DB2 Everyplace para Lotus Expeditor 6.1.1

IBM

Versión 9 Release 1.1

Guía de desarrollo de aplicaciones

DB2® Everyplace para Lotus Expeditor 6.1.1

IBM

Versión 9 Release 1.1

Guía de desarrollo de aplicaciones

Contenido

IBM DB2 Everyplace, Guía de aplicaciones y desarrollo Versión 9.1.1

Visión general del producto

DB2 Everyplace forma parte de la solución IBM On Demand Business para sincronizar datos entre dispositivos portátiles y servidores de bases de datos corporativos.

Mediante DB2 Everyplace, los profesionales que se desplazan con frecuencia (tales como vendedores, inspectores, auditores, técnicos de mantenimiento, médicos, agentes inmobiliarios y tasadores de seguros) pueden acceder a datos vitales necesarios mientras están fuera de su centro de trabajo. Las empresas pueden transferir sus datos corporativos a dispositivos portátiles. Mediante la base de datos portátil DB2 Everyplace, puede acceder a bases de datos situadas en dispositivos portátiles y realizar actualizaciones en ellas. Mediante el DB2 Everyplace Sync Server y el Sync Client, puede sincronizar datos de dispositivos portátiles con otras fuentes de datos de la empresa.

Novedades de la Versión 9.1.1

DB2 Everyplace Versión 9.1.1 ofrece nuevas características que satisfacen las necesidades de su empresa. Este tema presenta las características y mejoras de este release del producto.

Nuevos productos soportados

- Soporte para el sistema operativo Windows Vista

  Windows Vista se ha probado con la base de datos DB2 Everyplace y DB2 Everyplace Sync Client. Puede instalar estos dos componentes de DB2 Everyplace en Windows Vista. DB2 Everyplace Sync Server no está soportado en este sistema operativo.
- Soporte para la base de datos Apache Derby

  DB2 Everyplace ahora soporta Apache Derby como base de datos de cliente. Puede utilizar la base de datos Derby en lugar de la base de datos DB2 Everyplace para sincronizar datos. Se ha añadido información sobre la correlación de tipos de datos entre Derby y las diversas bases de datos fuente. Para más información, consulte “Correlaciones de tipos de datos entre DB2 Everyplace y fuentes de datos” en la página 45.

Sincronización

- Soporte para Estándar de cifrado de datos triple (triple DES)

  Con el soporte para triple DES, DB2 Everyplace ofrece mayor seguridad durante el transporte de datos.
- Soporte para controlador JDBC de Tipo 4 y controlador JDBC de Tipo 2 Universal

  DB2 Everyplace ahora permite el acceso a DB2 como una base de datos fuente a través del controlador JDBC de Tipo 4 y el controlador JDBC de Tipo 2 Universal.
- Soporte para la sincronización de tipos de datos BLOB

  Ahora puede sincronizar tipos de datos BLOB entre dispositivos y bases de datos fuente. Los dispositivos Symbian no están soportados para esta característica; DB2 para z/OS como base de datos fuente tampoco está soportado.

Base de datos y desarrollo de aplicaciones

- Rendimiento de consultas mejorado
- Soporte para Estándar de cifrado de datos triple (triple DES)

  Con el soporte para triple DES, DB2 Everyplace ofrece mayor seguridad para las tablas de dispositivos portátiles.
- Nueva tabla base del catálogo del sistema DB2eSYSINDEXES

  Esta nueva tabla base del catálogo del sistema almacena la información sobre índices de la base de datos. Ahora puede ver información detallada sobre índices utilizando la sentencia SELECT. Al

desarrollar aplicaciones, puede utilizar la función de DB2 CLI SQLStatistics y el método JDBC DatabaseMetaData.getIndexInfo para recuperar la información. Para obtener más información, consulte “Visión general de las tablas de las bases de datos portátiles DB2 Everyplace” en la página 22 y “Interfaz DatabaseMetaData” en la página 68.

- Nuevo atributo de conexión SQL_ATTR_FORCE_DISCONNECT

  Puede detener operaciones de base de datos de larga duración utilizando este nuevo atributo de conexión. Para obtener más información sobre la utilización del nuevo atributo, consulte “Detención de operaciones de base de datos de larga duración” en la página 27.

Caso práctico de ejemplo de DB2 Everyplace

DB2 Everyplace puede aumentar la productividad y eficiencia del personal móvil de una empresa. En este ejemplo, un tasador de seguros utiliza un dispositivo portátil donde se ejecuta una aplicación DB2 Everyplace.

Los tasadores de seguros son los encargados de inspeccionar los bienes dañados de los clientes que presentan una reclamación. En la mayoría de las empresas, el tasador visita los bienes del reclamante, rellena formularios para validar o rechazar la reclamación y evalúa el importe de los daños por los que se debe indemnizar al reclamante. Más adelante, cuando el tasador regresa a su oficina, la información de los formularios se entra manualmente en el sistema informático de la empresa, lo cual es un proceso tedioso y caro.

Este proceso se puede simplificar considerablemente proporcionando a los tasadores un dispositivo portátil donde se ejecuta una aplicación DB2 Everyplace. Mediante su dispositivo portátil, el tasador puede consultar, dondequiera que esté, su plan de inspecciones, su ruta de trabajo y la información de la póliza de seguro del reclamante. El tasador también puede cumplimentar el formulario de tasación en el dispositivo portátil. A continuación, el tasador puede sincronizar los datos de su dispositivo portátil con el sistema informático de la empresa, mediante la transferencia de los nuevos datos del formulario de tasación a la base de datos corporativa de la empresa. Si el tasador necesita información in situ, puede sincronizar inmediatamente los datos de su dispositivo portátil con el sistema informático de la empresa mediante un módem o conexión inalámbrica.

De esta forma, el proceso de evaluar reclamaciones puede prescindir totalmente del papel como soporte de información, lo cual supone un gran ahorro de costes para la empresa de seguros. Además, las reclamaciones se liquidan con más rapidez, pues el tasador tiene acceso inmediato a las bases de datos corporativas de la empresa.

Componentes de la solución DB2 Everyplace

La solución DB2 Everyplace para la sincronización de datos móviles incluye estos componentes: la base de datos móvil DB2 Everyplace, DB2 Everyplace Sync Server y DB2 Everyplace Sync Client.

Base de datos móvil DB2 Everyplace

El motor de la base de datos móvil DB2 Everyplace se ejecuta en un dispositivo portátil y guarda una copia local de datos procedentes de un sistema fuente. Los usuarios pueden utilizar el dispositivo portátil para acceder a estos datos y modificarlos. La base de datos móvil DB2 Everyplace se incluye con DB2 Everyplace Database Edition, DB2 Everyplace Enterprise Edition y la característica Mobility on Demand de DB2.

La base de datos móvil DB2 Everyplace es una base de datos relacional que reside en un dispositivo móvil del usuario. El usuario puede interaccionar con la base de datos mediante métodos JDBC (Java Database Connectivity) u ODBC (Open Database Connectivity).

DB2 Everyplace Sync Server

El DB2 Everyplace Sync Server es un servlet que sincroniza datos y resuelve conflictos entre bases de datos integradas en dispositivos portátiles y una base de datos fuente. Cuando instala DB2 Everyplace, se instala el servlet de DB2 Everyplace Sync Server y un servidor de aplicaciones

integrado de funcionalidad limitada. Puede también configurar el DB2 Everyplace Sync Server para que se ejecute dentro de un servidor de aplicaciones autónomo, tal como WebSphere Application Server versión 6.

Puede administrar el DB2 Everyplace Sync Server mediante dos herramientas:

El Centro de administración de dispositivos portátiles
Esta herramienta gráfica le ayuda a gestionar y proporcionar servicios de sincronización a grupos de usuarios con necesidades similares de sincronización de datos.

La herramienta para realizar scripts de XML
La herramienta para realizar scripts de XML automatiza tareas que de otro se ejecutarían utilizando el Centro de administración de dispositivos portátiles. Puede también utilizar la herramienta para realizar scripts de XML para copiar o trasladar suscripciones, conjuntos de suscripción, usuarios, o grupos desde un servidor a otros varios servidores.

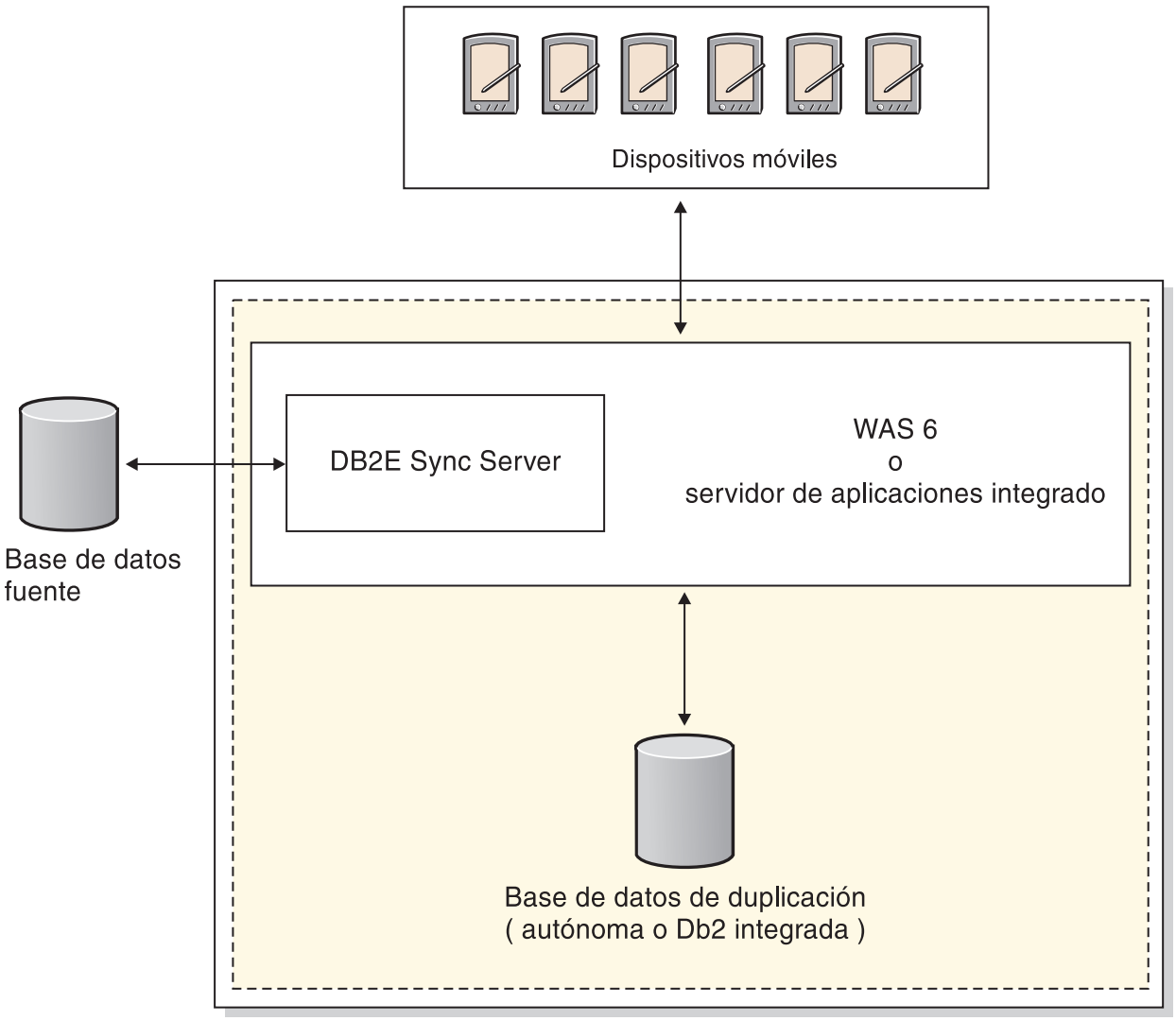
Base de datos duplicada DB2 Everyplace
La base de datos de duplicación DB2 Everyplace almacena los datos que desea sincronizar entre los dispositivos portátiles y las bases de datos corporativas. El DB2 Everyplace Sync Server utiliza la base de datos de réplica para resolver conflictos entre los dispositivos portátiles y para minimizar la carga de trabajo en los sistemas de bases de datos corporativos.

Si tiene una copia autónoma de DB2 Versión 9.1 en el sistema, cree o catalogue la base de datos de réplica en la instancia local de DB2 de DB2 Everyplace. Si no tiene una copia de DB2 Versión 9.1 en el sistema, DB2 Everyplace instala una versión integrada, restringida, de DB2 Versión 9.1 para que actúe como base de datos de réplica.

DB2 Everyplace Sync Client
El DB2 Everyplace Sync Client es un componente que las aplicaciones portátiles utilizan para sincronizar datos con el DB2 Everyplace Sync Server. Gestiona la sincronización bidireccional de datos relacionales corporativos con la base de datos portátil DB2 Everyplace. El DB2 Everyplace Sync Client también facilita la distribución y actualización de aplicaciones portátiles en los dispositivos portátiles, así como la ejecución de procedimientos almacenados que residen en una base de datos DB2.

Cómo se intercambian los datos entre DB2 Everyplace y los dispositivos portátiles

El DB2 Everyplace Sync Server define relaciones y derechos de acceso mediante objetos de duplicación de DB2 Versión 9.1, tales como usuarios, grupos, suscripciones y conjuntos de suscripción. Esta información así como una copia local de los datos fuente se guardan en la base de réplica DB2 Everyplace. El DB2 Everyplace Sync Server transfiere los datos a dispositivos portátiles mediante la interfaz TCP/IP proporcionada por el servidor de aplicaciones. Los dispositivos portátiles puede sincronizar datos mediante un canal cualquiera que sea compatible con TCP/IP, tal como una conexión USB directa o una conexión Internet.



Funciones de accesibilidad

Las funciones de accesibilidad ayudan a un usuario que tiene una discapacidad física, como, por ejemplo, movilidad restringida o visión limitada, a utilizar productos de software de forma satisfactoria.

El Programa de instalación, el Asistente de configuración y el Centro de administración de dispositivos portátiles son todos ellos accesibles e incluyen las funciones siguientes de accesibilidad:

- Utilización de todas las funciones mediante el teclado en lugar del ratón.
- Personalización del tamaño y color de los fonts.
- Recepción de indicaciones de alerta visuales o sonoras.
- Posibilidad de utilizar aplicaciones de accesibilidad que hacen uso de la API de accesibilidad de Java™.
- Se incluye documentación en un formato accesible.

Entrada de datos con teclado

Puede utilizar teclas o combinaciones de teclas para realizar operaciones que también se pueden realizar mediante un ratón. Puede acceder a menús dependientes del contexto desde la barra de menús, en lugar de pulsar el botón derecho del ratón.

Pantalla accesible

DB2 Everyplace tiene características que mejoran la interfaz de usuario y la accesibilidad para los usuarios con visión reducida. Estas mejoras de accesibilidad incluyen la posibilidad de personalizar propiedades de los fonts.

Atributos de fonts

Puede seleccionar el color, tamaño y font para el texto de los menús y ventanas de diálogo.

No dependencia del color

No es necesario que el usuario distinga los colores para utilizar cualquier función de este producto.

Posibilidad de utilizar colores de alto contraste

El Centro de administración de dispositivos portátiles se visualiza correctamente cuando se utiliza una combinación de colores de alto contraste.

No hay contenido que centellee o aparezca de modo intermitente

Ningún texto ni elemento de la interfaz gráfica de usuario parpadea ni aparece de modo intermitente durante el funcionamiento.

Indicaciones de alerta alternativas

Puede especificar si desea recibir alertas mediante señales sonoras o visuales.

Compatibilidad con las tecnologías de asistencia a discapacitados

La interfaz del Centro de administración de dispositivos portátiles es compatible con la API de Accesibilidad de Java, lo que permite el uso de lectores de pantalla y otras tecnologías de asistencia por personas con discapacidades.

Documentación accesible

El centro de información de DB2 Everyplace contiene documentación accesible sobre DB2 Everyplace.

Desarrollo

DB2 Everyplace permite el desarrollo de aplicaciones utilizando diversas interfaces API e idiomas.

En esta sección se tratan los temas siguientes.

Desarrollo de aplicaciones Java utilizando DB2 Everyplace

DB2 Everyplace proporciona una API Java que puede utilizar para desarrollar aplicaciones que hacen uso de la base de datos DB2 Everyplace.

Esta sección contiene los temas siguientes:

Desarrollo de aplicaciones Java para DB2 Everyplace

Para desarrollar una aplicación DB2 Everyplace utilizando Java, puede utilizar el SDK (Software Developer's Kit) de Java junto con la interfaz JDBC (Java Database Connectivity) de DB2 Everyplace para Java.

Para desarrollar aplicaciones Java para DB2 Everyplace:

1. Importe el paquete java.sql y cualquier otra clase de Java que sea necesaria.
2. Conéctese a la base de datos utilizando la clase DriverManager o bien la interfaz DataSource. Para obtener información detallada, vea la aplicación de ejemplo Java. La sintaxis del URL JDBC es jdbc:subprotocolo:subnombre. El subprotocolo de DB2 Everyplace es db2e.

   Restricción: DB2 Everyplace no permite la ejecución paralela de procesos en Symbian. Para poder acceder a una base de datos desde una segunda hebra, se debe cerrar el objeto Connection de la primera hebra para poder establecer la conexión en la segunda hebra. Varias hebras no pueden compartir un mismo objeto Connection.
3. Cree un objeto Statement.
4. Acceda a la base de datos (la lógica de la aplicación va aquí):
   a. Ejecute una sentencia de SQL utilizando el objeto Statement.
   b. Recupere datos del objeto ResultSet devuelto (si la sentencia de SQL que ha ejecutado es una consulta).
5. Libere recursos de base de datos y JDBC cerrando los objetos ResultSet, Statement, y Connection.

Visión general de los proveedores de sincronización de Java de DB2 Everyplace

En este tema se describe la API de Java Sync Client soportada por DB2 Everyplace. La API es un conjunto de bibliotecas que permiten que los programadores creen aplicaciones que sincronicen los datos entre DB2 Everyplace y bases de datos relacionales corporativas. Funciona junto con DB2 Everyplace Sync Server para simplificar la sincronización de archivos y datos relacionales. Sync Server permite la resolución de conflictos y gestiona el movimiento de datos hacia dispositivos portátiles y desde dispositivos portátiles.

La API de Sync Client Java consta de dos tipos de proveedores de sincronización:

- Proveedores de sincronización nativa de DB2Everyplace
- Proveedores de sincronización Java de DB2 Everyplace

Puede encontrar documentación de la API en el directorio <DSYPATH>\doc*idioma*\javadoc\SyncClientJavaAPI\, donde <DSYPATH> es el directorio donde está instalado DB2 Everyplace e *idioma* representa un idioma, por ejemplo, en_US. Los archivos de ejemplo proporcionan información sobre cómo crear aplicaciones Java en el dispositivo cliente basándose en esos proveedores.

Java DB2 Everyplace Sync Client para Derby

Java DB2 Everyplace Sync Client (Java Sync Client) para Derby permite crear aplicaciones que sincronizan suscripciones con una base de datos Derby. Java Sync Client para Derby es un conjunto de bibliotecas que trabajan con DB2 Everyplace Sync Server para simplificar la sincronización de datos relacionales entre bases de datos corporativas y un cliente Derby. El DB2 Everyplace Sync Server gestiona el intercambio de datos con el dispositivo.

Este tema incluye la siguiente información sobre Java Sync Client para Derby:

- Software necesario para ejecutar Java Sync Client para Derby
- Características no soportadas por Derby
- Estructura de directorios de Java Sync Client para Derby
- Definición de la variable de entorno CLASSPATH

Software necesario para ejecutar Java Sync Client para Derby

Para ejecutar Java Sync Client para Derby, necesita los productos de software siguientes:

- DB2 Everyplace Versión 8.2 o posterior
- Lotus Expeditor 6.1.1

Controladores Derby soportados

- Integrado versión 10

Características no soportadas por Java Sync Client para Derby

- El procedimiento remoto CALL.
- Suscripciones personalizadas en el servidor.
- Tiempos de espera excedidos de red, que se establecen con el valor config.createSyncService para la propiedad isync.timeout.
- Cifrado sobre cable (over-the-wire), que se habilita con el campo **Nivel de cifrado** en la pestaña Identificación de las ventanas Crear suscripción o Editar suscripción del Centro de administración de dispositivos portátiles (aunque puede seguir especificando valores de cifrado, estos no se aplicarán para clientes Derby).
- Cifrado por tabla, que se habilita con el campo **Cifrar en dispositivo** en la ventana **Definir suscripción de duplicación** del Centro de administración de dispositivos portátiles.

  Consejo: Para cifrar la base de datos Derby, añada la opción siguiente (que aparece en negrita) al URL de JDBC:

  jdbc:cloudspace:mydb;create=true; **dataEncryption=true;bootPassword=Db2jeveryPlace**
- Sincronización de bases de datos con SSL (Capa de sockets seguros).
- Si utiliza el entorno de ejecución Java JCLDesktop, los valores de isync.encoding ASCII y UTF-8, o UTF8, para la configuración de página de códigos de sincronización Derby no están soportados. Si no se establece la propiedad isync.encoding, se utilizará la página de códigos predeterminada para su plataforma de cliente, que funcionará según lo previsto. Si es necesario utilizar la codificación UTF-8 o ASCII, debe utilizar el entorno de ejecución Java 5 para este release.

Sincronización nativa de DB2 Everyplace

Los proveedores de sincronización nativa proporcionan la interfaz Java que invoca a las bibliotecas del cliente de sincronización nativa.

Nota: Los proveedores de sincronización nativa no permiten la utilización de la seguridad de hebra en este release. La aplicación se debe encargar de coordinar la sincronización de las hebras.

En DB2 Everyplace versión 9.1, solamente se puede utilizar un único tipo de proveedor de sincronización nativa de DB2 Everyplace:

Codificación de caracteres en aplicaciones Java

Las series de caracteres Java están representadas utilizando Unicode. Sin embargo, la aplicación puede especificar la codificación de caracteres de los datos de destino definiendo la propiedad isync.encoding en la API ISyncManager.getISyncService. Consulte la propiedad isync.encoding en ISyncManager.getISyncService para obtener más información sobre las codificaciones permitidas.

Codificación de caracteres en aplicaciones Java

Los programas Java utilizan texto Unicode internamente; sin embargo, los datos de tipo carácter de una tabla DB2 Everyplace pueden estar en un formato distinto de Unicode, dependiendo del sistema operativo y del lenguaje en el que se haya creado la tabla. Puede especificar dinámicamente el formato de codificación de los datos.

En el caso de los sistemas operativos Windows CE y Symbian OS, el controlador JDBC de DB2 Everyplace recupera texto de la base de datos e inserta texto en la base de datos utilizando el formato UTF-8. Para los demás sistemas operativos soportados, se utiliza la codificación de caracteres predeterminada del sistema. La codificación predeterminada se suele determinar mediante el atributo ″file.encoding″ de la propiedad del sistema Java.

Por ejemplo, en el sistema operativo Windows, un usuario puede elegir utilizar una versión Unicode o no Unicode de la interfaz CLI; por tanto, en la misma máquina, una base de datos podría tener codificación de formato UTF-8 y una codificación de página de códigos local. Para permitir que una aplicación JDBC acceda a datos de ambas bases de datos, DB2 Everyplace proporciona una manera de que los usuarios indiquen dinámicamente el formato de codificación de datos que una aplicación debe utilizar.

El controlador JDBC de DB2 Everyplace convierte las series Java en bytes según el formato especificado por la aplicación. El formato especificado por la aplicación altera temporalmente la codificación de caracteres predeterminada del sistema operativo.

El usuario puede especificar dinámicamente el formato de codificación de datos de la aplicación por medio de la interfaz JDBC. Para ello:

1. Cree un objeto java.util.Properties.
   - Clave: DB2e_ENCODING
   - Valor: codificación de caracteres.

   Utilice el valor UTF-8 para especificar que DB2 Everyplace debe utilizar la codificación UTF-8, o utilice cualquier codificación de caracteres compatible con la JVM.
2. Utilice uno de los dos métodos siguientes para pasar el objeto java.util.Properties:
   - Para establecer conexión con un URL de base de datos determinado:

     Utilice el método estático Connection getConnection(String url, Properties info) de la clase DriverManager del paquete java.sql.
   - Para crear una conexión de base de datos con un URL determinado:

Utilice el método Connection connect(String url, Properties info) de la clase de interfaz Driver del paquete java.sql.

Desarrollo con JDBC

Este tema describe los principios básicos del desarrollo de aplicaciones JDBC que interaccionan con DB2 Everyplace.

Recuperación e inserción gradual de datos a través de JDBC

La interfaz JDBC permite la recuperación e inserción gradual de datos. La función InputStreams encapsula la lógica para fragmentar los datos en porciones. El comportamiento de la infraestructura de BLOB cambia al permitir que se invaliden los objetos Blob cuando se cierra el objeto de sentencia asociado o al recuperar la fila siguiente. Esto hace que sea necesario mantener el cursor en la misma fila mientras se trabaja con este objeto.

La sección siguiente contiene ejemplos de inserción y recuperación de un BLOB:

Inserción de un BLOB

```
// CREATE TABLE t1 (c1 INT PRIMARY KEY NOT NULL, c2 BLOB(5M));

            PreparedStatement pstmt

= conn.prepareStatement ("INSERT INTO t1 VALUES (?,?)");

            pstmt.setInt (1, 100);

            File fBlob = new File ( "image1.gif" );

            FileInputStream is = new FileInputStream ( fBlob );

      pstmt.setBinaryStream (2, is, (int) fBlob.length() );

            pstmt.execute();

      ...
```

Recuperación de un BLOB

```
Ejemplo para JDBC:  Recuperación de un BLOB

        // CREATE TABLE t1 (c1 INT PRIMARY KEY NOT NULL, c2 BLOB(5M));

            Statement stmt = conn.createStatement ();

            ResultSet rs= stmt.executeQuery("SELECT * FROM t1");

            while (rs.next()) {

            int val1 = rs.getInt(1);

            InputStream val2 = rs.getBinaryStream(2);

           ...

                }

            rs.close();

            ...
```

Establecimiento de atributos de sentencias JDBC

Para facilitar la migración de las aplicaciones JDBC, DB2 Everyplace permite establecer atributos de sentencia (tales como el control del bit indicador y la habilitación o inhabilitación de la reorganización de tablas) mediante propiedades de la conexión.

Los atributos se convierten en valores predeterminados para las instancias de java.sql.Statement, java.sql.PreparedStatement y java.sql.CallableStatement que se acaban de crear. Este método es una alternativa a la conversión de un objeto java.sql.Statement en un objeto com.ibm.db2e.jdbc.DB2eStatement que pueda invocar métodos para establecer estos atributos.

Ejemplos

Utilizando java.sql.DriverManager

```
Properties pt = new Properties();
pt.setProperty("ENABLE_REORG", "false");
pt.setProperty("ENABLE_DELETE_PHYSICAL_REMOVE" , "true");
pt.setProperty("ENABLE_DIRTY_BIT_SET_BY_APPLICATION" , "false");
pt.setProperty("ENABLE_READ_INCLUDE_MARKED_DELETE" , "true");
con = DriverManager.getConnection(url, pt);
```

Utilizando java.sql.Driver

```
Properties pt = new Properties();
pt.setProperty("ENABLE_REORG", "false");
pt.setProperty("ENABLE_DELETE_PHYSICAL_REMOVE" , "true");
pt.setProperty("ENABLE_DIRTY_BIT_SET_BY_APPLICATION" , "false");
pt.setProperty("ENABLE_READ_INCLUDE_MARKED_DELETE" , "true");
pt.setProperty("ENABLE_TABLE_CHECKSUM", "true");
con = Driver.getConnection(url, pt);
```

Utilizando javax.sql.DataSource

```
com.ibm.db2e.jdbc.DB2eDataSource ds =
   new com.ibm.db2e.jdbc.DB2eDataSource();
ds.setUrl(url);
ds.setReorg(false);
ds.setDeletePhysicalRemove(true);
ds.setDirtyBitSetByApplication(false);
ds.setReadIncludeMarkedDelete(true);
ds.setEnabledTableChecksum(true);
con = ds.getConnection();
```

Información de consulta de API

Interfaz java.sql.Driver

Método: Connection connect(String url, Properties info)

Descripción: Intenta establecer una conexión de base de datos con el URL indicado.

Tabla 1. Pares clave/valor para info

Clave	Valor
ENABLE_REORG	True o false. El valor predeterminado es true.
ENABLE_DELETE_PHYSICAL_REMOVE	True o false. El valor predeterminado es false.
ENABLE_DIRTY_BIT_SET_BY_APPLICATION	True o false. El valor predeterminado es false.
ENABLE_READ_INCLUDE_MARKED_DELETE	True o false. El valor predeterminado es false.
ENABLE_TABLE_CHECKSUM	True o false. El valor predeterminado es false.

Método: Connection connect(String url, java.util.Hashtable info)

Descripción: Método sobrecargado de DB2 Everyplace para las plataformas que no soportan java.util.Properties

Tabla 2. Pares clave/valor para info

Clave	Valor
ENABLE_REORG	True o false. El valor predeterminado es true.
ENABLE_DELETE_PHYSICAL_REMOVE	True o false. El valor predeterminado es false.
ENABLE_DIRTY_BIT_SET_BY_APPLICATION	True o false. El valor predeterminado es false.
ENABLE_READ_INCLUDE_MARKED_DELETE	True o false. El valor predeterminado es false.
ENABLE_TABLE_CHECKSUM	True o false. El valor predeterminado es false.

Interfaz javax.sql.DataSource

Tabla 3. Propiedades específicas de DB2 Everyplace para la interfaz DataSource

Nombre de propiedad	Tipo	Descripción
reorg	boolean	Habilita o inhabilita la reorganización de tablas.
deletePhysicalRemove	boolean	Habilita/inhabilita la eliminación física de registros.
dirtyBitSetByApplication	boolean	Permite/no permite que la aplicación establezca el bit indicador.
readIncludeMarkedDelete	boolean	Habilita/inhabilita la lectura de registros suprimidos de forma lógica.
isEnabledTableChecksum	boolean	Habilitar/inhabilitar sumas de comprobación para valores de base de datos.

Métodos:

```
void setReorg(boolean enable)
void setDeletePhysicalRemove(boolean enable)
void setDirtyBitSetByApplication(boolean enable)
void setReadIncludeMarkedDelete(boolean enable)
void setEnabledTableChecksum(boolean enable)

boolean isReorg()
boolean isDeletePhysicalRemove()
boolean isDirtyBitSetByApplication()
boolean isEnabledTableChecksum()
```

Establecimiento de la ubicación de JSR-169 para desarrollar aplicaciones Java

Si utiliza la máquina virtual IBM J9 para desarrollar aplicaciones Java, y su perfil J9 no incluye el paquete JSR-169, DB2 Everyplace incluye un paquete JSR-169 para su comodidad.

El paquete JSR-169 es necesario para desarrollar aplicaciones Java con DB2 Everyplace si utiliza una máquina virtual J9. La especificación JSR-169 define un paquete opcional JDBC para la implementación Java 2 Micro Edition (J2ME), Connected Device Configuration (CDC) Foundation Profile. El controlador JDBC de DB2 Everyplace implementa el conjunto de interfaces estándar que define la especificación JSR-169.

Para utilizar el paquete JSR-169 que se proporciona con DB2 Everyplace, especifique la ubicación del paquete realizando los pasos siguientes:

1. Cambie la variable del sistema CLASSPATH para que incluya el archivo jdbc.jar:

- <DSYPATH>\Clients\<*plataforma*>\database\jdbc\jdbc.jar donde *plataforma* es linux, neutrino, win32 o wince.
- <DSYPATH>\Clients\palmos\database\JDBC\midp20\
- <DSYPATH>\Clients\<*plataforma*>\database\armi\ donde *plataforma* es symbian7, symbian 7s o symbian9.
- <DSYPATH>\Clients\<*plataforma*>\database\wins\ donde *plataforma* es symbian7, symbian 7s o symbian9.

2. Si hay alguna excepción de seguridad J9, especifique la ubicación del archivo jdbc.jar en la CLASSPATH de arranque utilizando la opción -Xbootclasspath:.

SQL y procedimientos almacenados específicos de la plataforma

Este tema proporciona información que le ayudará a desarrollar aplicaciones utilizando SQL y procedimientos almacenados. También describe cómo utilizar el adaptador de consultas y procedimientos almacenados remotos.

Visión general de los marcadores de parámetros

Para las sentencias de SQL que deben ejecutarse muchas veces, a menudo es ventajoso preparar una vez la sentencia de SQL y reutilizar el plan de consulta utilizando marcadores de parámetros para sustituir los valores de entrada durante la ejecución.

En DB2 Everyplace, un marcador de parámetro se representa mediante un carácter ″?″ e indica el lugar en el va a sustituirse una variable de aplicación en una sentencia SQL. Se hace referencia a los marcadores de parámetros por un número y están numerados secuencialmente de izquierda a derecha, comenzando en el 1. Antes de emitir la sentencia de SQL, la aplicación debe vincular un área de almacenamiento variable a cada marcador de parámetros especificado en la sentencia de SQL. Además, las variables vinculadas deben ser un área de almacenamiento válida y deben contener valores de datos de entrada cuando la sentencia preparada se ejecuta para la base de datos.

El ejemplo siguiente muestra una sentencia de SQL que contiene dos marcadores de parámetros.

```
SELECT * FROM customers WHERE custid = ? AND lastname = ?
```

Ejemplos de utilización de marcadores de parámetros

DB2 Everyplace proporciona un variado conjunto de interfaces estándar, incluido JDBC para acceder a los datos de modo eficaz. Los fragmentos de código de ejemplo contenidos en este tema muestran el uso de una sentencia preparada con marcadores de parámetros para cada API de acceso a datos.

Tome en consideración el esquema de tabla siguiente para la tabla t1, en el que la columna c1 es la clave primaria para la tabla t1.

Tabla 4. Esquema de tabla de ejemplo

Nombre de la columna	Tipos de datos de DB2 Everyplace	Admite nulos
c1	INTEGER	falso
c2	SMALLINT	verdadero
c3	CHAR(20)	verdadero
c4	VARCHAR(20)	verdadero
c5	DECIMAL(8,2)	verdadero
c6	DATE	verdadero
c7	TIME	verdadero

Tabla 4. Esquema de tabla de ejemplo (continuación)

Nombre de la columna	Tipos de datos de DB2 Everyplace	Admite nulos
c8	TIMESTAMP	verdadero
c9	BLOB(30)	verdadero

Los ejemplos siguientes ilustran el modo de insertar una fila en la tabla t1 utilizando una sentencia preparada.

Ejemplo de JDBC

```
public static void parameterExample1() {

   String driver = "com.ibm.db2e.jdbc.DB2eDriver";
   String url    = "jdbc:db2e:mysample";
   Connection conn = null;
   PreparedStatement pstmt = null;

   try
   {
      Class.forName(driver);

      conn = DriverManager.getConnection(url);

      // preparar la sentencia
      pstmt = conn.prepareStatement("INSERT INTO t1 VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)");

      // asociar los parámetros de entrada
      pstmt.setInt(1, 1);
      pstmt.setShort(2, (short)2);
      pstmt.setString(3, "data1");
      pstmt.setString(4, "data2");
      pstmt.setBigDecimal(5, new java.math.BigDecimal("12.34"));
      pstmt.setDate(6, new java.sql.Date(System.currentTimeMillis() ) );
      pstmt.setTime(7,  new java.sql.Time(System.currentTimeMillis() ) );
      pstmt.setTimestamp (8,  new java.sql.Timestamp(System.currentTimeMillis() ) );
      pstmt.setBytes(9, new byte[] { (byte)'X', (byte)'Y', (byte)'Z' } );

      // ejecutar la sentencia
      pstmt.execute();

      pstmt.close();

      conn.close();
   }
   catch (SQLException sqlEx)
   {
      while(sqlEx != null)
      {
          System.out.println("SQLERROR: \n" + sqlEx.getErrorCode() +
             ", SQLState: " + sqlEx.getSQLState() +
             ", Message: " + sqlEx.getMessage() +
             ", Vendor: " + sqlEx.getErrorCode() );
         sqlEx = sqlEx.getNextException();
      }
   }
   catch (Exception ex)
   {
      ex.printStackTrace();
   }
}
```

Marcadores de parámetros soportados por DB2 Everyplace

DB2 Everyplace solamente puede utilizar marcadores de parámetros no tipificados, que se pueden utilizar en posiciones determinadas de una sentencia de SQL. Este tema muestra las restricciones existentes respecto a la utilización de marcadores de parámetros.

Un marcador de parámetro, que se representa mediante un símbolo de final de interrogación (?), es un espacio reservado en una sentencia de SQL cuyo valor se obtiene durante la ejecución de la sentencia. Una aplicación utiliza la función SQLBindParameter() para asociar marcadores de parámetros de enlace con variables de aplicación. Durante la ejecución de las funciones de DB2 CLI SQLExecute() y SQLExecDirect(), los valores de estas variables sustituyen a cada uno de los marcadores de parámetros respectivos. Durante el proceso puede tener lugar una conversión de datos.

Tabla 5. Restricciones sobre la utilización de los marcadores de parámetros

Ubicación de los marcadores de parámetros no tipificados	**Tipo de datos**
Expresión: Solo en una lista de selección	Error
Expresión: Ambos operandos de un operador aritmético	Error
Predicado: Operando de la parte izquierda de un predicado IN	Error
Predicado: Ambos operandos de un operador relacional	Error
Función: Operando de una función de agregación	Error

Adaptador de consultas y procedimientos almacenados remotos

DB2 Everyplace incluye un adaptador de consultas y procedimientos almacenados remotos. Este adaptador permite que la aplicación DB2 Everyplace utilice la arquitectura de DB2 Everyplace Sync Server para invocar un procedimiento almacenado situado en una fuente de datos remota.

Los resultados del procedimiento almacenado se devuelven directamente a la aplicación en el dispositivo. La llamada al procedimiento almacenado permite a una aplicación de DB2 Everyplace acceder directamente a datos de un servidor remoto sin necesidad de sincronización.

El adaptador de consultas y procedimientos almacenados remotos habilita algunas capacidades exclusivas del motor de base de datos DB2 Everyplace, tal como la capacidad de llamar de modo remoto a un procedimiento almacenado de DB2 Versión 9.1. Este tema describe los requisitos y técnicas para utilizar el adaptador de consultas y procedimientos almacenados remotos en una aplicación de DB2 Everyplace.

A partir de la versión 9.1, DB2 Everyplace es ahora compatible con IPv6. Puede utilizar el procedimiento almacenado remoto especificando un URL válido de IPv6 o IPv4. Por ejemplo, los formatos siguientes son válidos:

```
http://[::1]:9081/db2e/com.ibm.mobileservices.adapter.agent.AgentServlet?DB=mysample
http://[::1]:9081/db2e/agent?DB=mysample

http://127.0.0.1:9081/db2e/com.ibm.mobileservices.adapter.agent.AgentServlet?DB=mysample
http://127.0.0.1:9081/db2e/agent?DB=mysample
```

Tipos de datos soportados para procedimientos almacenados

DB2 Everyplace soporta la llamada a procedimientos almacenados de un servidor DB2 remoto mediante la interfaz JDBC. La aplicación cliente utiliza la sentencia CALL para ejecutar el procedimiento almacenado remoto. La sentencia CALL da nombre al procedimiento al que se debe llamar y especifica sus parámetros. Se soportan los siguientes tipos: INTEGER, SMALLINT, DECIMAL, CHAR, VARCHAR, DATE, TIME, TIMESTAMP y BLOB.

Utilización del adaptador de consultas y procedimientos almacenados remotos

El adaptador de consultas y procedimientos almacenados remotos da soporte a las plataformas cliente Windows de 32 bits (no Unicode) y Windows CE. El adaptador de consultas y procedimientos almacenados remotos requiere que los procedimientos almacenados se registren en DB2.

Restricciones

Conexiones múltiples
DB2 Everyplace permite conexiones múltiples con bases de datos con algunas limitaciones. La conexión remota utiliza la base de datos local (la última conexión antes de la conexión remota) para almacenar los archivos temporales. Si no existe una conexión local, se utilizará el directorio actual.

Descriptor de sentencia
Asigne sólo un descriptor de sentencia para la conexión remota.

En Palm OS
Puede ser necesario aumentar el tamaño de pila de la aplicación.

En los sistemas operativos Windows de 32 bits
Durante la ejecución, los archivos DLL de IBM DB2 Everyplace Sync Client se deben incluir en el directorio local o en la vía de acceso del sistema.

En un procedimiento almacenado de DB2 Versión 9.1
Cuando se utiliza un objeto binario grande (BLOB) como parámetro de entrada o de salida, los cuatro primeros bytes de los datos del BLOB se reservan para indicar la longitud.

Procedimientos almacenados
DB2 Everyplace puede trabajar con procedimientos almacenados de DB2 Versión 9.1 solamente en las plataformas Windows y UNIX.

Tamaño del mensaje
No utilice llamadas de procedimiento almacenado remoto para transferir grandes volúmenes de datos. En lugar, utilice la sincronización de DB2 Everyplace. El tamaño de cada mensaje debe ser menor que 32 KB.

El ejemplo siguiente muestra cómo crear un procedimiento almacenado, una suscripción al procedimiento almacenado y una aplicación de DB2 Everyplace para utilizar el procedimiento almacenado. Esta aplicación de ejemplo permite al usuario de un dispositivo portátil comprobar el saldo de una cuenta y transferir dinero entre una cuenta de ahorros y una cuenta corriente utilizando una llamada a un procedimiento almacenado remoto de DB2 Everyplace.

Creación de un procedimiento almacenado utilizando la aplicación de ejemplo:

Este ejemplo utiliza un procedimiento almacenado denominado `MYPROC()`. Este procedimiento utiliza cinco parámetros: Account Name (Nombre de cuenta), Option (Opción), Transfer Amount (Importe para la transferencia), Saving Balance (Saldo de la cuenta de ahorros) y Checking Balance (Saldo de la cuenta corriente). La siguiente lista identifica la finalidad de cada uno de los parámetros:

1. Account Name: Parámetro de entrada para identificar la cuenta.
2. Opcional: Parámetro de entrada para determinar qué hacer. Hay tres opciones:
   - Consultar el saldo
   - Transferir desde cuenta de ahorros a cuenta corriente
   - Transferir desde cuenta corriente a cuenta de ahorros
3. Transfer Amount: Parámetro de entrada correspondiente al importe que se debe transferir entre la cuenta corriente y la cuenta de ahorros
4. Saving Balance: Parámetro de salida que devuelve el saldo de la cuenta de ahorros
5. Checking Balance: Parámetro de salida que devuelve el saldo de la cuenta corriente

El código siguiente crea el procedimiento almacenado:

```
SQL_API_RC SQL_API_FN
myProc(char * szName, int * nCmd, int * nAmount, int * nSaving, int * nChecking)
{
 SQLHENV  henv;
 SQLHDBC  hdbc;
 SQLHSTMT hstmt;
 SQLRETURN rc;
 int   nRetSize;

 SQLCHAR  str1[]="select saving, checking from db2e.myaccount where name = ?";
 SQLCHAR  str2[]="update db2e.myaccount set saving=saving - ?,
          checking=checking + ? where name=?";
 SQLCHAR  str3[]="update db2e.myaccount set saving=saving + ?,
          checking=checking - ? where name=?";

 //*************************************************************
 //* Preparar conexión y sentencia
 //*************************************************************
 rc = SQLAllocHandle( SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);
 //checkerror
 rc = SQLAllocHandle( SQL_HANDLE_DBC, henv, &hdbc);
 //checkerror
 rc = SQLSetConnectAttr(hdbc, SQL_ATTR_AUTOCOMMIT, SQL_AUTOCOMMIT_OFF, SQL_NTS);
 //checkerror
     rc = SQLConnect(hdbc, NULL, SQL_NTS, NULL, SQL_NTS, NULL, SQL_NTS);
 //checkerror
 rc = SQLAllocHandle( SQL_HANDLE_STMT, hdbc, &hstmt);
 //checkerror

 //*************************************************************
 //* Actualizar cuenta
 //*************************************************************
 if ( *nCmd == 2 || *nCmd == 3 ){
  if ( *nCmd == 2 ){ //Transferir de cuenta de ahorros a cuenta corriente
   rc = SQLPrepare(hstmt, str2, SQL_NTS); //checkerror
  }
  if ( *nCmd == 3 ){ //Transferir de cuenta corriente a cuenta de ahorros
   rc = SQLPrepare(hstmt, str3, SQL_NTS); //checkerror
  }
  rc = SQLBindParameter(hstmt,
    1,
    SQL_PARAM_INPUT,
    SQL_C_LONG,
    SQL_INTEGER,
    0,
    0,
    (SQLPOINTER)nAmount,
    0,
    NULL ); //checkerror
  rc = SQLBindParameter(hstmt,
    2,
    SQL_PARAM_INPUT,
    SQL_C_LONG,
    SQL_INTEGER,
    0,
    0,
    (SQLPOINTER)nAmount,
    0,
    NULL ); //checkerror

  rc = SQLBindParameter(hstmt,
    3,
    SQL_PARAM_INPUT,
    SQL_C_CHAR,
    SQL_CHAR,
    0,
```

```
    0,
    (SQLPOINTER)szName,
    0,
    NULL ); //checkerror
  rc = SQLExecute(hstmt); //checkerror
 }

 //*************************************************************
 //* Recuperar saldo de la cuenta
 //*************************************************************
 rc = SQLPrepare(hstmt, str1, SQL_NTS); //checkerror
 rc = SQLBindParameter(hstmt,
   1,
   SQL_PARAM_INPUT,
   SQL_C_CHAR,
   SQL_CHAR,
   0,
   0,
   (SQLPOINTER)szName,
   0,
   NULL );//checkerror
 rc = SQLExecute(hstmt);//checkerror
 if ( rc == SQL_SUCCESS || rc == SQL_SUCCESS_WITH_INFO )
 {
  while ( (rc = SQLFetch(hstmt) ) == SQL_SUCCESS ){
   rc = SQLGetData( hstmt,
    (SQLSMALLINT)1,
    SQL_C_LONG,
    nSaving,
    sizeof(int) ,
    &nRetSize ) ; //checkerror
   rc = SQLGetData( hstmt,
    (SQLSMALLINT)2,
    SQL_C_LONG,
    nChecking,
    sizeof(int) ,
    &nRetSize ) ; //checkerror
  }
 }
 //*************************************************************
 //* Limpiar
 //*************************************************************
     rc = SQLEndTran( SQL_HANDLE_DBC, hdbc, SQL_COMMIT );
 SQLFreeHandle(SQL_HANDLE_STMT, hstmt);
 SQLDisconnect(hdbc);
 SQLFreeHandle(SQL_HANDLE_DBC, hdbc);
 SQLFreeHandle(SQL_HANDLE_ENV, henv);
 return (0);
}
```

En la plataforma Windows, después de crear el procedimiento almacenado en forma de biblioteca de enlace dinámico (mydll.dll), cópielo en el directorio \SQLLIB\function. A continuación, registre el procedimiento almacenado.

1. Abra una ventana de mandatos de DB2.
2. Conéctese a la base de datos MYSAMPLE utilizando el mandato siguiente:

   ```
   DB2 CONNECT TO MYSAMPLE
   ```

3. Registre el procedimiento almacenado utilizando un script denominado regscript.scr para configurar opciones. Para este script se utiliza el código siguiente:

   ```
   CREATE PROCEDURE db2e.MYPROC (IN szName CHAR(16),
                                 IN nCmd INTEGER,
                                 IN nAmount INTEGER,
                                 OUT nSaving INTEGER,
                                 OUT nChecking INTEGER )
   DYNAMIC RESULT SETS 1
   ```

```
LANGUAGE C
PARAMETER STYLE GENERAL
NO DBINFO
FENCED
MODIFIES SQL DATA
PROGRAM TYPE SUB
EXTERNAL NAME 'mydll!myProc'@
```

Para ejecutar el script, escriba el siguiente mandato: `db2 -td@ -vf regscript.scr`

El procedimiento almacenado db2e.MYPROC ya está configurado: A continuación, cree una suscripción utilizando el Centro de administración de dispositivos portátiles.

Creación de la suscripción personalizada para la aplicación de ejemplo:

Para crear una suscripción personalizada para la aplicación de ejemplo:

Ejecute los pasos siguientes en el Centro de administración de dispositivos portátiles.

1. Abra el Centro de administración de dispositivos portátiles.
2. Pulse con el botón derecho del ratón en la carpeta **Suscripciones** en el panel izquierdo de la ventana Centro de administración de dispositivos portátiles y seleccione **Crear → Suscripción personalizada**. Se abrirá la ventana Crear suscripción personalizada.
3. Escriba `subex` en el campo **Nombre**.
4. Pulse el botón **Iniciar personalizador**. Se abrirá la ventana Base de datos fuente.
   a. En el campo **ID de usuario**, escriba el ID de usuario de DB2 que tenga privilegios de acceso para la base de datos de destino.
   b. Escriba la contraseña del ID de usuario en los campos **Contraseña** y **Verificar contraseña**.
   c. En el campo **Otro**, escriba:
   `dbname=mysample;procname=db2e.MYPROC`
5. Pulse **Bien** para cerrar la ventana Base de datos fuente. Pulse **Bien** para cerrar la ventana Crear suscripción personalizada.

Después de crear la suscripción personalizada, cree un usuario, un grupo y un conjunto de suscripción.

Prueba del adaptador de consultas y procedimientos almacenados remotos:

Este ejemplo utiliza una aplicación de consola Windows para DB2 Everyplace para probar el adaptador de consultas y procedimientos almacenados remotos. La aplicación de ejemplo se denomina myclient.exe. Utiliza los tres parámetros siguientes:

1. Account Name (Nombre de cuenta): Identifica la cuenta a la que se debe acceder.
2. Option (Opción): Identifica la acción que se debe realizar. Las opciones son las siguientes:
   1: Check balance (Comprobar el saldo).
   2: Transfer from savings to checking (Transferir de cuenta de ahorros a cuenta corriente).
   3: Transfer from checking to savings (Transferir de cuenta corriente a cuenta de ahorros).
3. Amount (Importe): Importe que se debe transferir entre la cuenta corriente y la de ahorros.

Por ejemplo, para transferir 1000 dólares de la cuenta de ahorros a la cuenta corriente en la cuenta Michael, escriba el mandato siguiente: `myclient.exe Michael 2 1000`.

Suponiendo que Michael tenga 5000 dólares en cada cuenta antes de la transferencia, se devolverá la siguiente respuesta:

```
Saving = 4000
Checking = 6000
```

Restricciones para conjuntos de resultados:

Los conjuntos de resultados resultan útiles para recuperar datos de un procedimiento almacenado. Si una aplicación de cliente ejecuta un procedimiento almacenado que genera un conjunto de resultados, luego puede utilizar las funciones CLI o métodos JDBC normales como SQLFetch() y SQLGetData() para recuperar los datos. DB2 Everyplace no soporta múltiples conjuntos de resultados.

Desarrollo avanzado con DB2 Everyplace

Este tema describe el desarrollo de aplicaciones que hace uso de las funciones avanzadas de DB2 Everyplace.

Visión general de las tablas de las bases de datos portátiles DB2 Everyplace

Una base de datos portátil DB2 Everyplace comprende varias tablas de catálogo del sistema y tablas definidas por el usuario.

Cada una de las tablas se almacena en dos archivos: uno para los datos propiamente dichos y otro para los índices. Todos los índices se conservan en el mismo archivo de índices. A diferencia de DB2 Versión 9.1, las bases de datos portátiles DB2 Everyplace no tienen nombre y no se pueden catalogar ni descatalogar. Por lo tanto, el nombre de la base de datos se pasa por alto.

Una base de datos portátil DB2 Everyplace no es más que un conjunto de archivos que se pueden copiar o mover a otra ubicación. Una base de datos portátil DB2 Everyplace debe contener las siguientes tablas de catálogos del sistema:

- DB2eSYSTABLES
- DB2eSYSCOLUMNS
- DB2eSYSINDEXES
- DB2eSYSRELS
- DB2eSYSUSERS (esta tabla se crea si se utiliza cifrado local de los datos)

Las tablas de catálogos del sistema contienen metadatos sobre tablas definidas por el usuario. Por ejemplo, si elimina archivos de una tabla definida por el usuario sin suprimir la entrada correspondiente en las tablas de catálogos, ocasionará una incoherencia.

Para acceder a las tablas de catálogos de una consulta, deberá utilizar los identificadores delimitados. Por ejemplo, la consulta siguiente devolverá el valor 1 si existe la tabla T:

```
SELECT 1 FROM "DB2eSYSTABLES" WHERE TNAME = 'T'
```

Tablas base del catálogo del sistema de DB2 Everyplace

El gestor de bases de datos crea y mantiene un conjunto de tablas base del catálogo del sistema. Este apéndice contiene una descripción de cada tabla base del catálogo del sistema, incluyendo nombres de columnas y tipos de datos.

Todas las tablas base del catálogo del sistema las crea el gestor de bases de datos. Las tablas base del catálogo del sistema no se pueden crear ni eliminar explícitamente. Las tablas base del catálogo del sistema se actualizan durante el funcionamiento normal en respuesta a las sentencias SQL de definición de datos, rutinas de entorno y ciertos programas de utilidad. Los datos en las tablas base del catálogo del sistema están disponibles mediante programas de utilidad de búsqueda SQL normales. Las tablas base del catálogo del sistema no se pueden modificar utilizando mandatos de manipulación de datos de SQL normales. Para poder acceder a las tablas de catálogo del sistema es necesario utilizar un identificar delimitado.

