHGPF) コマンドに指定する必要があります。Information Center のトピック「制御言語 (CL)」に、CHGNODGRPA コマンドの詳細情報があります。

この最初の例には、ライブラリー LIB1 の GROUP1 というノード・グループの区分化属性を変更する方法が示されています。

```
CHGNODGRPA NODGRP(LIB1/GROUP1) PTNNBR(1019)
           NODNBR(2)
```

この例では、区画番号 1019 が指定され、1019 にハッシュされるレコードはすべて、ノード番号 2 に書き込まれます。これにより、特定の区画番号をノード・グループ内の特定のノードに指定することができます。

2 番目の例では、GROUP2 という名前のノード・グループの区分化属性が変更されています。(GROUP2 は、ライブラリー探索リスト *LIBL を使用して検出されます。) 比較データ値 (CMPDTA) パラメーターに指定された値がハッシュされて、その結果の区画番号は、既存のノード番号からノード番号 3 に変更されます。(ハッシュと区分化については、『DB2 マルチシステムでの区分化』に説明があります。)

```
CHGNODGRPA NODGRP(GROUP2) CMPDTA('CHICAGO')
           NODNBR(3)
```

このノード・グループを使用して作成されたファイルと、文字フィールドからなる区分化キーを持つファイルはすべて、'CHICAGO' を含むレコードをノード番号 3 の区分化キーに格納します。区分化キーに複数のフィールドを持つファイルを可能にするために、最大 300 の値をデータ比較 (CMPDTA) パラメーターに指定することができます。

CMPDTA パラメーターに値を入力する場合は、文字データの大文字小文字を区別するように注意が必要です。すなわち、'Chicago' と 'CHICAGO' では同じ区画番号になりません。数値データは、単なる数字として入力する必要があります。小数点、先行ゼロ、または後続ゼロは使用しないでください。

すべての値は、区画番号を獲得するためにハッシュされます。これらは次に、ノード番号 (NODNBR) パラメーターに指定されたノード番号に対応付けられます。完了メッセージ CPC3207 のテキストは、変更された区画番号を示します。CHGNODGRPA コマンドを何度も、異なる多数の値に対して出すと、同じ区画番

号を 2 回変更する確率が高くなることに注意してください。このことが起こると、最新の変更において指定されたノード番号が、ノード・グループに対して有効になります。

関連概念

6 ページの『ノード・グループの作成コマンド』
2 つの CL コマンドの例で、ノード・グループの作成 (CRTNODGRP) コマンドによってノード・グループを作成する方法を示します。

21 ページの『DB2 マルチシステムでの区分化』
区分化 とは、ノード・グループ内でノード間にファイルを配布するプロセスのことをいいます。

24 ページの『DB2 マルチシステムでのデータ分散のカスタマイズ』
データの配置についてはシステムが担当するため、ユーザーはレコードの実際の所在を知る必要はありません。ただし、特定のレコードが特定のシステムに常に格納されるようにしたい場合は、ノード・グループ属性の変更 (CHGNODGRPA) コマンドを使用して、これらのレコードの所在を指定することができます。

関連資料

制御言語

ノード・グループの削除コマンド

ノード・グループの削除 (DLTNODGRP) コマンドは、以前に作成されたノード・グループを削除します。

このコマンドは、ノード・グループを使用して作成されたファイルには影響を与えません。

次の例は、GROUP1 という名前のノード・グループを削除する方法を示しています。このノード・グループによって作成されたファイルはいずれも影響を受けません。

```
DLTNODGRP NODGRP(LIB1/GROUP1)
```

ノード・グループの削除によって、そのノード・グループを使用して作成されたファイルは影響を受けませんが、ノード・グループの使用後にそれらを削除することはお勧めできません。ノード・グループを削除するとすぐに、DSPNODGRP コマンドを使用して、ノードと区分化構造を表示することができなくなるためです。ただし、ファイル記述の表示 (DSPFD) コマンドを使用して TYPE(*NODGRP) を指定すると、特定のファイルに関連するノード・グループを表示することができます。

関連資料

制御言語

DB2 マルチシステムでの分散ファイル

分散ファイル とは、複数の System i モデルに分散されたデータベース・ファイルのことをいいます。

分散ファイルは更新でき、SQL、QUERY ツール、および物理ファイル・メンバーの表示 (DSPPFM) コマンドなどの方法を使用してアクセスすることができます。データベース操作の場合、分散ファイルは、ほとんどの部分でローカル・ファイルのように処理されます。分散ファイルに対する CL コマンドの変化については、『分散ファイルに対する CL コマンドの動作方法』のトピックに情報があります。

データベース・ファイルを分散ファイルとして作成するには、物理ファイルの作成 (CRTPF) コマンドまたは SQL CREATE TABLE ステートメントのいずれかを使用することができます。このトピックでは、両方の方法についてさらに詳しく説明します。CRTPF コマンドには 2 つのパラメーターがあります。これらはノード・グループ (NODGRP) と区分化キー (PTNKEY) で、ファイルを分散ファイルとして作成します。分散データベース・ファイルは、CRTPF コマンドの実行時に指定した名前とライブラリーを持つオブジェクト (PF 属性を持つ *FILE) として表示されます。

既存の非分散データベース物理ファイルを分散ファイルに変更したい場合は、物理ファイルの変更 (CHGPF) コマンドを使用して、これを行うことができます。CHGPF コマンドでは、NODGRP および PTNKEY の各パラメーターを使用して、分散ファイルに変更を加えることができます。CHGPF コマンドを使用すると、NODGRP および PTNKEY の各パラメーターの値を指定することによって、既存の分散データベース・ファイルのデータ区分化属性を変更することもできます。これらのパラメーターに値を指定すると、ノード・グループ内の区分化表に従って、データが再分配されます。

注: 分散ファイルに変更される物理ファイルに基づく論理ファイルもすべて、分散ファイルになります。大きなファイルまたは大規模ネットワークの場合、データの再分配に長い時間がかかる可能性があるため、頻繁に行わないようにしてください。

関連概念

6 ページの『ノード・グループの作成コマンド』
2 つの CL コマンドの例で、ノード・グループの作成 (CRTNODGRP) コマンドによってノード・グループを作成する方法を示します。

16 ページの『分散ファイルに対する CL コマンドの動作方法』
分散ファイルのシステム・オブジェクト・タイプは *FILE であるため、物理ファイルにアクセスする CL コマンドの多くは、分散ファイルに対して実行することができます。ただし、一部の CL コマンドは、非分散ファイルではなく分散ファイルに対して出されると、動作が変わります。

物理ファイルの作成コマンドおよび SQL CREATE TABLE ステートメント

物理ファイルの作成 (CRTPF) コマンドまたは SQL CREATE TABLE ステートメントのいずれかを使用することにより分散物理ファイルを作成できます。

区分化ファイルを作成したい場合は、次のパラメーターを CRTPF コマンドに指定する必要があります。

- NODGRP パラメーターのノード・グループ名
- 区分化キーとして使用される 1 つまたは複数のフィールド (PTNKEY パラメーターを使用)

区分化キーは、データの各レコードが物理的に存在する場所 (ノード) を判別します。区分化キーは、CRTPF コマンドの実行時、または SQL CREATE TABLE ステートメントの実行時に指定します。各レコードの区分化キーを構成するフィールドの値は、ハッシュ・アルゴリズムによって処理されて、レコードの所在を決めます。

単一フィールドの区分化キーがある場合は、そのフィールド内で同じ値を持つレコードすべてが、同じシステム上に存在します。

分散物理ファイルを作成したい場合は、ユーザー・プロファイルがノード・グループ内のすべてのノードに存在しなければなりません。また、ユーザー・プロファイルに、すべてのノードで分散ファイルを作成するために必要な権限がなければなりません。特定のライブラリーに分散ファイルを作成する必要がある場合は、そのライブラリーがノード・グループ内のすべてのノードに存在しなければなりません。また、ユーザー・プロファイルに、これらのライブラリーにファイルを作成するために必要な権限がなければなりません。これらの要素があてはまらない場合、ファイルは作成されません。

システムの構成方法によっては、リモート・システムで使用しているユーザー・プロファイルに影響することがあります。ユーザー・プロファイルをリモート・システムで確実に使用されるようにするには、そのシステムをセキュア・ロケーションとして構成してください。システムがセキュアなロケーションとして構成されているかどうかを判別するには、構成リストの処理 (WRKCFGL) コマンドを使用します。

次の例は、区分化され (NODGRP パラメーターを使用して指定され)、社員番号 (EMPNUM) フィールドに単一の区分化キーを持つ PAYROLL という名前の物理ファイルを作成する方法を示しています。

```
CRTPF FILE(PRODLIB/PAYROLL) SCRFILE(PRODLIB/DDS) SRCMBR(PAYROLL)
      NODGRP(PRODLIB/PRODGROUP) PTNKEY(EMPNUM)
```

CRTPF コマンドを実行すると、システムで分散物理ファイルが作成され、分散ファイルに関連するローカル・データが保持されます。CRTPF コマンドは、ノード・グループに指定されたすべてのリモート・システムでも物理ファイルを作成します。

物理ファイルの所有権とすべてのシステムでの共通権限は、整合性がとれています。この整合性には、CRTPF コマンドの AUT パラメーターに指定された権限もすべて含まれています。

SQL CREATE TABLE ステートメントは、ノード・グループと区分化キーを指定するために使用することもできます。次の例では、PAYROLL という SQL 表が作成されます。この例では、IN *ノード・グループ名* 文節と PARTITIONING KEY 文節を使用しています。

```
CREATE TABLE PRODLIB/PAYROLL
             (EMPNUM INT, EMPLNAME CHAR(12), EMPFNAME CHAR (12))
             IN PRODLIB/PRODGROUP
             PARTITIONING KEY (EMPNUM)
```

PARTITIONING KEY 文節が指定されていない場合、1 次キーの最初の列が定義されていれば、それが最初の区分化キーとして使用されます。 1 次キーが定義されていない場合は、日付、時刻、タイム・スタンプ、または浮動小数点のデータ型を持たない表に定義された最初の列が、区分化キーとして使用されます。

ファイルが区分化されているかどうかを表示するには、ファイル記述の表示 (DSPFD) コマンドを使用します。ファイルが区分化されている場合、DSPFD コマンドはノード・グループの名前およびファイル・オブジェクトにあるノード・グループ (区画マップ全体を含む) の詳細を示し、区分化キーのフィールドをリストします。

DB2 マルチシステムで分散ファイルを使用する際に知っておかなければならない制約事項のリストは、『DB2 マルチシステムにおいて分散ファイルを作成または処理するときの制限』のトピックにあります。

関連概念

分散データベース・プログラミング

DB2 マルチシステムにおいて分散ファイルを作成または処理するときの制限

分散ファイルを作成または処理する場合は、いくつかの制限に注意する必要があります。

以下のような制限があります。

- 先変更先出し (FCFO) アクセス・パスは使用することができません。これは、複数のノード間でアクセス・パスが区分化されるためです。
- 分散ファイルが持つことができるメンバーの数は最大 1 つです。
- 分散ファイルは、一時ライブラリー (QTEMP) では認められません。
- 区分化キー内のデータの更新能力は限定されています。一般に、区分化キーを選択する場合は、値が更新されないフィールドを選択する必要があります。区分化キーの更新は、更新によってレコードが異なるノードに区分化されない限り許されます。
- 日付、時刻、タイム・スタンプ、または浮動小数点の各数字フィールドは、区分化キーには使用することができません。
- ソース物理ファイルはサポートされていません。

- 外部記述ファイルは分散ファイルでサポートされています。プログラム記述ファイルはサポートされていません。
- アクセス・パスが固有の場合、区分化キーは、固有キー・アクセス・パスの一部でなければなりません。
- 制約はサポートされています。参照制約は、親ファイルおよび外部キー・ファイルの両方のノード・グループが同じであり、区分化キーのすべての フィールドが制約に含まれている場合にのみサポートされます。区分化キーは、制約フィールドの一部でなければなりません。また、固有の 1 次制約の場合、アクセス・パスが固有であれば、区分化キーは固有キー・アクセス・パスのサブセットでなければなりません。
- CRTPF コマンドでは、システム・パラメーターには *LCL という値 (CRTPF SYSTEM(*LCL)) が指定されていなければなりません。 SYSTEM(*RMT) は許可されません。
- 論理ファイルが分散ファイルに対して作成されると、その論理ファイルも分散ファイルになります。すなわち、指定のノードにある物理ファイルの一部に対してだけローカル論理ファイルを作成することはできないということです。SQL ビューは、ビューが結合であり、基礎となる物理ファイルが同じノード・グループを持たない場合、この例外となります。この場合、ビューはローカル・システムでのみ作成されます。このビューが分配されない場合でも、そのビューを QUERY すると、そのビューが作成されたノードだけではなく、すべてのノードからデータが検索されます。

  結合ファイルは、SQL を使用したときのみ作成できます。

  DDS 作成論理ファイルには、基礎ファイルが 1 つだけ認められています。
- コード化文字セット識別コード (CCSID) と分類順序 (SRTSEQ) 表は、元のシステムから変換されます。
- 可変長レコード (VLR) 処理はサポートされていません。これは、可変長フィールドが分散ファイルでサポートされていないということではありません。この制限は、ファイルのオープン時に VLR 処理を要求する言語とアプリケーションだけに当てはまります。
- ファイル終わり遅延 (EOFDLY) 処理はサポートされていません。
- データ・ファイル・ユーティリティー (DFU) は、分散ファイルに対しては機能しません。これは、DFU が相対レコード番号処理を使用してレコードにアクセスするためです。
- 分散ファイルを、独立補助記憶域プール (IASP) に存在するライブラリーの中に作成することはできません。

分散ファイルを作成した後のシステムの活動

ファイルが作成されるとすぐに、システムはそのデータを確実に区分化し、ファイルを並行レベルで維持します。

ファイルが作成されるとすぐに、以下の活動が自動的に行われます。

- ファイルで作成されるすべての索引は、すべてのノードで作成されます。
- DB2 マルチシステムでの分散ファイルの使用に関する権限の変更情報は、すべてのノードに送信されます。
- システムによって、ファイルの移動とそのライブラリーの名前変更が防止されます。
- ファイル自体の名前が変更されると、その新しい名前がすべてのノードに反映されます。
- オブジェクトの割り振り (ALCOBJ)、物理ファイル・メンバーの再編成 (RGZPFM)、およびジャーナル物理ファイルの開始 (STRJRNPF) などのいくつかのコマンドは、ここでファイルのすべての部分に影響を与えます。これにより、区分化ファイルを処理するときに、ローカル・ファイルという概念を維持す

ることができます。これらの CL コマンドの完全なリストについては、『CL コマンド: DB2 マルチシステムでの分散ファイルのすべての部分に影響を与えるもの』を参照してください。

オブジェクト割り振り (ALCOBJ) コマンドは、ノード・グループ内のどのノードからでも出すことができます。これにより、すべての部分がロックされて、ローカル・ファイルを割り振るときに付与されるのと同じ保全性が確保されます。これらのアクションはすべてシステムによって処理されるため、各ノードにコマンドを入力する必要がなくなります。

物理ファイル・ジャーナルの開始 (STRJRNPF) コマンドの場合、ジャーナル処理は各システムで開始されます。したがって、各システムは独自のジャーナルとジャーナル・レシーバーを備えている必要があります。各システムには独自のジャーナル項目があります。このジャーナル項目を使用する回復は、各システムに対して個々に実行されなければなりません。ジャーナル処理を開始、終了するコマンドは、ノード・グループ内のすべてのシステムに同時に影響を与えます。

- オブジェクトのダンプ (DMPOBJ)、オブジェクトの保管 (SAVOBJ)、およびオブジェクト・ロックの処理 (WRKOBJLCK) などのいくつかのコマンドは、そのコマンドが出されたシステム上のファイルの部分にのみ影響を与えます。これらの CL コマンドの完全なリストについては、『CL コマンド: DB2 マルチシステムでの分散ファイルのローカル部分にのみ影響を与えるもの』を参照してください。

あるファイルが分散ファイルとして作成されてからすぐにそのファイルをオープンすると、すべてのリモート・システムへの接続が作成されるとともに、ファイルのローカル部分がオープンされます。作成されたファイルは、ノード・グループ内のすべてのシステムからアクセスすることができます。システムは、ファイル入出力タスク (たとえば、GETS、PUT、および UPDATES) を完了するためにどのノードとレコードを使用する必要があるかも判別します。この活動のいずれかを物理的に指示したり、指定する必要はありません。

Distributed Relational Database Architecture™ (DRDA®) の要求と分散データ管理機能 (DDM) の要求は、分散ファイルをターゲットにすることができることに注意してください。 DRDA または DDM を使用してリモート・システム上のデータベース・ファイルにアクセスする、以前に配布されたアプリケーションは、データベース・ファイルが分散ファイルに変更された場合でも、引き続き機能することができます。

レコードの到着順が、ローカル・データベース・ファイルと分散データベース・ファイルでは異なることに注意する必要があります。

分散ファイルはシステムにまたがって物理的に分配されるため、レコードの到着順または相対レコード番号に依存することはできません。ローカル・データベース・ファイルでは、レコードは順番に処理されます。たとえば、ローカル・データベース・ファイルにデータを挿入する場合、そのデータは、最初のレコードから最後のレコードへという順序で挿入されます。すべてのレコードが順番に挿入されます。各ノードのレコードは、レコードがローカル・ファイルに挿入される場合と同じ方法で挿入されます。

データがローカル・データベース・ファイルから読み取られると、そのデータは、最初のレコードから最後のレコードへと読み取られます。これは、分散データベース・ファイルの場合にはあてはまりません。分散データベース・ファイルでは、データベースは最初のノードの最初のレコード (最初のレコードから最後のレコードへ)、次に 2 番目のノードという順序で読み取られます。たとえば、レコード 27 に対する読み取りは、単一レコードに対する読み取りではなくなります。分散ファイルでは、ノード・グループ内の各ノードに、それぞれ別のレコード 27 を含むことができます。

関連概念

18 ページの『CL コマンド: DB2 マルチシステムでの分散ファイルのすべての部分に影響を与えるもの』
一部の CL コマンドを実行すると、分散ファイルのすべての部分に影響が及びます。

19 ページの『DB2 マルチシステムでのジャーナル処理の考慮事項』
物理ファイル・ジャーナルの開始 (STRJRNPF) コマンドと物理ファイル・ジャーナルの終了 (ENDJRNPF) コマンドは、他のシステムに配布されますが、実際のジャーナル処理は、各システムで独立して、各システムの独自のジャーナル・レシーバーに対して起こります。

『CL コマンド: DB2 マルチシステムでの分散ファイルのローカル部分にのみ影響を与えるもの』
一部の CL コマンドを実行すると、ローカル・システム上にある分散ファイルの部分にのみ影響が及びます (ローカル・システムとは、コマンドが実行されるシステムです)。

分散ファイルに対する CL コマンドの動作方法

分散ファイルのシステム・オブジェクト・タイプは *FILE であるため、物理ファイルにアクセスする CL コマンドの多くは、分散ファイルに対して実行することができます。ただし、一部の CL コマンドは、非分散ファイルではなく分散ファイルに対して出されると、動作が変わります。

関連概念

11 ページの『DB2 マルチシステムでの分散ファイル』
分散ファイル とは、複数の System i モデルに分散されたデータベース・ファイルのことをいいます。

関連資料

制御言語

CL コマンド: DB2 マルチシステムでの分散ファイルに対して実行できないもの:

CL コマンドまたは特定のパラメーターの一部は、分散ファイルに対して実行することができません。

これらの CL コマンドまたはパラメーターは、以下のとおりです。

- 論理ファイル・メンバーの変更 (CHGLFM) の SHARE パラメーター
- 物理ファイル・メンバーの変更 (CHGPFM) の SHARE パラメーター
- 複写オブジェクトの作成 (CRTDUPOBJ)
- 物理ファイル・メンバーの初期化 (INZPFM)
- オブジェクトの移動 (MOVOBJ)
- データベース・ファイルの位置決め (POSDBF)
- メンバーの除去 (RMVM)
- 分散ファイルを含むライブラリーでのライブラリーの名前変更 (RNMLIB)
- メンバーの名前変更 (RNMM)
- 統合ファイル・システム・コマンド、COPY

CL コマンド: DB2 マルチシステムでの分散ファイルのローカル部分にのみ影響を与えるもの:

一部の CL コマンドを実行すると、ローカル・システム上にある分散ファイルの部分にのみ影響が及びます (ローカル・システムとは、コマンドが実行されるシステムです)。

これらの CL コマンドは、以下のとおりです。

- ジャーナル処理済み変更適用 (APYJRNCHG)。このコマンドの追加情報については、『DB2 マルチシステムでのジャーナル処理の考慮事項』を参照してください。
- オブジェクト記述の表示 (DSPOBJD)

- オブジェクトのダンプ (DMPOBJ)
- ジャーナル・アクセス・パスの終了 (ENDJRNAP)
- ジャーナル処理済み変更の除去 (RMVJRNCHG)。このコマンドの追加情報については、『DB2 マルチシステムでのジャーナル処理の考慮事項』を参照してください。
- オブジェクトの復元 (RSTOBJ)
- オブジェクトの保管 (SAVOBJ)
- ジャーナル・アクセス・パスの開始 (STRJRNAP)

リモート・コマンド投入 (SBMRMTCMD) コマンドを使用すると、分散ファイルに関連するすべてのリモート・システムに対して、任意の CL コマンドを出すことができます。 CL コマンドをローカル・システムに出してから、分散ファイルで SBMRMTCMD コマンドを介して同じコマンドを出すと、分散ファイルのすべてのシステムに対して、CL コマンドを実行することができます。分散ファイルのすべての部分に自動的に実行される CL コマンドには、このことを行う必要はありません。

ファイル記述の表示 (DSPFD) コマンドを使用すると、分散ファイルのノード・グループ情報を表示することができます。DSPFD コマンドは、ノード・グループの名前、区分化キーのフィールド、およびノード・グループの完全記述を示します。この情報を表示するには、*ALL または *NODGRP を DSPFD コマンドの TYPE パラメーターに指定する必要があります。

物理ファイル・メンバーの表示 (DSPPFM) コマンドを使用すると、分散ファイルのローカル・レコードを表示することができます。ただし、ローカル・データだけでなくリモート・データを表示したい場合は、コマンドの開始レコード (FROMRCD) パラメーターに *ALLDATA を指定する必要があります。

オブジェクトの保管 (SAVOBJ) コマンド、またはオブジェクトの復元 (RSTOBJ) コマンドを分散ファイルに使用する場合は、分散ファイルの各部分を別々に保管、復元する必要があります。ファイルの部分は、分散ファイルの一部として維持される場合、保管されたときのシステムにのみ復元することができます。必要であれば、オブジェクトの割り振り (ALLOBJ) コマンドを使用して、ファイルのすべての部分のロックを獲得して、保管処理中にファイルに更新が加えられるのを防止することができます。

システムは、次の条件があてはまる場合、ファイルの復元時に、すべての論理ファイルを自動的に配布します。

- 論理ファイルが、非分散ファイルとして保管された。
- 論理ファイルが、基礎ファイルの配布時にシステムに復元される。

ファイルの保管部分を使用して、ローカル・ファイルを作成することもできます。これには、ファイルの部分を別のライブラリー、または分散ファイルの作成時に使用されたノード・グループになかったシステムのいずれかに、復元する必要があります。分散ファイル内のすべてのレコードをローカル・ファイルに収容するには、ファイルの各部分を同じシステムに復元してから、1 つの集合ファイルにレコードをコピーする必要があります。集合ファイルにレコードをコピーするには、ファイルのコピー (CPYF) コマンドを使用してください。

関連概念

14 ページの『分散ファイルを作成した後のシステムの活動』
ファイルが作成されるとすぐに、システムはそのデータを確実に区分化し、ファイルを並行レベルで維持します。

19 ページの『DB2 マルチシステムでのジャーナル処理の考慮事項』
物理ファイル・ジャーナルの開始 (STRJRNPF) コマンドと物理ファイル・ジャーナルの終了 (ENDJRNPF) コマンドは、他のシステムに配布されますが、実際のジャーナル処理は、各システムで独立して、各システムの独自のジャーナル・レシーバーに対して起こります。

『CL コマンド: DB2 マルチシステムでの分散ファイルのすべての部分に影響を与えるもの』
一部の CL コマンドを実行すると、分散ファイルのすべての部分に影響が及びます。

CL コマンド: DB2 マルチシステムでの分散ファイルのすべての部分に影響を与えるもの:

一部の CL コマンドを実行すると、分散ファイルのすべての部分に影響が及びます。

これらのコマンドをシステム上で実行すると、ノード・グループ内のすべてのノードで、コマンドが自動的に実行されます。

この規則によって、各システムに同じコマンドを入力しなくても、ノード・グループ間の整合性を維持することができます。権限を変更すると、ノード・グループ間でいくつかの矛盾が生じることがあります。たとえば、ユーザー ID がノード・グループ内のあるシステムから削除されると、ノード・グループ間で整合性を維持することができなくなります。

権限エラーは別々に処理されます。

次のコマンドは、分散ファイルのすべての部分に影響を与えます。

- 論理ファイル・メンバーの追加 (ADDLFM)
- 物理ファイル制約の追加 (ADDPFCST)
- 物理ファイル・メンバーの追加 (ADDPFM)
- 物理ファイル・トリガーの追加 (ADDPFTRG)
- オブジェクトの割り振り (ALCOBJ)
- 論理ファイルの変更 (CHGLF)
- オブジェクト所有者の変更 (CHGOBJOWN)
- 物理ファイルの変更 (CHGPF)
- 物理ファイル制約の変更 (CHGPFCST)
- 物理ファイル・メンバーのクリア (CLRPFM)
- ファイルのコピー (CPYF)。このコマンドの追加情報については、『DB2 マルチシステムにおける分散ファイルでのファイルのコピー (CPYF) コマンドの使用』を参照してください。
- 論理ファイルの作成 (CRTLF)
- オブジェクトの割り振り解除 (DLCOBJ)
- ファイルの削除 (DLTF)
- 物理ファイル変更ジャーナルの終了 (ENDJRNPF)。このコマンドの追加情報については、『DB2 マルチシステムでのジャーナル処理の考慮事項』を参照してください。
- オブジェクト権限の付与 (GRTOBJAUT)
- 物理ファイル制約の除去 (RMVPFCST)

- 物理ファイル・トリガーの除去 (RMVPFTRG)
- オブジェクトの名前変更 (RNMOBJ)
- 物理ファイル・メンバーの再編成 (RGZPFM)
- オブジェクト権限の取り消し (RVKOBJAUT)
- 物理ファイル・ジャーナルの開始 (STRJRNPF)。このコマンドの追加情報については、『DB2 マルチシステムでのジャーナル処理の考慮事項』を参照してください。

これらのコマンドでは、分散ファイル以外のオブジェクトが参照される場合は、ユーザーが各システムにそれらのオブジェクトを作成する必要があります。たとえば、物理ファイル・トリガーの追加 (ADDPFTRG) コマンドを使用する場合は、トリガー・プログラムがすべてのシステム上にあることを確認する必要があります。そうしないと、エラーが起こります。これと同じ概念は、物理ファイル・ジャーナルの開始 (STRJRNPF) コマンドに適用されます。ここで、ジャーナルは、すべてのシステムに存在しなければなりません。

ユーザー・プロファイルがリモート・ノードに存在しない場合に、GRTOBJAUT コマンドまたは RVKOBJAUT コマンドを出すと、プロファイルが存在するすべてのノードで権限が付与または取り消されます。プロファイルが存在しないノードでは無視されます。

関連概念

14 ページの『分散ファイルを作成した後のシステムの活動』
ファイルが作成されるとすぐに、システムはそのデータを確実に区分化し、ファイルを並行レベルで維持します。

16 ページの『CL コマンド: DB2 マルチシステムでの分散ファイルのローカル部分にのみ影響を与えるもの』
一部の CL コマンドを実行すると、ローカル・システム上にある分散ファイルの部分にのみ影響が及びます (ローカル・システムとは、コマンドが実行されるシステムです)。

DB2 マルチシステムでのジャーナル処理の考慮事項:

物理ファイル・ジャーナルの開始 (STRJRNPF) コマンドと物理ファイル・ジャーナルの終了 (ENDJRNPF) コマンドは、他のシステムに配布されますが、実際のジャーナル処理は、各システムで独立して、各システムの独自のジャーナル・レシーバーに対して起こります。

例としては、分散ファイルのある 2 つのシステム (A および B) が挙げられます。システム A とシステム B の両方にジャーナルとレシーバーを作成する必要があります。また、ジャーナル名とライブラリーは両方のシステムで同じでなければなりません。STRJRNPF コマンドを出すと、コマンドは両方のシステムに配布されて、ジャーナル処理は両方のシステムで開始されます。ただし、システム A のジャーナル・レシーバーには、システム A に常駐するファイルの部分に対して起こる変更を示すデータだけが含まれます。システム B のジャーナル・レシーバーには、システム B に常駐するファイルの部分に対して起こる変更を示すデータだけが含まれます。

これにより、バックアップ方法だけでなく、保管および復元の方法に対しても次のような影響が及びます。

- STRJRNPF コマンドを出したら、ファイルのノード・グループ内にある各システムから、データベース・ファイルを保管する必要があります。
- 各システムに対して、標準ジャーナル管理を行う必要があります。ジャーナル・レシーバーを正しく変更、保管して、各システムのディスク・スペース使用を管理できるようにする必要があります。あるいは、システム変更ジャーナル管理機能サポートを利用することができます。

注: ジャーナルの名前は各システムで同じでなければなりません。属性は同じでなくても構いません。したがって、たとえば、システムによって異なるジャーナル・レシーバーしきい値を指定して、各システムで使用可能なディスク・スペースを反映させることができます。

- 分散データベース・ファイルの部分を回復する必要がある場合は、分散ファイルの部分が存在するシステムから、ジャーナル・レシーバーを使用するだけですみます。このジャーナル・レシーバーから、ジャーナル処理された変更を、ジャーナル処理済み変更適用 (APYJRNCHG) コマンドを使用して適用したり、ジャーナル処理済み変更の除去 (RMVJRNCHG) コマンドを使用して除去したりします。
- 1 つのシステムからジャーナル・レシーバーを使用して、他のシステムのファイル部分にジャーナル変更を適用または除去することはできません。これは、各システム上にあるファイルの各部分が、独自の固有ジャーナル識別子 (JID) を持つためです。

関連概念

14 ページの『分散ファイルを作成した後のシステムの活動』
ファイルが作成されるとすぐに、システムはそのデータを確実に区分化し、ファイルを並行レベルで維持します。

16 ページの『CL コマンド: DB2 マルチシステムでの分散ファイルのローカル部分にのみ影響を与えるもの』
一部の CL コマンドを実行すると、ローカル・システム上にある分散ファイルの部分にのみ影響が及びます (ローカル・システムとは、コマンドが実行されるシステムです)。

DB2 マルチシステムにおける分散ファイルでのファイルのコピー・コマンド:

ファイルのコピー (CPYF) コマンドを出すと、システムはできるだけ早く CPYF コマンドを出そうとします。

指定されたコマンド・パラメーター、コピーに関与するファイル属性、およびコピーされるレコードのサイズと数はすべて、コマンドの実行速度に影響します。

分散ファイルのデータをコピーする場合、コピー・コマンドのパフォーマンスは、次のパラメーターだけを CPYF コマンドに使用することで改善することができます。すなわち、FROMFILE、 TOFILE、 FROMMBR、 TOMBR、 MBROPT、および FMTOPT(*NONE) または FMTOPT(*NOCHK) です。また、取り出しファイル (FROMFILE) と受け入れファイル (TOFILE) の各パラメーターは、ヌル可能フィールドを含むファイルを指定することはできません。一般に、コピー・コマンドの構文が簡単になるほど、最高速コピー処理が行われる確率は高くなります。分散ファイルへのコピー中に最高速コピー方式が使用されると、メッセージ CPC9203 が出されて、各ノードにコピーされるレコード数を示します。通常、このメッセージが出されないと、最高速コピーは実行されません。

分散ファイルにコピーする際には、最高速コピーが使用されるかどうかによって、以下のような違いがあることに注意してください。

- 最高速コピーの場合、レコードは各ノードごとにバッファーに入れられます。バッファーがいっぱいになると、特定のノードに送られます。レコードがいずれかのノード・バッファーに入れられた後でエラーが起こると、システムは、現在ノード・バッファー内にあるレコードすべてを正しいノードに送ろうとします。システムが特定のノードにレコードを送っているときにエラーが起こると、処理は次のノードに続けられ、システムがノード・バッファーすべてを送ろうとします。

  すなわち、エラーのある特定レコードに続くレコードは、分散ファイルに書き込まれるはずです。このアクションは、複数のノードで同時ブロック化が行われるために起こります。エラーのあるレコードに続くレコードを、分散ファイルに書き込みたくない場合は、CPYF コマンドに 1 以上の値を持つ ERRLVL(*NOMAX) または ERRLVL のいずれかを指定することによって、最高速コピーを使用させないようにすることができます。

最高速コピーを使用しないと、配布先ファイルのオープンが SEQONLY(*NO) オープンでない場合や、SEQONLY(*NO) オープンになるように設定されていない場合には、レコードのブロック化が行われます。

- 最高速コピーを使用すると、メンバーのオープンが SEQONLY (*NO) に変更されたことを示すメッセージが出されます。ただし、配布先ファイルは、2 回目には、レコードのブロック化を可能にするためにオープンされます。SEQONLY(*NO) への変更に関するメッセージは無視してください。
- 最高速コピーを使用すると、各ノードにコピーされるレコードの数を示す複数のメッセージが出されます。次に、コピーされるレコードの総数を示すメッセージが送られます。
- 最高速コピーを使用しないと、レコードがコピーされたことを示すメッセージの総数だけが送られます。各ノードにコピーされたレコードの数をリストするメッセージは送信されません。

以下の分散ファイルとのコピーの制限について考慮してください。

- FROMRCD パラメーターは、分散ファイルからコピーする場合、*START または 1 という値だけを使用して指定することができます。TORCD パラメーターは、分散ファイルからコピーするときに、デフォルト値 *END 以外の値を使用して指定することはできません。
- MBROPT(*UPDADD) パラメーターは、分散ファイルへのコピー時には指定できません。
- COMPRESS(*NO) パラメーターは、コピー先ファイルが分散ファイルであり、コピー元ファイルがデータベース削除可能ファイルである場合、指定できません。
- コピー印刷リストの場合、指定された RCDNBR 値は、レコードが分散ファイル・レコードの場合、特定ノードのファイル内のレコードの位置を示します。同じレコード番号がリスト上に複数回表示されます。各番号は、異なるノードからのレコードを示します。

DB2 マルチシステムでの区分化

区分化 とは、ノード・グループ内でノード間にファイルを配布するプロセスのことをいいます。

区分化は、ハッシュ・アルゴリズムを使用して行われます。新しいレコードが追加されると、ハッシュ・アルゴリズムが区分化キー内のデータに適用されます。ハッシュ・アルゴリズムの結果である 0 から 1023 の間の数字は、区分化マップに適用されて、レコードが存在するノードを判別します。

区分化マップは、QUERY 最適化、更新、削除、および結合にも使用されます。区分化マップをカスタマイズして、特定のキー値を特定のノードに使用することができます。

たとえば、入出力中、システムはハッシュ・アルゴリズムを区分化キー・フィールド内の値に適用します。この結果は、ファイル内に保存された区分化マップに適用されて、どのノードがレコードを保存したかを判別します。

次の例は、これらの概念が相互にどのように関連しているかを示しています。

社員番号は区分化キーであり、レコードは社員番号 56,000 のデータベースに入力されます。56,000 という値はハッシュ・アルゴリズムによって処理され、結果は 733 という区画番号になります。区画マップはノード・グループ・オブジェクトの一部であり、作成時に分散ファイルに格納されます。これには、区画番号 733 を示すノード番号 1 が含まれます。したがって、このコードは、ノード番号 1 を割り当てられたノード・グループ内のシステムに物理的に格納されます。区分化キー (PTNKEY パラメーター) は、区分化 (分散) ファイルを作成したときに、ユーザーによって指定されています。

区分化キー内のフィールドは、ヌル可能にすることができます。ただし、区分化キー内にヌル値を含むレコードは、常に、区画番号 0 にハッシュします。区分化キー内に有効な数のヌル値を持つファイルは、区画番号 0 でデータ・スキューになる可能性があります。これは、ヌル値を持つすべてのレコードが区画番号 0 にハッシュするためです。

ノード・グループ・オブジェクトと区分化分散リレーショナル・データベース・ファイルを作成した後、DSPNODGRP コマンドを使用して、区画番号とノード名の間の関係を表示することができます。『DB2 マルチシステムでの DSPNODGRP コマンドを使用したノード・グループの表示』のトピックには、区画番号、ノード・グループ、およびシステム名を表示する方法に関する詳細情報があります。

分散ファイルを作成する場合は、区分化キー・フィールドは物理ファイルの作成 (CRTPF) コマンドの PTNKEY パラメーターか、または SQL CREATE TABLE ステートメントの PARTITIONING KEY 文節に指定されます。データ・タイプ DATE、TIME、TIMESTAMP、および FLOAT を持つフィールドは、区分化キーでは使用できません。

関連概念

5 ページの『DB2 マルチシステムでのノード・グループの仕組み』
ノード・グループ とは、作成されたシステム上に格納されたシステム・オブジェクト (*NODGRP) のことをいいます。

8 ページの『ノード・グループの表示コマンド』
ノード・グループの表示 (DSPNODGRP) コマンドは、ノード・グループ内のノード (システム) を表示します。

10 ページの『ノード・グループ属性の変更コマンド』
ノード・グループ属性の変更 (CHGNODGRPA) コマンドは、ノード・グループのデータ区分化属性を変更します。

DB2 マルチシステムでの区分化の計画

ほとんどの場合、区分化および区分化キーをどのように使用したいかに関する計画を立てておく必要があります。

他のシステムに配置するために、データをどのように体系的に分割するか、QUERY に頻繁に結合させたいデータは何かを決める必要があります。また、選択を行う場合に有効な選択は何か、区分化キーを設定して必要なデータを獲得するための最も効率的な方法は何かを決める必要もあります。

区分化を計画する場合は、最高速システムがほとんどのデータを受け取るようにそれを設定することができます。どのシステムで対称マルチプロセッシング (SMP) 並行処理機能を利用してデータベースのパフォーマンスを改善するかを考える必要があります。QUERY 最適化プログラムがその分散アクセス計画を作成する場合、最適化プログラムは、要求ノード上のレコード数をカウントして、その数にノードの総数を乗算することに注意してください。ほとんどのレコードを SMP システムに入れることには利点がありますが、最適化プログラムは、これらの利点のいくつかを相殺する可能性があります。これは、最適化プログラムが、各ノードで等しい数のレコードをその計算に使用するためです。

区分化に影響を与えたい場合は、そのようにすることができます。たとえば、会社で、特定のシステムを使用して作業を完了する地域販売部門があるとします。区分化を使用すると、ある地域のローカル・データを、それぞれの地域の該当するシステムに格納することができます。したがって、米国北西部の社員が使用するシステムには、北西部のデータを格納します。