Tabla 6. Tablas base del catálogo del sistema

Descripción	Tabla base del catálogo
tablas	"DB2eSYSTABLES"
columnas	"DB2eSYSCOLUMNS"
índices	"DB2eSYSINDEXES" en la página 24
restricciones referenciales	"DB2eSYSRELS" en la página 24
usuarios	"DB2eSYSUSERS" en la página 24

DB2eSYSTABLES

Esta tabla base del catálogo del sistema contiene una fila para cada tabla que se crea. Todas las tablas de catálogo tienen entradas en el catálogo DB2eSYSTABLES.

Tabla 7. Tabla base del catálogo del sistema DB2eSYSTABLES

Nombre de la columna	Tipo de datos	Admite nulos	Descripción
TNAME	VARCHAR (129)		Nombre de tabla
NUMCOLS	INTEGER (4)		Número de columnas
FLAGS	INTEGER (4)		(Sólo uso interno)
NUMKEY	INTEGER (4)		Número de columnas en la clave primaria
CHK	BLOB (32767)	Sí	Restricción de comprobación (sólo uso interno)
IDXINFO	BLOB (4096)	Sí	Índice (sólo uso interno)
NUMREFS	INTEGER (4)	Sí	Clave primaria y foránea (sólo uso interno)
F_ID	INTEGER (4)	Sí	(Sólo uso interno)
PD	BLOB (4096)	Sí	(Sólo uso interno)

DB2eSYSCOLUMNS

Esta tabla base del catálogo del sistema contiene una fila para cada columna que se define para una tabla.

Tabla 8. Tabla base del catálogo del sistema DB2eSYSCOLUMNS

Nombre de la columna	Tipo de datos	Admite nulos	Descripción
CNAME	VARCHAR (129)		Nombre de la columna
TNAME	VARCHAR (129)		Nombre de tabla
TYPE	INTEGER (4)		Tipo de datos
ATTR	INTEGER (4)		(Sólo uso interno)
LENGTH	INTEGER (4)		Longitud de la columna
POS	INTEGER (4)		Número de columna
FLAGS	INTEGER (4)		(Sólo uso interno)
KEYSEQ	INTEGER (4)		Posición ordinal de la columna en la clave primaria
SCALE	INTEGER (4)		Escala para columna decimal
DEF	VARCHAR (32767)	Sí	Valor predeterminado (uso interno)

DB2eSYSINDEXES

La tabla DB2eSYSINDEXES se crea durante la iniciación de la base de datos. Esta tabla base del catálogo del sistema contiene una fila para cada columna de cada índice que se crea.

Tabla 9. Tabla base del catálogo del sistema DB2eSYSINDEXES

Nombre de la columna	Tipo de datos	Admite nulos	Descripción
INAME	VARCHAR (129)		Nombre de índice
TNAME	VARCHAR (129)		Nombre de tabla
CNAME	VARCHAR (129)		Nombre de la columna
NONUNIQUE	SMALLINT (4)		El índice es exclusivo o no.
TYPE	SMALLINT (4)		Tipo de índice
ORDINALPOSITION	SMALLINT (4)		Posición ordinal de la columna en la tabla
ASCORDESC	CHAR (1)		El orden en que se clasifican los valores de las columnas que forman el índice, ascendente o descendente
FILEIDX	INTEGER (4)		(Sólo uso interno)
RESERVED	BLOB (4096)	Sí	(Sólo uso interno)

DB2eSYSRELS

Esta tabla base del catálogo del sistema contiene una fila para cada restricción referencial.

Tabla 10. Tabla base del catálogo del sistema DB2eSYSRELS

Nombre de la columna	Tipo de datos	Admite nulos	Descripción
RMD_ID	INTEGER (4)		Clave primaria y foránea (sólo uso interno)
PKTABLE_NAME	VARCHAR (129)		Nombre de tabla padre
PKCOLUMN_NAME	VARCHAR (129)		Columna de clave primaria de la tabla padre
FKTABLE_NAME	VARCHAR (129)		Nombre de tabla hijo
FKCOLUMN_NAME	VARCHAR (129)		Nombre de la columna de clave foránea de la tabla hijo
ORDINAL_POSITION	INTEGER (4)		Posición de la columna en la referencia de clave foránea

DB2eSYSUSERS

La tabla DB2eSYSUSERS se crea automáticamente cuando se crea la primera tabla cifrada o cuando se ejecuta la primera sentencia GRANT. Esta tabla está estrechamente ligada a la base de datos y a los datos cifrados; no se puede mover a otra base de datos DB2 Everyplace que contenga datos cifrados distintos.

Esta tabla base del catálogo del sistema contiene una fila para cada nombre de usuario registrado que se define para una base de datos.

Tabla 11. Tabla base del catálogo del sistema DB2eSYSUSERS

Nombre de la columna	Tipo de datos	Admite nulos	Descripción
USERNAME	VARCHAR (129)		Parte de la clave primaria, sensible a las mayúsculas/minúsculas. El nombre del usuario asociado a esta fila.
DATABASENAME	VARCHAR (129)		Para uso futuro. Se almacena una serie vacía. Parte de la clave primaria.
TABLENAME	VARCHAR (129)		Para uso futuro. Se almacena una serie vacía. Parte de la clave primaria.
ENCMETHOD	VARCHAR (129)		Para uso futuro. Se almacena una serie vacía. Parte de la clave primaria.
PRIVILEGES	VARCHAR (129)	Sí	Define privilegios del usuario. Actualmente sólo se admite el valor 'E', que indica cifrado.
ENCKEYDATA	BLOB (280)	Sí	Se utiliza para regenerar la clave de cifrado.
ATTIME	TIMESTAMP (26)	Sí	Hora en que se ha añadido el usuario o se ha modificado por última vez el registro, la que sea más reciente de las dos.
VALIDATE	BLOB (280)	Sí	Verifica que el registro es auténtico y que el usuario lo ha añadido un usuario autentificado.
GRANTOR	VARCHAR (129)	Sí	El nombre del usuario que ha registrado el nombre de usuario de la columna 1.
INTERNALDATA	BLOB (255)	Sí	(Uso futuro interno)

Definición del atributo de suma de comprobación para detectar cambios en archivos

DB2 Everyplace permite utilizar un atributo de conexión llamado SQL_TABLE_CHECKSUM que permite que una aplicación detecte si software de proveedor ha modificado el contenido de una base de datos o se ha dañado el contenido de la base de datos.

Cuando el atributo SQL_TABLE_CHECKSUM_ATTR tiene el valor ON, DB2 Everyplace almacena los archivos con la suma de comprobación habilitada. Esta propiedad de conexión se utiliza con las funciones SQLSetConnectAttr() y SQLGetConnectAttr(). Para habilitar esta función, siga este paso:

Antes de crear una base de datos, ejecute SQLSetConnectAttr() y establezca SQL_ATTR_TABLE_CHECKSUM en SQL_TABLE_CHECKSUM_ON, como en este ejemplo de CLI:`rc = SQLSetConnectAttr(hdbc, SQL_ATTR_TABLE_CHECKSUM, (SQLPOINTER) SQL_TABLE_CHECKSUM_ON, 0);`

No puede cambiar este atributo en bases de datos existentes. Después de conectarse a una base de datos, las aplicaciones pueden utilizar la función SQLGetConnectAttr() para determinar si la propiedad de suma de comprobación está habilitada.

Tratamiento de conflictos de nombres entre tablas

En este tema se muestran algunos ejemplos de las maneras en que se pueden manejar los conflictos de denominación de archivos para las tablas definidas por el usuario.

Suponga que una aplicación ejecuta la sentencia CREATE TABLE siguiente:

```
CREATE TABLE T (PK INT NOT NULL PRIMARY KEY, A INT)
```

Una vez ejecutada esta sentencia, DB2 Everyplace creará los dos archivos siguientes para la tabla *T*:

- DSY_T (datos)
- DSY_iT (índices)

Si crea otra tabla y utiliza el nombre *iT*, DB2 Everyplace creará dos archivos adicionales: DSY_iT (datos) y DSY_iiT (índices). El archivo de índices de la tabla *T* y el archivo de datos de la tabla *iT* tienen un conflicto porque tienen el mismo nombre. Ambos archivos se denominan DSY_iT. Para evitar este problema, DB2 Everyplace da soporte a la correlación de nombres de archivo. Es decir, que DB2 Everyplace creará y gestionará por completo los nombres de archivo. Para utilizar esta características, las aplicaciones deben establecer el atributo de conexión y éste se tiene que ejecutar antes de que se cree la primera tabla. Por ejemplo, en la CLI:

```
SQLSetConnectAttr(hdbc, SQL_ATTR_FILENAME_FORMAT,
                  (SQLPOINTER)SQL_FILENAME_FORMAT_83, 0)
```

O en DB2eCLP:

```
DISABLE LONG FILENAME
```

Una vez ejecutado este mandato y creada la primera tabla, los archivos resultantes serán para la tabla *T*:

- 0001.DBd
- 0001.DBi

Conexión a la base de datos portátil DB2 Everyplace

Normalmente, las aplicaciones crean y acceden a tablas en una ubicación específica, por ejemplo, el directorio C:\TEMP. Al conectar con una base de datos portátil DB2 Everyplace, puede utilizar la llamada de CLI para especificar una ubicación.

En el ejemplo siguiente, *vía-acceso* representa la vía de acceso de la base de datos portátil DB2 Everyplace.

```
rc = SQLConnect(hdbc, vía-acceso, SQL_NTS, uid, SQL_NTS, pwd, SQL_NTS);
```

La vía de acceso puede incluir (aunque no es necesario) el nombre de la base de datos. Por lo tanto, los dos ejemplos siguientes son correctos, suponiendo que exista una base de datos portátil DB2 Everyplace en C:\TEMP.

```
rc = SQLConnect(hdbc, "C:\\TEMP\\mi_base_datos", SQL_NTS, uid, SQL_NTS, pwd, SQL_NTS);
rc = SQLConnect(hdbc, "C:\\TEMP\\", SQL_NTS, uid, SQL_NTS, pwd, SQL_NTS);
```

La conexión a memoria ampliada Sony Memory Stick en Palm OS requiere una especificación de vía de acceso especial, tal como muestra el ejemplo siguiente.

```
rc = SQLConnect(hdbc, "#0:\\", SQL_NTS, uid, SQL_NTS, pwd, SQL_NTS);
```

Mediante DB2eCLP, puede conectar con una ubicación específica utilizando el mandato ″CONNECT TO″. Por ejemplo, el mandato siguiente conecta con la base de datos portátil DB2 Everyplace en C:\TEMP\ en un sistema que ejecuta Windows:

```
CONNECT TO C:\TEMP\
```

PRECAUCIÓN:
Para las plataformas Windows y Windows CE, no es seguro invocar DB2 Everyplace desde dentro de DllMain. Esto es especialmente importante para la versión 8.2, pues DB2 Everyplace añadió una hebra de proceso de fondo para mejorar el rendimiento. Por ejemplo, una aplicación que llame a SQLConnect() desde dentro de DllMain producirá un punto muerto u otro resultado inesperado. Para obtener más información sobre este tema, consulte la documentación de Microsoft.

Serialización de conexiones

Una fuente de datos DB2 Everyplace acepta varias conexiones de un proceso a la vez. Cuando más de un proceso intenta conectarse a la misma fuente de datos al mismo tiempo, las peticiones se ponen en una cola mediante un mecanismo denominado *serialización de conexiones*.

Para la serialización de conexiones, es necesario que los desarrolladores decidan cuánto tiempo debe esperar una aplicación a obtener una conexión. Este intervalo, denominado *período de tiempo de espera*, se puede establecer utilizando el atributo SQL_ATTR_LOGIN_TIMEOUT de la función SQLSetConnectAttr(). El siguiente ejemplo de JDBC establece 10 como valor del período de tiempo de espera de conexión. Si la aplicación no puede conectarse a la base de datos en 10 segundos, devolverá un código de error.

El período de tiempo de espera

Para **JDBC:**

```
int waitTime = 10;
String url    = "jdbc:db2e:mysample";
Properties prop = new Properties();
prop.setProperty("LOGIN_TIMEOUT", Integer.toString(waitTime));
Connection con = driver.connect(url,prop);
```

Notas:

- El período de tiempo de espera predeterminado es 0 segundos.
- El período LOCK_TIMEOUT predeterminado es de 20 segundos.
- Una aplicación multihebra puede conectarse a una base de datos utilizando una hebra y desconectarse de la base de datos utilizando otra hebra.
- La serialización de conexiones puede no funcionar con una base de datos situada en una unidad de red.
- En un programa JDBC, el valor del tiempo de espera se ignorará y se establecerá en cero si se pasa en una propiedad al método DriverManager.getConnection().
- DB2 Everyplace permite el acceso simultáneo a bases de datos *dentro del mismo proceso* (o espacio de direcciones). Por ejemplo, el método connect de la interfaz java.sql.Driver soporta ENABLE_SHARED_DATABASE_ACCESS, una propiedad booleana que se puede establecer en true para permitir el acceso simultáneo. DB2 Everyplace soporta propiedades y métodos similares para otros lenguajes.

Detención de operaciones de base de datos de larga duración

El proceso de una operación de base de datos de larga duración, por ejemplo una consulta compleja o una sentencia DELETE que afecte a muchas filas, puede impedir que la aplicación realice otras operaciones de base de datos. En estos casos, es posible que desee detener la operación de base de datos de larga duración.

Para detener operaciones de base de datos de larga duración forzando la desconexión de la conexión:

1. Asigne un descriptor de entorno llamando a SQLAllocHandle() con un valor SQL_HANDLE_ENV para HandleType y un valor SQL_NULL_HANDLE para InputHandle. Por ejemplo:
   ```
   sqlrc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &hevn);
   ```
2. Asigne un descriptor de conexión llamando a SQLAllocHandle() con un valor SQL_HANDLE_DBC para HandleType y utilice el descriptor de entorno devuelto en el Paso 1 como el argumento InputHandle. Por ejemplo:
   ```
   sqlrc |= SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);
   ```
3. Establezca los atributos de conexión para su aplicación llamando a SQLSetConnectAttr() con el atributo SQL_ATTR_FORCE_DISCONNECT establecido en el valor SQL_FORCE_DISCONNECT_ON. Por ejemplo:
   ```
   sqlrc |= SQLSetConnectAttr(hdbc, SQL_ATTR_FORCE_DISCONNECT, (SQLPOINTER) SQL_FORCE_DISCONNECT_ON, 0);
   ```
4. Conéctese a una fuente de datos llamando a una función con el descriptor de conexión que ha asignado en el Paso 2 para cada fuente de datos a la que desee conectarse. Por ejemplo: `sqlrc |= SQLConnect(hdbc, server, SQL_NTS, uid, SQL_NTS, pwd, SQL_NTS);`

La aplicación termina inmediatamente la tarea de larga duración y fuerza la desconexión de la conexión. En este caso, la hebra que procesa la operación de base de datos de larga duración puede recibir un error (SQLState 08008) puesto que la tarea de larga duración no se ha procesado satisfactoriamente.

Comportamiento del cursor en el contexto de una conexión

Cursor de lectura general en conflictos de escritura procedentes de otro descriptor de sentencia

Una aplicación puede tener varios descriptores de sentencia que efectúen operaciones de lectura y escritura en la misma tabla al mismo tiempo. Los conflictos se producen cuando un descriptor está efectuando una operación de escritura en la tabla (por ejemplo, UPDATE, DELETE o INSERT) al tiempo que otro descriptor está en medio de una operación de lectura o escritura. En DB2 Everyplace, el cursor de lectura es estable y está siempre leyendo los datos más actuales. Sobrevive a los conflictos de escritura, sin tener en cuenta si está utilizando o no un índice.

Por ejemplo, supongamos que una aplicación tiene dos descriptores de sentencia:

- El descriptor 1 busca filas de la tabla T
- El descriptor 2 suprime filas de la misma tabla

Cada descriptor puede haber sido creado por hebras diferentes (por ejemplo, en un entorno de hebra de Java). A continuación se muestra una caso posible:

```
// Buscar 2 filas en la tabla T
   Statement handle 1: execute "SELECT A FROM T WHERE primary_key < 10"
   Statement handle 1: fetch one row; fetch another row
// Suprimir algunas filas de la tabla T
   Statement handle 2: prepare "DELETE FROM T WHERE primary_key = ?"
   Statement handle 2: execute
// Seguir buscando una fila más en T
   Statement handle 1: fetch one row
```

En este punto de la ejecución, el descriptor de sentencia número 1 puede seguir buscando la fila siguiente (si existe), sin tener en cuenta si se utiliza o no un índice. En el escenario anterior, *se* utiliza un índice debido a que hay una clave primaria. La idea es que DB2 Everyplace intentará volver a situar la posición del cursor del descriptor número 1, utilizando su posición actual, antes de avanzar. Si la posición actual ya no existe (por ejemplo, otro descriptor de sentencia suprimió la fila), el cursor simplemente avanza hacia la posición siguiente efectuando una búsqueda. Del mismo modo, si otro descriptor de sentencia ha suprimido la posición siguiente, el cursor podrá saltar sobre el ″espacio″ hacia la posición siguiente.

Cursor desplazable en conflictos de escritura procedentes de otro descriptor de sentencia

Considere un ejemplo similar al del tema anterior, pero en el que el cursor de lectura es un cursor desplazable. Si es un cursor desplazable ″insensible″, esto no será un problema debido a que por definición el conjunto de resultados no cambia. Si el cursor no es ″insensible″, su comportamiento coincidirá con el cursor de lectura regular descrito anteriormente. En esencia, el comportamiento del cursor de lectura después del conflicto es que el conjunto de resultados vuelva a calcularse con arreglo a los datos de tabla más actuales y que se mantenga el comienzo del conjunto de filas actual. El cursor se avanza hacia la fila siguiente en el caso de que se suprima la fila actual.

El ejemplo siguiente muestra el caso de un cursor desplazable utilizando DB2eCLP. Suponga que la tabla T tiene seis filas:

```
create table T (a int, b int)
     create index idx1 on T(a)
     insert into T values (1, 1)
     insert into T values (2, 2)
```

```
insert into T values (3, 1)
insert into T values (3, 2)
insert into T values (3, 3)
insert into T values (4, 4)
```

Abusando de su generosidad, piense en un ejemplo en el que la aplicación tuviera dos descriptores de sentencia, uno para la lectura y el otro para la supresión.

```
Statement handle 1: enable scrollable cursor;
Statement handle 1: execute "SELECT A FROM T WHERE a < 10"
Statement handle 2: prepare "DELETE FROM T WHERE a = ?"
Statement handle 1: fetchscroll with SQL_FETCH_FIRST
-- get (1, 1)
Statement handle 1: fetchscroll with SQL_FETCH_NEXT
-- get (2, 2)
Statement handle 1: fetchscroll with SQL_FETCH_NEXT
-- get (3, 1)
Statement handle 2: execute
--- suppose delete row (2, 2)
Statement handle 1: fetchscroll with SQL_FETCH_NEXT
-- re-compute previous rows, and return (3, 2)
Statement handle 1: fetchscroll with SQL_FETCH_PRIOR
-- get (3, 1)
Statement handle 1: fetchscroll with SQL_FETCH_PRIOR
-- get (1, 1)  note that (2, 2) is gone
Statement handle 1: fetchscroll with SQL_FETCH_ABSOLUTE, offset 2
-- get (3, 1)  note that (2, 2) is gone
Statement handle 1: fetchscroll with SQL_FETCH_ABSOLUTE, offset 5
-- get (4, 4)
```

Cursor en confirmación y retrotracción, incluida la modalidad de confirmación automática

Cualquiera que sea la modalidad de transacción o confirmación automática, un cursor abierto permanece abierto durante la confirmación, y un cursor abierto se cierra al retrotraer una transacción completa.

Después de una retrotracción de transacción parcial, tal como una sentencia ROLLBACK TO SAVEPOINT, un cursor abierto permanece abierto si se ha especificado la cláusula UPON ROLLBACK RETAIN CURSORS para el punto de rescate indicado. De lo contrario, el cursor se cierra.

Dependencia del objeto

Preparar una sentencia de SQL por medio de un descriptor de sentencia H puede crear cierta dependencia sobre determinados objetos. Por ejemplo, seleccionar filas de una tabla T por medio de un Idx de índice requiere la existencia de la tabla T y del Idx de índice. Si otro descriptor de sentencia ha suprimido estos objetos (por ejemplo, si se ha descartado el Idx de índice), volver a ejecutar la sentencia por medio de H forzará una recompilación de la sentencia de SQL. Como resultado, el plan de la consulta podría ser diferente o se podría devolver un error.

Ajuste de aplicaciones de base de datos

Los temas de esta sección describen técnicas para mejorar el rendimiento de las aplicaciones de base de datos.

Una fuente de datos DB2 Everyplace acepta varias conexiones de un proceso a la vez. Cuando más de un proceso intenta conectarse a la misma fuente de datos al mismo tiempo, las peticiones se ponen en una cola. No obstante, DB2 Everyplace permite varias conexiones de base de datos *dentro del mismo proceso* (o espacio de direcciones). Por ejemplo, el método connect de la interfaz java.sql.Driver soporta ENABLE_SHARED_DATABASE_ACCESS, una propiedad booleana que se puede establecer en true para permitir el acceso simultáneo.

Antes de desarrollar aplicaciones donde se utilizan conexiones múltiples, debe comprender los conceptos siguientes.

Problemas de simultaneidad

La *simultaneidad* hace referencia al uso compartido de recursos por parte de diversos usuarios interactivos o programas de aplicación al mismo tiempo. DB2 Everyplace soporta las transacciones simultáneas, lo que permite a una aplicación establecer varias conexiones distintas con la misma base de datos.

Al desarrollar este tipo de aplicación, tenga cuidado para evitar efectos no deseados como, por ejemplo:

- **Pérdida de actualizaciones**. Es posible que dos aplicaciones, A y B, lean la misma fila de la base de datos y que ambas calculen nuevos valores para una de sus columnas basándose en los datos que dichas aplicaciones lean. Si A actualiza la fila con su nuevo valor y luego B también actualiza la fila, la actualización realizada por A se perderá.
- **Acceso a datos no confirmados**. Es posible que la aplicación A actualice un valor en la base de datos y la aplicación B lea este valor antes de que se confirme. Entonces, si el valor de A no se confirma posteriormente sino que se retrotrae, los cálculos realizados por B se basarán en datos no confirmados (y supuestamente no válidos).
- **Lecturas no repetibles**. Algunas aplicaciones implican la siguiente secuencia de sucesos: la aplicación A lee una fila de la base de datos y, a continuación, pasa a procesar otras peticiones de SQL. Mientras tanto, la aplicación B modifica o suprime la fila y confirma el cambio. Posteriormente, si la aplicación A intenta volver a leer la fila original, recibirá la fila modificada o descubrirá que la fila original se ha suprimido.
- **Lecturas fantasma**. El fenómeno de las lecturas fantasma se produce cuando:
  1. La aplicación ejecuta una consulta.
  2. Otra aplicación inserta o actualiza datos que satisfacen los criterios de consulta de la aplicación.
  3. La aplicación repite la consulta del paso 1 (dentro de la misma unidad de trabajo), pero el conjunto de resultados es diferente porque incluye filas "fantasma" adicionales insertadas o actualizadas por la otra aplicación.

Este comportamiento se puede evitar en una aplicación de DB2 Everyplace gestionando bloqueos y niveles de aislamiento. Si la aplicación no necesita múltiples conexiones de base de datos, se pueden evitar todos los problemas de simultaneidad inhabilitando el acceso compartido por completo. Por ejemplo, el método connect de la interfaz java.sql.Driver soporta ENABLE_SHARED_DATABASE_ACCESS, una propiedad booleana que se puede establecer en false para inhabilitar el acceso simultáneo. DB2 Everyplace soporta propiedades y métodos similares para otros lenguajes. Para obtener más información, consulte la documentación de consulta.

Bloqueo de tablas

Un *bloqueo* asocia un recurso de gestor de bases de datos con una aplicación para controlar el modo en que las demás aplicaciones pueden acceder al mismo recurso. DB2 Everyplace da soporte al *bloqueo de tablas*. Es decir, una tabla se debe bloquear por completo o no se debe bloquear. No se pueden bloquear filas específicas de una tabla.

DB2 Everyplace da soporte a dos tipos de bloqueos de tabla:

- Bloqueos exclusivos, utilizados en sentencias DDL y DML.
- Bloqueos compartidos, utilizados en sentencias SELECT.

La tabla siguiente muestra cómo se pueden combinar estos tipos de bloqueos cuando varios usuarios o varias transacciones acceden a una tabla.

Tabla 12. Compatibilidad de bloqueos

	SHARED	**EXCLUSIVE**
SHARED	Compatible	Incompatible
EXCLUSIVE	Incompatible	Incompatible

Una aplicación puede bloquear una tabla llamando a la sentencia de SQL LOCK TABLE. Por ejemplo, el código siguiente obtiene un bloqueo exclusivo sobre la tabla EMP.

```
LOCK TABLE EMP IN EXCLUSIVE MODE
```

El bloqueo de tablas resulta apropiado para las transacciones de sólo grabación y para el acceso de un solo usuario. Cuando dos o más transacciones actualizan la misma tabla, el bloqueo de la tabla puede llevar a un punto muerto. Por ejemplo, considere el siguiente caso práctico:

1. Dos transacciones, la A y la B, obtienen un bloqueo compartido sobre la tabla T.
2. Posteriormente, las dos transacciones necesitan grabar en la tabla T, que necesita un bloqueo exclusivo.
3. Ninguna de las dos transacciones puede obtener un bloqueo exclusivo, y los bloqueos compartidos y los exclusivos son incompatibles.
4. Cada transacción espera a que la otra libere el bloqueo compartido, lo que tiene como resultado un punto muerto.

DB2 Everyplace proporciona un mecanismo de tiempo de espera excedido que las aplicaciones pueden utilizar para resolver puntos muertos. Si una aplicación no puede obtener un bloqueo dentro de un período de tiempo especificado, el motor de base de datos retrotraerá la transacción y devolverá el SQLSTATE 40001. El tiempo de espera predeterminado para los bloqueos es de 20 segundos.

Directrices para el bloqueo

Este tema proporciona las directrices que debe tener en cuenta al ajustar los bloqueos para controlar la concurrencia y la integridad de los datos.

- DB2 Everyplace bloquea tablas enteras.

  Una tabla se debe bloquear por completo o no se debe bloquear. No se pueden bloquear filas específicas de una tabla.
- Cree unidades de trabajo pequeñas con sentencias COMMIT frecuentes para permitir el acceso simultáneo a datos por parte de muchos usuarios.

  Incluya sentencias COMMIT cuando los datos que haya modificado sean coherentes. Cuando se emita una confirmación (COMMIT), se liberarán todos los bloqueos a excepción de los relacionados con cursores abiertos (en DB2 Everyplace, los cursores se retienen durante una confirmación (COMMIT)).

Tras una confirmación (COMMIT), todos los bloqueos restantes son compartidos (SHARED). Todos los bloqueos se liberan en una retrotracción (ROLLBACK).

- Especifique un nivel de aislamiento apropiado.

  Los bloqueos compartidos los adquieren los niveles de aislamiento serializables, de lectura repetible y de lectura confirmada, incluso en las aplicaciones de sólo lectura. Para liberar estos bloqueos, cierre los cursores que no estén en uso.

  El gestor de bases de datos garantiza que la aplicación no recupere datos no confirmados (filas que otras aplicaciones hayan actualizado pero que aún no se hayan confirmado) a menos que se utilice el nivel de aislamiento de lectura no confirmada.

- Utilice la sentencia LOCK TABLE del modo adecuado.

  Sólo se bloquea la tabla especificada en la sentencia LOCK TABLE. Las tablas padre y las dependientes de la tabla especificada no se bloquean. Deberá determinar si es necesario bloquear otras tablas para conseguir el resultado deseado en términos de simultaneidad y rendimiento. El bloqueo no se libera hasta que la unidad de trabajo se confirma o se retrotrae.

  LOCK TABLE IN SHARE MODE

  Los datos a los que acceda tienen que ser coherentes en el tiempo; es decir, deben ser datos actuales para una tabla en un determinado momento. Si la tabla tiene una actividad frecuente, el único modo de garantizar que toda la tabla permanezca estable es bloquearla. Por ejemplo, la aplicación quiere tomar una instantánea de una tabla. No obstante, durante el tiempo que la aplicación necesita para procesar algunas filas de una tabla, otras aplicaciones actualizan filas que aún no se han procesado. Esto se permite con la lectura repetible, pero esta acción no es la que desea.

  Como alternativa, la aplicación puede emitir la sentencia LOCK TABLE IN SHARE MODE: no se puede modificar ninguna fila, independientemente de si se ha recuperado o no. Puede recuperar tantas filas como necesite con la garantía que las filas que haya recuperado no se habrán modificado justo antes de recuperarlas.

  Con LOCK TABLE IN SHARE MODE, otros usuarios puede recuperar datos de la tabla, pero no pueden actualizar, suprimir ni insertar filas en la tabla.

  LOCK TABLE IN EXCLUSIVE MODE

  Con LOCK TABLE IN EXCLUSIVE MODE, se bloquea a todos los demás usuarios; ninguna otra aplicación puede acceder a la tabla a menos que se trate de una aplicación de lectura no confirmada.

- Cierre los cursores para liberar los bloqueos que retengan.

  En DB2 Everyplace, los cursores se retienen entre confirmaciones por omisión y se cierran implícitamente al ejecutar la siguiente sentencia. Si una aplicación ya no necesita el cursor en el momento de la confirmación, deberá cerrar el cursor explícitamente antes de confirmar la transacción para liberar sus bloqueos compartidos. Además, el nivel de aislamiento de una conexión sólo se puede establecer si no hay cursores abiertos y la confirmación automática está activada; de lo contrario, se devuelve el SQLSTATE HY011.

Niveles de aislamiento

Un *nivel de aislamiento* especifica hasta qué punto está aislada una transacción de otras transacciones en un entorno de múltiples conexiones. DB2 Everyplace es compatible con los siguientes niveles de aislamiento de SQL de ANSI:

Nota: Los niveles están listados con sus equivalentes para DB2 Versión 9.1 en orden decreciente de su efecto sobre el rendimiento, pero en orden creciente del cuidado necesario al acceder y actualizar datos (por ejemplo, el riesgo de situaciones de punto muerto varía según el nivel de aislamiento). Después de la tabla encontrará detalles sobre cada nivel.

Tabla 13. Niveles de aislamiento

Nivel de aislamiento ANSI SQL	Equivalente para DB2 Versión 9.1
SERIALIZABLE	Lectura repetible (RR)
REPEATABLE READ	Estabilidad de lectura (RS)
READ COMMITTED (valor predeterminado)	Estabilidad de cursor (CS)
READ UNCOMMITTED	Lectura no confirmada (UR)

SERIALIZABLE (DB2 Versión 9.1: Lectura repetible)

Bloquea la tabla en una unidad de trabajo. Una aplicación puede recuperar filas de la tabla y trabajar sobre ellas tantas veces como sea necesario. No obstante, se bloquea la tabla entera, no sólo las filas que se recuperan. Hasta que la unidad de trabajo finalice, ninguna otra aplicación podrá actualizar, suprimir ni insertar una fila que afecte a la tabla.

Las aplicaciones SERIALIZABLE no pueden ver cambios no confirmados realizados por otras aplicaciones. Por lo tanto, una sentencia SELECT emitida repetidamente en la unidad de trabajo dará el mismo resultado cada vez. Las actualizaciones perdidas, el acceso a datos no confirmados y las filas fantasmas no son posibles.

REPEATABLE READ (DB2 Versión 9.1: Estabilidad de lectura)

Puesto que DB2 Everyplace bloquea tablas enteras (no filas específicas), REPEATABLE READ tiene exactamente el mismo comportamiento que SERIALIZABLE.

READ COMMITTED (DB2 Versión 9.1: Estabilidad del cursor)

Se bloquea la tabla entera. Los bloqueos compartidos se liberan cuando se cierran los cursores asociados (los niveles de aislamiento superiores a READ COMMITTED retienen los bloqueos compartidos hasta el final de la transacción). Los bloqueos exclusivos se retienen hasta el final de la transacción.

Ninguna otra aplicación puede realizar otra operación DML sobre una tabla mientras un cursor abierto acceda a ésta. Las aplicaciones READ COMMITED no pueden ver cambios no confirmados de otras aplicaciones.

Son posibles tanto las lecturas no repetibles como las lecturas fantasma. READ COMMITTED es el nivel de aislamiento predeterminado; permite la máxima simultaneidad mientras se ven sólo las filas confirmadas de otras aplicaciones.

READ UNCOMMITTED (DB2 Versión 9.1: Lectura no confirmada)

Una aplicación puede acceder a *algunos* cambios no confirmados de otras transacciones: las tablas y los índices que otras aplicaciones están creando o descartando no están disponibles mientras la transacción está en proceso. Cualquier otro cambio se puede leer antes de que se confirma o se retrotraiga.

En este nivel, la aplicación no bloquea el acceso de otras aplicaciones a la tabla que está leyendo.

La tabla siguiente resume los niveles de aislamiento por lo que respecta a sus efectos no deseados.

Tabla 14. Resumen de los niveles de aislamiento

Nivel de aislamiento	Acceso a datos no confirmados	Lecturas no repetibles	Fenómeno de lectura fantasma
SERIALIZABLE	No es posible	No es posible	No es posible
REPEATABLE READ	No es posible	Es posible	Es posible
READ COMMITTED	No es posible	Es posible	Es posible
READ UNCOMMITTED	Es posible	Es posible	Es posible

La tabla siguiente puede ayudarle a elegir un nivel de aislamiento inicial para sus aplicaciones. Considere esta tabla como un punto de partida y consulte las explicaciones anteriores de los distintos niveles para obtener información sobre los factores que pueden hacer que otro nivel de aislamiento resulte más adecuado.

Tabla 15. Directrices para elegir un nivel de aislamiento

Tipo de aplicación	Se necesita una alta estabilidad de los datos	No se necesita una alta estabilidad de los datos
Transacciones de lectura-grabación	REPEATABLE READ	READ COMMITTED
Transacciones de sólo grabación	SERIALIZABLE o REPEATABLE READ	READ UNCOMMITTED

Otros puntos a tener en cuenta:

- Las sentencias INSERT, UPDATE y DELETE siempre funcionan del mismo modo independientemente del nivel de aislamiento. Sólo varía el funcionamiento de las sentencias SELECT.
- El nivel de aislamiento sólo se puede establecer al principio de una transacción, por lo que permanece en vigor mientras dure la unidad de trabajo.

Serialización de conexiones

Una fuente de datos DB2 Everyplace acepta varias conexiones de un proceso a la vez. Cuando más de un proceso intenta conectarse a la misma fuente de datos al mismo tiempo, las peticiones se ponen en una cola mediante un mecanismo denominado *serialización de conexiones*.

Para la serialización de conexiones, es necesario que los desarrolladores decidan cuánto tiempo debe esperar una aplicación a obtener una conexión. Este intervalo, denominado *período de tiempo de espera*, se puede establecer utilizando el atributo SQL_ATTR_LOGIN_TIMEOUT de la función SQLSetConnectAttr(). El siguiente ejemplo de JDBC establece 10 como valor del período de tiempo de espera de conexión. Si la aplicación no puede conectarse a la base de datos en 10 segundos, devolverá un código de error.

El período de tiempo de espera

Para **JDBC:**

```
int waitTime = 10;
String url    = "jdbc:db2e:mysample";
Properties prop = new Properties();
prop.setProperty("LOGIN_TIMEOUT", Integer.toString(waitTime));
Connection con = driver.connect(url,prop);
```

Notas:

- El período de tiempo de espera predeterminado es 0 segundos.
- El período LOCK_TIMEOUT predeterminado es de 20 segundos.
- Una aplicación multihebra puede conectarse a una base de datos utilizando una hebra y desconectarse de la base de datos utilizando otra hebra.
- La serialización de conexiones puede no funcionar con una base de datos situada en una unidad de red.
- En un programa JDBC, el valor del tiempo de espera se ignorará y se establecerá en cero si se pasa en una propiedad al método DriverManager.getConnection().
- DB2 Everyplace permite el acceso simultáneo a bases de datos *dentro del mismo proceso* (o espacio de direcciones). Por ejemplo, el método connect de la interfaz java.sql.Driver soporta ENABLE_SHARED_DATABASE_ACCESS, una propiedad booleana que se puede establecer en true para permitir el acceso simultáneo. DB2 Everyplace soporta propiedades y métodos similares para otros lenguajes.

Seguridad en DB2 Everyplace

Los temas de esta sección describen técnicas que puede utilizar para hacer que las aplicaciones sean más seguras.

Cifrado de datos locales

En DB2 Everyplace, el cifrado está pensado para proteger datos situados en un dispositivo portátil o incorporado. Este tema proporciona una rápida visión general de la importancia del cifrado de datos locales y una serie de tareas apropiadas para ayudarle a comenzar a utilizar esa función. También se describe cómo se habilita el cifrado para cada plataforma y se muestran las bibliotecas necesarias, además de las que necesita la base de datos portátil DB2 Everyplace.

Bibliotecas necesarias:

Para Windows:

- biblioteca de plug-in: CryptoPlugin.dll (proporcionada por DB2 Everyplace)
- biblioteca de cifrado: Crypt32.dll (paquete Cypher Strength Encryption de 128 bits, se proporciona con Internet Explorer 5.5 o superior). Vaya al sitio web http://www.microsoft.com/windows/ie/downloads/default.mspx para descargar Internet Explorer.

Para Windows CE/Pocket PC

- biblioteca de plug-in: CryptoPlugin.dll (proporcionada por DB2 Everyplace)
- biblioteca de cifrado: Microsoft High Encryption Pack para Pocket PC V1.0. Vaya a http://www.microsoft.com/windowsmobile/downloads/highencryption.mspx para descargar el paquete de cifrado. Este paquete forma parte de Pocket PC 2003, pero se debe instalar en Pocket PC 2002. Si la biblioteca CryptoPlugin.dll existe pero el paquete de cifrado no está instalado, las aplicaciones no podrán conectarse a ninguna base de datos (por ejemplo DB2eCLP no se podrá iniciar). Si una aplicación requiere el cifrado, instale Microsoft High Encryption Pack para Pocket PC. Si el cifrado no es necesario, suprima CryptoPlugin.dll del directorio de Windows en el dispositivo Pocket PC.

Para Linux

- biblioteca de plug-in: libcryptoplugin.so (proporcionada por DB2 Everyplace)
- biblioteca de cifrado: libpvcpkcs11.so (proporcionada por DB2 Everyplace)

¿Por qué utilizar el cifrado de datos locales?

Piense en una aplicación corporativa de ventas que contiene datos de contacto con los clientes. Un viajante de comercio puede llevar estos datos en su PDA cuando visite a un cliente. A menos que la aplicación o PDA brinden un sistema de almacenamiento seguro, se puede acceder fácilmente a los datos a través de la aplicación o investigando el sistema de archivos nativo del dispositivo portátil. Los datos sensibles al cifrado se convierten en un aspecto crucial en la protección de la información corporativa.

Objetivos del cifrado de datos locales

DB2 Everyplace proporciona una solución que permite que una aplicación implante una política de seguridad corporativa. El primer objetivo consiste en cifrar la información secreta o delicada almacenada en tablas de DB2 Everyplace. Los datos se cifran utilizando métodos de cifrado estándares tales como DES, que implanta claves de cifrado. El segundo objetivo consiste en proporcionar un marco seguro para

poder gestionar las claves utilizadas para cifrar los datos. Se requiere al usuario para que suministre un ID de usuario y una contraseña en el momento de conectar con la base de datos.

Para obtener información sobre la utilización del cifrado de datos, consulte los temas siguientes.

Establecimiento de una conexión con la base de datos portátil DB2 Everyplace

Cualquier interacción con la base de datos portátil DB2 Everyplace requiere que se establezca una conexión. Además, para que un usuario pueda acceder a tablas cifradas o crearlas, la aplicación debe conectar con DB2 Everyplace utilizando un ID de usuario y una contraseña que no estén vacíos.

Esta tarea forma parte de la tarea principal de cifrar datos locales. Después de completar estos pasos, vuelva al apartado “Cifrado de datos locales” en la página 37.

El ejemplo siguiente muestra cómo establecer conexión con una base de datos. El ejemplo utiliza la función de CLI:

```
rc = SQLConnect(hdbc, "C:\temp\", SQL_NTS, "user1", SQL_NTS, "pwd1", SQL_NTS)
```

donde *"C:\temp\"* es el directorio de la base de datos portátil DB2 Everyplace con la que la aplicación está conectada, utilizando el ID de usuario *"usuario1"* y la contraseña *"contras1"*.

Para una interfaz JDBC, se puede establecer una conexión de base de datos de forma parecida.

Cómo otorgar privilegios de cifrado a un usuario

Antes de crear la primera tabla cifrada, la aplicación debe otorgar privilegios de cifrado a un usuario.

Esta tarea forma parte de la tarea principal de cifrar datos locales. Después de completar estos pasos, vuelva al apartado “Cifrado de datos locales” en la página 37.

Por ejemplo, la aplicación puede emitir la sentencia siguiente de SQL:

```
rc = SQLExecDirect(..., "GRANT ENCRYPT ON DATABASE TO \"usuario1\"" +
                         " using \"pwd1\" new \"pwd1\"", SQL_NTS)
```

Al ejecutar esta sentencia de SQL, DB2 Everyplace creará una tabla de catálogos del sistema llamada DB2eSYSUSERS, y se insertará una fila en dicha tabla. Esto significa que el usuario *"usuario1"* ya estará registrado con la contraseña correspondiente y ahora tendrá todos los privilegios de cifrado, como por ejemplo para crear tablas cifradas y acceder a ellas.

Esta tabla está estrechamente vinculada con la base de datos y los datos cifrados, por lo que no basta con trasladarla a otra base de datos portátil DB2 Everyplace para acceder a datos cifrados. Esto es debido a que las distintas bases de datos tendrán claves diferentes para el cifrado o descifrado. Como consecuencia, si una persona tiene permitido acceder a tablas cifradas de una base de datos, dicha persona no puede acceder a otra base de datos utilizando un ID de usuario y una contraseña iguales. Al igual que con otras tablas de catálogos del sistema, una aplicación puede recuperar filas utilizando la sentencia de selección de SQL, pero no puede modificar los datos de esta tabla mediante las sentencias INSERT, DELETE, UPDATE, CREATE y DROP.

Creación de una tabla cifrada

Una vez que se ha establecido una conexión con la base de datos DB2 Everyplace y se han otorgado privilegios de cifrado a un usuario, la aplicación puede crear tablas cifradas mediante una sentencia CREATE TABLE ampliada.

Esta tarea forma parte de la tarea principal de cifrar datos locales. Después de completar estos pasos, vuelva al apartado “Cifrado de datos locales” en la página 37.

Puede elegir uno de los dos algoritmos de cifrado.

El ejemplo siguiente crea la tabla EMPLOYEE con el algoritmo DES:

```
SQLExecDirect(..., "CREATE TABLE EMPLOYEES (EMPNO INT PRIMARY KEY, NAME VARCHAR(30),
SALARY DECIMAL(10,2)) WITH ENCRYPTION", SQL_NTS)
```

El ejemplo siguiente crea la tabla EMPLOYEE con el algoritmo Triple DES:

```
SQLExecDirect(..., "CREATE TABLE EMPLOYEES (EMPNO INT PRIMARY KEY, NAME VARCHAR(30),
SALARY DECIMAL(10,2)) WITH ENCRYPTION ALGORITHM DES3", SQL_NTS)
```

Para el acceso subsiguiente a tablas cifradas: Si una base de datos contiene la tabla DB2eSYSUSERS, cualquier conexión de base de datos posterior pasará por la autentificación de usuario con el ID de usuario y la contraseña proporcionados. Si la autentificación falla, la aplicación sólo puede acceder a tablas no cifradas. La aplicación no puede crear nuevas tablas cifradas, eliminar tablas cifradas existentes ni acceder o actualizar datos cifrados.

Gestión de los privilegios de cifrado

Una vez que una aplicación conecta con una base de datos con el ID de usuario y la contraseña autentificados, la aplicación puede crear nuevos usuarios, cambiar contraseñas o eliminar del sistema un usuario registrado.

Esta tarea forma parte de la tarea principal de cifrar datos locales. Después de completar estos pasos, vuelva al apartado “Cifrado de datos locales” en la página 37.

La sintaxis para crear un nuevo usuario o cambiar una contraseña es:

```
 GRANT ENCRYPT ON DATABASE TO "nuevousuario" USING "contrasotorgante"
                              NEW "nuevacontras"
```

La sintaxis para eliminar un usuario registrado es:

```
 REVOKE ENCRYPT ON DATABASE FROM "usuario"
```

Nota: Si se eliminan todos los usuarios registrados de la tabla DB2eSYSUSERS (mediante la sentencia REVOKE), no se puede realizar ninguna otra operación de cifrado, incluido el acceso a una tabla cifrada existente. No existe ningún mecanismo de recuperación.

Cifrado utilizando DB2eCLP

Este tema contiene un ejemplo de sesión interactiva diseñada para mostrarle cómo utilizar el cifrado de datos en las aplicaciones. Se han añadido comentarios para explicar cada operación.

```
-- Cifrado utilizando DB2eCLP
--
-- Esto es una sesión de cifrado de ejemplo donde se utiliza el programa
-- de interfaz de línea de mandatos de ejemplo proporcionado, DB2eCLP.
--
-- Sólo se muestra el código de retorno de una sentencia si ésta ha
-- fallado; si ha finalizado satisfactoriamente, sólo se muestran los
-- resultados de las operaciones Select.
-- Los mandatos que se pueden entrar en DB2 Everyplace tienen por
-- prefijo la serie de caracteres "CLP:> ".
--
-- -- (CLI:SQLConnect, SQL:CREATE TABLE, SQL:GRANT, SQL:REVOKE)
--
-- Cuando se inicia DB2eCLP, automáticamente se conecta con
-- la base de datos predeterminada (en el directorio actual).
-- Esto es equivalente a:
--
CLP:> CONNECT TO anything;

-- debido a que no se proporciona ninguna vía de acceso
```

```
-- específica, tan solo el nombre "anything", esa sentencia
-- establece conexión con el directorio actual.
--
-- Ahora crearemos una tabla no cifrada que correlaciona
-- números con palabras suecas de contaje.

CLP:> CREATE TABLE swedish(nummer INT, ord VARCHAR(32));
CLP:> INSERT INTO swedish VALUES(1, 'ett');
CLP:> INSERT INTO swedish VALUES(3, 'tre');
CLP:> INSERT INTO swedish VALUES(4, 'fyra');
CLP:> INSERT INTO swedish VALUES(5, 'fem');
CLP:> INSERT INTO swedish VALUES(7, 'sju');
CLP:> INSERT INTO swedish VALUES(99, 'nittionio');

-- Examine los datos
CLP:> SELECT * FROM swedish;

NUMMER      ORD
----------- --------------------------------
          1 ett
          3 tre
          4 fyra
          5 fem
          7 sju
         99 nittionio
6 row(s) returned.

-- Ahora intentaremos crear la tabla correspondiente para inglés,
-- pero utilizando cifrado.

-- Puede elegir uno de los dos algoritmos de cifrado soportados por DB2 Everyplace
-- al cifrar una tabla con la sentencia CREATE TABLE: DES y triple DES.
-- Consulte el tema CREATE TABLE para obtener más información.

CLP:> CREATE TABLE english(number INT, word VARCHAR(32)) WITH ENCRYPTION;
Statement failed [sqlstate = 42501].

-- Esta sentencia falla porque todavía no estamos autorizados, tal como ha
-- indicado el código de error. Por tanto, tenemos que volver a conectar:
--
CLP:> CONNECT TO something USER jsk USING hemligt;

-- Este mandato conecta con la misma base de datos (directorio por
-- omisión/actual) pero con una identidad de usuario específica "jsk"
-- y utilizando la contraseña "hemligt".
-- El mandato CONNECT TO no es una sentencia de SQL, por lo que lo
-- que es interpretado por la aplicación DB2eCLP. El mandato desconecta
-- de la base de datos portátil DB2 Everyplace y se vuelve a conectar
-- a ella utilizando:
--    SQLConnect(hdbc, "something", SQL_NTS, "jsk", SQL_NTS, "hemligt", SQL_NTS);
--
-- Ahora hemos de crear el primer usuario autorizado. Cuando se crea
-- el primer usuario, éste tiene que tener el mismo nombre y la misma
-- contraseña que el usuario que ha iniciado la sesión:
--
CLP:> GRANT ENCRYPT ON DATABASE TO "jsk" USING "hemligt" NEW "hemligt";

-- Observe que para GRANT es necesario que el nombre y las contraseñas
-- estén entre comillas dobles. Esto es debido a que son sensibles a las
-- mayúsculas y minúsculas y la sentencia se pasa directamente a DB2 Everyplace.
--
-- Ahora que tenemos un usuario de cifrado autorizado, podemos crear
-- la tabla cifrada:
--

CLP:> CREATE TABLE english(number INT, word VARCHAR(32)) WITH ENCRYPTION;
CLP:> INSERT INTO english VALUES(1, 'one');
```

```
CLP:> INSERT INTO english VALUES(3, 'three');
CLP:> INSERT INTO english VALUES(4, 'four');
CLP:> INSERT INTO english VALUES(5, 'five');
CLP:> INSERT INTO english VALUES(7, 'seven');
CLP:> INSERT INTO english VALUES(99, 'ninety nine');

-- Examine los datos.
CLP:> SELECT * FROM english;

NUMBER      WORD
----------- --------------------------------
          1 one
          3 three
          4 four
          5 five
          7 seven
         99 ninety nine
6 row(s) returned.

-- Seleccionar un número aleatorio alto en sueco:
--
CLP:> SELECT * FROM swedish WHERE nummer > 42;

NUMMER      ORD
----------- --------------------------------

         99 nittionio
1 row(s) returned.

-- Seleccionar un número aleatorio alto en inglés:
--
CLP:> SELECT * FROM english WHERE number > 42;

NUMBER      WORD
----------- --------------------------------
         99 ninety nine
1 row(s) returned.

-- Traducir 'fyra' al inglés:
--
CLP:> SELECT word FROM swedish, english WHERE number = nummer AND ord = 'fyra';

WORD
--------------------------------
four
1 row(s) returned.

-- Obtener una tabla de traducción:
--
CLP:> SELECT number, ord, word FROM swedish, english WHERE number = nummer;

NUMBER      ORD                              WORD
----------- -------------------------------- --------------------------------
          1 ett                              one
          3 tre                              three
          4 fyra                             four
          5 fem                              five
          7 sju                              seven
         99 nittionio                        ninety nine
6 row(s) returned.

-- Intente autorizar a otro usuario para que acceda a los datos cifrados
-- con su propia contraseña:
--
CLP:> GRANT ENCRYPT ON DATABASE TO "xin" USING "notKnown" NEW "notKnown";
Statement failed [sqlstate = 42506].
```

```
-- Esta sentencia ha fallado porque el usuario que ha iniciado la sesión
-- se tiene que autovalidar para poder añadir un nuevo usuario; esto se
-- hace proporcionando su contraseña detrás de la cláusula USING.
--
CLP:> GRANT ENCRYPT ON DATABASE TO "xin" USING "hemligt" NEW "notKnown";

-- Volvamos a conectar con el nuevo usuario:
--
CLP:> CONNECT TO samething USER xin USING notknown;
Statement failed [sqlstate = 42505].

-- Esta vez falla porque la contraseña no es igual, por lo que no se
-- generará la misma clave y se deniega el acceso.
--
CLP:> CONNECT TO samething USER ksin USING notKnown;

-- Esta vez no fallará porque el usuario ksin no existe y, por consiguiente,
-- no intentamos autentificar el usuario.
-- Sin embargo, si utilizamos SQLGetInfo podremos distinguir este caso del
-- caso en que un usuario se ha autentificado satisfactoriamente.
--
CLP:> SELECT * FROM swedish;

NUMMER      ORD
----------- --------------------------------
          1 ett
          3 tre
          4 fyra
          5 fem
          7 sju
         99 nittionio
6 row(s) returned.

-- La selección de datos no cifrados funciona bien, sin embargo los datos
-- no cifrados no se pueden leer/actualizar a no ser que haya un usuario
-- autorizado conectado:
--
CLP:> SELECT * FROM english;
Statement failed [sqlstate = 42501].

-- Conectar como el nuevo usuario, finalmente con nombre de usuario y
-- contraseña correctos.
--
CLP:> CONNECT TO samething USER xin USING notKnown;

-- Verificar que estamos autentificados y podemos acceder a los datos.
--
CLP:> SELECT * FROM english;

NUMBER      WORD
----------- --------------------------------
          1 one
          3 three
          4 four
          5 five
          7 seven
         99 ninety nine
6 row(s) returned.

-- Añadir otro usuario:
--
CLP:> GRANT ENCRYPT ON DATABASE TO "thf" USING "notKnown" NEW "heimlich";

-- Listar los usuarios existentes actualmente:
--
CLP:> SELECT username, grantorname FROM "DB2eSYSUSERS";
```

```
USERNAME            GRANTORNAME
------------------- -------------------
jsk                 jsk
xin                 jsk
thf                 xin
3 row(s) returned.

-- Volver a conectar como "jsk":
--
CLP:> CONNECT TO itagain USER jsk USING hemligt;
Statement completed successfully.

-- Intentar cambiar la contraseña por "secret":
--
CLP:> GRANT ENCRYPT ON DATABASE TO "jsk" USING "secret" NEW "secret";
Statement failed [sqlstate = 42506].

-- Ah, hemos fallado porque es necesario que suministremos primero
-- nuestra contraseña antigua, y luego la nueva:
--
CLP:> GRANT ENCRYPT ON DATABASE TO "jsk" USING "hemligt" NEW "secret";

-- Intentarlo con la nueva contraseña:
--
CLP:> CONNECT TO itagain USER jsk USING secret;

-- Asegurarnos de que podemos acceder a datos cifrados:
--
CLP:> SELECT * FROM english;

NUMBER      WORD
----------- --------------------------------
          1 one
          3 three
          4 four
          5 five
          7 seven
         99 ninety nine
6 row(s) returned.

-- Vamos a eliminar el privilegio de cifrado de "xin":
--
CLP:> REVOKE ENCRYPT ON DATABASE FROM "xin";

-- Listar los usuarios
--
CLP:> SELECT username, grantorname FROM "DB2eSYSUSERS";

USERNAME            GRANTORNAME
------------------- -------------------
jsk                 jsk
thf                 xin
2 row(s) returned.

-- Volver a conectar con el usuario que ahora no existe, sin errores.
--
CLP:> CONNECT TO the database USER xin USING idontknow;

-- Intentar realizar operaciones de cifrado sin autorización:
--
CLP:> SELECT * FROM english;
Statement failed [sqlstate = 42501].

CLP:> DROP TABLE english;
Statement failed [sqlstate = 42501].

CLP:> REVOKE ENCRYPT FROM "jsk";
```

```
Statement failed [sqlstate = 42601].

CLP:> GRANT ENCRYPT ON DATABASE TO "xin" USING "idontknow" NEW "idontknow";
Statement failed [sqlstate = 42502].

-- Conectar como "thf":
--
CLP:> CONNECT TO the database USER thf USING heimlich;

-- Comprobar que podemos leer datos cifrados:
--
CLP:> SELECT * FROM english;

NUMBER      WORD
----------- --------------------------------
          1 one
          3 three
          4 four
          5 five
          7 seven
         99 ninety nine
6 row(s) returned.

-- Vamos a eliminar el privilegio del usuario conectado:
--
CLP:> REVOKE ENCRYPT ON DATABASE FROM "thf";

-- Asegurarnos de que ya no puede acceder a los datos:
--
CLP:> SELECT * FROM english;
Statement failed [sqlstate = 42501].

-- Si conectamos con la base de datos como el único usuario que queda, "jsk"
--
CLP:> CONNECT TO the database USER jsk USING secret;

-- Eliminamos el usuario conectado, éste ya no puede acceder a los datos.
-- Realmente, no hay manera de acceder a los datos cifrados de nuevo.
--

CLP:> REVOKE ENCRYPT ON DATABASE FROM "jsk";

-- Asegurarnos de que no queda ningún usuario:
--
CLP:> SELECT username, grantorname FROM "DB2eSYSUSERS";

USERNAME            GRANTORNAME
------------------- -------------------
0 row(s) returned.

-- Ahora no deberíamos poder acceder a los datos cifrados.
--
CLP:> SELECT * FROM english;
Statement failed [sqlstate = 42501].

-- Y así concluye la sesión de ejemplo.
```

Información de consulta para DB2 Everyplace

Esta sección proporciona información de consulta.