区分化を設定するには、CRTPF コマンドの PTNFILE および PTNMBR の各パラメーターを使用することができます。ノード・グループ属性の変更 (CHGNODGRPA) コマンドを使用すると、すでに区分化されたファイルを再分配することができます。

パフォーマンスの向上は、大きなファイル間で行われる QUERY で最高になります。トランザクション処理での使用頻度が高く、QUERY ではほとんど使用されないファイルは、区分化には適していないため、ローカル・ファイルとして残す必要があります。

結合処理では、特定のフィールドで 2 つのファイルを結合することが多い場合、そのフィールドを両方のファイルの区分化キーにする必要があります。また、フィールドが同じデータ・タイプであることも確認する必要があります。

関連概念

SQL プログラミング

データベース・プログラミング

24 ページの『DB2 マルチシステムでのデータ分散のカスタマイズ』

データの配置についてはシステムが担当するため、ユーザーはレコードの実際の所在を知る必要はありません。ただし、特定のレコードが特定のシステムに常に格納されるようにしたい場合は、ノード・グループ属性の変更 (CHGNODGRPA) コマンドを使用して、これらのレコードの所在を指定することができます。

DB2 マルチシステムでの区分化キーの選択

システムが最も効率的な方法で区分化ファイルを処理するために、区分化キーを設定または使用する際に検討できるヒントがいくつかあります。

以下のようなヒントがあります。

- 最も適切な区分化キーとは、多数の異なる値を持つキーであるため、区分化活動ではデータ・レコードの均等な配布が起こります。顧客番号、名字、クレーム番号、郵便番号、および電話の市外局番などが、区分化キーとして使用するのに適したカテゴリーの例としてあげられます。

  性別は、男と女の 2 つの選択しかないため、区分化キーには適していない例といえます。性別では、大量のデータが、複数のノードに配布されるのではなく、単一のノードに配布されてしまいます。また、QUERY を実行する場合は、区分化キーとして性別を使用すると、システムは多数のデータ・レコードを処理することになります。これは非効率的です。別のフィールドまたはデータ・フィールドを使用すると、QUERY の有効範囲を狭めて、処理をはるかに効率的にすることができます。性別に基づく区分化キーは、特定の値に基づく分配ではなく、データの均等な分配が必要な場合には不適切です。

  ローカル・ファイルを分散ファイルに変更する用意をする場合は、HASH 関数を使用して、データをどのように分配できるかを考えることができます。HASH 関数は、ローカル・ファイルの各種の列に対して使用できるため、実際にファイルを分散ファイルに変更する前に、異なる区分化キーを試すことができます。たとえば、ファイルの郵便番号フィールドを使用する場合は、そのフィールドを使用して HASH 関数を実行して、各区画番号にハッシュするレコード数を考えることができます。このことは、区分化キー・フィールドを選択するうえで、またはノード・グループで区画マップを作成するうえで役立ちます。
- 頻繁に更新する必要があるフィールドは選択しないでください。区分化キー・フィールドの制約の 1 つに、更新によってレコードが異なるノードに強制されない場合にのみ、その値を更新できるというものがあります。
- 区分化キーには多数のフィールドを使用しないでください。1 つのフィールドを使用する方法が最も適しています。多数のフィールドを使用すると、システムは入出力時にさらに多くの作業を実行することになります。
- 固定長文字または整数などの簡単なデータ・タイプを区分化キーとして選択してください。この考慮事項は、ハッシュが単純なデータ・タイプの単一フィールドに行われるため、パフォーマンスに役立つ場合があります。

- 区分化キーを選択する場合は、通常実行する QUERY の結合とグループ化の基準を考慮する必要があります。たとえば、結合に関与するファイルの結合フィールドとして使用されることのないフィールドを選択すると、逆に結合のパフォーマンスに影響が及ぶ可能性があります。分散ファイルに関与する QUERY の実行については、『DB2 マルチシステムでのパフォーマンスのための QUERY 設計』を参照してください。

関連概念

44 ページの『DB2 マルチシステムでのパフォーマンスのための QUERY 設計』
以下の指針に基づいて QUERY を設計できます。この方法により、分散ファイル使用の QUERY を実行するときに、より効率的に QUERY リソースを使用できます。

DB2 マルチシステムでのデータ分散のカスタマイズ

データの配置についてはシステムが担当するため、ユーザーはレコードの実際の所在を知る必要はありません。ただし、特定のレコードが特定のシステムに常に格納されるようにしたい場合は、ノード・グループ属性の変更 (CHGNODGRPA) コマンドを使用して、これらのレコードの所在を指定することができます。

たとえば、郵便番号 55902 のレコードすべてを、Minnesota 州 Minneapolis のシステムに常駐させたい場合を想定します。 CHGNODGRPA コマンドを出す場合は、郵便番号 55902 と、Minneapolis のローカル・ノードのシステム・ノード番号を指定する必要があります。

この時点で、郵便番号 55902 はノード・グループを変更していますが、データはまだ以前と同じ場所に配布されます。 CHGNODGRPA コマンドは、既存のファイルに影響を与えません。区分化ファイルが作成されると、その区分化ファイルは、そのときのノード・グループの情報のコピーを保持します。ノード・グループは、区分化ファイルに影響を与えずに変更、または削除することができます。有効にするために再分配されるレコードへの変更では、新しいノード・グループを使用して分散ファイルを再作成するか、または物理ファイルの変更 (CHGPF) コマンドを使用して、新しいあるいは更新されたノード・グループを指定することができます。

CHGPF コマンドを使用すると、次のことができます。

- すでに区分化されたファイルを再分配する。
- 区分化キーを変更する (たとえば、電話の市外局番からブランチ ID へ)。
- ローカル・ファイルを分散ファイルに変更する。
- 分散ファイルをローカル・ファイルに変更する。

注: CHGNODGRPA コマンドを使用して、すでに区分化されているファイルを再分配する必要があります。CHGNODGRPA コマンドを CHGPF コマンドとともに任意指定で使用すると、他のいずれかのタスクを実行することができます。

『ネットワークにシステムを追加する際の再分配の問題』トピックには、ローカル・ファイルを分散ファイルに変更する方法、または分散ファイルをローカル・ファイルに変更する方法に関する情報があります。

関連概念

10 ページの『ノード・グループ属性の変更コマンド』
ノード・グループ属性の変更 (CHGNODGRPA) コマンドは、ノード・グループのデータ区分化属性を変更します。

22 ページの『DB2 マルチシステムでの区分化の計画』
ほとんどの場合、区分化および区分化キーをどのように使用したいかに関する計画を立てておく必要があります。

43 ページの『ネットワークにシステムを追加する際の再分配の問題』
ノード・グループ間でのファイルの再分配は、非常に簡単な処理です。

区分化表

DB2 for i は、SQL を使用して区分化表をサポートします。

区分化表によって、データを複数のメンバーに保管できます。このとき、QUERY、挿入、更新、削除などのデータ操作の際には表が 1 つのオブジェクトのように見えます。それぞれの区画には、基になった表の設計特性 (列名とタイプ、制約、トリガーなど) が継承されます。

区分化によって、より多くのデータを表に保管できます。区分化しない場合、1 つの表に最大で 4,294,967,288 行、または最大サイズ 1.7 TB (1 TB は 1,099,511,627,776 バイト) を保管できます。しかし、区分化表の場合、多数の区画を含むことができ、それぞれの区画に表の最大サイズを適用できます。区分化表の最大サイズについて詳しくは、「DB2 for i ホワイト・ペーパー」を参照してください。

また、区分化によって、データベースのパフォーマンス、回復の容易性、および管理の容易性が改善される可能性があります。それぞれの区画を、他の区画とは独立して保管、復元、エクスポート、インポート、除去、または再編成することができます。さらに、区分化によって、区画としてグループ化されたレコードのセットをすばやく削除できます。区分化されない表のように、個々の行を処理する必要がありません。区画の除去操作は、区分化されない表の同等の行を削除する場合に比べてパフォーマンスが非常に優れています。

区分化表は、複数のメンバーからなるデータベース・ファイルです。それぞれの区分化表は、データベース・ファイルの各メンバーに対応します。したがって、メンバーに対して使用できるほとんどの CL コマンドは、区分化表の各区画に対しても有効です。

区分化表のサポートを活用するには、ご使用のシステムに DB2 マルチシステムをインストールしなければなりません。ただし、DB2 マルチシステムと区分化との間には、重要な違いがいくつかあります。DB2 マルチシステムの場合、以下の 2 つの方法でデータを区分化します。

- 分散表を作成して、複数のシステムまたは論理区画にデータを分散することができます。
- 区分化表を作成して、 1 つのシステムの同じデータベース表内の複数のメンバーにデータを区画化することができます。

どちらの場合も、区分化されていない表と同じようにアクセスできます。

関連概念

3 ページの『DB2 マルチシステム: 基本的な用語と概念』
分散ファイル とは、複数の System i モデルに分散されたデータベース・ファイルのことをいいます。以下に、DB2 マルチシステムによる分散ファイルの作成と使用に関する主な概念の一部を示します。

関連情報

DB2 for i5/OS ホワイト・ペーパー

区分化表の作成

CREATE TABLE ステートメントを使用して、新しい区分化表を作成できます。

表定義には、表名と、表の列の名前および属性を含める必要があります。さらに、表の他の属性 (たとえば 1 次キー) を定義に含めることもできます。

区分化には、ハッシュ区分化と範囲区分化の 2 つの方法があります。ハッシュ区分化は、ユーザーが指定した数の区画およびキー列の中にランダムな間隔で行を格納します。範囲区分化は、ユーザーが指定した列値の範囲に基づいて表を分割します。どちらのタイプの区分化を使用するかは、 PARTITION BY 文節を使って指定します。たとえば、ライブラリー PRODLIB 内の表 PAYROLL を 区分化キー EMPNUM を使って 4 つの区画に分ける場合、以下のコードを使用します。

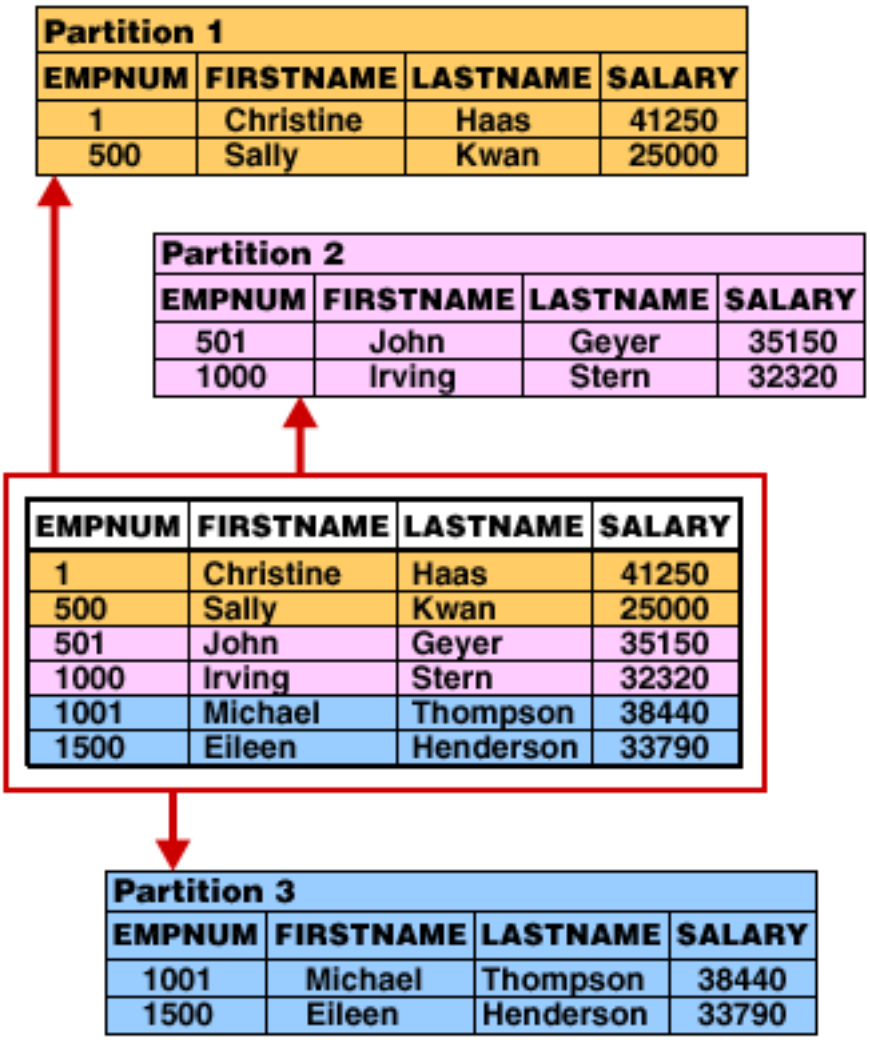
```
CREATE TABLE PRODLIB.PAYROLL
 (EMPNUM INT,
  FIRSTNAME CHAR(15),
    LASTNAME CHAR(15),
    SALARY INT)
PARTITION BY HASH(EMPNUM)
INTO 4 PARTITIONS
```

または、範囲で区切って PAYROLL を区画化するには、以下のコードを使用します。

```
CREATE TABLE PRODLIB.PAYROLL
 (EMPNUM INT,
  FIRSTNAME CHAR(15),
    LASTNAME CHAR(15),
    SALARY INT)
PARTITION BY RANGE(EMPNUM)
(STARTING FROM (MINVALUE) ENDING AT (500) INCLUSIVE,
STARTING FROM (501) ENDING AT (1000) INCLUSIVE,
STARTING FROM (1001) ENDING AT (MAXVALUE)
```

このステートメントによって、3 つの区画からなる表が生成されます。最初の区画には、EMPNUM が 500 以下のすべての行が含まれます。 2 番目の区画には、EMPNUM が 501 以上 1000 以下のすべての行が含まれます。 3 番目の区画には、EMPNUM が 1001 以上のすべての行が含まれます。以下の図は、これらの値に基づいてデータが区分化された表を示しています。

図 6. 区分化された従業員情報



区分化表が作成されるとき、システム提供の検査制約がそれぞれの区画に追加されます。ユーザーはこの検査制約を表示、変更、または削除することができません。

範囲区分化の場合、この検査制約は、データが適切な範囲にあることを検証します。または、ヌル値を許容する区画の場合、検査制約はデータがヌルかどうかを検証します。

ハッシュ分割化の場合、この検査制約は、 区画番号 = MOD(Hash(fields), 区画の数) + 1 という条件に基づいてデータを検証します (このハッシュ関数は、0 から 1023 までの値を戻します)。ヌル値は、常に最初の区画に格納されます。

区分化の文節および構文図については、「SQL 解説書」トピック・コレクションの『CREATE TABLE ステートメント』を参照してください。

関連概念

『非区分化表から区分化表へ』
区分化されていない表を区分化表に変更するには、 ALTER TABLE ステートメントの ADD *区分化文節* を使用します。既存の表を変更して区分化することは、新しい区分化表を作成することと同様です。

関連タスク

CREATE TABLE

関連資料

SQL 解説書

33 ページの『検査制約の最適化』
検査する区画の数を減らすために、最適化プログラムは検査制約を使用します (検査制約はユーザーによって追加されるか、区分化キーに対する暗黙的な検査制約として追加されます)。

既存の表の変更

既存の区分化されていない表を区分化表に変更したり、既存の区分化表の属性を変更することができます。また、区分化表を区分化されていない表に変更することもできます。

区分化表、および区分化されていない表を変更するには、 ALTER TABLE ステートメントを使用します。

ALTER TABLE ステートメントの詳細、および文節については、「SQL 解説書」トピック・コレクションで説明されています。

関連資料

ALTER TABLE

SQL 解説書

非区分化表から区分化表へ

区分化されていない表を区分化表に変更するには、 ALTER TABLE ステートメントの ADD *区分化文節* を使用します。既存の表を変更して区分化することは、新しい区分化表を作成することと同様です。

たとえば、範囲による区分化を非区分化表 PAYROLL に追加するには、次のコードを使用します。

```
ALTER TABLE PRODLIB.PAYROLL
ADD PARTITION BY RANGE(EMPNUM)
(STARTING(MINVALUE) ENDING(500) INCLUSIVE,
STARTING(501) ENDING(1000) INCLUSIVE,
STARTING(1001) ENDING MAXVALUE))
```

関連概念

26 ページの『区分化表の作成』
CREATE TABLE ステートメントを使用して、新しい区分化表を作成できます。

既存の区分化表の変更

ALTER TABLE ステートメントのいくつかの文節を使用して、既存の区分化表を変更することができます。

以下のような文節があります。

- ADD PARTITION

  この文節は 1 つまたは複数のハッシュ区画または範囲区画を既存の区分化表に追加します。以下の規則に従ってパラメーターを指定する必要があります。そうしないと、エラーが発生します。

  – 非区分化表に対して ADD PARTITION 文節を使用しないでください。

– ハッシュ区画を追加する場合、追加される区画の数を正の整数で指定する必要があります。
– 区画を識別する名前または整数を明示的に指定する場合、他の区画がそれを使用していないことを確認してください。
– 範囲区画を追加する場合、指定される範囲は、既存のいずれかの区画の範囲と重なり合ってはなりません。

たとえば、区分化キー EMPNUM を使用するライブラリー PRODLIB 内の表 PAYROLL に 4 つの区画を追加するには、以下のコードを使用します。

```
ALTER TABLE PRODLIB.PAYROLL
ADD PARTITION 4 HASH PARTITIONS
```

- ALTER PARTITION

この文節は指定された範囲区画の範囲を変更します。以下の条件を必ず満たす必要があります。
– 指定された区画が表に存在しなければなりません。
– 指定された範囲は、既存のいずれかの区画の範囲と重なり合ってはなりません。
– 区分化表の既存のすべての行は、 ALTER TABLE ステートメントによって指定される新しい範囲内に収まる必要があります。

- DROP PARTITION

この文節は区分化表から区画を除去します。指定された表が区分化表でない場合は、エラーが戻されます。区分化表の中で最後に残った区画が指定された場合、エラーが戻されます。

列のデータ・タイプの変更時の制限:

列のデータ・タイプを変更する場合、その列が区分化キーに含まれるならば、ターゲット・データ・タイプにはいくつかの制限が適用されます。

範囲による区分化の場合、表の区分化に使用される列のデータ・タイプを BLOB、CLOB、DBCLOB、DATALINK、浮動小数点タイプ、またはこれらのタイプに基づく特殊タイプに変更することはできません。ハッシュ区分化の場合、区分化キーに含まれる列のデータ・タイプを LOB、DATE、TIME、TIMESTAMP、浮動小数点タイプ、またはこれらのいずれかに基づく特殊タイプに変更することはできません。

関連資料

ALTER TABLE

区分化表から非区分化表へ

DROP PARTITIONING 文節は、区分化表を非区分化表に変更します。

指定された表がすでに非区分化表である場合は、エラーが戻されます。データを含んでいる区分化表を非区分化表に変更するとき、データ区画の間でデータを移動する必要があります。非区分化表の最大サイズを超える大きさの区分化表を非区分化表に変更することはできません。

区分化表での索引

索引は、区分化または非区分化として作成することができます。区分化された索引は、それぞれの区画ごとに個別の索引を作成します。区分化されない索引は、表内のすべての区画にまたがる単一の索引です。

区分化された索引を使用すると、QUERY を最適化できる可能性があります。固有索引を区分化する場合、索引内で指定される列は、データ区分化キーと同じ、またはスーパーセットでなければなりません。

区分化表に索引を作成するには、CREATE INDEX ステートメントを使用します。それぞれの区画に対する索引を作成するには、 PARTITIONED 文節を使用します。

```
CREATE INDEX PRODLIB.SAMPLEINDEX
ON PRODLIB.PAYROLL(EMPNUM) PARTITIONED
```

すべての区画にまたがる単一の索引を作成するには、 NOT PARTITIONED 文節を使用します。

```
CREATE INDEX PRODLIB.SAMPLEINDEX
ON PRODLIB.PAYROLL(EMPNUM) NOT PARTITIONED
```

区分化表に対しては、区分化されたエンコード・ベクトル索引 (EVI) だけを作成できます。区分化表に対して、非区分化 EVI を作成することはできません。

「SQL 解説書」トピック・コレクションの 『CREATE INDEX ステートメント』には、区分化表に対する索引の作成に関する詳細情報があります。

区分化表に対して SQL の固有索引、固有制約、または基本キー制約を作成するときには、以下の制限が適用されます。

- 固有索引のキーが、区分化されるキーと同じか、区分化されるキーのスーパーセットである場合に限り、索引を区分化することができます。
- デフォルト値が NOT PARTITIONED で固有索引を作成する場合、固有索引のキーが区分化されるキーのスーパーセットであれば、固有索引は区分化として作成されます。しかし、ユーザーが NOT PARTITIONED と明確に指定する場合、固有索引のキーが区分化されるキーのスーパーセットであれば、固有索引は区分化**されずに**作成されます。
- 既存の固有索引を制約に対して共有しようとすると、区分化索引の共有を試みる前に、既存の固有非区分化索引のチェックが行われます。参照制約の親キーとして使用される、一意的にキー付けされた論理ファイルを作成することをユーザーが望む場合に、このチェックは役立ちます。一意的にキー付けされた論理ファイルを作成してから、何らかの主キーまたは固有制約を追加することをユーザーが望む場合には、キー付き論理ファイル索引が、制約に対する非区分化索引として使用されることになります。区分化表の非区分化主キーを参照制約の親キーにしたい場合には、これは望ましい方法でしょう。以下に例を示します。

```
CREATE TABLE PRODLIB.PAYROLL (C1 INT)
   PARTITION BY HASH(C1) INTO 3 PARTITIONS
CREATE UNIQUE INDEX PRODLIB.PINDEX ON PRODLIB.PAYROLL(C1)
         NOT PARTITIONED
 ADDPFCST FILE(PRODLIB/PAYROLL) TYPE(*PRIKEY) KEY(C1)
```

関連概念

『QUERY のパフォーマンスおよび最適化』
区分化表は非常に大きくなることが多いため、 QUERY で区分化表を参照する場合には注意が必要です。複数の区画へのアクセスが、システムやアプリケーションにどのような影響を与えるかを理解しておくことが重要です。

関連タスク

CREATE INDEX

関連資料

SQL 解説書

QUERY のパフォーマンスおよび最適化

区分化表は非常に大きくなることが多いため、 QUERY で区分化表を参照する場合には注意が必要です。複数の区画へのアクセスが、システムやアプリケーションにどのような影響を与えるかを理解しておくことが重要です。

区分化表は、DB2 for i で SQL を使用する場合の最適化および並列処理をすべて利用できます。データベース・パフォーマンスと最適化のトピックには、QUERY の最適化に関する一般情報が記載されています。区分化表の場合、『データベース・パフォーマンスと最適化』のトピックで説明されているすべてのデータ・アクセス方式を使用して、それぞれの区画のデータにアクセスできます。さらに、DB2 対称マルチプロセッシング機能をインストール済みの場合、QUERY を実装する際に、最適化プログラムは並列データ・アクセス方式を使用できます。

区分化表のただ 1 つの区画だけにアクセスする QUERY の場合、その区画の別名を作成して、QUERY の中でその別名を使用すれば、パフォーマンスが改善される可能性があります。 QUERY は、非区分化表への QUERY として動作します。

概念上、区分化表はネストされた表として実装され、それぞれの区画は他の区画と合併 (UNION) されると考えることができます。

たとえば、以下の QUERY を実行した場合、

```
SELECT LASTNAME, SALARY FROM PRODLIB.PAYROLL
 WHERE SALARY > 20000
```

QUERY の実装を以下のように記述することができます。

```
SELECT LASTNAME, SALARY FROM
  (SELECT LASTNAME, SALARY FROM  PRODLB.PAYROLL(PART00001)
   UNION ALL
    SELECT LASTNAME, SALARY FROM  PRODLB.PAYROLL(PART00002)
  UNION ALL
  SELECT LASTNAME, SALARY FROM  PRODLB.PAYROLL(PART00003))
 X (LASTNAME, SALARY)
 WHERE X.SALARY > 20000
```

区分化表に対する QUERY の実装は、 Classic Query Engine (CQE) と SQL Query Engine (SQE) のどちらの QUERY エンジンが使われるかに依存します。 QUERY エンジンについての詳細は、「データベース・パフォーマンスと最適化」のトピック・コレクションの『SQE および CQE エンジン』に記載されています。それぞれのエンジンごとに、考慮事項があります。

関連概念

29 ページの『区分化表での索引』
索引は、区分化または非区分化として作成することができます。区分化された索引は、それぞれの区画ごとに個別の索引を作成します。区分化されない索引は、表内のすべての区画にまたがる単一の索引です。

SQE および CQE エンジン

関連情報

パフォーマンスおよび Query 最適化

SQL Query Engine を使用した QUERY

SQL Query Engine (SQE) は、動的区画拡張最適化プログラムを使用して区分化表のターゲット最適化を可能にします。

ターゲット最適化方式は、与えられた QUERY の構造が、区分化表内の特定の区画において、この特有な最適化を適用できる構造になっているかどうかを最初に判別します。ターゲット最適化が確実に可能であれば、最適化プログラムは個々の最適化を適用できる区画を判別します。次に、これらの区画が個別に最適化されます。残りの区画は、*一回のみ* 最適化を行います。

最適化プログラムは、QUERY または表環境が動的拡張を調整するかどうかを以下の特性に基づいて決定します。

- 表が範囲区分化され、QUERY がその範囲に対して述部選択を処理する。
- 表が、すべての区画ではなく 1 つまたは一部の区画に対する索引を持つ (*サブ区画スパン索引*)。
- 表の区画が比較的少ない。
- 表の制約定義で、特定の区画のみが参加することが指示されている。
- 推定実行時間が特定のしきい値を超える。

拡張が調整されると、最適化プログラムは適宜、既存の統計情報を使用してターゲット区画を判別します。たとえば、範囲区分化および述部選択に対して、最適化プログラムは統計情報または索引を調べて、対象となるメイン区画を判別します。ターゲット区画が確認されると、最適化プログラムは QUERY の書き直しを行います。ターゲット区画は UNION 操作で再定義されます。残りの区画は、*単一の表インスタンス* として存続します。その単一インスタンスは、ターゲット区画と一緒に UNION 操作に追加されます。書き直しが実行されるとすぐに、最適化プログラムは既存の最適化手法を使用して、各 UNION 部分のプランを決定します。最適化の最後に、単一インスタンスはそこに含まれる区画の別の UNION 操作に変換されます。単一インスタンス用に最適化されたプランは UNION サブツリー間で複製されるため、区分化表の最適化時間が大幅に短縮されます。

SQL Query Engine は、区分化表を最適化するために*論理区画除去* も使用します。この方式では、最適化プログラムが潜在的な区画除去機会を識別することができます。最適化プログラムは、ソース表から削減される回答セットを結合述部付きで区分化表の定義キーに適用できる機会を探索します。これらの機会が識別されると、最適化プログラムは QUERY ランタイムにロジックを構築し、ソース表の結果セットに基づいて区画を除去します。

たとえば、最適化プログラムが区分化表の区分化キーを伴う述部 (この例では、where 文節) を識別する QUERY について考えてみます。区画上の制約でキーの範囲が 0 から 10 に制限されており、 QUERY の述部でキー値が 11 であることが識別された場合、その区画は論理的に除去可能です。区画の実装はまだ QUERY に組み込まれていますが、述部に結合される制約の前処理によってパスが分断されることに注意してください。このロジックは、再使用の目的で組み込まれます。キー値が 10 であると識別される述部を持つ別の QUERY も、同様にこのプランを実行します。区画が実装から物理的に除去された場合には、その実装を再利用できなくなります。

これについて、より複雑な例を見てみます。

```
select *
from sales, timeDim
where sales.DateId = timeDim.DateId
and timeDim.Date > '09/01/2004'
```

この例では、sales は範囲区分化表で、 sales.DateId は区分化キーとして識別され、 sales.DateId = timeDim.DateId は削減可能な述部として識別されます。 QUERY は、(大まかに) 次のように変更されます。

```
with sourceTable as (select * from timeDim where  timeDim.Date > '09/01/2004')
select *
from sales, sourceTable
where sales.DateId = sourceTable.DateId
and sales.DateId IN (select DateId from sourceTable)
```

この例では、元の結合は存続しますが、さらにローカル述部 sales.DateId IN (select DateId from sourceTable) が追加されます。これで、この新規述部を区画定義に関する情報と結合して、参加する区画についての回答を作成できます。この出力は、以下の状態によって作成されます。

- 区分化キーへの結合の存在。
- 表からの結合をより少ない結合値のセットに削減できる。これらの結合値を使用して、区分化表内の区画を分断することができます。

検査制約の最適化:

検査する区画の数を減らすために、最適化プログラムは検査制約を使用します (検査制約はユーザーによって追加されるか、区分化キーに対する暗黙的な検査制約として追加されます)。

以下の例では、PAYROLL が範囲によって区分化されているとします。

```
SELECT LASTNAME, SALARY
 FROM PRODLIB.PAYROLL
 WHERE EMPNUM = :hv1
```

最適化プログラムは、表内のそれぞれの区画に新しい述部を以下のように追加します。

```
SELECT LASTNAME, SALARY FROM
 (SELECT LASTNAME, SALARY FROM  PRODLB.PAYROLL(PART00001)
  WHERE EMPNUM=:hv1 AND :hv1 <= 500
  UNION ALL
   SELECT LASTNAME, SALARY FROM  PRODLB.PAYROLL(PART00002)
  WHERE EMPNUM=:hv1 AND :hv1 >= 501 AND :hv1 <=1000
 UNION ALL
 SELECT LASTNAME, SALARY FROM  PRODLB.PAYROLL(PART00003)
  WHERE EMPNUM=:hv1 AND (:hv1 IS NULL OR :hv1 >= 1001))
 X (LASTNAME, SALARY)
```

hv1 の値が 603 の場合、区画 PART00002 の行だけが検査されます。

関連概念

26 ページの『区分化表の作成』
CREATE TABLE ステートメントを使用して、新しい区分化表を作成できます。

SQL Query Engine: 索引の使用法:

SQL Query Engine (SQE) は、区分化と非区分化の両方の索引を使用して QUERY を実装します。

区分化されない索引を使用する場合、最適化プログラムは、それぞれの区画に対して区分化されない索引を実装するよう選択する可能性があります。つまり、たとえば区分化されない索引が固有であっても、それぞれの区画の行が調査される可能性があります。

Classic Query Engine を使用した照会

Classic Query Engine (CQE) を使用して QUERY が実装される場合、実体化や索引の使用法を含め、QUERY がどのように最適化されるかを理解する必要があります。

実体化:

Classic Query Engine (CQE) の実行中に、区分化表はある条件の下で実体化されます。

これらの条件は、以下のとおりです。

- 結合 QUERY の一部である場合。
- GROUP BY 文節が含まれる場合。
- 列に関数が組み込まれている場合。

一時表のサイズを削減して、QUERY にかかる実行時間を短縮するために、一時表の作成時に行の削減に使用できる QUERY 内のすべての選択が処理されて、実体化の一時表のサイズ削減および処理時間の短縮が行われます。

注: 区分化表が実体化されるとき、一時表のサイズは 4,294,967,288 行を超えることはできません。一時表の行数を制限するために、選択処理の後入れ先出しが実行されます。行数が 4,294,967,288 を超えた場合、資源の限界エラーが発生して、QUERY が終了します。

CQE QUERY の最適化に関する考慮事項:

Classic Query Engine (CQE) 最適化プログラムは、区分化表の中の最初の区画メンバーを使用して QUERY を最適化します。このアクセス方式は、すべての区画の行へのアクセスに使用されます。

区分化表へのアクセスに DB2 対称マルチプロセッシング機能のデータ・アクセス方式を使用することはできませんが、一時表の処理、および一時索引の作成にこのアクセス方式を使用できます。

副 QUERY (SUBQUERY) を含む QUERY で区分化表が使用される場合、最適化プログラムは、結合された複合 QUERY として QUERY を実装しようとはしません。このため、結合のために区分化表が実体化されることはありません。

Classic Query Engine: 索引の使用法:

Classic Query Engine (CQE) は、非区分化索引を使用して QUERY を実装します。

区分化された索引が存在する場合、最適化プログラムは、区分化表の処理時にどのアクセス方式を使用するかに関する最適化の決定において、索引を使用しません。

最適化プログラムは区分化表に対する一時索引を作成するので、ライブ・データの順序付けの更新が可能になります。

エンコード・ベクトル索引 (EVI) は単一の区画に対してのみ作成されるため、 CQE は区分化表に対する QUERY の実装に EVI を使用することはできません。

保管と復元に関する考慮事項

他のデータベース・ファイルと同じように、区分化表を保管および復元することができます。

区分化表の区画はデータベース・メンバーであるため、 1 つずつ個別に、または複数を一括して保管および復元できます。区分化表の保管および復元時には、以下の項目を考慮してください。

- 区分化表が以前に存在していたのとは別のシステムに、区分化表の一部の区画だけを復元する場合、システムは復元に含まれなかった区画を作成します。復元から除外され、システムによって作成される区画には、データがまったく復元されません。
- 範囲によって区分化された表を保管した後、 1 つまたは複数の区画を除去した場合には、除去された区画を区分化表に復元することができます。
- ハッシュによって区分化された表を保管した後、区画を除去または追加することによって表を変更した場合には、保管された表をシステム上の表に復元することはできません。ただし、システム上の表が削除された場合には、保管された表を復元できます。

区分化表のすべての区画を処理するには、以下の保管および復元 CL コマンドを使用するアプリケーションを変更して、 `Member *ALL` を使用するようにしなければなりません。

- オブジェクトの復元 (RSTOBJ)
- オブジェクトの保管 (SAVOBJ)
- オブジェクトの保管と復元 (SAVRSTOBJ)

関連タスク

データベースのバックアップおよび回復

区分化表のジャーナル処理

複数メンバーからなる他のデータベース・ファイルと同じようにして、区分化表をジャーナル処理することができます。区分化表をジャーナル処理するとき、表のすべての区画は同じジャーナルによって記録されます。

区分化表のすべての区画を処理するには、以下のジャーナル処理 CL コマンドを使用するアプリケーションを変更して、 `Member *ALL` を使用するようにしなければなりません。

- ジャーナル処理済み変更適用 (APYJRNCHG)
- ジャーナル処理済み変更適用拡張 (APYJRNCHGX)
- ジャーナルの表示 (DSPJRN)
- ジャーナル項目の受信 (RCVJRNE)
- ジャーナル処理済み変更の除去 (RMVJRNCHG)
- ジャーナル項目の検索 (RTVJRNE)

関連概念

ジャーナル管理

従来のシステム・インターフェースに関する考慮事項

SQL 表は、1 つのメンバー (区画) からなるデータベース物理ファイルです。したがって、従来のシステム・アプリケーションがファイルにアクセスするとき、従来のシステム・アプリケーションはファイルのメンバーを開くことによってメンバーに対して読み取りおよび書き込みを実行します。

ファイル (SQL 表) が区分化されると、ファイルは複数メンバーからなるファイルとなり、従来のシステム・アプリケーションはメンバー名 (区画名) を指定しなければなりません。物理ファイルのすべてのメン

バーに基づく SQL 索引を使用するよう従来のシステム・アプリケーションを変更すれば、アプリケーションはデータの読み取りまたは書き込みの際、メンバー名を指定する必要がなくなります。

たとえば、以下のコードを使って SQL 索引を作成した場合、

```
CREATE INDEX LIBNAME.INDEXNAME
   ON LIBNAME.TABLENAME(COLUMNNAME)
   NOT PARTITIONED
```

従来のシステム・アプリケーションは、データがどのように区分化されているかを認識しなくても、区分化表のデータに対して読み取りおよび書き込みを実行できます。

表が区分化された場合 (複数メンバーからなるファイルになった場合)、それまで表に対して実行されていたすべての従来のシステム操作は、複数メンバーからなるファイルの各メンバーに対して実行されなければなりません。たとえば、RGZPFM FILE(LIBNAME/TABLENAME) は *FIRST メンバーだけを再編成するようになります。区分化表に対しては、各メンバーごとに物理ファイル・メンバーの再編成 (RGZPFM) コマンドを使用する必要があります。ファイル記述の表示 (DSPFD) コマンド DSPFD FILE(LIBNAME/TABLENAME) TYPE(*MBRLIST) は、ファイルのすべてのメンバーをリストします。

関連資料

物理ファイル・メンバーの再編成 (RGZPFM) コマンド

ファイル記述の表示 (DSPFD) コマンド

区分化表の制限

区分化表を使用する際には、以下の制限に注意してください。

- 区分化表に対する参照制約は許可されますが、親キー索引は非区分化索引でなければなりません。
- 区分化表に対する基本キー制約が追加された後、除去された場合には、基本キー索引もまた除去されます。
- 基本キー制約が区分化表に追加された後、ユーザーによって削除された場合には、ユーザーは表のキーを保持することができません。
- 既存の非区分化表に基本キー制約が存在しないものの、表にキーが設定されている場合には、表が区分化表に変更されるときにキーが削除されます。
- DB2 マルチシステムのファイル (分散表) は複数システムの間ですでに区分化されているため、単一システム上の複数のメンバーに区分化することはできません。
- 行を他の区画に移動するような区分化キーの更新は、許容さていません。
- 区分化キーの数は 120 に制限されています。
- すべての SQL 相対レコード処理は、DB2 マルチシステム・サポートと同様に行われます。相対レコード番号は、表全体ではなく、個々の区画ごとに決定されます。たとえば、レコード 27 に対する読み取りは、それぞれの区画のレコード 27 を読み取ることを意味します。それぞれの区画に固有のレコード 27 が存在し、すべて異なるレコードである可能性があります。
- 区分化キー列のデータ・タイプには、いくつかの制限があります。範囲による区分化の場合、表の区分化に使用される列のデータ・タイプを BLOB、CLOB、DBCLOB、DATALINK、浮動小数点タイプ、またはこれらのタイプに基づく特殊タイプにすることはできません。ハッシュ区分化の場合、区分化キーに含まれる列のデータ・タイプを LOB、DATE、TIME、 TIMESTAMP、浮動小数点タイプ、またはこれらのいずれかに基づく特殊タイプにすることはできません。
- 区分化表のすべての区画を処理するには、以下の CL コマンドを使用するアプリケーションを変更して、 Member *ALL を使用するようにしなければなりません。
  - 物理ファイル・メンバーのクリア (CLRPFM)

– インポート・ファイルからのコピー (CPYFRMIMPF)
– インポート・ファイルへのコピー (CPYTOIMPF)
– ネットワーク・ファイルの削除 (DLTNETF)
– QUERY ファイルのオープン (OPNQRYF)
– QUERY の実行 (RUNQRY)
– オブジェクト・ロックの処理 (WRKOBJLCK)
– ジャーナル処理済み変更適用 (APYJRNCHG)
– ジャーナル処理済み変更適用拡張 (APYJRNCHGX)
– ジャーナルの表示 (DSPJRN)
– ジャーナル項目の受信 (RCVJRNE)
– ジャーナル処理済み変更の除去 (RMVJRNCHG)
– ジャーナル項目の検索 (RTVJRNE)
– オブジェクトの復元 (RSTOBJ)
– オブジェクトの保管 (SAVOBJ)
– オブジェクトの保管と復元 (SAVRSTOBJ)