Correlaciones de tipos de datos entre DB2 Everyplace y fuentes de datos

Este tema muestra los tipos de datos de réplica y de cliente predeterminados con los que se correlacionan diversos tipos de datos de fuentes de datos de proceso de fondo.

Importante:

1. Debido a las diferencias inherentes entre los tipos de datos no DB2 y los tipos de datos DB2, puede no ser posible la creación de determinadas suscripciones y la duplicación o sincronización de determinados valores.
2. Si un tipo de datos no aparece listado en las tablas de correlaciones de tipos de datos, significa que no está soportado.
3. No puede incluir una tabla en una suscripción si la tabla contiene una columna de un tipo de datos no soportado.

Valores predeterminados soportados para las bases de datos

Las tablas siguientes listan los valores predeterminados que puede tener una columna de una tabla en una fuente de datos. DB2 Everyplace Sync Server puede sincronizar una tabla fuente que tenga columnas con cualquiera de los valores predeterminados listados. DB2 Everyplace Sync Server no puede sincronizar una tabla que tenga columnas con valores predeterminados que no aparezcan listados en la tabla de la fuente de datos.

Nota: Si utiliza IBM Toolbox para el controlador Java para conectarse a DB2 en AS/400, la tabla no podrá tener ninguna columna con valores predeterminados que no sean nulos.

Tabla 16. Valores predeterminados soportados de DB2

Tipo de datos de DB2	Valor predeterminado de DB2
BIGINT	constante, NULL
BLOB	NULL
CHAR(n)	constante, NULL
DATE	fecha actual, NULL
DECIMAL(p,s)	constante, NULL
DOUBLE	constante, NULL
FLOAT	constante, NULL
GRAPHIC(n)	NULL
INTEGER	constante, NULL
LONG VARCHAR	constante, NULL
LONG VARCHAR FOR BIT DATA	NULL
LONG VARGRAPHIC	NULL
REAL	constante, NULL
SMALLINT	constante, NULL

Tabla 16. Valores predeterminados soportados de DB2 (continuación)

TIME	hora actual, NULL
TIMESTAMP	indicación de la hora actual, NULL
VARCHAR(n)	constante, NULL
VARCHAR(n) FOR BIT DATA	NULL
VARGRAPHIC(n)	NULL

Tabla 17. Valores predeterminados soportados de Informix

Tipo de datos de Informix	**Valor predeterminado de Informix**
BLOB	NULL
CHAR	NULL
CHARACTER VARYING(m,r)	NULL
DATE	NULL
DATETIME HOUR TO SECOND	NULL
DATETIME HOUR TO FRACTION	NULL
DATETIME YEAR TO DAY	NULL
DATETIME YEAR TO SECOND	NULL
DATETIME YEAR TO FRACTION	NULL
DATETIME YEAR TO FRACTION(5)	NULL
DECIMAL(p,s)	NULL
DOUBLE PRECISION	NULL
FLOAT(n)	NULL
INT8	NULL
INTEGER	NULL
INTERVAL largest_qualifier(p) TO smallest_qualifier(s)	NULL
LVARCHAR	NULL
MONEY(p,s)	NULL
NCHAR(n)	NULL
NUMERIC(p,s)	NULL
NVARCHAR(m)	NULL
REAL	NULL
SMALLFLOAT	NULL
SMALLINT	NULL
VARCHAR(m)	NULL

Tabla 18. Valores predeterminados admitidos de Oracle

Tipo de datos de Oracle	**Valor predeterminado de Oracle**
BLOB	NULL
CHAR(n)	constante, NULL
DATE	SYSDATE, NULL
NUMBER(p,s)	constante, NULL
RAW(n)	NULL

Tabla 18. Valores predeterminados admitidos de Oracle (continuación)

TIMESTAMP	NULL
VARCHAR2(n)	constante, NULL

Tabla 19. Valores predeterminados soportados de Microsoft SQL Server

Tipo de datos de Microsoft SQL Server	Valor predeterminado de Microsoft SQL Server
BIGINT	constante, NULL
BIT	constante, NULL
CHAR	constante, NULL
DATETIME	NULL
DECIMAL	constante, NULL
FLOAT	constante, NULL
INTEGER	constante, NULL
MONEY	constante, NULL
NCHAR	constante, NULL
NUMERIC	constante, NULL
NVARCHAR	constante, NULL
REAL	constante, NULL
SMALLDATETIME	NULL
SMALLINT	constante, NULL
SMALLMONEY	constante, NULL
TINYINT	constante, NULL
VARBINARY o VARCHAR(MAX)	NULL
VARCHAR	constante, NULL

Correlaciones de tipos de datos de la familia DB2

La Tabla Correlación de tipos de datos fuente de DB2 Universal Database lista la correlación de tipos de datos que se realiza cuando el tipo de datos fuente es un tipo de datos de DB2 Versión 9.1 o DB2 Universal Database (UDB) Versión 8.2.

Tabla 20. Correlación de tipos de datos fuente de DB2 Universal Database

Tipo de datos fuente de DB2 Versión 9.1 y DB2 UDB Versión 8.2	Tipo de datos de réplica de DB2 Versión 9.1	Tipo de datos de dispositivo de DB2 Everyplace	Tipo de datos de dispositivo de IBM Cloudscape versión 10.1 y Derby
BIGINT	BIGINT	VARCHAR	BIGINT
BLOB(n [K|M|G])	BLOB	BLOB	BLOB
CHAR(n)	CHARACTER	CHARACTER	CHARACTER
CHAR(n) FOR BIT DATA	no soportado	no soportado	no soportado
CLOB(n [K|M|G])	no soportado	no soportado	no soportado
DATALINK	no soportado	no soportado	no soportado
DATE	DATE	DATE	DATE
DBCLOB(n [K|M|G])	no soportado	no soportado	no soportado

Tabla 20. Correlación de tipos de datos fuente de DB2 Universal Database (continuación)

Tipo de datos fuente de DB2 Versión 9.1 y DB2 UDB Versión 8.2	Tipo de datos de réplica de DB2 Versión 9.1	Tipo de datos de dispositivo de DB2 Everyplace	Tipo de datos de dispositivo de IBM Cloudscape versión 10.1 y Derby
DECIMAL(p,s)	DECIMAL	DECIMAL	DECIMAL
DOUBLE	FLOAT	VARCHAR	DOUBLE PRECISION
DOUBLE PRECISION	FLOAT	VARCHAR	DOUBLE PRECISION
FLOAT	FLOAT	VARCHAR	DOUBLE PRECISION
GRAPHIC(n)	GRAPHIC	CHARACTER	no soportado
INTEGER	INTEGER	INTEGER	INTEGER
LONG VARCHAR	LONG VARCHAR	VARCHAR	LONG VARCHAR
LONG VARCHAR FOR BIT DATA	LONG VARCHAR FOR BIT DATA	BLOB	LONG VARCHAR FOR BIT DATA
LONG VARGRAPHIC	LONG VARGRAPHIC	VARCHAR	no soportado
REAL	REAL	VARCHAR	REAL
SMALLINT	SMALLINT	SMALLINT	SMALLINT
TIME	TIME	TIME	TIME
TIMESTAMP	TIMESTAMP	TIMESTAMP	TIMESTAMP
VARCHAR(n)	VARCHAR	VARCHAR	VARCHAR
VARCHAR(n) FOR BIT DATA	VARCHAR() FOR BIT DATA	BLOB	VARCHAR(n) FOR BIT DATA
VARGRAPHIC(n)	VARGRAPHIC	VARCHAR	no soportado
XML	no soportado	no soportado	no soportado

Correlaciones de tipos de datos de Informix

La Tabla 21 lista la correlación de tipos de datos que se realizan cuando el tipo de datos fuente es un tipo de datos de Informix.

Tabla 21. Correlación de tipos de datos fuente de Informix

Tipo de datos fuente de Informix	Tipo de datos de réplica de DB2 Versión 9.1	Tipo de datos de dispositivo de DB2 Everyplace	Tipo de datos de dispositivo de IBM Cloudscape versión 10.1	Tipo de datos de dispositivo de Derby
BLOB	BLOB	BLOB	BLOB	BLOB
BOOLEAN	no soportado	no soportado	no soportado	no soportado
BYTE	no soportado	no soportado	no soportado	no soportado
CHAR(n)	CHARACTER	CHARACTER	CHARACTER	CHARACTER
CHARACTER VARYING(m,r)	VARCHAR	VARCHAR	VARCHAR	VARCHAR
CLOB	no soportado	no soportado	no soportado	no soportado
DATE	DATE	DATE	DATE	DATE
DATETIME HOUR TO SECOND	TIME	TIME	TIME	TIME

Tabla 21. Correlación de tipos de datos fuente de Informix (continuación)

Tipo de datos fuente de Informix	Tipo de datos de réplica de DB2 Versión 9.1	Tipo de datos de dispositivo de DB2 Everyplace	Tipo de datos de dispositivo de IBM Cloudscape versión 10.1	Tipo de datos de dispositivo de Derby
DATETIME HOUR TO FRACTION	TIMESTAMP	TIMESTAMP	TIMESTAMP	TIMESTAMP
DATETIME YEAR TO DAY	DATE	DATE	DATE	DATE
DATETIME YEAR TO SECOND	TIMESTAMP	TIMESTAMP	TIMESTAMP	TIMESTAMP
DATETIME YEAR TO FRACTION	TIMESTAMP	TIMESTAMP	TIMESTAMP	TIMESTAMP
DATETIME YEAR TO FRACTION(5)	TIMESTAMP	TIMESTAMP	TIMESTAMP	TIMESTAMP
DECIMAL(p,s)	DECIMAL	DECIMAL	DECIMAL	DECIMAL
DOUBLE PRECISION	DECIMAL	DECIMAL	DECIMAL	DECIMAL
FLOAT(n)	FLOAT	VARCHAR	FLOAT	FLOAT
INT8	BIGINT	VARCHAR	BIGINT	BIGINT
INTEGER	INTEGER	INTEGER	INTEGER	INTEGER
INTERVAL largest_qualifier(p) TO smallest_qualifier(s)	CHARACTER	CHARACTER	CHARACTER	CHARACTER
LVARCHAR	VARCHAR	VARCHAR	VARCHAR	VARCHAR
MONEY(p,s)	DECIMAL	DECIMAL	DECIMAL	DECIMAL
NCHAR(n)	CHARACTER	CHARACTER	CHARACTER	CHARACTER
NUMERIC(p,s)	NUMERIC	DECIMAL	NUMERIC	NUMERIC
NVARCHAR(m)	VARCHAR	VARCHAR	VARCHAR	VARCHAR
REAL	REAL	VARCHAR	REAL	REAL
SERIAL(n)	no soportado	no soportado	no soportado	no soportado
SERIAL8	no soportado	no soportado	no soportado	no soportado
SMALLFLOAT	REAL	VARCHAR	REAL	REAL
SMALLINT	SMALLINT	SMALLINT	SMALLINT	SMALLINT
TEXT	no soportado	no soportado	no soportado	no soportado
VARCHAR(m)	VARCHAR	VARCHAR	VARCHAR	VARCHAR

Correlaciones de tipos de datos de Oracle

La Tabla 22 en la página 50 lista las correlaciones de tipos de datos que se realizan cuando el tipo de datos fuente es un tipo de datos de Oracle.

Tabla 22. Correlación de tipos de datos fuente de Oracle

Tipo de datos fuente de Oracle	Tipo de datos de réplica de DB2	Tipo de datos de dispositivo de DB2 Everyplace	Tipo de datos de dispositivo de IBM Cloudscape versión 10.1 y Derby
BFILE	no soportado	no soportado	no soportado
BLOB	BLOB	BLOB	BLOB
CHAR(n)	CHARACTER	CHARACTER	CHARACTER
CLOB	no soportado	no soportado	no soportado
DATE	TIMESTAMP	TIMESTAMP	no soportado
FLOAT	no soportado	no soportado	no soportado
LONG	no soportado	no soportado	no soportado
LONG RAW	no soportado	no soportado	no soportado
NCHAR(n)	no soportado	no soportado	no soportado
NCLOB	no soportado	no soportado	no soportado
NUMBER(p,s)	DECIMAL	DECIMAL	DECIMAL
NVARCHAR2(n)	no soportado	no soportado	no soportado
RAW(n)	VARCHAR() BIT FOR DATA	BLOB	VARCHAR(n) FOR BIT DATA
REAL	no soportado	no soportado	no soportado
ROWID	no soportado	no soportado	no soportado
TIMESTAMP	TIMESTAMP	TIMESTAMP	TIMESTAMP
UROWID	no soportado	no soportado	no soportado
VARCHAR2(n)	VARCHAR	VARCHAR	VARCHAR

Correlaciones de tipos de datos de Microsoft SQL Server

La Tabla 23 lista las correlaciones de tipos de datos que se realizan cuando el tipo de datos fuente es Microsoft SQL Server. En DB2 Everyplace versión 8.1.4 y anteriores, la correlación para los tipos de datos BIT de Microsoft SQL Server es incoherente entre las suscripciones JDBC y las de subida. En las suscripciones JDBC, el tipo de datos BIT de Microsoft SQL Server se correlaciona con el tipo de datos SMALLINT de DB2 Everyplace. En las suscripciones de subida, el tipo de datos BIT de Microsoft SQL Server se correlaciona con el tipo de datos VARCHAR(1) de DB2 Everyplace. En DB2 Everyplace versión 8.2, el tipo de datos BIT se correlaciona con SMALLINT en ambos casos. Si desea utilizar el método antiguo, que tiene un comportamiento irregular, ejecute el script siguiente y reinicie el DB2 Everyplace Sync Server: `dsysetproperty "DatatypeMappings Generic Target:*" -7="12 VARCHAR"`

Tabla 23. Correlación de tipos de datos de Microsoft SQL Server

Tipo de fuente Microsoft SQL Server	Tipo de datos de réplica de DB2 Versión 9.1	Tipo de datos de dispositivo de DB2 Everyplace	Tipo de datos de dispositivo de IBM Cloudscape versión 10.1 y Derby
BIGINT	BIGINT	VARCHAR	BIGINT
BINARY	no soportado	no soportado	no soportado
BIT	SMALLINT	SMALLINT	SMALLINT
CHAR	CHARACTER	CHARACTER	CHARACTER
CURSOR	no soportado	no soportado	no soportado

Tabla 23. Correlación de tipos de datos de Microsoft SQL Server (continuación)

Tipo de fuente Microsoft SQL Server	Tipo de datos de réplica de DB2 Versión 9.1	Tipo de datos de dispositivo de DB2 Everyplace	Tipo de datos de dispositivo de IBM Cloudscape versión 10.1 y Derby
DATETIME	TIMESTAMP	TIMESTAMP	TIMESTAMP
DECIMAL	DECIMAL	DECIMAL	DECIMAL
FLOAT	FLOAT	VARCHAR	FLOAT
IMAGE	no soportado	no soportado	no soportado
INT	INTEGER	INTEGER	INTEGER
MONEY	DECIMAL	DECIMAL	DECIMAL
NCHAR	GRAPHIC	CHARACTER	CHARACTER
NTEXT	no soportado	no soportado	no soportado
NUMERIC	DECIMAL	DECIMAL	DECIMAL
NVARCHAR	VARGRAPHIC	VARCHAR	VARCHAR
REAL	REAL	VARCHAR	REAL
SMALLDATETIME	TIMESTAMP	TIMESTAMP	TIMESTAMP
SMALLINT	SMALLINT	SMALLINT	SMALLINT
SMALLMONEY	DECIMAL	DECIMAL	DECIMAL
TEXT	no soportado	no soportado	no soportado
TIMESTAMP	no soportado	no soportado	no soportado
TINYINT	SMALLINT	SMALLINT	SMALLINT
UNIQUEIDENTIFIER	no soportado	no soportado	no soportado
VARBINARY o VARBINARY(MAX)	BLOB	BLOB	BLOB
VARCHAR	VARCHAR	VARCHAR	VARCHAR

Restricciones de las correlaciones de tipos de datos

Cuando se llevan a cabo correlaciones de tipos de datos, existen las siguientes restricciones:

- Si se utilizan los tipos de réplica GRAPHIC y VARGRAPHIC de DB2, la base de datos de réplica DB2 se debe crear en DBCS.
- Las aplicaciones para dispositivos portátiles deben asegurarse de que el tipo de datos que se entre en una columna de la tabla de un dispositivo sea compatible con los tipos de columnas de las correspondientes tablas de réplica y fuente y que la longitud de los datos no sea superior a la longitud de las correspondientes columnas de réplica y fuente. Las aplicaciones fuente deben asegurarse de lo mismo con los datos entrados en una columna de una tabla fuente.
- Las columnas de Informix del tipo DECIMAL, NVARCHAR y VARCHAR no se deben definir utilizando la siguiente sintaxis: DECIMAL(p), NVARCHAR(m,r) y VARCHAR(m,r).
- Los tipos de datos siguientes no se pueden utilizar como claves primarias en DB2 Everyplace Sync Server:
  - **Tipo de DB2 Versión 9.1:** LONG VARCHAR, LONG VARCHAR FOR BIT DATA, LONG VARGRAPHIC y VARCHAR() FOR BIT DATA
  - **Tipo de Informix:** DATETIME e INTERVAL
  - **Tipo de Oracle:** RAW
  - **Tipo de SQL Server:** MONEY, REAL y SMALLMONEY

- Si utiliza una base de datos fuente Oracle, el espacio de tabla temporal predeterminado del propietario de las tablas fuente debe ser un espacio de tabla temporal. De lo contrario, si el espacio de tabla temporal predeterminado del propietario es el espacio de tabla SYSTEM o un espacio de tabla permanente gestionado localmente, la duplicación de datos BLOB fallará con el error ORA-03212.

Debido a las diferencias inherentes entre los tipos de datos no DB2 y los tipos de datos DB2, puede no ser posible la creación de determinadas suscripciones y la duplicación o sincronización de determinados valores. Si un tipo de datos no aparece listado en las tablas de correlaciones de tipos de datos, significa que no está soportado.

Restricciones sobre fuentes de datos para suscripciones de DataPropagator Restricciones

Las restricciones siguientes son aplicables a las plataformas Windows y UNIX:

- Antes de poder aplicar cambios, debe primero iniciar el programa Capture para capturar los cambios realizados en la base de datos fuente.
- No cree una suscripción de DataPropagator para tablas que tengan restricciones de integridad de referencia o activadores. De lo contrario, la duplicación fallaría y no se podría recuperar.
- Una base de datos de réplica solamente se debe duplicar con una sola base de datos fuente.
- Para los sistemas fuente z/OS, debe emitir un trabajo para vincular Capture a fin de iniciar el programa Capture en el sistema fuente. Para obtener más información sobre cómo vincular el programa Capture, consulte la documentación de DB2 Versión 9.1.
- Para iSeries:
  - Debe iniciar el programa Capture en la base de datos fuente antes de poder iniciar DB2 Everyplace Sync Server. Utilice el mandato STRDPRCAP para iniciar el programa Capture.
  - No utilice la herramienta para realizar scripts de XML para crear tablas de control para la base de datos fuente en iSeries, pues la herramienta no puede realizar esa función. Es decir, no establezca el atributo **CreateDPropRControlTables** de la etiqueta <AddReplMaster> en TRUE. Debe crear las tablas de control manualmente utilizando el mandato CRTDPRTBL.
  - Debe registrar por diario las tablas fuente de forma manual. Utilice el mandato STRJRNPF para registrar por diario las tablas fuente.

    Nota: Si utiliza la sentencia CREATE SCHEMA para crear una biblioteca que contiene las tablas fuente, el registro por diario se realizará automáticamente.

  Para obtener más información sobre los mandatos, consulte el Centro de información de iSeries y el manual ″SQL Replication Guide and Reference″.
- La base de datos de réplica debe estar situada en el mismo servidor que el motor de duplicación, que es un DB2 Everyplace Sync Server habilitado para la duplicación o un proceso de línea de mandatos donde se ejecuta el script dsyreplicate. El programa Capture de DataPropagator no puede capturar bases de datos remotas.
- Por omisión, DataPropagator crea las tablas de réplica en sus propios espacios de tablas, no gestionados por el sistema, si la suscripción se crea mediante el Centro de duplicación. La herramienta para realizar scripts de XML no crea las tablas en sus propios espacios de tablas, no gestionados por el sistema. Si desea que las tablas de control de DB2 Everyplace asociadas con las tablas de réplica se creen en los mismos espacios de tablas, debe alterar la creación de los espacios de tablas para especificar que el espacio de tablas está gestionado por el sistema, o debe aumentar el tamaño del contenedor de espacios de tablas para dar cabida a estas tablas de control (como recomendación general, aumente el contenedor hasta 5 veces el tamaño generado por Data Propagator). Si no especifica un espacio de tablas en el XML para crear la suscripción de DB2 Everyplace, se utiliza el espacio de tablas predeterminado USERSPACE1 para las tablas de control de DB2 Everyplace.
- Si `Replicate="FALSE"` no se especifica como atributo en la etiqueta AddDProprSubscription, la base de datos de réplica debe ser local con respecto a la máquina donde se ejecuta la herramienta para realizar scripts de XML, y el proceso de captura del fuente debe estar en ejecución.

- Por omisión, el valor CommitCount para una suscripción de tabla de DataPropagator es 0, el cual hace que todos los cambios de duplicación se realicen dentro de una transacción individual. Si se produce un error durante una duplicación, se retrotraen todos los cambios, y cuando se intenta la recuperación, los cambios no se aplican de nuevo. Sin embargo, el uso de esta función hace que aumente el espacio de transacción necesario, especialmente cuando una aplicación produce muchos cambios. Dependiendo del entorno cliente y los requisitos de la aplicación, puede ser necesario establecer CommitCount en un valor positivo o aumentar el espacio de registro de transacciones para permitir un valor de CommitCount igual a 0. Puede establecer el valor de CommitCount en el script XML por el que se crea o modifica la suscripción de tabla de DataPropagator, o puede modificar el valor de CommitCount utilizando el Centro de duplicación de DB2 Versión 9.1.
- El DB2 Everyplace Sync Server no sincroniza correctamente horas ni indicaciones horarias en formato de 24 horas debido a diferencias en la representación del tiempo en Java y en una fuente de datos. La hora ″24:00:00″, por ejemplo, se convierte en ″00:00:00″ y hace que los datos guardados en la base de datos portátil difieran de los datos guardados en la base de datos fuente. Siempre debe evitar el uso de una hora en este formato en sus aplicaciones.
- Si está añadiendo una tabla a una suscripción, sus nombres de esquema, nombres de columna y nombres de tabla para los sistema fuente y de destino no pueden ser palabras clave, palabras reservadas ni registros especiales de SQL o DB2 Versión 9.1.
- Una base de datos de réplica solamente puede duplicar con una sola base de datos fuente. DB2 Everyplace no permite asociar una base de datos de réplica con varias bases de datos fuente.
- En unos pocos casos, esto puede producir una incapacidad para crear una tabla debido a que el nombre de tabla o de columna no es exclusivo. En otros casos raros, esto puede hacer que un carácter individual se convierta en varios caracteres, como en el caso de la “s cerrada” alemana (ß), que se convertirá en “SS”.

Límites de DB2 Everyplace

Límites de base de datos y de SQL de DB2 Everyplace

La Tabla 24 describe las restricciones existentes respecto a la utilización de la base de datos y de SQL para DB2 Everyplace. Si se ajusta al caso más restrictivo, puede ser más fácil migrar sus programas a otras plataformas. Algunos dispositivos portátiles pueden tener restricciones adicionales respecto de estos límites debido a limitaciones de la memoria física y del sistema. Consulte la documentación proporcionada con el dispositivo portátil para obtener más información sobre estas limitaciones. A menos que se indique lo contrario, cada restricción es aplicable a todos los clientes.

Tabla 24. Límites de base de datos y de SQL de DB2 Everyplace

Descripción	Límite
Longitud máxima combinada para columnas INT, SMALLINT, CHAR, DECIMAL, DATE, TIME y TIMESTAMP en un solo registro	32767 bytes
Longitud máxima de una columna BLOB	2 Gigabytes -1 byte
Longitud máxima de una columna CHAR	32767 bytes
Longitud máxima de un sentencia de SQL	64 kilobytes
Longitud máxima de una columna VARCHAR	32767 bytes
Longitud máxima de una restricción de comprobación	32767 bytes
Longitud máxima de un nombre de columna (clientes Cloudscape 10.1 y Derby)	30
Longitud máxima de un nombre de columna (otros clientes)	128
Longitud máxima de un valor predeterminado	32767 bytes
Longitud máxima de una fila en una tabla	64 kilobytes

Tabla 24. Límites de base de datos y de SQL de DB2 Everyplace (continuación)

Descripción	Límite
Longitud máxima de un nombre de tabla	128
Longitud máxima de un identificador	128
Longitud máxima de un nombre de índice (clientes Cloudscape 10.1 y Derby)	18
Longitud máxima de un nombre de índice (otros clientes)	128
Longitud máxima de cada columna en un solo índice	1024 bytes
Número máximo de columnas en una clave foránea	8
Número máximo de columnas en un índice	8
Número máximo de columnas en una clave primaria	8
Número máximo de columnas en una tabla	256
Número máximo de índices en una tabla	15
Número máximo de localizadores de LOB	256
Número máximo de líneas en una tabla	Limitado por el tamaño de la tabla
Número máximo de descriptores de sentencia por conexión	256
Número máximo de tablas en un almacén de datos	65535
Tamaño máximo de un decimal	31 dígitos
Tamaño máximo de un literal	32672 bytes
Tamaño máximo de una tabla (en un sistema de 32 bits)	2 Gigabytes
Año máximo para un valor de fecha	9999
Año mínimo para un valor de fecha	0001

Límites de sincronización de DB2 Everyplace

DB2 Everyplace tiene límites adicionales sobre el tamaño de los campos que el programa sincroniza con el DB2 Everyplace Sync Server. Si la tabla siguiente no muestra el límite de tamaño de sincronización, es el mismo tamaño que el límite mostrado en la Tabla 24 en la página 53. A menos que se indique lo contrario, cada restricción es aplicable a todos los clientes.

Tabla 25. Límites de sincronización de DB2 Everyplace

Descripción	Límite
Longitud máxima de un nombre de columna	30
Longitud máxima de un nombre de tabla (clientes Cloudscape 10.1 y Derby)	126
Longitud máxima de una columna BLOB	4096 kilobytes
Longitud máxima de una fila en una tabla	4608 kilobytes

La sincronización para el tipo de datos BLOB no está soportada en los dispositivos Symbian.

Limitaciones especiales para la longitud de los nombres de tabla

Restricción: La longitud real de los nombres de tabla puede ser menor que 128 caracteres debido a la expansión producida después de que el nombre se convierta a UTF-8.

A partir de DB2 DB2 Everyplace versión 9.1, la longitud máxima de identificador para nombres de tabla, nombres de columna y nombres de usuario ha aumentado de 18 a 128 para las bases de datos recién creadas. Si existe una base de datos más antigua, la longitud máxima del identificador sigue siendo 18.

Restricción: Las versiones anteriores de DB2 Everyplace no dan soporte a los nombres de tabla que sean tan largos como en DB2 Everyplace versión 9.1. Para crear tablas que tengan nombres largos, debe crear la base de datos en DB2 Everyplace versión 9.1. De lo contrario, DB2 Everyplace limita la longitud de los nombres de tabla a la longitud máxima que está permitida en la versión utilizada para crear la tabla.

Palabras reservadas de DB2 Everyplace

Las siguientes palabras reservadas de DB2 Everyplace no se pueden utilizar como identificadores a menos que se especifiquen como identificadores delimitados. Esta restricción también es aplicable a los elementos que se añaden a suscripciones. Los identificadores no pueden ser palabras clave, palabras reservadas ni registros especiales que sean utilizados por:

- SQL
- DB2
- la fuente de datos

Ejemplo:

La sentencia siguiente causa un error de SQL:

```
CREATE TABLE tab1 (select int)
```

La sentencia siguiente no causa un error de SQL:

```
CREATE TABLE tab1 ("select" int)
```

Tabla 26. Palabras reservadas de DB2 Everyplace

ADD	ENCRYPTION	OR
ALGORITHM	ESCAPE	ORDER
ALL	EXCLUSIVE	PRIMARY
ALTER	EXPLAIN	QUERYNO
ALWAYS	FETCH	READ
AND	FOR	REFERENCES
AS	FOREIGN	RELEASE
ASC	FROM	REORG
BEGIN	GENERATED	RETAIN
BIT	GRANT	REVOKE
BLOB	GROUP	ROLLBACK
BY	IDENTITY	SAVEPIONT
CALL	IN	SELECT
CHAR	INDEX	SET
CHARACTER	INSERT	SHARE
CHECK	INT	SMALLINT
COLUMN	INTEGER	START
COMMIT	INTO	TABLE
CONCAT	IS	TIME
CREATE	KEY	TIMESTAMP
CURRENT	LIKE	TO
CURSORS	LIMIT	TRANSACTION
DATA	LOCK	TYPE
DATABASE	LOCKS	UNIQUE
DATE	MODE	UPDATE
DECIMAL	NEW	USING
DEFAULT	NOT	VALUES
DELETE	NULLSYM	VARCHAR
DESC	OF	WHERE
DISTINCT	ON	WITH
DROP	ONLY	WORK
ENCRYPT		

A efectos de compatibilidades futuras, no utilice como identificadores las palabras reservadas de IBM SQL e ISO/ANSI SQL92 siguientes. Las palabras reservadas de IBM SQL que actualmente no son utilizadas por DB2 Everyplace son las siguientes:

Tabla 27. Palabras reservadas de IBM SQL que actualmente no son utilizadas por DB2 Everyplace

ACQUIRE	DISCONNECT	LONG	RESULT
AFTER	DO	LOOP	RETURN
ALIAS	DOUBLE	MAX	RETURNS
ALLOCATE	DSSIZE	MICROSECOND	RIGHT
ALLOW	DYNAMIC	MICROSECONDS	ROW
ANY	EDITPROC	MIN	ROWS
ASUTIME	ELSE	MINUTE	RRN
AUDIT	ELSEIF	MINUTES	RUN
AUTHORIZATION	END	MODIFIES	SCHEDULE
AUX	END-EXEC	MONTH	SCHEMA
AUXILIARY	ERASE	MONTHS	SCRATCHPAD
AVG	EXCEPT	NAME	SECOND
BEFORE	EXCEPTION	NAMED	SECONDS
BETWEEN	EXECUTE	NHEADER	SECQTY
BINARY	EXISTS	NO	SECURITY
BUFFERPOOL	EXIT	NODEN	SIMPLE
CALLED	EXTERNAL	AME	SOME
CAPTURE	FENCED	NODENUMBER	SOURCE
CASCADED	FIELDPROC	NULLS	SPECIFIC
CASE	FILE	NUMPARTS	SQL
CAST	FINAL	OBID	STANDARD
CCSID	FREE	OPEN	STATIC
CLOSE	FULL	OPTIMIZATION	STATISTICS
CLUSTER	FUNCTION	OPTIMIZE	STAY
COLLECTION	GENERAL	OPTION	STOGROUP
COLLID	GO	OUT	STORES
COMMENT	GOTO	OUTER	STORPOOL
CONDITION	GRAPHIC	PACKAGE	STYLE
CONNECT	HANDLER	PAGE	SUBPAGES
CONNECTION	HAVING	PAGES	SUBSTRING
CONSTRAINT	HOUR	PARAMETER	SUM
CONTAINS	HOURS	PART	SYNONYM
CONTINUE	IDENTIFIED	PARTITION	TABLESPACE
COUNT	IF	PATH	THEN
COUNT_BIG	IMMEDIATE	PCTFREE	TRIGGER
CROSS	INDICATOR	PCTINDEX	TRIM
CURRENT_DATE	INNER	PIECESIZE	UNDO
CURRENT_LC_PATH	INOUT	PLAN	UNION
CURRENT_PATH	INSENSITIVE	POSITION	UNTIL
CURRENT_SERVER	INTEGRITY	PRECISION	USAGE
CURRENT_TIME	INTERSECT	PREPARE	USER
CURRENT_TIMESTAMP	ISOBID	PRIQTY	USING
CURRENT_TIMEZONE	ISOLATION	PRIVATE	VALIDPROC
CURRENT_USER	JAVA	PRIVILEGES	VARIABLE
DAY	JOIN	PROCEDURE	VARIANT
DAYS	LABEL	PROGRAM	VCAT
DBA	LANGUAGE	PSID	VIEW
DBINFO	LC_CTYPE	PUBLIC	VOLUMES
DBSPACE	LEAVE	READS	WHEN
DB2GENERAL	LEFT	RECOVERY	WHILE
DB2SQL	LINKTYPE LOCAL	RENAME	WLM
DECLARE	LOCALE	REPEAT	WRITE
DESCRIPTOR	LOCATOR	RESET	YEAR
DETERMINISTIC	LOCATORS	RESOURCE	YEARS
DISALLOW	LOCKSIZE	RESTRICT	

Tabla 28. Palabras reservadas de SQL92 de ISO/ANSI que no son utilizadas por IBM SQL

ABSOLUTE	FIRST	PAD
ACTION	FLOAT	PARTIAL
ARE	FOUND	PRESERVE
ASSERTION	FULL	PRIOR
AT	GET	REAL
BIT_LENGTH	GLOBAL	RELATIVE
BOTH	IDENTITY	SCROLL
CATALOG	INITIALLY	SESSIONSESSION_USER
CHAR_LENGTH	INPUT	SIZE
CHARACTER_LENGTH	INTERVAL	SPACESQLCODE
COALESCE	LAST	SQLERROR
COLLATE	LEADING	SQLSTATE
COLLATION	LEVEL	SYSTEM_USER
CONSTRAINTS	LOWER	TEMPORARY
CONVERT	MATCH	TIMEZONE_HOUR
CORRESPONDING	MODULE	TIMEZONE_MINUTE
DEALLOCATE	NAMES	TRAILING
DEC	NATIONAL	TRANSLATION
DEFERRABLE	NATURAL	TRUE
DEFERRED	NCHAR	UNKNOWN
DESCRIBE	NEXT	UPPER
DIAGNOSTICS	NULLIF	VALUE
DOMAIN	NUMERIC	VARYING
EXEC	OCTET_LENGTH	WHENEVER
EXTRACT	OUTPUT	ZONE
FALSE	OVERLAPS	

Visión general de las tablas de las bases de datos portátiles DB2 Everyplace

Una base de datos portátil DB2 Everyplace comprende varias tablas de catálogo del sistema y tablas definidas por el usuario.

Cada una de las tablas se almacena en dos archivos: uno para los datos propiamente dichos y otro para los índices. Todos los índices se conservan en el mismo archivo de índices. A diferencia de DB2 Versión 9.1, las bases de datos portátiles DB2 Everyplace no tienen nombre y no se pueden catalogar ni descatalogar. Por lo tanto, el nombre de la base de datos se pasa por alto.

Una base de datos portátil DB2 Everyplace no es más que un conjunto de archivos que se pueden copiar o mover a otra ubicación. Una base de datos portátil DB2 Everyplace debe contener las siguientes tablas de catálogos del sistema:

- DB2eSYSTABLES
- DB2eSYSCOLUMNS
- DB2eSYSINDEXES
- DB2eSYSRELS
- DB2eSYSUSERS (esta tabla se crea si se utiliza cifrado local de los datos)

Las tablas de catálogos del sistema contienen metadatos sobre tablas definidas por el usuario. Por ejemplo, si elimina archivos de una tabla definida por el usuario sin suprimir la entrada correspondiente en las tablas de catálogos, ocasionará una incoherencia.

Para acceder a las tablas de catálogos de una consulta, deberá utilizar los identificadores delimitados. Por ejemplo, la consulta siguiente devolverá el valor 1 si existe la tabla T:

```
SELECT 1 FROM "DB2eSYSTABLES" WHERE TNAME = 'T'
```

Tablas base del catálogo del sistema de DB2 Everyplace

El gestor de bases de datos crea y mantiene un conjunto de tablas base del catálogo del sistema. Este apéndice contiene una descripción de cada tabla base del catálogo del sistema, incluyendo nombres de columnas y tipos de datos.

Todas las tablas base del catálogo del sistema las crea el gestor de bases de datos. Las tablas base del catálogo del sistema no se pueden crear ni eliminar explícitamente. Las tablas base del catálogo del sistema se actualizan durante el funcionamiento normal en respuesta a las sentencias SQL de definición de datos, rutinas de entorno y ciertos programas de utilidad. Los datos en las tablas base del catálogo del sistema están disponibles mediante programas de utilidad de búsqueda SQL normales. Las tablas base del catálogo del sistema no se pueden modificar utilizando mandatos de manipulación de datos de SQL normales. Para poder acceder a las tablas de catálogo del sistema es necesario utilizar un identificar delimitado.

Tabla 29. Tablas base del catálogo del sistema

Descripción	Tabla base del catálogo
tablas	"DB2eSYSTABLES" en la página 23
columnas	"DB2eSYSCOLUMNS" en la página 23
índices	"DB2eSYSINDEXES" en la página 24
restricciones referenciales	"DB2eSYSRELS" en la página 24
usuarios	"DB2eSYSUSERS" en la página 24

DB2eSYSTABLES

Esta tabla base del catálogo del sistema contiene una fila para cada tabla que se crea. Todas las tablas de catálogo tienen entradas en el catálogo DB2eSYSTABLES.

Tabla 30. Tabla base del catálogo del sistema DB2eSYSTABLES

Nombre de la columna	Tipo de datos	Admite nulos	Descripción
TNAME	VARCHAR (129)		Nombre de tabla
NUMCOLS	INTEGER (4)		Número de columnas
FLAGS	INTEGER (4)		(Sólo uso interno)
NUMKEY	INTEGER (4)		Número de columnas en la clave primaria
CHK	BLOB (32767)	Sí	Restricción de comprobación (sólo uso interno)
IDXINFO	BLOB (4096)	Sí	Índice (sólo uso interno)
NUMREFS	INTEGER (4)	Sí	Clave primaria y foránea (sólo uso interno)
F_ID	INTEGER (4)	Sí	(Sólo uso interno)
PD	BLOB (4096)	Sí	(Sólo uso interno)

DB2eSYSCOLUMNS

Esta tabla base del catálogo del sistema contiene una fila para cada columna que se define para una tabla.

Tabla 31. Tabla base del catálogo del sistema DB2eSYSCOLUMNS

Nombre de la columna	Tipo de datos	Admite nulos	Descripción
CNAME	VARCHAR (129)		Nombre de la columna

Tabla 31. Tabla base del catálogo del sistema DB2eSYSCOLUMNS (continuación)

Nombre de la columna	Tipo de datos	Admite nulos	Descripción
TNAME	VARCHAR (129)		Nombre de tabla
TYPE	INTEGER (4)		Tipo de datos
ATTR	INTEGER (4)		(Sólo uso interno)
LENGTH	INTEGER (4)		Longitud de la columna
POS	INTEGER (4)		Número de columna
FLAGS	INTEGER (4)		(Sólo uso interno)
KEYSEQ	INTEGER (4)		Posición ordinal de la columna en la clave primaria
SCALE	INTEGER (4)		Escala para columna decimal
DEF	VARCHAR (32767)	Sí	Valor predeterminado (uso interno)

DB2eSYSINDEXES

La tabla DB2eSYSINDEXES se crea durante la iniciación de la base de datos. Esta tabla base del catálogo del sistema contiene una fila para cada columna de cada índice que se crea.

Tabla 32. Tabla base del catálogo del sistema DB2eSYSINDEXES

Nombre de la columna	Tipo de datos	Admite nulos	Descripción
INAME	VARCHAR (129)		Nombre de índice
TNAME	VARCHAR (129)		Nombre de tabla
CNAME	VARCHAR (129)		Nombre de la columna
NONUNIQUE	SMALLINT (4)		El índice es exclusivo o no.
TYPE	SMALLINT (4)		Tipo de índice
ORDINALPOSITION	SMALLINT (4)		Posición ordinal de la columna en la tabla
ASCORDESC	CHAR (1)		El orden en que se clasifican los valores de las columnas que forman el índice, ascendente o descendente
FILEIDX	INTEGER (4)		(Sólo uso interno)
RESERVED	BLOB (4096)	Sí	(Sólo uso interno)

DB2eSYSRELS

Esta tabla base del catálogo del sistema contiene una fila para cada restricción referencial.

Tabla 33. Tabla base del catálogo del sistema DB2eSYSRELS

Nombre de la columna	Tipo de datos	Admite nulos	Descripción
RMD_ID	INTEGER (4)		Clave primaria y foránea (sólo uso interno)
PKTABLE_NAME	VARCHAR (129)		Nombre de tabla padre
PKCOLUMN_NAME	VARCHAR (129)		Columna de clave primaria de la tabla padre
FKTABLE_NAME	VARCHAR (129)		Nombre de tabla hijo
FKCOLUMN_NAME	VARCHAR (129)		Nombre de la columna de clave foránea de la tabla hijo

Tabla 33. Tabla base del catálogo del sistema DB2eSYSRELS (continuación)

Nombre de la columna	Tipo de datos	Admite nulos	Descripción
ORDINAL_POSITION	INTEGER (4)		Posición de la columna en la referencia de clave foránea

DB2eSYSUSERS

La tabla DB2eSYSUSERS se crea automáticamente cuando se crea la primera tabla cifrada o cuando se ejecuta la primera sentencia GRANT. Esta tabla está estrechamente ligada a la base de datos y a los datos cifrados; no se puede mover a otra base de datos DB2 Everyplace que contenga datos cifrados distintos.

Esta tabla base del catálogo del sistema contiene una fila para cada nombre de usuario registrado que se define para una base de datos.

Tabla 34. Tabla base del catálogo del sistema DB2eSYSUSERS

Nombre de la columna	Tipo de datos	Admite nulos	Descripción
USERNAME	VARCHAR (129)		Parte de la clave primaria, sensible a las mayúsculas/minúsculas. El nombre del usuario asociado a esta fila.
DATABASENAME	VARCHAR (129)		Para uso futuro. Se almacena una serie vacía. Parte de la clave primaria.
TABLENAME	VARCHAR (129)		Para uso futuro. Se almacena una serie vacía. Parte de la clave primaria.
ENCMETHOD	VARCHAR (129)		Para uso futuro. Se almacena una serie vacía. Parte de la clave primaria.
PRIVILEGES	VARCHAR (129)	Sí	Define privilegios del usuario. Actualmente sólo se admite el valor 'E', que indica cifrado.
ENCKEYDATA	BLOB (280)	Sí	Se utiliza para regenerar la clave de cifrado.
ATTIME	TIMESTAMP (26)	Sí	Hora en que se ha añadido el usuario o se ha modificado por última vez el registro, la que sea más reciente de las dos.
VALIDATE	BLOB (280)	Sí	Verifica que el registro es auténtico y que el usuario lo ha añadido un usuario autentificado.
GRANTOR	VARCHAR (129)	Sí	El nombre del usuario que ha registrado el nombre de usuario de la columna 1.
INTERNALDATA	BLOB (255)	Sí	(Uso futuro interno)

Interfaces

Este tema describe las interfaces proporcionadas por DB2 Everyplace.

Interfaz de DB2 Everyplace Sync Client

Este tema describe las funciones proporcionadas por el DB2 Everyplace Sync Client.

Sistemas operativos soportados por las API de Java Sync

Las API de Java Sync están disponibles en los sistemas operativos siguientes:

- Windows (Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP, Windows 2003)

- Windows CE (con procesadores MIPS y ARM)
- Linux

API de IBM Java Sync

Puede crear aplicaciones Java utilizando Java Database Connectivity (JDBC) y la interfaz Java de IBM Sync a fin de integrar la base de datos portátil DB2 Everyplace y la funcionalidad de DB2 Everyplace Sync Server.

Para obtener información detallada sobre las interfaces, clases y excepciones que se proporcionan con las API de IBM Java Sync compatibles con DB2 Everyplace, consulte la documentación Javadoc situada en el directorio <DSYPATH>\Clients\javadoc, donde <DSYPATH> es el directorio donde está instalado DB2 Everyplace.

Interfaz JDBC

Este tema describe los métodos proporcionados por la Interfaz JDBC.

Visión general del soporte de JDBC de DB2 Everyplace

DB2 Everyplace soporta un subconjunto de los métodos definidos en la especificación de la API Java Database Connectivity (JDBC) que se proporciona en Sun Java Developer's Kit. La información sobre los métodos JDBC que se pueden utilizar con DB2 Everyplace está modificada con respecto a la documentación de Java Development Kit Versión 1.4.1 de Sun. DB2 Everyplace también da soporte a las interfaces Connection y Statement ampliadas.

Consulte los apartados "Clase DB2eConnection" en la página 64 para obtener más información.

El paquete opcional de JDBC para CDC/Foundation Profile (JSR 169) se puede utilizar para ejecutar aplicaciones Java para DB2 Everyplace. Sin embargo, DB2 Everyplace no es compatible con todos los métodos definidos en JSR 169 (por ejemplo, DB2 Everyplace no puede trabajar con el tipo CLOB, por lo que no puede utilizar la interfaz Clob).

Restricciones para suscripciones de tabla

Existen varias restricciones en DB2 Everyplace respecto a las suscripciones de tabla. Tenga en cuenta las limitaciones siguientes cuando planifique y despliegue sus aplicaciones portátiles.

- Si está utilizando una base de datos fuente que se ejecuta en Informix Dynamic Server, debe utilizar un calificador de esquema explícito para crear cada tabla de una suscripción.
- Si está añadiendo una tabla a una suscripción, sus nombres de esquema, nombres de columna y nombres de tabla para los sistema fuente y de destino no pueden ser palabras clave, palabras reservadas ni registros especiales de SQL, DB2 Versión 9.1 ni de la base de datos fuente que utilice.
- Si es necesario cambiar la estructura de una tabla fuente mediante una sentencia ALTER TABLE o DROP TABLE, y una sentencia CREATE TABLE, siga estos pasos:
  1. Elimine la tabla en todas las suscripciones.
  2. Ejecute la sentencia ALTER TABLE o DROP TABLE, y la sentencia CREATE TABLE.
  3. Añada de nuevo la tabla a las suscripciones.
- No puede utilizar la función de cifrado para más de una base de datos de destino con DB2 Everyplace Sync Client.
- Los clientes IBM Cloudscape y Derby no dan soporte a las siguientes características soportadas por el cliente DB2 Everyplace: varios servidores, clasificación de conjuntos de suscripciones/suscripciones/tablas, y cifrado de datos locales.
- La integridad referencial no se puede utilizar para las suscripciones de tabla de DataPropagator.

- Duplique siempre las suscripciones de tabla DataPropagator en el servidor de bases de datos de réplica. Esto significa que si la duplicación se debe realizar durante acciones administrativas, el Centro de administración de dispositivos portátiles se debe ejecutar en el servidor de bases de datos de réplica.
- Una suscripción DataPropagator no puede utilizar la misma base de datos de réplica que una suscripción JDBC.
- DB2 Everyplace no da soporte a nombres de objeto de base de datos que deben estar encerrados entre comillas dobles.
- En general, el tamaño máximo de una fila de una tabla lo limita la fuente de datos. La adición de la tabla a una suscripción a JDBC o DataPropagator restringe aún más el tamaño máximo de fila. La restricción adicional sobre el tamaño máximo de fila es de aproximadamente 125 bytes.
- El DB2 Everyplace Sync Server no sincroniza correctamente horas ni indicaciones horarias en formato de 24 horas debido a diferencias en la representación del tiempo en Java y en una fuente de datos. La hora ″24:00:00″ se convierte en ″00:00:00″ y hace que los datos guardados en la base de datos portátil difieran de los datos guardados en la base de datos fuente. Siempre debe evitar el uso de una hora en este formato en sus aplicaciones.
- Una base de datos de réplica solamente puede duplicar con una sola base de datos fuente. DB2 Everyplace no permite asociar una base de datos de réplica con varias bases de datos fuente.
- DB2 Everyplace convierte a mayúsculas los nombres de tabla y de columna de la base de datos fuente cuando aplica esos nombres en las bases de datos de réplica y de destino. En unos pocos casos, esto puede producir una incapacidad para crear una tabla debido a que el nombre de tabla o de columna no es exclusivo. En otros casos raros, esto puede hacer que un carácter individual se convierta en varios caracteres, como en el caso de la “s cerrada” alemana (ß), que se convertirá en “SS”.
- Cuando se crea una suscripción JDBC en DB2 Versión 9.1 para z/OS, DB2 Everyplace crea tablas e índices en estas tablas en la base de datos fuente. DB2 Everyplace crea los índices con los atributos predeterminados. Por lo tanto, DB2 Versión 9.1 para z/OS asigna los conjuntos de datos de índices a los grupos de almacenamiento predeterminados y les establece atributos de espacio predeterminados. Debido a que esto podría afectar al rendimiento de las inserciones masivas, puede mejorar el rendimiento modificando los atributos de los índices que comienzan con el prefijo DSY.

  Ejemplo: para modificar los atributos del índice DSYI19099874167 y convertirlos a los parámetros siguientes:

  – Capacidad máxima de direccionamiento para cada archivo: 256 megabytes
  – Grupo de almacenamiento del archivo: DB2ERSTG
  – Asignación primaria de espacio mínima para el archivo: 80 000 kilobytes
  – Asignación secundaria de espacio mínima para el archivo: 40 000 kilobytes

  Emita este mandato:

```
ALTER INDEX DSYI19099874167 PIECESIZE 256M using
  STOGROUP DB2ERSTG
  PRIQTY 80000
  SECQTY 40000
```

Restricciones de integridad referencial

Si las tablas fuente tienen restricciones de integridad referencial, tenga en cuenta las restricciones siguientes para evitar errores de sincronización y duplicación debidos a violaciones de las restricciones de integridad referencial:

- No actualice las claves primarias en la base de datos del cliente o en el dispositivo portátil.
- No suscriba tablas que tengan relaciones padre-hijo en las que intervengan ciclos (por ejemplo, bucles internos).
- Cuando cree suscripciones, añada tablas en el orden padre a hijo.
- Las relaciones de integridad referencial no pueden cruzar los límites de ninguna suscripción.

- No realice suscripciones a tablas que tengan activadores, a menos que las tablas sean el destino de una suscripción de carga.

Interfaz com.ibm.db2e.jbdc

Este tema describe las clases proporcionadas por el paquete com.ibm.db2e.jbdc de JDBC.

Clase DB2eConnection:

La clase DB2eConnection obtiene y establece determinados atributos de Connection. Para utilizar los métodos de la clase DB2eConnection sobre un objeto Connection, antes se tiene que convertir el objeto Connection en un objeto DB2eConnection. Estos métodos se implementan mediante llamadas a las funciones CLI/ODBC de SQLGetConnectAttr y SQLSetConnectAttr con los argumentos apropiados.

paquete com.ibm.db2e.jdbc

clase pública **DB2eConnection**

implanta Connection

La tabla siguiente lista los métodos de la clase DB2eConnection.

Tabla 35. Métodos de la clase DB2eConnection

Tipo de retorno del método	Método
`int`	**getLockTimeout()**Devuelve el número de segundos que transcurrirán antes de que se exceda el tiempo de espera de una petición de bloqueo.
`int`	**getBufferpoolSize()** Devuelve el tamaño, en bytes, de la agrupación de almacenamientos intermedios de la conexión DB2 Everyplace.
`boolean`	**isEnabledAccessHiddenVolume()**Devuelve el valor false.
`boolean`	**isEnabledFilenameFormat83()** Devuelve el valor true si el motor de base de datos crea nombres de archivo en formato 8.3. Devuelve el valor false si crea nombres de archivo en formato largo.
`boolean`	**isEnabledIOWritethrough()**Devuelve el valor true si el motor transfiere los cambios al disco inmediatamente.
`boolean`	**isEnabledSharedDatabaseAccess()** Devuelve el valor true si la conexión permite el acceso compartido a tablas.
`boolean`	**isEnabledTableChecksum()** Devuelve true si la base de datos utiliza archivos con sumas de comprobación.

Interfaz Java.sql

Este tema describe los paquetes proporcionados por el paquete java.sql.

Interfaz Blob:

La interfaz Blob representa (correlaciona) un BLOB de SQL en el lenguaje de programación Java™. Un BLOB de SQL es un tipo incorporado que almacena un objeto binario grande como valor de columna en una fila de una tabla de base de datos. Un objeto Blob es válido mientras dura la transacción en la que se ha creado.

Los métodos de las interfaces ResultSet y PreparedStatement, como por ejemplo getBlob y setBlob, permite que un programador acceda al BLOB de SQL. La interfaz Blob proporciona métodos para obtener la longitud del valor de un BLOB (objeto binario grande) de SQL y para materializar el valor de un BLOB en el cliente.

paquete java.sql

interfaz pública **Blob**

La Tabla 36 lista los métodos de la interfaz Blob a los que DB2 Everyplace da soporte.

Tabla 36. Métodos de la interfaz Blob

Tipo de valor de retorno del método	Método
`InputStream`	**getBinaryStream()** Recupera, como corriente de datos, el BLOB designado por esta instancia de Blob.
`byte[]`	**getBytes(long pos, int length)** Devuelve, como matriz de bytes, todo o parte del valor de BLOB designado por este objeto Blob.
`long`	**length()** Devuelve el número de bytes del valor de BLOB designado por este objeto Blob.
`long`	**position(Blob pattern, long start)** Recupera la posición de byte del valor de BLOB designado por este objeto Blob donde comienza el patrón.
`long`	**position(byte[] pattern, long start)** Recupera la posición de byte en la que comienza el patrón de matriz de bytes especificado dentro del valor de BLOB representado por este objeto Blob.
`OutputStream`	**setBinaryStream(long pos)** Recupera una corriente de datos que se puede utilizar para escribir en el valor de BLOB representado por este objeto Blob.
`void`	**setBinaryStream(int parameterIndex, InputStream x, int length)**
`int`	**setBytes(long pos, byte[] bytes)** Escribe la matriz de bytes proporcionada en el valor de BLOB representado por este objeto Blob, comenzando en la posición pos, y devuelve el número de bytes escritos.
`int`	**setBytes(long pos, byte[] bytes, int offset, int len)** Escribe la matriz de bytes proporcionada completa o parte de ella en el valor de BLOB representado por este objeto Blob y devuelve el número de bytes escritos.
`void`	**truncate(long len)** Trunca el valor de BLOB representado por este objeto Blob para que su longitud sea len bytes.

Interfaz CallableStatement:

La interfaz utilizada para ejecutar procedimientos almacenados SQL remotos. El parámetro de resultados debe registrarse como parámetro OUT. Los demás parámetros pueden utilizarse para la entrada, la salida o para ambos. A los parámetros se les hace referencia de modo secuencial, por número. El primer parámetro es 1.

```
call <nombre_procedimiento> (?,?, ...)
```

Los valores del parámetro IN se establecen utilizando los métodos de definición heredados de PreparedStatement. El tipo de todos los parámetros OUT deben registrarse antes de ejecutar el

procedimiento almacenado; sus valores se recuperan después de la ejecución por medio de los métodos get que se proporcionan en este punto. El tamaño del parámetro de salida se limita a 4K bytes.

`CallableStatement` puede devolver un `ResultSet`.

paquete java.sql

interfaz pública **CallableStatement**

amplía PreparedStatement

La Tabla 37 lista los métodos de la interfaz CallableStatement a los que DB2 Everyplace da soporte.

Tabla 37. Métodos de la interfaz CallableStatement

Tipo de valor de retorno del método	Método
`BigDecimal`	**getBigDecimal(int parameterIndex)** JDBC 2.0 - Obtiene el valor de un parámetro decimal de JDBC en forma de objeto `BigDecimal` en el lenguaje de programación Java.
`Blob`	**getBlob(int i)** JDBC 2.0 Obtiene el valor de un parámetro BLOB de JDBC en forma de objeto `Blob` en el lenguaje de programación Java.
`byte[]`	**getBytes(int parameterIndex)** Obtiene el valor de un parámetro BINARY o VARBINARY de JDBC como matriz de valores `byte` en el lenguaje de programación de Java.
`Date`	**getDate(int parameterIndex)** Obtiene el valor de un parámetro DATE de JDBC en forma de objeto `java.sql.Date`.
`int`	**getInt(int parameterIndex)** Obtiene el valor de un parámetro INTEGER de JDBC en forma de `int` en el lenguaje de programación de Java.
`Object`	**getObject(int parameterIndex)** Obtiene el valor de un parámetro en forma de `object` en el lenguaje de programación de Java.
`short`	**getShort(int parameterIndex)** Obtiene el valor de un parámetro SMALLINT de JDBC en forma de `short` en el lenguaje de programación de Java.
`String`	**getString(int parameterIndex)** Recupera el valor de un parámetro CHAR, VARCHAR o LONGVARCHAR de JDBC en forma de `String` en el lenguaje de programación de Java.
`Time`	**getTime(int parameterIndex)** Obtiene el valor de un parámetro TIME de JDBC en forma de objeto `java.sql.Time`.
`Timestamp`	**getTimestamp(int parameterIndex)** Obtiene el valor de un parámetro TIMESTAMP de JDBC en forma de objeto `java.sql.Timestamp`.
`void`	**registerOutParameter(int parameterIndex, int sqlType)** Registra el parámetro OUT en la posición ordinal `parameterIndex` para el tipo de JDBC `sqlType`.
`boolean`	**wasNull()** Indica si el último parámetro OUT leído tenía o no el valor NULL de SQL.

Interfaz Connection:

La interfaz Connection establece una conexión (sesión) con una base de datos específica. En el contexto de una conexión, se ejecutan sentencias de SQL y se devuelven resultados.

Una base de datos de Connection puede proporcionar información que describe sus tablas, su gramática de SQL soportada, sus procedimientos almacenados, las posibilidades de esta conexión, etc. Esta información se obtiene mediante el método getMetaData.

paquete java.sql

interfaz pública **Connection**

La Tabla 38 lista los métodos de la interfaz Connection a los que DB2 Everyplace da soporte.

Tabla 38. Métodos de la interfaz Connection

Tipo de valor de retorno del método	Método
`void`	**clearWarnings()** Borra todos los avisos de los que se ha informado para este objeto `Connection`.
`void`	**close()** Libera una base de datos de `Connection` y los recursos de JDBC de modo inmediato en vez de esperar que se liberen de modo automático.
`void`	**commit()** Hace que todos los cambios resultantes de la operación anterior de confirmación o retrotracción sean permanentes y libera los bloqueos de base de datos mantenidos actualmente por `Connection`.
`Blob`	**createBlob()** Crea un objeto `Blob`. El objeto devuelto no contiene ningún dato. Para añadir datos al objeto `Blob`, utilice los métodos `setBinaryStream` y `setBytes` de la interfaz `Blob`.
`Statement`	**createStatement()** Crea un objeto `Statement` para enviar sentencias de SQL a la base de datos.
`Statement`	**createStatement(int resultSetType, int resultSetConcurrency)** JDBC 2.0. Crea un objeto `Statement` que generará objetos `ResultSet` con el tipo y la simultaneidad indicados.
`boolean`	**getAutoCommit()** JDBC 4.0 Recupera la modalidad de confirmación automática para el objeto Connection.
`DatabaseMetaData`	**getMetaData()** Obtiene los metadatos correspondientes a la base de datos de este objeto `Connection`.
`int`	**getTransactionIsolation()** Obtiene el nivel de aislamiento de transacción de este objeto `Connection`.
`SQLWarning`	**getWarnings()** Devuelve el primer aviso notificado por llamadas a este objeto `Connection`.
`boolean`	**isClosed()** Comprueba si una conexión (`Connection`) está cerrada.
`CallableStatement`	**prepareCall(String sql)** Crea un objeto `CallableStatement` para llamar a procedimientos almacenados de base de datos.
`PreparedStatement`	**prepareStatement(String sql)** Crea un objeto `PreparedStatement` para enviar sentencias de SQL parametrizadas a la base de datos.

Tabla 38. Métodos de la interfaz Connection (continuación)

Tipo de valor de retorno del método	Método
PreparedStatement	**prepareStatement(String sql, int resultSetType, int resultSetConcurrency)** JDBC 2.0. Crea un objeto PreparedStatement que generará objetos ResultSet con el tipo y la simultaneidad indicados.
Savepoint	**setSavepoint()** Crea un punto de rescate sin nombre en la transacción actual y devuelve el nuevo objeto Savepoint correspondiente al punto de rescate.
Savepoint	**setSavepoint(String name)** Crea un punto de rescate con el nombre especificado en la transacción actual y devuelve el nuevo objeto Savepoint correspondiente al punto de rescate.
void	**rollback()** Elimina todos los cambios resultantes de la operación anterior de confirmación o retrotracción y libera los bloqueos de base de datos mantenidos actualmente por el objeto Connection.
void	**releaseSavepoint(Savepoint savepointname)** Elimina de la transacción actual el objeto Savepoint proporcionado.
void	**rollback(Savepoint savepointname)** Elimina todos los cambios que se hicieron después de definir el objeto Savepoint.
void	**setAutoCommit(boolean autoCommit)** Establece la modalidad de confirmación automática de este objeto Connection.
void	**setTransactionIsolation(int level)** Intenta cambiar el nivel de aislamiento de transacción de este objeto Connection.

Interfaz DatabaseMetaData:

La interfaz DatabaseMetaData proporciona información amplia sobre la base de datos como un todo.

Algunos de estos métodos toman argumentos String para los nombres de catálogo y de esquema. DB2 Everyplace pasa por alto estos argumentos.

Algunos de los métodos aquí contenidos devuelven listas de información en forma de objetos ResultSet. Puede utilizar los métodos ResultSet normales, como por ejemplo getString y getInt, para recuperar los datos de estos ResultSets.

Si no se dispone de un formulario de metadatos determinado, estos métodos emitirán una SQLException.

paquete java.sql

interfaz pública **DatabaseMetaData**

La Tabla 39 en la página 69 lista los campos de la interfaz DatabaseMetaData a los que DB2 Everyplace da soporte.

Tabla 39. Campos de DatabaseMetaData

Tipo de campo	Campo
`static int`	**columnNoNulls** Indica que es posible que la columna no admita valores nulos (NULL).
`static int`	**columnNullable** Indica que la columna permite definitivamente valores nulos (NULL).
`static int`	**columnNullableUnknown** Indica que la anulación de columnas es desconocida.

La Tabla 40 lista los métodos de la interfaz DatabaseMetaData a los que DB2 Everyplace da soporte.

Tabla 40. Métodos de la interfaz DatabaseMetaData

Tipo de valor de retorno del método	Método
`boolean`	**allProceduresAreCallable()** ¿Puede el usuario actual llamar a todos los procedimientos devueltos por el método getProcedures?
`boolean`	**allTablesAreSelectable()** ¿Puede el usuario actual utilizar todas las tablas devueltas por el método getTables en una sentencia SELECT?
`boolean`	**dataDefinitionCausesTransactionCommit()** ¿Puede una sentencia de definición de datos de una transacción forzar la confirmación de la transacción?
`boolean`	**dataDefinitionIgnoredInTransactions()** ¿Pasa por alto esta base de datos una sentencia de definición de datos de una transacción?
`boolean`	**deletesAreDetected(int type)** ¿Se puede detectar la supresión de una fila visible llamando al método ResultSet.rowDeleted?
`boolean`	**doesMaxRowSizeIncludeBlobs()** ¿Incluye el valor de retorno del método getMaxRowSize los tipos de datos SQL LONGVARCHAR y LONGVARBINARY?
`String`	**getCatalogSeparator()** Obtiene el objeto String que esta base de datos utiliza como separador entre un catálogo y un nombre de tabla.
`ResultSet`	**getColumns(String catalog, String schemaPattern, String tableNamePattern, String columnNamePattern)** Obtiene una descripción de las columnas de tabla disponibles en el catálogo especificado. El objeto ResultSet que este método devuelve se basa en la especificación JDK 1.3 y tiene 18 columnas.

Tabla 40. Métodos de la interfaz DatabaseMetaData (continuación)

Tipo de valor de retorno del método	Método
`Connection`	**getConnection()** JDBC 2.0 Obtiene la conexión que ha producido este objeto de metadatos.
`ResultSet`	**getCrossReference(String primaryCatalog, String primarySchema, String primaryTable, String foreignCatalog, String foreignSchema, String foreignTable)** Obtiene una descripción de las columnas de clave foránea de la tabla de claves foráneas que hace referencia a las columnas de clave primaria de la tabla de claves primarias (describe cómo una tabla importa la clave de otra). Normalmente, debe devolver un solo par de clave foránea/clave primaria (la mayoría de tablas únicamente importan una clave foránea de una tabla una vez.) Están ordenadas por FKTABLE_NAME y KEY_SEQ.
`int`	**getDatabaseMajorVersion()** JDBC 3.0 Obtiene el número de versión mayor de la base de datos.
`int`	**getDatabaseMinorVersion()** JDBC 3.0 Obtiene el número de versión menor de la base de datos.
`String`	**getDatabaseProductName()** Obtiene el nombre de este producto de base de datos.
`String`	**getDatabaseProductVersion()** Obtiene la versión de este producto de base de datos.
`int`	**getDefaultTransactionIsolation()** Obtiene el nivel de aislamiento de transacción predeterminado de la base de datos.
`int`	**getDriverMajorVersion()** Obtiene el número de versión mayor del controlador JDBC.
`int`	**getDriverMinorVersion()** Obtiene el número de versión menor del controlador JDBC.
`String`	**getDriverName()** Obtiene el nombre de este controlador JDBC.
`String`	**getDriverVersion()** Obtiene la versión de este controlador JDBC.
`String`	**getIdentifierQuoteString()** Obtiene la serie utilizada para citar los identificadores de SQL. Devuelve un espacio " " si no se soporta el entrecomillado de identificadores.
`ResultSet`	**getImportedKeys(String catalog, String schema, String table)** Obtiene una descripción de las columnas de clave primaria a las que hacen referencia las columnas de clave foránea de una tabla (las claves primarias importadas por una tabla).
`ResultSet`	**getExportedKeys(String catalog, String schema, String table)** Obtiene una descripción de las columnas de clave foránea que hacen referencia a las columnas de clave primaria de una tabla (las claves foráneas exportadas por una tabla).

Tabla 40. Métodos de la interfaz DatabaseMetaData (continuación)

Tipo de valor de retorno del método	Método
ResultSet	**getIndexInfo(String catalog, String schema, String table, Boolean approximate)** Obtiene una descripción de los índices de la tabla dada. Las columnas del conjunto de resultados están ordenadas por NON_UNIQUE, TYPE, INAME y ORDINAL_POSITION.
int	**getJDBCMajorVersion()** JDBC 3.0 Obtiene el número de versión mayor del controlador JDBC.
int	**getJDBCMinorVersion()** JDBC 3.0 Obtiene el número de versión menor del controlador JDBC.
int	**getMaxBinaryLiteralLength()** Obtiene la cantidad máxima de caracteres hexadecimales de un literal binario en línea.
int	**getMaxCatalogNameLength** Obtiene el número máximo de caracteres de un nombre de catálogo.
int	**getMaxCharLiteralLength()** Obtiene la longitud máxima para un literal de tipo carácter.
int	**getMaxColumnNameLength()** Obtiene el límite de longitud del nombre de una columna.
int	**getMaxColumnsInGroupBy()** Obtiene el número máximo de columnas en una cláusula GROUP BY.
int	**getMaxColumnsInIndex()** Obtiene el número máximo de columnas permitidas en un índice.
int	**getMaxColumnsInOrderBy()** Obtiene el número máximo de columnas en una cláusula ORDER BY.
int	**getMaxColumnsInSelect()** Obtiene el número máximo de columnas en una sentencia SELECT.
int	**getMaxColumnsInTable** Obtiene el número máximo de columnas que esta base de datos permite en una tabla.
int	**getMaxConnections()** Obtiene la cantidad máxima de conexiones de esta base de datos que pueden estar activas a la vez.
int	**getMaxCursorNameLength** Obtiene el número máximo de caracteres que esta base de datos permite en un nombre de cursor.
int	**getMaxIndexLength()** Obtiene la longitud máxima de un índice (en bytes).
int	**getMaxProcedureNameLength** Obtiene el número máximo de caracteres que esta base de datos permite en un nombre de procedimiento.

Tabla 40. Métodos de la interfaz DatabaseMetaData (continuación)

Tipo de valor de retorno del método	Método
`int`	**getMaxRowSize()** Obtiene la longitud máxima de una sola fila.
`int`	**getMaxSchemaNameLength** Obtiene el número máximo de caracteres que esta base de datos permite en un nombre de esquema.
`int`	**getMaxStatementLength()** Obtiene la longitud máxima de una sentencia de SQL.
`int`	**getMaxStatements()** Obtiene la cantidad máxima de sentencias activas que se pueden abrir a la vez en esta base de datos.
`int`	**getMaxTableNameLength()** Obtiene la longitud máxima de un nombre de tabla.
`int`	**getMaxTablesInSelect()** Obtiene el número máximo de tablas en una sentencia SELECT.
`int`	**getMaxUserNameLength()** ¿Cuál es la longitud máxima de un nombre de usuario?
`ResultSet`	**getPrimaryKeys(String catalog, String schema, String table)** Obtiene una descripción de las columnas de clave primaria de una tabla.
`int`	**getResultSetHoldability** JDBC 3.0 ¿Cuál es la retenibilidad predeterminada de este objeto ResultSet?
`String`	**getSearchStringEscape()** Obtiene la serie que se puede utilizar para los caracteres comodín de escape.
`int`	**getSQLStateType()** JDBC 3.0 Devuelve un valor para indicar si el SQLSTATE devuelto por SQLException.getSQLState es X/Open (ahora se conoce como Open Group) SQL CLI o SQL99.
`ResultSet`	**getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types)** Obtiene una descripción de las tablas disponibles en un catálogo. El objeto ResultSet que este método devuelve se basa en la especificación JDK 1.3 y tiene 5 columnas.
`String`	**getURL()** Obtiene el URL de esta base de datos.
`String`	**getUserName()** Obtiene el nombre de usuario tal como lo conoce la base de datos.
`boolean`	**insertsAreDetected(int type)** ¿Se puede detectar la inserción de una fila visible llamando al método ResultSet.rowInserted?

Tabla 40. Métodos de la interfaz DatabaseMetaData (continuación)

Tipo de valor de retorno del método	Método
boolean	**isCatalogAtStart()** ¿Aparece un catálogo al principio de un nombre de tabla totalmente calificado?
boolean	**isReadOnly()** ¿Esta base de datos está en modalidad de sólo lectura?
boolean	**nullPlusNonNullIsNull()** ¿Da soporte esta base de datos a concatenaciones entre valores NULL y no NULL siendo NULL?
boolean	**nullsAreSortedAtEnd()** ¿Se clasifican los valores NULL al final independientemente del orden de clasificación?
boolean	**nullsAreSortedAtStart()** ¿Se clasifican los valores NULL al principio independientemente del orden de clasificación?
boolean	**nullsAreSortedHigh()** ¿Se clasifican los valores NULL por lo alto?
boolean	**nullsAreSortedLow()** ¿Se clasifican los valores NULL por lo bajo?
boolean	**othersDeletesAreVisible(int type)** ¿Son visibles las supresiones realizadas por otros?
boolean	**othersInsertsAreVisible(int type)** ¿Son visibles las inserciones realizadas por otros?
boolean	**othersUpdatesAreVisible(int type)** ¿Son visibles las actualizaciones realizadas por otros?
boolean	**ownDeletesAreVisible(int type)** ¿Son visibles las supresiones propias de un conjunto de resultados?
boolean	**ownInsertsAreVisible(int type)** ¿Son visibles las inserciones propias de un conjunto de resultados?
boolean	**ownUpdatesAreVisible(int type)** ¿Son visibles las actualizaciones propias de un conjunto de resultados?
boolean	**storesLowerCaseIdentifiers()** ¿Esta base de datos trata los identificadores de SQL no entrecomillados que utilizan una combinación de mayúsculas y minúsculas como si no fueran sensibles a las mayúsculas y minúsculas y los almacena en minúsculas?
boolean	**storesLowerCaseQuotedIdentifiers()** ¿Esta base de datos trata los identificadores de SQL entrecomillados que utilizan una combinación de mayúsculas y minúsculas como si no fueran sensibles a las mayúsculas y minúsculas y los almacena en minúsculas?

Tabla 40. Métodos de la interfaz DatabaseMetaData (continuación)

Tipo de valor de retorno del método	Método
`boolean`	**storesMixedCaseIdentifiers()** ¿Esta base de datos trata los identificadores de SQL no entrecomillados que utilizan una combinación de mayúsculas y minúsculas como si no fueran sensibles a las mayúsculas y minúsculas y los almacena en una combinación de mayúsculas y minúsculas?
`boolean`	**storesMixedCaseQuotedIdentifiers()** ¿Esta base de datos trata los identificadores de SQL entrecomillados que utilizan una combinación de mayúsculas y minúsculas como si no fueran sensibles a las mayúsculas y minúsculas y los almacena en una combinación de mayúsculas y minúsculas?
`boolean`	**storesUpperCaseIdentifiers()** ¿Esta base de datos trata los identificadores de SQL no entrecomillados que utilizan una combinación de mayúsculas y minúsculas como si no fueran sensibles a las mayúsculas y minúsculas y los almacena en mayúsculas?
`boolean`	**storesUpperCaseQuotedIdentifiers()** ¿Esta base de datos trata los identificadores de SQL entrecomillados que utilizan una combinación de mayúsculas y minúsculas como si no fueran sensibles a las mayúsculas y minúsculas y los almacena en mayúsculas?
`boolean`	**supportsAlterTableWithAddColumn()** ¿Da soporte esta base de datos a ALTER TABLE con adición de columnas?
`boolean`	**supportsAlterTableWithDropColumn()** ¿Da soporte esta base de datos a ALTER TABLE con descarte de columnas?
`boolean`	**supportsANSI92EntryLevelSQL()** ¿Da soporte esta base de datos a la gramática de SQL de nivel de entrada ANSI92?
`boolean`	**supportsANSI92FullSQL()** ¿Da soporte esta base de datos a la gramática de SQL ANSI92 intermedia?
`boolean`	**supportsANSI92IntermediateSQL()** ¿Da soporte esta base de datos a la gramática de SQL ANSI92 completa?
`boolean`	**supportsBatchUpdates()** ¿Da soporte esta base de datos a las actualizaciones de proceso por lotes?
`boolean`	**supportsCatalogsInDataManipulation()** ¿Puede utilizarse un nombre de catálogo en una sentencia de manipulación de datos?
`boolean`	**supportsCatalogsInIndexDefinitions()** ¿Puede utilizarse un nombre de catálogo en una sentencia de definición de índices?
`boolean`	**supportsCatalogsInPrivilegeDefinitions()** ¿Puede utilizarse un nombre de catálogo en una sentencia de definición de privilegios?
`boolean`	**supportsCatalogsInProcedureCalls()** ¿Puede utilizarse un nombre de catálogo en una sentencia de llamada a un procedimiento?

Tabla 40. Métodos de la interfaz DatabaseMetaData (continuación)

Tipo de valor de retorno del método	Método
boolean	**supportsColumnAliasing()** Devuelve el valor true si se soporta la creación de alias de columnas.
boolean	**supportsCatalogsInTableDefinitions()** ¿Puede utilizarse un nombre de catálogo en una sentencia de definición de tablas?
boolean	**supportsConvert( )** ¿Da soporte esta base de datos a la función CONVERT?
boolean	**supportsConvert(int fromType, int toType)** ¿Da soporte esta base de datos a la función CONVERT para dos tipos de SQL especificados?
boolean	**supportsCoreSQLGrammar()** ¿Da soporte esta base de datos a la gramática de SQL básica para ODBC?
boolean	**supportsCorrelatedSubqueries()** ¿Da soporte esta base de datos a las subconsultas correlacionadas?
boolean	**supportsDataDefinitionAndDataManipulationTransactions()** ¿Da soporte esta base de datos tanto a las sentencias de definición de datos como a las de manipulación de datos en una transacción?
boolean	**supportsDataManipulationTransactionsOnly()** ¿Esta base de datos soporta sólo las sentencias de manipulación de datos en una transacción?
boolean	**supportsDifferentTableCorrelationNames()** Cuando se soportan nombres de correlación de tablas, ¿deben ser distintos a los nombres de las tablas?
boolean	**supportsExpressionsInOrderBy** ¿Da soporte esta base de datos a expresiones en listas ORDER BY?
boolean	**supportsExtendedSQLGrammar()** ¿Da soporte esta base de datos a la gramática de SQL ampliada para ODBC?
boolean	**supportsFullOuterJoins()** Devuelve el valor true si las uniones externas anidadas completas están soportadas.
boolean	**supportsGetGeneratedKeys()** JDBC 3.0 ¿Se pueden recuperar claves generadas después de haber ejecutado una sentencia?
boolean	**supportsGroupBy()** ¿Da soporte esta base de datos a la cláusula GROUP BY?
boolean	**supportsGroupByBeyondSelect()** ¿Permite esta base de datos una cláusula GROUP BY que incluya columnas no incluidas en la sentencia SELECT si todas las columnas de la sentencia SELECT están incluidas en la cláusula GROUP BY?

Tabla 40. Métodos de la interfaz DatabaseMetaData (continuación)

Tipo de valor de retorno del método	Método
boolean	**supportsGroupByUnrelated()** ¿Permite esta base de datos una cláusula GROUP BY que incluya una columna que no esté en la sentencia SELECT?
boolean	**supportsIntegrityEnhancementFacility()** ¿Da soporte esta base de datos a SQL Integrity Enhancement Facility (Recurso de mejora de la integridad de SQL)?
boolean	**supportsLikeEscapeClause()** ¿Da soporte esta base de datos a la especificación de una cláusula de escape LIKE?
boolean	**supportsLimitedOuterJoins()** ¿Proporciona esta base de datos soporte limitado para las uniones externas?
boolean	**supportsMinimumSQLGrammar()** ¿Da soporte esta base de datos a la gramática de SQL mínima para ODBC?
boolean	**supportsMixedCaseIdentifiers()** Devuelve el valor true si la base de datos trata los identificadores de SQL no entrecomillados que utilizan una combinación de mayúsculas y minúsculas como si fueran sensibles a las mayúsculas y minúsculas y los almacena en una combinación de mayúsculas y minúsculas.
boolean	**supportsMixedCaseQuotedIdentifiers()** Devuelve el valor true si la base de datos trata los identificadores de SQL entrecomillados que utilizan una combinación de mayúsculas y minúsculas como si fueran sensibles a las mayúsculas y minúsculas y los almacena en una combinación de mayúsculas y minúsculas.
boolean	**supportsMultipleOpenResultSets()** JDBC 3.0 ¿Puede un objeto CallableStatement devolver varios objetos ResultSet simultáneamente?
boolean	**supportsMultipleResultSets()** ¿Da soporte esta base de datos a la obtención de varios objetos ResultSet de una sola llamada al método execute?
boolean	**supportsMultipleTransactions()** ¿Permite esta base de datos que haya varias transacciones abiertas al mismo tiempo en distintas conexiones?
boolean	**supportsNamedParameters()** JDBC 3.0 ¿Da soporte esta base de datos a los parámetros con nombre para las sentencias invocables?
boolean	**supportsNonNullableColumns()** Devuelve el valor true si se pueden definir columnas como no nulas.
boolean	**supportsOpenCursorsAcrossCommit()** Devuelve el valor si la base de datos permite mantener cursores abiertos entre confirmaciones; de lo contrario, devuelve el valor false.

Tabla 40. Métodos de la interfaz DatabaseMetaData (continuación)

Tipo de valor de retorno del método	Método
boolean	**supportsOpenCursorsAcrossRollback()** ¿Permite esta base de datos mantener cursores abiertos entre retrotracciones?
boolean	**supportsOpenStatementsAcrossCommit()** ¿Permite esta base de datos mantener sentencias abiertas entre confirmaciones?
boolean	**supportsOpenStatementsAcrossRollback()** ¿Permite esta base de datos mantener sentencias abiertas entre retrotracciones?
boolean	**supportsOrderByUnrelated()** Devuelve el valor true si una cláusula ORDER BY puede utilizar columnas que no estén en la sentencia SELECT.
boolean	**supportsOuterJoins()** Devuelve el valor true si se soporta alguna forma de unión externa.
boolean	**supportsPositionedDelete()** Devuelve el valor true si se da soporte a una sentencia DELETE posicionada.
boolean	**supportsPositionedUpdate()** Devuelve el valor true si se da soporte a una sentencia UPDATE posicionada.
boolean	**supportsResultSetConcurrency(int type, int concurrency)** ¿Da soporte esta base de datos a un tipo de simultaneidad especificado en combinación con un tipo de conjunto de resultados especificado?
boolean	**supportsResultSetHoldability(int holdability)** JDBC 3.0 ¿Permite esta base de datos retener el conjunto de resultados especificado?
boolean	**supportsResultSetType(int type)** JDBC 2.0 Devuelve el valor true si la base de datos soporta el tipo de conjunto de resultados especificado.
boolean	**supportsSavepoints()** JDBC 3.0 ¿Da soporte esta base de datos a los puntos de guardar?
boolean	**supportsSchemasInDataManipulation()** ¿Puede utilizarse un nombre de esquema en una sentencia de manipulación de datos?
boolean	**supportsSchemasInIndexDefinitions()** ¿Puede utilizarse un nombre de esquema en una sentencia de definición de índices?
boolean	**supportsSchemasInPrivilegeDefinitions()** ¿Puede utilizarse un nombre de esquema en la definición de privilegios?
boolean	**supportsSchemasInProcedureCalls()** ¿Puede utilizarse un nombre de esquema en una llamada a un procedimiento?
boolean	**supportsSchemasInTableDefinitions()** Devuelve el valor true si el nombre de esquema se puede utilizar en una sentencia de definición de tablas.

Tabla 40. Métodos de la interfaz DatabaseMetaData (continuación)

Tipo de valor de retorno del método	Método
boolean	**supportsSelectForUpdate()** ¿Da soporte esta base de datos a las sentencias SELECT FOR UPDATE?
boolean	**supportsStatementPooling()** JDBC 3.0 ¿Da soporte esta base de datos a la agrupación de sentencias?
boolean	**supportsStoredProcedures()** ¿Da soporte esta base de datos a las llamadas a procedimientos almacenados que utilicen la sintaxis de escape del procedimiento almacenado?
boolean	**supportsSubqueriesInComparisons()** ¿Da soporte esta base de datos a las subconsultas en expresiones de comparación?
boolean	**supportsSubqueriesInExists()** ¿Da soporte esta base de datos a las subconsultas en expresiones EXISTS?
boolean	**supportsSubqueriesInIns()** ¿Da soporte esta base de datos a las subconsultas en expresiones IN?
boolean	**supportsSubqueriesInQuantifieds()** ¿Da soporte esta base de datos a las subconsultas en expresiones cuantificadas?
boolean	**supportsTableCorrelationNames()** ¿Da soporte esta base de datos a los nombres de correlación de tablas?
boolean	**supportsTransactions()** Devuelve el valor true si las transacciones están soportadas. De no ser así, el nivel de aislamiento es TRANSACTION_NONE.
boolean	**supportsTransactionIsolationLevel(int level)** Devuelve el valor true si la base de datos soporta el nivel de aislamiento de transacción especificado en *nivel*.
boolean	**supportsUnion()** ¿Da soporte esta base de datos a SQL UNION?
boolean	**supportsUnionAll()** ¿Da soporte esta base de datos a SQL UNION ALL?
boolean	**updatesAreDetected(int type)** ¿Se puede detectar la actualización de una fila visible llamando al método ResultSet.rowUpdated?
boolean	**usesLocalFilePerTable()** ¿Utiliza esta base de datos una fila para cada tabla?
boolean	**usesLocalFiles()** ¿Almacena esta base de datos las tablas en un archivo local?

La interfaz Driver:

La interfaz Driver es la infraestructura de Java SQL que admite varios controladores de base de datos.

Cuando se cargue una clase Driver, debería crearse una instancia de sí misma y registrarse con el DriverManager. Esto significa que un usuario puede cargar y registrar el controlador JDBC de DB2 Everyplace llamando a:

```
Class.forName("com.ibm.db2e.jdbc.DB2eDriver")
```

paquete java.sql

interfaz pública **Driver**

La Tabla 41 lista los métodos de la interfaz Driver a los que DB2 Everyplace da soporte.

Tabla 41. Métodos de la interfaz Driver

Tipo de valor de retorno del método	Método
`boolean`	**acceptsURL(String url)** Devuelve el valor true si el controlador considera que puede abrir una conexión con el URL indicado.
`Connection`	**connect(String url, java.util.Hashtable info)** Método sobrecargado de DB2 Everyplace para las configuraciones de bibliotecas de clases que no soportan java.util.Properties. Véase connect(String url, Properties info) para conocer los pares de clave/valor soportados.

Tabla 41. Métodos de la interfaz Driver (continuación)

Tipo de valor de retorno del método	Método
`Connection`	**connect(String url, Properties info)** Intenta crear una conexión de base de datos con el URL indicado. Se puede utilizar el argumento java.util.Properties para pasar pares arbitrarios de código/valor de serie como argumentos de conexión. DB2 Everyplace soporta los siguientes pares de clave y valor específicos del controlador: • Clave: DB2e_ENCODING Valor: codificación de caracteres • Clave: ENABLE_DELETE_PHYSICAL_REMOVE Valor: true, false Valor predeterminado: false • Clave: ENABLE_DIRTY_BIT_SET_BY_APPLICATION Valor: true, false Valor predeterminado: false • Clave: ENABLE_FILENAME_FORMAT_83 Valor: true, false Valor predeterminado: false • Clave: ENABLE_IO_WRITETHROUGH Valor: true, false Valor predeterminado (plataformas Windows y Linux x86 solamente): false Valor predeterminado (otras plataformas): true **Restricción:** La clave ENABLE_IO_WRITETHROUGH solamente afecta a las plataformas Windows y Linux x86. No tiene ningún efecto en las demás plataformas. • Clave: ENABLE_READ_INCLUDE_MARKED_DELETE Valor: true, false Valor predeterminado: false • Clave: ENABLE_REORG Valor: true, false Valor predeterminado: true • Clave: ENABLE_SHARED_DATABASE_ACCESS Valor: true, false Valor predeterminado: false • Clave ENABLE_TABLE_CHECKSUM Valor: true, false Valor predeterminado: false • Clave: LOCK_TIMEOUT Valor: número de segundos que se deberá esperar a obtener un bloqueo antes de retrotraer una transacción. Valor predeterminado: 20 • Clave: LOGIN_TIMEOUT Valor: número de segundos que se deberá esperar a que una petición de inicio de sesión finalice antes de devolver el control a la aplicación. Valor predeterminado: 0 • Clave: password Valor: contraseña del usuario • Clave: user Valor: nombre del usuario

Tabla 41. Métodos de la interfaz Driver (continuación)

Tipo de valor de retorno del método	Método
`int`	**getMajorVersion()** Obtiene el número de versión mayor del controlador.
`int`	**getMinorVersion()** Obtiene el número de versión menor del controlador.
`boolean`	**jdbcCompliant()** Informa sobre si éste es un controlador JDBC COMPLIANT™ genuino.

Ejemplo: El ejemplo siguiente muestra cómo crear un objeto Properties y cómo utilizar el método setProperty().

```
Properties props = new Properties();
props.setProperty("ENABLE_REORG", "false");
props.setProperty("LOCK_TIMEOUT", "200");
props.setProperty("ENABLE_SHARED_DATABASE_ACCESS" , "true");
props.setProperty("ENABLE_IO_WRITETHROUGH","true");
props.setProperty("ENABLE_TABLE_CHECKSUM", "true");
Connection con = DriverManager.getConnection(url, props);
```

Interfaz PreparedStatement:

La interfaz PreparedStatement crea un objeto que representa una sentencia de SQL compilada previamente.

Se ha compilado previamente una sentencia de SQL y se ha almacenado en un objeto PreparedStatement. Este objeto puede utilizarse para ejecutar de modo eficaz esta sentencia varias veces.

Nota: Los métodos ″setter″ para establecer valores de parámetros IN deben especificar tipos que sean compatibles con el tipo de SQL definido del parámetro de entrada. Por ejemplo, si el parámetro IN tiene el tipo de SQL INTEGER, se deberá utilizar el método setInt.

paquete java.sql

interfaz pública **PreparedStatement**

amplía Statement

La Tabla 42 lista los métodos de la interfaz PreparedStatement a los que DB2 Everyplace da soporte.

Tabla 42. Métodos de la interfaz PreparedStatement

Tipo de valor de retorno del método	Método
`void`	**clearParameters()** Borra los valores de parámetro actuales de modo inmediato.
`void`	**close()** Libera una base de datos de *PreparedStatement* y recursos de JDBC de modo inmediato, en vez de esperar que se liberen de modo automático.
`boolean`	**execute()** Ejecuta cualquier tipo de sentencia de SQL.

Tabla 42. Métodos de la interfaz PreparedStatement (continuación)

Tipo de valor de retorno del método	Método
`ResultSet`	**executeQuery()** Ejecuta la consulta de SQL en este objeto `PreparedStatement` y devuelve el conjunto de resultados generado por la consulta.
`int`	**executeUpdate()** Ejecuta la sentencia de SQL INSERT, UPDATE o DELETE en este objeto `PreparedStatement`.
`void`	**setBigDecimal (int parameterIndex, BigDecimal x)** Establece el parámetro designado en un valor `java.lang.BigDecimal`.
`void`	**setBinaryStream(int parameterIndex, InputStream x, int length)** Establece el parámetro designado en un valor `java.lang.InputStream` con una longitud de `x` bytes.
`void`	**setBoolean (int parameterIndex, boolean x)** Establece el parámetro designado en un valor boolean de Java. El controlador JDBC de DB2 Everyplace lo convierte a un valor SMALLINT de SQL cuando lo envía a la base de datos.
`void`	**setBlob(int i, Blob x)** JDBC 2.0 Establece un parámetro BLOB.
`void`	**setBytes(int parameterIndex, byte[ ] x)** Establece el parámetro designado en una matriz de bytes de Java.
`void`	**setDate(int parameterIndex, Date x)** Establece el parámetro designado en un valor java.sql.Date.
`void`	**setDouble(int parameterIndex, double x)** Establece el parámetro designado en un valor double de Java. El controlador JDBC de DB2 Everyplace lo convierte a un valor DECIMAL de SQL cuando lo envía a la base de datos.
`void`	**setFloat(int parameterIndex, float x)** Establece el parámetro designado en un valor float de Java. Cuando BigDecimal se convierte a float, si el BigDecimal es demasiado grande para representarlo como float, se convertirá a FLOAT.NEGATIVE_INFINITY o FLOAT.POSITIVE_INFINITY según corresponda.
`void`	**setInt (int parameterIndex, int x)** Establece el parámetro designado en un valor `int` de Java.
`void`	**setLong(int parameterIndex, long x)** Establece el parámetro designado en un valor long de Java.
`void`	**setNull (int parameterIndex, int sqlType)** Establece el parámetro designado en NULL de SQL.

Tabla 42. Métodos de la interfaz PreparedStatement (continuación)

Tipo de valor de retorno del método	Método
`void`	**setObject(int parameterIndex, Object x, int targetSqlType)** Establece el valor del parámetro designado con el objeto concreto. Restricciones de **DB2 Everyplace**: • `targetSqlType` debe corresponderse con uno de los tipos de datos a los que DB2 Everyplace da soporte. • Se da soporte a las conversiones básicas y de Serie. Por ejemplo, si `targetSqlType` es `Types.INTEGER`, x debería ser un objeto `Integer` o `String`. • Si `targetSqlType` es `Types.DECIMAL`, x también puede ser un objeto `Double`, `Float` o `Long`. • Si `targetSqlType` es `Types.SMALLINT`, x también puede ser un objeto `Boolean`.
`void`	**setShort (int parameterIndex, short x)** Establece el parámetro designado en un valor `short` de Java.
`void`	**setString (int parameterIndex, String x)** Establece el parámetro designado en un valor `String` de Java.
`void`	**setTime (int parameterIndex, Time x)** Establece el parámetro designado en un valor java.sql.Time.
`void`	**setTimestamp (int parameterIndex, Timestamp x)** Establece el parámetro designado en un valor java.sql.Timestamp.

Interfaz ResultSet:

La interfaz ResultSet proporciona acceso a una tabla de datos. Un objeto ResultSet normalmente se genera ejecutando una Sentencia.

Un objeto ResultSet mantiene un cursor apuntando hacia su fila de datos actual. Inicialmente, el cursor está colocado delante de la primera fila. El método `next()` mueve el cursor a la fila siguiente.

Los métodos getXXX recuperan valores de columna para la fila actual. Puede recuperar valores utilizando el número de índice de la columna o el nombre de la columna. En general, la utilización del índice de columnas es más eficaz. Las columnas se numeran a partir de uno. El controlador JDBC convierte los datos subyacentes al tipo Java especificado en el método ″getter″ y devuelve un valor Java adecuado.

paquete java.sql

interfaz pública **ResultSet**

La Tabla 43 lista los campos de la interfaz ResultSet a los que DB2 Everyplace da soporte.

Tabla 43. Campos de la interfaz ResultSet

Tipo de campo	Campo
`static int`	**CONCUR_READ_ONLY** Constante que indica la modalidad de simultaneidad para un objeto ResultSet que NO se puede actualizar. **Nota:** DB2 Everyplace no se puede utilizar con CONCUR_UPDATABLE. Si se especifica CONCUR_UPDATABLE para la modalidad de simultaneidad para un objeto ResultSet al crear un objeto Statement, el controlador de JDBC de DB2 Everyplace emite un SQLWarning en el objeto Connection que ha producido el objeto Statement y en su lugar utiliza CONCUR_READ_ONLY.
`static int`	**TYPE_FORWARD_ONLY** Constante que indica el tipo para un objeto ResultSet cuyo cursor sólo pueda moverse hacia adelante.
`static int`	**TYPE_SCROLL_INSENSITIVE** Constante que indica el tipo para un objeto ResultSet que pueda desplazarse pero que en general no resulte sensible a los cambios efectuados por otros. **Nota:** Utilice este tipo de objeto ResultSet con moderación, puesto que puede afectar al rendimiento. Este tipo utiliza SQL_INSENSITIVE como valor del atributo de sentencia CLI SQL_ATTR_CURSOR_SENSITIVITY. Consulte la documentación para la función de CLI SQLSetStmtAttr para obtener más detalles.
`static int`	**TYPE_SCROLL_SENSITIVE** Constante que indica el tipo para un objeto ResultSet que pueda desplazarse y que en general resulte sensible a los cambios efectuados por otros. **Nota:** Este tipo utiliza SQL_UNSPECIFIED como valor del atributo de sentencia CLI SQL_ATTR_CURSOR_SENSITIVITY. Consulte la documentación para la función de CLI SQLSetStmtAttr para obtener más detalles.

La Tabla 44 lista los métodos de la interfaz ResultSet a los que DB2 Everyplace da soporte.

Tabla 44. Métodos de la interfaz ResultSet

Tipo de valor de retorno del método	Método
`boolean`	**absolute(int row)** JDBC 2.0. Mueve el cursor al número de fila indicado del conjunto de resultados.
`void`	**afterLast()** JDBC 2.0. Mueve el cursor al final del conjunto de resultados, justo detrás de la última fila.
`void`	**beforeFirst()** JDBC 2.0. Mueve el cursor al principio del conjunto de resultados, justo delante de la primera fila.
`void`	**clearWarnings()** Borra todos los avisos de los que se ha informado para este objeto ResultSet.
`void`	**close()** Libera de inmediato los recursos de JDBC y base de datos de este objeto ResultSet en lugar de esperar a que suceda esto cuando se cierre automáticamente.

Tabla 44. Métodos de la interfaz ResultSet (continuación)

Tipo de valor de retorno del método	Método
`int`	**findColumn(String columnName)** Correlaciona el nombre de columna de ResultSet indicado con su índice de columnas de ResultSet.
`boolean`	**first()** JDBC 2.0. Mueve el cursor a la primera fila del conjunto de resultados.
`BigDecimal`	**getBigDecimal(int columnIndex)** JDBC 2.0. Obtiene el valor de una columna de la fila actual en forma de objeto java.math.BigDecimal con precisión completa.
`BigDecimal`	**getBigDecimal(int columnIndex, int scale)** Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet en forma de objeto java.math.BigDecimal en el lenguaje de programación Java.**En desuso.**
`BigDecimal`	**getBigDecimal(String columnName)** JDBC 2.0. Obtiene el valor de una columna de la fila actual en forma de objeto java.math.BigDecimal con precisión completa.
`BigDecimal`	**getBigDecimal(String columnName, int scale)** Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet en forma de objeto java.math.BigDecimal en el lenguaje de programación Java. **En desuso.**
`InputStream`	**getBinaryStream(int columnIndex)** Obtiene el valor de la columna designada contenida en la fila actual del objeto ResultSet, en forma de corriente binaria de bytes.
`InputStream`	**getBinaryStream(String columnName)** Obtiene el valor de la columna designada contenida en la fila actual del objeto ResultSet, en forma de corriente binaria de bytes.
`Blob`	**getBlob(int columnIndex)** JDBC 2.0. Obtiene un valor de BLOB de la fila actual de este objeto ResultSet.
`Blob`	**getBlob(String columnName)** JDBC 2.0. Obtiene un valor de BLOB de la fila actual de este objeto ResultSet.
`boolean`	**getBoolean(int columnIndex)** Obtiene el valor de una columna de la fila actual como boolean de Java. El controlador obtiene en primer lugar el valor de columna en forma de tipo de datos short de Java. Si el valor es igual a 1, se devuelve el valor true. De lo contrario, se devuelve el valor false.
`boolean`	**getBoolean(String columnName)** Obtiene el valor de una columna de la fila actual como boolean de Java. El controlador obtiene en primer lugar el valor de columna en forma de tipo de datos short de Java. Si el valor es igual a 1, se devuelve el valor true. De lo contrario, se devuelve el valor false.
`byte`	**getByte(int columnIndex)** Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet como byte en el lenguaje de programación Java.

Tabla 44. Métodos de la interfaz ResultSet (continuación)

Tipo de valor de retorno del método	Método
`byte`	**getByte(String columnName)** Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet como byte en el lenguaje de programación Java.
`byte[]`	**getBytes(int columnIndex)** Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet como matriz de bytes en el lenguaje de programación Java.
`byte[]`	**getBytes(String columnName)** Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet como matriz de bytes en el lenguaje de programación Java.
`int`	**getConcurrency()** JDBC 2.0. Devuelve la modalidad de simultaneidad del conjunto de resultados.
`Date`	**getDate(int columnIndex)** Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet en forma de objeto java.sql.Date en el lenguaje de programación Java.
`Date`	**getDate(int columnIndex, Calendar cal)** Devuelve el valor de la columna designada en la fila actual de este objeto ResultSet en forma de objeto java.sql.Date en el lenguaje de programación Java.
`Date`	**getDate(String columnName)** Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet en forma de objeto java.sql.Date en el lenguaje de programación Java.
`double`	**getDouble(int columnIndex)** Obtiene el valor de una columna de la fila actual como double de Java.
`double`	**getDouble(String columnName)** Obtiene el valor de una columna de la fila actual como double de Java.
`float`	**getFloat(int columnIndex)** Obtiene el valor de una columna de la fila actual como float de Java.
`float`	**getFloat(String columnName)** Obtiene el valor de una columna de la fila actual como float de Java.
`int`	**getInt(int columnIndex)** Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet en forma de entero en el lenguaje de programación Java.
`int`	**getInt(String columnName)** Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet en forma de entero en el lenguaje de programación Java.
`long`	**getLong(int columnIndex)** Obtiene el valor de una columna de la fila actual en forma de tipos datos long de Java.
`long`	**getLong(String columnName)** Obtiene el valor de una columna de la fila actual en forma de tipos datos long de Java.

Tabla 44. Métodos de la interfaz ResultSet (continuación)

Tipo de valor de retorno del método	Método
`ResultSetMetaData`	**getMetaData()** Recupera el número, los tipos y las propiedades de las columnas de este objeto ResultSet.
`Object`	**getObject(int columnIndex)** Obtiene el valor de una columna de la fila actual en forma de objeto Java.
`Object`	**getObject(String columnName)** Obtiene el valor de una columna de la fila actual en forma de objeto Java.
`int`	**getRow()** JDBC 2.0. Recupera el número de la fila actual.
`short`	**getShort(int columnIndex)** Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet en forma de tipo de datos short en el lenguaje de programación Java.
`short`	**getShort(String columnName)** Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet en forma de tipo de datos short en el lenguaje de programación Java.
`Statement`	**getStatement()** JDBC 2.0. Devuelve la sentencia (Statement) que ha producido este objeto ResultSet.
`String`	**getString(int columnIndex)** Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet como String en el lenguaje de programación Java.
`String`	**getString(String columnName)** Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet como String en el lenguaje de programación Java.
`Time`	**getTime(int columnIndex)** Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet en forma de objeto java.sql.Time en el lenguaje de programación Java.
`Time`	**getTime(String columnName)** Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet en forma de objeto java.sql.Time en el lenguaje de programación Java.
`Timestamp`	**getTimestamp(String columnName)** Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet en forma de objeto java.sql.Timestamp en el lenguaje de programación Java.
`Timestamp`	**getTimestamp(int columnIndex)** Obtiene el valor de la columna designada en la fila actual de este objeto ResultSet en forma de objeto java.sql.Timestamp en el lenguaje de programación Java.
`int`	**getType()** JDBC 2.0. Devuelve el tipo de este conjunto de resultados.

Tabla 44. Métodos de la interfaz ResultSet (continuación)

Tipo de valor de retorno del método	Método
SQLWarning	**getWarnings()** Devuelve el primer aviso informado por llamadas a este objeto ResultSet.
boolean	**isAfterLast()** JDBC 2.0. Indica si el cursor se encuentra detrás de la última fila del conjunto de resultados.
boolean	**isBeforeFirst()** JDBC 2.0. Indica si el cursor se encuentra delante de la primera fila del conjunto de resultados.
boolean	**isFirst()** JDBC 2.0. Indica si el cursor se encuentra en la primera fila del conjunto de resultados.
boolean	**isLast()** JDBC 2.0. Indica si el cursor se encuentra en la última fila del conjunto de resultados. Este método no recibe soporte para conjuntos de resultados de tipo TYPE_FORWARD_ONLY.
boolean	**last()** JDBC 2.0. Mueve el cursor a la última fila del conjunto de resultados.
boolean	**next()** Mueve el cursor una fila hacia abajo a partir de su posición actual.
boolean	**previous()** JDBC 2.0. Mueve el cursor a la fila anterior del conjunto de resultados.
boolean	**relative(int rows)** JDBC 2.0. Mueve el cursor un número relativo de filas, ya sea positivo o negativo.
boolean	**wasNull()** Informa sobre si la última columna leída tenía un valor NULL de SQL.

Interfaz ResultSetMetaData:

La interfaz ResultSetMetaData crea un objeto que se puede utilizar para averiguar los tipos y propiedades de las columnas de un ResultSet.

paquete java.sql

interfaz pública **ResultSetMetaData**

La Tabla 45 lista los campos de la interfaz ResultSetMetaData a los que DB2 Everyplace da soporte.

Tabla 45. Campos de la interfaz ResultSetMetaData

Tipo de campo	Campo
static int	**columnNoNulls** Constante que indica que una columna no admite valores nulos (NULL).
static int	**columnNullable** Constante que indica que una columna admite valores nulos (NULL).
static int	**columnNullableUnknown** Constante que indica que se desconoce la posibilidad de anular los valores de una columna.

La Tabla 46 lista los métodos de la interfaz ResultSetMetaData a los que DB2 Everyplace da soporte.

Tabla 46. Métodos de la interfaz ResultSetMetaData

Tipo de valor de retorno del método	Método
`String`	**getCatalogName(int column)** Obtiene un nombre de catálogo de tabla de la columna. DB2 Everyplace siempre devuelve "" (no aplicable).
`int`	**getColumnCount()** Devuelve el número de columnas de este objeto `ResultSet`.
`int`	**getColumnDisplaySize (int column)** Indica la anchura máxima normal en caracteres de la columna designada.
`String`	**getColumnLabel(int column)** Obtiene el título de columna sugerido para su utilización en copias impresas y pantallas.
`String`	**getColumnName (int column)** Obtiene el nombre de la columna designada.
`int`	**getColumnType (int column)** Obtiene el tipo SQL de la columna designada.
`String`	**getColumnTypeName(int column)** Recupera un nombre de tipo específico de la base de datos de columna.
`int`	**getPrecision (int column)** Obtiene el número de dígitos decimales de la columna designada.
`int`	**getScale (int column)** Obtiene el número de dígitos de la columna designada que se encuentran a la derecha de la cola decimal.
`String`	**getSchemaName(int column)** Obtiene el nombre de esquema de la tabla de una columna. DB2 Everyplace siempre devuelve "" (no aplicable).
`int`	**isNullable (int column)** Indica la posibilidad de anulación de valores de la columna designada.
`boolean`	**isWritable(int column)** Indica si es posible que una grabación en la columna resulte satisfactoria.