DB2 マルチシステムで使用可能なスカラー関数

DB2 マルチシステムでは、分散ファイルを処理する際にスカラー関数を使用することができます。

これらの関数を使用すると、ファイルが分散された後のデータの場所だけでなく、ファイルにおけるデータの分散方法を判別することができます。データベース管理者は、分散ファイルを処理する場合、これらの関数をデバッグ用ツールとして使用することができます。

これらのスカラー関数には PARTITION、HASH、NODENAME、および NODENUMBER があります。 これらの関数は、SQL または QUERY ファイルのオープン (OPNQRYF) コマンドによって使用することができます。

関連資料

SQL 解説書

制御言語

DB2 マルチシステムでの PARTITION

PARTITION 関数によって、分散リレーショナル・データベースの特定の行が格納されている区画番号を判別することができます。

区画番号を知っていると、ノード・グループ内のどのノードにその区画番号が含まれるかを判別することができます。

DB2 マルチシステムでの PARTITION の例

ここでは、PARTITION 関数を使用する例を示します。

- EMPLOYEE 表の行すべての PARTITION 番号を検出します。

  SQL ステートメント:

```
SELECT PARTITION(CORPDATA.EMPLOYEE), LASTNAME
   FROM CORPDATA.EMPLOYEE
```

OPNQRYF コマンド:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE))
        FORMAT(FNAME)
        MAPFLD((PART1 '%PARTITION(1)'))
```

- EMPLOYEE 表から、区画番号が 100 に等しいすべての行に社員番号 (EMPNO) を選択します。

SQL ステートメント:

```
SELECT EMPNO
   FROM CORPDATA.EMPLOYEE
   WHERE PARTITION(CORPDATA.EMPLOYEE) = 100
```

OPNQRYF コマンド:

```
OPNQRYF FILE((EMPLOYEE)) QRYSLT('%PARTITION(1) *EQ 100')
```

- EMPLOYEE 表と DEPARTMENT 表を結合し、2 つの表の行が同じ区画番号を持つ結果の行すべてを選択します。

SQL ステートメント:

```
SELECT *
   FROM CORPDATA.EMPLOYEE X, CORPDATA.DEPARTMENT Y
   WHERE PARTITION(X)=PARTITION(Y)
```

OPNQRYF コマンド:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE) (CORPDATA/DEPARTMENT))
        FORMAT(FNAME)
        JFLD((1/PART1 2/PART2 *EQ))
        MAPFLD((PART1 '%PARTITION(1)')
               (PART2 '%PARTITION(2)'))
```

DB2 マルチシステムでの HASH

HASH 関数は、指定の式にハッシュ関数を適用することによって、区画番号を戻します。

DB2 マルチシステムでの HASH の例

区分化キーが EMPNO と LASTNAME から構成されている場合、区画が何になるべきかを判別するために HASH 関数を使用します。

- この QUERY では、EMPLOYEE のすべての行の区画番号が戻されます。

SQL ステートメント:

```
SELECT HASH(EMPNO, LASTNAME)
   FROM CORPDATA.EMPLOYEE
```

OPNQRYF コマンド:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE))
        FORMAT(FNAME)
        MAPFLD((HASH '%HASH(1/EMPNO, 1/LASTNAME)'))
```

DB2 マルチシステムでの NODENAME

NODENAME 関数によって、分散リレーショナル・データベースの特定の行が格納されているリレーショナル・データベース (RDB) の名前を判別することができます。

ノード名を知っていると、その行を含むシステム名を判別することができます。これは、ある行を特定のノードに再配布する必要があるかどうかを判別する際に役立ちます。

DB2 マルチシステムでの NODENAME の例

ここでは、NODENAME 関数を使用する例を示します。

- EMPLOYEE 表の行すべてのノード名と区画番号と、各行の EMPNO 列の対応する値を検出します。

  SQL ステートメント:

```
SELECT NODENAME(CORPDATA.EMPLOYEE), PARTITION(CORPDATA.EMPLOYEE), EMPNO
   FROM CORPDATA.EMPLOYEE
```

- EMPLOYEE 表のすべてのレコードのノード名を検出します。

  OPNQRYF コマンド:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE))
        FORMAT(FNAME)
        MAPFLD((NODENAME '%NODENAME(1)'))
```

- EMPLOYEE 表と DEPARTMENT 表を結合し、社員番号 (EMPNO) を選択して、生成された結合に各行が含まれるときの元のノードを判別します。

  SQL ステートメント:

```
SELECT EMPNO, NODENAME(X), NODENAME(Y)
   FROM CORPDATA.EMPLOYEE X, CORPDATA.DEPARTMENT Y
   WHERE X.DEPTNO=Y.DEPTNO
```

  OPNQRYF コマンド:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE) (CORPDATA/DEPARTMENT))
        FORMAT(FNAME)
        JFLD((EMPLOYEE/DEPTNO DEPARTMENT/DEPTNO *EQ))
        MAPFLD((EMPNO 'EMPLOYEE/EMPNO')
               (NODENAME1 '%NODENAME(1)')
               (NODENAME2 '%NODENAME(2)'))
```

- EMPLOYEE 表と DEPARTMENT 表を結合し、2 つの表の行が同じノードにある結果の行すべてを選択します。

  SQL ステートメント:

```
SELECT *
   FROM CORPDATA.EMPLOYEE X, CORPDATA.DEPARTMENT Y
   WHERE NODENAME(X)=NODENAME(Y)
```

  OPNQRYF コマンド:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE) (CORPDATA/DEPARTMENT))
        FORMAT(FNAME)
        JFLD((1/NODENAME1 2/NODENAME2 *EQ))
        MAPFLD((NODENAME1 '%NODENAME(1)')
               (NODENAME2 '%NODENAME(2)'))
```

DB2 マルチシステムでの NODENUMBER

NODENUMBER 関数によって、分散リレーショナル・データベースの特定の行が格納されているノード番号を判別することができます。

ノード番号とは、ノード・グループの作成時に、ノード・グループ内で各ノードに割り当てられる固有の番号のことをいいます。ノード番号を知っていると、その行を含むシステム名を判別することができます。これは、ある行を特定のノードに再配布する必要があるかどうかを判別する際に役立ちます。

DB2 マルチシステムでの NODENUMBER の例

ここでは、NODENUMBER 関数を使用する例を示します。

CORPDATA.EMPLOYEE が分散表の場合、各行のノード番号と社員名が戻されます。

SQL ステートメント:

```
SELECT NODENUMBER(CORPDATA.EMPLOYEE), LASTNAME
   FROM CORPDATA.EMPLOYEE
```

OPNQRYF コマンド:

```
OPNQRYF FILE((CORPDATA/EMPLOYEE))
        FORMAT(FNAME)
        MAPFLD((NODENAME '%NODENUMBER(1)')
               (LNAME '1/LASTNAME'))
```

DB2 マルチシステムでの特殊レジスター

DB2 マルチシステムでは、特殊レジスターのすべてのインスタンスが、リモート・ノードへ QUERY を送信する前にコーディネーター・ノードで解決されます。(コーディネーター・ノード とは、QUERY が開始されたシステムのことをいいます。) このように、すべてのノードが、整合のとれた特殊レジスター値によって QUERY を実行します。

次に、特殊レジスターに関する規則を示します。

- CURRENT SERVER は、常にコーディネーター・ノードのリレーショナル・データベース名を戻します。
- USER 特殊レジスターは、コーディネーター・ノード上でジョブを実行するユーザー・プロファイルを戻します。
- CURRENT DATE、CURRENT TIME、および CURRENT TIMESTAMP は、コーディネーター・ノードの時刻機構 (TOD) によります。
- CURRENT TIMEZONE は、コーディネーター・ノードのシステム値 QUTCOFFSET の値です。

DB2 マルチシステムでの相対レコード番号関数

相対レコード番号 (RRN) 関数は、分散ファイルのノードに格納された行の相対レコード番号を戻します。

RRN は分散ファイルに固有のものではありません。RRN を NODENAME または NODENUMBER のいずれかと結合すると、ファイル内の固有レコードが指定されます。

DB2 マルチシステムでのパフォーマンスとスケーラビリティー

DB2 マルチシステム では、データベースの容量を増やして、QUERY のパフォーマンスの向上を実現し、より簡単な方式によって、リモート・データベース・アクセスを可能にすることができます。

DB2 マルチシステムを使用すると、ユーザーおよびアプリケーションは、ローカル・システムからファイルにアクセスするだけで済みます。ローカル・ファイルではなく、分散ファイル内のデータにアクセスできるようにするために、コードを変更する必要はありません。分散リレーショナル・データベース・アーキテクチャー (DRDA) および分散データ管理機能 (DDM) などの機能では、リモート・データにアクセスするために、リモート・ファイルまたはリモート・システムに対してアクセスを明示的に指示する必要があります。DB2 マルチシステムは、ユーザーが意識しなくてもすむようにリモート・アクセスを処理します。

DB2 マルチシステムには、データベース拡張のための単純な拡張パスも用意されています。

DB2 マルチシステムを使用する必要がある理由

パフォーマンスの向上が、特定の QUERY にとって非常に問題になる場合があります。

処理すべき大量のデータがあるのに対して、結果セットが比較的小さい QUERY の場合には、ファイルが分配されるシステムの数に比例してパフォーマンスが向上することがテストによりわかっています。たとえば、500 万レコードのファイルから上位 10 位までの高額所得者を QUERY するとします。DB2 マルチシステムでは、ファイルを 2 つのシステム間で等分に区分化することによって、QUERY に対する応答時間をほぼ 2 分の 1 にすることができます。3 つのシステムに区分化すると、応答時間は、単一のシステムで QUERY を実行した場合のほぼ 3 分の 1 になります。これは最適な条件の場合であり、データをノード間で移動する必要がある複雑な結合の場合にはあてはまりません。

ファイルが非常に小さいか、または主に単一レコードの読み取り処理か書き込み処理に使用される場合、ファイルを区分化しても、パフォーマンスはほとんどあるいはまったく向上しません。それどころか、パフォーマンスが若干低下する場合があります。このような場合、QUERY のパフォーマンスはむしろ物理的な接続の速度によって決まります。ただしこのような場合でも、ノード・グループ内のすべてのシステムのユーザーは、データが配布されていても、使いなれた従来のローカル・ファイル・データベース方式を使用して、そのデータにアクセスすることができるという利点があります。ユーザーは、すべての環境において、このローカル・システムの透過性と、ノード・グループ内のシステム間のデータ冗長度の除去という恩恵に浴しています。

別の並列機能の DB2 UDB Symmetric Multiprocessing もパフォーマンスを向上できます。対称マルチプロセッシング (SMP) を使用すると、区分化されたファイルが処理されて、システムのいずれかがマルチプロセッサー・システムである場合、パフォーマンス向上において乗数効果が得られます。ファイルを 3 つのシステム間で区分化して、各システムが 4 ウェイのプロセッサー・システムである場合は、DB2 マルチシステムと SMP の機能がともに作動します。前に示した 500 万レコードの例をとると、応答時間は、並列機能をまったく使用しないで QUERY が実行された場合のおよそ 12 分の 1 になります。ファイル・サイズと QUERY の詳細度によって、実際の向上に影響が及ぶ場合があります。

QUERY を実行すると、QUERY を実行するための大量の作業が並列して実行されます。これにより、QUERY 処理全体のパフォーマンスが向上します。システムは、QUERY を分割し、QUERY の適切な部分を適切なシステム上で処理します。これにより、処理がますます効率的に、また自動的に行われるようになります。何も指定しなくても、このような、より効率的な処理を行うことができます。

注: QUERY によっては、パフォーマンスが向上しない場合があります。特に、大量のデータを移動しなければならない場合などです。

各ノードは、そのノードに物理的に格納されたレコードだけを処理する必要があります。QUERY で、区分化キーに対する選択を指定した場合、QUERY 最適化プログラムが、1 つのノードだけを QUERY する必要があると判別することがあります。次の例では、郵便番号フィールドが、ORDERS ファイルの SQL ステートメント内の区分化キーです。

```
SELECT NAME, ADDRESS, BALANCE FROM PRODLIB/ORDERS WHERE ZIP='48009'
```

ステートメントが実行されると、最適化プログラムは、1 つのノードだけを QUERY する必要があると判別します。 48009 という郵便番号を含むすべてのレコードが、同じノードに配布されることを覚えておいてください。

次の SQL ステートメントの例では、ノード・グループ内のすべての System i モデルのプロセッサー機能を使用して、ステートメントを並列処理することができます。

```
SELECT ZIP, SUM(BALANCE) FROM PRODLIB/ORDERS GROUP BY ZIP
```

最適化プログラムに、適切なデータを含むシステム宛てにだけ直接入出力要求を出させると、1 つまたは複数のシステムがアクティブではない場合にでも QUERY を実行できるという利点も得られます。この例として、ある業務の各部門が異なるシステムにデータを保存するように区分化されたファイルがあげられます。あるシステムが使用できない場合でも、残りの部門に関連するデータに対してファイル入出力操作を実行することができます。活動状態ではない部門に対する入出力要求は失敗します。

最適化プログラムは、2 フェーズのコミット・プロトコルを使用して、データの保全性を保証します。複数のシステムがアクセスされるため、コミットメント制御を要求すると、すべてのシステムが、保護会話を使用します。*保護会話*とは、トランザクションの途中、または単一データベース操作の途中でシステム障害が発生した場合、その時点までに行われた変更すべてがロールバックされるということを意味します。

保護会話を使用すると、一部のコミットメント制御オプションは、パフォーマンスを高めるために、リモート・ノードで変更されます。発信待機オプションは Y に、ボート読み取り専用オプションは Y に設定されます。パフォーマンスをさらに向上させるには、コミットメント変更オプション (QTNCHGCO) API を使用して、QUERY を開始するシステム上で発信待機オプションを N に変更します。これらのコミットメント・オプションの値の効果については、Information Center のトピック『API』を参照してください。

関連資料

Application programming interfaces

DB2 マルチシステムでのパフォーマンス向上のヒント

分散データ (DSTDTA) パラメーターにいくつかの値を指定することで、パフォーマンスを向上することができます。

優れたパフォーマンスを確保する方法の 1 つに、*BUFFERED をデータベース・ファイルの一時変更 (OVRDBF) コマンドの DSTDTA パラメーターに指定する方法があります。これは、システムに対して、ファイルに即時更新を加えるという犠牲を払う可能性があっても、できるだけ早く分散ファイルからデータを検索するように指示するものです。DSTDTA(*BUFFERED) は、ファイルが読み取り専用のためにオープンされている場合のパラメーターのデフォルト値です。

DSTDTA パラメーターには、*CURRENT および *PROTECTED という値もあります。*CURRENT を使用すると、他のユーザーによるファイルへの更新が可能になりますが、パフォーマンスが若干低下します。ファイルを更新のためにオープンした場合は、DSTDTA(*CURRENT) がデフォルト値になります。*PROTECTED では、*CURRENT と同様のパフォーマンスが得られますが、*PROTECTED を使用すると、ファイルのオープン中に他のユーザーが更新を行えなくなります。

DB2 マルチシステムによるデータベース・システムの拡張方法

分散リレーショナル・データベース・ファイルを使用すると、 System i モデルの構成をより簡単に拡張することができます。

DB2 マルチシステムの前には、1 つのシステムから 2 つのシステムに移行したい場合、解決しなければならないデータベースの問題がいくつかありました。ユーザーの半分を新しいシステムに移動する場合、データの半分もその新しいシステムに移動することになります。これにより、データベース関連のアプリケーションをすべて再作成することになりますが、これは、アプリケーションがデータの所在を知っていなければならないためです。アプリケーションを作成し直したら、分散リレーショナル・データベース・アーキテクチャー (DRDA または分散データ管理機能 (DDM) などのリモート・アクセスを使用して、複数のシステムにあるファイルにアクセスする必要があります。これを行わない場合は、なんらかのデータ複製機能が使用されます。データ複製機能を使用すると、データのコピーが複数存在し、さらに多くの記憶域を使用することになり、システムはファイルの複数のコピーを同じレベルで保持するための作業も行います。

DB2 マルチシステムでは、構成に新しいシステムを追加するためのプロセスが大幅に簡素化されています。データベース・ファイルは、複数のシステム間で区分化されます。次に、アプリケーションが新しいシステムに移動します。アプリケーションは変更されません。アプリケーションにプログラミング変更を加える必要はありません。ユーザーは、新しいシステム上で実行できるようになり、すぐに同じデータにアクセスすることができます。さらに拡張が必要になる場合は、追加のシステムを含む新しいノード・グループ間でファイルを再分配することができます。

ネットワークにシステムを追加する際の再分配の問題

ノード・グループ間でのファイルの再分配は、非常に簡単な処理です。

物理ファイルの変更 (CHGPF) コマンドを使用すると、ファイルに新しいノード・グループを指定するか、またはファイルの新しい区分化キーを指定することができます。CHGPF コマンドを使用すると、ローカル・ファイルを分散ファイルにするか、分散ファイルをローカル・ファイルにするか、または分散ファイルをノードの異なるセット間、または異なる区分化キーによって再分配することができます。

再分配のプロセスでは、ファイル内のほとんどすべてのレコードの移動が必要な場合があることに注意してください。非常に大きなファイルの場合、処理に長時間かかり、その間ファイルのデータが使用不可になることがあります。したがって、ファイルの再分配を頻繁に行ったり、適切な計画なしに行ったりはしないでください。

ローカル物理ファイルを分散ファイルに変更するには、ノード・グループ (NODGRP) と区分化キー (PTNKEY) の各パラメーターを CHGPF コマンドに指定する必要があります。このコマンドが実行されると、ファイルが変更されてノード・グループ内のノード間に分配され、さらに、PTNKEY パラメーターに指定された区分化キーを使用して、既存のデータも分配されます。

分散ファイルをローカル・ファイルに変更するには、NODGRP(*NONE) を CHGPF コマンドに指定する必要があります。これにより、ファイルのリモート部分すべてが削除され、すべてのデータがローカル・システムに返されます。

分散ファイルの区分化キーを変更するには、CHGPF コマンドの PTNKEY パラメーターに必要なフィールドを指定してください。これによって、ファイルが分配されるシステムに影響が及ぶことはありません。ハッシュ・アルゴリズムを新しい区分化キーに適用する必要があるため、データのすべてが再分配されます。

ファイルが分配されるシステムの新しい集合を指定するには、CHGPF コマンドのノード・グループ (NODGRP) パラメーターにノード・グループ名を指定してください。これにより、ファイルは、システムの新しい集合に分配されます。新しい区分化キーを PTNKEY パラメーターに指定することができます。PTNKEY パラメーターが指定されていないか、または *SAME が指定されている場合は、既存の区分化キーが使用されます。

CHGPF コマンドは、ノード・グループに新しいシステムが追加された場合、ファイルの新しい部分の作成を行います。CHGPF コマンドは、システムが新しいノード・グループにない場合、ファイルの部分的削除を行います。ノード・グループを削除して再作成し、CRTPF コマンドを使用してファイルを再分配したい場合は、CHGPF コマンドの NODGRP パラメーターにノード・グループ名を指定する必要があります。これは、ノード・グループ名が、ファイルが最初に作成されたときに使用されたものと同じであってもです。これは、システムにそのノード・グループを調べさせて、ファイルを再分配させたいということを示します。ただし、ノード・グループを NODGRP パラメーターに指定して、システムが、ファイル内に現在格納されているノード・グループとそのノード・グループが同じであると認めた場合は、PTNKEY を同時に指定しないかぎり、再分配は起こりません。

参照制約のあるファイルでは、CHGPF コマンドを使用して親ファイルと従属ファイルを分散ファイルにしたい場合、次の作業を行う必要があります。

1. 参照制約を除去する。制約を除去しないと、最初に分配するファイルに制約エラーが起こります。これは、参照制約関係内の他のファイルが、まだ分散ファイルではないためです。
2. CHGPF コマンドを使用して、両方のファイルを分散ファイルにする。
3. 参照制約をもう一度追加する。

関連概念

24 ページの『DB2 マルチシステムでのデータ分散のカスタマイズ』
データの配置についてはシステムが担当するため、ユーザーはレコードの実際の所在を知る必要はありません。ただし、特定のレコードが特定のシステムに常に格納されるようにしたい場合は、ノード・グループ属性の変更 (CHGNODGRPA) コマンドを使用して、これらのレコードの所在を指定することができます。

DB2 マルチシステムでのパフォーマンスのための QUERY 設計

以下の指針に基づいて QUERY を設計できます。この方法により、分散ファイル使用の QUERY を実行するときに、より効率的に QUERY リソースを使用できます。

さらに分散ファイルを使用する QUERY が組み込まれる方法についても説明します。これにより、QUERY を調整して、分散環境においてより効率的に実行できるようにします。

分散ファイルは、SQL、QUERY ファイルのオープン (OPNQRYF) コマンド、またはシステム上のいずれかの QUERY インターフェースを使用して QUERY を実行することができます。QUERY は、単一ファイル QUERY または結合 QUERY のいずれかにすることができます。分散ファイルとローカル・ファイルの組み合わせを結合して使用することができます。

このトピックは、ユーザーが、非分散環境での QUERY の実行と最適化に詳しいことを前提としています。これらのトピックについてさらに詳しい情報が必要な場合は、次のようにしてください。

- SQL ユーザーは、「SQL 解説書」および「SQL プログラミング 概念」の情報を参照する必要があります。
- 非 SQL ユーザーは、「データベース・プログラミング」および「制御言語 (CL)」の情報を参照する必要があります。

またこのトピックでは、並列処理機能を活用して、データ移動を最小化することによって、分散 QUERY のパフォーマンスを改善する方法も示しています。

関連概念

23 ページの『DB2 マルチシステムでの区分化キーの選択』
システムが最も効率的な方法で区分化ファイルを処理するために、区分化キーを設定または使用する際に検討できるヒントがいくつかあります。

SQL プログラミング

データベース・プログラミング

関連資料

SQL 解説書

制御言語

DB2 マルチシステムでの最適化: 概要

分散 QUERY は、分散レベルとローカル・レベルで最適化されます。

- 分散レベルでの最適化は、QUERY を最も効率的なステップに分割して、どのノードがこれらのステップを処理するかに焦点を当てています。 分散最適化プログラムは、分散 QUERY に特有のものです。分散最適化プログラムについてはこのトピックで説明します。
- ローカル (ステップ) レベルでの最適化は、非分散環境で起こるものと同じ最適化です。この最適化プログラムは、ご存じのものと考えられます。ローカル・レベルでの最適化については、このトピックでは最小限の説明をします。

分散最適化の基本的な前提事項として、各データ・ノードに格納されたレコードの数がほぼ等しく、分散 QUERY のすべてのシステムが類似の構成であるということがあります。分散最適化プログラムによって行われる決定は、システムおよびコーディネーター・ノード・システムのデータ統計に基づいています。

分散 QUERY に複数のステップが必要な場合は、一時結果ファイルが使用されます。*一時結果ファイル* とは、特定の QUERY ステップの結果を含めるために使用されるシステム作成の一時ファイル (ライブラリー QRECOVERY に保存) のことをいいます。一時結果ファイルの内容は、次の QUERY ステップの入力として使用されます。

DB2 マルチシステムでの単一ファイル QUERY の実施と最適化

単一ファイル QUERY を行うために、QUERY が指定されたシステムであるコーディネーター・ノードは、QUERY を送信するためのファイルのノードを判別します。これらのノードは、QUERY を実行して、QUERY されたレコードをコーディネーター・ノードに返します。

このトピックの例はすべて、DEPARTMENT および EMPLOYEE という分散ファイルを使用します。これらのファイルのノード・グループは、SYSA、SYSB、および SYSC からなります。データは、部門番号で区分化されます。

次の SQL ステートメントは、分散ファイル DEPARTMENT を作成します。

```
CREATE TABLE DEPARTMENT
       (DEPTNO CHAR(3) NOT NULL,
       DEPTNAME VARCHAR(20) NOT NULL,
       MGRNO CHAR(6),
       ADMRDEPT CHAR(3) NOT NULL)
      IN NODGRP1 PARTITIONING KEY(DEPTNO)
```

表 2. DEPARTMENT 表

ノード	レコード番号	DEPTNO	DEPTNAME	MGRNO	ADMRDEPT
SYSA	1	A00	Support services	000010	A00
SYSB	2	A01	Planning	000010	A00
SYSC	3	B00	Accounting	000050	B00
SYSA	4	B01	Programming	000050	B00

次の SQL ステートメントは、分散ファイル EMPLOYEE を作成します。

```
CREATE TABLE EMPLOYEE
       (EMPNO CHAR(6) NOT NULL,
       FIRSTNME VARCHAR(12) NOT NULL,
       LASTNAME VARCHAR(15) NOT NULL,
       WORKDEPT CHAR(3) NOT NULL,
       JOB CHAR(8),
       SALARY DECIMAL(9,2))
      IN NODGRP1 PARTITIONING KEY(WORKDEPT)
```

表 3. EMPLOYEE 表

ノード	レコード番号	EMPNO	FIRSTNME	LASTNAME	WORK DEPT	JOB	SALARY
SYSA	1	000010	Christine	Haas	A00	Manager	41250
SYSA	2	000020	Sally	Kwan	A00	Clerk	25000
SYSB	3	000030	John	Geyer	A01	Planner	35150
SYSB	4	000040	Irving	Stern	A01	Clerk	32320
SYSC	5	000050	Michael	Thompson	B00	Manager	38440
SYSC	6	000060	Eileen	Henderson	B00	Accountant	33790
SYSA	7	000070	Jennifer	Lutz	B01	Programmer	42325
SYSA	8	000080	David	White	B01	Programmer	36450

以下の QUERY は定義済みの分散ファイル EMPLOYEE を使用し、索引 EMPIDX はフィールド SALARY に対して作成されます。この QUERY は SYSA に対して入力されます。

SQL ステートメント:

```
SELECT * FROM EMPLOYEE WHERE SALARY > 40000
```

OPNQRYF コマンド:

```
OPNQRYF FILE((EMPLOYEE)) QRYSLT('SALARY > 40000')
```

この場合、SYSA は QUERY を、SYSA を含む EMPLOYEE のすべてのノードに送ります。各ノードは QUERY を実行し、レコードを SYSA に返します。分散索引は、ファイル EMPLOYEE のフィールド SALARY 上に存在するため、各ノードに実行される最適化では、索引を使用するかどうかを決定します。

次の例では、QUERY は SYSA に指定されますが、QUERY は EMPLOYEE ファイルが存在するノードのサブセットに送られます。この場合、QUERY は SYSA でのみローカルに実行されます。

SQL ステートメント:

```
SELECT * FROM EMPLOYEE WHERE WORKDEPT = 'A00'
```

OPNQRYF コマンド:

```
OPNQRYF FILE((EMPLOYEE)) QRYSLT('WORKDEPT = 'A00')
```

分散 QUERY 最適化プログラムは、分離可能なレコード選択 WORKDEPT = 'A00' があることを判別します。この QUERY では区分化キー WORKDEPT を含みます。最適化プログラムは、'A00' という値をハッシュし、ハッシュ値に基づいて、この条件を満たすすべてのレコードがあるノードを検索します。この場合、この条件を満たすすべてのレコードは SYSA 上にあるため、QUERY はそのノードにのみ送られます。QUERY は SYSA で発生したため、QUERY は SYSA でローカルに実行されます。

次の条件は、QUERY が実行されるノードの数を一部に限定します。

- 区分化キーのすべてのフィールドは、分離可能なレコード選択でなければならない
- すべての述部は、等号 (=) 演算子を使用しなければならない
- 区分化キーのすべてのフィールドは、リテラルに比較されなければならない

注: パフォーマンス上の理由から、QUERY を特定のノードに指定するために、区分化キーに一致するレコード選択述部を指定する必要があります。スカラー関数 NODENAME、PARTITION、および NODENUMBER を使用するレコード選択も、QUERY を特定ノードに指示することができます。

DB2 マルチシステムでのレコード順序の実施と最適化

QUERY に順序を指定した場合、順序の基準がその QUERY とともに送られるため、各ノードは順序付けを並行して実行することができます。最終組み合わせまたは分類がコーディネーター・ノードに対して実行されるかどうかは、指定する QUERY のタイプによって決まります。

各ノードから受け取った順序付けられたレコードの組み合わせは、最適化されています。組み合わせは、コーディネーター・ノードがレコードを受け取ると起こります。分類に対して組み合わせが持つ主なパフォーマンス上の利点は、すべてのノードからのレコードすべてを分類しなくても、レコードを返せることです。

各ノードから受け取った順序付けられたレコードの分類によって、レコードが返される前に、各ノードからのレコードすべてが読み取られ、分類されて、一時結果ファイルに書き込まれます。

組み合わせは、順序が指定されていて、UNION も最終グループ化も必要ではない場合に起こります。そうでない場合、順序 QUERY では、コーディネーター・ノードに対して、最終分類が実行されます。

データのコピー許可 (ALWCPYDTA) パラメーターは、分散 QUERY の各ノードが順序基準をどのように処理するかに影響を与えます。この ALWCPYDTA パラメーターは、QUERY ファイルのオープン (OPNQRYF) および SQL 開始 (STRSQL) の各 CL コマンドと、SQLxxx の作成 (CRTSQLxxx) プリコンパイラー・コマンドに指定されます。

- ALWCPYDTA(*OPTIMIZE) を使用すると、各ノードは、分類または索引のどちらを使用して順序基準を実施するかを決めることができます。このオプションは最適です。
- OPNQRYF コマンドと QUERY API (QQQQRY) では、ALWCPYDTA(*YES) または ALWCPYDTA(*NO) を使用すると、各ノードは、指定の順序フィールドと正確に一致する索引を自動的に使用します。これは、最適化プログラムがローカル・ファイルで順序を処理する方法よりも制限されています。

DB2 マルチシステムでの UNION 文節と DISTINCT 文節の実施と最適化

結合された SELECT ステートメントが分散ファイルを参照する場合、このステートメントは分散 QUERY として処理されます。

ステートメントの処理は並行して起こります。結合された各 SELECT のレコードは、結合操作を実行するために、コーディネーター・ノードに返されます。この点で、結合演算子は順次処理されます。

ORDER BY 文節が結合 QUERY とともに指定されている場合、各ノードからのレコードはすべてコーディネーター・ノードで受け取られて、レコードが返される前に分類されます。

DISTINCT 文節が分散 QUERY に指定されている場合、ORDER BY 文節を追加すると、ORDER BY 文節が指定されていない場合よりも速くレコードを返すことができます。DISTINCT に ORDER BY を指定すると、各ノードは、レコードを並列に順序付けすることができます。コーディネーター・ノードの最終組み合わせでは、各ノードから順序付けられたレコードを読み取って、それらのレコードを正しい順序で組み合わせ、最終分類を行わずに、重複するレコードを削除することができます。

DISTINCT 文節が ORDER BY 文節なしで指定されている場合、各ノードからのレコードすべてが、分類が実行されるコーディネーター・ノードに送られます。重複するレコードは、分類されたレコードが返されるときに削除されます。

DB2 マルチシステムでの DSTDTA パラメーターおよび ALWCPYDTA パラメーターの処理

データのコピー許可 (ALWCPYDTA) パラメーターは、データベース・ファイルの一時変更 (OVRDBF) コマンドのデータの配布 (DSTDTA) パラメーターに指定された値を変更することができます。

一時変更コマンドにライブ・データ (DSTDTA(*CURRENT)) を使用するように指定し、次のいずれかがあてはまる場合には、以下のようになります。

- 一時コピーが必要であり、ALWCPYDTA(*YES) が指定されている
- パフォーマンスを向上させるために一時コピーが選択されて、ALWCPYDTA(*OPTIMIZE) が指定されている

この場合、DSTDTA は DSTDTA(*BUFFERED) に変更されます。

DSTDTA(*BUFFERED) を受け入れることが不可能であり、QUERY が一時コピーを必要としない場合は、ALWCPYDTA(*YES) を指定して、DSTDTA(*CURRENT) を有効に保つ必要があります。

DB2 マルチシステムでの結合操作の実施と最適化

非分散結合 QUERY でのパフォーマンスの考慮事項だけでなく、分散ファイルに関与する QUERY についてもパフォーマンスの考慮事項があります。

結合は、データが区画互換の場合にのみ実行することができます。分散 QUERY 最適化プログラムは、データを区画互換にする計画を生成します。これには、ノード間でのデータの移動が関与する場合があります。

データは、両方のファイルの区分化キーのデータが、同じノード・グループを使用して同じノードにハッシュする場合、*区画互換* になります。たとえば、同じ数値を長整数フィールドまたは短整数フィールドのいずれかに格納しても、同じ値にハッシュします。

次のデータ・タイプは区画互換です。

- 長整数 (4 バイト)、短整数 (2 バイト)、パック 10 進数、およびゾーン数値
- 固定長および可変長の SBCS 文字と、DBCS 混用、DBCS 択一、または DBCS 専用
- 固定長および可変長のグラフィック

日付、時刻、タイム・スタンプ、および浮動小数点の数値データ・タイプは、区分化キーにできないため、区画互換ではありません。

分散ファイルに関与する結合は、照合、指定、再区分化、およびブロードキャストの 4 つのタイプに分類されます。次のセクションでは、結合のタイプを定義して、異なる結合タイプの例を示します。

DB2 マルチシステムでの照合結合

照合結合では、結合されるファイルの対応するレコードが、同じノード上に存在します。

結合されるファイルの区分化キーの値は、区画互換です。結合を実行するために、データを別のノードに移動する必要はありません。この方式は、区分化キーのすべてのフィールドが結合フィールドであり、結合演算子が = (等号) 演算子である QUERY の場合にのみ有効です。また、最初のファイルの区分化キーの n 番目のフィールド (ここで、n は区分化キーのフィールド数 1) は、2 番目のファイルの区分化キーの n 番目のフィールドに結合されなければならず、n 番目のフィールドのデータ・タイプは区画互換でなければなりません。区分化キーのすべてのフィールドが結合に含まれていなければならないことに注意してください。区分化キーのフィールドを含まない追加結合述部は、照合結合を実行する能力に影響を与えません。

次の例では、結合述部に両方のファイルの区分化キー・フィールドが関与し、フィールドが区画互換であるため、照合結合を実行することができます。これは、DEPTNO および WORKDEPT の突き合わせ値が同じノード上にあることを暗黙指定します。

SQL ステートメント:

```
SELECT DEPTNAME, FIRSTNME, LASTNAME
          FROM DEPARTMENT, EMPLOYEE
          WHERE DEPTNO=WORKDEPT
```

OPNQRYF コマンド:

```
OPNQRYF FILE((DEPARTMENT) (EMPLOYEE))
               FORMAT(JOINFMT)
               JFLD((DEPTNO WORKDEPT *EQ))
```

この QUERY によって戻されるレコードは次のとおりです。

表 4. QUERY 結果の表示

DEPTNAME	FIRSTNME	LASTNAME
Support services	Christine	Haas
Support services	Sally	Kwan
Planning	John	Geyer
Planning	Irving	Stern
Accounting	Michael	Thompson
Accounting	Eileen	Henderson
Programming	Jennifer	Lutz
Programming	David	White

次の例では、追加結合述部の MGRNO=EMPNO は、照合結合を実行するための能力に影響を与えません。これは、区分化キーがやはり結合述部に関与するためです。

```
SQL:       SELECT DEPTNAME, FIRSTNME, LASTNAME
              FROM DEPARTMENT, EMPLOYEE
              WHERE DEPTNO=WORKDEPT AND MGRNO=EMPNO

OPNQRYF:   OPNQRYF FILE((DEPARTMENT) (EMPLOYEE))
                   FORMAT(JOINFMT)
                   JFLD((DEPTNO WORKDEPT *EQ) (MGRNO EMPNO *EQ))
```

この QUERY によって戻されるレコードは次のとおりです。

表 5. QUERY 結果の表示

DEPTNAME	FIRSTNME	LASTNAME
Support services	Christine	Haas
Accounting	Michael	Thompson

DB2 マルチシステムでの指定結合

指定結合では、少なくとも 1 つのファイルの区分化キーが、結合フィールドとして使用されます。

結合フィールドは、他のファイルの区分化キーには一致しません。ファイルのレコードは、結合フィールド値のハッシュに基づいて、2 番目のファイルの区画マップとノード・グループを使用して、2 番目のファイルのノードに指定、または送信されます。レコードが、一時分散ファイルを介して 2 番目のファイルのノードに移動するとすぐに、照合結合が使用されてデータを結合します。この方式は、区分化キーのすべてのフィールドが、少なくとも 1 つのファイルの結合フィールドである等結合 (equijoin) QUERY にのみ有効です。

次の QUERY では、結合フィールド (WORKDEPT) はファイル EMPLOYEE の区分化キーであり、結合フィールド (ADMRDEPT) は DEPARTMENT の区分化キーではありません。データを移動しないで結合を行うと、DEPARTMENT のレコード 2 を EMPLOYEE のレコード 1 と 2 に結合する必要があり、これらのレコードは異なるノードに保管されるため、結果レコードは失われます。

SQL ステートメント:

```
SELECT DEPTNAME, FIRSTNME, LASTNAME
          FROM DEPARTMENT, EMPLOYEE
          WHERE ADMRDEPT = WORKDEPT AND JOB = 'Manager'
```

OPNQRYF コマンド:

```
OPNQRYF FILE((DEPARTMENT) (EMPLOYEE))
               FORMAT(JOINFMT)
               QRYSLT('JOB *EQ 'Manager')
               JFLD((ADMRDEPT WORKDEPT *EQ))
```

QUERY を実行するために必要な DEPARTMENT のレコードが読み取られ、ADMRDEPT のデータは、区分化マップとノード・グループ EMPLOYEE を使用してハッシュされます。一時ファイルが作成されて、次のようになります。

表 6. 一時表

古いノード	新しいノード	DEPTNAME	ADMRDEPT (新しい区分化キー)
SYSA	SYSA	Support services	A00
SYSB	SYSA	Planning	A00
SYSC	SYSC	Accounting	B00
SYSA	SYSC	Programming	B00

この一時表は、EMPLOYEE に結合されます。結合は、ADMRDEPT が WORKDEPT と区画互換であるため作動します。

表 7. EMPLOYEE 結合表

DEPTNAME	FIRSTNME	LASTNAME
Support services	Christine	Haas
Planning	Christine	Haas
Accounting	Michael	Thompson
Programming	Michael	Thompson

DB2 マルチシステムでの再区分化結合

再区分化結合では、ファイルの区分化キーは結合フィールドとして使用されません。

両方のファイルのレコードは、各ファイルの結合フィールド値をハッシュすることによって移動する必要があります。ファイルの区分化キー・フィールドは、いずれも結合基準に含まれないため、ファイルは、1 つまたは複数の結合フィールドを含む新しい区分化キーでハッシュすることによって、再区分化する必要があります。この方式は、等結合 (equijoin) QUERY にのみ有効です。