Interfaz Statement:

La interfaz Statement crea un objeto que se utiliza para ejecutar una sentencia de SQL estático y obtener los resultados producidos por la misma.

paquete java.sql

interfaz pública **Statement**

La Tabla 47 lista los campos de la interfaz Statement a los que DB2 Everyplace da soporte.

Tabla 47. Campos de la interfaz Statement

Tipo de campo	Campo
`static int`	**SUCCESS_NO_INFO** Constante que indica que una sentencia de proceso por lotes se ha ejecutado satisfactoriamente, pero que no se dispone de un recuento del número de filas que ha afectado.

La Tabla 48 en la página 90 lista los métodos de la interfaz Statement a los que DB2 Everyplace da soporte.

Tabla 48. Métodos de la interfaz Statement

Tipo de valor de retorno del método	Método
`void`	**addBatch(String sql)** JDBC 2.0 Añade un mandato SQL al proceso por lotes actual de mandatos para la sentencia.
`void`	**cancel()**
`void`	**clearBatch()** JDBC 2.0 Convierte el conjunto de mandatos en el proceso por lotes actual vacío.
`void`	**close()** Libera los recursos JDBC y de base de datos de este objeto `Statement` de inmediato, en lugar de esperar a que suceda esto cuando se cierre automáticamente.
`boolean`	**execute(String sql)** Ejecuta una sentencia de SQL que puede devolver varios resultados.
`int[]`	**executeBatch()** JDBC 2.0 Somete un proceso por lotes de mandatos a la base de datos para su ejecución.
`ResultSet`	**executeQuery(String sql)** Ejecuta una sentencia de SQL que devuelve un solo objeto `ResultSet`.
`int`	**executeUpdate(String sql)** Ejecuta una sentencia INSERT, UPDATE o DELETE de SQL.
`Connection`	**getConnection()** JDBC 2.0. Devuelve el objeto Connection que ha producido este objeto Statement.
`boolean`	**getMoreResults()** Pasa al siguiente resultado de un objeto Statement. DB2 Everyplace siempre devuelve el valor false (no hay más resultados).
`ResultSet`	**getResultSet()** Devuelve el resultado actual en forma de objeto `ResultSet`.
`int`	**getResultSetConcurrency()** JDBC 2.0. Recupera la simultaneidad de conjuntos resultantes.
`int`	**getResultSetType()** JDBC 2.0. Determina el tipo del conjunto de resultados.
`int`	**getUpdateCount()** Devuelve el resultado actual como número de actualización; si el resultado es un ResultSet o no hay más resultados, se devuelve -1.

Interfaz Javax.sql

Este tema describe los paquetes proporcionados por el paquete javax.sql.

Interfaz DataSource:

Fábrica de conexiones con la fuente de datos física a la que representa este objeto DataSource. El método preferido de obtener una conexión es una sustitución del recurso de DriverManager, un objeto DataSource.

Una instancia de un objeto DataSource puede utilizarse en un programa autónomo para crear objetos Connection. En el ejemplo siguiente, se utiliza una instancia de DB2eDataSource para crear una conexión (Connection) con una base de datos portátil DB2 Everyplace con el url ″jdbc:db2e:myDataSource″:

```
com.ibm.db2e.jdbc.DB2eDataSource ds = new com.ibm.db2e.jdbc.DB2eDataSource();
    ds.setUrl("jdbc:db2e:myDataSource");
    Connection con = ds.getConnection();
```

paquete javax.sql

interfaz pública **DataSource**

La Tabla 49 y la Tabla 50 listan las propiedades de la interfaz DataSource a las que DB2 Everyplace da soporte. A las propiedades se puede acceder utilizando los métodos ″getter″ y ″setter″. (Las propiedades de DataSource siguen el convenio especificado para las propiedades de los componentes de JavaBeans™ en la Especificación JavaBeans 1.01).

Tabla 49. Propiedades de DataSource estándar a las que DB2 Everyplace da soporte

Nombre de propiedad	Tipo	Descripción	Métodos de acceso
description	String	descripción de esta fuente de datos	String getDescription() void setDescription(String Description)
password	String	contraseña de la base de datos	String getPassword() void setPassword(String password)
user	String	nombre de la cuenta del usuario	String getUser() void setUser(String user)

La Tabla 50 lista las propiedades soportadas de la interfaz DataSource que son específicas para DB2 Everyplace.

Tabla 50. Propiedades específicas de DB2 Everyplace para la interfaz DataSource

Nombre de propiedad	Tipo	Descripción	Métodos de acceso
deletePhysicalRemove	boolean	habilitar/inhabilitar la eliminación física de registros.	boolean isDeletePhysicalRemove() void setDeletePhysicalRemove(boolean enable)
dirtyBitSetByApplication	boolean	permitir/no permitir que la aplicación establezca el bit indicador	boolean isDirtyBitSetByApplication() void setDirtyBitSetByApplication(boolean enable)
encoding	String	codificación de caracteres	String getEncoding() void setEncoding(String encoding)
filenameFormat83	boolean	nombres de archivo cortos (formato 8.3) o largos	boolean isFilenameFormat83() void setFilenameFormat83(boolean enable)
ioMode	int	transferir los cambios directamente al soporte de almacenamiento o dejar que el SO se ocupe de ellos.	int getIoMode() void setIoMode(int mode)
lockTimeout	int	tiempo de espera de bloqueo en segundos (valor predeterminado = 20)	int getLockTimeout() void setLockTimeout(int seconds)

Tabla 50. Propiedades específicas de DB2 Everyplace para la interfaz DataSource (continuación)

Nombre de propiedad	Tipo	Descripción	Métodos de acceso
readIncludeMarkedDelete	boolean	habilitar/inhabilitar la lectura de registros suprimidos de forma lógica	boolean isReadIncludeMarkedDelete() void setReadIncludeMarkedDelete(boolean enable)
reorg	boolean	habilitar/inhabilitar reorganización	boolean isReorg() void setReorg(boolean enable)
sharedDatabaseAccess	boolean	acceso compartido o exclusivo a la base de datos	boolean isSharedDatabaseAccess() void setSharedDatabaseAccess(boolean enable)
tableChecksum	boolean	habilitar/inhabilitar sumas de comprobación para archivos de base de datos	boolean isEnabledTableChecksum() void setEnabledTableChecksum(boolean enable)
url	String	fuente de datos	String getUrl() void setUrl(String url)

La Tabla 51 lista los métodos de la interfaz DataSource a los que DB2 Everyplace da soporte.

Tabla 51. Métodos de la interfaz DataSource

Tipo de valor de retorno del método	Método
`int`	**getBufferpoolSize()** Obtiene el tamaño de la agrupación de almacenamientos intermedios de la conexión, expresado en bytes.
`Connection`	**getConnection()** Intenta establecer una conexión con la fuente de datos a la que representa este objeto DataSource.
`Connection`	**getConnection (java.lang.String username, java.lang.String password)** Intenta establecer una conexión con la fuente de datos a la que representa este objeto DataSource.
`int`	**getLoginTimeout()** Obtiene el tiempo máximo en segundos que puede esperar esta fuente de datos mientras intenta conectarse a una base de datos.
`java.io.PrintWriter`	**getLogWriter()** Recupera el transcriptor de anotaciones para este objeto DataSource.
`int`	**setBufferpoolSize(int size)** Establece la cantidad de memoria, en bytes, que la base de datos DB2 Everyplace debe reservar para sus agrupaciones de almacenamientos intermedios. Si este valor no es un múltiplo de 4K (4096 bytes), DB2 Everyplace realiza un redondeo por defecto al siguiente múltiplo de 4K más pequeño.
`void`	**setLoginTimeout(int seconds)** Establece el tiempo máximo en segundos que esperará esta fuente de datos mientras intenta conectarse a una base de datos.

Tabla 51. Métodos de la interfaz DataSource (continuación)

Tipo de valor de retorno del método	Método
`void`	**setLogWriter(java.io.PrintWriter out)** Establece el transcriptor de anotaciones para este objeto DataSource en el objeto java.io.PrintWriter especificado.

DB2 Everyplace incluye los siguientes tamaños predefinidos de la agrupación de almacenamientos intermedios:

Tabla 52. Constantes predefinidas para el tamaño de la agrupación de almacenamientos intermedios

Constante	Tamaño en bytes de la agrupación de almacenamientos intermedios
SQL_BUFFERPOOL_SIZE_DEFAULT	El valor predeterminado correspondiente a la plataforma en la que se está ejecutando DB2 Everyplace.
SQL_BUFFERPOOL_SIZE_64K	65 536
SQL_BUFFERPOOL_SIZE_128K	131 072
SQL_BUFFERPOOL_SIZE_256K	262 144
SQL_BUFFERPOOL_SIZE_512K	524 288
SQL_BUFFERPOOL_SIZE_1024K	1 048 576
SQL_BUFFERPOOL_SIZE_2048K	2 097 152
SQL_BUFFERPOOL_SIZE_4096K	4 194 304
SQL_BUFFERPOOL_SIZE_8172K	8 388 608
SQL_BUFFERPOOL_SIZE_1M	1 048 576
SQL_BUFFERPOOL_SIZE_2M	2 097 152
SQL_BUFFERPOOL_SIZE_4M	4 194 304
SQL_BUFFERPOOL_SIZE_8M	8 388 608

Importante:

- El valor mínimo de SQL_ATTR_BUFFERPOOL_SIZE es SQL_BUFFERPOOL_SIZE_64K. Si invoca SQLSetConnectAttr() y especifica un valor menor que SQL_BUFFERPOOL_SIZE_64K, SQlSetConnectAttr() devuelve el SQLSTATE **HY**024.
- Si el motor de la base de datos no puede asignar la cantidad de memoria especificada en el atributo de conexión SQL_ATTR_BUFFERPOOL_SIZE, el motor intentará utilizar una configuración menor para la agrupación de almacenamientos intermedios. SQLConnect() devolverá el SQLSTATE 01000.
- Si no existe memoria suficiente para la configuración mínima de la agrupación de almacenamientos intermedios, SQLConnect() devolverá el SQLState 58004.
- No puede cambiar el tamaño de la agrupación de almacenamientos intermedios si ya existe una conexión con la base de datos. Las conexiones nuevas utilizarán el tamaño de agrupación de almacenamientos intermedios de la conexión existente. SQLConnect() devolverá un aviso.

Soporte de idioma nacional (NLS)

Este tema contiene información sobre el soporte de idioma nacional (NLS) proporcionado por DB2 Everyplace, incluyendo información sobre los países, idiomas y páginas de códigos (conjuntos de códigos) soportados, y cómo configurar y utilizar las características NLS de DB2 Everyplace con los dispositivos y las aplicaciones. DB2 Everyplace da soporte a juegos de caracteres de un solo byte, de doble byte, multibyte y Unicode. Las codificaciones Unicode y no Unicode (ANSI) están soportadas en todos los sistemas operativos Windows.

Nota sobre las codificaciones de caracteres: DB2 Everyplace Sync Client y la base de datos dan soporte a codificaciones de páginas de códigos locales en las plataformas Win32. Para DB2 Everyplace Sync Client, la codificación de caracteres puede especificarse con el archivo de página de códigos config-isyn o la propiedad isync.encoding de iscServiceOpenEx(). Esta propiedad indica a DB2 Everyplace Sync Server que convierta los datos del origen a la codificación especificada para el cliente. Para la base de datos DB2 Everyplace, se utiliza la codificación de caracteres predeterminada del sistema. Es importante observar que para que los datos sincronizados se muestren correctamente en el cliente, la codificación de DB2 Everyplace Sync Client debe coincidir con la codificación del sistema. Por ejemplo, si la codificación de DB2 Everyplace Sync Client está establecida en UTF-8 y el valor predeterminado del sistema operativo está establecido en CP1252, los caracteres especiales pueden aparecer alterados al consultar la base de datos cliente.

Soporte NLS de DB2 Everyplace por sistema operativo

La Tabla 53 muestra los sistemas operativos y los correspondientes idiomas para los cuales existe soporte NLS.

Tabla 53. Soporte NLS

Idioma	Windows	WinCE	Linux	Symbian OS
Inglés	CP1252/ ISO8859-1/ UCS-2	UCS-2	CP1252/ ISO8859-1/ UTF-8	UCS-2
Francés	CP1252/ ISO8859-1/ UCS-2	UCS-2	CP1252/ ISO8859-1/ UTF-8	UCS-2
Alemán	CP1252/ ISO8859-1/ UCS-2	UCS-2	CP1252/ ISO8859-1/ UTF-8	UCS-2
Italiano	CP1252/ ISO8859-1/ UCS-2	UCS-2	CP1252/ ISO8859-1/ UTF-8	UCS-2
Español	CP1252/ ISO8859-1/ UCS-2	UCS-2	CP1252/ ISO8859-1/ UTF-8	UCS-2
Chino simplificado	Página de códigos/ UCS-2	UCS-2	Página de códigos/ UTF-8	N/D
Chino tradicional	CP950/ UCS-2	UCS-2 • Instalar habilitador para Pocket PC	CP950/ UTF-8	N/D
Coreano	CP1363/ UCS-2	UCS-2 • Instalar habilitador	CP970/ UTF-8	N/D
Japonés	CP943/ UCS-2	UCS-2	CP954/ UTF-8	N/D
Hebreo	N/D	N/D	N/D	N/D

Tabla 53. Soporte NLS (continuación)

Idioma	Windows	WinCE	Linux	Symbian OS
Checo	ISO8859-2/ CP1250/ UCS-2	UCS-2 • Instalar habilitador	N/D	UCS-2
Árabe	N/D	N/D	N/D	N/D
Portugués de Brasil	CP1252/ ISO8859-1/ UCS-2	UCS-2	N/D	N/D
Húngaro	ISO8859-2/ CP1250/ UCS-2	UCS-2	N/D	N/D
Polaco	ISO8859-2/ CP1250/ UCS-2	UCS-2	N/D	N/D
Eslovaco	ISO8859-2/ CP1250/ UCS-2	UCS-2	N/D	N/D

En los sistemas operativos WinCE y Symbian, solamente se puede utilizar Unicode (UCS-2). En los sistemas operativos Linux, la información sobre el entorno local se utiliza para determinar la página de códigos correcta. En las plataformas Windows, las aplicaciones Unicode utilizan Unicode (la aplicación utiliza las API Unicode de DB2 Everyplace) y el resto de aplicaciones utilizan la página de códigos. DB2 Everyplace no proporciona funciones de conversión de página de códigos. Las bases de datos portátiles DB2 Everyplace creadas en un sistema utilizando una página de códigos específica sólo se pueden desplegar en sistemas que utilicen la misma página de códigos. Las tablas que se crearon con una página de códigos específica se pueden utilizar en todos los dispositivos que dan soporte a esa página de códigos, excepto cuando sea necesario un habilitador de idioma determinado. Las aplicaciones que acceden a una base de datos portátil DB2 Everyplace deben encargarse de interpretar correctamente los datos de tipo carácter.

En los sistemas Linux el usuario es el responsable de establecer el valor del entorno local correctamente. (Consulte las páginas man correspondientes a setlocale para obtener una introducción a los nombres de entornos locales en los sistemas Linux). Por ejemplo, puede exportar la variable de entorno LC_CTYPE a ″ja_JP.UTF-8″ y, a continuación, llamar a `setlocale(LC_CTYPE, "")` dentro de la aplicación. Las series codificadas en UTF-8 se procesan en cualquier entorno local con DB2 Everyplace especificando UTF-8 en el nombre del entorno local. Por ejemplo: de_DE.UTF-8.

DB2 Everyplace detecta la codificación utilizada actualmente examinando el entorno local establecido o disponible actualmente.

Habilitadores de idioma de DB2 Everyplace

Para asegurarse de que el dispositivo portátil utilizado puede visualizar correctamente todos los caracteres del idioma que se está utilizando, puede instalar habilitadores de idioma en el dispositivo portátil. La tabla siguiente lista los habilitadores que puede utilizar con DB2 Everyplace.

Tabla 54. Habilitadores de idioma para dispositivos portátiles

Idioma	Habilitador y sistema operativo
Chino tradicional	• Gismosoft Chinese Small_Knife 2.0 sólo para Pocket PC
Checo	• Sunnysoft InterWrite5.5P Pro para Windows CE
Coreano	• CessHan for Casio E-115 1.0 en Windows CE

Soporte de Unicode de DB2 Everyplace

En los sistemas operativos que dan soporte a Unicode (Windows, Windows NT y Windows 2000), DB2 Everyplace acepta series de caracteres Unicode sólo como series de caracteres de Entrada/Salida. Estas series de caracteres UNICODE se guardan en formato UTF–8 dentro del motor de DB2 Everyplace. Un carácter Unicode puede necesitar de uno a tres bytes de espacio de almacenamiento después de la conversión a UTF–8. Una serie de caracteres almacenada en un servidor de bases de datos tal como DB2 Versión 9.1 puede necesitar más espacio cuando la serie de caracteres se descargue y almacene en una base de datos Unicode de DB2 Everyplace.

Notas sobre la interfaz Unicode de CLI:

- Las funciones Unicode de la CLI de DB2 Everyplace tienen un carácter ″W″ añadido al final. Mediante la macro Unicode (que es el valor predeterminado del sistema en Windows CE), las funciones normales de la CLI se correlacionan automáticamente con las funciones Unicode correspondientes. Para escribir código portátil, defina la macro ″Unicode″ y deje que el sistema realice las conversiones.
- Cuando el soporte Unicode está habilitado, los tipos de datos SQL_C_CHAR, SQL_C_TCHAR y SQL_C_WCHAR tienen el mismo significado.
- Muchas funciones de la CLI tienen una longitud de serie (o almacenamiento intermedio) como parámetro de entrada/salida.
  - Para las funciones en las que el *Tipo de argumento* es SQLCHAR* (o SQLWCHAR* para la función W), la longitud es el número de caracteres. Por ejemplo:

    ```
    SQLRETURN   SQLExecDirect    (SQLHSTMT          hstmt,
                                  SQLCHAR     FAR   *szSqlStr,
                                  SQLINTEGER        cbSqlStr);
    ```

    La serie de caracteres Unicode `L"ABCD"` tiene cuatro caracteres.
  - Para las funciones en las que el *Tipo de argumento* es SQLPOINTER, la longitud es el número de bytes. Por ejemplo:

    ```
    SQLRETURN  SQLGetData       (SQLHSTMT          hstmt,
                                 SQLUSMALLINT      icol,
                                 SQLSMALLINT       fCType,
                                 SQLPOINTER        rgbValue,
                                 SQLINTEGER        cbValueMax,
                                 SQLINTEGER  FAR   *pcbValue);
    ```

    Las longitudes del parámetro de entrada `cbValueMax` y del parámetro de salida `*pcbValue` están en bytes. La serie de caracteres Unicode `L"ABCD"` tiene ocho bytes.
- Las funciones Unicode también pueden aceptar SQL_NTS para indicar una serie de caracteres terminada en NULL.

Consejos para escribir código portátil:

- Utilice SQLTCHAR en vez de SQLCHAR o SQLWCHAR.
- Utilice las funciones `_tcs`*`XXXX`* en vez de `str` *`XXXX`* (ANSI) o `wcs`*`XXXX`* (Unicode). Por ejemplo, utilice _tcslen() en vez de wcslen() o strlen().
- Utilice _TEXT() (o TEXT()) para acomodar series literales. Por ejemplo, `_TEXT("ABCD")` se puede interpretar como una serie de caracteres ANSI o Unicode dependiendo de la definición de la macro.
- Utilice `sizeof(NombreMatriz)/sizeof(TCHAR)` para determinar el tamaño de una matriz de caracteres.

DB2eCLP

DB2eCLP es una herramienta que le permite emitir directamente sentencias de SQL desde una interfaz de línea de mandatos.

Esta herramienta es una aplicación que interactúa con DB2 Everyplace a través de una interfaz de línea de mandatos. Se utiliza para la base de datos portátil de DB2 Everyplace en dispositivos portátiles y no es utilizada por el Sync Server. Para encontrar esta herramienta, vaya a uno de los directorios siguientes:

Para usuarios de sistemas de escritorio:

- **Windows** - C:\Archivos de programa\IBM\Lotus\Expeditor\rcp\eclipse\plugins\com.ibm.db2e.win32.x86_9.1.1.0-*fecha_compilación*\os\win32\x86\ donde C:\Archivos de programa\IBM\Lotus\Expeditor\ es el directorio de instalación base de Lotus Expeditor y *fecha_compilación* es la fecha de compilación del producto Expeditor final. Para ejecutar esta herramienta, vaya al directorio indicado y ejecute DB2eCLP.exe.
- **Linux** - /opt/IBM/Lotus/Expeditor/rcp/eclipse/plugins/com.ibm.db2e.linux.x86_9.1.1.0-*fecha_compilación*/os/linux/x86/ donde /opt/IBM/Lotus/Expeditor/ es el directorio de instalación base y *fecha_compilación* es la fecha de compilación del producto Expeditor final. Para Linux, tiene que actualizar la variable LD_LIBRARY_PATH para que incluya el directorio arriba indicado, antes de ejecutar el mandato DB2eCLP.

Para usuarios de dispositivos: Las versiones de dispositivo de la herramienta DB2eCLP se encuentran en el directorio \utils del CD de Lotus Expeditor Client. Seleccione su sistema operativo e idioma en los subdirectorios \utils. Para instalar estos archivos, siga las instrucciones correspondientes a su dispositivo:

- Para instalar DB2eCLP en un dispositivo Windows Mobile 2003SE, Windows Mobile 5.0 o WinCE 5.0:
  1. Abra el Explorador mediante ActiveSync, cambie a la carpeta de instalación de Lotus Expeditor (por ejemplo, \eclipse).
  2. Vaya a la carpeta instalada de DLL nativas de DB2e (por ejemplo, eclipse\plugins\com.ibm.db2e.win32.arm_9.1.0.0-20070409\os\win32\arm) y copie DB2eCLP.exe en esta carpeta.
  3. En el dispositivo, toque en **Programas > Explorador de archivos** para ir a la carpeta de instalación de DLL nativas de DB2e y luego toque en DB2eCLP.exe para iniciar el programa.
- Para instalar DB2eCLP en un dispositivo Symbian S60 tercera edición:
  1. Toque en DB2eCLP.sis en la máquina de desarrollo y luego responda **Sí** a la pregunta de instalación. Realice las operaciones de instalación restantes en el dispositivo.
  2. Para iniciar el programa CLP, toque en el icono DB2eCLP dentro de la carpeta Instalaciones.

Mandatos de DB2eCLP

Este tema muestra los mandatos que puede utilizar en DB2eCLP.

Puede emitir sentencias SQL directamente desde una interfaz de línea de mandatos. Por ejemplo:

```
SELECT * FROM PHONEBOOK
```

En las plataformas Windows, Linux, cada sentencia debe terminar con un punto y coma. Por ejemplo:

```
SELECT * FROM PHONEBOOK;
```

DB2eCLP también permite ejecutar algunos mandatos ampliados:

$file *[archivo de entrada] [archivo de salida]*
: Ejecuta sentencias SQL de un archivo de entrada y escribe el resultado en un archivo de salida. Para todas las plataformas, puede especificar la vía de acceso completa.

AUTOCOMMIT OFF|ON
: Especifica si la aplicación confirma cada sentencia predeterminada (el valor predeterminado del motor es ON). Cuando la modalidad de confirmación automática está activada (true), cada sentencia se trata como una sola transacción completa. AUTOCOMMIT OFF cambia la modalidad de transacción a manual, lo que permite que las aplicaciones retrotraigan o confirmen el trabajo.

BLASTDB
: Descarta todas las tablas de usuario de la base de datos.

CONNECT TO *arg1*

Desconecta automáticamente la aplicación de la conexión actual y vuelve a conectar la aplicación con una base de datos local *arg1*). La especificación está en la llamada de CLI SQLConnect(). El delimitador para las vías de acceso correspondientes a CLI-SQLConnect es \ (barra inclinada invertida) o bien / (barra inclinada). Ambos delimitadores se reconocen en todas las plataformas y se correlacionan con el delimitador apropiado cuando se accede al sistema de archivos, lo que permite que las bases de datos residan en directorios diferentes. Por ejemplo,

```
connect to c:\temp\
create table t (a int)
insert into t values (10)
select * from t
```

CONNECT TO *arg1* **USER** *arg2* **USING** *arg3*

Conecta la aplicación a una base de datos local (**arg1**) utilizando el nombre de usuario (*arg2*) y la contraseña (*arg3*) especificados. Esta información se necesita para acceder a tablas cifradas. Si la aplicación ya está conectada a la base de datos, la conexión en cuestión se descartará.

Un nombre de directorio puede incluir un espacio. Por ejemplo, C:\System\program files\ es una estructura de directorios válida, pero debe utilizar la misma estructura de directorios que exista en su máquina.

DB2LOOK

Extrae las sentencias DDL (lenguaje de definición de datos) necesarias para recrear los objetos de base de datos de la base de datos. DB2LOOK genera las sentencias DDL por tipo de objeto.

Nota: Puede que el DDL generado no reproduzca exactamente todas las características de los objetos SQL originales. Compruebe el DDL generado por DB2LOOK.

DBCHECK archivosalida

Ejecuta la herramienta de comprobación de la integridad de los datos y graba el resultado en un archivo de salida en el directorio de la base de datos. Este mandato sólo recibe soporte en sistemas operativos Linux y Windows de 32 bits.

DESCRIBE SELECT

Describe el tipo, la columna y la longitud del nombre de los datos devueltos por una sentencia SELECT. Por ejemplo:

```
DESCRIBE SELECT * FROM PHONEBOOK
```

DISABLE APPLICATION SET DIRTY

Inhabilita la activación del bit indicador por DB2eCLP.

DISABLE LONG FILENAME

Crea archivos en el formato de nombres 8.3.

DISABLE PHYSICAL DELETE

Inhabilita la modalidad de supresión física (es el valor predeterminado).

DISABLE READ DELETED

Inhabilita la lectura de las filas suprimidas.

DISABLE REORG

Inhabilita la reorganización de tablas.

ENABLE APPLICATION SET DIRTY

Habilita la activación del bit indicador por DB2eCLP.

ENABLE LONG FILENAME

Crea archivos en el formato de nombres largo (es el valor predeterminado).

ENABLE PHYSICAL DELETE

Habilita la modalidad de supresión física. Las filas suprimidas ya no se podrán leer.

ENABLE READ DELETED
Habilita la lectura de las filas suprimidas.

ENABLE REORG
Habilita la reorganización de tablas automáticamente (es el valor predeterminado).

HELP Lista todos los mandatos disponibles.

LIST COLUMNS
Lista todas las columnas de las tablas de usuario de la base de datos.

LIST INDEX
Lista todos los índices ordenados por tabla, nombre, nombre de índice y columna.

LIST TABLES
Lista todas las tablas de usuario de la base de datos.

VERSION
Imprime la cadena de caracteres de la versión de la base de datos portátil DB2 Everyplace. Este mandato devuelve la misma serie de caracteres que la función SQLGetInfo().

Importación y exportación de datos utilizando la herramienta DB2eCLP

La herramienta DB2eCLP para las plataformas Symbian OS, Windows CE, Windows y Linux incorporado permite importar datos de un archivo a DB2 Everyplace y exportar datos de DB2 Everyplace a un archivo.

- **Para importar datos de un archivo de un dispositivo portátil a DB2 Everyplace:**
  1. Escriba `IMPORT FROM nombre_archivo OF DEL INSERT INTO nombre_tabla [(column list)]` donde *nombre_archivo* es el nombre del archivo del que se debe realizar la importación.
- **Para exportar datos de DB2 Everyplace a un archivo:**
  1. Escriba `EXPORT TO nombre_archivo OF DEL sentencia` donde *nombre_archivo* es el nombre del archivo en el que se deben grabar los datos. *sentencia* es la sentencia SELECT para seleccionar los datos que se deben exportar. Por ejemplo, para exportar todos los datos de la tabla denominada mitabla a un archivo denominado miarchivo.txt, escriba:

     ```
     EXPORT TO miarchivo.txt OF DEL SELECT * FROM mitabla
     ```

Cuando se encuentra un error, las herramientas de Importación/Exportación notifican el número de registros procesados.

Soporte de SQL en DB2 Everyplace

Este tema describe las funciones de SQL que se pueden utilizar en DB2 Everyplace.

Sentencias de SQL válidas en DB2 Everyplace

Se pueden emitir interactivamente sentencias de SQL ejecutables desde el dispositivo portátil utilizando DB2eCLP, o bien las sentencias se pueden utilizar en aplicaciones para acceder a datos de una base de datos portátil DB2 Everyplace. La tabla siguiente muestra las sentencias de SQL que se pueden utilizar con DB2 Everyplace.

Tabla 55. Sentencias de SQL admitidas

Sentencia de SQL	Función
ALTER TABLE	Modifica una tabla añadiendo una o más columnas o cambiando la longitud de una o más columnas VARCHAR.
CALL	Llama a un procedimiento almacenado remoto utilizando el Adaptador de consultas y procedimientos almacenados remotos (AgentAdapter) de DB2 Everyplace Sync Server

Tabla 55. Sentencias de SQL admitidas (continuación)

Sentencia de SQL	Función
CREATE INDEX	Crea un índice.
CREATE TABLE	Define una tabla.
DATE	Devuelve una fecha de un valor.
DELETE	Suprime una o más filas de una tabla.
DROP	Suprime una tabla o un índice de una base de datos.
EXPLAIN	Obtiene información sobre la selección de una vía de acceso para una sentencia SELECT.
GRANT	Otorga privilegios de cifrado a un usuario.
INSERT	Inserta una o más filas en una tabla.
LOCK TABLE	Adquiere un bloqueo de tabla compartido o exclusivo sobre una tabla especificada.
REORG TABLE	Elimina o reduce el espacio de almacenamiento desaprovechado correspondiente a la tabla especificada.
REVOKE	Revoca privilegios de cifrado de un usuario.
ROLLBACK	Retrotrae los cambios de base de datos hechos dentro de una unidad de trabajo o punto de rescate.
SAVEPOINT	Establece un punto de rescate dentro de una transacción.
SELECT	Especifica una tabla de resultados consultada a partir de una o más tablas.
TIME	Devuelve una hora de un valor.
TIMESTAMP	Devuelve una indicación de la hora de un valor.
UPDATE	Actualiza los valores de una o más columnas en una o más filas de una tabla.

La “Mensajes de SQLSTATE notificados por SQL” en la página 158 lista los SQLSTATE notificados por el motor SQL de DB2 Everyplace.

La longitud de una sentencia de SQL no puede ser superior a 64.000 caracteres.

El catálogo incluye las tablas del sistema DB2 Everyplace que DB2 Everyplace gestiona: DB2eSYSTABLES, DB2eSYSRELS y DB2eSYSCOLUMNS.

ALTER TABLE

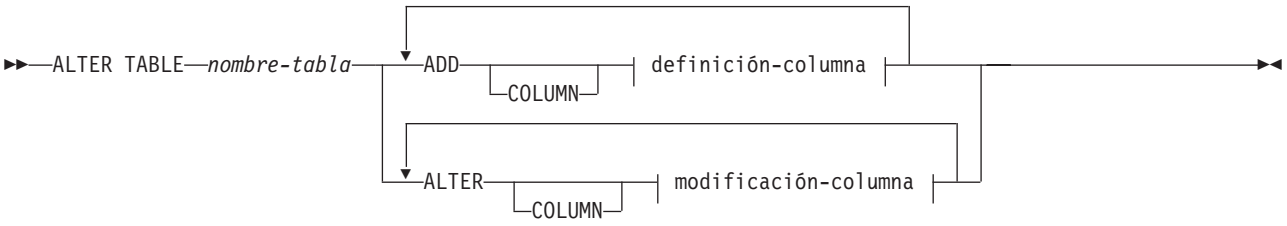
La sentencia ALTER TABLE modifica tablas existentes de uno de los modos que se indican a continuación:

- Añadiendo una o más columnas a una tabla.
- Cambiando la longitud de una o más columnas VARCHAR.

Invocación

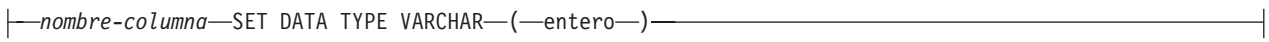
Esta sentencia puede utilizarse en una aplicación utilizando las funciones de DB2 CLI o emitirse mediante la aplicación DB2eCLP.

Sintaxis

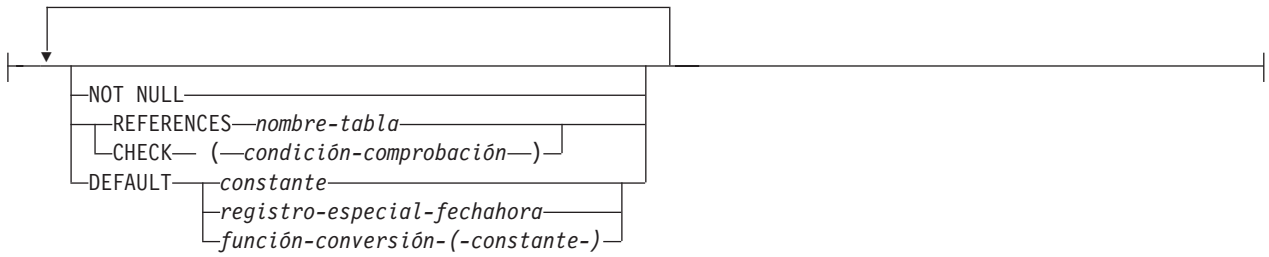


definición-columna::

modificación-columna::



opciones-columna::



Descripción

nombre-tabla
Especifica la tabla que se debe modificar. El nombre puede tener hasta 128 bytes de longitud. El nombre debe identificar una tabla contenida en el catálogo.

Cuando un nombre de tabla contenga espacios en blanco o caracteres especiales, se deberán utilizar identificadores delimitados (con comillas dobles).

Los nombres de tabla pueden incluir caracteres DBCS (de doble byte).

Restricción: Los archivos de datos creados por el sistema que corresponden a tablas creadas y denominadas por nombres de usuario no distinguen entre caracteres en mayúsculas y minúsculas. Por ejemplo, el archivo de datos para una tabla denominada TB se denomina DSY_TB. El archivo de datos para una tabla denominada ″tb″ también es DSY_TB. Por consiguiente le recomendamos firmemente que, para asegurarse de la integridad de los datos, no denomine una tabla utilizando una serie de caracteres idéntica, a excepción de la consideración sobre mayúsculas y minúsculas, a la de un nombre de tabla existente.

nombre-columna
Especifica una columna de la tabla. El nombre puede tener hasta 128 bytes de longitud. El nombre no puede estar calificado y no puede utilizarse un mismo nombre para más de una columna de la tabla.

Los nombres de columna se convierten a mayúsculas antes de almacenarse en el catálogo. Puede utilizar identificadores delimitados (con comillas dobles) para impedir esa conversión. Debe utilizar identificadores delimitados cuando un nombre de columna contenga espacios en blanco o caracteres especiales.

Los nombres de columna pueden incluir caracteres DBCS.

tipo-datos
Es uno de los tipos soportados por la sentencia CREATE TABLE.

opciones-columna
Define opciones adicionales referentes a las columnas de la tabla.

NOT NULL
Impide que la columna contenga valores nulos.

Si no se especifica NOT NULL, la columna puede contener valores nulos, y su valor predeterminado es el valor nulo o el valor proporcionado por la cláusula DEFAULT.

REFERENCES *nombre-tabla*
Consulte la descripción de REFERENCES en el tema siguiente.

CHECK (*condición-comprobación***)**
Consulte la descripción de CHECK en el tema siguiente.

DEFAULT
Proporciona un valor predeterminado cuando no se especifica un valor en una sentencia INSERT.

Si no se especifica DEFAULT en una definición de columna, se utiliza el valor nulo como valor predeterminado de la columna. Si dicha columna se define como NOT NULL, no tendrá un valor predeterminado válido.

constante
Especifica una constante como valor predeterminado de la columna. La constante especificada debe:

- Representar un valor que se pueda asignar a la columna
- No tener dígitos distintos de cero más allá de la escala del tipo de datos de la columna si la constante es una constante decimal.

  Ejemplo: 1,234 no puede ser el valor predeterminado para una columna DECIMAL(5,2).

registro-especial-fechahora
Especifica como valor predeterminado de la columna el valor que tiene el registro especial de fecha-hora (CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP) cuando se ejecuta INSERT. El tipo de datos de la columna debe corresponder al registro especial especificado (por ejemplo, el tipo de datos debe ser DATE cuando se especifica CURRENT DATE).

función-conversión
Especifica la función de conversión como valor predeterminado de la columna. Esta forma de valor predeterminado solamente se puede utilizar con columnas que estén definidas como BLOB o como el tipo de datos datetime (DATE, TIME o TIMESTAMP).

constante
Especifica una constante como argumento. La constante debe cumplir las reglas de una constante para el tipo de datos. Si la función-conversión es BLOB, la constante debe ser una constante de tipo carácter.

REFERENCES *nombre-tabla*
La tabla especificada en la cláusula REFERENCES debe identificar una tabla base que esté descrita en el catálogo, pero no debe identificar una tabla del catálogo.

Una restricción referencial es un duplicado si su clave foránea es igual que la tabla de clave foránea de una restricción referencial especificada anteriormente.

En la explicación siguiente, supongamos que T2 representa la tabla padre identificada y T1 representa la tabla que se está creando.

La clave foránea debe tener el mismo número de columnas que la clave padre de T2 y la descripción de la columna enésima de la clave foránea debe ser comparable con la descripción de la columna

enésima de esa clave padre. Las columnas de fecha-hora no se consideran comparables con las columnas de tipo carácter para los efectos de esta regla. DB2 Everyplace no puede trabajar con claves foráneas.

CHECK (*condición-comprobación***)**
Define una restricción de comprobación. Una *condición-comprobación* es una condición de búsqueda. Una referencia de columna debe ser una columna de la tabla que se está creando. Los valores que se insertan o actualizan en una tabla deben cumplir las restricciones de comprobación que existan.

Si una restricción de comprobación se especifica como parte de una definición de columna, la referencia de columna sólo puede hacerse a la misma columna. Las restricciones de comprobación especificadas como parte de una definición de tabla pueden tener referencias que identifican columnas definidas previamente en la sentencia CREATE TABLE. No se verifica si las restricciones de comprobación contienen incoherencias, condiciones duplicadas o condiciones equivalentes. Por tanto, se pueden definir restricciones de comprobación contradictorias o redundantes.

Aunque se puede especificar la condición de comprobación ″IS NOT NULL″, es recomendable utilizar el atributo NOT NULL para habilitar directamente la capacidad de una columna para contener nulos. Por ejemplo, CHECK (salario + prima > 30000) se acepta si salario es igual a NULL, pues las restricciones de comprobación deben ser ciertas o desconocidas, y en este caso el salario es un valor desconocido. En cambio, CHECK (salario IS NOT NULL) sería una condición falsa y una vulneración de la restricción si salario es igual a NULL.

Las restricciones de comprobación se aplican cuando se actualizan o insertan filas en una tabla.

Todas las restricciones de comprobación definidas en una sentencia CREATE TABLE se combinan y almacenan en el catálogo del sistema. En DB2 Everyplace, esta restricción de comprobación combinada tiene un límite de 32767 bytes.

Reglas

- El total real del número de bytes de una fila no debe ser superior a 65.536.
- Las columnas cuyo tipo de datos es BLOB no pueden tener restricciones de comprobación, predefinidas ni de clave foránea (SQLSTATE 42962).
- Las columnas que contienen datos de tipo BLOB no se pueden utilizar en la clave primaria de una sentencia CREATE TABLE.

Notas

- El usuario no puede modificar las tablas del sistema. Cualquier intento que se realice producirá un SQLSTATE 42832.
- No es posible añadir una columna de clave primaria. Cualquier intento que se realice producirá un SQLSTATE 42601. No obstante, se pueden añadir columnas con valores predeterminados, restricciones de comprobación y restricciones referenciales.
- Al igual que la sentencia CREATE TABLE, una sentencia ALTER TABLE se puede retrotraer en una transacción.
- Al modificar la longitud de una columna, la columna que se debe modificar debe ser de tipo VARCHAR y la longitud especificada debe ser igual o superior a la longitud de la columna existente; de lo contrario, se producirá el SQLSTATE 42837.
- La combinación de cláusulas ADD y ALTER COLUMN en una sentencia ALTER TABLE causará un error de sintaxis.
- Se puede invocar una operación REORG después de ejecutar satisfactoriamente una sentencia ALTER TABLE ADD COLUMN. Esto depende del tamaño de la tabla y del nivel de umbral de reorganización del usuario.
- Las columnas se deben crear utilizando nombres en mayúsculas. Los nombres que mezclan mayúsculas y minúsculas pueden ocasionar errores en algunos lenguajes.
- Números de bytes de datos: La lista siguiente contiene el número de bytes de las columnas de acuerdo con el tipo de datos. Este recuento puede cambiar con cada release de DB2 Everyplace. Cada registro

también incluye información sobre los valores NULL. La información NULL requiere 4 bytes para cada grupo de 32 columnas. Un valor NULL sigue utilizando el tamaño fijo de la columna.

Tipo de datos	Número de bytes de la columna
INTEGER	4
SMALLINT	4
DECIMAL(n, m)	4 – 20
CHAR(n)	n+1
VARCHAR(n)	i+5 donde i es la longitud real
BLOB	i+4 donde i es la longitud real
DATE	4
TIME	4
TIMESTAMP	12

Ejemplo

El ejemplo siguiente muestra alguno de los modos en que se puede utilizar ALTER TABLE.

```
CREATE TABLE t1 (c1 INT PRIMARY KEY NOT NULL, c2 VARCHAR(10));

CREATE TABLE t2 (c1 DATE);

CREATE TABLE t3 (c1 TIME, c2 INT PRIMARY KEY NOT NULL);

ALTER TABLE t2 ADD COLUMN c2 INT REFERENCES t1;

ALTER TABLE t2 ADD c3 INT CHECK (c3 > 1) DEFAULT 10 ADD c4 DECIMAL(5,2) NOT NULL;

ALTER TABLE t2 ADD c5 TIMESTAMP DEFAULT CURRENT TIMESTAMP ADD COLUMN c6 CHAR(20)
DEFAULT 'xyz' ADD c7 INT REFERENCES t3;

CREATE TABLE t4 (c1 INT, c2 VARCHAR(2), c3 VARCHAR(10));

ALTER TABLE t1 ALTER c2 SET DATA TYPE VARCHAR(20)
ALTER c3 SET DATA TYPE VARCHAR(100);
```

CALL

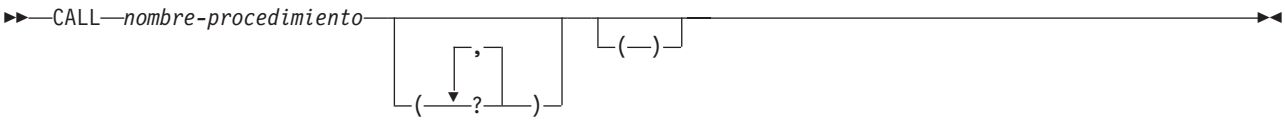
Invoca un procedimiento almacenado definido con el Adaptador de consultas y procedimientos almacenados remotos para DB2 Everyplace Sync Server. Un procedimiento almacenado se ejecuta en la ubicación de la base de datos remota y devuelve datos a la aplicación cliente de DB2 Everyplace.

Los programas que utilizan la sentencia de SQL CALL están diseñados para que se ejecuten en dos partes, una en el cliente y la otra en el servidor.

Invocación

Los procedimientos almacenados remotos se invocan desde una aplicación de DB2 Everyplace pasando la sintaxis de sentencia CALL siguiente a SQLPrepare(), seguida de SQLExecute().

Sintaxis



Descripción

nombre-procedimiento
Identifica el procedimiento que se debe llamar en el servidor remoto. El procedimiento identificado debe estar definido en la suscripción Custom del Sync Server actual.

? El signo de interrogación *?* en el diagrama de la sintaxis de la sentencia CALL denota un marcador de parámetro que se corresponde a un argumento para un procedimiento almacenado. Todos los argumentos se deben pasar utilizando los marcadores de parámetros.

Reglas

Ninguna

Notas

La sentencia CALL utiliza el adaptador de consultas y procedimientos almacenados remotos que se incluye con DB2 Everyplace Sync Server. Es necesario DB2 Everyplace Sync Server para utilizar la sentencia CALL en las aplicaciones de DB2 Everyplace. DB2 Everyplace no da soporte a los procedimientos almacenados locales.

Importante: Cuando recupera un conjunto de resultados utilizando el procedimiento almacenado, el conjunto de resultados completo debe caber en el dispositivo portátil.

Ejemplo

El ejemplo siguiente sólo muestra el código de la sentencia CALL en una aplicación de ejemplo.

Se define un procedimiento almacenado MYPROC() en el servidor fuente para la base de datos `mysample`. Una suscripción Custom está definida en el DB2 Everyplace Sync Server con los atributos siguientes:

```
ID de usuario:  db2admin
Contraseña:  db2admin
Otros:  dbname=mysample;procname= db2e.MYPROC
```

Programa de ejemplo utilizando la sentencia CALL:

```
int main(int argc, char * argv[])
{
  SQLHENV  henv;
  SQLHDBC  hdbc;
  SQLHSTMT hstmt;
  SQLRETURN rc;
  SQLCHAR  strSQL[] = "CALL db2e.MYPROC(?,?,?,?,?)";
  int   nInd4, nInd5;
  int   nSaving = 0, nChecking =0 ;
  int   nCmd =0, nAmount=0;
  SQLCHAR  strConnect[254];

 //*************************************************************
 //* Comprobar parámetros de entrada
 //*************************************************************
 if ( argc < 4 ){
   printf("\nUsage : myClient AccountName Cmd Amount");
```

```
  printf("\n     cmd 1 : query balance");
  printf("\n     cmd 2 : Transfer from Saving to Checking");
  printf("\n     cmd 3 : Transfer from Checking to Saving");
  return (99);
}
nCmd = atoi(argv[2]);
nAmount = atoi(argv[3]);

//*************************************************************
//* Asignar descriptores
//*************************************************************
rc = SQLAllocHandle( SQL_HANDLE_ENV,
     SQL_NULL_HANDLE,
     &henv; //checkerror
rc = SQLAllocHandle( SQL_HANDLE_DBC,
     henv,
     &hdbc); //checkerror
if (argc == 5){
strcpy(strConnect,"http://");
strcat(strConnect,argv[4]);
 strcat(strConnect,
 "/servlet/com.ibm.mobileservices.adapter.agent.AgentServlet?DB=mysample");
}else{
 strcpy(strConnect,
            "http://127.0.0.1:8080/db2e/servlet/

             com.ibm.mobileservices.adapter.agent.AgentServlet?DB=mysample");
}

//*************************************************************
//* Conectar con la base de datos remota
//*************************************************************
rc = SQLConnect(hdbc,
 strConnect,
 SQL_NTS,
 "userex", SQL_NTS,
 "userex", SQL_NTS  ); //checkerror
rc = SQLAllocHandle( SQL_HANDLE_STMT,
     hdbc,
     &hstmt); //checkerror
//*************************************************************
//* Preparar, enlazar y ejecutar la sentencia
//*************************************************************
rc = SQLPrepare(hstmt,strSQL, SQL_NTS); //checkerror
rc = SQLBindParameter(hstmt,
  1,
  SQL_PARAM_INPUT,
  SQL_C_CHAR,
  SQL_CHAR,
  0,
  0,
  (SQLPOINTER)argv[1],
  0,
  NULL ); //checkerror
rc = SQLBindParameter(hstmt,
  2,
  SQL_PARAM_INPUT,
  SQL_C_LONG,
  SQL_INTEGER,
  0,
  0,
  (SQLPOINTER)&nCmd,
  sizeof(int),
  NULL); //checkerror
rc = SQLBindParameter(hstmt,
  3,
  SQL_PARAM_INPUT,
```

```
  SQL_C_LONG,
  SQL_INTEGER,
  0,
  0,
  (SQLPOINTER)&nAmount,
  sizeof(int),
  NULL ); //checkerror
rc = SQLBindParameter(hstmt,
  4,
  SQL_PARAM_OUTPUT,
  SQL_C_LONG,
  SQL_INTEGER,
  0,
  0,
  (SQLPOINTER)&nSaving,
  sizeof(int),
  &nInd4 ); //checkerror
rc = SQLBindParameter(hstmt,
  5,
  SQL_PARAM_OUTPUT,
  SQL_C_LONG,
  SQL_INTEGER,
  0,
  0,
  (SQLPOINTER)&nChecking,
  sizeof(int),
  &nInd5 ); //checkerror
rc = SQLExecute(hstmt); //checkerror
//*****************************************************************
//* Imprimir el saldo
//*****************************************************************
printf("\nSaving = %d",nSaving);
printf("\nChecking = %d",nChecking);

SQLFreeHandle(SQL_HANDLE_STMT, hstmt);
SQLDisconnect(hdbc);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc);
SQLFreeHandle(SQL_HANDLE_ENV, henv);
return 0;
```

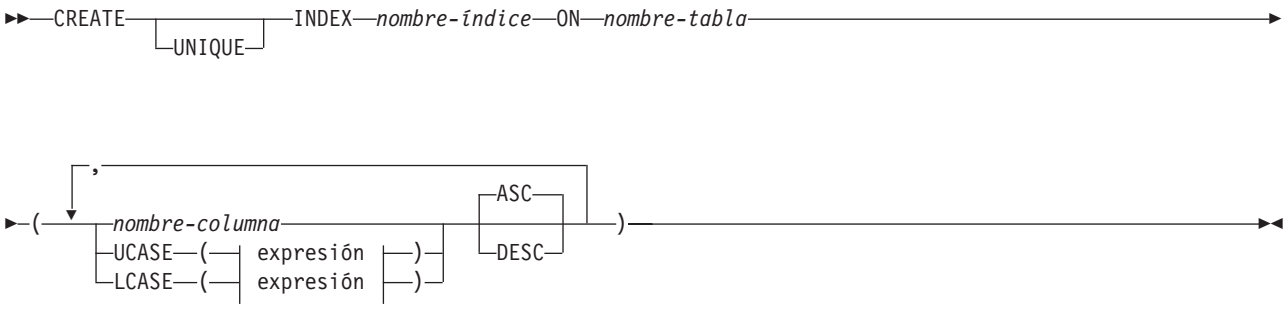
CREATE INDEX

La sentencia CREATE INDEX crea un índice sobre una tabla de DB2 Everyplace.

Invocación

Esta sentencia se puede utilizar en un programa de aplicación utilizando las funciones de DB2 CLI o se puede emitir a través del DB2eCLP.

Sintaxis



Descripción

UNIQUE

Si existe la tabla especificada por ON nombre-tabla, UNIQUE impide que la tabla contenga dos o más filas con el mismo valor para la clave de índice. DB2 Everyplace aplica esta unicidad de valores al final de cada sentencia de SQL de actualización o inserción de filas. DB2 Everyplace también aplica la unicidad de valores durante la ejecución de la sentencia CREATE INDEX. El índice no se creará si la tabla ya contiene filas que tienen valores de clave duplicados.

Cuando utiliza la opción UNIQUE, los valores nulos se tratan como cualquier otro valor. Por ejemplo, si la clave es una columna individual que puede contener valores nulos, esa columna no puede contener más de un valor nulo.

Importante: No cree índices exclusivos con fines de sincronización. La utilización de varios dispositivos portátiles junto con valores exclusivos débilmente distribuidos puede causar una vulneración de una restricción de unicidad.

INDEX *nombre-índice*

Designa el índice.

ON *nombre-tabla*

El parámetro *nombre-tabla* designa una tabla en la que debe crearse un índice.

nombre-columna

Para un índice, el nombre de columna identifica una columna que debe formar parte de la clave de índice.

Cada nombre de columna debe ser un nombre no calificado que identifique una columna de la tabla. Utilice ocho columnas o menos; los nombres de columna no se pueden repetir (SQLSTATE 42711).

La longitud de cada una de las columnas especificadas no puede ser superior a 1024 bytes.

ASC Ordena las entradas de índice en orden ascendente para cada columna. Es el valor predeterminado.

DESC Ordena las entradas de índice en orden descendente para cada columna.

LCASE / UCASE

Las funciones LCASE/UCASE utilizan como entrada una serie de caracteres y devuelven una serie de caracteres en la que todos los caracteres están convertidos a minúsculas o mayúsculas, respectivamente. El argumento debe ser una expresión cuyo valor sea un tipo de datos CHAR o VARCHAR. El resultado de la función tiene el mismo tipo de datos que el argumento. Si el argumento puede ser nulo, el resultado también; si el argumento es nulo, el resultado es el valor nulo.

Los caracteres alfabéticos del argumento se convierten de acuerdo con el valor del entorno local LC_CTYPE que esté vigente para la sentencia. Por ejemplo, los caracteres a-z se convierten a A-Z, y los caracteres con signos diacríticos se convierten en su equivalente LCASE/UCASE, si existe. Los caracteres que no se pueden convertir, permanecen sin convertir en la serie de caracteres.