SQL ステートメント:

```
SELECT DEPTNAME, FIRSTNME, LASTNAME
          FROM DEPARTMENT, EMPLOYEE
          WHERE MGRNO = EMPNO
```

OPNQRYF コマンド:

```
OPNQRYF FILE((DEPARTMENT) (EMPLOYEE))
                FORMAT(JOINFMT)
                JFLD((MGRNO EMPNO *EQ))
```

この例では、MGRNO も EMPNO も区分化キーではないため、データを再分配する必要があります。

DEPARTMENT のデータは再分配されます。

表 8. 再分配される DEPARTMENT のデータ

古いノード	新しいノード	DEPTNAME	MGRNO (新しい区分化キー)
SYSA	SYSB	Support services	000010
SYSB	SYSB	Planning	000010
SYSC	SYSC	Accounting	000050
SYSA	SYSC	Programming	000050

EMPLOYEE のデータは再配布されます。

表 9. 再配布される EMPLOYEE のデータ

古いノード	新しいノード	FIRSTNME	LASTNAME	EMPNO (新しい区分化キー)
SYSA	SYSB	Christine	Haas	000010
SYSA	SYSC	Sally	Kwan	000020
SYSB	SYSA	John	Geyer	000030
SYSB	SYSB	Irving	Stern	000040
SYSC	SYSC	Michael	Thompson	000050
SYSC	SYSA	Eileen	Henderson	000060
SYSA	SYSB	Jennifer	Lutz	000070
SYSA	SYSC	David	White	000080

この QUERY によって戻されるレコードは次のとおりです。

表 10. QUERY 結果の表示

DEPTNAME	FIRSTNME	LASTNAME
Support services	Christine	Haas
Planning	Christine	Haas
Accounting	Michael	Thompson
Programming	Michael	Thompson

DB2 マルチシステムでのブロードキャスト結合

ブロードキャスト結合では、1 つのファイルの選択レコードすべてが、結合が実行される前に、他のファイルのノードすべてに送信またはブロードキャストされます。

これは、非等結合 (nonequijoin) QUERY 以外のすべてに使用される結合方式です。この方式は、結合基準が、日付、時刻、タイム・スタンプ、または浮動小数点数値のいずれかのデータ・タイプを持つフィールドを使用する場合にも使用されます。

次の例では、分散 QUERY 最適化プログラムは、EMPLOYEE をブロードキャストするように決めています。これは、選択 JOB = 'Manager' を適用することによって、より小さいレコード・セットがブロードキャストされるためです。ノード・グループ内の各ノードの一時ファイルには、選択されたレコードすべてが含まれます。(レコードは各ノードで複写されます。)

SQL ステートメント:

```
SELECT DEPTNAME, FIRSTNME, LASTNAME
        FROM DEPARTMENT, EMPLOYEE
        WHERE DEPTNO <> WORKDEPT AND JOB = 'Manager'
```

OPNQRYF コマンド:

```
OPNQRYF FILE((DEPARTMENT) (EMPLOYEE))
                FORMAT(JOINFMT)
                QRYSLT('JOB *EQ 'Manager')
                JFLD((DEPTNO WORKDEPT *NE))
```

分散 QUERY 最適化プログラムは、次の 2 つの選択レコードを各ノードに送ります。

表 11. 分散 QUERY 最適化プログラムが各ノードに送信するレコード

古いノード	新しいノード	FIRSTNME	LASTNAME	WORKDEPT
SYSA	SYSA、SYSB、SYSC	Christine	Haas	A00
SYSC	SYSA、SYSB、SYSC	Michael	Thompson	B00

この QUERY によって戻されるレコードは次のとおりです。

表 12. QUERY 結果の表示

DEPTNAME	FIRSTNME	LASTNAME
Support services	Michael	Thompson
Planning	Christine	Haas
Planning	Michael	Thompson
Accounting	Christine	Haas
Programming	Christine	Haas
Programming	Michael	Thompson

DB2 マルチシステムでの結合の最適化

分散 QUERY 最適化プログラムは、分散ファイルを結合する計画を生成します。

分散 QUERY 最適化プログラムは、ファイル・サイズ、各ファイルに選択された予期されるレコード数、および可能な分散結合のタイプを調べます。そのうえで、最適化プログラムは QUERY を複数のステップに分割します。各ステップでは、次のステップへの入力として使用される中間結果ファイルが作成されます。

最適化中、分散結合のタイプに基づいて、各結合ステップのコストが計算されます。このコストには、結合ステップに必要なデータ移動の量が部分的に反映されます。このコストは、最終分散計画を判別するために使用されます。

各ステップでは、できるだけ多くの処理が完了されます。たとえば、指定のステップに限定されたレコード選択は、そのステップ中に実行され、各ステップでは、できるだけ多くのファイルが結合されます。各結合ステップでは、複数のタイプの分散結合が行われる場合があります。照合結合と指定結合は、必要なファイ

ルを最初に指定することによって、 1 つの照合結合にすることができます。指定結合と再区分化結合は、すべてのファイルを最初に指定してから、結合を実行することによって、一緒にすることができます。指定結合と再区分化結合は、実際には、結合が行われる前に 1 つまたは複数のファイルが指定される照合結合であることに注意してください。

分散ファイルをローカル・ファイルに結合すると、分散 QUERY 最適化プログラムは、分散ファイルを結合するときにコストを計算する場合と同様に、コストを計算します。このコストに基づいて、分散 QUERY 最適化プログラムは次のいずれかのアクションを行うことができます。

- すべてのローカル・ファイルを、分散ファイルのデータ・ノードにブロードキャストして、照合結合を実行する。
- すべてのローカル・ファイルおよび分散ファイルを、最大分散ファイルのデータ・ノードにブロードキャストして、照合結合を実行する。
- 分散ファイルをコーディネーター・ノードに指定して、そこで結合を実行する。

DB2 マルチシステムでの結合フィールドに対する区分化キー:

結合のタイプに関する前述部分から、照合結合を除くすべての分散結合タイプにデータ移動が必要なことがわかります。

データ移動の必要性を除去して、パフォーマンスを最大化するには、照合結合が可能であるようにすべての QUERY を作成する必要があります。すなわち、分散ファイルの区分化キーは、ファイルを結合するために使用されるフィールドに一致しなければなりません。実行頻度の高い QUERY の場合は、区分化キーを順序またはグループ化基準に一致させるよりも、結合フィールドに一致させる方が重要です。

DB2 マルチシステムでのグループ化の実施と最適化

分散ファイルを使用する QUERY でのグループ化の実施方式は、区分化キーがグループ化基準に含まれるかどうかによって決まります。

グループ化は、下記のセクションで説明するように、1 ステップのグループ化、または 2 ステップのグループ化のいずれかを使用して実施されます。

DB2 マルチシステムでの 1 ステップのグループ化

区分化キーのすべてのフィールドが GROUP BY フィールドである場合、グループ化は、1 ステップのグループ化を使用して実行することができます。これは、グループのデータすべてが同じノード上にあるためです。

次のコードは 1 ステップのグループ化の例を示しています。

SQL ステートメント:

```
SELECT WORKDEPT, AVG(SALARY)
       FROM EMPLOYEE
       GROUP BY WORKDEPT
```

OPNQRYF コマンド:

```
OPNQRYF FILE((EMPLOYEE)) FORMAT(GRPFMT)
             GRPFLD(WORKDEPT)
             MAPFLD((AVGSAL '%AVG(SALARY)'))
```

WORKDEPT は区分化キーとグループ化フィールドの両方であるため、WORKDEPT の類似の値はすべて同じノード上に存在します。たとえば、A00 の値はすべて SYSA にあり、A01 の値はすべて SYSB 上に

存在します。また、B00 の値はすべて SYSC に存在し、B01 の値はすべて SYSA に存在します。このグループ化は、3 つのノードすべてで並行して実行されます。

1 ステップのグループ化を実行するには、区分化キーのすべてのフィールドがグループ化フィールドでなければなりません。追加の非区分化キー・フィールドをグループ化フィールドにすることもできます。

DB2 マルチシステムでの 2 ステップのグループ化

区分化キーがグループ化フィールドには含まれない場合、グループ化は、2 ステップのグループ化を使用して実行することができます。これは、フィールドの類似値が同じノード上にないためです。

次のコードは 2 ステップのグループ化の例を示しています。

SQL ステートメント:

```
SELECT JOB, AVG(SALARY)
       FROM EMPLOYEE
       GROUP BY JOB
```

OPNQRYF コマンド:

```
OPNQRYF FILE((EMPLOYEE)) FORMAT(GRPFMT2)
             GRPFLD(JOB)
             MAPFLD((AVGSAL '%AVG(SALARY)'))
```

この例では、JOB が Clerk であるグループにおいて、Clerk という値が、EMPLOYEE 分散ファイルの 2 つの異なるノードにあることに注意してください。グループ化は、まず 3 つのノードすべてで並行してグループ化を実行することによって実行されます。これにより、コーディネーター・ノードの一時ファイルに置かれる初期グループ化が起こります。QUERY は変更され、コーディネーター・ノードでグループ化が再び実行されて、グループ結果の最終セットが得られます。

ファイル全体のグループ化 (フィールドによるグループがない) は、常に 2 つのステップを使用して実行されます。

QUERY に、HAVING 文節または OPNQRYF コマンドのグループ選択式 (GRPSLT) パラメーターが含まれる場合、最初のグループ化ステップのすべてのグループが、コーディネーター・ノードに戻されます。HAVING 文節または GRPSLT パラメーターは、2 番目のグループ化ステップの一部として処理されます。

QUERY に DISTINCT 列 (集合) が含まれて、2 ステップのグループ化が必要な場合、最初のステップではグループ化は実行されません。かわりに、すべてのレコードがコーディネーター・ノードに戻され、すべてのグループ化が、コーディネーター・ノードにおいて 2 番目のステップの一部として実行されます。

DB2 マルチシステムでのグループ化と結合

QUERY に結合が含まれる場合、実行可能なグループ化のタイプを決めるために使用される区分化キーは、結合を実行するために必要とされたデータの再区分化に基づいています。

次の例では、再区分化結合がグループ化の前に実行されます。これによって、MGRNO という新しい区分化キーが生成されます。MGRNO は区分化キーであるため、グループ化は 1 ステップのグループ化を使用して実行することができます。

SQL ステートメント:

```
SELECT MGRNO, COUNT(*)
       FROM DEPARTMENT, EMPLOYEE
       WHERE MGRNO = EMPNO
       GROUP BY MGRNO
```

OPNQRYF コマンド:

```
OPNQRYF FILE((DEPARTMENT) (EMPLOYEE)) FORMAT(GRPFMT2)
            JFLD((MGRNO EMPNO *EQ))
            GRPFLD(MGRNO)
            MAPFLD((CNTMGR '%COUNT'))
```

次の例では、再区分化結合がグループ化の前に実行されます。これによって、EMPNO という新しい区分化キーが生成されます。EMPNO は WORKDEPT に代わる区分化キーであるため、グループ化は 1 ステップのグループ化を使用して実行することができません。

SQL ステートメント:

```
SELECT WORKDEPT, COUNT(*)
         FROM DEPARTMENT, EMPLOYEE
         WHERE MGRNO = EMPNO
         GROUP BY WORKDEPT
```

OPNQRYF コマンド:

```
OPNQRYF FILE((DEPARTMENT) (EMPLOYEE)) FORMAT(GRPFMT3)
            JFLD((MGRNO EMPNO *EQ))
            GRPFLD(WORKDEPT)
            MAPFLD((CNTDEPT '%COUNT'))
```

DB2 マルチシステムでの SUBQUERY サポート

分散ファイルを SUBQUERY で指定することができます。

SUBQUERY は、独自の検索条件を含むことができ、これらの検索条件は逆に SUBQUERY を含むことができます。したがって、SQL ステートメントは SUBQUERY の階層を含むことができます。SUBQUERY を含む階層の要素は、それらが含む SUBQUERY よりも高いレベルにあります。

関連概念

SQL プログラミング

DB2 マルチシステムでのアクセス計画

分散ファイルを参照する QUERY に格納されたアクセス計画は、ローカル・ファイルに格納されたアクセス計画とは異なります。

分散 QUERY に格納されたアクセス計画は、QUERY が複数のステップに分割される方法と QUERY の各ステップが実行されるノードに関する情報を含む分散アクセス計画です。ステップが各ノードでローカルに実施される方法についての情報は、アクセス計画には保管されません。この情報は実行時に作成されます。

DB2 マルチシステムでの再使用可能なオープン・データ・パス

再使用可能なオープン・データ・パス (ODP) には、分散 QUERY に関する特殊な考慮事項があります。分散 QUERY の他のほとんどと同様に、ODP には分散およびローカルの 2 つのレベルがあります。

分散 ODP は調整 ODP です。分散 ODP は、QUERY をユーザーに関連付けて、ローカル ODP を制御します。ローカル ODP は、QUERY に関与する各システムにあり、分散 ODP を介して要求をとります。

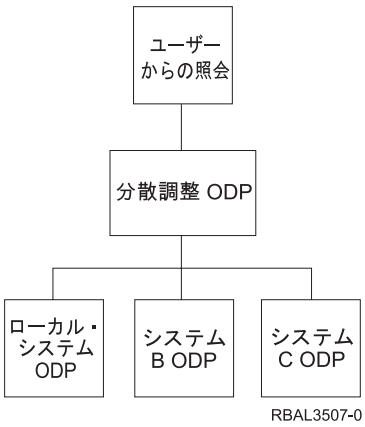


図 7. ODP の例

たとえば、SQL FETCH を実行するための要求が出された場合、その要求は分散 ODP に対して出されます。システムは、その要求をとって、ローカル ODP に対して適切なレコード検索を実行します。

分散 QUERY では、分散 ODP を再使用可能にして、1 つまたは複数のローカル ODP を再使用不可にすることができます。ただし、分散 QUERY ODP が再使用不可の場合は、ローカル ODP も常に再使用不可です。これは、次のことを可能にするために認められています。

- 各ローカル・システムが、そのローカル ODP をオープンする最善の方法 (再使用可能か再使用不可か) を決定することができる
- 通信などの活動資源を維持するために、ローカル ODP 方式に関係なく、分散 ODP を可能なかぎり再使用可能としてオープンする

システムは、基礎となるローカル ODP が再使用不可であっても、可能であればいつでも分散 ODP を再使用可能にしようとします。このことが起こると、システムは ODP リフレッシュを次のように処理します。

- 各ローカル ODP で反復する
- 再使用可能ローカル ODP のリフレッシュを実行する
- 『ハード』・クローズと再使用不可の ODP の再オープンを実行する
- 必要な分散 ODP 自体の残りのリフレッシュがあればリフレッシュを完了する

分散 ODP は、ローカル ODP よりも再使用可能である場合がよくあります。これは、LIKE 文節のホスト変数や、索引対索引作成処理を実行できるように再使用不可を選択する最適化プログラムなど、ローカル ODP を再使用不可にするものによって、分散 ODP が影響を受けないためです。ローカル ODP に影響を与えるものの一部によって、分散 ODP は再使用不可になります。これには、次のものがあります。

- ソート以外での一時ファイルの使用。これらは複数ステップの分散 QUERY と呼ばれ、最適化プログラムのデバッグ・メッセージ CPI4343 が通知されます。
- ライブラリー・リストの変更。これは、QUERY されるファイルに影響を与える可能性があります。
- OVRDBF の変更。これは、QUERY されるファイルに影響を与えます。
- 特殊レジスター USER または CURRENT TIMEZONE の値の変更。
- ジョブ CCSID の変更。
- リソースの再利用 (RCLRSC) コマンドの発行。

ローカル ODP の再使用可能性は、非分散 QUERY ODP にすでに存在する条件と同じ条件によって影響を受けます。したがって、ローカル QUERY ODP に適用されるものと同じ考慮事項が適用されます。

DB2 マルチシステムでの一時結果書き込み機能

一時結果書き込み機能は、常に活動状態にあるシステム開始ジョブです。

システムでは、一時結果書き込み機能は QQQTEMP1 および QQQTEMP2 と呼ばれる対のジョブです。一時結果書き込み機能は、QUERY を実行するジョブからの要求を処理します。これらの要求は、実行すべき QUERY (QUERY ステップの) と、QUERY 結果から充てんされるシステム一時ファイルの名前から構成されます。一時結果書き込み機能は要求を処理して、一時ファイルを充てんします。この中間一時ファイルは、元の QUERY を完了するために要求ジョブによって使用されます。

次の例は、一時結果書き込み機能を必要とする QUERY と、QUERY を処理するために必要なステップを示しています。

SQL ステートメント:

```
SELECT COUNT(*)
             FROM DEPARTMENT a, EMPLOYEE b
             WHERE a.ADMRDEPT = b.WORKDEPT
                AND b.JOB = 'Manager'
```

OPNQRYF コマンド:

```
 OPNQRYF FILE((DEPARTMENT) (EMPLOYEE))
                      FORMAT(FMTFILE)
                      MAPFLD((CNTFLD '%COUNT'))
                      JFLD((1/ADMRDEPT 2/WORKDEPT))
                      QRYSLT('2/JOB = 'Manager')
```

WORKDEPT は、EMPLOYEE の区分化キーですが、ADMRDEPT は DEPARTMENT の区分化キーではありません。QUERY は 2 つのステップで処理する必要がありますが、最適化プログラムは QUERY を次のステップに分割します。

```
  INSERT INTO SYS_TEMP_FILE
  SELECT a.DEPTNAME, a.ADMRDEPT
  FROM DEPARTMENT a
```

および

```
  SELECT COUNT(*) FROM SYS_TEMP_FILE x, EMPLOYEE b
  WHERE x.ADMRDEPT = b.WORKDEPT AND b.JOB = 'Manager'
```

一時結果書き込み機能がジョブ (QUERY 属性の変更 (CHGQRYA) コマンド・オプションによって制御される) で使用できる場合、最適化プログラムは次のことを行います。

1. 一時ファイル (SYS_TEMP_FILE) をライブラリー QRECOVERY に作成します。
2. SYS_TEMP_FILE を充てんする要求を一時結果書き込み機能に送ります。
3. 最終 QUERY のオープンを続けて終了します (一時結果書き込み機能が一時ファイルを充てんする間)。
4. 最終 QUERY がオープンされたら、一時結果書き込み機能が一時ファイルの充てんを終了するまで待ってから、呼び出し側に制御を戻します。

関連概念

60 ページの『DB2 マルチシステムでの QUERY 属性の変更コマンドへの変更』
QUERY 属性の変更コマンドには、分散 QUERY に適用できる 2 つのパラメーターがあります。

一時結果書き込み機能ジョブ: DB2 マルチシステムでの利点

要求の処理で一時結果書き込み機能ジョブを使用した場合の利点は、一時結果書き込み機能が、メイン・ジョブが QUERY の別のステップを処理しているのと同時に (並行して) その要求を処理できることです。

一時結果書き込み機能を使用した場合のパフォーマンスの利点は、以下のとおりです。

- 一時結果書き込み機能は、その QUERY ステップを完了するうえで完全対称マルチプロセッシング (SMP) 並列サポートを使用することができます。これに対して、メイン・ジョブは、簡単に並列処理を行うことのない、より複雑なステップ (QUERY 最適化、索引分析など) を続けることができます。
- 分散ファイル処理には通信対話が必要であるため、通常、送受信に費やすかなりの待機時間を一時結果書き込み機能に負荷分担させて、メイン・ジョブに他の作業を実行させることができます。

一時結果書き込み機能ジョブ: DB2 マルチシステムでの不利な点

一時結果書き込み機能には、QUERY での効用を判別するときに考慮する必要がある不利な点もあります。

以下のような不利な点があります。

- 一時結果書き込み機能は独立したジョブです。このため、メイン・ジョブとの矛盾が生じる可能性があります。以下に例を示します。
  - メイン・ジョブが、ファイルをロックする場合があります。この場合、一時結果書き込み機能はファイルにアクセスできず、その QUERY ステップを完了できません。
  - メイン・ジョブは、コミットメント制御のもとで分散ファイルを作成して、作成をまだコミットしていない場合があります。この場合、一時結果書き込み機能はファイルにアクセスすることができません。
- 一時結果書き込み機能は、メイン・ジョブと同じ方法で処理できない状態に陥る場合があります。たとえば、照会メッセージが通知された場合、一時結果書き込み機能が取り消されることがあるのに対して、メイン・ジョブはこのメッセージを無視して処理を続けることができます。
- 一時結果書き込み機能は、システム上のすべてのジョブによって共用されます。いくつかのジョブが一時結果書き込み機能に対する要求を持つ場合、これらの要求は、書き込み機能がその処理を試みる間待機します。

  注: システムは、3 つのアクティブな一時結果書き込み機能ジョブのペアの状態で出荷されます。
- QUERY を分析しようとすると (たとえば、デバッグ・メッセージなど)、一時結果書き込み機能が関与する場合、処理が複雑になるおそれがあります (これは、QUERY のステップが独立したジョブとして実行されるためです)。

注: システムは、一時結果書き込み機能をコミットメント制御 *CS または *ALL のもとで実行される QUERY に使用することを認めません。これは、メイン・ジョブがファイル内のレコードをロックして、一時書き込み機能がこれらのレコードからロックされて、終了できないようにする可能性があるためです。

DB2 マルチシステムでの一時結果書き込み機能の制御

デフォルトでは、QUERY は一時結果書き込み機能を使用しません。ただし、一時結果書き込み機能の使用は、QUERY 属性の変更 (CHGQRYA) コマンドを使用して使用可能にすることができます。

CHGQRYA コマンドの非同期ジョブの使用 (ASYNCJ) パラメーターは、一時結果書き込み機能の使用を制御するために使用されます。ASYNCJ パラメーターには、次のオプションがあります。

- *DIST または *ANY を使用すると、一時結果書き込み機能ジョブを、分散ファイルに関与する QUERY に使用することができます。
- *LOCAL または *NONE を使用すると、一時結果書き込み機能が分散ファイルに関与する QUERY に使用されるのを防止することができます。

DB2 マルチシステムでの最適化プログラムのメッセージ

i5/OS 分散 QUERY 最適化プログラムを使用すると、ジョブがデバッグ・モードにあるときに、現在の QUERY 処理に関する通知メッセージを得ることができます。

これらのメッセージは、分散 QUERY の処理方法を示すもので、既存の最適化プログラムのメッセージに追加されたものです。これらのメッセージは、QUERY ファイルのオープン (OPNQRYF) コマンド、DB2 UDB Query Manager and SQL Development Kit、対話式 SQL、組み込み SQL、および高水準言語 (HLL) の場合に表示されます。すべてのメッセージは、ジョブ・ログに表示されます。ユーザーに要求されるのは、ジョブをデバッグ・モードにしておくことだけです。

分散 QUERY のパフォーマンスは、データベース・マネージャーによってジョブ・ログに入れられた通知メッセージを使用して評価することができます。データベース・マネージャーは、次の分散メッセージのいずれか、または既存の最適化プログラム・メッセージを必要に応じて送信することができます。アンパーサンド変数 *(&1、&X)* は、次のメッセージがジョブ・ログに表示されるときのオブジェクト名または別の置換値のいずれかを含む置換変数です。

- CPI4341 分散 QUERY を実行中です。
- CPI4342 QUERY の分散結合を実行中です。
- CPI4343 分散 QUERY ステップの *&2* の *&1* に対する最適化プログラムのデバッグ・メッセージ。
- CPI4345 QUERY に作成された一時分散結果ファイル *&4*。

これらのメッセージは、分散 QUERY がどのように実行されるかに関するフィードバックを提供するとともに、場合によっては、QUERY の実行速度を高めるための手助けとして行える改善点を示すこともあります。以下のメッセージの原因とユーザーの処置を、この段落で示します。実際のメッセージ・ヘルプは、これよりも詳しく、各メッセージの意味と応答を判別するときに使用する必要があります。

CPI4341

分散 QUERY を実行中です。

このメッセージは、単一分散ファイルが QUERY されて、複数のステップで処理されていないことを示します。このメッセージは、QUERY が実行されたファイルのノードをリスト表示します。

CPI4342

QUERY の分散結合を実行中です。

このメッセージは、分散結合が起こったことを示します。このメッセージは、結合されたファイルだけでなく、結合が実行されたノードもリスト表示します。

CPI4343

分散 QUERY ステップの *&2* の *&1* に対する最適化プログラムのデバッグ・メッセージ。

このメッセージは、分散 QUERY が複数のステップで処理されて、現在のステップ番号をリスト表示することを示します。このメッセージに続いて、そのステップに関するすべての最適化プログラム・メッセージが示されます。

CPI4345

QUERY に作成された一時分散結果ファイル &4。

このメッセージは、一時分散結果ファイルが作成されたことを示し、その一時ファイルがなぜ必要であったかを示す理由コードをリストします。このメッセージには、ファイルを作成するために使用された区分化キーと、一時ファイルが作成されたノードも示されます。

次の例は、分散 QUERY が処理される方法を判別するために生成される、分散最適化プログラム・メッセージを調べる方法を示しています。この例では、分散ファイル EMPLOYEE と DEPARTMENT を使用しています。

```
SQL:      SELECT A.EMPNO, B.MGRNO, C.MGRNO, D.EMPNO
             FROM   EMPLOYEE A, DEPARTMENT B, DEPARTMENT C, EMPLOYEE D
             WHERE  A.EMPNO=B.MGRNO
                          AND  B.ADMRDEPT=C.DEPTNO
                          AND  C.DEPTNO=D.WORKDEPT

OPNQRYF:  OPNQRYF FILE((EMPLOYEE) (DEPARTMENT) (DEPARTMENT) (EMPLOYEE))
                  FORMAT(JFMT)
                  JFLD((1/EMNO 2/MGRNO *EQ)
                       (2/ADMRDEPT 3/DEPTNO)
                       (3/DEPTNO 4/WORKDEPT))
```

分散最適化プログラム・メッセージの次のリストが生成されます。

- CPI4343 分散 QUERY ステップの *&4* の *&1* に対する最適化プログラムのデバッグ・メッセージは、次のようになります。
  - CPI4345 QUERY に作成された一時分散結果ファイル *QQTDF0001。

    ファイル B は、一時ファイル *QQTDF0001 に指定されています。
- CPI4343 分散 QUERY ステップの *&4* の *&2* に対する最適化プログラムのデバッグ・メッセージは、次のようになります。
  - CPI4342 QUERY の分散結合を実行中です。

    ファイル B、C および *QQTDF0001 が結合されました。これは、照合結合 (ファイル B と C の間) と指定結合 (ファイル *QQTDF0001 と) の組み合わせです。
  - CPI4345 QUERY に作成された一時分散結果ファイル *QQTDF0002。

    一時分散ファイル *QQTDF0002 が、ファイル B、C、および *QQTDF0001 を結合した結果を含むために作成されました。このファイルは指定されています。
- CPI4343 分散 QUERY ステップの *&4* の *&3* に対する最適化プログラムのデバッグ・メッセージは、次のようになります。
  - CPI4345 QUERY に作成された一時分散結果ファイル *QQTDF0003。

    ファイル A は、一時ファイル *QQTDF0003 に指定されています。
- CPI4343 分散 QUERY ステップの &4 の &4 に対する最適化プログラムのデバッグ・メッセージは、次のようになります。
  - CPI4342 QUERY の分散結合を実行中です。

    ファイル *QQTDF0002 と *QQTDF0003 が結合されました。これは、両方のファイルが指定されてから結合が発生したため、再区分化結合でした。

パフォーマンスのために QUERY を調整するときに使用できる追加ツールには、SQL プログラムとパッケージに適用される SQL 情報の印刷 (PRTSQLINF) と、QUERY 属性の変更 (CHGQRYA) の各 CL コマンドがあります。

DB2 マルチシステムでの QUERY 属性の変更コマンドへの変更

QUERY 属性の変更コマンドには、分散 QUERY に適用できる 2 つのパラメーターがあります。

ASYNCJ (非同期ジョブ使用) と APYRMT (リモート適用) という 2 つのパラメーターがあります。

注: 他のパラメーターとは違い、ASYNCJ と APYRMT にはシステム値がありません。デフォルト値以外の値が必要な場合、その値は各ジョブで変更する必要があります。

関連資料

57 ページの『DB2 マルチシステムでの一時結果書き込み機能』
一時結果書き込み機能は、常に活動状態にあるシステム開始ジョブです。

DB2 マルチシステムでの非同期ジョブ使用パラメーター

非同期ジョブ使用 (ASYNCJ) パラメーターを使用すると、一時結果書き込み機能の使用を制御することができます。

ASYNCJ パラメーターには次のオプションがあります。

- *ANY - これを使用すると、一時結果書き込み機能ジョブを、分散ファイルを必要とするデータベース QUERY に使用することができます。
- *DIST - これを使用すると、一時結果書き込み機能ジョブを、分散ファイルを必要とするデータベース QUERY に使用することができます。
- *LOCAL - これを使用すると、一時結果書き込み機能ジョブを、ローカル・ファイルの QUERY にのみ使用することができます。このオプションは使用できますが、一時結果書き込み機能をローカル QUERY 処理に使用するためのシステム・サポートはありません。*LOCAL が、一時結果書き込み機能を分散 QUERY で使用不可にするために追加されていますが、通信を非同期で実行することもできます。
- *NONE - 一時結果書き込み機能をまったく使用しません。また、分散処理が実行される場合、通信は同期的に実行されます。このことは、QUERY を分析するときに非常に役立ちます。これは、リモート・システムからの QUERY デバッグ・メッセージをローカル・システムに戻すことができるからです。

次の例は、分散ファイル処理で非同期ジョブ使用を使用不可にする方法を示しています。

```
CHGQRYA ASYNCJ(*LOCAL)
```

このコマンドは、分散ファイルに関与する QUERY に非同期ジョブが使用されないようにするものです。

次の例は、非同期ジョブ使用を完全に使用不可にする方法を示しています。

```
CHGQRYA ASYNCJ(*NONE)
```

このコマンドは、すべての QUERY に非同期ジョブが使用されないようにするものです。また、分散ファイルに関与する QUERY では、リモート・システムへの通信が同期形式で実行されます。

次の例は、CHGQRYA コマンドをデバッグ開始 (STRDBG) コマンドと組み合わせて使用して、分散 QUERY を分析する方法を示しています。

```
STRDBG UPDPROD(*YES)
CHGQRYA ASYNCJ(*NONE)
STRSQL
       SELECT COUNT(*) FROM EMPLOYEE A
```

次のデバッグ・メッセージは、ジョブ・ログに入れられます。

```
        Current connection is to relational database SYSA.
        DDM job started.
        Optimizer debug messages for distributed query step 1 of 2 follow:
        Temporary distributed result file *QQTDF0001 built for query.
        Following messages created on target system SYSB.
        Arrival sequence access was used for file EMPLOYEE.
        Arrival sequence access was used for file EMPLOYEE.
```

```
Optimizer debug messages for distributed query step 2 of 2 follow:
Arrival sequence access was used for file EMPLOYEE.
ODP created.
Blocking used for query.
```

DB2 マルチシステムでのリモート適用パラメーター

リモート適用 (APYRMT) パラメーターを使用すると、他の CHGQRYA オプションを、分散 QUERY 要求の処理で使用される関連のリモート・システム・ジョブに適用するかどうかを指定することができます。

APYRMT パラメーターには次のオプションがあります。

- *YES - CHGQRYA オプションをリモート・ジョブに適用します。これには、リモート・ジョブが CHGQRYA コマンドを使用するための権限が必要です。権限がないと、リモート・ジョブにエラーが通知されます。
- *NO - CHGQRYA オプションをローカルにのみ適用します。

注: APYRMT パラメーターの QUERY 属性が *YES で、リモート・システム・ジョブの適用に関して QAQQINI ファイルに *NO がある場合は、APYRMT パラメーターの QUERY 属性は *NO でオーバーライドされます。

次の例は、CHGQRYA オプションがリモートに適用されないようにするものです。

```
CHGQRYA DEGREE(*NONE) APYRMT(*NO)
```

この場合、SMP (symmetric multiprocessing) 並列サポートがコーディネーター・ノードで防止されますが、リモート・システムは独自の並列度を選択することができます。

これらのパラメーターだけでなく、時間制限の QUERY (QRYTIMLMT) がどのように作動するかも注意する必要があります。パラメーターに指定された時間制限は、分散 QUERY の各ステップ (ローカルおよびリモート) に適用されます。これは、QUERY 全体には適用されません。したがって、ある QUERY ステップが時間制限に到達しても、別のステップは問題なく続けられるという場合もあります。時間制限オプションは非常に役に立ちますが、分散 QUERY では注意して使用する必要があります。

パフォーマンスの考慮事項の要約

分散ファイルを使用する QUERY を開発する際に、以下のパフォーマンス要因を考慮する必要があります。

1. OPNQRYF コマンドと QUERY API (QQQQRY) の場合、ALWCPYDTA(*OPTIMIZE) を指定すると、各ノードで索引または分類を選択して、指定の順序に対応することができます。
2. OPNQRYF コマンドおよび QUERY API (QQQQRY) の場合、ALWCPYDTA(*YES) または ALWCPYDTA(*NO) を指定すると、各ノードは、強制的に、指定の順序フィールドに合致した索引を使用させられます。これは、最適化プログラムが非分散ファイルの順序を処理する場合よりも高い拘束性があります。
3. ORDER BY 文節を DISTINCT 選択に追加すると、要求システムで最終分類が不要になるため、レコードを戻す速度を上げることができます。
4. グループ化フィールドに区分化キーのフィールドすべてを含めると、通常、1 ステップのグループ化が起こります。このパフォーマンスは、2 ステップのグループ化よりも改善されます。
5. 区分化キーのすべてのフィールドを結合基準に含めると、通常、照合分散結合が起こります。
6. 区分化キーのすべてのフィールドを、分離可能な等しいレコード選択に含めると、通常、1 つのノードのみで処理される QUERY が起こります。

7. 次のスカラー関数のいずれかを、分離可能な等しいレコード選択に含めると、通常、QUERY は 1 つのノードのみで処理されます。
   - NODENAME
   - NODENUMBER
   - PARTITION

DB2 マルチシステムの関連情報

他の Information Center のトピック・コレクションには、DB2 マルチシステムのトピック・コレクションに関連した情報が記載されています。

- 「Application programming interfaces」は、熟練したプログラマーを対象として、一部のオペレーティング・システム機能に対してアプリケーション・プログラミング・インターフェース (API) を使用する方法を示しています。
- 「Backup, Recovery and Media Services (BRMS)」には、次のような設定と管理に関する情報が示されています。

  – ジャーナル処理、アクセス・パス保護、およびコミットメント制御

  – ユーザー補助記憶域プール (ASP)

  – ディスク保護 (デバイス・パリティー、ミラー化、およびチェックサム)

  このトピック・コレクションでは、バックアップ・メディアおよび保管・復元操作に関するパフォーマンス情報も提供しています。また、活動状態の間の保存サポートの使用、保存と別のリリースへの復元、およびプログラミングのヒントと技法などの拡張バックアップおよび回復のトピックについても説明しています。
- 制御言語 (CL) はすべての CL コマンドの概要を示して、それらをコーディングするために必要な構文の規則について説明しています。
- 「データベース・プログラミング」では、システム上でデータベース・ファイルを作成、記述、および更新する方法も含めて、i5/OS データベース編成の詳しい説明を行っています。
- 「分散データベース・プログラミング」には、分散リレーショナル・データベース・アーキテクチャー (DRDA) を使用する分散リレーショナル・データベースにおいて System i 製品を準備し、管理する方法が記載されています。この資料では、類似のシステム環境における複数のシステムで、分散リレーショナル・データベースを計画、設定、プログラミング、管理、および操作する方法について説明しています。
- 「i5/OS および関連ソフトウェアのインストール、アップグレード、または削除」は、計画情報を記載し、オペレーティング・システムとライセンス・プログラムをインストールする手順をステップバイステップで説明しています。
- 「OptiConnect」は、光ファイバー・ケーブルを使用して複数のシステムと接続できる OptiConnect サポートについて説明しています。OptiConnect を使用すると、システム間データベースへのアクセス速度が向上するだけでなく、別のシステムに作業の負荷を分担させることができます。また、構成、導入、および操作に関する情報も示されています。
- 「SQL プログラミング」には、DB2 QUERY マネージャーおよび SQL 開発キット・ライセンス・プログラムの使用方法を示してあります。このトピック・コレクションは、データベース・ライブラリー内のデータにアクセスする方法と、組み込み SQL ステートメントを含むアプリケーション・プログラムを作成、実行、およびテストする方法を示しています。また、このトピック・コレクションは、SQL ステートメントの例および対話式 SQL 機能の説明を記載し、SQL ステートメントを COBOL、ILE

COBOL、PL/I、ILE C、FORTRAN/400、RPG、ILE RPG、および REXX™ で使用する際の共通概念と規則についても説明しています。「SQL プログラミング」および「データベース・プログラミング」では、DB2 UDB Symmetric Multiprocessing についても説明しています。

- 「DB2 for i5/OS SQL 解説書」には、DB2 SQL ステートメントの使用方法に関する説明と、これらのステートメントの正しい使用に関する詳細が記載されています。ステートメントの例には、構文図、パラメーター、および定義が示されています。 SQL 限界のリストと SQL 連絡域 (SQLCA) および SQL 記述子域 (SQLDA) の説明も示されています。

関連資料

1 ページの『DB2 マルチシステムの PDF ファイル』
この情報の PDF ファイルを表示または印刷できます。

コードに関するライセンス情報および特記事項

IBM は、お客様に、すべてのプログラム・コードのサンプルを使用することができる非独占的な著作使用権を許諾します。お客様は、このサンプル・コードから、お客様独自の特別のニーズに合わせた類似のプログラムを作成することができます。

強行法規で除外を禁止されている場合を除き、IBM、そのプログラム開発者、および供給者は「プログラム」および「プログラム」に対する技術的サポートがある場合にはその技術的サポートについて、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。

いかなる場合においても、IBM および IBM のサプライヤーならびに IBM ビジネス・パートナーは、その予見の有無を問わず発生した以下のものについて賠償責任を負いません。

1. データの喪失、または損傷。
2. 直接損害、特別損害、付随的損害、間接損害、または経済上の結果的損害
3. 逸失した利益、ビジネス上の収益、あるいは節約すべかりし費用

国または地域によっては、法律の強行規定により、上記の責任の制限が適用されない場合があります。

付録. 特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒242-8502
神奈川県大和市下鶴間1623番14号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム（本プログラムを含む）との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、IBM 機械コードのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。サンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。 IBM は、このサンプル・プログラムの使用から生ずるいかなる損害に対しても、責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _年を入れる_. All rights reserved.

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

本書「DB2 マルチシステム」には、プログラムを作成するユーザーが IBM i のサービスを使用するためのプログラミング・インターフェースが記述されています。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

以下は、IBM Corporation の商標です。

DB2
Distributed Relational Database Architecture
DRDA
i5/OS
IBM
IBM (ロゴ)
REXX
System i

Adobe、Adobe ロゴ、PostScript、PostScript ロゴは、Adobe Systems Incorporated の米国およびその他の国における登録商標または商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

使用条件

これらの資料は、以下の条件に同意していただける場合に限りご使用いただけます。

個人使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

商業的使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。 ただし、IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態で提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。

Printed in Japan

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21