Importante: La Versión 8 de DB2 Everyplace utilizaba un algoritmo para determinar las formas en mayúscula y minúscula de los caracteres de Latin1. La Versión 9 de DB2 Everyplace utiliza funciones del sistema operativo para manejar caracteres que están fuera del juego de caracteres Latin1. Como resultado, el comportamiento de las aplicaciones que hacen uso de LCASE/UCASE en las plataformas Linux puede cambiar cuando estos programas se ejecutan en DB2 Everyplace Versión 9. Las consultas SELECT que hacen uso de LCASE/UCASE en cláusulas WHERE pueden devolver resultados diferentes. Si las consultas SELECT utilizan índices creados con LCASE/UCASE, puede ser necesario eliminar los índices y volver a crearlos después de la migración para obtener resultados correctos.

Ejemplo: Una columna puede contener información que es una mezcla de datos de tipo UCASE y LCASE. Para facilitar las búsquedas, puede hacer que DB2 Everyplace trate todos los datos como pertenecientes a un solo tipo de letra (mayúsculas o minúsculas). Utilice el siguiente mandato de SQL para crear un índice en el que las entradas de la columna JOB de la tabla EMPLOYEE tienen datos de tipo UCASE en orden ascendente.

```
CREATE INDEX IDX_JOB ON EMPLOYEE (UCASE(JOB) ASC);
```

La tabla siguiente muestra los entornos locales predeterminados que las funciones LCASE/UCASE utilizan en cada plataforma.

Tabla 56. Entornos locales predeterminados utilizados por las funciones LCASE/UCASE

Sistema operativo	Entorno local
Linux	Entorno local predeterminado del sistema operativo. Se puede alterar mediante la función setlocale().
Windows	Entorno local predeterminado del sistema operativo. Se puede alterar mediante la función _tsetlocale().
Windows CE	Entorno local predeterminado del sistema operativo

Restricción: En los sistemas Windows, debe enlazar la aplicación con la misma biblioteca de ejecución C que DB2e.dll. En Visual Studio, seleccione **Configuración → Generación de código C/C++ → Utilizar biblioteca de ejecución → DLL multihebra**.

Los ejemplos siguientes muestran cómo cambiar el entorno local del sistema al alemán en los sistemas Windows y Linux.

Sistemas Windows

```
#include <locale.h>
_tsetlocale(LC_CTYPE, TEXT("German"));
```

Sistemas Linux

```
#include <locale.h>
setlocale(LC_CTYPE, "de_DE.UTF-8");
```

Puede también exportar la variable de entorno LC_CTYPE al entorno local de destino, por ejemplo:

```
export LC_CTYPE = "en_US.UTF-8"
```

Después de exportar el entorno local a una variable de entorno, puede definir ese entorno local en la aplicación utilizando este código:

```
#include <locale.h>
setlocale(LC_CTYPE, "");
```

Reglas

- La sentencia CREATE INDEX puede contener un máximo de 8 columnas.
- Se puede crear un máximo de 15 índices en una tabla sin una clave primaria. Se puede crear un máximo de 14 índices en una tabla con una clave primaria.
- La sentencia CREATE INDEX fallará si se intenta crear un índice que coincida con un índice existente. Se considera que dos descripciones de índice son iguales si se cumplen estas dos condiciones:
  - El conjunto de columnas y su orden en el índice es el mismo que el de un índice existente.
  - Los atributos de ordenación son iguales.
- Las columnas que contienen datos de tipo BLOB no se pueden utilizar en una sentencia CREATE INDEX.

Notas

- DB2 Everyplace da soporte a la exploración bidireccional de índices. Los dos índices siguientes tienen el mismo propósito aunque tienen definiciones diferentes.

  ```
  CREATE INDEX IDX1 ON EMPLOYEE (JOB ASC)
  CREATE INDEX IDX1 ON EMPLOYEE (JOB DESC)
  ```

  En general, los índices se deberían crear sin especificar la dirección del orden. Normalmente, tener menos índices viene a significar unos costes de mantenimiento de índices menor.
- DB2 Everyplace da soporte a la exploración de prefijos de los índices. Considere el siguiente ejemplo. Se crea el índice siguiente.

  ```
  CREATE INDEX J1 ON T (A, B, C, D, E, F, G, K)
  ```

  No es necesario crear otro índice en `T (A,B,C,D)`.
- Si la tabla no contiene datos, CREATE INDEX crea una descripción del índice; las entradas de índice se crean al insertar datos en la tabla.
- Para crear un índice para el bit indicador, utilice el ejemplo siguiente:

  ```
  CREATE INDEX <nombre índice>
    ON <nombre tabla>
    ($dirty)
  ```

Ejemplo

Cree un índice llamado JOB_BY_DPT en la tabla EMPLOYEE. Las entradas de índice se disponen en orden ascendente para cada puesto de trabajo (JOB) dentro de cada departamento (WORKDEPT).

```
CREATE INDEX JOB_BY_DPT
ON EMPLOYEE (WORKDEPT, JOB)
```

CREATE TABLE

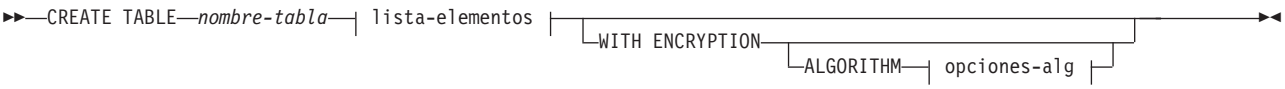
La sentencia CREATE TABLE define una tabla.

La definición debe incluir el nombre de la tabla y los nombres y atributos de sus columnas. La definición puede también incluir otros atributos de la tabla, tales como su clave primaria.

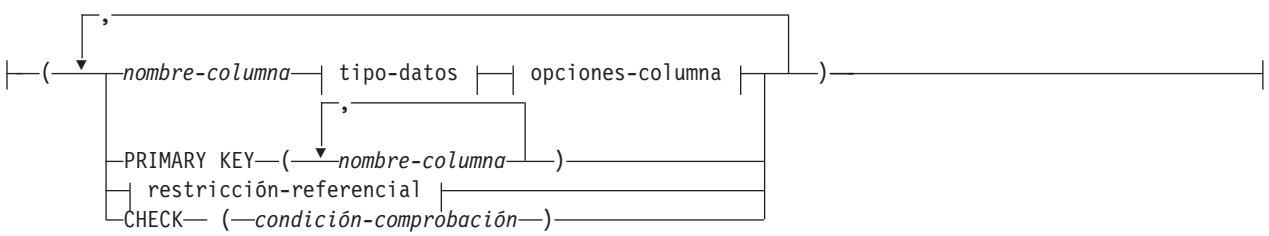
Invocación

Esta sentencia puede utilizarse en un programa de aplicación emitido mediante DB2eCLP.

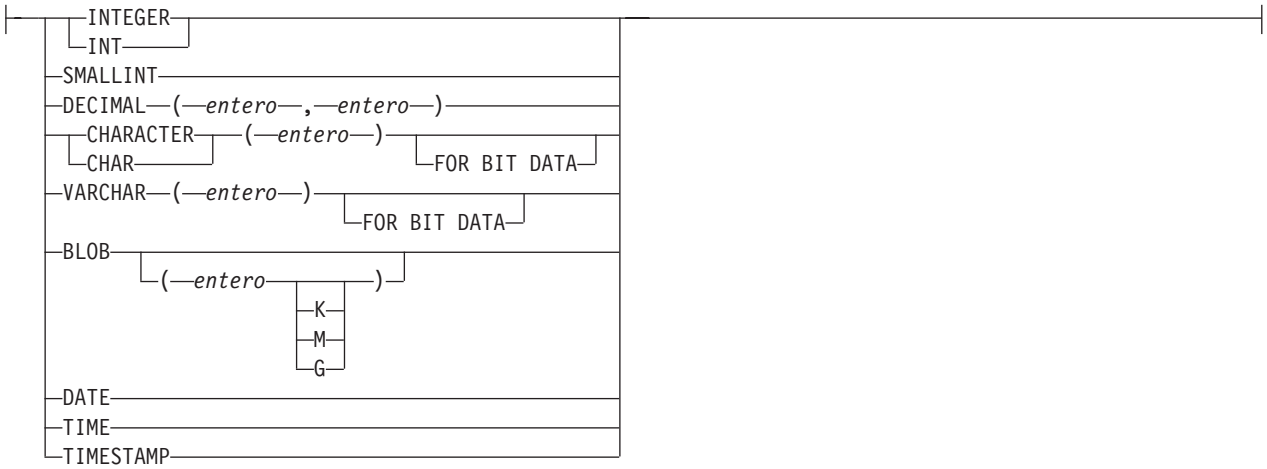
Sintaxis



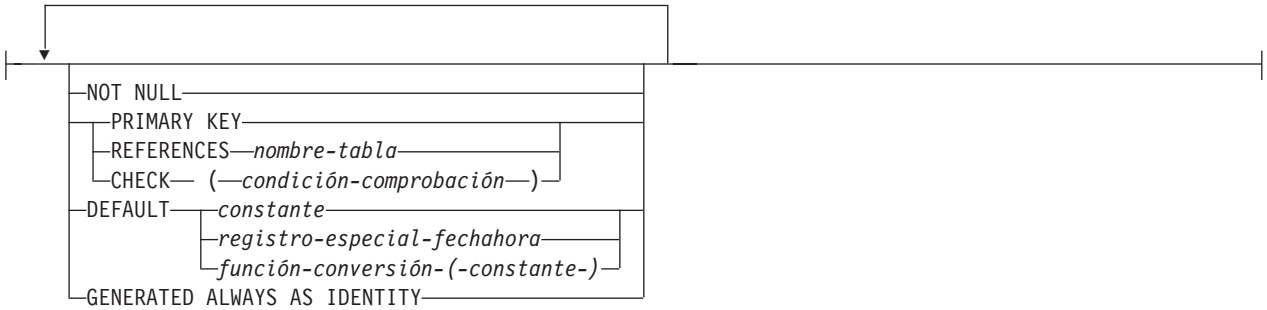
lista-elementos:



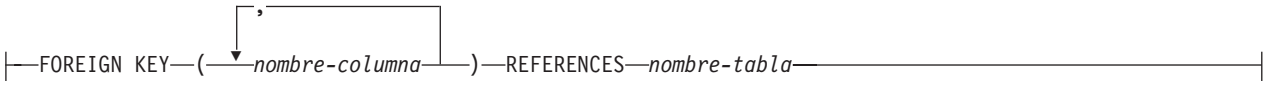
tipo-datos:



opciones-columna:



restricción-referencial:



opciones-alg:

Descripción

nombre-tabla

Designa la tabla. El nombre puede tener un máximo de 128 bytes de longitud. Los nombres de tabla pueden incluir caracteres DBCS (de doble byte). El nombre no debe identificar una tabla contenida en el catálogo. El nombre debe ser exclusivo para el dispositivo portátil.

DB2 Everyplace convierte los nombres de las tablas a mayúsculas antes de almacenar los nombres en el catálogo. Puede utilizar identificadores delimitados (con comillas dobles) para impedir esa conversión. Debe utilizar identificadores delimitados cuando un nombre de tabla contenga espacios en blanco o caracteres especiales.

Restricción: Los archivos de datos creados por el sistema que corresponden a tablas creadas y denominadas por nombres de usuario no distinguen entre caracteres en mayúsculas y minúsculas. Por ejemplo, el archivo de datos para una tabla denominada TB se denomina DSY_TB. El archivo de datos para una tabla denominada "tb" también es DSY_TB. Por consiguiente le recomendamos firmemente que, para asegurarse de la integridad de los datos, no denomine una tabla utilizando una serie de caracteres idéntica, a excepción de la consideración sobre mayúsculas y minúsculas, a la de un nombre de tabla existente.

WITH ENCRYPTION
Crea una tabla de usuario cifrada. Si no especifica el algoritmo de cifrado, el valor predeterminado es DES. Para cifrar una tabla, el usuario debe estar autentificado y conectado. Antes de poder cifrar una tabla, utilice la sentencia GRANT para otorgarse permiso para crear tablas cifradas. Para obtener más información, consulte el apartado "GRANT" en la página 124.

Puede cifrar una tabla sólo cuando la crea. No puede añadir o eliminar cifrado en una tabla existente.

opciones_alg
Especifica el algoritmo de cifrado para una tabla de usuario cifrada. Los valores pueden ser DES o DES3. DES es el acrónimo de Data Encryption Standard (Estándar de cifrado de datos). DES3 es triple DES. Para obtener más información sobre estos dos términos, consulte el "Glosario" en la página 163.

nombre-columna
Designa una columna de la tabla. El nombre puede tener un máximo de 128 bytes de longitud. Los nombres de columna pueden incluir caracteres DBCS. El nombre no puede estar calificado y no puede utilizarse un mismo nombre para más de una columna de la tabla.

DB2 Everyplace convierte los nombres de las columnas a mayúsculas antes de almacenar los nombres en el catálogo. Puede utilizar identificadores delimitados (con comillas dobles) para impedir esa conversión. También debe utilizar identificadores delimitados cuando un nombre de columna contenga espacios en blanco o caracteres especiales.

tipo-datos
Es uno de los tipos mostrados en la lista siguiente:

INTEGER o INT
Entero de cuatro bytes, con signo, comprendido entre 2147483647 y -2147483648.

SMALLINT
Entero de dos bytes, con signo, comprendido entre -32768 y 32767.

DECIMAL*(entero-precisión, entero-escala)*
Un número decimal. El primer entero indica la precisión del número; es decir, el número total de dígitos; su valor está comprendido entre 1 y 31. El segundo entero es la escala del número; es decir, el número de dígitos situados a la derecha de la coma decimal; su valor está comprendido entre 0 y la precisión del número.

CHAR*(entero)* **oCHARACTER***(entero)*
Serie de caracteres de longitud fija, cuya longitud es *entero*, que está comprendido entre 1 y 32767.

FOR BIT DATA
Especifica que el contenido de la columna se debe tratar como datos (binarios) de bits. La longitud está comprendida entre 1 byte y 254 bytes. DB2 Everyplace no realiza conversiones de páginas de códigos cuando intercambia datos con otros sistemas. DB2 Everyplace realiza las comparaciones en binario, independientemente de la secuencia de clasificación de la base de datos.

VARCHAR*(entero)*
Serie de caracteres de longitud variable, cuya longitud máxima es *entero*, que está comprendido entre 1 y 32767.

FOR BIT DATA
Especifica que el contenido de la columna se debe tratar como datos (binarios) de bits. La longitud está comprendida entre 1 byte y 32672 bytes. DB2 Everyplace no realiza conversiones de páginas de códigos cuando intercambia datos con otros sistemas. DB2 Everyplace realiza las comparaciones en binario, sin importar la secuencia de clasificación de la base de datos. VARCHAR FOR BIT DATA se correlaciona con los tipos de datos siguientes:

Tabla 57. Correlaciones de tipos de datos para tipos de datos VARCHAR FOR BIT DATA

API	java.sql.Types	Tipo Java	DB2eType
JDBC	VARBINARY	byte[]	

BLOB o BINARY LARGE OBJECT*(entero [K | M | G])*

Serie de objeto binario grande de la longitud máxima especificada *entero* en bytes. La longitud de un BLOB puede estar comprendida entre 1 byte y 2 147 483 647 bytes.

Las formas cortas *K*, *M* y *G* se pueden utilizar como modificadores de longitud en las creaciones de tablas.

- *K* significa kilobyte (1024 bytes)
- *M* significa megabyte (1024 K)
- *G* significa gigabytes (1024 M)

Si se especifica *entero* a solas, esa es la longitud máxima.

Si se especifica *entero K* (en mayúsculas o minúsculas), la longitud máxima es 1 024 veces el valor de *entero*. El valor máximo para entero es 2 097 152. Si se especifica un múltiplo de K, M o G que produce un valor igual a 2 147 483 648, el valor real utilizado es 2 147 483 647 (es decir, 2 gigabytes menos 1 byte), que es la longitud máxima para una columna LOB.

Si se especifica *entero M*, la longitud máxima es 1 048 576 veces el valor de *entero*. El valor máximo de *entero* es 2 048.

Si se especifica *entero G*, la longitud máxima es 1 073 741 824 veces el valor de *entero*. El valor máximo de *entero* es 2.

Si se omite la especificación de la longitud, se utiliza la longitud 1 048 576 (1 megabyte).

Se permite un número cualquiera de espacios en blanco entre *entero* y K, M o G, y no necesario que exista un espacio en blanco.

Por ejemplo:

```
CREATE TABLE t1 (c1 BLOB(5M));  // 5 M BLOB
CREATE TABLE t2 (c1 blob( 20 K ), c2 blob (1G));  // 20 K and 1 G BLOB
CREATE TABLE t3 (c1 BLOB (1024)); // 1 K BLOB
```

Un BLOB que tenga más de 32 kilobytes se almacena en un archivo aparte con el convenio de denominación DSY_b*xxxnombre-tabla*:

- *xxx*: Tres dígitos que representan la secuencia de creación del archivo.
- *nombre-tabla*: El nombre de la tabla donde se inserta el BLOB.

Cada archivo sólo almacena un BLOB de este tipo.

DATE
Una fecha. Un valor de entrada puede tener uno de estos formatos: MM/DD/AAAA,

AAAA-MM-DD o MM.DD.AAAA. DB2 Everyplace muestra el valor de fecha solamente en el formato ISO, AAAA-MM-DD. El año de un valor de fecha puede estar comprendido entre 0001 y 9999.

El registro especial CURRENT DATE también genera la fecha actual en el formato ISO.

TIME
Una hora. Es un valor de entrada que puede tener uno de los formatos siguientes: HH:MM am (o pm), HH:MM:SS, HH.MM am (o pm), o HH.MM.SS. Los SS, segundos, son opcionales con los formatos HH:MM:SS o HH.MM.SS. DB2 Everyplace muestra el valor de hora solamente en el formato ISO, HH:MM:SS.

El registro especial CURRENT TIME también genera la hora actual en el formato ISO.

TIMESTAMP
Una indicación de fecha y hora. Un valor de entrada debe tener el siguiente formato: AAAA-MM-DD-HH.MM.SS.ZZZZZZ. DB2 Everyplace muestra el valor de fecha y hora en el formato siguiente: AAAA-MM-DD-HH.MM.SS.ZZZZZZ.

El registro especial CURRENT TIMESTAMP también genera la indicación de hora actual.

opciones-columna
Define opciones adicionales referentes a las columnas de la tabla. Estas opciones para columnas pueden ser cualquiera de las siguientes:

NOT NULL
Impide que la columna contenga valores nulos.

Si no especifica NOT NULL, la columna puede contener valores nulos. El valor predeterminado de la columna es el valor nulo o el valor proporcionado por la cláusula DEFAULT.

PRIMARY KEY
Proporciona un método abreviado para definir una clave primaria formada por una sola columna. Si especifica PRIMARY KEY en la definición de la columna C, el efecto es el mismo que si especifica PRIMARY KEY(C) como cláusula separada.

REFERENCES *nombre-tabla*
vea la descripción de *restricción-referencial*.

CHECK (*condición-comprobación***)**
vea la descripción de *restricción-referencial*.

DEFAULT
Proporciona un valor predeterminado cuando no se especifica un valor en una sentencia INSERT.

Si no incluye DEFAULT de una definición de columna, DB2 Everyplace utiliza el valor nulo como valor predeterminado de la columna. Si define una columna como NOT NULL, la columna no tiene un valor predeterminado válido. Si la cláusula DEFAULT no se puede especificar, DB2 Everyplace devuelve el SQLState 42623.

constante
Especifica una constante como valor predeterminado de la columna. La constante especificada debe:

- Representar un valor que se pueda asignar a la columna

  Ejemplo: `CREATE TABLE t1 (c1 INT DEFAULT 100, c2 VARCHAR(10) FOR BIT DATA DEFAULT x'FFFF');`
- No tener dígitos distintos de cero más allá de la escala del tipo de datos de la columna si la constante es una constante decimal.

  Ejemplo: 1,234 no puede ser el valor predeterminado para una columna DECIMAL(5,2).

registro-especial-fechahora
Especifica como valor predeterminado de la columna el valor que tiene el registro especial de

fecha-hora (CURRENT DATE, CURRENT TIME o CURRENT TIMESTAMP) cuando se ejecuta INSERT. El tipo de datos de la columna debe ser el tipo de datos que corresponde al registro especial especificado.

Ejemplo: Si especifica CURRENT DATE, el tipo de datos de la columna debe ser DATE.

función-conversión
Especifica la función de conversión como valor predeterminado de la columna. Esta forma de valor predeterminado solamente se puede utilizar con columnas que estén definidas como BLOB o como el tipo de datos datetime (DATE, TIME o TIMESTAMP).

constante
Especifica una constante como argumento. La constante debe cumplir las reglas de una constante para el tipo de datos. Si la función-conversión es BLOB, la constante debe ser una constante de tipo carácter.

GENERATED ALWAYS AS IDENTITY
Cuando crea una tabla, puede especificar una columna como "GENERATED ALWAYS AS IDENTITY". DB2 Everyplace crea el valor de esta columna cada vez que se realiza una operación INSERT o INSERT con sub–SELECT. El tipo de datos de esta columna debe ser un tipo numérico (INTEGER, SMALLINT o DECIMAL). DB2 Everyplace crea automáticamente números secuenciales exclusivos a partir de 1, que se incrementan en 1 cada vez.

El valor generado para la columna IDENTITY comienza en 1 y aumenta de 1 en 1 cada vez que una fila se inserta en la tabla. De este modo, se garantiza la exclusividad, aunque DB2 Everyplace no crea automáticamente un índice en una columna IDENTITY. Si desea disponer de un índice en una columna IDENTITY, deberá o crear un índice explícitamente, o especificar la columna como PRIMARY KEY. Cuando una columna IDENTITY alcanza su valor máximo, la emisión de sentencias INSERT subsiguientes produce un error (SQLSTATE 23522). El valor máximo de una columna IDENTITY de los tipos INT y SMALLINT es el valor máximo permitido para esos dos tipos. El valor máximo de una columna IDENTITY de tipo DECIMAL está determinado por:

- La precisión y escala del tipo de datos
- El valor máximo permitido para una columna IDENTITY: 2.15* (10∧18) (19 dígitos decimales))

a menor de estas restricciones el límite de rango. Para una columna IDENTITY de tipo DECIMAL, la parte fraccionaria del valor es siempre 0. DB2 Everyplace aumenta la parte entera del tipo en una unidad cada vez.

Solamente puede definir la especificación IDENTITY para las columnas cuyo tipo de datos es uno de los 3 tipos numéricos: INT, SMAIINT, DECIMAL. En otro caso, se emite un error (SQLSTATE 42815). Puede existir como máximo una sola columna IDENTITY por tabla (de lo caso contrario, se emite el error SQLSTATE 428C1). El usuario no puede proporcionar un valor para una columna IDENTITY en una sentencia INSERT (debe tomar por omisión el valor generado por el sistema de DB2 Everyplace), ni tampoco actualizar (UPDATE) una columna IDENTITY.

PRIMARY KEY (*nombre-columna***, ...)**
Define una clave primaria formada por las columnas identificadas. Esta cláusula no se puede especificar más de una vez y las columnas indicadas no deben estar definidas como NOT NULL. Cada nombre de columna debe identificar una columna de la tabla y una misma columna no se debe identificar más de una vez.

El número de columnas especificadas no debe ser superior a 8.

Se crea automáticamente un índice exclusivo en las columnas especificadas.

Sólo se puede definir una sola clave primaria para una tabla.

El atributo de longitud de cada columna especificada no debe ser mayor que 1024 bytes.

restricción-referencial
Define una restricción referencial.

FOREIGN KEY (*nombre-columna***, ...)**
Define una restricción referencial con el nombre de restricción especificado.

Supongamos que T1 denota la tabla objeto de la sentencia. La clave foránea de la restricción referencial está formada por las columnas identificadas. Cada nombre de la lista de nombres de columna debe identificar una columna de T1 y una misma columna no se debe identificar más de una vez. El número de columnas especificadas no debe ser superior a 8. DB2 Everyplace no da soporte a las claves foráneas.

REFERENCES *nombre-tabla*
La tabla especificada en la cláusula REFERENCES debe identificar una tabla base que esté descrita en el catálogo, pero no debe identificar una tabla del catálogo.

Una restricción referencial es un duplicado si su clave foránea es igual que la tabla de clave foránea de una restricción referencial especificada anteriormente.

En la explicación siguiente, supongamos que T2 representa la tabla padre identificada y T1 representa la tabla que se está creando.

La clave foránea debe tener el mismo número de columnas que la clave padre de T2 y la descripción de la columna enésima de la clave foránea debe ser comparable con la descripción de la columna enésima de esa clave padre. Las columnas de fecha-hora no se consideran comparables con las columnas de tipo carácter para los efectos de esta regla. DB2 Everyplace no da soporte a las claves foráneas.

CHECK (*condición-comprobación***)**
Define una restricción de comprobación. Una *condición-comprobación* es una condición de búsqueda. Una referencia de columna debe ser una columna de la tabla que se está creando. Los valores que se insertan o actualizan en una tabla deben cumplir las restricciones de comprobación que existan.

Si una restricción de comprobación se especifica como parte de una definición de columna, la referencia de columna sólo puede hacerse a la misma columna. Las restricciones de comprobación especificadas como parte de una definición de tabla pueden tener referencias que identifican columnas definidas previamente en la sentencia CREATE TABLE. No se verifica si las restricciones de comprobación contienen incoherencias, condiciones duplicadas o condiciones equivalentes. Por tanto, se pueden definir restricciones de comprobación contradictorias o redundantes.

Aunque se puede especificar la condición de comprobación ″IS NOT NULL″, es recomendable utilizar el atributo NOT NULL para habilitar directamente la capacidad de una columna para contener nulos. Por ejemplo, CHECK (salario + prima > 30000) se acepta si salario es igual a NULL, pues las restricciones de comprobación deben ser ciertas o desconocidas, y en este caso el salario es un valor desconocido. En cambio, CHECK (salario IS NOT NULL) sería una condición falsa y vulneraría la restricción si salario = NULL.

Las restricciones de comprobación se aplican cuando se actualizan o insertan filas en una tabla.

Todas las restricciones de comprobación definidas en una sentencia CREATE TABLE se combinan y almacenan en el catálogo del sistema. En DB2 Everyplace, esta restricción de comprobación combinada tiene un límite de 512 bytes.

Reglas

- El total real del número de bytes de una fila no debe ser superior a 65 536.

  Consulte el apartado “Notas” en la página 117 para obtener más información.
- Las columnas cuyo tipo de datos es BLOB no pueden tener restricciones de comprobación, predefinidas ni de clave foránea (SQLSTATE 42962).
- Las columnas que contienen datos de tipo BLOB no se pueden utilizar en la clave primaria de una sentencia CREATE TABLE.

Notas

- Si especifica demasiadas columnas en la definición de tabla, DB2 Everyplace devuelve el SQLSTATE 54011. Consulte "Límites de DB2 Everyplace" en la página 53 para obtener información sobre el número máximo de columnas por tabla.
- Las tablas y las columnas se deben crear utilizando nombres en mayúsculas. Los nombres que mezclan mayúsculas y minúsculas pueden ocasionar errores en algunos lenguajes.
- Si crea una nueva tabla en su dispositivo portátil, la tabla no se crea automáticamente en la base de datos corporativa cuando sincroniza el dispositivo portátil con el servidor. Es necesario crear la tabla en la base de datos corporativa antes de realizar la sincronización.
- Números de bytes de datos: La lista siguiente contiene los números de bytes de las columnas de acuerdo con su tipo de datos. Esto puede cambiar en cada release. Cada registro también incluye información sobre los valores NULL. La información NULL requiere 4 bytes para cada grupo de 32 columnas. Un valor NULL sigue utilizando el tamaño fijo de la columna.

Tipo de datos	Número de bytes de la columna
INTEGER	4
SMALLINT	4
DECIMAL(n, m)	4 – 20
CHAR(n)	n+1
VARCHAR(n)	i+5 donde i es la longitud real
BLOB	i+4 donde i es la longitud real
DATE	4
TIME	4
TIMESTAMP	12

Ejemplo

En este ejemplo se crea una tabla EMPLOYEE con los nombres de columna EMPNO, FIRSTNAME, LASTNAME, DEPT, PHONENO, SALARY y HIREDATE. CHAR significa que la columna puede contener datos de tipo carácter. NOT NULL significa que la columna no puede contener un valor nulo. VARCHAR significa que la columna puede contener datos de tipo carácter de longitud variable. La clave primaria está formada por la columna EMPNO.

```
CREATE TABLE EMPLOYEE
  (EMPNO       CHAR(3)    PRIMARY KEY,
   FIRSTNAME   VARCHAR(12)   NOT NULL,
   LASTNAME    VARCHAR(15)   NOT NULL,
   DEPT        CHAR(3),
   PHONENO     CHAR(4),
   SALARY      INT,
   HIREDATE    DATE)
```

COMMIT

La sentencia COMMIT concluye una unidad de trabajo y confirma los cambios de base de datos que fueron realizados por esa unidad de trabajo.

Invocación

Esta sentencia se puede integrar en un programa de aplicación o ser emitida interactivamente. Es una sentencia ejecutable que se puede preparar dinámicamente.

Sintaxis

Descripción

Concluye la unidad de trabajo en la que se ejecuta la sentencia COMMIT y se inicia una nueva unidad de trabajo. Se confirman todos los cambios realizados por las sentencias siguientes ejecutadas durante la unidad de trabajo: ALTER, CREATE, DROP, GRANT, LOCK TABLE, REVOKE, y las sentencias de cambio de datos (INSERT, DELETE, UPDATE).

Se liberan todos los bloqueos adquiridos por la unidad de trabajo posteriores a su iniciación.

Los cursores abiertos permanecen abiertos, y el cursor se coloca delante de la siguiente fila lógica del conjunto de resultados. Se liberan todos los localizadores de LOB. Esto es así aunque los localizadores estén asociados a valores de LOB recuperados mediante un cursor.

Se liberan todos los puntos de rescate que se definieron dentro de la transacción.

Notas

Cada proceso de aplicación debe concluir explícitamente su unidad de trabajo antes de que finalice. Si el programa de aplicación concluye normalmente sin una sentencia COMMIT o ROLLBACK, el motor de base de datos hará una retrotracción implícita de la última unidad de trabajo activa.

Ejemplo

Confirmar modificaciones hechas en la base de datos desde el último punto de confirmación.

```
COMMIT WORK
```

DATE

La función DATE devuelve una fecha de un valor.

Invocación

Esta sentencia puede utilizarse en una aplicación utilizando las funciones de DB2 CLI o emitirse mediante DB2eCLP.

Sintaxis

Descripción

expresión
Especifica un valor. El argumento debe ser una fecha, una indicación de la hora o una representación válida de la fecha o la indicación de la hora en forma de serie.

Reglas

Si el argumento puede ser nulo, el resultado también; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo

Supongamos que la columna RECEIVED (del tipo de datos TIMESTAMP) tiene un valor de datos de '2003-12-25-17.12.30.000000'.

- DATE(RECEIVED) es '2003-12-25' (del tipo de datos DATE).
- DATE('2003-12-25') es '2003-12-25' (del tipo de datos DATE).
- DATE('25.12.2003') es '2003-12-25' (del tipos de datos DATE).

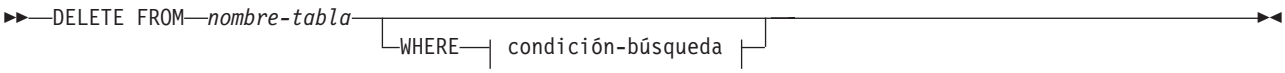
DELETE

La sentencia DELETE suprime una o más filas de una tabla.

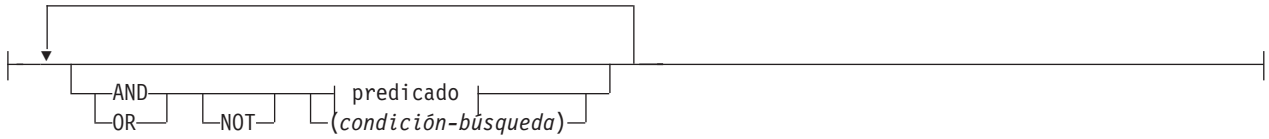
Invocación

Esta sentencia se puede utilizar en un programa de aplicación utilizando las funciones de DB2 CLI o se puede emitir mediante el DB2eCLP.

Sintaxis



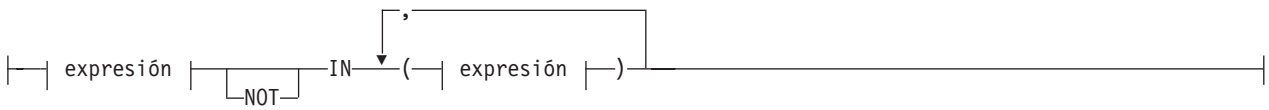
condición-búsqueda:



predicado:

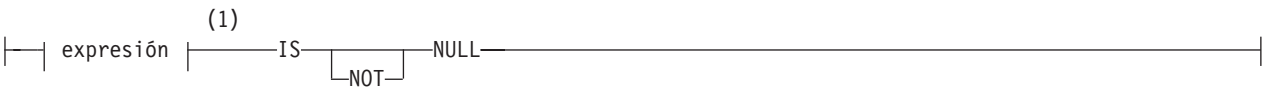
predicado básico:

predicado IN:

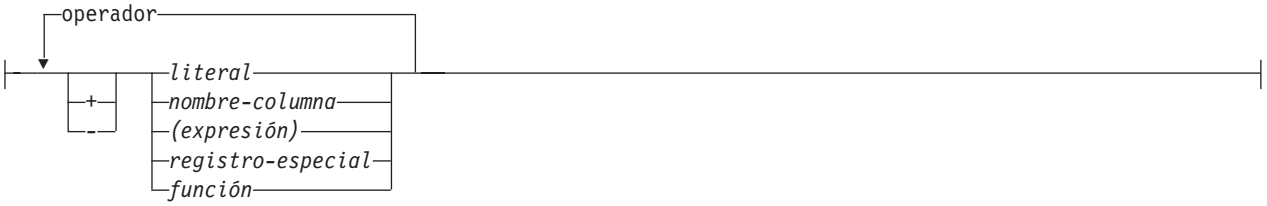


predicado LIKE:

predicado NULL:



expresión:



operador:

Notas:

1 Las expresiones BLOB solamente están permitidas en predicados NULL.

Descripción

FROM *nombre-tabla*
Designa la tabla en la que deben suprimirse filas. El nombre debe especificar una tabla existente en el catálogo, pero no una tabla de catálogo.

WHERE
Especifica una condición que selecciona las filas que deben suprimirse. Se puede omitir la cláusula o especificar una condición de búsqueda. Si se omite la cláusula, se suprimen todas las filas de la tabla.

condición-búsqueda
Una *condición-búsqueda* especifica una condición que es verdadera, falsa o desconocida para una fila determinada.

El resultado de una *condición-búsqueda* se obtiene aplicando los *operadores lógicos* especificados (AND, OR, NOT) al resultado de cada predicado especificado. Un predicado compara dos valores. Si no se especifican operadores lógicos, el resultado de la condición de búsqueda es el resultado del predicado especificado.

Las condiciones de búsqueda que están entre paréntesis se evalúan en primer lugar. Si no se especifica un orden de evaluación mediante el uso de paréntesis, NOT se aplica antes que AND y AND se aplica antes que OR. El orden en el que se evalúan los operadores con igual nivel de prioridad es indefinido, para permitir la optimización de las condiciones de búsqueda.

La *condición-búsqueda* se aplica cada fila de la tabla y las filas suprimidas son aquéllas para las cuales el resultado de la *condición-búsqueda* es verdadero.

Cada *nombre-columna* especificado en la condición de búsqueda debe identificar una columna de la tabla.

NOT
Si se especifica NOT, se invierte el resultado del predicado.

expresión
Identifica un operando del predicado. La *expresión* puede ser un literal, nombre de columna, registro especial o función.

No se da soporte a las operaciones aritméticas sobre los tipos de datos BLOB(n), DATE, TIME y TIMESTAMP.

literal
Un *literal* puede ser un valor cuyo tipo de datos es INTEGER, SMALLINT, DECIMAL, CHAR(n), VARCHAR(n), BLOB(n), DATE, TIME o TIMESTAMP.

nombre-columna
Identifica la columna que es un operando del predicado.

registro-especial
Identifica el registro especial que es un operando del predicado. Se pueden utilizar los registros especiales CURRENT DATE, CURRENT TIME y CURRENT TIMESTAMP para generar la fecha, la hora o la indicación de hora actuales.

función
Puede incluir sólo las funciones MOD, LENGTH y RTRIM.

operador relacional
Puede ser cualquiera de los operadores siguientes:

= Igual a.

<> No igual a.

< Menor que.

> Mayor que.

<= Menor o igual que.

>= Mayor o igual que.

LIKE Coincide con una serie de caracteres. Utilice un símbolo de subrayado de SBCS (juego de caracteres de un solo byte) para representar un carácter SBCS individual. Utilice un símbolo de subrayado de DBCS (juego de caracteres de doble byte) para representar un carácter DBCS individual. Por ejemplo, la condición `WHERE PART_NUMBER LIKE '_0'` devuelve todos los números de pieza de 2 dígitos que terminan en 0 (20, 30 y 40, por ejemplo). Utilice un símbolo de porcentaje (del juego de caracteres SBCS o DBCS) para representar una cadena de caracteres SBCS o DBCS, o la ausencia de caracteres. Por ejemplo, la condición `WHERE DEPT_NUMBER LIKE '2%'` obtiene todos los números de departamento (DEPT_NUMBER) que comienzan con el número 2 (por ejemplo, 20, 27 ó 234).

NOT LIKE
Indica que al menos uno de los caracteres es diferente.

IS NULL
Contiene el valor nulo.

IS NOT NULL
No contiene el valor nulo.

AND
Si se especifica, el operador lógico AND se aplica al resultado de cada predicado especificado.

OR
Si se especifica, el operador lógico OR se aplica al resultado de cada predicado especificado.

Reglas

Ninguna.

Notas

- Un DELETE lógico no se aplica nunca a registros suprimidos lógicamente.

Ejemplo

Supresión del número de empleado (EMPNO) 003002 de la tabla EMPLOYEE:

```
DELETE FROM EMPLOYEE
  WHERE EMPNO = '003002'
```

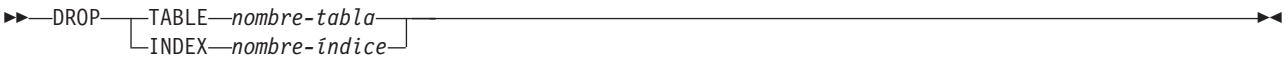
DROP

La sentencia DROP suprime una tabla o índice.

Invocación

Esta sentencia se puede utilizar en un programa de aplicación utilizando las funciones de DB2 CLI o se puede emitir a través del DB2eCLP.

Sintaxis



Descripción

TABLE *nombre-tabla*
Designa la tabla base que se debe eliminar. *nombre-tabla* debe identificar una tabla que esté descrita en el catálogo (SQLSTATE 42704).

INDEX *nombre-índice*
Designa el índice que se debe eliminar. *nombre-índice* debe identificar un índice que esté descrito en el catálogo (SQLSTATE 42704). No puede ser un índice que el sistema necesite para una clave primaria (SQLSTATE 42704).

Reglas

Ninguna.

Notas

- La eliminación de tablas o índices mientras se está utilizando una tabla (es decir, cuando un descriptor de sentencia está activo para una consulta que hace uso de esa tabla o índice) invalida los correspondientes descriptores de sentencia.

Ejemplo

Eliminación de la tabla EMPLOYEE.

```
DROP TABLE EMPLOYEE
```

EXPLAIN

La sentencia EXPLAIN obtiene información sobre la selección de una vía de acceso para una sentencia SELECT.

La información obtenida se coloca en una tabla de usuario llamada DB2ePLANTABLE.

La sentencia EXPLAIN se puede utilizar en los sistemas operativos siguientes:

- Windows (Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP y Windows 2003)
- Linux

Invocación

Esta sentencia se puede utilizar en un programa de aplicación utilizando las funciones de DB2 CLI o se puede emitir a través del DB2eCLP.

Sintaxis

Descripción

SET QUERYNO = entero
Asocia *entero* a la sentencia SELECT. Se asigna el valor *entero* a la columna QUERYNO en cada fila insertada por la sentencia EXPLAIN en la tabla PLAN.

Sentencia SELECT
Especifica un conjunto de columnas nuevas en el formato de la tabla de resultados de una sentencia SELECT.

Reglas

El valor *entero* debe ser positivo.

Notas

- Cuando utiliza la sentencia EXPLAIN, se crea automáticamente por omisión la tabla DB2ePLANTABLE, si no existe.
- Para crear explícitamente la tabla DB2ePLANTABLE, siga el ejemplo siguiente:

```
create table "DB2ePLANTABLE"
 (query_no int, plan_no int, table_name char(128), index_name char(128),
sort_temp char(1), expl_timestamp timestamp, remarks varchar(300))
```

La Tabla 58 describe las columnas de DB2ePLANTABLE.

Tabla 58. Información sobre las columnas de DB2ePLANTABLE

Nombre de la columna	Descripción
query_no	Número entero que asocia la sentencia EXPLAIN con los datos de salida dentro de DB2ePLANTABLE.
plan_no	Número entero que representa los pasos en los que se ejecuta la sentencia (en orden ascendente).
table_name	El nombre de la tabla o nombre correlacionado que identifica unívocamente la tabla, o un valor nulo si no es aplicable un nombre.
index_name	El nombre del índice (si se utiliza) para acceder a la tabla. Devuelve un valor nulo si no se utiliza ningún índice.

Tabla 58. Información sobre las columnas de DB2ePLANTABLE (continuación)

Nombre de la columna	Descripción
sort_temp	'Y' significa que es necesario realizar una clasificación en una tabla temporal para procesar una sentencia GROUP BY o ORDER BY. Si se devuelve un valor nulo, significa que no es necesaria ninguna tabla temporal de clasificación.
expl_timestamp	Indicación horaria del momento en que se ejecuta la sentencia EXPLAIN.
comentarios	La columna de comentarios contiene el valor nulo. El usuario puede añadir comentarios a esta columna con fines contables.

- DB2ePLANTABLE es una tabla de usuario que puede ser modificada o eliminada por cualquier aplicación.

Ejemplo

Al desarrollar una nueva aplicación, es aconsejable determinar qué vía de acceso se elige para una sentencia SELECT. En este ejemplo, una nueva aplicación consulta las tablas SALES y EMPLOYEES. La sentencia EXPLAIN muestra si se han elegido los índices apropiados para la sentencia SELECT.

```
 EXPLAIN SET QUERYNO = 100 FOR
  SELECT E.EMPNAME, S.SALES_AMOUNT
      FROM SALES S, EMPLOYEES E
            WHERE S.EMPNO = E.EMPNO
                  AND S.MONTH = ?

Index XSALES on SALES(MONTH)
Index XEMP on EMPLOYEES(EMPNO)

SELECT QUERY_NO, PLAN_NO, TABLE_NAME, INDEX_NAME,  SORT_TEMP
   FROM "DB2ePLANTABLE"

QUERY_NO  PLAN_NO  TABLE_NAME  INDEX_NAME  SORT_TEMP
--------------------------------------------------
   100     1        SALES        XSALES        -
   100     2        EMPLOYEE     XEMP          -
```

GRANT

La sentencia GRANT proporciona permiso para crear, consultar y manipular tablas cifradas dentro de la base de datos.

Para realizar la operación GRANT, el usuario debe estar conectado y autentificado. Si una base de datos no está cifrada, el primer usuario de la misma puede otorgarse a sí mismo la autentificación necesaria para realizar la operación GRANT. (Para obtener más información sobre cómo hacer esto, vea el ejemplo 1 más abajo).

Cuando se emiten sentencias GRANT sobre otros usuarios, los derechos de acceso de dichos usuarios entran en vigor en posteriores llamadas de conexión, a menos que la conexión modifique sus propios derechos de acceso.

Para cambiar su propia contraseña, debe realizar una operación GRANT sobre su propio ID de usuario.

Invocación

Esta sentencia se puede utilizar en un programa de aplicación utilizando las funciones de DB2 CLI o se puede emitir a través del DB2eCLP.

Sintaxis

Descripción

usuario_nuevo
Identifica el usuario al que se le otorgan los privilegios de cifrado.

contraseña_otorgante
Contraseña del usuario autentificado al que se están otorgando los privilegios de cifrado del usuario nuevo.

contraseña_nueva
Contraseña del usuario al que se le otorgan los privilegios de cifrado.

Reglas

- Los parámetros tanto de nombre de usuario como de contraseña tienen una longitud limitada de 254 bytes.
- Para caracteres de byte múltiple, internamente se utiliza la codificación UTF-8 para el almacenamiento. Por consiguiente, los nombres de usuario escritos utilizando juegos de caracteres internacionales tienen una longitud limitada.
- DB2 Everyplace requiere que el otorgante (es decir, el usuario conectado en ese momento) vuelva a entrar la contraseña de otorgante para poder otorgar privilegios al usuario nuevo. Estas restricciones aseguran que el otorgante esté presente físicamente en el dispositivo.
- Las contraseñas e idusuarios deben estar delimitados mediante comillas dobles.

Notas

- Si es usted un usuario existente, debe estar conectado y autentificado para cambiar su propia contraseña Sólo puede cambiar su propia contraseña.
- La sentencia GRANT no se puede utilizar con marcadores de parámetros ni con la función SQLPrepare().
- Intentar conseguir los privilegios de GRANT mientras se está conectado con un usuario no autorizado devuelve SQLSTATE 42502. Especificar una contraseña incorrecta con la sentencia GRANT ocasiona un SQLSTATE 42506.
- Mientras se ejecute GRANT en una transacción manual, las sentencias SELECT y DML se bloquearán hasta que la transacción en cuestión se confirme o se retrotraiga.

Ejemplo

Ejemplo 1: El primer usuario se otorga a sí mismo la autentificación necesaria para realizar la operación GRANT, sobre una base de datos que todavía no se ha cifrado:

```
GRANT ENCRYPT ON DATABASE TO "jsk" USING "foo" NEW "foo"
```

Ejemplo 2: Ahora se crea y autentifica el usuario ″jsk″ (del *Ejemplo 1* anterior), que posee la conexión. Para que ″jsk″ añada otro usuario:

```
GRANT ENCRYPT ON DATABASE TO "xin" USING "foo" NEW "bar"
```

Ejemplo 3: El usuario ″jsk″, conectado actualmente, cambia su propia contraseña:

```
GRANT ENCRYPT ON DATABASE TO "jsk" USING "foo" NEW "fie"
```

Ejemplo 4: El usuario ″jsk″, que sigue conectado actualmente, utiliza su nueva contraseña para añadir otro usuario:

```
GRANT ENCRYPT ON DATABASE TO "thf" USING "fie" NEW "fum"
```

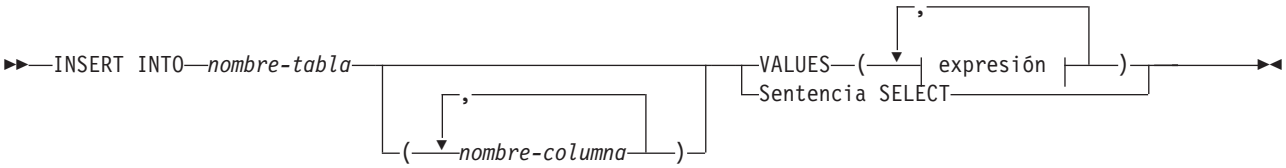
INSERT

La sentencia INSERT inserta una o más filas en una tabla utilizando los valores proporcionados.

Invocación

Esta sentencia se puede utilizar en un programa de aplicación utilizando las funciones de DB2 CLI o se puede emitir a través del DB2eCLP.

Sintaxis



expresión:

operador:

Descripción

INTO *nombre-tabla*

Designa la tabla de la operación de inserción. El nombre debe especificar una tabla existente, pero no una tabla de catálogo.

(*nombre-columna***,...)**

Especifica las columnas para las que se proporcionan valores de inserción. Cada nombre de columna debe ser un nombre no calificado que identifica una columna de la tabla. Una misma columna no puede estar identificada más de una vez.

La omisión de la lista de columnas equivale a una especificación implícita de una lista que designa, de izquierda a derecha, cada columna de la tabla.

VALUES

Presenta una fila de valores para insertar.

El número de valores para cada fila debe ser igual al número de nombres de la lista de columnas. El primer valor se inserta en la primera columna de la lista, el segundo valor en la segunda columna y así sucesivamente.

expresión

La expresión puede ser un literal, un registro especial, una función o una expresión compleja.

No se da soporte a las operaciones aritméticas sobre los tipos de datos CHAR, VARCHAR, BLOB(n), DATE, TIME y TIMESTAMP.

literal
Un literal puede ser un valor de cualquier tipo de datos soportado como INTEGER, SMALLINT, DECIMAL, CHAR(n), VARCHAR(n), BLOB(n), DATE, TIME o TIMESTAMP.

registro-especial
Se pueden utilizar los registros especiales CURRENT DATE, CURRENT TIME y CURRENT TIMESTAMP para generar la fecha, la hora y la indicación de hora actuales.

Sentencia SELECT
Especifica un conjunto de columnas nuevas en el formato de la tabla de resultados de una sentencia SELECT. Puede existir una, más de una o ninguna. Si la tabla de resultados está vacía, SQLCODE se establece en +100 y SQLSTATE se establece en '02000'. El objeto base de la sentencia SELECT no puede ser el objeto base de la sentencia INSERT.

Reglas

Valores predeterminados
Se inserta un valor predeterminado o nulo en las columnas que no aparecen en la lista de columnas. Las columnas que no aceptan valores predeterminados o nulos se deben incluir en la lista de columnas.

Longitud
Si el valor para insertar en una columna es un número, la columna debe ser una columna numérica que pueda representar la parte entera del número. Si el valor para insertar en una columna es una cadena de caracteres, la columna debe ser una columna de tipo carácter cuya longitud sea como mínimo la longitud de la cadena.

Asignación
Los valores de inserción se asignan a las columnas de acuerdo con las reglas de asignación descritas en la documentación de DB2 Versión 9.1.

Ejemplos

Ejemplo 1: Inserción en la tabla EMPLOYEE de un empleado cuyas especificaciones son las siguientes:
- Número de empleado (EMPNO): 002001
- Nombre (FIRSTNAME): John
- Apellido (LASTNAME): Harrison
- Número de departamento (DEPT): 600
- Número de teléfono (PHONENO): 4900
- Salario (SALARY): 50000
- Fecha de contrato (HIREDATE): 01/12/1989

```
INSERT INTO EMPLOYEE
 VALUES ('002001', 'John', 'Harrison', '600', '4900', 50000, '01/12/1989')
```

Ejemplo 2: Inserción en la tabla EMPLOYEE de un nuevo empleado cuyas especificaciones son las siguientes:
- Número de empleado (EMPNO): 003002
- Nombre (FIRSTNAME): Jim
- Apellido (LASTNAME): Gray

```
INSERT INTO EMPLOYEE (EMPNO, FIRSTNAME, LASTNAME)
  VALUES ('003002', 'Jim', 'Gray')
```

Ejemplo 3: Creación de una tabla EMP_ACT_COUNT. Cargar EMP_ACT_COUNT con las filas de la tabla EMP_ACT con un número de empleado (EMPNO) con el número de proyectos implicados.

```
CREATE TABLE EMP_ACT_COUNT
   ( EMPNO CHAR(6)  NOT NULL,
     COUNT          INTEGER)

INSERT INTO EMP_ACT_COUNT
    SELECT EMPNO, COUNT(*)
   FROM EMP_ACT
   GROUP BY EMPNO
```

Restricciones:

1. Los tipos de datos de columna de la sentencia SELECT deben ser idénticos a las definiciones de columna de la tabla de destino (exceptuando la capacidad para contener nulos).
2. No están permitidas las cláusulas ORDER BY y LIMIT.
3. No puede insertar valores en una columna ROWID de Oracle. Si intenta insertar valores en este tipo de columna, DB2 Everyplace devuelve el SQLSTATE 428C9.

LOCK TABLE

La sentencia LOCK TABLE permite a un sentencia adquirir explícitamente un bloqueo de tabla compartido o exclusivo sobre una tabla especificada. El bloqueo de la tabla durará hasta el final de la transacción actual. No es posible bloquear tablas del sistema con esta sentencia.

Invocación

Esta sentencia puede utilizarse en una aplicación utilizando las funciones de DB2 CLI o emitirse mediante DB2eCLP.

Sintaxis

Descripción

nombre-tabla

Especifica la tabla que se debe bloquear. El nombre puede tener hasta 128 bytes de longitud. El nombre debe identificar una tabla contenida en el catálogo.

Cuando un nombre de tabla contenga espacios en blanco o caracteres especiales, se deberán utilizar identificadores delimitados (con comillas dobles).

Los nombres de tabla pueden incluir caracteres DBCS (de doble byte).

Restricción: Los archivos de datos creados por el sistema que corresponden a tablas creadas y denominadas por nombres de usuario no distinguen entre caracteres en mayúsculas y minúsculas. Por ejemplo, el archivo de datos para una tabla denominada TB se denomina DSY_TB. El archivo de datos para una tabla denominada "tb" también es DSY_TB. Por consiguiente le recomendamos firmemente que, para asegurarse de la integridad de los datos, no denomine una tabla utilizando una serie de caracteres idéntica, a excepción de la consideración sobre mayúsculas y minúsculas, a la de un nombre de tabla existente.

Notas

- Esta sentencia no se puede utilizar para bloquear tablas del sistema. Cualquier intento que se realice producirá un SQLSTATE 42832.

- DB2 Everyplace proporciona un mecanismo de tiempo de espera excedido que las aplicaciones pueden utilizar para resolver puntos muertos. Si una aplicación no puede obtener un bloqueo dentro de un período de tiempo especificado, el motor de base de datos retrotraerá la transacción y devolverá el SQLSTATE 40001. El tiempo de espera predeterminado para los bloqueos es de 20 segundos.

Ejemplo

El código siguiente obtiene un bloqueo exclusivo sobre la tabla EMP.

```
LOCK TABLE EMP IN EXCLUSIVE MODE
```

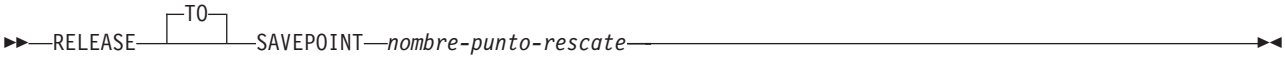
RELEASE SAVEPOINT

Utilice la sentencia RELEASE SAVEPOINT cuando desee dejar de mantener el punto de rescate especificado. Una vez ejecutada esta sentencia, ya no podrá hacer una retrotracción hasta ese punto de rescate.

Invocación

Esta sentencia se puede integrar en un programa de aplicación o ser emitida interactivamente. Es una sentencia ejecutable que se puede preparar dinámicamente.

Sintaxis



Descripción

nombre-punto-rescate

Especifica el punto de rescate que se debe liberar. DB2 Everyplace también libera los puntos de rescate que estén anidados dentro del punto de rescate especificado. Una vez ejecutada la sentencia, yo no podrá hacer una retrotracción hasta ese punto de rescate ni hasta ningún punto de rescate anidado dentro de aquél. Si el punto de rescate especificado no existe en el nivel de punto de rescate actual (vea la sección ″Reglas″ en la descripción de la sentencia SAVEPOINT), DB2 Everyplace devuelve el código SQLSTATE 3B001. El nombre de punto de rescate especificado no puede comenzar con 'SYS' (SQLSTATE 42939).

Notas

Una vez liberado un punto de rescate, puede reutilizar el nombre en otra sentencia SAVEPOINT, sin importar si se ha especificado o no la palabra clave UNIQUE en una sentencia SAVEPOINT anterior donde se especifique este mismo nombre de punto de rescate.

Ejemplo

Liberar un punto de rescate llamado SAVEPOINT1.

```
RELEASE SAVEPOINT SAVEPOINT1
```

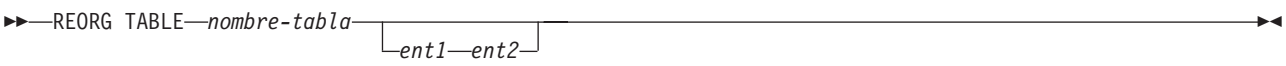
REORG TABLE

La sentencia REORG TABLE comprime los datos de la tabla especificada.

Invocación

Esta sentencia se puede utilizar en un programa de aplicación utilizando las funciones de DB2 CLI o se puede emitir mediante el DB2eCLP.

Sintaxis



Descripción

REORG TABLE *nombre-tabla*
Identifica la tabla que es objeto de la operación de reorganización. El nombre debe identificar una tabla existente.

ent1
Porcentaje mínimo opcional de bytes que es necesario recuperar.

ent2
Número mínimo de bytes que es necesario recuperar para que se ejecute la compresión de la tabla.

Reglas

- REORG TABLE sólo se puede ejecutar en modalidad de confirmación automática (autocommit=on o bien autocommit=true).
- Si se llama a REORG TABLE desde dentro de una transacción se producirá un SQLState 42887.
- Los valores opcionales *int1* y *int2* deben especificarse conjuntamente o no especificar ninguno de ellos.
- El valor opcional *int1* debe ser un número no negativo.
- El valor opcional *int1* debe estar comprendido entre 0 y 100.

Notas

- DB2 Everyplace puede invocar internamente una reorganización de tabla.
- El primer parámetro opcional es el porcentaje de bytes no utilizables que la tabla debe contener (por ejemplo, un 10 por ciento significa que ″como mínimo el 10 por ciento del espacio no es utilizable″). El segundo parámetro opcional es el número de bytes no utilizables que la tabla debe contener (por ejemplo, 1000 significará que ″como mínimo 1000 bytes deben ser espacio no utilizable″). Para que tenga lugar una reorganización real de la tabla, se deben cumplir ambos criterios.
- Si no se especifica ningún parámetro, DB2 Everyplace utilizará valores predeterminados para estas opciones. El porcentaje predeterminado es 30 y el número de bytes predeterminado es 6144. Así, ″reorg table t1″ es lo mismo que ″reorg table t1 30 6144″.
- Si la modalidad de reorganización se establece en habilitada, DB2 Everyplace reorganizará automáticamente una tabla. Si la reorganización está habilitada, después de ejecutar una sentencia DELETE o UPDATE se ejecuta una operación ″reorg table nombre_tabla 50 30270″ para la tabla de destino. Si la reorganización está habilitada, al final del proceso de una sentencia DROP TABLE se ejecuta una operación ″reorg table DB2eSYSTABLES 30 10240″ (también para DB2eSYSCOLUMNS y DB2eSYSRELS).
- En un programa C/C++, la modalidad de reorganización se establece utilizando la función SQLSetStmtAttr de CLI/ODBC con el atributo SQL_ATTR_REORG_MODE. En un programa Java, establece la modalidad de reorganización el método enableReorg de la interfaz DB2eStatement. El valor predeterminado consiste en que la reorganización esté habilitada.
- La reorganización de una tabla comprime el archivo de datos que contiene la tabla, reclamando físicamente el espacio no utilizable creado por supresiones y actualizaciones. Luego se actualizan los índices de la tabla de forma que apunten a la nueva dirección física de las filas.
- Las tablas base del catálogo del sistema de DB2 Everyplace se pueden reorganizar.
- Las reorganizaciones implícitas de las tablas del catálogo del sistema tras una sentencia DROP TABLE sólo se desencadenan cuando la modalidad de confirmación automática está activada.

- No se debe producir ninguna otra actividad en la base de datos mientras se ejecute una sentencia REORG TABLE. Si se ejecuta cualquier otra sentencia sobre la base de datos mientras se está realizando una reorganización de datos, se devolverá un SQLSTATE 57011.

Ejemplos

El mandato siguiente reorganiza la tabla VNNURSE utilizando los valores predeterminados.

```
REORG TABLE VNNURSE
```

REVOKE

La sentencia REVOKE permite que un usuario conectado y autentificado revoque los privilegios de cifrado de un usuario existente.

Invocación

Esta sentencia se puede utilizar en un programa de aplicación utilizando las funciones de DB2 CLI o se puede emitir a través del DB2eCLP.

Sintaxis

Descripción

usuario
Identifica el usuario cuyos privilegios de cifrado se revocan.

Reglas

- El parámetro de usuario debe ser un identificador delimitado. Tiene una longitud limitada a 254 bytes.
- Para caracteres de byte múltiple, internamente se utiliza la codificación UTF-8 para el almacenamiento. Por consiguiente, los nombres de usuario escritos utilizando juegos de caracteres internacionales tienen una longitud limitada.
- Si se eliminan todos los usuarios que tienen privilegios de cifrado, se puede seguir accediendo a las tablas cifradas durante la sesión actual. Una vez que finalice la sesión actual, las tablas cifradas dejarán de estar disponibles.

Notas

- Para revocar privilegios de un usuario existente, un usuario debe estar conectado y autentificado. Si es usted un usuario conectado y autentificado, puede revocar privilegios de cualquier usuario, incluido usted mismo.
- Cuando se emiten sentencias REVOKE sobre otros usuarios, los derechos de acceso de dichos usuarios entran en vigor en posteriores llamadas a CONNECT, a menos que la conexión modifique sus propios derechos de acceso.
- La sentencia REVOKE no se puede utilizar con marcadores de parámetros ni con la función SQLPrepare().
- Intentar conseguir los privilegios de REVOKE mientras se está conectado como usuario no autorizado devuelve SQLSTATE 42502. Intentar revocar (REVOKE) privilegios de un usuario inexistente da como resultado SQLSTATE 42501.
- Mientras se ejecute REVOKE en una transacción manual, las sentencias SELECT y DML se bloquearán hasta que la transacción en cuestión se confirme o se retrotraiga.
- Cuando se revoca un privilegio SELECT desde un usuario, la revocación entra en vigor en la siguientes sentencia SELECT.

Ejemplo

El usuario conectado y autentificado actualmente elimina los privilegios de cifrado del usuario ″jsk″:

```
REVOKE ENCRYPT ON DATABASE FROM "jsk"
```

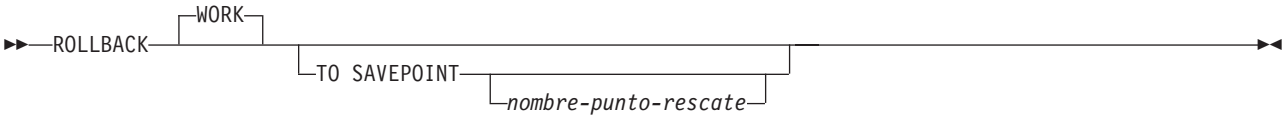
ROLLBACK

La sentencia ROLLBACK se utiliza para retrotraer los cambios de base de datos hechos dentro de una unidad de trabajo o punto de rescate.

Invocación

Esta sentencia se puede integrar en un programa de aplicación o ser emitida interactivamente. Es una sentencia ejecutable que se puede preparar dinámicamente.

Sintaxis



Descripción

Concluye la unidad de trabajo en la que se ejecuta la sentencia ROLLBACK y se inicia una nueva unidad de trabajo. Se retrotraen todos los cambios hechos en la base de datos durante la unidad de trabajo.

La creación de valores de secuencia e identidad no está bajo el control de la transacción. Los valores creados por la inserción de filas en una tabla que tiene una columna de identidad son independientes de la emisión de la sentencia ROLLBACK.

TO SAVEPOINT

Especifica que se debe realizar una retrotracción parcial (ROLLBACK TO SAVEPOINT). Si no existe ningún punto de rescate activo en el nivel de punto de rescate actual (vea la sección ″Reglas″ en la descripción de la sentencia SAVEPOINT), DB2 Everyplace devuelve un error y el código SQLSTATE 3B502. Después de una retrotracción satisfactoria, el punto de rescate sigue existiendo, pero los puntos de rescate anidados se liberan y dejan de existir. Se considera que los puntos de rescate anidados, si los hay, se han retrotraído y luego liberado como parte de la retrotracción hasta el punto de rescate actual. Si no se proporciona ningún nombre de punto de rescate, la retrotracción se realiza hasta el punto de rescate definido más recientemente dentro del nivel de punto de rescate actual.

Si se omite esta cláusula, la sentencia ROLLBACK retrotrae la transacción completa. Además, se liberan los puntos de rescate existentes en la transacción.

nombre-punto-rescate

Especifica el punto de rescate hacia el que revierte DB2 Everyplace durante la operación de retrotracción. DB2 Everyplace invierte todos los cambios que se realizaron en datos y esquemas desde que se estableció el punto de rescate. Después de una operación satisfactoria de retrotracción, el punto de rescate sigue existiendo.

Cuando utiliza el argumento *nombre-punto-rescate*, debe seguir las restricciones siguientes:

- El nombre especificado en *nombre-punto-rescate* no puede comenzar con 'SYS'. Si especifica un nombre de punto de rescate que comienza con 'SYS', DB2 Everyplace devuelve el SQLSTATE 42939.
- Si el nombre especificado en *nombre-punto-rescate* no existe, DB2 Everyplace devuelve un error y el SQLSTATE 3B001.

Notas

- La retrotracción de una unidad de trabajo hace que se liberen todos los bloqueos establecidos. Se cierran todos los cursores abiertos. Se liberan todos los localizadores de LOB.
- Si el programa termina de forma anómala, se realiza una retrotracción implícita de la unidad de trabajo.
- El efecto de una sentencia ROLLBACK TO SAVEPOINT en los cursores depende de las sentencias existentes dentro del punto de rescate:
  - Si el punto de rescate contiene DDL o DML del cual depende un cursor, el cursor se marca como no válido. Si el usuario intenta utilizar este cursor, DB2 Everyplace devuelve un error y el código SQLSTATE 57007.
  - Si el cursor se especifica en el punto de rescate, el cursor permanece abierto y se coloca delante de la siguiente fila lógica del conjunto de resultados.
  - En otro caso, el cursor permanece abierto y posicionado, y no es afectado por la sentencia ROLLBACK TO SAVEPOINT.
- Los nombres de sentencias preparadas dinámicamente siguen siendo válidos, aunque la sentencia puede ser preparada de nuevo implícitamente, como resultado de operaciones de DDL que se retrotraen dentro del punto de rescate.
- Después de una sentencia ROLLBACK TO SAVEPOINT, todos los bloqueos se conservan si el punto de rescate especificado en la sentencia se ha creado con la opción ON ROLLBACK RETAIN LOCKS. En otro caso, todos los bloqueos establecidos con posterioridad al punto de rescate se liberan.
- Todos los localizadores de LOB se conservan después de una operación ROLLBACK TO SAVEPOINT.

Ejemplo

Suprimir todas las alteraciones que se han realizado desde la última operación commit o rollback.

```
ROLLBACK WORK
```

SAVEPOINT

Utilice la sentencia SAVEPOINT para establecer un punto de rescate dentro de una transacción.

Invocación

Esta sentencia se puede integrar en un programa de aplicación o ser emitida interactivamente. Es una sentencia ejecutable que se puede preparar dinámicamente.

Sintaxis

Descripción

nombre-punto-rescate
Especifica el nombre de un punto de rescate. El nombre de punto de rescate especificado no puede comenzar con 'SYS' (SQLSTATE 42939). Si ya existe un punto de rescate con ese nombre y se definió como exclusivo (UNIQUE) dentro de este nivel de punto de rescate, se devuelve un error (SQLSTATE 3B501).

UNIQUE
Especifica que la aplicación no reutilizará este nombre de punto de rescate mientras el punto de rescate esté activo dentro del nivel actual de punto de rescate. Si ya existe el nombre de punto de rescate dentro de este nivel de punto de rescate, se devuelve un error (SQLSTATE 3B501).

ON ROLLBACK RETAIN CURSORS
Cuando se realiza una retrotracción hasta este punto de rescate, especifica el comportamiento del sistema con respecto a los cursores que se abrieron después de emitir la sentencia SAVEPOINT. Esta cláusula indica que, cuando sea posible, los cursores no se ven afectados por una operación ROLLBACK TO SAVEPOINT. Para las situaciones en las que los cursores son afectados por la retrotracción hasta el punto de rescate, vea la sección referente a "ROLLBACK". A menos que se especifique esta cláusula, todos los cursores abiertos se cerrarán después de una sentencia ROLLBACK TO SAVEPOINT donde se especifique este punto de rescate.

ON ROLLBACK RETAIN LOCKS
Cuando se realiza una retrotracción hasta este punto de rescate, especifica el comportamiento del sistema con respecto a los bloqueos que se establecieron después de definir el punto de rescate. Los bloqueos establecidos después de definir el punto de rescate no se supervisan, y no se retrotraen (liberan) cuando se retrotrae hasta el punto de rescate. Si no especifica esta cláusula, en el siguiente suceso de retrotracción DB2 Everyplace libera todos los bloqueos establecidos desde que se definió el punto de rescate.

Reglas

- Se inicia un nuevo nivel de punto de rescate cada vez que se inicia una nueva transacción. Un nivel de punto de rescate finaliza cuando finaliza la transacción correspondiente. Cada vez que finaliza un nivel de punto de rescate, se liberan todos los puntos de rescate contenidos en ese nivel.
- Son aplicables las reglas siguientes respecto a las acciones realizadas dentro de un nivel de punto de rescate:
  - Los puntos de rescate solamente se pueden especificar para acciones realizadas dentro del nivel de punto de rescate en el que están establecidos. No puede liberar, destruir ni retrotraer para un punto de rescate establecido fuera del nivel de punto de rescate actual.
  - Todos los puntos de rescate activos establecidos dentro del nivel de punto de rescate actual se liberan automáticamente cuando finaliza el nivel de punto de rescate.
  - La unicidad de un nombre de punto de rescate solamente se cumple dentro del nivel de punto de rescate actual. Los nombres de puntos de rescate que están activos en otros niveles de punto de rescate se pueden reutilizar en el nivel actual sin afectar a los puntos de rescate situados en otros niveles.

Notas

- Cuando no se especifica la cláusula UNIQUE, el nombre de punto de rescate puede ser reutilizado por otro punto de rescate dentro del nivel de punto de rescate. Si dentro del nivel de punto de rescate ya existe un punto de rescate con el mismo nombre, se destruye el punto de rescate existente y se crea uno nuevo con el mismo nombre en el punto actual del proceso. Se considera que el nuevo punto de rescate es el último punto de rescate establecido por la aplicación. Observe que la destrucción de un punto de rescate mediante la reutilización de su nombre por otro punto de rescate no libera los puntos de rescate establecidos con posterioridad al punto de rescate destruido. Estos puntos de rescate subsiguientes solamente se pueden liberar mediante la sentencia RELEASE SAVEPOINT, la cual libera el punto de rescate especificado y todos los establecidos posteriormente.
- Si se especifica la cláusula UNIQUE, el nombre de punto de rescate solamente se puede reutilizar después de haber liberado un punto de rescate existente que tenga el mismo nombre.

Ejemplo

Utilice este ejemplo para realizar una operación de retrotracción para puntos de rescate anidados. Primero, cree una tabla llamada DEPARTMENT. Inserte una fila antes de iniciar SAVEPOINT1; inserte otra fila e inicie SAVEPOINT2; luego inserte una tercera fila e inicie SAVEPOINT3:

```
CREATE TABLE DEPARTMENT (DEPTNO   CHAR(6),
                         DEPTNAME VARCHAR(20),
                         MGRNO    INTEGER)
```

```
INSERT INTO DEPARTMENT VALUES ('A20', 'MARKETING', 301)

SAVEPOINT SAVEPOINT1 ON ROLLBACK RETAIN CURSORS

INSERT INTO DEPARTMENT VALUES ('B30', 'FINANCE', 520)

SAVEPOINT SAVEPOINT2 ON ROLLBACK RETAIN CURSORS

INSERT INTO DEPARTMENT VALUES ('C40', 'IT SUPPORT', 430)

SAVEPOINT SAVEPOINT3 ON ROLLBACK RETAIN CURSORS

INSERT INTO DEPARTMENT VALUES ('R50', 'RESEARCH', 150)
```

En este momento, existe la tabla DEPARTMENT con las filas A20, B30, C40 y R50. Si ahora emite este mandato:

```
ROLLBACK TO SAVEPOINT SAVEPOINT3
```

la fila R50 ya no existe en la tabla DEPARTMENT. Si luego emite:

```
ROLLBACK TO SAVEPOINT SAVEPOINT1
```

la tabla DEPARTMENT sigue existiendo, pero las filas insertadas desde que se estableció SAVEPOINT1 (B30 y C40) ya no están en la tabla.

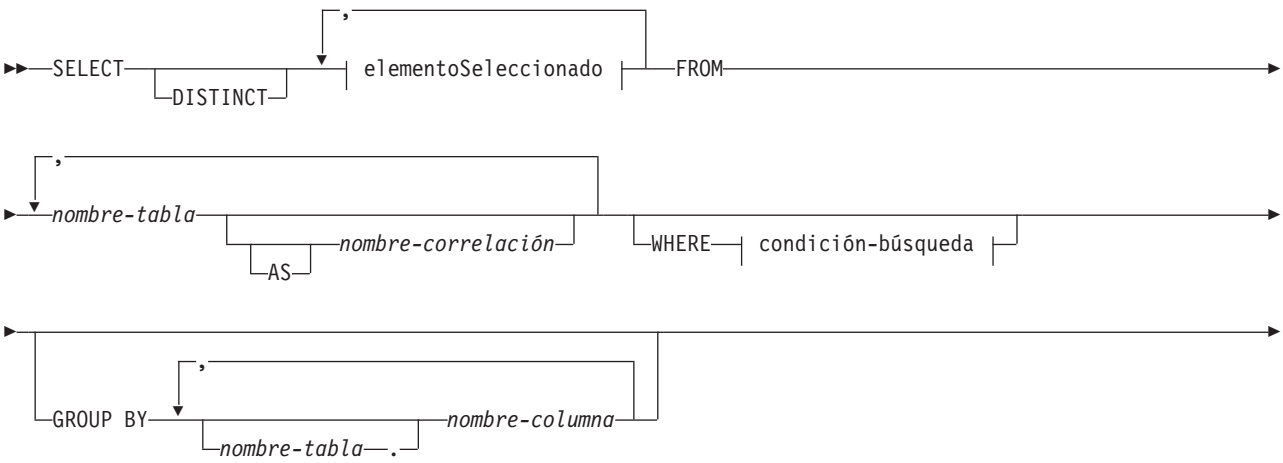
SELECT

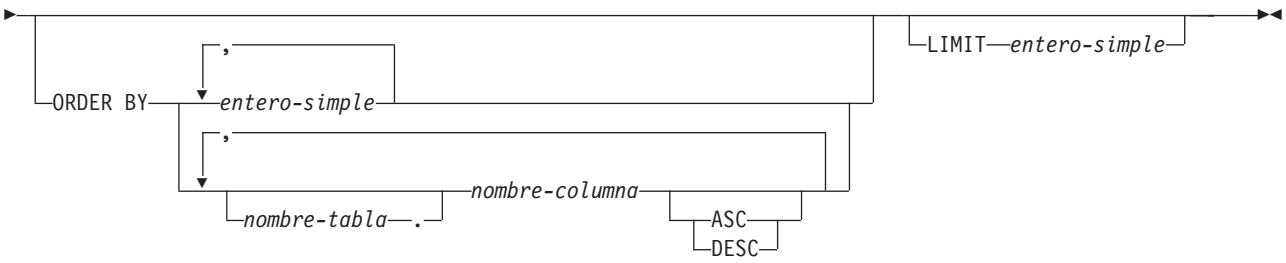
La sentencia SELECT es una forma de consulta.

Invocación

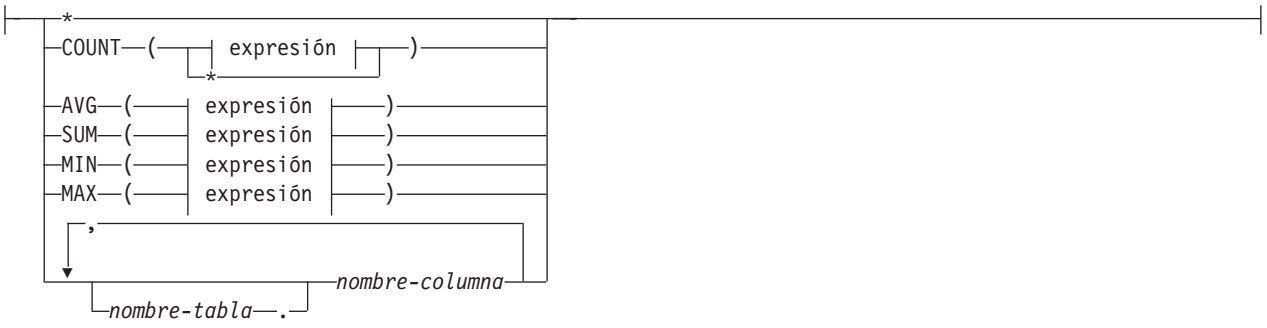
Esta sentencia se puede utilizar en un programa de aplicación utilizando las funciones de DB2 CLI o se puede emitir mediante la herramienta DB2eCLP.

Sintaxis

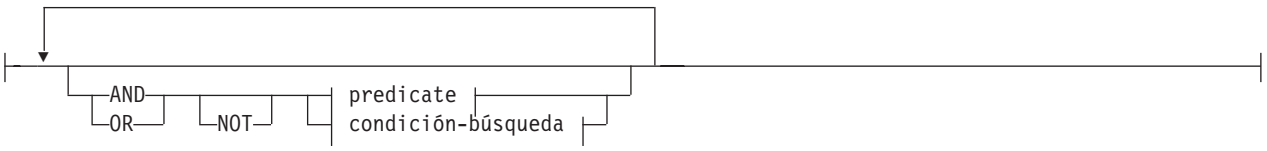




elementoSeleccionado:



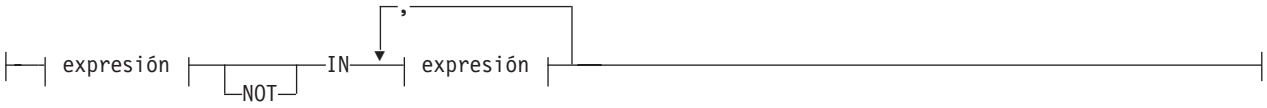
condición-búsqueda:



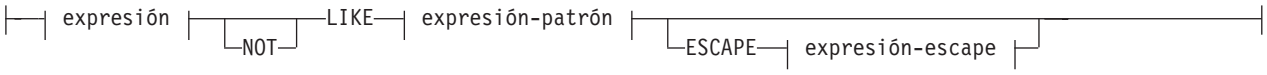
predicado:

predicado básico:

predicado IN:

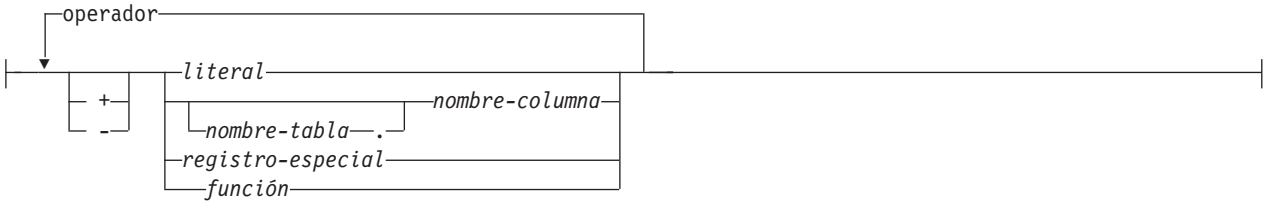


predicado LIKE:



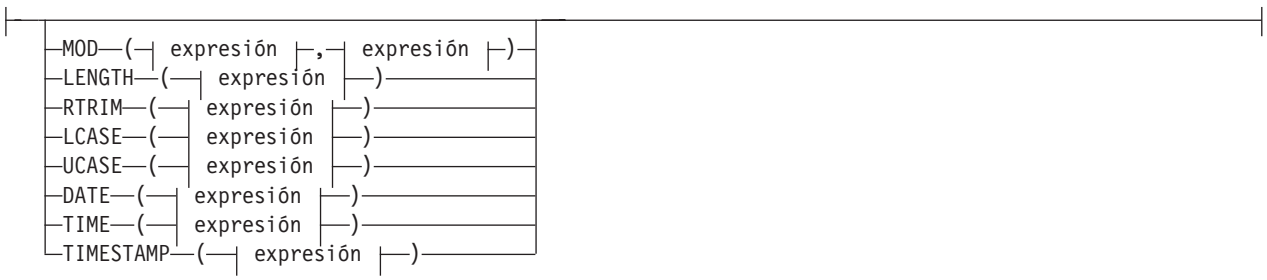
predicado NULL:

expresión:



operador:

función:



Notas:

1 Las expresiones BLOB solamente están permitidas en predicados NULL.

Descripción

elementoSeleccionado

* Especifica todas las columnas. Si se especifica un asterisco (*), debe ser el único elemento seleccionado.

COUNT(*)
La función COUNT obtiene el número de filas o valores de un conjunto de filas o valores. El argumento de COUNT(*) es un conjunto de filas. El resultado es el número de filas del conjunto. En la cuenta se incluyen también las filas que sólo contengan valores nulos.

expresión
La *expresión* puede ser un literal, nombre de columna, registro especial o función. Las funciones válidas son: COUNT, AVG, SUM, MIN, MAX, MOD, LENGTH, RTRIM, LCASE, UCASE, DATE, TIME y TIMESTAMP.

No se da soporte a las operaciones aritméticas sobre los tipos de datos CHAR, VARCHAR y BLOB(n). En DATE, TIME y TIMESTAMP sólo se permite la resta.

literal
Un *literal* puede ser un valor cuyo tipo de datos es INTEGER, SMALLINT, DECIMAL, CHAR(n), VARCHAR(n), BLOB(n), DATE, TIME y TIMESTAMP.

nombre-tabla
Designa la tabla donde reside la columna que es objeto de la consulta.

. Carácter que separa las dos partes que forman el identificador de columna, *nombre-tabla.nombre-columna*.

nombre-columna
Designa la columna que es objeto de la consulta.

COUNT(*expresión***)**
El argumento de COUNT(*expresión*) es un conjunto de filas. La función se aplica al conjunto de filas que se obtiene a partir de los valores argumento por eliminación de los valores nulos. El resultado es el número de valores no nulos del conjunto, incluidos los duplicados.

AVG(*expresión***)**
La función AVG(*expresión*) obtiene el valor promedio de los valores de *expresión*. Los valores argumento deben ser números y su suma debe estar dentro del rango del tipo de datos del resultado. La función se aplica al conjunto de valores que se obtiene a partir de los valores argumento por eliminación de los valores nulos. El resultado puede ser nulo.

SUM(*expresión***)**
La función SUM(*expresión*) obtiene la suma de los valores de *expresión*. Los valores argumento deben ser números y su suma debe estar dentro del rango del tipo de datos del resultado. La función se aplica al conjunto de valores que se obtiene a partir de los valores argumento por eliminación de los valores nulos.

MIN(*expresión***)**
La función MIN(*expresión*) obtiene el valor mínimo del conjunto de valores de *expresión*. Los valores argumento pueden ser cualquier tipo de datos interno excepto BLOB. La función se aplica al conjunto de valores que se obtiene a partir de los valores argumento por eliminación de los valores nulos.

MAX(*expresión***)**
La función MAX(*expresión*) obtiene el valor máximo del conjunto de valores de *expresión*. Los valores argumento pueden ser cualquier tipo de datos interno excepto BLOB. La función se aplica al conjunto de valores que se obtiene a partir de los valores argumento por eliminación de los valores nulos.

FROM
La cláusula FROM especifica una tabla de resultados intermedia.

Si se especifica una referencia de tabla, la tabla de resultados intermedia es simplemente el resultado de esa referencia de tabla. Si se especifica más de una tabla de referencia, la tabla de resultados intermedia consta de todas las combinaciones posibles de las filas de las referencias de tabla especificadas (el producto cartesiano). Cada fila del resultado es una fila de la primera referencia de

tabla concatenada con una fila de la segunda referencia de tabla, concatenada a su vez con una fila de la tercera y así sucesivamente. El número de filas del resultado es el producto del número de filas de todas las referencias de tabla individuales.

nombre-tabla
Cada *nombre-tabla* especificado como referencia de tabla debe identificar una tabla existente.

AS
Identifica la definición de tabla.

nombre-correlación
Cada *nombre-correlación* identifica el *nombre-tabla* inmediato anterior. Si se especifica un nombre de correlación para una tabla, cualquier referencia calificada a una columna de la tabla debe utilizar el nombre de correlación en lugar del nombre de tabla. Si se especifica dos veces un mismo *nombre de tabla*, al menos una de las especificaciones debe ir seguida por un *nombre de correlación*. El *nombre de correlación* se utiliza para calificar referencias a las columnas de la tabla. Como calificador, el nombre de correlación se puede utilizar para evitar ambigüedades o para establecer una referencia correlacionada. Puede también utilizarse simplemente como nombre abreviado para una tabla.

WHERE
Especifica una condición que selecciona filas. Se puede omitir la cláusula o especificar una condición de búsqueda. Si se omite la cláusula, se seleccionan todas las filas de la tabla.

condición-búsqueda
Una *condición-búsqueda* especifica una condición que es verdadera, falsa o desconocida para una fila determinada.

El resultado de una *condición-búsqueda* se obtiene aplicando los *operadores lógicos* especificados (AND, OR, NOT) al resultado de cada predicado especificado. Un predicado compara dos valores. Si no se especifican operadores lógicos, el resultado de la condición de búsqueda es el resultado del predicado especificado.

Las condiciones de búsqueda que están entre paréntesis se evalúan en primer lugar. Si no se especifica un orden de evaluación mediante el uso de paréntesis, NOT se aplica antes que AND y AND se aplica antes que OR. El orden en el que se evalúan los operadores con igual nivel de prioridad es indefinido, para permitir la optimización de las condiciones de búsqueda.

La *condición-búsqueda* se aplica a cada fila de la tabla y las filas seleccionadas son aquellas para las cuales el resultado de la *condición-búsqueda* es verdadero.

Cada *nombre-columna* especificado en la condición de búsqueda debe identificar una columna de la tabla.

NOT
Si se especifica NOT, se invierte el resultado del predicado.

expresión
La *expresión* puede ser un literal, nombre de columna, registro especial o función.

Operaciones aritméticas sobre CHAR, VARCHAR, BLOB(n). Sólo se da soporte a la resta para DATE, TIME y TIMESTAMP.

literal
Un *literal* puede ser un valor cuyo tipo de datos es INTEGER, SMALLINT, DECIMAL, CHAR(n), VARCHAR(n), BLOB(n), DATE, TIME y TIMESTAMP.

nombre-tabla
Designa la tabla donde reside la columna que es un operando del predicado.

. Carácter que separa las dos partes que forman el identificador de columna, *nombre-tabla.nombre-columna*.

nombre-columna
Identifica la columna que es un operando del predicado.

registro-especial
Identifica el registro especial que es un operando del predicado. Se pueden utilizar los registros especiales CURRENT DATE, CURRENT TIME y CURRENT TIMESTAMP para generar la fecha, la hora y la indicación de hora actuales.

función

MOD(*expresión, expresión***)**
La función MOD(*expresión, expresión*) obtiene el resto de dividir el primer argumento por el segundo. El resultado es negativo sólo si el primer argumento es negativo.

El primer y segundo argumento puede ser de tipo SMALLINT o INTEGER.

El resultado de la función es SMALLINT si ambos argumentos son SMALLINT; en otro caso, el resultado es INTEGER. El resultado puede ser nulo; si cualquiera de los argumentos es nulo, el resultado es el valor nulo.

LENGTH(*expresión***)**

La función LENGTH(*expresión*) obtiene la longitud de un valor.

El argumento puede ser una expresión que devuelve un valor de los siguientes tipos de datos incorporados:

- VARCHAR
- CHAR
- BLOB

El resultado de la función es un entero. Si el argumento puede ser nulo, el resultado también; si el argumento es nulo, el resultado es el valor nulo.

El resultado es la longitud del argumento. La longitud de una serie de longitud variable es la longitud real, no la longitud máxima.

La longitud de un BLOB es el número de bytes utilizados para representar el valor.

Considere una columna de tipo VARCHAR(50) llamada ADDRESS con un valor de '895 Don Mills Road'. `LENGTH(ADDRESS)` devuelve el valor 18.

RTRIM(*expresión***)**

La función RTRIM(*expresión*) elimina espacios en blanco del final de la serie.

El argumento puede ser de tipo de datos CHAR o VARCHAR.

El tipo de datos resultante de la función es siempre VARCHAR.

El parámetro de longitud del tipo de datos devuelto es el mismo que el parámetro de longitud del tipo de datos del argumento.

La longitud real del resultado para series de caracteres es la longitud de la expresión de la serie menos el número de bytes de caracteres en blanco eliminados. La longitud real del resultado para series de gráficos es la longitud (en el número de caracteres de doble byte) de la expresión de la serie menos el número de caracteres en blanco de doble byte eliminados. Si se eliminan todos los caracteres, el resultado es una serie de longitud variable y vacía (con longitud cero).

Si el argumento puede ser nulo, el resultado también; si el argumento es nulo, el resultado es el valor nulo.

Considere una columna de tipo CHAR(50) llamada NAME con un valor de 'Cliff '. `RTRIM(NAME)` devuelve 'Cliff'. `LENGTH(RTRIM(NAME))` devuelve 5.

LCASE/UCASE
Las funciones LCASE/UCASE utilizan como entrada una serie de caracteres y devuelven una

serie de caracteres en la que todos los caracteres están convertidos a minúsculas o mayúsculas, respectivamente. El argumento debe ser una expresión cuyo valor sea un tipo de datos CHAR o VARCHAR. El resultado de la función tiene el mismo tipo de datos que el argumento. Si el argumento puede ser nulo, el resultado también; si el argumento es nulo, el resultado es el valor nulo.

Los caracteres alfabéticos del argumento se convierten de acuerdo con el valor del entorno local LC_CTYPE que esté vigente para la sentencia. Por ejemplo, los caracteres a-z se convierten a A-Z, y los caracteres con signos diacríticos se convierten en su equivalente LCASE/UCASE, si existe. Los caracteres que no se pueden convertir, permanecen sin convertir en la serie de caracteres.

Importante: La Versión 8 de DB2 Everyplace utilizaba un algoritmo para determinar las formas en mayúscula y minúscula de los caracteres de Latin1. La Versión 9 de DB2 Everyplace utiliza funciones del sistema operativo para manejar caracteres que están fuera del juego de caracteres Latin1. Como resultado, el comportamiento de las aplicaciones que hacen uso de LCASE/UCASE en las plataformas Linux puede cambiar cuando estos programas se ejecutan en DB2 Everyplace Versión 9. Las consultas SELECT que hacen uso de LCASE/UCASE en cláusulas WHERE pueden devolver resultados diferentes. Si las consultas SELECT utilizan índices creados con LCASE/UCASE, puede ser necesario eliminar los índices y volver a crearlos después de la migración para obtener resultados correctos.

Ejemplo: Para asegurar que los caracteres contenidos en el valor de la columna JOB de la tabla EMPLOYEE se devuelvan en minúsculas, utilice esta sentencia de SQL:

```
SELECT LCASE(JOB)
     FROM EMPLOYEE
     WHERE EMPNO = '000020';
```

La tabla siguiente muestra los entornos locales predeterminados que las funciones LCASE/UCASE utilizan en cada plataforma.

Tabla 59. Entornos locales predeterminados utilizados por las funciones LCASE/UCASE

Sistema operativo	Entorno local
Linux	Entorno local predeterminado del sistema operativo. Se puede alterar mediante la función setlocale().
Windows	Entorno local predeterminado del sistema operativo. Se puede alterar mediante la función _tsetlocale().
Windows CE	Entorno local predeterminado del sistema operativo

Restricción: En los sistemas Windows, debe enlazar la aplicación con la misma biblioteca de ejecución C que DB2e.dll. En Visual Studio, seleccione **Configuración → Generación de código C/C++ → Utilizar biblioteca de ejecución → DLL multihebra**.

Los ejemplos siguientes muestran cómo cambiar el entorno local del sistema al alemán en los sistemas Windows y Linux.

Sistemas Windows

```
#include <locale.h>
_tsetlocale(LC_CTYPE, TEXT("German"));
```

Sistemas Linux

```
#include <locale.h>
setlocale(LC_CTYPE, "de_DE.UTF-8");
```

Puede también exportar la variable de entorno LC_CTYPE al entorno local de destino, por ejemplo:

```
export LC_CTYPE = "en_US.UTF-8"
```

Después de exportar el entorno local a una variable de entorno, puede definir ese entorno local en la aplicación utilizando este código:

```
#include <locale.h>
setlocale(LC_CTYPE, "");
```

DATE
La función DATE devuelve una fecha de una expresión que evalúa a un tipo de datos DATE, TIMESTAMP o de cadena de caracteres.

TIME
La función TIME devuelve una hora de una expresión que evalúa a un tipo de datos TIME, TIMESTAMP o de cadena de caracteres.

TIMESTAMP
La función TIMESTAMP devuelve una fecha de una expresión que evalúa a un tipo de datos TIMESTAMP o de cadena de caracteres.

registro-especial
Se pueden utilizar los registros especiales CURRENT DATE, CURRENT TIME y CURRENT TIMESTAMP para generar la fecha actual, la hora actual y la indicación de la hora actual.

predicado básico
Puede ser cualquiera de los operadores siguientes:

= Igual a.

<> No igual a.

< Menor que.

> Mayor que.

<= Menor o igual que.

>= Mayor o igual que.

LIKE

expresión-patrón
Coincide con una serie de caracteres. Utilice un símbolo de subrayado de SBCS (juego de caracteres de un solo byte) para representar un carácter SBCS individual. Utilice un símbolo de subrayado de DBCS (juego de caracteres de doble byte) para representar un carácter DBCS individual. Por ejemplo, la condición `WHERE PART_NUMBER LIKE '_0'` devuelve todos los números de pieza de 2 dígitos que terminan en 0 (20, 30 y 40, por ejemplo). Utilice un símbolo de porcentaje (del juego de caracteres SBCS o DBCS) para representar una cadena de caracteres SBCS o DBCS, o la ausencia de caracteres. Por ejemplo, la condición `WHERE DEPT_NUMBER LIKE '2%'` obtiene todos los números de departamento (DEPT_NUMBER) que comienzan con el número 2 (por ejemplo, 20, 27 ó 234).

Para buscar coincidencias con datos de serie que contengan caracteres de porcentaje o de subrayado, DB2 Everyplace soporta la cláusula ESCAPE en predicados LIKE.

expresión-escape

Este argumento opcional especifica un carácter que cambia el significado de los caracteres de porcentaje y de subrayado de una expresión, lo que permite al predicado LIKE buscar coincidencias con valores que contengan caracteres de

porcentaje y de subrayado. DB2 Everyplace no soporta un carácter de escape predeterminado. Utilice *expresión-escape* para indicar el carácter de escape.

El valor de *expresión-escape* se puede especificar mediante una constante o una variable del lenguaje principal con la restricción de que el resultado de la expresión debe ser un carácter que contenga exactamente un byte y no debe ser el propio carácter de porcentaje o de subrayado (SQLState 22019). Si *expresión-escape* no es válido, DB2 Everyplace devuelve el SQLState 22025.

Un carácter de subrayado, un signo de porcentaje o un carácter de escape pueden representar una ocurrencia literal de ellos mismos cuando *expresión-patrón* contenga un carácter de escape. Esto es verdadero cuando el carácter en cuestión va precedido de un número impar de caracteres de escape sucesivos. En caso contrario, no es verdadero.Los ejemplos siguientes muestran el efecto de ocurrencias sucesivas del carácter que, en este caso, es la barra inclinada invertida.

Tabla 60. Ejemplos de patrones de escape

Serie patrón	Patrón real
\%	Signo de porcentaje
\\%	Barra inclinada invertida seguida de cero o más caracteres arbitrarios
\\\%	Barra inclinada invertida seguida de un signo de porcentaje

NOT LIKE
Indica que al menos uno de los caracteres es diferente.

IN Encuentra las coincidencias de una colección de valores. El predicado IN compara un valor con un conjunto de valores.

Ejemplos:

SELECT lname, fname FROM emp WHERE state IN ('CA', 'AZ', 'OR');

SELECT c1 FROM t1 WHERE c1*5-6 IN (mod(c2,2)+5,c3+4/2);

NOT IN
No coincide con una colección de valores. El predicado NOT IN compara un valor con un colección conjunto de valores.

Ejemplos:

SELECT empid FROM emp WHERE city NOT IN ('San Jose', 'Morgan Hill', 'Santa Clara');

IS NULL
Contiene el valor nulo.

IS NOT NULL
No contiene el valor nulo.

AND
Si se especifica, el operador lógico AND se aplica al resultado de cada predicado especificado.

OR
Si se especifica, el operador lógico OR se aplica al resultado de cada predicado especificado.

GROUP BY
Especifica una tabla de resultados intermedia formada por un agrupamiento de las filas de R. R es el resultado de la cláusula anterior de la subselección.

ORDER BY
Especifica una ordenación de las filas de la tabla de resultados.

nombre-columna
Suele designar una columna de la tabla de resultados. En este caso, *nombre-columna* debe ser el nombre de una columna que aparece en la lista de selección.

entero-simple
Debe ser superior a 0 y no ser superior al número de columnas de la tabla de resultados. El entero *n* identifica la columna que ocupa la posición *n* en la tabla de resultados.

ASC
Utiliza los valores de la columna en orden ascendente.

DESC
Utiliza los valores de la columna en orden descendente.

LIMIT *entero-simple*
Limita el número de filas que se devuelven a la aplicación al primer número *n* de filas del conjunto de respuestas donde *n* es un entero. Debe ser superior a 0.

operador
Puede ser uno de los operadores siguientes

+ Sumar

- Restar

***** Multiplicar

/ Dividir por

|| La función (*expresión* || *expresión*) devuelve la concatenación de dos argumentos de serie. Los dos argumentos deben ser de tipos compatibles.

El resultado de la función es una serie. Su longitud es la suma de las longitudes de los dos argumentos. Si el argumento puede ser nulo, el resultado también; si el argumento es nulo, el resultado es el valor nulo.

Reglas

Las columnas con datos de tipo BLOB no se pueden utilizar en cláusulas GROUP BY, ORDER BY ni DISTINCT.

Notas

- Una sentencia SELECT DISTINCT puede contener un máximo de ocho columnas.
- Una cláusula GROUP BY puede contener un máximo de ocho columnas.
- Una cláusula ORDER BY puede contener un máximo de ocho columnas.
- Todas las columnas que se han especificado en la cláusula ORDER BY deben aparecer en la lista de selección. Por ejemplo, la consulta siguiente no es válida:

  ```
  SELECT EMPNO, FIRSTNAME FROM EMPLOYEE ORDER BY LASTNAME
  ```

  La consulta siguiente es válida:

  ```
  SELECT LASTNAME, EMPNO, FIRSTNAME FROM EMPLOYEE ORDER BY LASTNAME
  ```

Ejemplos

Ejemplo 1: Este ejemplo selecciona los empleados (EMPNO y LASTNAME) de la tabla EMPLOYEE que se contrataron después de la fecha 01/01/1980 y los ordena de acuerdo con su apellido (LASTNAME).

```
SELECT EMPNO, LASTNAME FROM EMPLOYEE
 WHERE HIREDATE > '01/01/1980'
 ORDER BY LASTNAME
```

Ejemplo 2: Este ejemplo calcula el salario promedio para cada departamento de la tabla EMPLOYEE.

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

Ejemplo 3: Calcular el volumen máximo de ventas de cada región de ventas y visualizar el resultado por región, en orden de mayor a menor volumen de ventas.

```
SELECT REGION, MAX(SALES_VOL) FROM SALES
 GROUP BY REGION ORDER BY 2 DESC
```

START TRANSACTION

La sentencia START TRANSACTION inicia una nueva unidad de trabajo.

Invocación

Esta sentencia se puede integrar en un programa de aplicación o ser emitida interactivamente. Es una sentencia ejecutable que se puede preparar dinámicamente.

Sintaxis

Descripción

Cuando el usuario emite una sentencia START TRANSACTION, DB2 Everyplace trata las sentencias subsiguientes como una nueva unidad de trabajo. Si esta sentencia se ejecuta dentro de una unidad de trabajo, se devuelve un error (SQLState 25501).

Reglas

La sentencia START TRANSACTION no se puede ejecutar dentro de una unidad de trabajo.

Notas

Esta sentencia inicia explícitamente una nueva unidad de trabajo para una conexión de base de datos donde se utiliza la modalidad de confirmación automática.

Ejemplo

Iniciar una nueva unidad de trabajo para una conexión de base de datos donde actualmente se utiliza la modalidad de confirmación automática.

```
START TRANSACTION
```

TIME

La función TIME devuelve una hora de un valor.

Invocación

Esta sentencia puede utilizarse en una aplicación utilizando las funciones de DB2 CLI o emitirse mediante DB2eCLP.

Sintaxis

Descripción

expresión
Especifica un valor. El argumento debe ser una hora, una indicación de la hora o una representación válida de la hora o la indicación de la hora en forma de serie.

Reglas

El resultado de la función es una hora: la parte del argumento correspondiente a la hora. Si el argumento puede ser nulo, el resultado también; si el argumento es nulo, el resultado es el valor nulo.

Ejemplo

Supongamos que la columna RECEIVED (del tipo de datos TIMESTAMP) tiene un valor de datos de '2003-12-25-17.12.30.000000'.

- TIME(RECEIVED) es '17:12:30' (del tipo de datos TIME).
- TIME('17.12.30') es '17:12:30' (del tipo de datos TIME).
- TIME('05:12 PM') es '17:12:00' (del tipo de datos TIME).

TIMESTAMP

La función TIMESTAMP devuelve una indicación de la hora de un valor.

Invocación

Esta sentencia puede utilizarse en una aplicación utilizando las funciones de DB2 CLI o emitirse mediante DB2eCLP.

Sintaxis

Descripción

expresión
Especifica un valor. El argumento debe ser una indicación de la hora o una representación válida de la indicación de la hora en forma de serie.

Reglas

Si el argumento puede ser nulo, el resultado también; si el argumento es nulo, el resultado es el valor nulo.

Notas

Cuando el método getString() de JDBC para DB2 Everyplace se llama sobre una columna TIMESTAMP devuelve una indicación de la hora en formato ISO "AAAA-MM-DD-HH.MM.SS.ZZZZZZ". Esto es distinto al formato de escape de la indicación horaria de JDBC "AAAA-MM-DD HH:MM:SS.FFFFFFFFF". JDBC de DB2 Versión 9.1 devuelve una indicación horaria con el formato "AAAA-MM-DD HH:MM:FFFFFF".

Ejemplo

```
Si HORA(IH2) <= HORA(IH1)

entonces HORA(RESULTADO) = HORA(IH1) - HORA(IH2).

Si HORA(IH2) > HORA(IH1)

entonces HORA(RESULTADO) = 24 + HORA(IH1) - HORA(IH2)

y se suma 1 a DÍA(IH2).
```

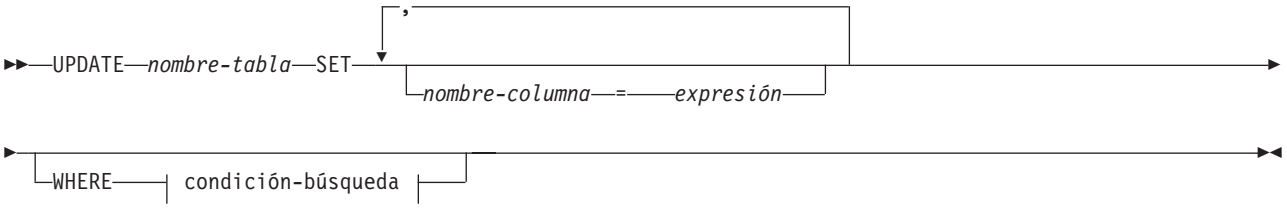
UPDATE

La sentencia UPDATE actualiza los valores de columnas especificadas en las filas de una tabla.

Invocación

Esta sentencia se puede utilizar en un programa de aplicación utilizando las funciones de DB2 CLI o se puede emitir mediante la herramienta DB2eCLP.

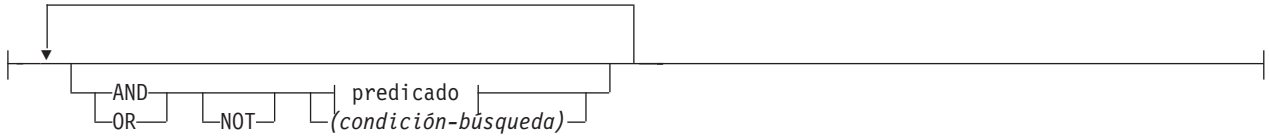
Sintaxis



expresión:

operador:

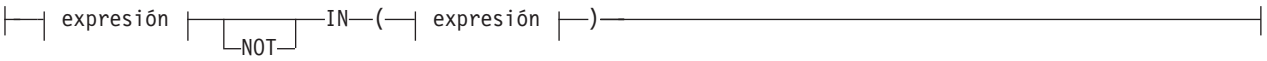
condición-búsqueda:



predicado:

predicado básico:

predicado IN:



predicado LIKE:

predicado NULL:

operador relacional:

Notas:

1 Las expresiones BLOB solamente están permitidas en predicados NULL.

Descripción

nombre-tabla
Es el nombre de la tabla para actualizar. El nombre debe identificar una tabla descrita en el catálogo, pero no una tabla de catálogo.

SET
Define la asignación de valores a nombres de columna.

nombre-columna
Designa una columna para actualizar. El *nombre-columna* debe identificar una columna de la tabla especificada. No se puede especificar una columna más de una vez (SQLSTATE 42701).

expresión
Una *expresión* puede ser un literal, nombre de columna o registro especial.

No se da soporte a las operaciones aritméticas sobre los tipos de datos BLOB(n), DATE, TIME y TIMESTAMP.

literal
Un *literal* puede ser un valor cuyo tipo de datos es INTEGER, SMALLINT, DECIMAL, CHAR(n), VARCHAR(n), BLOB(n), DATE, TIME o TIMESTAMP.

registro-especial
Se pueden utilizar los registros especiales CURRENT DATE, CURRENT TIME y CURRENT TIMESTAMP para generar la fecha, la hora y la indicación de hora actuales.

WHERE
Define una condición que indica las filas que deben actualizarse. Se puede omitir la cláusula o proporcionar una condición de búsqueda. Si se omite la cláusula, se actualizan todas las filas de la tabla.

condición-búsqueda
Una *condición-búsqueda* especifica una condición que es verdadera, falsa o desconocida para una fila determinada.

El resultado de una *condición-búsqueda* se obtiene aplicando los *operadores lógicos* especificados (AND, OR, NOT) al resultado de cada predicado especificado. Un predicado compara dos valores. Si no se especifican operadores lógicos, el resultado de la condición de búsqueda es el resultado del predicado especificado.

Las condiciones de búsqueda que están entre paréntesis se evalúan en primer lugar. Si no se especifica un orden de evaluación mediante el uso de paréntesis, NOT se aplica antes que AND y AND se aplica antes que OR. El orden en el que se evalúan los operadores con igual nivel de prioridad es indefinido, para permitir la optimización de las condiciones de búsqueda.

La *condición-búsqueda* se aplica cada fila de la tabla y las filas actualizadas son aquéllas para las cuales el resultado de la *condición-búsqueda* es verdadero.

Cada *nombre-columna* especificado en la condición de búsqueda debe identificar una columna de la tabla.

Puede utilizar las funciones CONCAT, MOD, LENGTH y RTRIM en la expresión predicado de la condición de búsqueda. Para obtener más información sobre la función MOD, consulte “SELECT” en la página 135.

NOT
Si se especifica NOT, se invierte el resultado del predicado.

operador relacional
Puede ser cualquiera de los operadores siguientes:

= Igual a.

<> No igual a.

< Menor que.

> Mayor que.

<= Menor o igual que.

>= Mayor o igual que.

LIKE Coincide con una serie de caracteres. Utilice un símbolo de subrayado de SBCS (juego de caracteres de un solo byte) para representar un carácter SBCS individual. Utilice un símbolo de subrayado de DBCS (juego de caracteres de doble byte) para representar un carácter DBCS individual. Por ejemplo, la condición `WHERE PART_NUMBER LIKE '_0'` devuelve todos los números de pieza de 2 dígitos que terminan en 0 (20, 30 y 40, por ejemplo). Utilice un símbolo de porcentaje (del juego de caracteres SBCS o DBCS) para representar una cadena de caracteres SBCS o DBCS, o la ausencia de caracteres. Por ejemplo, la condición `WHERE DEPT_NUMBER LIKE '2%'` obtiene todos los números de departamento (DEPT_NUMBER) que comienzan con el número 2 (por ejemplo, 20, 27 ó 234).

NOT LIKE
Indica que al menos uno de los caracteres es diferente.

IS NULL
Contiene el valor nulo.

IS NOT NULL
No contiene el valor nulo.

AND
Si se especifica, el operador lógico AND se aplica al resultado de cada predicado especificado.

OR
Si se especifica, el operador lógico OR se aplica al resultado de cada predicado especificado.

Reglas

- ***Asignación***: Los valores de actualización se asignan a columnas de acuerdo con las reglas de asignación descritas en la documentación de DB2 Versión 9.1.
- UPDATE no se aplica nunca a registros suprimidos lógicamente.
- No puede actualizar valores en una columna ROWID de Oracle. Si intenta actualizar valores en este tipo de columna, DB2 Everyplace devuelve el SQLSTATE 428C9.

Notas

- En la modalidad del sistema, el bit indicador se activa por omisión. Si está ejecutando su aplicación en la modalidad del sistema (SQL_DIRTYBIT_SET_BY_SYSTEM), no puede activar manualmente el bit indicador. Si intenta activar el bit indicador, se producirá un error.

Ejemplo

Este ejemplo cambia el número de teléfono (PHONENO) del número de empleado (EMPNO) '003002' para que sea '1234' en la tabla EMPLOYEE.

```
UPDATE EMPLOYEE
 SET PHONENO = '1234'
 WHERE EMPNO = '003002'
```

Tipos de datos soportados para procedimientos almacenados

DB2 Everyplace soporta la llamada a procedimientos almacenados de un servidor DB2 remoto mediante la interfaz JDBC. La aplicación cliente utiliza la sentencia CALL para ejecutar el procedimiento

almacenado remoto. La sentencia CALL da nombre al procedimiento al que se debe llamar y especifica sus parámetros. Se soportan los siguientes tipos: INTEGER, SMALLINT, DECIMAL, CHAR, VARCHAR, DATE, TIME, TIMESTAMP y BLOB.

Marcadores de parámetros soportados por DB2 Everyplace

DB2 Everyplace solamente puede utilizar marcadores de parámetros no tipificados, que se pueden utilizar en posiciones determinadas de una sentencia de SQL. Este tema muestra las restricciones existentes respecto a la utilización de marcadores de parámetros.

Un marcador de parámetro, que se representa mediante un símbolo de final de interrogación (?), es un espacio reservado en una sentencia de SQL cuyo valor se obtiene durante la ejecución de la sentencia. Una aplicación utiliza la función SQLBindParameter() para asociar marcadores de parámetros de enlace con variables de aplicación. Durante la ejecución de las funciones de DB2 CLI SQLExecute() y SQLExecDirect(), los valores de estas variables sustituyen a cada uno de los marcadores de parámetros respectivos. Durante el proceso puede tener lugar una conversión de datos.

Tabla 61. Restricciones sobre la utilización de los marcadores de parámetros

Ubicación de los marcadores de parámetros no tipificados	**Tipo de datos**
Expresión: Solo en una lista de selección	Error
Expresión: Ambos operandos de un operador aritmético	Error
Predicado: Operando de la parte izquierda de un predicado IN	Error
Predicado: Ambos operandos de un operador relacional	Error
Función: Operando de una función de agregación	Error

Tipos de datos predeterminados y simbólicos de SQL

La Tabla 62 describe los tipos de datos predeterminados y simbólicos para cada tipo de datos de SQL.

Tabla 62. Tipos de datos predeterminados y simbólicos de SQL

Tipo de datos de SQL	**Tipo de datos simbólico de SQL**	**Tipo de datos C simbólico predeterminado**
BLOB	SQL_BLOB	SQL_C_BINARY
CHAR	SQL_CHAR	SQL_C_CHAR
DATE	SQL_TYPE_DATE	SQL_C_TYPE_DATE
DECIMAL	SQL_DECIMAL	SQL_C_CHAR
INTEGER	SQL_INTEGER	SQL_C_LONG
SMALLINT	SQL_SMALLINT	SQL_C_SHORT
TIME	SQL_TYPE_TIME	SQL_C_TYPE_TIME
TIMESTAMP	SQL_TYPE_TIMESTAMP	SQL_C_TYPE_TIMESTAMP
VARCHAR	SQL_VARCHAR	SQL_C_CHAR

Compatibilidad entre tipos de datos para las operaciones de asignación y comparación

Las operaciones de asignación se realizan durante la ejecución de las sentencias INSERT y UPDATE. Las operaciones de comparación se realizan durante la ejecución de las sentencias que incluyen predicados.

Los tipos de datos de los operandos que intervienen deben ser compatibles, tal como se muestra en las tablas de la Tabla 63 a la Tabla 65.

Si la columna del tipo de datos contiene:

X Los tipos de datos de los operandos son compatibles.

en blanco
Los tipos de datos de los operandos no son compatibles.

Tabla 63. Compatibilidad entre tipos de datos, tabla 1

SQL, tipo de datos	INT	SMALLINT	DECIMAL	BLOB
INT	X	X	X	
VARCHAR				
BLOB				X
DECIMAL	X	X	X	
CHAR				
SMALLINT	X	X	X	
DATE				
TIME				
TIMESTAMP				

Tabla 64. Compatibilidad entre tipos de datos, tabla 2

SQL, tipo de datos	CHAR	VARCHAR
INT		
VARCHAR	X	X
BLOB		
DECIMAL		
CHAR	X	X
SMALLINT		
DATE	X	X
TIME	X	X
TIMESTAMP	X	X

Tabla 65. Compatibilidad entre tipos de datos, tabla 3

SQL, tipo de datos	DATE	TIME	TIMESTAMP
INT			
VARCHAR	X	X	X
BLOB			
DECIMAL			
CHAR	X	X	X
SMALLINT			
DATE	X		
TIME		X	
TIMESTAMP			X

Atributos de tipos de datos

Se muestra información sobre los atributos de tipo de datos siguientes:

- Precisión
- Escala
- Longitud
- Tamaño de visualización

Precisión

La precisión de una columna numérica o parámetro hace referencia al número máximo de dígitos que utiliza el tipo de datos de la columna o parámetro. La precisión de una columna no numérica o parámetro hace referencia generalmente a la longitud máxima o longitud definida de la columna o parámetro. La Tabla 66 define la precisión para cada tipo de datos SQL.

Tabla 66. Precisión

fSqlType	Precisión
SQL_CHAR SQL_VARCHAR	La longitud definida de la columna o parámetro. Por ejemplo, la precisión de una columna definida como CHAR(10) es 10.
SQL_DECIMAL	El número máximo de dígitos definido. Por ejemplo, la precisión de una columna definida como DECIMAL(10,3) es 10.
SQL_SMALLINT	5. El argumento *cbParamDef* de SQLBindParameter() se pasa por alto para este tipo de datos.
SQL_INTEGER	10. El argumento *cbParamDef* de SQLBindParameter() se pasa por alto para este tipo de datos.
SQL_BLOB	La longitud definida de la columna o parámetro. Por ejemplo, la precisión de una columna definida como BLOB(10) es 10.
SQL_DATE	10 (el número de caracteres en el formato aaaa-mm-dd). El argumento *cbParamDef* de SQLBindParameter() se pasa por alto para este tipo de datos.
SQL_TIME	8 (el número de caracteres en el formato hh:mm:ss). El argumento *cbParamDef* de SQLBindParameter() se pasa por alto para este tipo de datos.
SQL_TIMESTAMP	26 (El número de caracteres en el formato ″aaaa-mm-dd-hh.mm.ss.ffffff″ que utiliza el tipo de datos de TIMESTAMP.)

Escala

La escala de una columna numérica o parámetro hace referencia al número máximo de dígitos a la derecha de la coma decimal. La Tabla 67 en la página 154 define la escala para cada tipo de datos SQL.

Tabla 67. Escala

fSqlType	Escala
SQL_CHAR SQL_VARCHAR	No aplicable.
SQL_DECIMAL	El número de dígitos definido a la derecha de la coma decimal. Por ejemplo, la escala de una columna definida como DECIMAL(10, 3) es 3.
SQL_SMALLINT SQL_INTEGER	0
SQL_BLOB	No aplicable.
SQL_DATE SQL_TIME	No aplicable.
SQL_TIMESTAMP	6 (El número de dígitos a la derecha de la coma decimal en el formato "aaaa-mm-dd-hh.mm.ss.ffffff".)

Longitud

La longitud de una columna es el número máximo de bytes devueltos a la aplicación en el momento en que los datos se transfieren a su tipo de datos C predeterminado. Para los datos de tipo carácter, la longitud no incluye el byte de finalización de nulo. Observe que la longitud de una columna puede ser diferente del número de bytes necesarios para almacenar los datos en la fuente de datos.

La Tabla 68 define la longitud para cada tipo de datos SQL.

Tabla 68. Longitud

fSqlType	Longitud
SQL_CHAR SQL_VARCHAR	La longitud definida de la columna. Por ejemplo, la longitud de una columna definida como CHAR(10) es 10.
SQL_DECIMAL	El número máximo de dígitos más dos. Puesto que estos tipos de datos se devuelven en forma de series de caracteres, se necesitan caracteres para los dígitos, un signo y una coma decimal. Por ejemplo, la longitud de una columna definida como DECIMAL(10,3) es 12.
SQL_SMALLINT	2 (dos bytes).
SQL_INTEGER	4 (cuatro bytes).
SQL_BLOB	La longitud definida de la columna. Por ejemplo, la longitud de una columna definida como BLOB(10) es 10.
SQL_DATE SQL_TIME	6 (el tamaño de la estructura DATE_STRUCT o TIME_STRUCT).
SQL_TIMESTAMP	16 (el tamaño de la estructura TIMESTAMP_STRUCT).

Tamaño de visualización

El tamaño de visualización de una columna es el número máximo de bytes necesarios para visualizar datos en formato de caracteres. La Tabla 69 en la página 155 define el tamaño de visualización para cada tipo de datos SQL.

Tabla 69. Tamaño de visualización

fSqlType	Tamaño de visualización
SQL_CHAR SQL_VARCHAR	La longitud definida de la columna. Por ejemplo, el tamaño de visualización de una columna definida como CHAR(10) es 10.
SQL_DECIMAL	La precisión de la columna más dos (un signo, dígitos de precisión y una coma decimal). Por ejemplo, el tamaño de visualización de una columna definida como DECIMAL(10,3) es 12.
SQL_SMALLINT	6 (un signo y 5 dígitos).
SQL_INTEGER	11 (un signo y 10 dígitos).
SQL_BLOB	La longitud definida de la columna por 2 (cada byte binario está representado por un número hexadecimal de 2 dígitos). Por ejemplo, el tamaño de visualización de una columna definida como BLOB(10) es 20.
SQL_DATE	10 (una fecha en el formato aaaa-mm-dd).
SQL_TIME	8 (una hora en el formato hh:mm:ss).
SQL_TIMESTAMP	26 (una indicación de fecha y hora en el formato aaaa-mm-dd-hh.mm.ss.ffffff).

Reglas de resta para DATE, TIME y TIMESTAMP

La única operación aritmética que se puede realizar en valores de fecha y hora es la resta. El operador de resta sólo se puede utilizar con valores de fecha y hora cuando ambos operandos son fechas, ambos operandos son horas o ambos operandos son indicaciones de la hora.

DATE

El resultado de restar una fecha (FECHA2) a otra (FECHA1) es una duración entre fechas que especifica el número de años, meses y días entre las dos fechas. El tipo de datos del resultado es DECIMAL(8,0). Por ejemplo, el resultado de FECHA('3/15/2000') – FECHA('12/31/1999') es 00000215 (una duración de 0 años, 2 meses y 15 días).

Si FECHA1 es superior o igual a FECHA2, FECHA2 se restará de FECHA1. Por el contrario, si FECHA1 es inferior a FECHA2, FECHA1 se restará de FECHA2 y el signo del resultado será negativo. La siguiente descripción de procedimiento aclara los pasos necesarios para obtener el resultado de la operación = FECHA1 - FECHA2.

```
 Si DÍA(FECHA2) <= DÍA(FECHA1)

entonces DÍA(RESULTADO) = DÍA(FECHA1) - DÍA(FECHA2).

Si DÍA(FECHA2) > DÍA(FECHA1)

entonces DÍA(RESULTADO) = N + DÍA(FECHA1) - DÍA(FECHA2)

donde N = el último día de MES(FECHA2).

entonces se suma 1 a MES(FECHA2).

Si MES(FECHA2) <= MES(FECHA1)

entonces MES(RESULTADO) = MES(FECHA1) - MES(FECHA2).

Si MES(FECHA2) > MES(FECHA1)
```

```
entonces MES(RESULTADO) = 12 + MES(FECHA1) - MES(FECHA2).

entonces se suma 1 a AÑO(FECHA2).

AÑO(RESULTADO) = AÑO(FECHA1) - AÑO(FECHA2).
```

TIME

El resultado de restar una hora (HORA2) a otra (HORA1) es una duración entre horas que especifica el número de horas, minutos y segundos entre las dos horas. El tipo de datos del resultado es DECIMAL(6,0). Por ejemplo, el resultado de HORA('11:02:26') – HORA('00:32:56') es 102930 (una duración de 10 horas, 29 minutos y 30 segundos).

Si HORA1 es superior o igual a HORA2, HORA2 se restará de HORA1. Por el contrario, si HORA1 es inferior a HORA2, HORA1 se restará de HORA2 y el signo del resultado será negativo. La siguiente descripción de procedimiento aclara los pasos necesarios para obtener el resultado de la operación = HORA1 - HORA2.

```
Si SEGUNDO(HORA2) <= SEGUNDO(HORA1)

entonces SEGUNDO(RESULTADO) = SEGUNDO(HORA1) - SEGUNDO(HORA2).

Si SEGUNDO(HORA2) > SEGUNDO(HORA1)

entonces SEGUNDO(RESULTADO) = 60 + SEGUNDO(HORA1) - SEGUNDO(HORA2).

entonces se suma 1 a MINUTO(HORA2).

Si MINUTO(HORA2) <= MINUTO(HORA1)

entonces MINUTO(RESULTADO) = MINUTO(HORA1) - MINUTO(HORA2).

Si MINUTO(HORA2) > MINUTO(HORA1)

entonces MINUTO(RESULTADO) = 60 + MINUTO(HORA1) - MINUTO(HORA2).

entonces se suma 1 a HORA(HORA2).

HORA(RESULTADO) = HORA(HORA1) - HORA(HORA2).
```

TIMESTAMP

El resultado de restar una indicación de la hora (IH2) a otra (IH1) es una duración entre indicaciones de la hora que especifica el número de años, meses, días, horas, minutos, segundos y microsegundos entre las dos indicaciones de la hora. El tipo de datos del resultado es DECIMAL(20,6).

Si IH1 es superior o igual a IH2, IH2 se restará de IH1. Por el contrario, si IH1 es inferior a IH2, IH1 se restará de IH2 y el signo del resultado será negativo. La siguiente descripción de procedimiento aclara los pasos necesarios para obtener el resultado de la operación = IH1 - IH2:

```
Si MICROSEGUNDO(IH2) <= MICROSEGUNDO(IH1)

entonces MICROSEGUNDO(RESULTADO) = MICROSEGUNDO(IH1) - MICROSEGUNDO(IH2).

Si MICROSEGUNDO(IH2) > MICROSEGUNDO(IH1)

entonces MICROSEGUNDO(RESULTADO) = 1000000 + MICROSEGUNDO(IH1) - MICROSEGUNDO(IH2)

y se suma 1 a SEGUNDO(IH2).
```

Los segundos y los minutos que forman parte de las indicaciones de la hora se restan del modo especificado en las reglas para restar horas. La parte correspondiente a la fecha de las indicaciones de la hora se resta del modo especificado en las reglas para restar fechas.

Mensajes de SQLState en DB2 Everyplace

Este tema describe los valores de retorno de SQLstate para las especificaciones de la API.

Mensajes de SQLState notificados por JDBC

La Tabla 70 describe los mensajes que solamente son devueltos por JDBC. El controlador JDBC también puede devolver los mensajes listados en ″Mensajes de SQLState notificados por SQL″ y ″Mensajes de SQLState notificados por CLI″.

Tabla 70. Mensajes de SQLState notificados por JDBC

SQLSTATE	Descripción	Explicación
0100C	Se ha devuelto uno o más conjuntos de resultados ad hoc.	DB2 Everyplace no da soporte a ResultSet.CONCUR_UPDATABLE para la modalidad de simultaneidad de un objeto ResultSet. En su lugar se utiliza ResultSet.CONCUR_READ_ONLY.
0641E	Hay una sentencia SELECT en el proceso por lotes.	No se admite una sentencia SELECT en el proceso por lotes.
0643E	No hay ninguna sentencia en el proceso por lotes.	El proceso por lotes no tiene ninguna sentencia.
22005	Error de asignación.	Un tipo de parámetro es incompatible con el tipo de datos destino.
22011	Se ha producido un error de subserie de caracteres.	Posición ordinal no válida para el primer byte del valor BLOB que ha de extraerse.
58030	Error de E/S.	Se ha producido un error de E/S.
HY001	Error de asignación de memoria.	DB2 Everyplace no puede asignar memoria que es necesaria para permitir la ejecución o finalización de la función.
HY009	Valor no válido de argumento.	El valor de argumento para start no era mayor que 0.
HY013	Error inesperado de gestión de la memoria.	DB2 Everyplace no puede acceder a memoria que es necesaria para permitir la ejecución o finalización de la función.
HY090	Longitud no válida de la serie de caracteres o del almacenamiento intermedio.	El patrón de búsqueda es nulo.
HY100	Tipo de opción de exclusividad fuera de rango.	Se ha especificado un valor *Exclusivo* no válido para el parámetro de entrada.
S1010	Error de secuencia de función.	El método get `CallableStatement` se llama sin llamar en primer lugar a `registerOutParameter`.
XJ010	No se puede establecer el punto de rescate.	No se puede establecer un punto de rescate porque los puntos de rescate no se pueden utilizar en la modalidad de confirmación automática.
XJ011	Nombre no válido de punto de rescate.	El nombre del punto de rescate no puede ser nulo.
XJ013	No existe ID para los puntos de rescate especificados.	No existe información de ID para los puntos de rescate especificados.
XJ014	No existe nombre para puntos de rescate sin nombre.	No existe información de nombre disponible para los puntos de rescate sin nombre.
XJ081	Se ha pasado un valor de parámetro no válido a un método de la interfaz JDBC.	Uno de los parámetros que se pasó a un método de JDBC tiene un valor no válido.

Mensajes de SQLSTATE notificados por SQL

La Tabla 71 lista todos los SQLSTATE de las sentencias de SQL notificados por el motor SQL de DB2 Everyplace.

Tabla 71. Mensajes de SQLSTATE notificados por SQL

SQLSTATE	Descripción	Explicación
01000	Aviso.	Mensaje informativo. (La función devuelve SQL_SUCCESS_WITH_INFO).
01004	Valor truncado.	El valor fue truncado por una función de conversión de tipos de datos o función de ajuste del sistema.
01550	No se creó el índice.	No se creó el índice porque ya existe un índice con la descripción especificada.
02000	No se encontró ninguna fila.	No se encontró ninguna fila durante la ejecución de una sentencia FETCH, DELETE o UPDATE.
07001	Número incorrecto de parámetros.	No se ha enlazado un marcador de parámetro.
07005	Parámetro no válido.	El nombre de sentencia del cursor designa una sentencia preparada que no se puede asociar con un cursor.
07006	Variable no válida.	No se puede utilizar una variable de lenguaje principal de entrada debido a su tipo de datos.
08002	La conexión ya existe.	Ya existe una conexión.
22001	Es necesario truncar el valor.	Es necesario truncar el valor mediante una función de conversión de tipos de datos o función de ajuste del sistema.
22002	No se ha proporcionado ningún indicador de nulos.	No se puede asignar un valor NULL debido a la falta de espacio de almacenamiento.
22003	Valor numérico fuera de rango.	Un valor numérico no está dentro del rango de su columna destino.
22007	Formato no válido de fecha y hora.	La serie de caracteres utilizada para representar un valor de fecha y hora tiene una sintaxis incorrecta.
22008	Valor de fecha y hora fuera de rango.	La serie de caracteres utilizada para representar un valor de fecha y hora está fuera de rango.
22012	División por cero.	Se ha intentado una operación de división por cero.
22019	Carácter de escape no válido en predicado LIKE.	El predicado LIKE contiene un carácter de escape no válido.
22025	Secuencia de escape no válida.	La serie de caracteres del predicado LIKE contiene una secuencia de escape no válida.
22504	Carácter MBCS fragmentado.	Los datos contienen un carácter de múltiples bytes mal formado.
23502	Valor nulo no permitido.	No está permitida la asignación de un valor nulo a una columna definida como NOT NULL.
23505	Los valores no son exclusivos.	La operación no era válida debido a que produciría claves duplicadas.
23513	Valor no válido.	La fila resultante de la sentencia INSERT o UPDATE no se ajusta a la definición de la restricción de comprobación.
23515	Se ha especificado más de una cláusula de clave primaria.	Se ha especificado más de una cláusula de clave primaria.
23522	Se ha agotado el rango de valores para una columna de identidad o secuencia.	Se ha agotado el rango de valores para una columna de identidad o secuencia.

Tabla 71. Mensajes de SQLSTATE notificados por SQL (continuación)

SQLSTATE	Descripción	Explicación
24000	Estado no válido del cursor.	*StatementHandle* estaba en un estado ejecutado, pero no había ningún conjunto de resultados asociado al *StatementHandle*.
24501	Cursor no abierto.	No es válida una operación FETCH debido a que no se ha generado ningún conjunto de resultados.
24504	Estado no válido del cursor.	El cursor identificado en la sentencia UPDATE, DELETE, SET o GET no está situado en una fila.
25501	Sentencia no permitida en el contexto.	La sentencia solamente se permite como primera sentencia de una unidad de trabajo.
3B001	No se puede establecer el punto de rescate.	No se puede establecer un punto de rescate porque los puntos de rescate no se pueden utilizar en la modalidad de confirmación automática.
3B002	Se ha alcanzado el número máximo de puntos de rescate.	Se ha alcanzado el número máximo de puntos de rescate.
3B501	El punto de rescate ya existe.	Ya existe un punto de rescate con el mismo nombre, y este nombre no se puede reutilizar.
3B502	El punto de rescate no existe.	No existe un punto de rescate con el nombre especificado.
34000	El nombre de cursor no es válido.	El nombre de cursor no es válido.
42501	El ID de autorización no tiene permitido realizar la operación especificada sobre el objeto identificado.	El usuario actual está intentando eliminar un privilegio de un usuario que no existe.
42502	El ID de autorización no tiene permitido realizar la operación especificada.	El usuario actual no tiene una conexión autentificada. Cuando una aplicación (que no tiene la biblioteca de cifrado ni la CryptoPlugin.dll) ejecute un cifrado relacionado con mandatos de SQL (GRANT, REVOKE y CREATE TABLE), se devolverá un error ″42502″. Esto es así para impedir que las aplicaciones caigan.
42505	Se ha producido un error de autorización de conexión.	Un usuario registrado intenta conectar y no se le puede autentificar.
42506	Error de autorización de propietario.	No se ha podido autentificar el usuario conectado. (Contraseña incorrecta.)
42510	Algoritmo de cifrado no válido.	El algoritmo de cifrado especificado para una tabla de usuario cifrada no es válido. Los valores válidos son DES y DES3.
42601	Error de sintaxis.	Se ha detectado un error de sintaxis en la sentencia de SQL.
42603	Una constante de tipo carácter no tiene un delimitador final.	Una constante de tipo carácter o identificador delimitado no tiene un delimitador final.
42604	Se ha detectado una constante numérica o de tipo carácter que no es válida.	Una constante numérica o de tipo carácter no es válida.
42606	Se ha detectado una constante hexadecimal no válida.	Una constante hexadecimal no es válida.
42610	Utilización no válida de un marcador de parámetro.	La sentencia contiene un marcador de parámetro que no es válido. Vea la Tabla 5 en la página 17 para conocer la utilización válida de los marcadores de parámetros.
42611	Especificación de longitud no válida.	Una especificación de longitud excede el límite.
42614	Una palabra clave duplicada no es válida.	Una palabra clave duplicada no es válida.

Tabla 71. Mensajes de SQLSTATE notificados por SQL (continuación)

SQLSTATE	Descripción	Explicación
42621	La restricción de comprobación no es válida.	La restricción de comprobación no es válida.
42622	Nombre demasiado largo.	El nombre de un identificador es demasiado largo.
42623	No se puede especificar una cláusula DEFAULT.	No se puede especificar una cláusula DEFAULT.
42702	Referencia ambigua a un nombre de columna.	Existe más de una posible columna referenciada.
42703	Nombre de columna no definido.	Un nombre de columna no está en las tablas referenciadas.
42704	Objeto no definido.	La tabla no existe.
42710	El objeto designado ya existe.	Ya existe una tabla con el mismo nombre.
42711	Nombre de columna duplicado.	Un mismo nombre de columna está especificado más de una vez.
42802	El número de valores no coincide con el número de columnas.	El número de valores asignados no es el mismo que el número de columnas especificadas o implícitas.
42803	Una referencia de columna contenida en la lista de selección no está especificada en la cláusula GROUP BY.	La lista de selección contiene un nombre de columna y una función de agregación, pero no existe ninguna cláusula GROUP BY.
42815	El tipo de datos, longitud, escala, valor o CCSID no es válido.	El tipo de datos, longitud, escala, valor o CCSID no es válido.
42818	Tipos de datos incompatibles de los operandos.	Los tipos de datos de los operandos de una operación no son compatibles.
42820	Valor literal fuera de rango.	El valor numérico especificado no está dentro del rango aceptable.
42821	Tipos de datos incompatibles.	Un valor no es compatible con el tipo de datos de una columna destino.
42822	Elemento no válido de ORDER BY.	El elemento de ORDER BY no está en la lista de selección.
42824	Operando no válido de LIKE.	Un operando de LIKE no es una serie de caracteres o el primer operando no es una columna.
42829	FOR UPDATE OF no es válido.	FOR UPDATE OF no es válido, pues la tabla de resultados designada por el cursor no se puede modificar.
42830	La clave foránea no se ajusta a la descripción de la clave padre.	La clave foránea no se ajusta a la descripción de la clave padre.
42831	La clave primaria tiene columnas que pueden contener valores nulos.	Las columnas especificadas en la clave primaria no pueden contener nulos.
42832	Acceso no autorizado a objetos del sistema.	La operación no está permitida para objetos del sistema.
42837	No se puede alterar la columna.	La columna no se puede alterar, porque sus atributos no son compatibles con los atributos actuales de la columna.
42884	Nombre de función no conocido.	No se ha encontrado ninguna función o procedimiento con el nombre especificado y argumentos compatibles.
42887	Función no admitida.	La función no se puede utilizar en el release actual.
42894	El valor predeterminado (DEFAULT) no es válido.	El valor predeterminado (DEFAULT) no es válido.
428C1	Sólo se puede especificar una columna ROWID para una tabla.	Sólo se puede especificar una columna ROWID para una tabla.

Tabla 71. Mensajes de SQLSTATE notificados por SQL (continuación)

SQLSTATE	Descripción	Explicación
428C9	No se puede especificar una columna ROWID como columna de destino de una operación INSERT o UPDATE.	No se puede especificar una columna ROWID como columna de destino de una operación INSERT o UPDATE.
42902	Referencia duplicada a una tabla de objetos.	La tabla de objetos de la sentencia INSERT también se identifica en una cláusula FROM.
42903	Existe una referencia no válida en una cláusula WHERE o SET.	Una cláusula WHERE o SET contiene una referencia como, por ejemplo, una función de columna, que no es válida.
42962	No se puede utilizar como clave una columna de tipo LOB.	No se puede utilizar como clave primaria una columna de tipo LOB.
54001	Sentencia demasiado larga.	La sentencia de la consulta es demasiado larga.
54002	Una constante de tipo carácter es demasiado larga.	Una constante de tipo carácter es demasiado larga.
54008	Clave demasiado larga.	Demasiadas columnas en una clave primaria, una clave foránea o en el índice.
54010	La longitud del registro de tabla es demasiado larga.	La longitud del registro de la tabla es demasiado larga.
54011	Se han especificado demasiadas columnas para una tabla o vista.	Se han especificado demasiadas columnas para una tabla o vista.
55002	DB2ePLANTABLE no está definido correctamente.	EXPLAIN no se puede ejecutar con una declaración incorrecta de DB2ePLANTABLE.
55009	Archivo de sólo lectura.	El archivo es de sólo lectura. En un entorno de sólo lectura, solamente se pueden ejecutar consultas SELECT.
57001	Tabla no disponible.	REORG no se puede ejecutar en una tabla que está bajo el ámbito de una transacción.
57011	Falta de memoria.	El sistema no puede asignar memoria.
57014	El proceso se canceló debido a una interrupción.	La ejecución de una consulta se cancela debido a una interrupción del usuario.
58004	Error interno del sistema (continuar).	Se ha producido un error no grave del sistema.
58005	Error interno del sistema (detener).	Se ha producido un error grave del sistema.

Listado de los SQLSTATE

Este tema le ayudará a interpretar los mensajes de error generados desde SQL o CLI.

- "Resumen de códigos de clase de SQLState" en la página 162 contiene un listado de las categorías generales de los errores.
- "Mensajes de SQLSTATE notificados por SQL" en la página 158y "Mensajes de SQLState notificados por JDBC" en la página 157 contienen descripciones de cada error y, para SQL, proporcionan el nombre de la función que lo generó.

Puede también obtener descripciones de los estados de SQL (SQLSTATE) utilizando un procesador de línea de mandatos de DB2, si tiene instalado DB2 Versión 9.1:

1. Para abrir el procesador de línea de mandatos, seleccione **Inicio → Programas → DB2 → Procesador de línea de mandatos**.
2. En la línea de mandatos, escriba `? [SQLSTATE]`.

Resumen de códigos de clase de SQLState

Los dos primeros caracteres de los mensajes de SQLState están en **negrita** para indicar el código de clase. Estos códigos de clase aparecen *resumidos* en la Tabla 72.

Tabla 72. Códigos de la clase SQLState

Código	Clase
00	Terminación satisfactoria no calificada
01	Aviso
02	No hay datos
07	Error de SQL dinámico
08	Excepción de conexión
09	Excepción de acción desencadenada
0A	Característica no soportada
0F	Símbolo no válido
21	Violación de la cardinalidad
22	Excepción de datos
23	Violación de restricción
24	Estado no válido del cursor
25	Estado no válido de la transacción
26	Identificador no válido de sentencia de SQL
28	Especificación no válida de autorización
2D	Terminación no válida de transacción
2E	Nombre no válido de conexión
3B	Punto de rescate no válido
34	Nombre no válido de cursor
38	Excepción de función externa
39	Excepción de llamada a función externa
40	Retrotracción de transacción
42	Error de sintaxis o violación de regla de acceso
44	Violación de opción con comprobación
46	DDL de Java
51	Estado no válido de aplicación
54	Límite excedido de SQL o del producto
55	Objeto no encontrado en estado previo necesario
56	Diversos errores de SQL o del producto
57	Recurso no disponible o intervención del operador
58	Recurso erróneo del sistema
59	Error del administrador de DB2 Everyplace
HY	Generado por el controlador CLI u ODBC de DB2
IM	Generado por el gestor de controladores ODBC

Glosario

Caracteres especiales

$DSYINSTDIR
Hace referencia al directorio en el que está instalado DB2 Everyplace en un sistema Linux o UNIX.

<DSYPATH>
Hace referencia al directorio en el que está instalado DB2 Everyplace en un sistema Windows.

A

agenda personal electrónica (PDA)
Dispositivo de bolsillo que se utiliza para tareas personales de organización (tales como llevar una agenda de actividades o tomar notas) y que incluye servicios de teléfono, fax y conexión a una red.

archivo de anotaciones cronológicas
Objeto del Centro de administración de dispositivos portátiles que contiene mensajes de error de sincronización y sus descripciones.

autenticación
Proceso de validar el ID y la contraseña de un usuario por comparación con las entradas de la base de datos de control para comprobar que el usuario está autorizado para utilizar DB2 Everyplace Sync Server para sincronizar datos.

autorización
En la seguridad de sistemas informáticos, derecho que se otorga a un usuario para comunicarse con un sistema o hacer uso de él.

B

base de datos corporativa
Véase *base de datos fuente*.

base de datos de destino
Base de datos de DB2 Everyplace contenida en un dispositivo portátil en la que se copian datos procedentes de una base de datos fuente.

base de datos de réplica
Base de datos que DB2 Everyplace Sync Server utiliza internamente para almacenar los datos necesarios para la sincronización y duplicación.

base de datos fuente
Base de datos que reside en un servidor fuente que contiene los datos que deben copiarse en un sistema destino.

base de datos local
Base de datos que está ubicada físicamente en el sistema que se está utilizando. Compárese con *base de datos remota*.

base de datos maestra
Véase *base de datos fuente*.

base de datos remota
Base de datos que está ubicada físicamente en un sistema distinto del que se está utilizando. Compárese con base de datos local. El sistema remoto puede ser o no portátil.

BLOB Véase *objeto binario grande*.

C

calificador de Apply
Serie de caracteres que identifica definiciones de suscripción que son exclusivas de cada instancia del programa Apply de DataPropagator.

Centro de administración de dispositivos portátiles (MDAC)
Interfaz gráfica que permite al usuario crear, editar y visualizar objetos de sincronización y sus relaciones entre sí. El Centro de administración de dispositivos portátiles también permite ver el estado de sincronización de clientes individuales y mensajes de error.

Centro de control
Interfaz gráfica que muestra objetos de base de datos (tales como bases de datos y tablas) y su relación entre ellos. Desde el Centro de control, se pueden realizar las tareas proporcionadas por las herramientas DBA Utility, Visual Explain y Performance Monitor.

Centro de control de DB2
Véase *Centro de control*.

clave Columna o colección ordenada de columnas que se identifican en la descripción de una tabla, índice o restricción de referencia.

clave primaria
Clave exclusiva que forma parte de la definición de una tabla. Una clave primaria es la clave padre predeterminada de una definición de restricción referencial. Cuando se utiliza DB2 Everyplace Sync Server Versión 7, cada fuente de duplicación debe tener una sola clave primaria.

cliente
Programa o usuario que se comunica con un servidor de bases de datos y accede a él. Los clientes se definen mediante el Centro de administración de dispositivos portátiles.

conjunto de suscripción
Objeto del Centro de administración de dispositivos portátiles que contiene suscripciones de duplicación. Para que los miembros de un grupo accedan a los datos y archivos definidos en las suscripciones de duplicación, se debe crear un conjunto de suscripción, asignarle suscripciones y luego asignarlo a un grupo. El objeto del conjunto de suscripción sustituye al objeto de la aplicación.

consulta
Petición de información de la base de datos basada en condiciones específicas; por ejemplo, una petición de una lista de todos los clientes de una tabla de clientes cuyo saldo sea superior a 150.000 Pts.

D

DBCS Véase *juego de caracteres de doble byte*.

DB2 DataPropagator
Producto de duplicación que proporciona un método automático de duplicación de datos fuente en sistemas de destino. Durante la sincronización de datos portátiles, las bases de datos reflejo y remota actúan como fuente y destino de los datos. DataPropagator duplica en la base de datos remota los cambios realizados por los sistemas cliente en la base de datos reflejo, y también duplica en la base de datos reflejo los cambios procedentes de la base de datos remota.

DES Véase *Estándar de cifrado de datos*.

detección de conflictos
Proceso de detectar una fila no actualizada en una tabla destino que fue actualizada por una aplicación de usuario. Cuando se detecta un conflicto, se rechaza la transacción que provocó el conflicto.

DHCP Véase *Protocolo de configuración dinámica de sistemas principales*.

dispositivo de bolsillo
Cualquier dispositivo informático que se pueda sostener en la mano. Son ejemplos de tales dispositivos los sistemas PC de bolsillo y las agendas personales electrónicas (personal digital assistants, PDA).

DPROP
Véase *DB2 DataPropagator*.

duplicación
Proceso en el que se toman los cambios almacenados en el archivo de anotaciones o diario de la base de datos, en el servidor fuente, y se aplican en un servidor destino.

E

Estándar de cifrado de datos (DES)
Algoritmo de cifrado diseñado para cifrar y descifrar datos utilizando una clave privada.

Estándar de cifrado de datos triple (triple DES)
Algoritmo de cifrado por bloques que se puede utilizar para cifrar datos transmitidos entre sistemas gestionados y el servidor de gestión. Triple DES es una mejora de seguridad de DES que emplea tres operaciones de bloque DES sucesivas.

F

filtro Dispositivo o programa que separa datos, señales o información de acuerdo con criterios determinados.

filtro de datos
Véase *filtro*.

fuente de duplicación
Tabla de base de datos que está definida como fuente de la duplicación. Una vez definida una tabla de base de datos como fuente de duplicación, la tabla puede aceptar peticiones de copia.

G

grupo Colección de clientes que tienen necesidades similares respecto a la sincronización de datos portátiles. Para cada grupo se definen características de sincronización, tales como qué aplicaciones necesitan acceder los usuarios del grupo para realizar sus tareas y a qué conjuntos de datos corporativos necesitan acceder.

I

IBM Sync
Nombre del icono que sirve para representar el componente cliente del software de DB2 Everyplace Sync Server.

informática distribuida
Uso de una infraestructura informática que incluye dispositivos de información desde los cuales el usuario puede acceder a una amplia gama de servicios a través de una red (incluidos los servicios que generalmente se ofrecen a través de Internet). Estos dispositivos de información pueden ser televisores, automóviles, teléfonos, refrigeradores y hornos microondas. La informática distribuida proporciona un acceso apropiado a la información pertinente y la capacidad para responder de acuerdo con esa información.

J

juego de caracteres de doble byte (double-byte character set, DBCS)
Juego de caracteres en el que cada carácter se representa mediante dos bytes.

L

LAN inalámbrica
En las aplicaciones inalámbricas, un usuario de portátiles se puede conectar a una red de área local (LAN) mediante una conexión de radiofrecuencia. La tecnologías inalámbricas para la conexión LAN incluyen espectro de velocidad, microondas y luz infrarroja.

Lenguaje de consulta estructurada (Structured Query Language, SQL)
Lenguaje de programación que se utiliza para definir y manejar datos de una base de datos relacional.

LOB Véase *objeto grande*.

M

MDAC
Véase *Centro de administración de dispositivos portátiles*.

O

objeto Cualquier elemento que se pueda crear o manipular con SQL; por ejemplo, tablas, vistas, índices o paquetes. En la programación o el diseño orientado a objetos, abstracción que consta de datos y operaciones asociadas a dichos datos.

objeto binario grande (BLOB)
Secuencia de bytes cuyo tamaño está comprendido entre 0 y 2 gigabytes. Esta secuencia de bytes no tiene una página de códigos ni un juego de caracteres asociados. Los objetos de imagen, audio y vídeo se almacenan en forma de objetos BLOB.

objeto de sincronización
Elemento gestionable, dentro del Centro de administración de dispositivos portátiles, que contiene información sobre aspectos del proceso de sincronización de datos utilizado. Existen cinco tipos de objetos de sincronización: grupo, cliente, conjunto de suscripción, suscripción y archivo de anotaciones.

objeto grande (large object, LOB)
Secuencia de bytes cuyo tamaño puede ser de hasta 2 gigabytes. Puede ser de uno de estos tres tipos: BLOB (binario), CLOB (caracteres de un solo byte o mixtos) o DBCLOB (caracteres de doble byte).

ODBC
Véase *Open Database Connectivity*.

Open Database Connectivity (ODBC)
API que permite acceder a sistemas de gestión de bases de datos utilizando SQL invocable, el cual no necesita la utilización de un preprocesador de SQL. La arquitectura ODBC permite a los usuarios añadir módulos, denominados controladores de bases de datos, que enlazan la aplicación, en tiempo de ejecución, a los sistemas de gestión de bases de datos que hayan elegido. No es necesario enlazar directamente las aplicaciones con los módulos de todos los sistemas de gestión de bases de datos soportados.

P

PDA Véase *agenda personal electrónica*.

persistente
Relativo a datos que se conservan al pasar de una sesión a otra, generalmente en almacenamiento no volátil, tal como un sistema de base de datos o un directorio.

portátil
Relativo a los procesos de cálculo que se realizan en un sistema portátil o dispositivo de bolsillo por parte de un usuario que se desplaza con frecuencia y utiliza diferentes tipos de conexiones de red (por ejemplo, línea conmutada, LAN o comunicaciones inalámbricas).

privilegio
Derecho a tener acceso a un objeto específico de la base de datos de un modo específico. Estos derechos los controlan los usuarios con autorización SYSADM (administrador del sistema), autorización DBADM (administrador de bases de datos) o los creadores de los objetos. Los privilegios incluyen derechos tales como crear, suprimir y seleccionar datos de las tablas.

Protocolo de configuración dinámica de sistemas principales (Dynamic Host Configuration Protocol, DHCP)
Protocolo de Internet para automatizar la configuración de sistemas que hacen uso de TCP/IP.

pulsar Uso de un puntero para interaccionar con un dispositivo de bolsillo.

PVC Véase *informática distribuida*.

R

RAS Véase *servicio de acceso remoto*.

renovar
Proceso en el que todos los datos de interés de una tabla de usuario se copian en la tabla destino, sustituyendo los datos existentes.

S

servicio de acceso remoto (Remote Access Service, RAS)
Programa de Windows que gestiona las conexiones entre dos sistemas.

servidor corporativo
Véase *servidor fuente*.

servidor de bases de datos
Unidad operativa que proporciona servicios para bases de datos.

servidor fuente
Ubicación de base de datos de la fuente de duplicación.

sesión de sincronización
Transacción en la que los usuarios de dispositivos portátiles, o *clientes*, envían someten cambios realizados en copias locales de datos fuente y reciben los cambios realizados en datos fuente (contenidos en el servidor remoto) desde la última vez que se sincronizaron los datos.

sincronización
Véase *sincronización de datos portátiles*.

sincronización de datos
Véase *sincronización de datos portátiles*.

sincronización de datos portátiles
Proceso en dos etapas por medio del cual los usuarios de dispositivos portátiles, o

clientes, someten cambios hechos en copias locales de datos fuente y reciben los cambios realizados en datos fuente (contenidos en una base de datos remota) desde la última vez que se sincronizaron los datos.

sistema de gestión de bases de datos (database management system, DBMS)
Programa informático de gestión de datos, que proporciona control centralizado de servicios, independencia de los datos y estructuras físicas complejas para conseguir un acceso eficaz, integridad, recuperación, control de concurrencia, privacidad y seguridad.

sistema de nivel medio
Máquina donde está instalado DB2 Everyplace Sync Server. En una configuración de la sincronización con dos niveles, los términos sistema de nivel medio y sistema fuente designan una misma máquina.

SQL Véase *Lenguaje de consulta estructurada*.

suscripción
Especificación de cómo debe duplicarse la información de una base de datos fuente en una base de datos destino. Una suscripción le permite definir qué subconjuntos de datos y archivos se pueden copiar desde la base de datos fuente. Puede crear dos tipos de suscripciones: suscripciones para archivos almacenados en el servidor fuente y suscripciones para tablas contenidas en la base de datos fuente.

T

tabla de destino
Tabla en la que se copian datos de una tabla fuente. Las tablas de réplica del servidor de nivel medio son destinos, como también lo son las tablas de DB2 Everyplace que se encuentran en el dispositivo portátil.

tabla fuente
Tabla que contiene los datos que se deben copiar en una tabla destino. La tabla fuente debe ser una tabla fuente de duplicación. Compárese con *tabla destino*.

tabla temporal
Tabla creada durante el proceso de una sentencia de SQL para que contenga resultados intermedios.

triple DES
Véase *Estándar de cifrado de datos triple*.

U

unión Operación relacional que permite recuperar datos de dos o más tablas mediante la asociación de valores coincidentes de columnas.

V

vinculación
En SQL, proceso por el cual la salida del precompilador SQL se convierte en una estructura utilizable llamada plan de acceso. Durante este proceso, se seleccionan vías de acceso a los datos y se realiza una cierta comprobación de autorización.

vista Tabla lógica que consta de datos generados por una consulta.

Avisos

Es posible que IBM no comercialice en todos los países algunos productos, servicios o características descritos en este manual. Consulte al representante local de IBM para obtener información sobre los productos y servicios que actualmente pueden adquirirse en su zona geográfica. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implicar que sólo se puede utilizar dicho producto, programa o servicio de IBM. En su lugar se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no infrinja ninguno de los derechos de propiedad intelectual de IBM. Sin embargo, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patentes en tramitación que afecten al tema tratado en este documento. La posesión de este documento no confiere ninguna licencia sobre dichas patentes. Puede realizar consultas sobre licencias escribiendo a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
Estados Unidos

En el caso de consultas sobre licencias referentes a información de doble byte (DBCS), consulte al Departamento de Propiedad Intelectual de IBM en su país o envíe consultas por escrito a:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japón

El párrafo siguiente no es aplicable al Reino Unido ni a ningún país en el que tales disposiciones sean incompatibles con la legislación local: INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL", SIN GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN DE DERECHOS, COMERCIABILIDAD O IDONEIDAD PARA UN FIN DETERMINADO. Algunos estados no permiten la exclusión de garantías expresas o implícitas en determinadas transacciones, por lo que es posible que esta declaración no sea aplicable en su caso.

Esta publicación puede contener inexactitudes técnicas o errores tipográficos. Periódicamente se efectúan cambios en la información aquí contenida; dichos cambios se incorporarán a las nuevas ediciones de la publicación. IBM puede efectuar, en cualquier momento y sin previo aviso, mejoras y/o cambios en los productos y/o programas descritos en esta publicación.

Las referencias hechas en esta publicación a sitios Web que no son de IBM se proporcionan sólo para la comodidad del usuario y no constituyen un aval de esos sitios Web. La información contenida en esos sitios Web no forma parte de la información del presente producto IBM y el usuario es responsable de la utilización de esos sitios Web.

Cuando envía información a IBM, IBM puede utilizar o distribuir dicha información en la forma en que IBM considere adecuada, sin contraer por ello ninguna obligación con el remitente.

Los licenciatarios de este programa que deseen obtener información sobre él con el fin de habilitar: (i) el intercambio de información entre programas creados de forma independiente y otros programas (incluido este) y (ii) el uso mutuo de la información intercambiada, deben ponerse en contacto con:

IBM Corporation

J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
Estados Unidos

Dicha información puede estar disponible, sujeta a los términos y condiciones apropiados, incluido en algunos casos, el pago de una tarifa.

El programa bajo licencia descrito en este manual y todo el material bajo licencia asociado a él, los proporciona IBM según los términos del Convenio del Cliente IBM, el Convenio Internacional de Licencia de Programas de IBM o cualquier convenio equivalente entre el usuario e IBM.

Los datos de rendimiento contenidos en este documento se obtuvieron en un entorno controlado. Por tanto, los resultados obtenidos en otros entornos operativos pueden variar significativamente. Algunas mediciones pueden haberse hecho en sistemas experimentales y no es seguro que estas mediciones sean las mismas en los sistemas disponibles comercialmente. Además, algunas mediciones pueden haberse calculado mediante extrapolación. Los resultados reales pueden variar. Los usuarios del presente manual deben verificar los datos aplicables para su entorno específico.

La información referente a productos que no son de IBM se ha obtenido de los proveedores de esos productos, de sus anuncios publicados o de otras fuentes disponibles públicamente. IBM no ha probado esos productos y no puede confirmar la exactitud del rendimiento, la compatibilidad ni cualquier otra afirmación referente a productos no IBM. Las preguntas sobre las prestaciones de productos no IBM deben dirigirse a los proveedores de esos productos.

Todas las declaraciones de intenciones de IBM están sujetas a cambio o cancelación sin previo aviso, y sólo representan objetivos.

Esta publicación puede contener ejemplos de datos e informes que se utilizan en operaciones comerciales diarias. Para ilustrarlos de la forma más completa posible, los ejemplos incluyen nombre de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier similitud con nombres y direcciones utilizados por una empresa real es totalmente no intencionada.

LICENCIA DE COPYRIGHT:

Este manual puede contener programas de aplicación de ejemplo escritos en lenguaje fuente, que muestra técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo de la forma que desee, sin pago alguno a IBM, con los fines de desarrollar, utilizar, comercializar o distribuir programas de aplicación de acuerdo con la interfaz de programación de aplicaciones correspondiente a la plataforma operativa para la que están escritos los programas de ejemplo. Estos ejemplos no se han probado exhaustivamente bajo todas las condiciones. Por tanto, IBM no puede asegurar ni implicar la fiabilidad, utilidad o función de estos programas.

Cada copia o porción de estos programas de ejemplo o cualquier trabajo derivado debe incluir una nota de copyright como la siguiente:

© (nombre de la empresa) (año). Partes de este código derivan de programas de ejemplo de IBM Corp. © Copyright IBM Corp. _especifique el año o años_. Reservados todos los derechos.

Este producto incluye software desarrollado por 3Com y sus colaboradores.:

Copyright (c) 1998 3Com/Palm Computing Division. Reservados todos los derechos. La redistribución y el uso en los formatos fuente y binario, con o sin modificación, están permitidos siempre que se cumplan las condiciones siguientes:

1. Las redistribuciones de código fuente deben conservar la nota de copyright mostrada anteriormente, la presente lista de condiciones y la nota de renuncia que viene a continuación.

2. Las redistribuciones en formato binario deben reproducir la nota de copyright mostrada anteriormente, la presente lista de condiciones y la nota de renuncia que viene a continuación en la documentación y/u otros elementos proporcionados con la distribución.
3. Todos los textos publicitarios que mencionen características o el uso de este software deben mostrar el siguiente reconocimiento: Este producto incluye software desarrollado por 3Com y sus colaboradores.
4. Ni 3Com ni los nombres de sus colaboradores pueden utilizarse para avalar o promocionar productos derivados de este software sin permiso específico previo por escrito.

3COM Y SUS COLABORADORES PROPORCIONAN ESTE SOFTWARE ''TAL CUAL'', SIN GARANTÍAS EXPRESAS NI IMPLÍCITAS, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE COMERCIABILIDAD O IDONEIDAD PARA UN FIN DETERMINADO. EN NINGÚN CASO 3COM NI SUS COLABORADORES SERÁN RESPONSABLES POR NINGÚN DAÑO DIRECTO, INDIRECTO, INCIDENTAL, ESPECIAL, EJEMPLAR NI CONSECUENTE (INCLUIDOS, PERO SIN LIMITARSE A ELLOS, LA OBTENCIÓN DE BIENES O SERVICIOS SUSTITUTOS; LA PÉRDIDA DE USO, DATOS O BENEFICIOS; O LA INTERRUPCIÓN DE LA ACTIVIDAD COMERCIAL) CUALQUIERA QUE SEA LA CAUSA (INCLUIDA LA NEGLIGENCIA), QUE SURJA COMO CONSECUENCIA DEL USO DEL PRESENTE SOFTWARE, INCLUSO SI SE INFORMA DE LA POSIBILIDAD DE TALES DAÑOS.

Marcas registradas

Los términos siguientes, que pueden estar indicados por un asterisco (*), son marcas registradas de International Business Machines Corporation en los Estados Unidos y/o en otros países.

AIX
AS/400
Cloudscape
DataPropagator
DB2
DB2 Connect
DB2 Universal Database
Everyplace
IBM
iSeries
OS/390
WebSphere
z/OS
zSeries

Los términos siguientes son marcas registradas de otras empresas:

Intel, Intel Inside (logotipos), MMX y Pentium son marcas registradas de Intel Corporation en los Estados Unidos y/o en otros países.

Microsoft, Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en los Estados Unidos y/o en otros países.

Linux es una marca registrada de Linus Torvalds en los Estados Unidos y/o en otros países.

Java y todas las marcas registradas basadas en Java son marcas registradas de Sun Microsystems, Inc. en los Estados Unidos y/o en otros países.

UNIX es una marca registrada de The Open Group en los Estados Unidos y/o en otros países.

Otros nombres de empresas, productos o servicios, que pueden estar indicados por un doble asterisco (**), pueden ser marcas registradas o marcas de servicio de otras empresas.

Índice

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

Número de Programa: