DB2

IBM

DB2 Versión 9
para Linux, UNIX y Windows

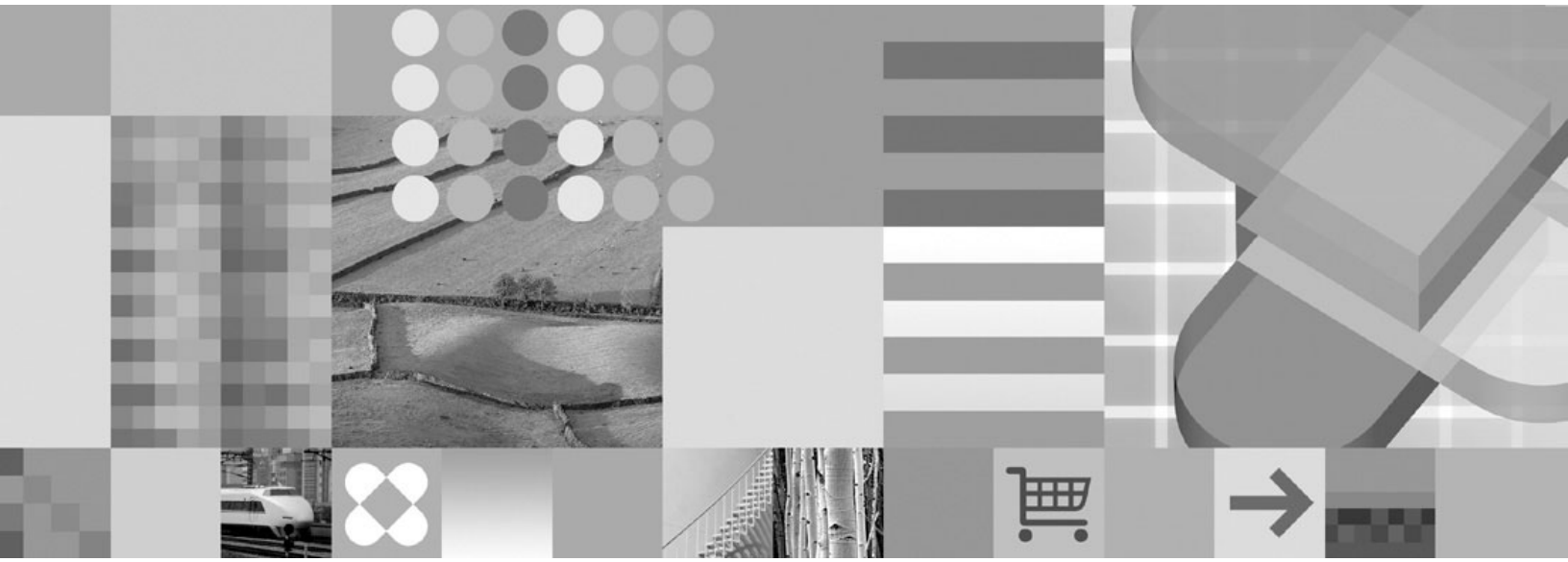
Desarrollo de aplicaciones ADO.NET y OLE DB

SC11-3178-00

DB2®

IBM

DB2 Versión 9
para Linux, UNIX y Windows



Desarrollo de aplicaciones ADO.NET y OLE DB

SC11-3178-00

Antes de utilizar esta información y el producto al que da soporte, asegúrese de leer la información general incluida en el apartado *Avisos*.

Información sobre la edición

Esta publicación es la traducción del original inglés *DB2 Version 9 for Linux, UNIX, and Windows Developing ADO.NET and OLE DB Applications*, (SC10-4230-00).

Puede realizar pedidos de publicaciones en línea o a través del representante de IBM de su localidad.

- Para realizar pedidos de publicaciones en línea, vaya a IBM Publications Center en www.ibm.com/shop/publications/order
- Para encontrar el representante de IBM correspondiente a su localidad, vaya a IBM Directory of Worldwide Contacts en www.ibm.com/planetwide

Para realizar pedidos de publicaciones en marketing y ventas de DB2 de los EE.UU. o de Canadá, llame al número 1-800-IBM-4YOU (426-4968).

Cuando envía información a IBM, otorga a IBM un derecho no exclusivo para utilizar o distribuir dicha información en la forma en que IBM considere adecuada, sin contraer por ello ninguna obligación con el remitente.

Contenido

Capítulo 1. Desarrollo ADO.NET para bases de datos DB2

Desarrollo de aplicaciones ADO.NET

En años recientes, Microsoft ha estado promocionando una nueva plataforma de desarrollo de software para Windows, conocida como .NET Framework. .NET Framework es la alternativa de Microsoft para la tecnología de Component Object Model (COM). Los puntos siguientes resaltan las características claves de .NET Framework:

- Puede codificar las aplicaciones .NET en más de cuarenta lenguajes de programación diferentes. Los lenguajes más populares para el desarrollo de .NET son C# y Visual Basic .NET.
- La biblioteca de clase de .NET Framework proporciona los bloques de creación con los que han de crearse las aplicaciones .NET. Esta biblioteca de clase tiene un lenguaje agnóstico y proporciona interfaces al sistema operativo y servicios de aplicación.
- La aplicación .NET (sin tener en cuenta el lenguaje) se compila en Intermediate Language (IL), un tipo de código de bytes.
- Common Language Runtime (CLR) es el corazón de .NET Framework, compilando el código de IL al vuelo y ejecutándolo a continuación. Al ejecutar el código IL compilado, CLR activa los objetos, verifica su autorización de seguridad, asigna su memoria, los ejecuta y limpia su memoria una vez haya finalizado la ejecución.

Por medio de estas características, .NET Framework facilita una amplia gama de implementaciones de aplicación (por ejemplo, los formatos de Windows, los formatos de Web y los servicios Web), el desarrollo de aplicación rápida y el desarrollo de aplicación protegido. COM y COM+ han demostrado que eran inadecuados o engorrosos para todas las características mencionadas anteriormente.

.NET Framework proporciona un amplio soporte de acceso de datos por medio de ADO.NET. ADO.NET da soporte tanto al acceso conectado como al desconectado. El componente clave del acceso de datos desconectado en ADO.NET es la clase DataSet, cuyas instancias actúan como antememoria de la base de datos que reside en la memoria de la aplicación. El acceso conectado en ADO.NET no requiere clases adicionales.

Tanto para el acceso conectado como para el desconectado, las aplicaciones utilizan bases de datos a través de lo que se conoce como proveedor de datos .NET. Diversos productos de la base de datos incluyen sus propios proveedores de datos .NET, incluyendo DB2 para Windows.

Un proveedor de datos .NET presenta implementaciones de las siguientes clases básicas:

- Conexión: Establece y gestiona una conexión de base de datos.
- Mandato: Ejecuta una sentencia SQL en una base de datos.
- DataReader: Lee y devuelve datos del conjunto de resultados de una base de datos.
- DataAdapter: Enlaza una instancia de DataSet a una base de datos. Por medio de una instancia de DataAdapter, DataSet puede leer y grabar datos de tabla de la base de datos.

DB2 UDB para Windows incluye tres proveedores de datos .NET:

- DB2 .NET Data Provider: Proveedor de datos ADO.NET gestionado de alto rendimiento. Este es el proveedor de datos .NET recomendado para su utilización con bases de datos de la familia de DB2. El acceso a la base de datos de ADO.NET utilizando el DB2 .NET Data Provider tiene menos restricciones y proporciona un rendimiento significativamente mejor que el de los proveedores de OLE DB y ODBC .NET Bridge.
- OLE DB .NET Data Provider: Proveedor Bridge que alimenta las peticiones ADO.NET para IBM OLE DB Provider (por medio del módulo interop COM). Este .NET Data Provider no es el que se recomienda para acceder a las bases de datos de la familia de DB2. El DB2 .NET Data Provider es más rápido y tiene más características.
- ODBC DB .NET Data Provider: Proveedor Bridge que alimenta las peticiones ADO.NET para IBM ODBC Driver. Este .NET Data Provider no es el que se recomienda para acceder a las bases de datos de la familia de DB2. El DB2 .NET Data Provider es más rápido y tiene más características.

Software de desarrollo .NET soportado

Para desarrollar y desplegar aplicaciones .NET que se ejecuten en bases de datos de DB2, necesitará utilizar software de desarrollo y sistemas operativos soportados.

Sistemas operativos soportados para el desarrollo y despliegue de aplicaciones .NET Framework 1.1:

- Windows 2000
- Windows XP (edición de 32 bits)
- Windows Server 2003 (edición de 32 bits)

Nota: Windows 98, Windows ME y Windows NT también reciben soporte, pero sólo para ejecutarse en aplicaciones del cliente DB2.

Sistemas operativos soportados para el desarrollo y despliegue de aplicaciones .NET Framework 2.0:

- Windows 2000, Service Pack 3
- Windows XP, Service Pack 2 (ediciones de 32 bits y de 64 bits)
- Windows Server 2003 (ediciones de 32 bits y de 64 bits)

Nota: Windows 98 y Windows ME también reciben soporte, pero sólo para ejecutarse en aplicaciones del cliente DB2.

Software de desarrollo soportado para aplicaciones .NET Framework:

Además de un cliente DB2, necesitará una de las opciones siguientes para desarrollar aplicaciones .NET Framework.

- Visual Studio 2003 (para aplicaciones .NET Framework 1.1)
- Visual Studio 2005 (para aplicaciones .NET Framework 2.0)
- .NET Framework 1.1 Software Development Kit y .NET Framework Versión 1.1 Redistributable Package (para aplicaciones .NET Framework 1.1)
- .NET Framework 2.0 Software Development Kit y .NET Framework Versión 2.0 Redistributable Package (para aplicaciones .NET Framework 2.0)

Software de despliegue soportado para aplicaciones .NET Framework:

Además de un cliente DB2, necesitará una de las dos opciones siguientes para desplegar aplicaciones .NET Framework.

- .NET Framework Versión 1.1 Redistributable Package (para aplicaciones .NET Framework 1.1)
- .NET Framework Versión 2.0 Redistributable Package (para aplicaciones .NET Framework 2.0)

Configuración del entorno de desarrollo de aplicaciones Windows

Cuando se instala Cliente DB2 en sistemas operativos Windows, el programa de instalación actualiza el registro de configuración con las variables de entorno INCLUDE, LIB y PATH. La instalación establece la variable de entorno para todo el sistema, DB2INSTANCE, en la instancia por omisión creada, denominada DB2. DB2PATH se establece dentro de una ventana de mandatos de DB2 cuando se abre la ventana.

Puede alterar estas variables de entorno para definir valores para la máquina o usuario conectado actualmente. Tenga precaución cuando modifique estas variables de entorno. No modifique la variable de entorno DB2PATH. DB2INSTANCE se define como variable de entorno a nivel de sistema. No es necesario hacer uso de la variable de registro DB2INSTDEF de DB2, que define el nombre de instancia por omisión que se debe utilizar si no se establece DB2INSTANCE.

Procedimiento:

Para modificar los valores de variables de entorno, utilice el Panel de control de Windows.

Si utiliza la variable `%DB2PATH%` en un mandato, coloque la vía de acceso completa entre comillas, de esta forma:

```
set LIB="%DB2PATH%%\lib";%LIB%
```

El valor de instalación por omisión para esta variable es `\Archivos de programa\IBM\SQLLIB`, que contiene un espacio, por lo que es necesario utilizar comillas para no ocasionar un error.

Además, debe seguir los pasos específicos siguientes para ejecutar aplicaciones de DB2:

- Cuando cree programas C o C++, la variable de entorno `INCLUDE` debe incluir `%DB2PATH%\INCLUDE` como primer directorio.

  Para hacerlo, actualice el archivo de configuración del entorno para el compilador:

  Microsoft Visual C++ 6.0

  ```
  "C:\Archivos de programa\Microsoft Visual Studio\VC98\bin\
  vcvars32.bat"
  ```

  Microsoft Visual C++ .NET

  ```
  "C:\Archivos de programa\Microsoft Visual Studio
  .NET\Common7\Tools\vsvars32.bat"
  ```

  Estos archivos tienen los mandatos siguientes:

  Microsoft Visual C++ 6.0

  ```
  set INCLUDE=%MSVCDir%\ATL\INCLUDE;%MSVCDir%\INCLUDE;
     %MSVCDir%\MFC\INCLUDE;%INCLUDE%
  ```

Microsoft Visual C++ .NET

```
@set INCLUDE=%MSVCDir%\ATLMFC\INCLUDE;...;
   %FrameworkSDKDir%\include;%INCLUDE%
```

Para utilizar cualquiera de estos archivos con DB2, primero traslade %INCLUDE%, que define la vía %DB2PATH%\INCLUDE, desde el final de la lista hasta el comienzo, de esta manera:

Microsoft Visual C++ 6.0

```
set INCLUDE=%INCLUDE%;%MSVCDir%\ATL\INCLUDE;
   %MSVCDir%\INCLUDE;%MSVCDir%\MFC\INCLUDE
```

Microsoft Visual C++ .NET

```
@set INCLUDE=%INCLUDE%;%MSVCDir%\ATLMFC\INCLUDE;...;
   %FrameworkSDKDir%\include
```

- Cuando cree programas Micro Focus COBOL, la variable de entorno COBCPY debe apuntar a %DB2PATH%\INCLUDE\cobol_mf.
- Cuando cree programas IBM COBOL, la variable de entorno SYSLIB debe apuntar a %DB2PATH%\INCLUDE\cobol_a.
- Asegúrese de que la variable de entorno LIB apunta a %DB2PATH%\lib utilizando este mandato:

  ```
  set LIB="%DB2PATH%\lib";%LIB%
  ```

  Nota: Para permitir el desarrollo cruzado de aplicaciones de 64 bits desde un entorno de 32 bits, consulte el apartado Migración de aplicaciones de bases de datos de 32 bits para ejecutarlas en instancias de 64 bits
- Asegúrese de que la variable de entorno DB2COMM esté definida en el servidor de una base de datos remota.
- Asegúrese que el servicio de seguridad se ha iniciado en el servidor para la autenticación SERVER, y en el cliente, cuando se utilice la autenticación CLIENT.

  Nota: Puesto que la autenticación CLIENT se produce en la parte de cliente en lugar de la parte de servidor, la aplicación cliente se ejecuta bajo el contexto del usuario. La API de autenticación de Win32 requiere determinados privilegios que el usuario puede tener o no tener. Para asegurarse que la autenticación CLIENT tiene lugar satisfactoriamente, las solicitudes de autenticación se pasan de la aplicación cliente al servidor de seguridad (que se ejecuta, por omisión, bajo una sistema local de cuenta privilegiada y tiene el derecho de llamar a la API de autenticación).

  Para iniciar manualmente el servicio de seguridad, utilice el mandato NET START DB2NTSECSERVER.

  Normalmente, la única situación en que deseará iniciar automáticamente el servicio de seguridad será cuando la estación de trabajo actúe como cliente DB2 que se conecta a un servidor que esta configurado para la autenticación de clientes. Para hacer que el servicio de seguridad se inicie automáticamente, siga estos pasos:

  Windows 2000 y Windows Server 2003

  1. Pulse el botón Inicio.
  2. Para Windows 2000, pulse "Configuración" y luego pulse "Panel de control".

     Para Windows Server 2003, pulse "Panel de control".
  3. Pulse Herramientas Administrativas.
  4. Pulse Servicios.

5. En la ventana Servicios, resalte Servidor de Seguridad DB2.
6. Si esta opción no muestra los valores ″Iniciado″ y ″Automático″, pulse Acción en el menú superior.
7. Pulse Propiedades.
8. Vaya a la página General.
9. Seleccione ″Automático″ en el menú desplegable Tipo de Arranque.
10. Pulse Bien.
11. Rearranque la máquina para que los valores seleccionados sean efectivos.

Windows XP

1. Pulse el botón Inicio.
2. Pulse Configuración.
3. Pulse Panel de Control.
4. Pulse Rendimiento y Mantenimiento.
5. Pulse Herramientas Administrativas.
6. Pulse Servicios.
7. En la ventana Servicios, resalte Servidor de Seguridad DB2.
8. Si esta opción no muestra los valores ″Iniciado″ y ″Automático″, pulse Acción en el menú superior.
9. Pulse Propiedades.
10. Vaya a la página General.
11. Seleccione ″Automático″ en el menú desplegable Tipo de Arranque.
12. Pulse Bien.
13. Rearranque la máquina para que los valores seleccionados sean efectivos.

El gestor de bases de datos en un entorno Windows XP, Windows Server 2003 o Windows 2000 se implanta como servicio y, por tanto, no devuelve errores ni avisos cuando se inicia el servicio, aunque se puedan haber producido problemas. Esto supone que cuando ejecuta el mandato `db2start` o `NET START`, no se emiten avisos si el subsistema de comunicaciones no arranca. Por consiguiente, el usuario debe siempre examinar los registros de sucesos o el registro de notificaciones de Administración de DB2 para conocer los errores que se puedan haber producido al ejecutar estos mandatos.

Si piensa utilizar la CLI de DB2 o Java, vaya a la tarea apropiada:

– Setting up the Windows CLI environment
– Setting up the Windows CLI environment

Tareas relacionadas:

- “Migración de aplicaciones de bases de datos de 32 bits para ejecutarlas en instancias de 64 bits” en *Guía de migración*
- “Setting up the Windows CLI environment” en *Call Level Interface Guide and Reference, Volume 1*
- “Instalación del controlador IBM DB2 para JDBC y SQLJ” en *Desarrollo de aplicaciones Java*
- “Configuración del sistema operativo para el desarrollo de aplicaciones de bases de datos” en *Iniciación al desarrollo de aplicaciones de bases de datos*
- “Setting environment variables on Windows” en *Administration Guide: Implementation*

- "Setting the default instance when using multiple DB2 copies (Windows)" en *Administration Guide: Implementation*

Información relacionada:

- "Sistemas operativos soportados para el desarrollo de aplicaciones de bases de datos" en *Iniciación al desarrollo de aplicaciones de bases de datos*

Integración de DB2 en Visual Studio

IBM Database Add-Ins para Visual Studio 2003 y 2005 son un conjunto de características que se integran de modo imperceptible en el entorno de desarrollo de Visual Studio para que pueda trabajar con servidores de DB2 y desarrollar objetos, funciones y procedimientos de DB2. Estos add-ins se han diseñado para presentar una interfaz sencilla con las bases de datos de DB2. Por ejemplo, en vez de utilizar SQL, la creación de objetos de base de datos puede efectuarse utilizando asistentes. Y para las situaciones en las que necesite escribir código de SQL, el editor SQL de DB2 integrado tiene las siguientes características:

- Texto SQL en color para una mejor legibilidad
- Integración con la función Microsoft Visual Studio IntelliSense, que proporciona la complementación automática inteligente al escribir scripts DB2.

Con los IBM Database Add-Ins para Visual Studio, podrá:

- Abrir diversas herramientas de administración y desarrollo de DB2
- Crear y gestionar proyectos de DB2 en el Explorador de soluciones
- Acceder y gestionar conexiones de datos DB2 (en Visual Studio 2005 podrá hacer esto desde el Server Explorer; en Visual Studio 2003, podrá hacer esto desde el IBM Explorer)
- Crear y modificar scripts de DB2, incluyendo scripts para crear procedimientos almacenados, funciones, tablas, vistas, índices y activadores.

A continuación se proporcionan los medios por los que los IBM Database Add-Ins para Visual Studio pueden instalarse en el sistema.

Visual Studio 2003

Los IBM Database Add-Ins para Visual Studio 2003 se incluyen con los servidores de DB2 Client y DB2. La instalación de DB2 detecta la presencia de Visual Studio 2003 y si está instalado, se registran los add-ins. Si instala Visual Studio 2003 después de instalar un producto DB2, ejecute el programa de utilidad ″Registrar módulos adicionales de Visual Studio″ en el menú de inicio de la instancia de DB2.

Visual Studio 2005

Los IBM Database Add-Ins para Visual Studio 2005 se incluyen como componente instalable de modo independiente con los servidores de DB2 Client y DB2. Una vez se haya acabado de instalar el producto de DB2, aparecerá una opción para instalar los IBM Database Add-Ins para Visual Studio 2005. Si no ha instalado Visual Studio 2005 en el sistema, no se instalarán los add-ins. Una vez haya instalado Visual Studio 2005, podrá instalar los add-ins en cualquier momento desde el menú de configuración del producto de DB2.

Capítulo 2. Aplicaciones de programación para DB2 .NET Data Provider

DB2 .NET Data Provider

DB2 .NET Data Provider amplía el soporte de DB2 para la interfaz ADO.NET. DB2 .NET Data Provider proporciona un acceso seguro y de alto rendimiento a datos DB2.

DB2 .NET Data Provider le permite que sus aplicaciones .NET accedan a los siguientes sistemas de gestión de bases de datos:

- DB2 Database para Linux, UNIX y Windows, Versión 9
- DB2 Universal Database Versión 8 para sistemas Windows, UNIX y Linux
- DB2 Universal Database Versión 6 (o posterior) para OS/390 y z/OS, a través de DB2 Connect
- DB2 Universal Database Versión 5, Release 1 (o posterior) para AS/400 y iSeries, a través de DB2 Connect
- DB2 Universal Database Versión 7.3 (o posterior) para VSE & VM, a través de DB2 Connect

Para desarrollar y ejecutar aplicaciones que hacen uso de DB2 .NET Data Provider, es necesario .NET Framework, Versión 2.0 ó 1.1.

Además de DB2 .NET Data Provider, IBM Database Development Add-In permite desarrollar de forma rápida y sencilla aplicaciones .NET para bases de datos DB2 en Visual Studio 2005. También puede utilizar Add-In para crear objetos de base de datos como índices y tablas, y desarrollar objetos del servidor, como procedimientos almacenados de SQL y funciones definidas por el usuario.

Requisitos del sistema de bases de datos DB2 .NET Data Provider

DB2 .NET Data Provider permite que las aplicaciones .NET accedan a los siguientes sistemas de gestión de bases de datos:

- DB2 Versión 9 para Linux, UNIX y Windows
- DB2 Universal Database Versión 8 para Linux, UNIX y Windows
- DB2 Universal Database Versión 6 (o posterior) para OS/390 y z/OS, a través de DB2 Connect
- DB2 Universal Database Versión 5, Release 1 (o posterior) para AS/400 y iSeries, a través de DB2 Connect
- DB2 Universal Database Versión 7.3 (o posterior) para VSE & VM, a través de DB2 Connect

Antes de utilizar el programa de instalación de un cliente o servidor DB2 para instalar DB2 .NET Data Provider, debe tener ya instalado en el sistema .NET Framework (Versión 1.1 o Versión 2.0). Si .NET Framework no está instalado, el programa de instalación del cliente o servidor DB2 no instalará DB2 .NET Data Provider.

Para DB2 Universal Database para AS/400 e iSeries, es necesario el siguiente arreglo de programa en el servidor: APAR ii13348.

.NET Framework Versión 1.0 y Visual Studio .NET 2002 no se pueden utilizar con DB2 DB2 .NET Data Provider Versión 9.

Codificación genérica con las clases básicas comunes de ADO.NET

.NET Framework versión 2.0 resalta un espacio de nombres denominado `System.Data.Common`, el cual presenta un conjunto de clases básicas que puede compartir cualquier .NET Data Provider. Esta acción facilita un enfoque de desarrollo de la aplicación de base de datos ADO.NET genérico, que resalta una interfaz de programación constante. Las clases principales del DB2 .NET Data Provider para .NET Framework 2.0 se heredan de las clases básicas de `System.Data.Common`. En consecuencia, las aplicaciones de ADO.NET genéricas funcionarán con las bases de datos de DB2 por medio del DB2 .NET Data Provider.

El siguiente C# demuestra un enfoque genérico para establecer una conexión de base de datos.

```
DbProviderFactory factory = DbProviderFactories.GetFactory("IBM.Data.DB2");
DbConnection conn = factory.CreateConnection();
DbConnectionStringBuilder sb = factory.CreateConnectionStringBuilder();

if( sb.ContainsKey( "Database" ) )
{
   sb.Remove( "database" );
   sb.Add( "database", "SAMPLE" );
}

conn.ConnectionString = sb.ConnectionString;

conn.Open();
```

El objeto `DbProviderFactory` es el punto en el que comienza cualquier aplicación ADO.NET genérica. Este objeto crea instancias genéricas de objetos de proveedor de datos .NET, como por ejemplo conexiones, adaptadores de datos, mandatos y lectores de datos, que funcionan con un producto de base de datos específico. En el caso del ejemplo anterior, la serie `"IBM.Data.DB2"` que se pasa al método `GetFactory` identifica de modo exclusivo a DB2 .NET Data Provider y da como resultado la inicialización de una instancia de `DbProviderFactory` que crea instancias de objeto de proveedor de la base de datos específicas para DB2 .NET Data Provider. El objeto `DbConnection` puede conectarse a las bases de datos de la familia de DB2, igual que el objeto `DB2Connection`, que se hereda realmente desde `DbConnection`. Utilizando la clase `DbConnectionStringBuilder`, podrá determinar las palabras clave de la serie de conexión para un proveedor de datos y generar una serie de conexión personalizada. El código del ejemplo anterior comprueba si existe una palabra clave denominada `"database"` en DB2 .NET Data Provider y si es así, genera una serie de conexión para conectar con la base de datos de SAMPLE.

Conexión a una base de datos desde una aplicación utilizando DB2 .NET Data Provider

Cuando se utiliza DB2 .NET Data Provider, se establece una conexión con una base de datos a través de la clase `DB2Connection`. Primero, el usuario debe crear una serie de caracteres para contener los parámetros de conexión.

Son ejemplos de series de conexión:

```
String connectString = "Database=SAMPLE";
// Cuando se utiliza, intenta conectar con la base de datos SAMPLE.
```

```
String cs = "Server=srv:50000;Database=SAMPLE;UID=db2adm;PWD=ab1d;Connect Timeout=30";
// Cuando se utiliza, intenta conectar con la base de datos SAMPLE del servidor
// 'srv' a través del puerto 50000 y utilizando 'db2adm' y 'ab1d' // como ID de usuario y //
contraseña, respectivamente. Si el intento de conexión tarda más de treinta segundos,
el intento terminará y se generará un error.
```

Para crear la conexión de base de datos, pase la serie de conexión `connectString` al constructor de `DB2Connection`. Luego utilice el método `Open` del objeto `DB2Connection` para conectar formalmente con la base de datos especificada en `connectString`.

Conexión a una base de datos en C#:

```
String connectString = "Database=SAMPLE";
DB2Connection conn = new DB2Connection(connectString);
conn.Open();
return conn;
```

Conexión a una base de datos en Visual Basic .NET:

```
Dim connectString As String = "Database=SAMPLE"
Dim conn As DB2Connection = new DB2Connection(connectString)
conn.Open()
Return conn
```

Agrupación de conexiones con DB2 .NET Data Provider

La primera vez que se abra una conexión en la base de datos de DB2, se creará una agrupación de conexiones. Cuando se cierren las conexiones, éstas entran la agrupación, preparada para que la utilicen otras aplicaciones que necesitan conexiones. DB2 .NET Data Provider permite por omisión la agrupación de conexiones. Puede desactivar la agrupación de conexiones utilizando el par de palabra clave/valor de la serie de conexiones `Pooling=false`.

Puede controlar el comportamiento de la agrupación de conexiones estableciendo las palabras clave de la serie de conexiones para lo siguiente:

- El tamaño de agrupación mínimo y máximo (tamaño de agrupación mín, tamaño de agrupación máx)
- El tiempo que una conexión puede estar desocupada antes de devolverla a la agrupación (tiempo de vida para la conexión)
- Si la conexión actual se colocará o no en la agrupación de conexiones cuando se cierre (restauración de conexión)

Ejecución de sentencias de SQL desde una aplicación utilizando DB2 .NET Data Provider

Cuando se utiliza DB2 .NET Data Provider, la sentencias de SQL se ejecutan a través de la clase `DB2Command` utilizando sus métodos `ExecuteReader()` y `ExecuteNonQuery()`, y sus propiedades `CommandText`, `CommandType` y `Transaction`. Para las sentencias de SQL que generan datos de salida, se debe utilizar el método `ExecuteReader()` y sus resultados se pueden recuperar de un objeto `DB2DataReader`. Para todas las demás sentencias de SQL, se debe utilizar el método `ExecuteNonQuery()`. La propiedad `Transaction` del objeto `DB2Command` se debe inicializar para un objeto `DB2Transaction`. El objeto `DB2Transaction` es el encargado de retrotraer y confirmar las transacciones de base de datos.

Ejecución de una sentencia UPDATE en C#:

```
// se supone la existencia de una conexión DB2Connection
DB2Command cmd = conn.CreateCommand();
DB2Transaction trans = conn.BeginTransaction();
cmd.Transaction = trans;
cmd.CommandText = "UPDATE staff " +
                  "  SET salary = (SELECT MIN(salary) " +
                  "  FROM staff " +
    "               WHERE id >= 310) " +
                  "  WHERE id = 310";
cmd.ExecuteNonQuery();
```

Ejecución de una sentencia UPDATE en Visual Basic .NET:

```
' se supone la existencia de una conexión DB2Connection
DB2Command cmd = conn.CreateCommand();
DB2Transaction trans = conn.BeginTransaction();
cmd.Transaction = trans;
cmd.CommandText = "UPDATE staff " +
                  "  SET salary = (SELECT MIN(salary) " +
                  "  FROM staff " +
                  "                  WHERE id >= 310) " +
                  "  WHERE id = 310";
cmd.ExecuteNonQuery();
```

Ejecución de una sentencia SELECT en C#:

```
// se supone la existencia de una conexión DB2Connection
DB2Command cmd = conn.CreateCommand();
DB2Transaction trans = conn.BeginTransaction();
cmd.Transaction = trans;
cmd.CommandText = "SELECT deptnumb, location " +
                  "  FROM org " +
                  "  WHERE deptnumb < 25";
DB2DataReader reader = cmd.ExecuteReader();
```

Ejecución de una sentencia SELECT en Visual Basic .NET:

```
' se supone la existencia de una conexión DB2Connection
Dim cmd As DB2Command = conn.CreateCommand()
Dim trans As DB2Transaction = conn.BeginTransaction()
cmd.Transaction = trans
cmd.CommandText = "UPDATE staff " +
                  "  SET salary = (SELECT MIN(salary) " +
                  "  FROM staff " +
                  "                  WHERE id >= 310) " +
                  "  WHERE id = 310"
cmd.ExecuteNonQuery()
```

Una vez que su aplicación ha efectuado una transacción de base de datos, debe retrotraerla o confirmarla. Esto se realiza mediante los métodos `Commit()` y `Rollback()` de un objeto `DB2Transaction`.

Retrotracción o confirmación de una transacción en C#:

```
// se supone la utilización del objeto DB2Transaction conn
trans.Rollback();
...
trans.Commit();
```

Retrotracción o confirmación de una transacción en C#:

```
' se supone la utilización del objeto DB2Transaction conn
trans.Rollback();
...
trans.Commit()
```

Lectura de conjuntos de resultados de una aplicación utilizando DB2 .NET Data Provider

Cuando se utiliza DB2 .NET Data Provider, la lectura de los conjuntos de resultados se realiza mediante un objeto DB2DataReader. Se utiliza el método Read() de DB2DataReader para avanzar hacia la fila siguiente del conjunto de resultados. Para extraer los datos de las columnas del resultado se utilizan los métodos GetString(), GetInt32(), GetDecimal(), y otros métodos existentes para todos los tipos de datos disponibles. El método Close() de DB2DataReader se utiliza para cerrar el objeto DB2DataReader, lo cual debe hacerse siempre al terminar de leer el resultado.

Lectura de un conjunto de resultados en C#:

```
// se supone la utilización de un lector DB2DataReader
Int16 deptnum = 0;
String location="";

// Visualizar los resultados de la consulta
while(reader.Read())
{
  deptnum = reader.GetInt16(0);
  location = reader.GetString(1);
  Console.WriteLine("    " + deptnum + " " + location);
}
reader.Close();
```

Lectura de un conjunto de resultados en Visual Basic .NET:

```
' se supone la utilización de un lector DB2DataReader
Dim deptnum As Int16 = 0
Dim location As String ""

' Visualizar los resultados de la consulta
Do While (reader.Read())
  deptnum = reader.GetInt16(0)
  location = reader.GetString(1)
  Console.WriteLine("    " &deptnum &" " &location)
Loop
reader.Close();
```

Invocación de procedimientos almacenados desde una aplicación utilizando DB2 .NET Data Provider

Cuando utiliza DB2 .NET Data Provider, puede invocar procedimientos almacenados utilizando un objeto DB2Command. El valor por omisión de la propiedad CommandType es CommandType.Text. Este valor es el apropiado para sentencias de SQL y también se puede utilizar para invocar procedimientos almacenados. Pero la invocación de procedimientos almacenados es más fácil si define el valor de CommandType como CommandType.StoredProcedure. En este caso, solo es necesario que especifique el nombre del procedimiento almacenado y los parámetros.

Los ejemplos siguientes muestran cómo invocar un procedimiento almacenado llamado INOUT_PARAM, con la propiedad CommandType establecida en CommandType.StoredProcedure o CommandType.Text.

Invocación de un procedimiento almacenado utilizando CommandType.StoredProcedure como valor de la propiedad CommandType de DB2Command en C#:

```
// se supone la existencia de una conexión DB2Connection
DB2Transaction trans = conn.BeginTransaction();
DB2Command cmd = conn.CreateCommand();
String procName = "INOUT_PARAM";
cmd.Transaction = trans;
cmd.CommandType = CommandType.StoredProcedure;
cmd.CommandText = procName;

// Registro de los parámetros de entrada-salida y parámetros de
salida para DB2Command
...

// Invocación del procedimiento almacenado
Console.WriteLine("  Call stored procedure named " + procName);
cmd.ExecuteNonQuery();
```

Invocación de un procedimiento almacenado utilizando `CommandType.Text` como valor de la propiedad CommandType de `DB2Command` en C#:

```
// se supone la existencia de una conexión DB2Connection
DB2Transaction trans = conn.BeginTransaction();
DB2Command cmd = conn.CreateCommand();
String procName = "INOUT_PARAM";
String procCall = "CALL INOUT_PARAM (?, ?, ?)";
cmd.Transaction = trans;
cmd.CommandType = CommandType.Text;
cmd.CommandText = procCall;

// Registro de los parámetros de entrada-salida y parámetros de
salida para DB2Command
...

// Invocación del procedimiento almacenado
Console.WriteLine("  Call stored procedure named " + procName);
cmd.ExecuteNonQuery();
```

Invocación de un procedimiento almacenado utilizando `CommandType.StoredProcedure` como valor de la propiedad CommandType de `DB2Command` en Visual Basic .NET:

```
' se supone la utilización de un lector DB2DataReader
Dim trans As DB2Transaction = conn.BeginTransaction()
Dim cmd As DB2Command = conn.CreateCommand()
Dim procName As String = "INOUT_PARAM"
cmd.Transaction = trans
cmd.CommandType = CommandType.StoredProcedure
cmd.CommandText = procName

' Registro de los parámetros de entrada-salida y parámetros de salida
para DB2Command
...

' Invocación del procedimiento almacenado
Console.WriteLine("  Call stored procedure named " &procName)
cmd.ExecuteNonQuery()
```

Invocación de un procedimiento almacenado utilizando `CommandType.Text` como valor de la propiedad CommandType de `DB2Command` en Visual Basic .NET:

```
' se supone la utilización de un lector DB2DataReader
Dim trans As DB2Transaction = conn.BeginTransaction()
Dim cmd As DB2Command = conn.CreateCommand()
Dim procName As String = "INOUT_PARAM"
Dim procCall As String = "CALL INOUT_PARAM (?, ?, ?)"
cmd.Transaction = trans
cmd.CommandType = CommandType.Text
cmd.CommandText = procCall
```

```
' Registro de los parámetros de entrada-salida y parámetros de salida
para DB2Command
...

' Invocación del procedimiento almacenado
Console.WriteLine("  Call stored procedure named " &procName)
cmd.ExecuteNonQuery()
```

Capítulo 3. Creación de aplicaciones para DB2 .NET Data Provider

Creación de aplicaciones Visual Basic .NET

DB2 proporciona un archivo de proceso por lotes, bldapp.bat, para compilar y enlazar las aplicaciones Visual Basic .NET de DB2, ubicadas en el directorio sqllib\samples\.NET\vb junto a los programas de ejemplo que pueden crearse con este archivo. El archivo de proceso por lotes toma un parámetro, %1, para el nombre del archivo fuente que debe compilarse (sin la extensión .vb).

Procedimiento:

Para crear el programa `DbAuth` a partir del archivo fuente `DbAuth.vb`, entre:

```
bldapp DbAuth
```

Para asegurarse de que tenga los parámetros que necesitará cuando ejecute el archivo ejecutable, puede especificar distintas combinaciones de parámetros en función del número que haya entrado:

1. Ningún parámetro. Entre sólo el nombre del programa:
   ```
   DbAuth
   ```
2. Un parámetro. Entre el nombre del programa más el alias de la base de datos:
   ```
   DbAuth <alias_base_datos>
   ```
3. Dos parámetros. Entre el nombre del programa más el ID de usuario y la contraseña:
   ```
   DbAuth <ID_usuario> <contraseña>
   ```
4. Tres parámetros. Entre el nombre del programa más el alias de la base de datos, el ID de usuario y la contraseña:
   ```
   DbAuth <alias_base_datos> <ID_usuario> <contraseña>
   ```
5. Cuatro parámetros. Entre el nombre del programa más el nombre del servidor, el número de puerto, el ID de usuario y la contraseña:
   ```
   DbAuth <servidor> <número_puerto> <ID_usuario> <contraseña>
   ```
6. Cinco parámetros. Entre el nombre del programa más el alias de la base de datos, el nombre del servidor, el número de puerto, el ID de usuario y la contraseña:
   ```
   DbAuth <alias_base_datos> <servidor> <número_puerto> <ID_usuario> <ctrsña>
   ```

Para crear y ejecutar el programa de ejemplo de `LCTrans` debe seguir instrucciones más detalladas que se proporcionan en el archivo fuente, `LCTrans.vb`.

Tareas relacionadas:

- “Creación de rutinas CLR (Common Language Runtime) .NET” en la página 80

Información relacionada:

- "Opciones de compilación y enlace para aplicaciones Visual Basic .NET" en la página 17
- "Ejemplos de Visual Basic .NET" en *Temas de ejemplos*

Ejemplos relacionados:

- "bldapp.bat -- Builds Visual Basic .Net applications on Windows"
- "DbAuth.vb -- How to Grant, display and revoke privileges on database"
- "LCTrans.vb -- Demonstrates loosely coupled transactions"

Creación de aplicaciones C# .NET

DB2 proporciona un archivo de proceso por lotes, bldapp.bat, para compilar y vincular las aplicaciones C# .NET de DB2, ubicadas en el directorio sqllib\samples\.NET\cs, junto a los programas de ejemplo que pueden crearse con este archivo. El archivo de proceso por lotes toma un parámetro, %1, para el nombre del archivo fuente que debe compilarse (sin la extensión .cs).

Procedimiento:

Para crear el programa `DbAuth` a partir del archivo fuente `DbAuth.cs`, entre:

```
bldapp DbAuth
```

Para asegurarse de que tenga los parámetros que necesitará cuando ejecute el archivo ejecutable, puede especificar distintas combinaciones de parámetros en función del número que haya entrado:

1. Ningún parámetro. Entre sólo el nombre del programa:
   ```
   DbAuth
   ```
2. Un parámetro. Entre el nombre del programa más el alias de la base de datos:
   ```
   DbAuth <alias_base_datos>
   ```
3. Dos parámetros. Entre el nombre del programa más el ID de usuario y la contraseña:
   ```
   DbAuth <ID_usuario> <contraseña>
   ```
4. Tres parámetros. Entre el nombre del programa más el alias de la base de datos, el ID de usuario y la contraseña:
   ```
   DbAuth <alias_base_datos> <ID_usuario> <contraseña>
   ```
5. Cuatro parámetros. Entre el nombre del programa más el nombre del servidor, el número de puerto, el ID de usuario y la contraseña:
   ```
   DbAuth <servidor> <número_puerto> <ID_usuario> <ctrsña>
   ```
6. Cinco parámetros. Entre el nombre del programa más el alias de la base de datos, el nombre del servidor, el número de puerto, el ID de usuario y la contraseña:
   ```
   DbAuth <alias_base_datos> <servidor> <número_puerto> <ID_usuario> <contraseña>
   ```

Para crear y ejecutar el programa de ejemplo de `LCTrans` debe seguir instrucciones más detalladas que se proporcionan en el archivo fuente, `LCTrans.cs`.

Tareas relacionadas:

- "Creación de rutinas CLR (Common Language Runtime) .NET" en la página 80

Información relacionada:

- "Opciones de compilación y enlace para aplicaciones C# .NET" en la página 19

- "Ejemplos de C#" en *Temas de ejemplos*

Ejemplos relacionados:

- "bldapp.bat -- Builds C# applications on Windows"
- "DbAuth.cs -- How to Grant, display and revoke privileges on database"
- "LCTrans.cs -- Demonstrates loosely coupled transactions (CSNET)"

Opciones de compilación y enlace para aplicaciones Visual Basic .NET

La tabla siguiente muestra las opciones de compilación y enlace que DB2 recomienda para crear aplicaciones de Visual Basic .NET en Windows con el compilador Microsoft Visual Basic .NET, tal como muestra el archivo de proceso por lotes `bldapp.bat`.

Opciones de compilación y enlace para bldapp
Opciones de compilación y enlace para aplicaciones VB .NET autónomas: **%BLDCOMP%** Variable del compilador. El valor por omisión es `vbc`, que es el compilador de Microsoft Visual Basic .NET. **/r:"%DB2PATH%"\bin\%VERSION%IBM.Data.DB2.dll** Referencia a la biblioteca de enlace de datos de DB2 para la versión de infraestructura de .NET que está utilizando. **%VERSION%** Existen dos versiones de la infraestructura de .NET soportadas para las aplicaciones. DB2 tiene una biblioteca de enlace de datos para cada una de ellas en subdirectorios independientes. Para .NET Framework versión 2.0, `%VERSION%` apunta al subdirectorio `netf20\`; para .NET Framework versión 1.1, `%VERSION%` apunta al subdirectorio `netf11\`.

Opciones de compilación y enlace para bldapp

Opciones de compilación y enlace para el programa de ejemplo emparejado débilmente, LCTrans:

%BLDCOMP%
Variable del compilador. El valor por omisión es `vbc`, que es el compilador de Microsoft Visual Basic .NET.

/out:RootCOM.dll
Salida de la biblioteca de enlace de datos `RootCOM`, utilizada por la aplicación `LCTrans`, del archivo fuente `RootCOM.vb`,

/out:SubCOM.dll
Salida de la biblioteca de enlace de datos `SubCOM`, utilizada por la aplicación `LCTrans`, del archivo fuente `SubCOM.vb`,

/target:library %1.cs
Crear la biblioteca de enlace de datos a partir del archivo fuente de entrada (`RootCOM.vb` o `SubCOM.vb`).

/r:System.EnterpriseServices.dll
Referencia a la biblioteca de enlace de datos de Microsoft Windows System Enterprise Services.

/r:"%DB2PATH%"\bin\%VERSION%IBM.Data.DB2.dll
Referencia a la biblioteca de enlace de datos de DB2 para la versión de infraestructura de .NET que está utilizando.

%VERSION%
Existen dos versiones de la infraestructura de .NET soportadas para las aplicaciones. DB2 tiene una biblioteca de enlace de datos para cada una de ellas en subdirectorios independientes. Para .NET Framework versión 2.0, `%VERSION%` apunta al subdirectorio `etf12\`; para .NET Framework versión 1.1, `%VERSION%` apunta al subdirectorio `netf11\`.

/r:System.Data.dll
Referencia a la biblioteca de enlace de datos de Microsoft Windows System Data.

/r:System.dll
Referencia a la biblioteca de enlace de datos de Microsoft Windows System.

/r:System.Xml.dll
Referencia a la biblioteca de enlace de datos de Microsoft Windows System XML (para `SubCOM.vb`).

/r:SubCOM.dll
Referencia a la biblioteca de enlace de datos de SubCOM (para `RootCOM.vb` y `LCTrans.vb`).

/r:RootCOM.dll
Referencia a la biblioteca de enlace de datos de RootCOM (para `LCTrans.vb`).

Consulte la documentación del compilador para conocer otras opciones de compilador.

Tareas relacionadas:

• "Creación de aplicaciones Visual Basic .NET" en la página 15

Ejemplos relacionados:

• "bldapp.bat -- Builds Visual Basic .Net applications on Windows"

Opciones de compilación y enlace para aplicaciones C# .NET

La tabla siguiente muestra las opciones de compilación y enlace que DB2 recomienda para crear aplicaciones de C# en Windows con el compilador Microsoft C#, tal como muestra el archivo de proceso por lotes `bldapp.bat`.

Opciones de compilación y enlace para bldapp
Opciones de compilación y enlace para aplicaciones C# autónomas: **%BLDCOMP%** Variable del compilador. El valor por omisión es `csc`, que es el compilador C# de Microsoft. **/r:"%DB2PATH%"\bin\%VERSION%IBM.Data.DB2.dll** Referencia a la biblioteca de enlace de datos de DB2 para la versión de infraestructura de .NET que está utilizando. **%VERSION%** Existen dos versiones de la infraestructura de .NET soportadas para las aplicaciones. DB2 tiene una biblioteca de enlace de datos para cada una de ellas en subdirectorios independientes. Para .NET Framework versión 2.0, `%VERSION%` apunta al subdirectorio `netf20\`; para .NET Framework versión 1.1, `%VERSION%` apunta al subdirectorio `netf11\`.

Opciones de compilación y enlace para bldapp

Opciones de compilación y enlace para el programa de ejemplo emparejado débilmente, LCTrans:

%BLDCOMP%
Variable del compilador. El valor por omisión es csc, que es el compilador C# de Microsoft.

/out:RootCOM.dll
Salida de la biblioteca de enlace de datos RootCOM, utilizada por la aplicación LCTrans, del archivo fuente RootCOM.cs,

/out:SubCOM.dll
Salida de la biblioteca de enlace de datos SubCOM, utilizada por la aplicación LCTrans, del archivo fuente SubCOM.cs,

/target:library %1.cs
Crear la biblioteca de enlace de datos a partir del archivo fuente de entrada (RootCOM.cs o SubCOM.cs).

/r:System.EnterpriseServices.dll
Referencia a la biblioteca de enlace de datos de Microsoft Windows System Enterprise Services.

/r:"%DB2PATH%"\bin\%VERSION%IBM.Data.DB2.dll
Referencia a la biblioteca de enlace de datos de DB2 para la versión de infraestructura de .NET que está utilizando.

%VERSION%
Existen dos versiones de la infraestructura de .NET soportadas para las aplicaciones. DB2 tiene una biblioteca de enlace de datos para cada una de ellas en subdirectorios independientes. Para .NET Framework versión 2.0, %VERSION% apunta al subdirectorio etf12\; para .NET Framework versión 1.1, %VERSION% apunta al subdirectorio netf11\.

/r:System.Data.dll
Referencia a la biblioteca de enlace de datos de Microsoft Windows System Data.

/r:System.dll
Referencia a la biblioteca de enlace de datos de Microsoft Windows System.

/r:System.Xml.dll
Referencia a la biblioteca de enlace de datos de Microsoft Windows System XML (para SubCOM.cs).

/r:SubCOM.dll
Referencia a la biblioteca de enlace de datos de SubCOM (para RootCOM.cs y LCTrans.cs).

/r:RootCOM.dll
Referencia a la biblioteca de enlace de datos de RootCOM (para LCTrans.cs).

Consulte la documentación del compilador para conocer otras opciones de compilador.

Tareas relacionadas:

• "Creación de aplicaciones C# .NET" en la página 16

Ejemplos relacionados:

• "bldapp.bat -- Builds C# applications on Windows"

Capítulo 4. Desarrollo de rutinas externas

Rutinas externas

Las rutinas externas son rutinas cuya lógica está implementada en una aplicación de lenguaje de programación que reside fuera de la base de datos, en el sistema de archivos del servidor de bases de datos. La asociación de la rutina con la aplicación de código externo está indicada mediante la especificación de la cláusula EXTERNAL en la sentencia CREATE de la rutina.

Puede crear procedimientos externos, funciones externas y métodos externos. Aunque todos ellos se implementan en lenguajes de programación externos, cada tipo funcional de rutina tiene características distintas. Antes de decidir la implementación de una rutina externa, es importante que primero comprenda qué son las rutinas externas, cómo se implementan y se utilizan leyendo el tema ″Visión general de las rutinas externas″. Una vez comprenda esto, podrá obtener más información acerca de las rutinas externas en los temas de los enlaces relacionados para tomar decisiones informadas sobre cuándo y cómo utilizarlas en el entorno de bases de datos.

Conceptos relacionados:

- “Funciones de tabla de OLE DB definidas por el usuario” en *Desarrollo de SQL y rutinas externas*
- “Visión general de las rutinas externas” en la página 23
- “Visión general de las rutinas” en *Desarrollo de SQL y rutinas externas*
- “Soporte para el desarrollo de rutinas externas en lenguajes .NET CLR” en la página 64
- “Soporte para el desarrollo de rutinas externas en C” en *Desarrollo de SQL y rutinas externas*
- “Soporte para el desarrollo de rutinas externas en C++” en *Desarrollo de SQL y rutinas externas*
- “Software soportado de desarrollo de rutinas Java” en *Desarrollo de SQL y rutinas externas*
- “Rutinas de ejecución en el lenguaje común (CLR) .NET” en la página 63
- “Procedimientos COBOL” en *Desarrollo de SQL y rutinas externas*
- “Rutinas DB2GENERAL” en *Desarrollo de SQL y rutinas externas*
- “Funciones de rutinas externas” en la página 24
- “Rutinas Java” en *Desarrollo de SQL y rutinas externas*
- “Diseño de rutinas de automatización de OLE” en *Desarrollo de SQL y rutinas externas*

Tareas relacionadas:

- “Creación de rutinas C y C++” en *Desarrollo de SQL y rutinas externas*
- “Creación de rutinas externas” en la página 60
- “Desarrollo de rutinas” en *Desarrollo de SQL y rutinas externas*

Información relacionada:

- "Soporte para el desarrollo de procedimientos externos en COBOL" en *Desarrollo de SQL y rutinas externas*
- "Sentencia CREATE FUNCTION (Tabla externa)" en *Consulta de SQL, Volumen 2*
- "Sentencia CREATE PROCEDURE (Externo)" en *Consulta de SQL, Volumen 2*

Beneficios del uso de rutinas

Se pueden obtener las ventajas siguientes mediante la utilización de rutinas:

Encapsular la lógica de aplicación que se puede invocar desde una interfaz de SQL
En un entorno con distintas aplicaciones cliente que tengan requisitos comunes, el uso eficaz de las rutinas puede simplificar la reutilización, estandarización y mantenimiento del código. Si es necesario cambiar un aspecto determinado del comportamiento de una aplicación en un entorno en el que se utilizan rutinas, sólo deberá modificarse la rutina afectada que encapsule el comportamiento. Sin una rutina, será necesario modificar la lógica de aplicación de cada aplicación.

Permitir un acceso controlado a otros objetos de base de datos
Las rutinas pueden utilizarse para controlar el acceso a los objetos de base de datos. Puede que un usuario no tenga permiso para emitir generalmente una determinada sentencia de SQL, como por ejemplo CREATE TABLE; sin embargo, se le puede otorgar permiso para invocar rutinas que contengan una o varias implementaciones concretas de la sentencia, simplificando así la gestión de privilegios mediante la encapsulación de los privilegios.

Mejorar el rendimiento de las aplicaciones reduciendo el tráfico de la red
Cuando se ejecutan aplicaciones en un sistema cliente, cada sentencia de SQL se envía por separado desde el sistema cliente al sistema servidor de bases de datos que deba ejecutarse y cada conjunto de resultados se devuelve por separado. Esto puede derivar en niveles elevados de tráfico de red. Si es posible identificar un trabajo que requiera extensa interacción con la base de datos y escasa interacción con el usuario, resulta sensato instalar este trabajo en el servidor, para minimizar la cantidad de tráfico de red y permitir que el trabajo se realice en los servidores de bases de datos más potentes.

Permitir una ejecución de SQL más rápida y eficaz
Como las rutinas son objetos de base de datos, son más eficientes en la transmisión de datos y peticiones de SQL que las aplicaciones de cliente. Por tanto, las sentencias de SQL que se ejecuten en rutinas tendrán un rendimiento mejor que si se ejecutan en aplicaciones cliente. Las rutinas que se crean con la cláusula NOT FENCED se ejecutan en el mismo proceso que el gestor de bases de datos y, por tanto, pueden utilizar la memoria compartida para la comunicación, lo que puede dar como resultado que mejore el rendimiento de la aplicación.

Permitir la interoperatividad de la lógica implementada en distintos lenguajes de programación
Como los distintos programadores pueden implementar lenguajes de programación diferentes y como suele ser aconsejable reutilizar el código siempre que sea posible, las rutinas de DB2 proporcionan soporte a un alto grado de interoperatividad.

- Las aplicaciones cliente en un lenguaje de programación pueden invocar rutinas que estén implementadas en un lenguaje de programación

distinto. Por ejemplo, las aplicaciones cliente en C pueden invocar rutinas de ejecución en el lenguaje común .NET.
- Las rutinas pueden invocar otras rutinas con independencia del tipo de rutina o de la implementación de la rutina. Por ejemplo, un procedimiento de Java puede invocar una función escalar de SQL incorporado.
- Las rutinas creadas en un servidor de bases de datos de un sistema operativo pueden invocarse desde un cliente DB2 que se ejecute en un sistema operativo distinto.

Las ventanas que se han descrito antes son simplemente algunas de las múltiples ventajas asociadas con la utilización de las rutinas. La utilización de las rutinas puede resultar beneficiosa para una serie usuarios, entre los que se encuentran administradores de bases de datos, diseñadores de bases de datos y desarrolladores de aplicaciones de bases de datos. Por este motivo, existen muchas aplicaciones útiles de rutinas que es posible que desee explorar.

Existen varias clases de rutinas dirigidas a determinadas necesidades funcionales, así como varias implementaciones de rutinas. La elección del tipo de rutina y su implementación puede afectar al grado en que se presentan los beneficios anteriores. En general, las rutinas son una forma efectiva de encapsular la lógica a fin de poder ampliar el SQL y mejorar la estructura, el mantenimiento y, potencialmente, el rendimiento de las aplicaciones.

Conceptos relacionados:
- "Funciones de tabla definidas por el usuario" en *Desarrollo de SQL y rutinas externas*
- "Funciones escalares externas" en la página 26
- "Invocación de rutinas" en *Desarrollo de SQL y rutinas externas*

Tareas relacionadas:
- "Depuración de rutinas" en *Desarrollo de SQL y rutinas externas*

Visión general de las rutinas externas

Visión general de las rutinas externas

Las rutinas externas se caracterizan principalmente por el hecho de que su lógica de rutina se implementa en código de lenguaje de programación y no en SQL.

Antes de decidir la implementación de una rutina externa, es importante que comprenda qué son las rutinas externas, cómo se implementan y cómo pueden utilizarse. Los temas sobre los conceptos siguientes lo ayudarán a comprender las rutinas externas para poder tomar decisiones informadas sobre cuándo y cómo utilizarlas en el entorno de bases de datos:
- "Funciones de rutinas externas" en la página 24
- Creación de rutinas externas
- Gestión de bibliotecas o clases de rutinas externas
- Lenguajes de programación soportados para el desarrollo de rutinas externas
- Soporte de 32 y 64 bits para rutinas externas
- Estilos de parámetros para rutinas externas
- Restricciones en las rutinas externas

Cuando haya comprendido los conceptos sobre rutinas externas, puede comenzar a:

- "Creación de rutinas externas" en la página 60

Conceptos relacionados:

- "Soporte de 32 bits y 64 bits para rutinas externas" en la página 54
- "Características de las funciones y métodos externos" en la página 25
- "Creación de rutinas externas" en la página 45
- "Funciones de rutinas externas" en la página 24
- "Gestión de bibliotecas y clases de rutinas externas" en la página 49
- "Estilos de parámetros de rutinas externas" en la página 46
- "Rutinas externas" en la página 21
- "Rendimiento de las rutinas con bibliotecas de 32 bits en servidores de bases de datos de 64 bits" en la página 55
- "Restricciones de las rutinas externas" en la página 57
- "Interfaces API y lenguajes de programación soportados para el desarrollo de rutinas externas" en la página 38
- "Soporte para el tipo de datos XML en las rutinas externas" en la página 56

Tareas relacionadas:

- "Creación de rutinas externas" en la página 60

Funciones de rutinas externas

Las rutinas externas proporcionan soporte para las funciones más comunes de rutinas, así como soporte para funciones adicionales no soportadas por rutinas SQL. Las siguientes funciones son exclusivas de las rutinas externas:

Acceso a archivos, datos y aplicaciones que residen fuera de la base de datos
Las rutinas externas pueden acceder y manipular datos o archivos que residan fuera de la propia base de datos. También pueden invocar aplicaciones que residan fuera de la base de datos. Los datos, archivos o aplicaciones podrían residir, por ejemplo, en el sistema de archivos del servidor de base de datos o en la red disponible.

Variedad de opciones de estilos de parámetros de rutinas externas
La implementación de rutinas externas en un lenguaje de programación se puede realizar utilizando varias opciones de estilos de parámetros. Aunque puede haber un estilo de parámetro preferido para un determinado lenguaje de programación, a veces es posible escoger. Algunos estilos de parámetros proporcionan soporte para pasar información adicional sobre propiedades de rutinas y de bases de datos a la rutina y de recibirla de esta en una estructura denominada estructura *dbinfo* que puede resultar útil dentro de la lógica de la rutina.

Conservación de estado entre invocaciones de funciones externas con un área reutilizable
Las funciones externas definidas por el usuario proporcionan soporte para la conservación de estado entre invocaciones de función para un conjunto de valores. Esto se realiza con una estructura denominada *scratchpad*. Esto puede resultar útil tanto para las funciones que devuelven valores agregados como para las funciones que necesitan lógica de configuración inicial, como inicialización de almacenamientos intermedios.

Los tipos de llamada identifican invocaciones de funciones externas individuales

Las funciones externas definidas por el usuario se invocan varias veces para un conjunto de valores. Cada invocación se identifica con un valor de tipo de llamada al que se puede hacer referencia dentro de la lógica de la función. Por ejemplo, hay tipos de llamada especiales para la primera invocación de una función, para las llamadas de captación de datos y para la invocación final. Los tipos de llamada resultan útiles porque se puede asociar lógica específica a un tipo de llamada determinado.

Conceptos relacionados:

- "Soporte de 32 bits y 64 bits para rutinas externas" en la página 54
- "Características de las funciones y métodos externos" en la página 25
- "Rutinas externas" en la página 21
- "Características de procedimientos de SQL" en *Desarrollo de SQL y rutinas externas*
- "Visión general de las rutinas externas" en la página 23
- "Restricciones de las rutinas externas" en la página 57
- "SQL en rutinas externas" en *Desarrollo de SQL y rutinas externas*
- "Soporte para el tipo de datos XML en las rutinas externas" en la página 56

Características de las funciones y métodos externos

Características de las funciones y métodos externos

Las funciones externas y los métodos externos proporcionan soporte para funciones que, en el caso de un determinado conjunto de datos de entrada, se podrían invocar múltiples veces y producir un conjunto de valores de salida.

Para obtener más información sobre las características de las funciones y métodos externos, consulte estos temas:

- "Funciones escalares externas" en la página 26
- "Modelo de proceso de los métodos y las funciones escalares externas" en la página 28
- "Funciones de tablas externas" en la página 29
- "Modelo de proceso para las funciones de tablas externas" en la página 29
- "Modelo de ejecución de funciones de tabla para Java" en la página 31
- "Áreas reutilizables para funciones externas y métodos" en la página 33
- "Áreas reutilizables en sistemas operativos de 32 bits y 64 bits" en la página 37

Estas características son exclusivas para las funciones y métodos externos y no atañen a las funciones SQL ni a los métodos SQL.

Conceptos relacionados:

- "Visión general de las rutinas externas" en la página 23
- "Restricciones de las rutinas externas" en la página 57
- "Funciones escalares externas" en la página 26
- "Funciones de rutinas externas" en la página 24
- "Rutinas externas" en la página 21
- "Modelo de proceso de los métodos y las funciones escalares externas" en la página 28

Funciones escalares externas

Las funciones escalares externas tienen implementada su lógica en un lenguaje de programación externo.

Estas funciones pueden desarrollarse y utilizarse para ampliar el conjunto de funciones de SQL existentes y pueden invocarse del mismo modo que las funciones incorporadas de DB2, como por ejemplo, LENGTH y COUNT. Es decir, se puede hacer referencia a ellas en sentencias de SQL siempre que la expresión sea válida.

La ejecución de funciones escalares externas se lleva a cabo en el servidor de bases de datos, aunque, a diferencia de las funciones escalares de SQL incorporadas o definidas por el usuario, la lógica de las funciones externas puede acceder al sistema de archivos del servidor de bases de datos, realizar llamadas al sistema o acceder a una red.

Las funciones escalares externas pueden leer datos de SQL, pero no pueden modificarlos.

Las funciones escalares externas se pueden invocar repetidamente para una sola referencia de la función y pueden mantener el estado entre estas invocaciones mediante un área reutilizable, que es un almacenamiento intermedio de memoria. Esto puede resultar potente si una función requiere una lógica de configuración inicial, pero costosa. La lógica de configuración se puede realizar en una primera invocación, utilizando el área reutilizable para almacenar algunos valores, cuyo acceso o actualización es posible en invocaciones siguientes de la función escalar.

Características de las funciones escalares externas

- Se puede hacer referencia a ellas como parte de una sentencia de SQL en cualquier parte en que esté soportada una expresión.
- La sentencia de SQL que realiza la invocación puede utilizar directamente la salida de una función escalar.
- Para las funciones escalares externas definidas por el usuario, se puede mantener el estado entre las invocaciones iterativas de la función empleando un área reutilizable.
- Pueden proporcionar una ventaja para el rendimiento cuando se utilizan en predicados, puesto que se ejecutan en el servidor. Si una función se puede aplicar a una fila candidata en el servidor, puede, a menudo, dejar de tomar en consideración la fila antes de transmitirla a la máquina cliente, con lo que se reduce la cantidad de datos que se deben pasar desde el servidor al cliente.

Limitaciones

- No se puede realizar la gestión dentro de una función escalar. Es decir, no se puede emitir COMMIT ni ROLLBACK dentro de una function escalar.
- No pueden devolver conjuntos de resultados.
- Las funciones escalares están pensadas para devolver un solo valor escalar por cada conjunto de valores de entrada.
- Las funciones escalares externas no están pensadas para que se utilicen en una sola invocación. Están diseñadas de forma que, para cada referencia a la función y cada conjunto determinado de valores de entrada, la función se invoque una vez por cada valor de entrada y devuelva un solo valor escalar. En la primera invocación, las funciones

escalares pueden estar diseñadas para realizar algún trabajo de configuración o para almacenar información, cuyo acceso sea posible en invocaciones posteriores. Las funciones escalares de SQL se ajustan mejor a la funcionalidad que requiere una sola invocación.

- En una base de datos de una sola partición, las funciones escalares externas pueden incluir sentencias de SQL. Estas sentencias pueden leer datos de tablas, pero no pueden modificarlos. Si la base de datos tiene más de una partición, no debe haber sentencias de SQL en una función escalar externa. Las funciones escalares de SQL pueden contener sentencias de SQL que lean o modifiquen datos .

Usos frecuentes

- Ampliar el conjunto de funciones incorporadas de DB2.
- Realizar operaciones lógicas dentro de una sentencia de SQL que SQL no puede realizar de forma nativa.
- Encapsular una consulta escalar que se reutilice habitualmente como subconsulta en las sentencias de SQL. Por ejemplo, dado un código postal, buscar en una tabla la ciudad en la que se encuentra el código postal.

Lenguajes soportados

- C
- C++
- Java
- OLE
- Lenguajes de ejecución en el lenguaje común .NET

Notas:

1. Existe una capacidad limitada para crear funciones de agregación. Conocidas también como funciones de columna, estas funciones reciben un conjunto de valores semejantes (una columna de datos) y devuelven una sola respuesta. Una función de agregación definida por el usuario sólo se puede crear si su fuente es una función de agregación incorporada. Por ejemplo, si existe un tipo diferenciado `SHOESIZE` que está definido con el tipo base INTEGER, es posible definir una función, `AVG(SHOESIZE)`, como función de agregación basada en la función de agregación incorporada existente `AVG(INTEGER)`.
2. También puede crear funciones que devuelvan una fila. Éstas se conocen como funciones de fila y sólo se pueden utilizar como funciones de transformación para los tipos estructurados. La salida de una función de salida es una única fila.

Conceptos relacionados:

- "Beneficios del uso de rutinas" en la página 22
- "Características de las funciones y métodos externos" en la página 25
- "Modelo de proceso de los métodos y las funciones escalares externas" en la página 28
- "Funciones de tabla de OLE DB definidas por el usuario" en *Desarrollo de SQL y rutinas externas*
- "Áreas reutilizables para funciones externas y métodos" en la página 33
- "Funciones de tablas externas" en la página 29

Tareas relacionadas:

- "Invocación de funciones o métodos escalares" en *Desarrollo de SQL y rutinas externas*
- "Invocación de funciones de tabla definidas por el usuario" en *Desarrollo de SQL y rutinas externas*

Información relacionada:

- "Sentencia CREATE FUNCTION" en *Consulta de SQL, Volumen 2*

Modelo de proceso de los métodos y las funciones escalares externas

El modelo de proceso para los métodos y las UDF escalares que se definen con la especificación FINAL CALL es el siguiente:

Llamada FIRST
: Éste es un caso especial de la llamada NORMAL, identificado como FIRST para permitir que la función realice cualquier proceso inicial. Se evalúan los argumentos y se pasan a la función. Normalmente, la función devolverá un valor en esta llamada, pero puede devolver un error, en cuyo caso no se realiza ninguna llamada NORMAL ni FINAL. Si se devuelve un error en una llamada FIRST, lo debe borrar el método o la UDF antes de volver, puesto que no se realizará ninguna llamada FINAL.

Llamada NORMAL
: Son las llamadas a la función que van de la segunda a la última, según indiquen los datos y la lógica de la sentencia. Se espera que la función devuelva un valor con cada llamada NORMAL después de que se evalúen y pasen los argumentos. Si una llamada NORMAL devuelve un error, no se realiza ninguna otra llamada NORMAL pero se realiza la llamada FINAL.

Llamada FINAL
: Ésta es una llamada especial, que se realiza en el proceso de fin de sentencia (o en el cierre (CLOSE) de un cursor), siempre y cuando la llamada FIRST sea satisfactoria. En una llamada FINAL no se pasa ningún valor de argumento. Esta llamada se lleva a cabo para que la función pueda borrar los recursos. La función no devuelve un valor en esta llamada, pero puede devolver un error.

Si se trata de métodos o UDF escalares que no se han definido con FINAL CALL, sólo se realizan llamadas NORMAL a la función, la cual habitualmente devuelve un valor para cada llamada. Si una llamada NORMAL devuelve un error, o si la sentencia encuentra otro error, no se realiza ninguna otra llamada a la función.

Nota: Este modelo describe el proceso de errores corriente para los métodos y las UDF escalares. En el caso de una anomalía del sistema o un problema de comunicación, no se puede efectuar una llamada indicada por el modelo de proceso de errores. Por ejemplo, para una UDF FENCED, si el proceso protegido `db2udf` termina de algún modo de forma prematura, DB2 no puede efectuar las llamadas indicadas.

Conceptos relacionados:

- "Características de las funciones y métodos externos" en la página 25
- "Funciones escalares externas" en la página 26

Funciones de tablas externas

Una función de tabla definida por el usuario entrega una tabla al SQL en el que se le hace referencia. Una referencia a una UDF de tabla sólo es válida en una cláusula FROM de una sentencia SELECT. Cuando utilice funciones de tabla, tenga en cuenta lo siguiente:

- Aunque una función de tabla entrega una tabla, la interfaz física entre DB2 y la UDF es de una fila cada vez. Hay cinco tipos de llamadas que se pueden realizar a una función de tabla: OPEN, FETCH, CLOSE, FIRST y FINAL. La existencia de las llamadas FIRST y FINAL depende de cómo se defina la UDF. Para distinguir estas llamadas, se utiliza el mismo mecanismo de *tipo-llamada* que se usa para las funciones escalares.
- No se tienen que devolver todas las columnas de resultado definidas en la cláusula RETURNS de la sentencia CREATE FUNCTION para la función de tabla. La palabra clave DBINFO de CREATE FUNCTION y el argumento *dbinfo* correspondiente permiten una optimización consistente en que sólo haya que devolver las columnas necesarias para una referencia a una función de tabla determinada.
- Los valores de columna individuales devueltos se ajustan al formato de los valores devueltos por las funciones escalares.
- La sentencia CREATE FUNCTION para una función de tabla tiene la especificación CARDINALITY. Esta especificación permite que el creador informe al optimizador de DB2 del tamaño aproximado del resultado, de forma que el optimizador pueda tomar decisiones mejores cuando se haga referencia a la función.

  Independientemente de lo que se haya especificado como CARDINALITY de una función de tabla, tenga cuidado respecto a la escritura de una función con una cardinalidad infinita, es decir, una función que siempre devuelva una fila en una llamada FETCH. Existen muchas situaciones en las que DB2 espera la condición de fin de tabla como catalizador dentro del proceso de la consulta. La utilización de GROUP BY u ORDER BY son ejemplos de este caso. DB2 no puede formar los grupos de agregación hasta que se llega al fin de tabla, ni puede realizar ninguna clasificación sin tener todos los datos. Por lo tanto, una función de tabla que nunca devuelva la condición de fin de tabla (valor '02000' de estado de SQL) puede ocasionar un bucle infinito del proceso si se utiliza con una cláusula GROUP BY u ORDER BY.

Conceptos relacionados:

- "Características de las funciones y métodos externos" en la página 25
- "Modelo de proceso para las funciones de tablas externas" en la página 29

Información relacionada:

- "Sentencia CREATE FUNCTION" en *Consulta de SQL, Volumen 2*
- "Pase de argumentos a rutinas C, C++, OLE o COBOL" en *Desarrollo de SQL y rutinas externas*

Modelo de proceso para las funciones de tablas externas

El modelo de proceso para las UDF de tabla que se definen con la especificación FINAL CALL es el siguiente:

Llamada FIRST

Esta llamada se realiza antes de la primera llamada OPEN y su objetivo consiste en permitir que la función realice cualquier proceso inicial. Antes

de esta llamada, el área reutilizable se borra. Se evalúan los argumentos y se pasan a la función. La función no devuelve una fila. Si la función devuelve un error, no se realiza ninguna otra llamada a la misma.

Llamada OPEN
Esta llamada se realiza para permitir que la función realice un proceso especial de apertura (OPEN) específico de la exploración. El área reutilizable (si existe) no se borra antes de la llamada. Se evalúan los argumentos y se pasan. La función no devuelve una fila en una llamada OPEN. Si la función devuelve un error de la llamada OPEN, no se realizará ninguna llamada FETCH ni CLOSE, pero sí se realizará la llamada FINAL al final de la sentencia.

Llamada FETCH
Se siguen produciendo llamadas FETCH hasta que la función devuelve un valor de SQLSTATE que significa fin-de-tabla. Es en estas llamadas donde la UDF desarrolla y devuelve una fila de datos. Se pueden pasar valores de argumentos a la función, pero éstos apuntan a los mismos valores que se han pasado al ejecutar OPEN. Por lo tanto, es posible que los valores de argumentos no sean actuales y no se debe confiar en ellos. Si tiene necesidad de mantener valores actuales entre las invocaciones de una función de tabla, utilice un área reutilizable. La función puede devolver un error de una llamada FETCH, y la llamada CLOSE se seguirá produciendo.

Llamada CLOSE
Se realiza esta llamada cuando finaliza la exploración o sentencia, siempre que la llamada OPEN haya resultado satisfactoria. Los posibles valores de argumentos no serán actuales. La función puede devolver un error.

Llamada FINAL
Se realiza la llamada FINAL al final de la sentencia, siempre que la llamada FIRST haya resultado satisfactoria. Esta llamada se lleva a cabo para que la función pueda borrar los recursos. La función no devuelve un valor en esta llamada, pero puede devolver un error.

Para las UDF de tabla no definidas con FINAL CALL, sólo se realizan las llamadas OPEN, FETCH y CLOSE a la función. Antes de cada llamada OPEN, se borra el área reutilizable (si existe).

La diferencia entre las UDF de tabla definidas con FINAL CALL y las definidas con NO FINAL CALL se puede observar examinando un escenario que implique una unión o una subconsulta, en que el acceso de la función de tabla es el acceso ″interior″. Por ejemplo, en una sentencia como la siguiente:

```
SELECT x,y,z,... FROM table_1 as A,
  TABLE(table_func_1(A.col1,...)) as B
  WHERE...
```

En este caso, el optimizador abrirá una exploración de table_func_1 para cada fila de table_1. Esto es debido a que el valor de la col1 de table_1, que se pasa a table_func_1, se utiliza para definir la exploración de la función de tabla.

Para las UDF de tabla con NO FINAL CALL, la secuencia de llamadas OPEN, FETCH, FETCH, ..., CLOSE se repite para cada fila de table_1. Observe que cada llamada OPEN obtendrá un área reutilizable limpia. Puesto que la función de tabla no sabe, al final de cada exploración, si se producirán más exploraciones, la tiene que limpiar completamente durante el proceso de cierre (CLOSE). Esto puede resultar ineficaz si existe un proceso significativo de apertura de una sola vez que se debe repetir.

Las UDF de tabla con FINAL CALL proporcionan una llamada FIRST de una sola vez y una llamada FINAL de una sola vez. Estas llamadas se utilizan para amortizar los costes de inicialización y terminación a lo largo de todas las exploraciones de la función de tabla. Al igual que anteriormente, las llamadas OPEN, FETCH, FETCH, ..., CLOSE se realizan para cada fila de la tabla exterior pero, dado que la función de tabla sabe que obtendrá una llamada FINAL, no es necesario que efectúe una limpieza completa en la llamada CLOSE (ni que la reasigne en la OPEN posterior). Observe también que el área reutilizable no se borra entre las exploraciones, en gran parte porque los recursos de la función de tabla se repartirán a lo largo de las exploraciones.

A expensas de gestionar dos tipos de llamadas adicionales, la UDF de tabla puede alcanzar una eficacia mayor en estos escenarios de unión y subconsulta. La decisión de si se debe definir la función de tabla como FINAL CALL depende del modo en que se pretenda utilizar.

Conceptos relacionados:

- "Características de las funciones y métodos externos" en la página 25
- "Modelo de ejecución de funciones de tabla para Java" en la página 31
- "Funciones de tabla definidas por el usuario" en *Desarrollo de SQL y rutinas externas*

Información relacionada:

- "Sentencia CREATE FUNCTION (Tabla externa)" en *Consulta de SQL, Volumen 2*
- "Sentencia CREATE FUNCTION (Tabla externa OLE DB)" en *Consulta de SQL, Volumen 2*
- "Sentencia CREATE FUNCTION (Escalar de SQL, tabla o fila)" en *Consulta de SQL, Volumen 2*

Modelo de ejecución de funciones de tabla para Java

Para las funciones de tabla escritas en Java que utilizan PARAMETER STYLE DB2GENERAL, es importante comprender lo que sucede en cada punto del proceso de DB2 de una sentencia determinada. La tabla siguiente detalla esta información para una función de tabla habitual. Se abarcan los casos de NO FINAL CALL y de FINAL CALL, suponiendo en ambos casos SCRATCHPAD.

Punto en el tiempo de exploración	**NO FINAL CALL LANGUAGE JAVA SCRATCHPAD**	**FINAL CALL LANGUAGE JAVA SCRATCHPAD**
Antes de la primera OPEN para la función de tabla	No hay llamadas.	• Se llama al constructor de clases (por medio de la nueva área reutilizable). Se llama al método de UDF con la primera (FIRST) llamada. • El constructor inicializa las variables de clase y área reutilizable. El método conecta con el servidor de Web.

Punto en el tiempo de exploración	NO FINAL CALL LANGUAGE JAVA SCRATCHPAD	FINAL CALL LANGUAGE JAVA SCRATCHPAD
En cada OPEN de la función de tabla	• Se llama al constructor de clases (por medio de la nueva área reutilizable). Se llama al método de UDF con la llamada OPEN. • El constructor inicializa las variables de clase y área reutilizable. El método conecta con el servidor de Web y abre la exploración de los datos de la Web.	• Se abre el método de UDF con la llamada OPEN. • El método abre la exploración de los datos de la Web que desee. (Puede ser capaz de evitar que se tenga que volver a abrir después de una reposición de CLOSE, según lo que se guarde en el área reutilizable.)
En cada FETCH para una nueva fila de datos de la función de tabla	• Se llama al método de UDF con la llamada FETCH. • El método capta y devuelve la siguiente fila de datos, o bien EOT.	• Se llama al método de UDF con la llamada FETCH. • El método capta y devuelve la nueva fila de datos, o bien EOT.
En cada CLOSE de la función de tabla	• Se llama al método de UDF con la llamada CLOSE. Método `close()`, si existe para la clase. • El método cierra su exploración de la Web y se desconecta del servidor de Web. No es necesario que `close()` haga nada.	• Se llama al método de UDF con la llamada CLOSE. • El método puede reposicionarse al principio de la exploración o cerrarla. Puede guardar cualquier estado en el área reutilizable, el cual persistirá.
Después de la última CLOSE de la función de tabla	No hay llamadas.	• Se llama al método de UDF con la llamada FINAL. Se llama al método `close()`, si existe para la clase. • El método se desconecta del servidor de Web. No es necesario que el método `close()` haga nada.

Notas:

1. El término ″método de UDF″ hace referencia al método de clase de Java que implementa la UDF. Se trata del método identificado en la cláusula EXTERNAL NAME de la sentencia CREATE FUNCTION.
2. Para las funciones de tabla que tengan especificado NO SCRATCHPAD, las llamadas al método de UDF son las indicadas en esta tabla, pero, puesto que el usuario no solicita ninguna continuidad por medio de un área reutilizable, DB2 hará que se cree una instancia de un nuevo objeto antes de cada llamada, llamando al constructor de clases. No está claro que las funciones de tabla con NO SCRATCHPAD (y, por lo tanto, sin continuidad) puedan realizar acciones útiles, pero se soportan.

Conceptos relacionados:

- "Rutinas DB2GENERAL" en *Desarrollo de SQL y rutinas externas*
- "Características de las funciones y métodos externos" en la página 25
- "Modelo de proceso para las funciones de tablas externas" en la página 29
- "Rutinas Java" en *Desarrollo de SQL y rutinas externas*

Tareas relacionadas:

- "Diseño de rutinas Java" en *Desarrollo de SQL y rutinas externas*

Información relacionada:

- "Sentencia CREATE FUNCTION (Tabla externa)" en *Consulta de SQL, Volumen 2*

Áreas reutilizables para funciones externas y métodos

Un *área reutilizable* permite que una función definida por el usuario o un método guarden su estado entre una invocación y la siguiente. Por ejemplo, a continuación se identifican dos situaciones en las que guardar el estado entre las invocaciones es beneficioso:

1. Las funciones o los métodos que, para ser correctos, dependen de que se guarde el estado.

   Un ejemplo de función o método de este tipo es una simple función de contador que devuelve un '1' la primera vez que recibe llamada, e incrementa el resultado en uno en cada llamada sucesiva. En algunas circunstancias, este tipo de función se podría utilizar para numerar las filas del resultado de una sentencia SELECT:

   ```
   SELECT counter(), a, b+c, ...
     FROM tablex
     WHERE ...
   ```

   La función necesita un lugar en el que almacenar el valor actual del contador entre las invocaciones, donde se garantice que el valor será el mismo para la siguiente invocación. Luego, en cada invocación se puede incrementar el valor y devolverlo como resultado de la función.

   Este tipo de rutina es NOT DETERMINISTIC. Su salida no depende únicamente de los valores de sus argumentos de SQL.

2. Las funciones o los métodos en que se puede mejorar el rendimiento mediante la posibilidad de realizar algunas acciones de inicialización.

   Un ejemplo de función o método de este tipo, que puede formar parte de una aplicación de documento, es una función *match* (de coincidencia), que devuelve 'Y' si un documento determinado contiene una serie indicada, y 'N' en caso contrario:

   ```
   SELECT docid, doctitle, docauthor
     FROM docs
     WHERE match('myocardial infarction', docid) = 'Y'
   ```

   Esta sentencia devuelve todos los documentos que contienen el valor de serie de texto concreto representado por el primer argumento. Lo que *match* desearía hacer es:

   - Sólo la primera vez.

     Recuperar una lista de todos los ID de documento que contengan la serie '`myocardial infarction`' desde la aplicación de documento, que se mantiene fuera de DB2. Esta recuperación es un proceso costoso, por lo que la función desearía hacerlo una sola vez, y guardar la lista en algún lugar para tenerla a mano en las llamadas posteriores.

   - En cada llamada.

     Utilizar la lista de los ID de documento guardada durante la primera llamada, para ver si el ID de documento que se pasa como segundo argumento está incluido en la lista.

   Este tipo de rutina es DETERMINISTIC. Su respuesta sólo depende de los valores de los argumentos de entrada. Lo que se muestra aquí es una función cuyo rendimiento, y no su corrección, depende de la posibilidad de guardar información entre una llamada y la siguiente.

Ambas necesidades se satisfacen con la posibilidad de especificar un área reutilizable (SCRATCHPAD) en la sentencia CREATE:

```
CREATE FUNCTION counter()
  RETURNS int ... SCRATCHPAD;

CREATE FUNCTION match(varchar(200), char(15))
  RETURNS char(1) ... SCRATCHPAD 10000;
```

La palabra clave SCRATCHPAD indica a DB2 que asigne y mantenga un área reutilizable para una rutina. El tamaño por omisión de un área reutilizable es de 100 bytes, pero el usuario puede determinar el tamaño (en bytes) correspondiente a un área reutilizable. El ejemplo de *match* presenta 10000 bytes de longitud. DB2 inicializa el área reutilizable con ceros binarios antes de la primera invocación. Si el área reutilizable se define para una función de tabla, y si la función de tabla también se define con NO FINAL CALL (valor por omisión), DB2 renueva el área reutilizable antes de cada llamada a OPEN. Si se especifica la opción de función de tabla FINAL CALL, DB2 no examinará ni cambiará el contenido del área reutilizable después de su inicialización. Para las funciones escalares definidas con áreas reutilizables, DB2 tampoco examina ni cambia el contenido del área después de su inicialización. En cada invocación se pasa a la rutina un puntero al área reutilizable, y DB2 conserva la información del estado de la rutina en el área reutilizable.

Así, para el ejemplo de *counter*, el último valor devuelto se puede conservar en el área reutilizable. Y, en el ejemplo de *match*, se puede conservar en el área reutilizable la lista de documentos, en caso de que el área reutilizable sea suficientemente grande; de lo contrario, se puede asignar memoria para la lista y conservar la dirección de la memoria adquirida en el área reutilizable. Las áreas reutilizables pueden tener una longitud variable: la longitud se define en la sentencia CREATE para la rutina.

El área reutilizable únicamente se aplica a la referencia individual a la rutina en la sentencia. Si en una sentencia existen varias referencias a una rutina, cada referencia tiene su propia área reutilizable, por lo que estas áreas no se pueden emplear para realizar comunicaciones entre referencias. El área reutilizable sólo se aplica a un único agente de DB2 (un agente es una entidad de DB2 que realiza el proceso de todos los aspectos de una sentencia). No existe un ″área reutilizable global″ para coordinar el compartimiento de la información de las áreas reutilizables entre los agentes. Esto es importante, en concreto, para aquellas situaciones en que DB2 establece varios agentes para procesar una sentencia (ya sea en una base de datos de una sola partición o de varias). En estos casos, aunque una sentencia puede contener una sola referencia a una rutina, pueden existir varios agentes que realicen el trabajo y cada uno de ellos tendrá su propia área reutilizable. En una base de datos de varias particiones, donde una sentencia que hace referencia a una UDF ha de procesar datos en varias particiones e invocar la UDF en cada partición, el área reutilizable sólo se aplica a una partición. Como consecuencia, existe un área reutilizable en cada partición en que se ejecuta la UDF.

Si la ejecución correcta de una función depende de que haya una sola área reutilizable por cada referencia a la función, registre la función como DISALLOW PARALLEL. Esto causará que la función se ejecute en una única partición y, de esta manera, se garantizará que exista una única área reutilizable por cada referencia a la función.

Puesto que es reconocido que una UDF o un método pueden requerir recursos del sistema, la UDF o el método se pueden definir con la palabra clave FINAL CALL.

Esta palabra clave indica a DB2 que llame a la UDF o al método durante el proceso de fin de sentencia, para que la UDF o el método puedan liberar sus recursos del sistema. Es vital que una rutina libere cualquier recurso que adquiera; incluso una pequeña fuga se puede convertir en una gran fuga en un entorno en que la sentencia se invoque repetidamente, y una gran fuga puede provocar una detención de DB2 por anomalía grave.

Puesto que el área reutilizable tiene un tamaño fijo, es posible que la UDF o el método incluyan una asignación de memoria y por ello puedan utilizar la llamada final para liberar la memoria. Por ejemplo, la función *match* anterior no puede predecir cuántos documentos coincidirán con la serie de texto indicada. Por lo tanto, la siguiente resultará una definición mejor para *match*:

```
CREATE FUNCTION match(varchar(200), char(15))
  RETURNS char(1) ... SCRATCHPAD 10000 FINAL CALL;
```

Para las UDF o los métodos que emplean un área reutilizable y están referidos en una subconsulta, DB2 puede efectuar una llamada final, si la UDF o el método se especifican así, y renovar el área reutilizable entre invocaciones de la subconsulta. El usuario se puede proteger frente a esta posibilidad, si las UDF o los métodos se utilizan alguna vez en subconsultas, definiendo la UDF o el método con FINAL CALL y utilizando el argumento de tipo de llamada o bien comprobando siempre el estado de *cero binario* del área reutilizable.

Si especifica FINAL CALL, observe que la UDF o el método recibirán una llamada del tipo FIRST. Se puede utilizar esta posibilidad para adquirir e inicializar algún recurso persistente.

A continuación se muestra un ejemplo sencillo de Java de una UDF que utiliza un área reutilizable para calcular la suma de los cuadrados de las entradas de una columna. Este ejemplo toma una columna y devuelve una columna que contiene la suma acumulada de cuadrados desde el principio de la columna hasta la entrada de la fila actual:

```
CREATE FUNCTION SumOfSquares(INTEGER)
RETURNS INTEGER
EXTERNAL NAME 'UDFsrv!SumOfSquares'
DETERMINISTIC
NO EXTERNAL ACTION
FENCED
NOT NULL CALL
LANGUAGE JAVA
PARAMETER STYLE DB2GENERAL
NO SQL
SCRATCHPAD 10
FINAL CALL
DISALLOW PARALLEL
NO DBINFO@


// Suma de cuadrados utilizando la UDF Scratchpad
public void SumOfSquares(int inColumn,
                         int outSum)
throws Exception
{
  int sum = 0;
  byte[] scratchpad = getScratchpad();

  // variables a leer de área SCRATCHPAD
  ByteArrayInputStream byteArrayIn = new ByteArrayInputStream(scratchpad);
  DataInputStream dataIn = new DataInputStream(byteArrayIn);
```

```
  // variables a grabar en área SCRATCHPAD
  byte[] byteArrayCounter;
  int i;
  ByteArrayOutputStream byteArrayOut = new ByteArrayOutputStream(10);
  DataOutputStream dataOut = new DataOutputStream(byteArrayOut);

  switch(getCallType())
  {
    case SQLUDF_FIRST_CALL:
      // inicializar datos
   sum = (inColumn * inColumn);
      // guardar datos en área SCRATCHPAD
      dataOut.writeInt(sum);
      byteArrayCounter = byteArrayOut.toByteArray();
      for(i = 0; i < byteArrayCounter.length; i++)
      {
        scratchpad[i] = byteArrayCounter[i];
      }
      setScratchpad(scratchpad);
   break;
    case SQLUDF_NORMAL_CALL:
      // leer datos de área SCRATCHPAD
      sum = dataIn.readInt();
      // trabajar con datos
      sum = sum + (inColumn * inColumn);
      // guardar datos en área SCRATCHPAD
      dataOut.writeInt(sum);
      byteArrayCounter = byteArrayOut.toByteArray();
      for(i = 0; i < byteArrayCounter.length; i++)
      {
        scratchpad[i] = byteArrayCounter[i];
      }
      setScratchpad(scratchpad);
break;
  }
  // establecer valor de salida
  set(2, sum);
} // UDF SumOfSquares
```

Tenga en cuenta que se trata de una función incorporada de DB2 que realiza la misma función que la UDF SumOfSquares. Se ha elegido este ejemplo para demostrar el uso de un área reutilizable.

Conceptos relacionados:

- "Scratchpad como parámetro de función de C o C++" en *Desarrollo de SQL y rutinas externas*
- "Áreas reutilizables en sistemas operativos de 32 bits y 64 bits" en la página 37
- "Características de las funciones y métodos externos" en la página 25
- "Modelo de proceso de los métodos y las funciones escalares externas" en la página 28

Información relacionada:

- "Sentencia CREATE FUNCTION" en *Consulta de SQL, Volumen 2*
- "Sentencia CREATE METHOD" en *Consulta de SQL, Volumen 2*
- "Sentencia CREATE TYPE (Estructurado)" en *Consulta de SQL, Volumen 2*

Ejemplos relacionados:

- "udfsrv.c -- Library of user-defined functions (UDFs) called by the client"
- "UDFCreate.db2 -- How to catalog the Java UDFs contained in UDFsrv.java "
- "UDFsrv.java -- Provide UDFs to be called by UDFcli.java (JDBC)"

Áreas reutilizables en sistemas operativos de 32 bits y 64 bits

Para hacer que el código de la UDF o del método se pueda transportar entre sistemas operativos de 32 bits y 64 bits, debe ir con precaución en la manera de crear y utilizar las áreas reutilizables que contengan valores de 64 bits. Es recomendable no declarar una variable de longitud explícita para una estructura de área reutilizable que contenga uno o más valores de 64 bits, tales como los punteros de 64 bits o las variables BIGINT `sqlint64`.

A continuación se muestra una declaración de estructura de ejemplo correspondiente a un área reutilizable:

```
struct sql_scratchpad
{
   sqlint32 length;
   char data[100];
};
```

Cuando se define su propia estructura para el área reutilizable, una rutina tiene dos opciones:

1. Redefinir el área reutilizable `sql_scratchpad` entera, en cuyo caso es necesario incluir un campo de longitud explícito. Por ejemplo:

   ```
   struct sql_spad
   {
     sqlint32 length;
     sqlint32 int_var;
     sqlint64 bigint_var;
   };
   void SQL_API_FN routine( ..., struct sql_spad* scratchpad, ... )
   {
     /* Utilizar área reutilizable */
   }
   ```

2. Redefinir solamente la porción de datos del área reutilizable `sql_scratchpad`, en cuyo caso no se necesita ningún campo de longitud.

   ```
   struct spaddata
   {
     sqlint32 int_var;
     sqlint64 bigint_var;
   };
   void SQL_API_FN routine( ..., struct sql_scratchpad* spad, ... )
   {
     struct spaddata* scratchpad = (struct spaddata*)spad->data;
     /* Utilizar área reutilizable */
   }
   ```

Puesto que la aplicación no puede cambiar el valor del campo de longitud del área reutilizable, el hecho de codificar la rutina tal como se muestra en el primer ejemplo no brinda ningún beneficio significativo. El segundo ejemplo también se puede transportar entre sistemas de distintos tamaños de palabra, por lo que representa la manera preferible de escribir la rutina.

Conceptos relacionados:

- "Características de las funciones y métodos externos" en la página 25
- "Áreas reutilizables para funciones externas y métodos" en la página 33
- "Funciones escalares externas" en la página 26
- "Funciones de tabla definidas por el usuario" en *Desarrollo de SQL y rutinas externas*

Tareas relacionadas:

- "Invocación de rutinas de 32 bits en un servidor de bases de datos de 64 bits" en *Desarrollo de SQL y rutinas externas*

API y lenguajes de programación soportados

Interfaces API y lenguajes de programación soportados para el desarrollo de rutinas externas

Puede desarrollar rutinas externas de DB2 (procedimientos y funciones) utilizando las siguientes API y los lenguajes de programación asociados:

- ADO.NET
  - Lenguajes de programación .NET Common Language Runtime
- CLI
- SQL incorporado
  - C
  - C++
  - COBOL (solo para procedimientos)
- JDBC
  - Java
- OLE
  - Visual Basic
  - Visual C++
  - Cualquier otro lenguaje de programación que soporte esta API.
- OLE DB (solo para funciones de tabla)
  - Cualquier lenguaje de programación que soporte esta API.
- SQLJ
  - Java

Conceptos relacionados:

- "Visión general de las rutinas externas" en la página 23
- "Soporte para el desarrollo de rutinas externas en lenguajes .NET CLR" en la página 64
- "Soporte para el desarrollo de rutinas externas en C" en *Desarrollo de SQL y rutinas externas*
- "Soporte para el desarrollo de rutinas externas en C++" en *Desarrollo de SQL y rutinas externas*
- "Software soportado de desarrollo de rutinas Java" en *Desarrollo de SQL y rutinas externas*
- "Elección de una interfaz de programación de aplicaciones" en *Iniciación al desarrollo de aplicaciones de bases de datos*
- "Interfaces soportadas de programación de aplicaciones de bases de datos" en *Iniciación al desarrollo de aplicaciones de bases de datos*

Información relacionada:

- "Soporte para el desarrollo de procedimientos externos en COBOL" en *Desarrollo de SQL y rutinas externas*
- "Lenguajes de programación y compiladores soportados para el desarrollo de aplicaciones de bases de datos" en *Iniciación al desarrollo de aplicaciones de bases de datos*

Comparación de las API y lenguajes de programación soportados para el desarrollo de rutinas externas

Antes de empezar a implementar rutinas externas, es importante tener en cuenta las características y las limitaciones de las distintas interfaces de programación de aplicaciones (API) y lenguajes de programación soportados para las rutinas externas. Con ello se asegurará de que elige la implementación adecuada desde el principio y de que están disponibles las características de rutinas que necesite.

Tabla 1. Comparación de las API y lenguajes de programación de rutinas externas

API y lenguaje de programación	Soporte de características	Rendimiento	Seguridad	Escalabilidad	Limitaciones
SQL (incluye SQL PL)	• SQL es un lenguaje de alto nivel fácil de aprender y de utilizar y que agiliza la implementación. • Los elementos SQL Procedural Language (SQL PL) permiten utilizar la lógica de flujo de control en operaciones y consultas SQL. • Soporte a tipos de datos firmes.	• Muy bueno. • El rendimiento de las rutinas SQL es mejor que el de las rutinas Java. • El rendimiento de las rutinas SQL es tan bueno como el de las rutinas externas de C y C++ creadas con la cláusula NOT FENCED.	• Muy seguro. • Los procedimientos SQL se ejecutan en la misma memoria que el gestor de bases de datos.	• Gran escalabilidad.	• No pueden acceder al sistema de archivos del servidor de bases de datos. • No pueden invocar aplicaciones que residan fuera de la base de datos.

Tabla 1. Comparación de las API y lenguajes de programación de rutinas externas (continuación)

API y lenguaje de programación	Soporte de características	Rendimiento	Seguridad	Escalabilidad	Limitaciones
SQL incorporado (incluye C y C++)	• Lenguaje de programación de bajo nivel, pero potente.	• Muy bueno. • El rendimiento de las rutinas de C y C++ es mejor que el de las rutinas Java. • El rendimiento de las rutinas de C y C++ creadas con la cláusula NOT FENCED es tan bueno como el de las rutinas SQL.	• Las rutinas de C y C++ son susceptibles a los errores de programación. • Los programadores deben tener un buen dominio del lenguaje C para evitar cometer errores de memoria y de manipulación de punteros muy comunes que hacen que la implementación de las rutinas sea más pesada y requiera más tiempo. • Las rutinas de C y C++ deben crearse con la cláusula FENCED y la cláusula NOT THREADSAFE para evitar la interrupción del gestor de bases de datos en caso de que se produzca una excepción en la rutina en tiempo de ejecución. Éstas son las cláusulas por omisión. La utilización de estas cláusulas puede tener un cierto efecto negativo sobre el rendimiento pero garantiza una ejecución segura. Consulte: Seguridad de las rutinas .	• La escalabilidad es limitada cuando se crean rutinas de C y C++ con las cláusulas FENCED y NOT THREADSAFE. Estas rutinas se ejecutan en un proceso *db2fmp* independiente del proceso del gestor de bases de datos. Se necesita un proceso *db2fmp* para cada rutina que se ejecute de forma simultánea.	• Existen múltiples estilos para pasar los parámetros soportados y esto puede resultar confuso. Los usuarios deben utilizar siempre que sea posible el estilo de los parámetros SQL.

Tabla 1. Comparación de las API y lenguajes de programación de rutinas externas (continuación)

API y lenguaje de programación	Soporte de características	Rendimiento	Seguridad	Escalabilidad	Limitaciones
SQL incorporado (COBOL)	• Lenguaje de programación de alto nivel adecuado para el desarrollo de aplicaciones empresariales que suelen ir orientadas a archivos. • Utilizado de forma generalizada en el pasado para las aplicaciones empresariales de producción, aunque su popularidad está decayendo. • COBOL no contiene soporte a los punteros y es un lenguaje de programación iterativo y lineal.	• El rendimiento de las rutinas de COBOL no es tan bueno como el de las rutinas creadas con el resto de opciones de implementación de rutinas externas.	• No hay información disponible en estos momentos.	• No hay información disponible en estos momentos.	• Es posible crear e invocar procedimientos de COBOL de 32 bits en instancias de DB2 de 64 bits; sin embargo, el rendimiento de estas rutinas no será tan bueno como el de los procedimientos de COBOL de 64 bits de una instancia de DB2 de 64 bits.
JDBC (Java) y SQLJ (Java)	• Lenguaje de programación de alto nivel orientado a objetos adecuado para el desarrollo de aplicaciones autónomas, applets y servlets. • Los objetos y los tipos de datos de Java facilitan el establecimiento de conexiones con la base de datos, la ejecución de las sentencias SQL y la manipulación de los datos.	• El rendimiento de las rutinas de Java es tan bueno como el de las rutinas de C y C++ o el de las rutinas SQL.	• Las rutinas de Java son más seguras que las rutinas de C y C++ porque la Máquina virtual Java (JVM) es quien gestiona el control de las operaciones peligrosas. Esto aumenta la fiabilidad y hace que resulte difícil que el código de una rutina de Java perjudique a otra rutina que se ejecute en el mismo proceso.	• Buena escalabilidad • Las rutinas Java creadas con la cláusula FENCED THREADSAFE (el valor por omisión) tienen una buena escalabilidad. Todas las rutinas Java protegidas comparten algunas JVM. Es posible que se utilice más de una JVM en el sistema si la pila de Java de un proceso db2fmp concreto está próxima a su agotamiento.	• Para evitar operaciones potencialmente peligrosas, no se permiten llamadas JNI (Java Native Interface) desde las rutinas Java.

Tabla 1. Comparación de las API y lenguajes de programación de rutinas externas (continuación)

API y lenguaje de programación	Soporte de características	Rendimiento	Seguridad	Escalabilidad	Limitaciones
Lenguajes soportados de ejecución en el lenguaje común (CLR) .NET (incluidos C#, Visual Basic y otros)	• Parte del modelo de código gestionado .NET de Microsoft. • El código fuente se compila en el bytecode de lenguaje intermedio (IL) que se puede interpretar mediante la unidad ejecutable de lenguaje común (CLR) Microsoft .NET Framework. • Los ensamblajes CLR se pueden crear a partir de subensamblajes que se hayan compilado con diferente código fuente de lenguaje de programación .NET, lo que permite a los usuarios reutilizar e integrar módulos de código escritos en varios lenguajes.	• Las rutinas CLR solo se pueden crear con la cláusula FENCED NOT THREADSAFE para minimizar la posibilidad de una interrupción del gestor de bases de datos durante el tiempo de ejecución. Esto puede tener un cierto efecto negativo sobre el rendimiento.	• Las rutinas CLR solo se pueden crear con la cláusula FENCED NOT THREADSAFE. Por lo tanto, son seguras porque se ejecutarán fuera del gestor de bases de datos en un proceso db2fmp aparte.	• No hay información disponible.	• Consulte el tema ″Restricciones de las rutinas CLR .NET″.

Tabla 1. Comparación de las API y lenguajes de programación de rutinas externas (continuación)

API y lenguaje de programación	Soporte de características	Rendimiento	Seguridad	Escalabilidad	Limitaciones
• OLE	• Las rutinas de OLE se pueden implantar en Visual C++, en Visual Basic y en otros lenguajes soportados por OLE.	• La velocidad de las rutinas automatizadas OLE dependerá del lenguaje utilizado para implementarlas. En general, son más lentas que las rutinas C/C++ que no son OLE. • Las rutinas OLE solo se pueden ejecutar en modalidad FENCED NOT THREADSAFE y, por lo tanto, las rutinas automatizadas OLE tienen una buena escalabilidad.	• No hay información disponible.	• No hay información disponible.	• No hay información disponible.

Tabla 1. Comparación de las API y lenguajes de programación de rutinas externas (continuación)

API y lenguaje de programación	Soporte de características	Rendimiento	Seguridad	Escalabilidad	Limitaciones
• OLE DB	• OLE DB se puede utilizar para crear funciones de tabla definidas por el usuario. • Las funciones OLE DB establecen conexión con fuentes de datos externas OLE DB.	• El rendimiento de las funciones OLE DB depende del proveedor de OLE DB; sin embargo, en general las funciones OLE DB ofrecen un mejor rendimiento que las funciones Java equivalente en términos lógicos pero menor que las funciones C, C++ o SQL equivalentes en términos lógicos. No obstante, algunos predicados de la consulta en que se invoca la función se pueden evaluar en el proveedor de OLE DB, por lo que quedará reducido el número de filas que DB2 tenga que procesar, lo que suele dar lugar a un mejor rendimiento.	• No hay información disponible.	• No hay información disponible.	• OLE DB sólo se puede utilizar para crear funciones de tabla definidas por el usuario.

Conceptos relacionados:

- "Interfaces soportadas de programación de aplicaciones de bases de datos" en *Iniciación al desarrollo de aplicaciones de bases de datos*
- "Soporte para el desarrollo de rutinas externas en lenguajes .NET CLR" en la página 64
- "Soporte para el desarrollo de rutinas externas en C" en *Desarrollo de SQL y rutinas externas*
- "Soporte para el desarrollo de rutinas externas en C++" en *Desarrollo de SQL y rutinas externas*
- "Software soportado de desarrollo de rutinas Java" en *Desarrollo de SQL y rutinas externas*

- "Interfaces API y lenguajes de programación soportados para el desarrollo de rutinas externas" en la página 38

Información relacionada:

- "Lenguajes de programación y compiladores soportados para el desarrollo de aplicaciones de bases de datos" en *Iniciación al desarrollo de aplicaciones de bases de datos*
- "Soporte para el desarrollo de procedimientos externos en COBOL" en *Desarrollo de SQL y rutinas externas*

Creación de rutinas externas

Las rutinas externas se crean de manera parecida a cómo se crean rutinas con otras implementaciones. Sin embargo, se necesitan algunos pasos adicionales porque, para implementar la rutina, hay que escribir código fuente, compilarlo y desplegarlo.

Una rutina externa consta de dos partes:

- La sentencia CREATE que define la rutina.
- La biblioteca o clase externa que implementa el cuerpo de la rutina.

Cuando la sentencia CREATE que define una rutina se ejecuta satisfactoriamente, la rutina se crea en la base de datos. La sentencia debe definir como mínimo el nombre de la rutina, la signatura de parámetros que se usará en la implementación de la rutina, y la ubicación de la biblioteca o clase externa construida a partir del código fuente de la implementación de la rutina.

El código de implementación de las rutinas externas debe estar escrito en uno de los lenguajes de programación soportados y luego hay que incorporarlo en un archivo de clase o biblioteca que debe estar instalado en el sistema de archivos del servidor de base de datos.

Una rutina externa no se puede invocar satisfactoriamente hasta que se ha creado en la base de datos y hasta que la biblioteca o clase asociada a la rutina se ha colocado en la ubicación especificada por la cláusula EXTERNAL.

El desarrollo de rutinas externas suele consistir en las siguientes tareas:

- Determinar qué tipo funcional de rutina hay que implementar.
- Elegir uno de los lenguajes de programación soportados para las rutinas externas de cara a su implementación.
- Diseñar la rutina.
- Conectar con una base de datos y crear la rutina en la base de datos.
  - Para ello, se ejecuta una de las sentencias CREATE PROCEDURE, CREATE FUNCTION o CREATE METHOD o se utiliza una herramienta gráfica que automatice este paso.
  - Esta tarea, también conocida como definición o registro de una rutina, se puede realizar en cualquier momento antes de invocar la rutina, excepto en las circunstancias siguientes:
    - Para las rutinas Java que hacen referencia a un archivo o archivos JAR externos, el código y los archivos JAR externos se deben codificar y compilar antes de que la rutina se cree en la base de datos utilizando la sentencia CREATE específica del tipo de rutina.

- Las rutinas que ejecutan sentencias SQL y se refieren directamente a sí mismas se deben crear en la base de datos emitiendo la sentencia CREATE antes de que se precompile y enlace el código externo asociado con la rutina. Esto también es aplicable a las situaciones en que exista un ciclo de referencias; por ejemplo: la Rutina A hace referencia a la Rutina B, la cual hace referencia a la Rutina A.

- Escribir el código de la lógica de la rutina para que se corresponda con la definición de la rutina.
- Construir la rutina y generar un archivo de clase o biblioteca.
  - En el caso de las rutinas de SQL incorporado, incluye: precompilar, compilar y enlazar el código, así como enlazar el paquete de la rutina con la base de datos destino.
  - En el caso de las rutinas de SQL no incorporado, incluye: compilar y enlazar el código.
- Desplegar el archivo de clase o biblioteca en el servidor de base de datos, en la ubicación especificada en la definición de la rutina.
- Otorgar el privilegio EXECUTE sobre la rutina al invocador o invocadores de la rutina (si no son los que han definido la rutina).
- Invocar, probar y depurar la rutina.

Todos los pasos necesarios para crear una rutina externa se pueden realizar mediante el procesador de línea de mandatos de DB2 o mediante una ventana de mandatos de DB2. Hay herramientas que pueden ayudar a automatizar algunos de estos pasos o todos ellos.

Conceptos relacionados:

- "Funciones de rutinas externas" en la página 24
- "Rutinas externas" en la página 21
- "Visión general de las rutinas externas" en la página 23

Tareas relacionadas:

- "Creación de rutinas externas" en la página 60

Estilos de parámetros de rutinas externas

Las implementaciones de rutinas externas se tienen que ajustar a un convenio determinado para el intercambio de valores de parámetros de la rutina. Estos convenios se conocen como *estilos de parámetros*. Se especifica un estilo de parámetros de rutina externa cuando se crea la rutina, especificando la cláusula PARAMETER STYLE. Los estilos de parámetros caracterizan la especificación y el orden en el que se pasarán los valores de los parámetros a la implementación de la rutina externa. también especifican qué pasará si se pasan valores adicionales a la implementación de la rutina externa. Por ejemplo, algunos estilos de parámetros especifican que para cada valor de parámetro de rutina se debe pasar un valor de indicador de nulo adicional por separado a la implementación de la rutina para proporcionar información sobre la capacidad de nulos de los parámetros, lo que de lo contrario no se puede determinar fácilmente con un tipo de datos del lenguaje de programación nativo.

La tabla siguiente proporciona una lista de los estilos de parámetros disponibles, las implementaciones de rutina que dan soporte a cada estilo de parámetros, los tipos de rutinas funcionales que dan soporte a cada estilo de parámetros y una descripción del estilo de parámetros:

Tabla 2. Estilos de parámetros

Estilo de parámetros	Lenguaje soportado	Tipo de rutina soportado	Descripción
SQL [1]	• C/C++ • OLE • Lenguajes de ejecución en el lenguaje común .NET • COBOL [2]	• UDF • procedim. almacenados • métodos	Además de los parámetros que se pasan durante la invocación, se pasan a la rutina los argumentos siguientes por este orden: • Un indicador nulo para cada parámetro o resultado declarado en la sentencia CREATE. • El SQLSTATE que se debe devolver a DB2. • El nombre calificado de la rutina. • El nombre específico de la rutina. • La serie de diagnóstico de SQL que se debe devolver a DB2. Según las opciones especificadas en la sentencia CREATE y el tipo de rutina, se pueden pasar a la rutina los argumentos siguientes por este orden: • Un almacenamiento intermedio para el área reutilizable. • El tipo de llamada de la rutina. • La estructura dbinfo (contiene información acerca de la base de datos).
DB2SQL [1]	• C/C++ • OLE • Lenguajes de ejecución en el lenguaje común .NET • COBOL	• procedim. almacenados	Además de los parámetros que se pasan durante la invocación, se pasan al procedimiento almacenado los argumentos siguientes por este orden: • Un vector que contiene un indicador nulo para cada parámetro de la sentencia CALL. • El SQLSTATE que se debe devolver a DB2. • El nombre calificado del procedimiento almacenado. • El nombre específico del procedimiento almacenado. • La serie de diagnóstico de SQL que se debe devolver a DB2. Si se especifica la cláusula DBINFO en la sentencia CREATE PROCEDURE, se pasa al procedimiento almacenado una estructura dbinfo (contiene información acerca de la base de datos).
JAVA	• Java	• UDF • procedim. almacenados	Las rutinas con PARAMETER STYLE JAVA utilizan un convenio de pase de parámetros que cumple con la especificación del lenguaje Java y las rutinas de SQLJ. Para los procedimientos almacenados, los parámetros INOUT y OUT se pasarán como matrices de entrada única para facilitar la devolución de valores. Además de los parámetros IN, OUT e INOUT, las signaturas de método Java de los procedimientos almacenados incluyen un parámetro del tipo ResultSet[] para cada conjunto de resultados especificado en la cláusula DYNAMIC RESULT SETS de la sentencia CREATE PROCEDURE. Para las UDF y los métodos con PARAMETER STYLE JAVA, no se pasan más argumentos que los especificados en la invocación de la rutina. Las rutinas PARAMETER STYLE JAVA no proporcionan soporte a las cláusulas DBINFO o PROGRAM TYPE. Para las UDF, PARAMETER STYLE JAVA sólo puede especificarse si no se han especificado como parámetros tipos de datos estructurados y no se ha especificado como tipo de retorno ningún tipo de datos estructurado, CLOB, DBCLOB ni BLOB (SQLSTATE 429B8). Además las UDF de PARAMETER STYLE JAVA no proporcionan soporte a las funciones de tabla, los tipos de llamada ni áreas reutilizables.
DB2GENERAL	• Java	• UDF • procedim. almacenados • métodos	Este tipo de rutina utilizará un convenio de pase de parámetros definido para su uso con los métodos Java. A no ser que se desarrollen UDF de tabla o UDF con áreas reutilizables o que se tenga que acceder a la estructura dbinfo, es recomendable utilizar PARAMETER STYLE JAVA. Para las rutinas con PARAMETER STYLE DB2GENERAL, no se pasa ningún argumento adicional aparte de los especificados en la invocación de la rutina.

Tabla 2. Estilos de parámetros (continuación)

Estilo de parámetros	Lenguaje soportado	Tipo de rutina soportado	Descripción
GENERAL	• C/C++ • Lenguajes de ejecución en el lenguaje común .NET • COBOL	• procedim. almacenados	Un procedimiento almacenado con PARAMETER STYLE GENERAL recibe parámetros de la sentencia CALL en la aplicación o rutina que realiza la invocación. Si se especifica la cláusula DBINFO en la sentencia CREATE PROCEDURE, se pasa al procedimiento almacenado una estructura dbinfo (contiene información acerca de la base de datos). GENERAL es el equivalente de los procedimientos almacenados SIMPLE en DB2 Universal Database para z/OS y OS/390.
GENERAL WITH NULLS	• C/C++ • Lenguajes de ejecución en el lenguaje común .NET • COBOL	• procedim. almacenados	Un procedimiento almacenado con PARAMETER STYLE GENERAL WITH NULLS recibe parámetros de la sentencia CALL en la aplicación o rutina que realiza la invocación. También se incluye un vector que contiene un indicador nulo para cada parámetro de la sentencia CALL. Si se especifica la cláusula DBINFO en la sentencia CREATE PROCEDURE, se pasa al procedimiento almacenado una estructura dbinfo (contiene información acerca de la base de datos). GENERAL WITH NULLS es el equivalente de los procedimientos almacenados SIMPLE WITH NULLS en DB2 Universal Database para z/OS y OS/390.

Nota:

1. Para las UDF y los métodos, PARAMETER STYLE SQL es equivalente a PARAMETER STYLE DB2SQL.
2. COBOL sólo se puede utilizar para desarrollar procedimientos almacenados.
3. Los métodos de ejecución en el lenguaje común .NET no están soportados.

Conceptos relacionados:

- "Rutinas DB2GENERAL" en *Desarrollo de SQL y rutinas externas*
- "Rutinas Java" en *Desarrollo de SQL y rutinas externas*
- "Visión general de las rutinas externas" en la página 23
- "Parámetros de las rutinas de CLR .NET" en *Desarrollo de SQL y rutinas externas*
- "Parámetros en rutinas C y C++" en *Desarrollo de SQL y rutinas externas*
- "Parámetros en rutinas Java" en *Desarrollo de SQL y rutinas externas*
- "Parámetros de los procedimientos de SQL" en *Desarrollo de SQL y rutinas externas*

Tareas relacionadas:

- "Tipos diferenciados como UDF o parámetros de método" en *Desarrollo de SQL y rutinas externas*
- "Pase de parámetros de tipo estructurado a rutinas externas" en *Desarrollo de SQL y rutinas externas*

Información relacionada:

- "Sentencia CREATE FUNCTION" en *Consulta de SQL, Volumen 2*
- "Sentencia CREATE METHOD" en *Consulta de SQL, Volumen 2*
- "Sentencia CREATE PROCEDURE" en *Consulta de SQL, Volumen 2*
- "Sentencia CREATE TYPE (Estructurado)" en *Consulta de SQL, Volumen 2*
- "Pase de argumentos a rutinas C, C++, OLE o COBOL" en *Desarrollo de SQL y rutinas externas*

Gestión de bibliotecas o clases de rutinas

Gestión de bibliotecas y clases de rutinas externas

Para desarrollar e invocar rutinas externas satisfactoriamente, los archivos de bibliotecas y de clases de rutinas externas deben desplegarse y gestionarse debidamente.

La gestión de archivos de bibliotecas y de clases de rutinas externas puede ser mínima si se tiene cuidado cuando se crean las rutinas externas por primera vez y se despliegan archivos de bibliotecas y clases de rutinas externas.

Las principales consideraciones a tener en cuenta sobre la gestión de rutinas externas son las siguientes:

- Despliegue de archivos de bibliotecas y de clases de rutinas externas
- Seguridad de archivos de bibliotecas y de clases de rutinas externas
- Resolución de bibliotecas y clases de rutinas externas
- Modificaciones de archivos de bibliotecas y de clases de rutinas externas
- Copia de seguridad y restauración de archivos de bibliotecas y de clases de rutinas externas

Los administradores del sistema, los administradores de bases de datos y los desarrolladores de aplicaciones de bases de datos deben responsabilizarse de garantizar que los archivos de bibliotecas y de clases de las rutinas externas están protegidos y correctamente conservados durante las tareas de desarrollo de rutinas y de administración de bases de datos.

Conceptos relacionados:

- "Copia de seguridad y restauración de archivos de bibliotecas y clases de rutinas externas" en la página 53
- "Despliegue de bibliotecas y clases de rutinas externas" en la página 49
- "Visión general de las rutinas externas" en la página 23
- "Resolución de bibliotecas y clases de rutinas externas" en la página 51
- "Restricciones de las rutinas externas" en la página 57
- "Seguridad de archivos de bibliotecas o clases de rutinas externas" en la página 51
- "Modificaciones de archivos de bibliotecas y clases de rutinas externas" en la página 52

Información relacionada:

- "Sentencia ALTER FUNCTION" en *Consulta de SQL, Volumen 2*
- "Sentencia ALTER METHOD" en *Consulta de SQL, Volumen 2*
- "Sentencia ALTER PROCEDURE" en *Consulta de SQL, Volumen 2*
- "Sentencia CREATE FUNCTION" en *Consulta de SQL, Volumen 2*
- "Sentencia CREATE METHOD" en *Consulta de SQL, Volumen 2*
- "Sentencia CREATE PROCEDURE" en *Consulta de SQL, Volumen 2*

Despliegue de bibliotecas y clases de rutinas externas

El despliegue de bibliotecas y clases de rutinas externas consiste en copiar las bibliotecas y clases de las rutinas externas en el servidor de bases de datos cuando se han creado a partir del código fuente.

Los archivos de biblioteca, clase o ensamblaje de la rutina externa se tienen que copiar en el *directorio function* de DB2 o en un subdirectorio de este directorio en el servidor de bases de datos. Esta es la ubicación recomendada para el despliegue de rutinas externas. Para obtener más información sobre el directorio de función, consulte la descripción de la cláusula EXTERNAL correspondiente a cualquiera de las siguientes sentencias SQL: CREATE PROCEDURE o CREATE FUNCTION.

Puede copiar la clase, biblioteca o ensamblaje de la rutina externa en otras ubicaciones de directorios del servidor, en función de la API y el lenguaje de programación utilizados para implementar la rutina, aunque no se recomienda hacerlo en general. Si se hace, para invocar satisfactoriamente la rutina debe anotar el nombre de la vía de acceso calificada al completo y asegurarse de que se utilice este valor con la cláusula EXTERNAL NAME.

Los archivos de biblioteca y de clase se pueden copiar en el sistema de archivos del servidor de bases de datos mediante las herramientas de transferencia de archivos disponibles de manera más general. Las rutinas Java fr un sistema informático en el que esté instalado un cliente DB2 se pueden copiar en un servidor de bases de datos DB2 mediante procedimientos especiales definidos por el sistema diseñados específicamente para ello. Encontrará más detalles en los temas sobre rutinas Java.

Cuando ejecute la sentencia CREATE del lenguaje SQL adecuado correspondiente al tipo de rutina, CREATE PROCEDURE o CREATE FUNCTION, asegúrese de especificar las cláusulas adecuadas, prestando especial atención en la cláusula EXTERNAL NAME.

- Especifique la cláusula `LANGUAGE` con el valor adecuado correspondiente a la API o lenguaje de programación elegido. Ejemplos: CLR, C, JAVA.
- Especifique la cláusula `PARAMETER STYLE` con el nombre del estilo de parámetros soportado que se ha implementado en el código de la rutina.
- Especifique la cláusula `EXTERNAL` con el nombre del archivo de biblioteca, clase o ensamblaje que se ha de asociar a la rutina utilizando uno de los valores siguientes:
  - el nombre de vía de acceso calificado al completo del archivo de biblioteca, clase o ensamblaje de la rutina.
  - el nombre de vía de acceso relativa del archivo de biblioteca, clase o ensamblaje de la rutina con relación al directorio de función.

Por omisión, DB2 buscará el archivo de biblioteca, clase o ensamblaje por el nombre en el directorio de función, a menos que se especifique un nombre de vía de acceso completamente calificado o relativo para el archivo en la cláusula EXTERNAL.

Conceptos relacionados:

- "Gestión y rendimiento de bibliotecas de rutinas externas" en la página 53
- "Modificaciones de archivos de bibliotecas y clases de rutinas externas" en la página 52
- "Resolución de bibliotecas y clases de rutinas externas" en la página 51
- "Seguridad de archivos de bibliotecas o clases de rutinas externas" en la página 51
- "Copia de seguridad y restauración de archivos de bibliotecas y clases de rutinas externas" en la página 53
- "Gestión de bibliotecas y clases de rutinas externas" en la página 49

Seguridad de archivos de bibliotecas o clases de rutinas externas

Las bibliotecas de las rutinas externas se almacenan en el sistema de archivos en el servidor de bases de datos, y el gestor de bases de datos de DB2 no hace copia de ellas ni las protege de ningún modo. Para que se pueda seguir invocando satisfactoriamente las rutinas, es imprescindible que la biblioteca asociada a la rutina continúe existiendo en la ubicación especificada en la cláusula EXTERNAL de la sentencia CREATE utilizada para crear la rutina. No mueva ni suprima bibliotecas de rutinas después de crear las rutinas; de lo contrario, las invocaciones de las rutinas fallarían.

Para evitar que las bibliotecas de rutinas se supriman o sustituyan de forma accidental o intencionada, debe restringir el acceso a los directorios del servidor de bases de datos que contienen bibliotecas de rutinas y restringir el acceso a los archivos de bibliotecas de rutinas. Esto se puede realizar utilizando mandatos del sistema operativo para establecer permisos sobre directorios y archivos.

Conceptos relacionados:

- "Copia de seguridad y restauración de archivos de bibliotecas y clases de rutinas externas" en la página 53
- "Gestión de bibliotecas y clases de rutinas externas" en la página 49
- "Gestión y rendimiento de bibliotecas de rutinas externas" en la página 53
- "Modificaciones de archivos de bibliotecas y clases de rutinas externas" en la página 52

Resolución de bibliotecas y clases de rutinas externas

La resolución de bibliotecas de rutinas externas de DB2 se realiza a nivel de instancia de DB2. Esto significa que, en instancias de DB2 que contienen varias bases de datos de DB2, se pueden crear rutinas externas en una base de datos que utilicen bibliotecas de rutinas externas que ya se utilicen para una rutina en otra base de datos.

La resolución de rutinas externas a nivel de instancia da soporte a la reutilización de código, permitiendo asociar varias definiciones de rutina a una sola biblioteca. Cuando las bibliotecas de rutinas externas no se reutilizan de esta forma, sino que existen copias de la biblioteca de la rutina externa en el sistema de archivos del servidor de bases de datos, pueden producirse conflictos de nombres de bibliotecas. Esto puede suceder concretamente cuando hay varias bases de datos en una sola instancia y las rutinas de cada base de datos se asocian a sus propias copias de bibliotecas y clases de cuerpos de rutinas. Se produce un conflicto cuando el nombre de una biblioteca o de una clase utilizado por una rutina de una base de datos es idéntico al nombre de la biblioteca o clase utilizado por una rutina de otra base de datos (de la misma instancia).

Para minimizar la probabilidad de que esto suceda, se recomienda almacenar una sola copia de una biblioteca de rutina en el directorio de función de nivel de instancia (directorio sqllib/function) y que la cláusula EXTERNAL de todas las definiciones de rutinas de cada una de las bases de datos haga referencia a la biblioteca exclusiva.

Si se tienen que crear dos bibliotecas de rutinas funcionalmente diferentes con el mismo nombre, es importante realizar pasos adicionales para minimizar la probabilidad de que se produzcan conflictos de nombres de bibliotecas.

Para rutinas C, C++, COBOL y ADO.NET:

Los conflictos de nombres de bibliotecas se pueden minimizar o resolver:

1. Almacenando las bibliotecas con los cuerpos de las rutinas en distintos directorios para cada base de datos.
2. Creando las rutinas con un valor de cláusula EXTERNAL NAME que especifique la vía de acceso completa de una determinada biblioteca (en lugar de una vía de acceso relativa).

Para rutinas Java:

Los conflictos de nombres de clase no se pueden resolver moviendo los archivos de clase en cuestión a directorios diferentes, porque la variable de entorno CLASSPATH tiene efecto en toda la instancia. La primera clase que se encuentra en CLASSPATH es la que se utiliza. Por lo tanto, si tiene dos rutinas Java diferentes que hacen referencia a una clase con el mismo nombre, una de las rutinas utilizará una clase incorrecta. Existen dos soluciones posibles: renombre las clases afectadas o cree una instancia distinta para cada base de datos.

Conceptos relacionados:

- "Copia de seguridad y restauración de archivos de bibliotecas y clases de rutinas externas" en la página 53
- "Gestión de bibliotecas y clases de rutinas externas" en la página 49
- "Gestión y rendimiento de bibliotecas de rutinas externas" en la página 53
- "Modificaciones de archivos de bibliotecas y clases de rutinas externas" en la página 52
- "Seguridad de archivos de bibliotecas o clases de rutinas externas" en la página 51

Modificaciones de archivos de bibliotecas y clases de rutinas externas

Puede ser necesario modificar la lógica de una rutina externa existente después de que una rutina externa se despliegue y se utilice en un entorno del sistema de bases de datos de producción. Se pueden realizar modificaciones en las rutinas existentes, pero hay que realizarlas cuidadosamente para definir un momento de toma de control claro para las actualizaciones y para minimizar el riesgo de que se produzcan interrupciones en cualquier invocación concurrente de la rutina.

Si una biblioteca de rutina externa necesita una actualización, no vuelva a compilar y a enlazar la rutina con el mismo archivo de destino (por ejemplo, `sqllib/function/foo.a`) que utiliza la rutina actual mientras se está ejecutando el gestor de bases de datos. Si una invocación de rutina actual accede a una versión en antememoria del proceso de la rutina y se sustituye la biblioteca subyacente, la invocación de la rutina puede fallar. Si hay que modificar el cuerpo de una rutina sin detener y volver a iniciar DB2, siga los pasos siguientes:

1. Cree la nueva biblioteca de rutina externa con otro nombre de archivo de biblioteca o de clase.
2. Si se trata de una rutina de SQL incorporado, enlace el paquete de la rutina a la base de datos con el mandato BIND.
3. Utilice la sentencia ALTER ROUTINE para cambiar la definición de la rutina para que la cláusula EXTERNAL NAME haga referencia a la biblioteca o clase de la rutina actualizada. Si el cuerpo de rutina que se debe actualizar lo utilizan rutinas catalogadas en varias bases de datos, las acciones recomendadas en esta sección se deberán llevar a cabo para cada base de datos afectada.

4. Para actualizar rutinas Java incorporadas en archivos JAR, tiene que emitir una sentencia CALL SQLJ.REFRESH_CLASSES() a fin de forzar que DB2 cargue las nuevas clases. Si no emite la sentencia CALL SQLJ.REFRESH_CLASSES() después de actualizar las clases de rutinas Java, DB2 continúa utilizando las versiones anteriores de las clases. DB2 renueva las clases cuando se produce una operación COMMIT o ROLLBACK.

Una vez actualizada la definición de la rutina, las siguientes invocaciones de la rutina cargarán y ejecutarán la nueva biblioteca o clase de rutina externa.

Conceptos relacionados:

- "Copia de seguridad y restauración de archivos de bibliotecas y clases de rutinas externas" en la página 53
- "Gestión de bibliotecas y clases de rutinas externas" en la página 49
- "Gestión y rendimiento de bibliotecas de rutinas externas" en la página 53
- "Seguridad de archivos de bibliotecas o clases de rutinas externas" en la página 51

Copia de seguridad y restauración de archivos de bibliotecas y clases de rutinas externas

No se hace copia de seguridad de las bibliotecas de rutinas externas con otros objetos de base de datos cuando se realiza una copia de seguridad de la base de datos. Tampoco se restauran cuando se restaura una base de datos.

Si el objetivo de una operación de copia de seguridad y restauración de una base de datos es redesplegar una base de datos, los archivos de bibliotecas de rutinas externas del sistema de archivos del servidor de bases de datos original se tienen que copiar en el sistema de archivos del servidor de bases de datos de destino, de forma que se conserven los nombres de vías de acceso relativas de las bibliotecas de rutinas externas.

Conceptos relacionados:

- "Gestión de bibliotecas y clases de rutinas externas" en la página 49
- "Gestión y rendimiento de bibliotecas de rutinas externas" en la página 53
- "Modificaciones de archivos de bibliotecas y clases de rutinas externas" en la página 52
- "Seguridad de archivos de bibliotecas o clases de rutinas externas" en la página 51

Gestión y rendimiento de bibliotecas de rutinas externas

La gestión de bibliotecas de rutinas externas puede afectar al rendimiento de las rutinas, puesto que el gestor de bases de datos de DB2 coloca en antememoria de forma dinámica las bibliotecas de rutinas externas a fin de mejorar el rendimiento de acuerdo con el uso de las rutinas. para conseguir un rendimiento óptimo de las rutinas externas, tenga en cuenta lo siguiente:

- Haga que el número de rutinas de cada biblioteca sea lo más reducido posible. Es mejor tener muchas bibliotecas de rutinas externas pequeñas que unas pocas de gran tamaño.
- Agrupe dentro del código fuente las funciones de las rutinas que se suelen invocar juntas. Cuando el código se compila en una biblioteca de rutinas externas, los puntos de entrada de las rutinas que se invocan con frecuencia estarán juntos, lo que permite al gestor de bases de datos ofrecer un mejor

soporte de colocación en antememoria. El soporte mejorado de colocación en antememoria se debe a la eficacia que se puede obtener al cargar una sola biblioteca de rutinas externas una vez y luego invocar varias funciones de rutinas externas dentro de dicha biblioteca.

Para las rutinas externas implementadas en el lenguaje de programación C o C++, el coste de cargar una biblioteca se paga una vez para las bibliotecas que se encuentran en uso de forma coherente por parte de las rutinas C. Después de la primera invocación de la rutina, no es necesario que todas las invocaciones posteriores, de la misma hebra del proceso, vuelvan a cargar la biblioteca de la rutina.

Conceptos relacionados:

- "Copia de seguridad y restauración de archivos de bibliotecas y clases de rutinas externas" en la página 53
- "Gestión de bibliotecas y clases de rutinas externas" en la página 49
- "Modificaciones de archivos de bibliotecas y clases de rutinas externas" en la página 52

Tareas relacionadas:

- "Sustitución de una biblioteca compartida de AIX" en *Desarrollo de SQL y rutinas externas*

Soporte de 32 bits y 64 bits para rutinas externas

El soporte para rutinas externas de 32 bits y 64 bits está determinado por la especificación de una de las dos cláusulas siguientes en la sentencia CREATE para las rutinas: cláusula FENCED o cláusula NOT FENCED.

El cuerpo de la rutina de una rutina externa se escribe en un lenguaje de programación y se compila en una biblioteca o archivo de clase que luego se carga y se ejecuta cuando se invoca la rutina. La especificación de la cláusula FENCED o NOT FENCED determina si la rutina externa se ejecuta en un entorno con barrera diferenciado del gestor de bases de datos o en el mismo espacio de direcciones que el gestor de bases de datos, lo que puede ofrecer un mejor rendimiento mediante el uso de memoria compartida en vez de TCPIP para las comunicaciones. Por omisión, las rutinas siempre se crean con barrera independientemente de las otras cláusulas seleccionadas.

La tabla siguiente ilustra el soporte de DB2 para ejecutar rutinas de 32 bits y 64 bits con y sin barrera en servidores de base de datos de 32 bits y 64 bits que ejecutan el mismo sistema operativo.

Tabla 3. Soporte para rutinas externas de 32 y de 64 bits

Ancho de bits de la rutina	Servidor de 32 bits	Servidor de 64 bits
Procedimiento con barrera de 32 bits o UDF	Soportado	Soportado
Procedimiento con barrera de 64 bits o UDF	No soportado (4)	Soportado
Procedimiento sin barrera de 32 bits o UDF	Soportado	Soportado (2)
Procedimiento sin barrera de 64 bits o UDF	No soportado (4)	Soportado

Las notas a pie de página de la tabla anterior corresponden a:

- (1) Ejecutar una rutina de 32 bits en un servidor de 64 bits no es tan rápido como ejecutar una rutina de 64 bits en un servidor de 64 bits.
- (2) Las rutinas de 32 bits deben crearse como FENCED y NOT THREADSAFE para funcionar en un servidor de 64 bits.
- (3) No es posible invocar rutinas de 32 bits en servidores de base de datos de 64 bits Linux IA.
- (4) Las aplicaciones y las rutinas de 64 bits y no pueden ejecutarse en espacios de direcciones de 32 bits.

Lo importante a tener en cuenta en la tabla es que los procedimientos sin barrera de 32 bits no pueden ejecutarse en un servidor DB2 de 64 bits. Si debe desplegar rutinas sin barrera de 32 bits en plataformas de 64 bits, elimine la cláusula NOT FENCED de las sentencias CREATE para estas rutinas antes de catalogarlas.

Conceptos relacionados:

- "Gestión de bibliotecas y clases de rutinas externas" en la página 49
- "Rutinas externas" en la página 21
- "Visión general de las rutinas externas" en la página 23
- "Creación de rutinas externas" en la página 45
- "Funciones de rutinas externas" en la página 24
- "Implementación de las rutinas externas" en *Desarrollo de SQL y rutinas externas*

Tareas relacionadas:

- "Creación de rutinas externas" en la página 60

Rendimiento de las rutinas con bibliotecas de 32 bits en servidores de bases de datos de 64 bits

Es posible invocar rutinas con bibliotecas de rutinas de 32 bits en servidores de bases de datos de DB2 de 64 bits. Sin embargo, su rendimiento no es tan bueno como el de invocar una rutina de 64 bits en un servidor de 64 bits. La razón de la disminución del rendimiento es que, antes de cada intento de ejecutar una rutina de 32 bits en un servidor de 64 bits, primero se intenta invocarla como una biblioteca de 64 bits. Si esto falla, entonces la biblioteca se invocará como una biblioteca de 32 bits. Un intento fallido de invocar una biblioteca de 32 bits como si fuera una biblioteca de 64 bits genera un mensaje de error (SQLCODE -444) en el archivo db2diag.log.

Las clases de Java son independientes del número de bits. Solamente las máquinas virtuales de Java (JVM) se clasifican como de 32 bits o de 64 bits. DB2 solo da soporte al uso de JVM que tengan el mismo número de bits que la instancia en que se están utilizando. Es decir, en una instancia de DB2 de 32 bits solamente puede utilizarse una JVM de 32 bits y en una instancia de DB2 de 64 bits solamente puede utilizarse una JVM de 64 bits. Esto garantiza el correcto funcionamiento de las rutinas de Java y el mejor rendimiento posible.

Conceptos relacionados:

- "Visión general de las rutinas externas" en la página 23

Soporte para el tipo de datos XML en las rutinas externas

Los procedimientos y las funciones externas escritas en los siguientes lenguajes de programación soportan parámetros y variables de tipo de datos XML:

- C
- C++
- COBOL
- Java
- Lenguajes .NET CLR

Las rutinas OLE y OLEDB externas no soportan parámetros de tipo de datos XML.

Los valores de tipo de datos XML se representan en el código de las rutinas externas de la misma manera que los tipos de datos CLOB.

Cuando se declaran parámetros de tipo de datos XML para las rutinas externas, las sentencias CREATE PROCEDURE y CREATE FUNCTION que se utilizarán para crear las rutinas en la base de datos deben especificar que el tipo de datos XML se debe almacenar en forma de tipo de datos CLOB. El tamaño del valor de CLOB debe acercarse al tamaño del documento XML representado por el parámetro XML.

En la siguiente línea de código aparece una sentencia CREATE PROCEDURE de un procedimiento externo implementado en el lenguaje de programación C con un parámetro XML llamado `parm1`:

```
CREATE PROCEDURE myproc(IN parm1 XML AS CLOB(2M), IN parm2 VARCHAR(32000))
LANGUAGE C
FENCED
PARAMETER STYLE SQL
EXTERNAL NAME 'mylib!myproc';
```

Hay que tener en cuenta consideraciones parecidas al crear funciones definidas por usuario (UDF) externas, como se ve en el siguiente ejemplo:

```
CREATE FUNCTION myfunc (IN parm1 XML AS CLOB(2M))
RETURNS SMALLINT
LANGUAGE C
PARAMETER STYLE SQL
DETERMINISTIC
NOT FENCED
NULL CALL
NO SQL
NO EXTERNAL ACTION
EXTERNAL NAME 'mylib1!myfunc'
```

Los valores XML se serializan antes de pasarlos a rutinas externas.

En el código de las rutinas externas, los valores de los parámetros y variables XML se acceden, establecen y modifican de la misma manera que en las aplicaciones de bases de datos.

Conceptos relacionados:

- "Visión general de las rutinas externas" en la página 23
- "Encoding considerations for passing XML data in routine parameters" en *XML Guide*

- "Parámetros y variables del tipo de datos XML en funciones de SQL" en *Desarrollo de SQL y rutinas externas*
- "Soporte de XML y XQuery en procedimientos SQL" en *Desarrollo de SQL y rutinas externas*

Tareas relacionadas:

- "Especificación de un tipo de columna de datos" en *Desarrollo de SQL y rutinas externas*

Información relacionada:

- "Tipos de datos de SQL soportados para el Proveedor de datos DB2 .NET" en la página 66
- "Tipos de datos de SQL soportados en rutinas DB2GENERAL" en *Desarrollo de SQL y rutinas externas*
- "Tipos de datos SQL soportados en rutinas Java" en *Desarrollo de SQL y rutinas externas*

Restricciones de las rutinas externas

Se aplican las siguientes restricciones a las rutinas externas y deben tenerse en cuenta a la hora de desarrollar o depurar rutinas externas:

Restricciones que se aplican a todas las rutinas externas::

- En las rutinas externas no se pueden crear hebras nuevas.
- Las API de nivel de conexión no se pueden llamar desde dentro de funciones o métodos externos.
- Desde rutinas externas no es posible recibir entradas desde el teclado ni visualizar salidas en la salida estándar. No utilice corrientes de entrada-salida estándar. Por ejemplo:
  - En código de rutinas Java externas, no emita los métodos `System.out.println()`.
  - En código de rutinas C o C++ externas, no emita `printf()`.
  - En código de rutinas COBOL externas, no emita `display`

  Aunque las rutinas externas no pueden visualizar datos en una salida estándar, sí pueden incluir código que grabe datos en un archivo del sistema de archivos del servidor de bases de datos.

  Para las rutinas delimitadas (fenced) que se ejecutan en entornos UNIX, el directorio de destino en que se debe crear el archivo, o el propio archivo, debe tener los permisos adecuados, de forma que el propietario del archivo `sqllib/adm/.fenced` pueda crearlo o grabar en él. Si se trata de rutinas no delimitadas (not fenced), el propietario de la instancia debe tener permisos de creación, lectura y grabación para el directorio en que se abra el archivo.

  Nota: DB2 no intenta sincronizar la entrada externa ni la salida que realice una rutina con transacciones propias de DB2. Así, por ejemplo, si una UDF graba en un archivo durante una transacción y más tarde se retira dicha transacción por cualquier motivo, no se realiza ningún intento de descubrir ni deshacer las grabaciones efectuadas en el archivo.
- En rutinas externas no pueden ejecutarse sentencias o mandatos relacionados con la conexión. Esta restricción es aplicable a las siguientes sentencias:
  - BACKUP

  - CONNECT
  - CONNECT TO
  - CONNECT RESET
  - CREATE DATABASE
  - DROP DATABASE
  - FORWARD RECOVERY
  - RESTORE
- No es recomendable utilizar funciones del sistema operativo dentro de las rutinas. El uso de estas funciones no está restringido, excepto en los casos siguientes:
  - **No deben instalarse manejadores de señales definidos por el usuario para rutinas externas. Si no se cumple con esta restricción, se pueden producir anomalías inesperadas durante la ejecución de las rutinas externas, terminaciones anormales de base de datos u otros problemas. La instalación de manejadores de señales también puede interferir en el funcionamiento de la JVM para las rutinas de Java.**
  - Las llamadas al sistema que terminan un proceso pueden terminar de forma anómala uno de los procesos de DB2 y causar como consecuencia una anomalía del sistema o de la aplicación de base de datos.

    Otras llamadas al sistema también pueden ocasionar problemas si interfieren en el funcionamiento normal del gestor de bases de datos DB2. Por ejemplo, una función que intente descargar una biblioteca que contenga una función definida por el usuario desde la memoria podría causar problemas graves. Preste atención al programar y al probar las rutinas externas que contengan llamadas al sistema.
- Las rutinas externas no deben contener mandatos que terminen el proceso actual. Una rutina externa siempre tiene que devolver el control al gestor de bases de datos DB2 sin terminar el proceso actual.
- Las bibliotecas, clases o ensamblajes de rutinas externas no deben actualizarse mientras la base de datos está activa, excepto en casos especiales. Si es necesario realizar una actualización mientras el gestor de bases de datos de DB2 está activo, y no es posible detener y volver a iniciar la instancia, cree la biblioteca, clase o ensamblaje nuevo para la rutinas con uno diferente. A continuación, utilice la sentencia ALTER para cambiar el valor de la cláusula EXTERNAL NAME de la rutina externa de modo que haga referencia al nombre del archivo de biblioteca, clase o ensamblaje nuevo.
- La variable de entorno `DB2CKPTR` no está disponible en las rutinas externas. Las demás variables de entorno cuyos nombres empiezan por 'DB2' se capturan en el momento en que se inicia el gestor de bases de datos y pueden utilizarse en las rutinas externas.
- Algunas variables de entorno cuyos nombres no comienzan por 'DB2' no están disponibles para las rutinas externas delimitadas (fenced). Por ejemplo, la variable de entorno `LIBPATH` no puede utilizarse. Sin embargo, estas variables sí están disponibles para las rutinas externas que no están delimitadas (not fenced).
- Los valores de variables de entorno establecidos una vez iniciado el gestor de bases de datos de DB2 no están disponibles para las rutinas externas.
- Dentro de las rutinas externas, debe limitarse el uso de recursos protegidos, recursos a los que únicamente puede acceder un proceso a la vez. Si han de utilizarse, procure reducir la probabilidad de puntos muertos cuando dos rutinas externas intenten acceder al recurso protegido. Si se producen puntos muertos en dos o más rutinas, mientras se intenta acceder al recurso protegido, el gestor

de bases de datos de DB2 no será capaz de detectar o resolver la situación. Esto ocasionará que los procesos de rutinas externas se queden colgados.

- La memoria para los parámetros de rutinas externas no debe asignarse explícitamente en el servidor de bases de datos DB2. El gestor de bases de datos de DB2 asigna automáticamente almacenamiento basándose en la declaración de parámetros de la sentencia CREATE para la rutina. No modifique ningún puntero del almacenamiento para los parámetros de las rutinas externas. Un intento de cambiar un puntero por un puntero del almacenamiento creado localmente puede tener como consecuencia fugas de memoria, corrupción de los datos o terminaciones anormales.
- No utilice datos estáticos ni globales en las rutinas externas. DB2 no puede garantizar que la memoria utilizada por las variables estáticas o globales no se toque entre las invocaciones de rutinas externas. Para las UDF y los métodos, puede emplear áreas reutilizables a fin de almacenar los valores que se utilizarán entre las invocaciones.
- Los valores de todos los parámetros de SQL se colocan en el almacenamiento intermedio. Esto significa que se realiza una copia del valor y se le pasa a la rutina externa. Si se efectúan cambios en los parámetros de entrada de una rutina externa, estos cambios no repercutirán en los valores de SQL ni en el proceso. Sin embargo, si una rutina externa graba, en un parámetro de entrada o de salida, más datos de los especificados en la sentencia CREATE, se habrá corrompido la memoria y es posible que la rutina termine anormalmente.

Restricciones que se aplican solamente a los procedimientos externos:

- Cuando se devuelvan conjuntos de resultados desde procedimientos almacenados anidados, podrá abrir un cursor con el mismo nombre en varios niveles de anidamiento. No obstante, las aplicaciones anteriores a la versión 8 sólo podrán acceder al primer conjunto de resultados que se haya abierto. Esta restricción no se aplica a los cursores abiertos con un nivel de paquete diferente.

Restricciones que se aplican solamente a las funciones externas:

- Las funciones externas no pueden devolver conjuntos de resultados. Todos los cursores abiertos dentro de una función externa deben estar cerrados en el momento en que se complete la invocación a la llamada final de la función.
- Las asignaciones dinámicas de memoria en una rutinas externa deben liberarse antes de que la rutina externa devuelva el control. De lo contrario, se producirán fugas de memoria y constante aumento del consumo de memoria de un proceso de DB2 puede tener como consecuencia que el sistema de bases de datos se quede sin memoria.

  Para las funciones definidas por el usuario externas y los métodos externos, pueden utilizarse áreas reutilizables para asignar la memoria dinámica necesaria para la invocación de múltiples funciones. Si se utilizan áreas reutilizables de este modo, especifique el atributo FINAL CALL en las sentencias CREATE FUNCTION o CREATE METHOD. Así se garantiza que la memoria asignada se liberará antes de que la rutina devuelva el control.

Conceptos relacionados:

- "Visión general de las rutinas externas" en la página 23
- "Manejo de tipos de datos de SQL en rutinas C y C++" en *Desarrollo de SQL y rutinas externas*

Información relacionada:

- "Tipos de datos de SQL soportados en la automatización de OLE" en *Desarrollo de SQL y rutinas externas*
- "Tipos de datos SQL soportados en OLE DB" en *Desarrollo de SQL y rutinas externas*
- "Correlaciones de tipos de datos entre DB2 y OLE DB" en la página 137
- "Tipos de datos de SQL soportados en las aplicaciones de SQL incorporado para C y C++" en *Desarrollo de aplicaciones de SQL incorporado*
- "Sentencia ALTER FUNCTION" en *Consulta de SQL, Volumen 2*
- "Sentencia ALTER METHOD" en *Consulta de SQL, Volumen 2*
- "Sentencia ALTER PROCEDURE" en *Consulta de SQL, Volumen 2*
- "Sentencia CALL" en *Consulta de SQL, Volumen 2*
- "Sentencia CREATE FUNCTION" en *Consulta de SQL, Volumen 2*
- "Sentencia CREATE METHOD" en *Consulta de SQL, Volumen 2*
- "Sentencia CREATE PROCEDURE" en *Consulta de SQL, Volumen 2*

Creación de rutinas externas

Las rutinas externas, incluidos procedimientos y funciones, se crean de forma parecida a las rutinas con otras implementaciones, aunque algunos pasos adicionales necesarios porque la implementación de la rutina requiere la codificación, compilación y despliegue del código fuente.

Elegirá implementar una rutina externa si:

- Desea encapsular lógica compleja en una rutina que acceda a la base de datos o que realice una acción fuera de la base de datos.
- Necesita que la lógica encapsulada se invoque desde cualquiera de estos elementos: diversas aplicaciones, el CLP, otra rutina (procedimiento, función (UDF) o método) o un activador.
- Se siente más cómodo al codificar esta lógica en un lenguaje de programación en lugar de utilizar sentencias de SQL y SQL PL.
- Necesita la lógica de rutina para realizar operaciones externas a la base de datos, tales como escribir o leer un archivo del servidor de base de datos, la ejecución de otra aplicación, o lógica que no puede representarse con sentencias de SQL y SQL PL.

Requisitos previos:

- Conocimiento de la implementación de rutinas externas. Para obtener información sobre las rutinas externas en general, consulte el tema:
  - "Rutinas externas" en la página 21
  - "Creación de rutinas externas" en la página 45
- Debe instalarse el Cliente DB2.
- El servidor de bases de datos debe ejecutar un sistema operativo que dé soporte a los compiladores del lenguaje de programación de la implementación elegida y al software de desarrollo.
- Los compiladores necesarios y el soporte de ejecución para el lenguaje de programación elegido deben estar instalados en el servidor de base de datos
- Autorización para ejecutar la sentencia CREATE PROCEDURE, CREATE FUNCTION o CREATE METHOD.

Restricciones:

Para obtener una lista de las restricciones asociadas con rutinas externas, consulte:

- "Restricciones de las rutinas externas" en la página 57

Procedimiento:

1. Codifique la lógica de la rutina en el lenguaje de programación elegido.
   - Para obtener información general sobre rutinas externas, características de las rutinas y la implementación de características de las rutinas, consulte los temas a los que se hace referencia en la sección Requisitos previos.
   - Utilice o importe los archivos de cabecera necesarios para dar soporte a la ejecución de sentencias de SQL.
   - Declare las variables y los parámetros correctamente utilizando tipos de datos del lenguaje de programación que se correlacionen con tipos de datos de SQL de DB2.
2. Los parámetros deben declararse de acuerdo con el formato requerido por el estilo de parámetro para el lenguaje de programación elegido. Si desea más información sobre los parámetros y las declaraciones de prototipo, consulte:
   - "Estilos de parámetros de rutinas externas" en la página 46
3. Construya el código en una biblioteca o archivo de clase.
4. Copie la biblioteca o archivo de clase en el directorio de función DB2 en el servidor de base de datos. Es recomendable almacenar ensamblajes o bibliotecas asociadas con rutinas de DB2 en el directorio de función. Para conocer más acerca del directorio de función, consulte la cláusula EXTERNAL de una de las sentencias siguientes: CREATE PROCEDURE o CREATE FUNCTION.

   Puede copiar el ensamblaje en otro directorio del servidor si lo desea, pero, para invocar satisfactoriamente la rutina, debe anotar el nombre de vía de acceso completamente calificado del ensamblaje porque lo necesitará en el paso siguiente.
5. Ejecute de forma dinámica o estática la sentencia CREATE de lenguaje SQL correspondiente para el tipo de rutina: CREATE PROCEDURE o CREATE FUNCTION.
   - Especifique la cláusula `LANGUAGE` con el valor adecuado correspondiente a la API o lenguaje de programación elegido. Ejemplos: CLR, C, JAVA.
   - Especifique la cláusula `PARAMETER STYLE` con el nombre del estilo de parámetros soportado que se ha implementado en el código de la rutina.
   - Especifique la cláusula `EXTERNAL` con el nombre del archivo de biblioteca, clase o ensamblaje que se ha de asociar a la rutina utilizando uno de los valores siguientes:
     – el nombre de vía de acceso calificado al completo del archivo de biblioteca, clase o ensamblaje de la rutina.
     – el nombre de vía de acceso relativa del archivo de biblioteca, clase o ensamblaje de la rutina con relación al directorio de función.

     Por omisión, DB2 buscará el archivo de biblioteca, clase o ensamblaje por el nombre en el directorio de función, a menos que se especifique un nombre de vía de acceso completamente calificado o relativo para el archivo en la cláusula EXTERNAL.
   - Especifique DYNAMIC RESULT SETS con un valor numérico si la rutina es un procedimiento y ha de devolver uno o varios conjunto de resultados al llamador.
   - Especifique las demás cláusulas necesarias para caracterizar la rutina.

Para invocar la rutina externa, consulte Invocación de la rutina

Conceptos relacionados:

- "Rutinas externas" en la página 21
- "Creación de rutinas externas" en la página 45
- "Restricciones de las rutinas externas" en la página 57
- "Estilos de parámetros de rutinas externas" en la página 46
- "Invocación de rutinas" en *Desarrollo de SQL y rutinas externas*
- "Visión general de las rutinas externas" en la página 23

Capítulo 5. Rutinas de ejecución en el lenguaje común (CLR) .NET

Rutinas de ejecución en el lenguaje común (CLR) .NET

En DB2, una rutina de ejecución en el lenguaje común (CLR) es una rutina externa creada ejecutando una sentencia CREATE PROCEDURE o CREATE FUNCTION que hace referencia a un ensamblaje .NET como su cuerpo de código externo.

Los términos siguientes son importantes en el contexto de las rutinas CLR:

.NET Framework
Un entorno de desarrollo de aplicaciones de Microsoft que comprende tanto la CLR como la biblioteca de clases de .NET Framework diseñada para proporcionar un entorno de programación coherente con miras al desarrollo e integración de partes de código.

Ejecución en el lenguaje común (CLR)
El intérprete de ejecución para todas las aplicaciones .NET Framework.

lenguaje intermedio (IL)
Tipo de código de bytes compilado que se interpreta mediante la CLR de .NET Framework. El código fuente de todos los lenguajes compatibles con .NET se compila en el código de bytes IL.

ensamblaje
Un archivo que contiene código de bytes IL. Puede ser una biblioteca o un ejecutable.

Es posible implementar rutinas CLR en los lenguajes que se puedan compilar en un ensamblaje IL. Estos lenguajes incluyen los siguientes, pero no están limitados a ellos: Managed C++, C#, Visual Basic y J#.

Antes de desarrollar una rutina CLR, es importante comprender los conceptos básicos de las rutinas y las características exclusivas y específicas de las rutinas CLR. Para informarse más acerca de las rutinas y de las rutinas CLR, consulte:

- "Beneficios del uso de rutinas" en la página 22
- "Tipos de datos de SQL soportados para el Proveedor de datos DB2 .NET" en la página 66
- "Parámetros de rutinas .NET CLR" en la página 68
- "Devolución de conjuntos de resultados desde procedimientos .NET CLR" en la página 71
- "Restricciones de las rutinas CLR .NET" en la página 74
- "Errores relacionados con rutinas CLR .NET" en la página 89

Desarrollar una rutina CLR es fácil. Para obtener instrucciones paso a paso sobre cómo desarrollar una rutina CLR y ejemplos completos, consulte:

- "Creación de rutinas CLR .NET desde ventanas de mandatos de DB2" en la página 77
- "Ejemplos de procedimientos CLR .NET en C#" en la página 103
- "Ejemplos de funciones CLR .NET en C#" en la página 126

Conceptos relacionados:

- "Autorizaciones y vinculación de rutinas que contienen SQL" en *Desarrollo de SQL y rutinas externas*
- "Beneficios del uso de rutinas" en la página 22
- "Estilos de parámetros de rutinas externas" en la página 46
- "Rutinas externas" en la página 21
- "SQL en rutinas externas" en *Desarrollo de SQL y rutinas externas*

Tareas relacionadas:

- "Creación de rutinas CLR (Common Language Runtime) .NET" en la página 80
- "Creación de código de rutinas .NET CLR" en la página 79
- "Creación de rutinas .NET CLR" en la página 75
- "Depuración de rutinas .NET CLR" en la página 88
- "Ejemplos de rutinas .NET CLR" en la página 93

Ejemplos relacionados:

- "SpCreate.db2 -- Creates the external procedures implemented in spserver.cs"
- "SpServer.cs -- C# external code implementation of procedures created in spcat.db2"
- "SpCreate.db2 -- Creates the external procedures implemented in spserver.vb"
- "SpServer.vb -- VB.NET implementation of procedures created in SpCat.db2"

Software de desarrollo de rutinas .NET CLR soportado

Soporte para el desarrollo de rutinas externas en lenguajes .NET CLR

Para desarrollar rutinas externas en lenguajes .NET CLR y ejecutarlos satisfactoriamente, tendrá que utilizar software de desarrollo, versiones de clientes y servidores de base de datos DB2 y sistemas operativos soportados.

Sistemas operativos soportados para el desarrollo de rutinas .NET CLR con .NET Framework 1.1 o .NET Framework 2.0:

- Windows 2000
- Windows XP (edición de 32 bits)
- Windows Server 2003 (edición de 32 bits)

Servidores y clientes de bases de datos DB2 soportados para el desarrollo de rutinas .NET CLR:

Se deben instalar los siguientes servidores y clientes mínimos de DB2:

- Servidor DB2: la versión mínima soportada es DB2 Versión 8.2.
- Cliente DB2: la versión mínima soportada es DB2 Versión 7.2.

Software de desarrollo necesario para rutinas .NET CLR:

Uno de los dos productos de software siguientes debe estar instalado en el mismo sistema que el servidor de bases de datos de DB2:

- Microsoft .NET Framework, Versión 1.1

- Microsoft .NET Framework, Versión 2.0

Microsoft .NET Framework está disponible de forma independiente o como parte de uno de los siguientes Kits de desarrollo de software:

- Kit de desarrollo de software de Microsoft .NET Framework Versión 1.1
- Kit de desarrollo de software de Microsoft .NET Framework Versión 2.0

Las rutinas externas .NET CLR se pueden implementar en cualquier lenguaje que Microsoft .NET Framework pueda compilar en un conjunto IL. Estos lenguajes incluyen los siguientes, pero no están limitados a ellos: Managed C++, C#, Visual Basic y J#.

Conceptos relacionados:

- "Software de desarrollo .NET soportado" en la página 2
- "Rutinas de ejecución en el lenguaje común (CLR) .NET" en la página 63
- "Herramientas para desarrollar rutinas .NET CLR" en la página 65

Tareas relacionadas:

- "Creación de rutinas CLR (Common Language Runtime) .NET" en la página 80

Información relacionada:

- "Lenguajes de programación y compiladores soportados para el desarrollo de aplicaciones de bases de datos" en *Iniciación al desarrollo de aplicaciones de bases de datos*

Herramientas para desarrollar rutinas .NET CLR

las herramientas pueden realizar la tarea de desarrollar rutinas .NET CLR que puedan interactuar con la base de datos de DB2 de forma más rápida y sencilla.

Las rutinas .NET CLR se pueden desarrollar en Microsoft Visual Studio .NET mediante herramientas gráficas disponibles en:

- IBM DB2 Development Add-In para Microsoft Visual Studio .NET 1.2

Las siguientes interfaces de línea de mandatos, proporcionadas con DB2, también permiten desarrollar rutinas .NET CLR en servidores de bases de datos de DB2:

- Procesador de línea de mandatos de DB2 (DB2 CLP)
- Ventana de mandatos de DB2

Conceptos relacionados:

- "Integración de DB2 en Visual Studio" en la página 6
- "Soporte para el desarrollo de rutinas externas en lenguajes .NET CLR" en la página 64
- "Herramientas para desarrollar rutinas" en *Desarrollo de SQL y rutinas externas*

Diseño de rutinas .NET CLR

Diseño de rutinas .NET CLR

Al diseñar rutinas .NET CLR, deberá tener en cuenta consideraciones generales sobre el diseño de rutinas externas y consideraciones sobre el diseño específico de .NET CLR.

Requisitos previos:

Conocimientos y experiencia en el desarrollo de aplicaciones de .NET y un conocimiento general de las rutinas externas. Los temas siguientes proporcionan parte de la información necesaria sobre requisitos previos.

Para obtener más información sobre las funciones y usos de las rutinas externas consulte:

- Rutinas externas

Para obtener más información sobre las características de las rutinas .NET CLR, consulte:

- rutinas .NET CLR

Procedimiento:

Con el conocimiento de los requisitos previos, el diseño de rutinas de SQL incorporado consiste principalmente en conocer las funciones y características exclusivas de las rutinas .NET CLR:

- Conjuntos de inclusión que proporcionan soporte para le ejecución de sentencias de SQL en rutinas .NET CLR (IBM.Data.DB2)
- Tipos de datos de SQL soportados en rutinas .NET CLR
- Parámetros de rutinas .NET CLR
- Devolución de conjuntos de resultados desde rutinas .NET CLR
- Valores de modalidad de control de ejecución y seguridad para rutinas .NET CLR
- Restricciones de rutinas .NET CLR
- Devolución de conjuntos de resultados desde procedimientos .NET CLR

Después de conocer las características de .NET CLR, puede: ″Crear rutinas .NET CLR″.

Conceptos relacionados:

- “Rutinas de ejecución en el lenguaje común (CLR) .NET” en la página 63
- “Parámetros de rutinas .NET CLR” en la página 68
- “Modalidades de seguridad y de ejecución para rutinas CLR” en la página 73

Tareas relacionadas:

- “Devolución de conjuntos de resultados desde procedimientos .NET CLR” en la página 71

Información relacionada:

- “Restricciones de las rutinas CLR .NET” en la página 74
- “Tipos de datos de SQL soportados para el Proveedor de datos DB2 .NET” en la página 66

Tipos de datos de SQL soportados para el Proveedor de datos DB2 .NET

La tabla siguiente lista las correlaciones entre los tipos de datos `DB2Type` en el Proveedor de datos DB2 .NET, el tipo de datos DB2 y el tipo de datos .NET Framework correspondiente:

Tabla 4. Correlación de tipos de datos DB2 con tipos de datos .NET

DB2Type Enum	Tipo de datos DB2	Tipo de datos .NET
SmallInt	SMALLINT	Int16
Integer	INTEGER	Int32
BigInt	BIGINT	Int64
Real	REAL	Single
Real370(2)	REAL	Single
Double	DOUBLE PRECISION	Double
Float	FLOAT	Double
Decimal	DECIMAL	Decimal
Numeric	DECIMAL	Decimal
Date	DATE	DateTime
Time	TIME	TimeSpan
Timestamp	TIMESTAMP	DateTime
Char	CHAR	String
VarChar	VARCHAR	String
LongVarChar(1)	LONG VARCHAR	String
Binary	CHAR FOR BIT DATA	Byte[]
VarBinary	VARCHAR FOR BIT DATA	Byte[]
LongVarBinary(1)	LONG VARCHAR FOR BIT DATA	Byte[]
Graphic	GRAPHIC	String
VarGraphic	VARGRAPHIC	String
LongVarGraphic(1)	LONG GRAPHIC	String
Clob	CLOB	String
Blob	BLOB	Byte[]
DbClob	DBCLOB(N)	String
Xml(3)	XML	IBM.Data.DB2Types.DB2Xml

Notas:

1. Estos tipos de datos no están soportados en las rutinas de ejecución en el lenguaje común DB2 .NET. Sólo están soportados en las aplicaciones cliente.
2. Una propiedad DB2Parameter.Value del tipo DB2Type.Xml puede aceptar variables de los tipos siguientes: Serie, Byte[], DB2Xml y XmlReader.
3. La enumeración de Real370 sólo puede utilizarse para los parámetros de las sentencias SQL que se ejecutan en DB2 UDB para las bases de datos OS/390 y z/OS.

Nota: La estructura `dbinfo` pasa a las funciones y procedimientos CLR como un parámetro. El área reutilizable y el tipo de llamada para las UDF CLR también pasan a las rutinas CLR como parámetros. Si desea información sobre los tipos de datos CLR adecuados para estos parámetros, consulte el tema relacionado:

- Parámetros de las rutinas CLR

Conceptos relacionados:

- "Rutinas de ejecución en el lenguaje común (CLR) .NET" en la página 63
- "Diseño de rutinas .NET CLR" en la página 65
- "Estilos de parámetros de rutinas externas" en la página 46
- "Parámetros de rutinas .NET CLR" en la página 68

Tareas relacionadas:

- "Creación de rutinas CLR .NET desde ventanas de mandatos de DB2" en la página 77
- "Ejemplos de funciones CLR .NET en C#" en la página 126
- "Ejemplos de procedimientos CLR .NET en C#" en la página 103
- "Pase de parámetros de tipo estructurado a rutinas externas" en *Desarrollo de SQL y rutinas externas*

Ejemplos relacionados:

- "SpCreate.db2 -- Creates the external procedures implemented in spserver.cs"
- "SpServer.cs -- C# external code implementation of procedures created in spcat.db2"
- "SpCreate.db2 -- Creates the external procedures implemented in spserver.vb"
- "SpServer.vb -- VB.NET implementation of procedures created in SpCat.db2"

Parámetros de rutinas .NET CLR

La declaración de parámetros de las rutinas .NET CLR debe cumplir con los requisitos de uno de los estilos de parámetros soportados y debe respetar los requisitos de palabra clave de parámetro del lenguaje .NET determinado que se utilice para la rutina. Si la rutina ha de emplear un área reutilizable o la estructura `dbinfo` o ha de tener la interfaz de parámetros PROGRAM TYPE MAIN, existen detalles adicionales a considerar. Este tema aborda todas las consideraciones sobre los parámetros CLR.

Estilos de parámetros soportados para las rutinas CLR:

El estilo de parámetros de la rutina se debe especificar durante la creación de la rutina en la cláusula EXTERNAL de la sentencia CREATE para la rutina. El estilo de parámetros se debe reflejar adecuadamente en la implementación del código de la rutina CLR externa. Están soportados los estilos de parámetros siguientes de DB2 para las rutinas CLR:

- `SQL` (Soportado para procedimientos y funciones)
- `GENERAL` (Soportado para procedimientos únicamente)
- `GENERAL WITH NULLS` (Soportado para procedimientos únicamente)
- `DB2SQL` (Soportado para procedimientos y funciones)

Si desea más información sobre estos estilos de parámetros, consulte el tema:

- Estilos de parámetros para rutinas externas

Indicadores de nulo de los parámetros de rutinas CLR:

Si el estilo de parámetros elegido para una rutina CLR requiere que se especifiquen indicadores de nulo para los parámetros, los indicadores de nulo se han de pasar a la rutina CLR como valores de tipo System.Int16 o en un valor System.Int16[] cuando el estilo de parámetros exija un vector de indicadores de nulo.

Cuando el estilo de parámetros impone que se pasen los indicadores de nulo a la rutina como parámetros diferenciados, tal como requiere el estilo de parámetros SQL, se necesita un indicador de nulo System.Int16 para cada parámetro.

En los lenguajes .NET, los parámetros diferenciados deben ir precedidos de una palabra clave para indicar si el parámetro se pasa por valor o por referencia. La misma palabra clave que se utilice para un parámetro de rutina se debe utilizar para el parámetro de indicador de nulo asociado. Las palabras clave que indican si un argumento se pasa por valor o por referencia se describen con más detalle a continuación.

Si desea obtener más información sobre el estilo de parámetros SQL y los otros estilos de parámetros soportados, consulte el tema:

- Estilos de parámetros para rutinas externas

Pase de parámetros de rutinas CLR por valor o por referencia:

Las rutinas de lenguaje .NET que se compilan en el código de bytes del lenguaje intermedio (IL) requieren que los parámetros vayan precedidos de palabras clave que indiquen las propiedades determinadas del parámetro, tales como si el parámetro se pasa por valor o por referencia, o si es un parámetro de sólo entrada o de sólo salida.

Las palabras clave de parámetro son específicas del lenguaje .NET. Por ejemplo, para pasar un parámetro por referencia en `C#`, la palabra clave de parámetro es `ref`, mientras que, en `Visual Basic`, un parámetro por referencia se indica mediante la palabra clave `byRef`. Las palabras clave se deben emplear para indicar el uso del parámetro de SQL (`IN`, `OUT`, `INOUT`) que se ha especificado en la sentencia CREATE para la rutina.

Se imponen las normas siguientes al aplicar palabras clave de parámetro a los parámetros de rutinas de lenguaje .NET en las rutinas de DB2:

- Los parámetros de tipo `IN` se deben declarar *sin* palabra clave de parámetro en C#, y se deben declarar con la palabra clave `byVal` en `Visual Basic`.
- Los parámetros de tipo `INOUT` se deben declarar con la palabra clave específica del lenguaje que indique que el parámetro se pasa por referencia. En `C#`, la palabra clave adecuada es `ref`. En `Visual Basic`, la palabra clave adecuada es `byRef`.
- Los parámetros de tipo `OUT` se deben declarar con la palabra clave específica del lenguaje que indique que el parámetro es de sólo salida. En `C#`, utilice la palabra clave `out`. En Visual Basic, el parámetro se debe declarar con la palabra clave `byRef`. Los parámetros de sólo salida siempre deben tener asignado un valor antes de que la rutina vuelva al llamador. Si la rutina no asigna un valor al parámetro de sólo salida, se generará un error cuando se compile la rutina .NET.

A continuación, se muestra el aspecto de un prototipo de procedimiento en C# del estilo de parámetros SQL para una rutina que devuelve un solo parámetro de salida `language`.

```
public static void Counter  (out String language,
                             out Int16  languageNullInd,
                             ref String sqlState,
                                 String funcName,
                                 String funcSpecName,
```

```
                                ref String sqlMsgString,
                                    Byte[] scratchPad,
                                    Int32  callType);
```

Es evidente que se implementa el estilo de parámetros SQL por el parámetro de indicador de nulo adicional, `languageNullInd`, asociado con el parámetro de salida `language`, los parámetros para pasar el SQLSTATE, el nombre de rutina, el nombre de rutina específico y un mensaje de error de SQL opcional definido por el usuario. Se han especificado palabras clave para los parámetros del modo siguiente:

- En C#, no es necesaria ninguna palabra clave de parámetro para los parámetros de sólo entrada.
- En C#, la palabra clave 'out' indica que la variable es un parámetro de sólo salida y que el llamador no ha inicializado su valor.
- En C#, la palabra clave 'ref' indica que el llamador ha inicializado el parámetro y que la rutina puede modificar este valor opcionalmente.

Consulte la documentación específica del lenguaje .NET en relación con el pase de parámetros para informarse sobre las palabras clave de parámetro en ese lenguaje.

Nota:

> DB2 controla la asignación de memoria para todos los parámetros y mantiene las referencias CLR a todos los parámetros pasados a o desde una rutina.

No es necesario ningún marcador de parámetro para los conjuntos de resultados de procedimiento:

Los marcadores de parámetros no son necesarios en la declaración de un procedimiento para un conjunto de resultados que se va a devolver al llamador. Cualquier sentencia de cursor que no se haya cerrado desde dentro de un procedimiento almacenado CLR se devolverá a su llamador como conjunto de resultados.

Si desea más información sobre los conjuntos de resultados en las rutinas CLR, consulte:

- "Devolución de conjuntos de resultados desde procedimientos .NET CLR" en la página 71

Estructura Dbinfo como parámetro CLR:

La estructura `dbinfo` utilizada para pasar parámetros adicionales de información de base de datos a y desde una rutina está soportada para las rutinas CLR mediante el uso de una clase `dbinfo` de IL. Esta clase contiene todos los elementos que se encuentran en la estructura `sqludf_dbinfo` del lenguaje `C`, a excepción de los campos de longitud asociados con las series. La longitud de cada serie se puede encontrar utilizando la propiedad de lenguaje .NET `Length` de la serie determinada.

Para acceder a la clase `dbinfo`, simplemente incluya el conjunto `IBM.Data.DB2` en el archivo que contenga la rutina y añada un parámetro del tipo `sqludf_dbinfo` a la signatura de la rutina, en la posición especificada por el estilo de parámetros utilizado.

Área reutilizable de UDF como parámetro CLR:

Si se solicita un área reutilizable para una función definida por el usuario, se pasa a la rutina como parámetro `System.Byte[]` del tamaño especificado.

Parámetro de llamada final o de tipo de llamada de UDF CLR:

Para las funciones definidas por el usuario que han solicitado un parámetro de llamada final o para las funciones de tabla, el parámetro de tipo de llamada se pasa a la rutina como tipo de datos `System.Int32`.

PROGRAM TYPE MAIN está soportado para los procedimientos CLR:

PROGRAM TYPE MAIN está soportado para los procedimientos .NET CLR. Los procedimientos definidos para el uso de Program Type MAIN deben tener la signatura siguiente:

```
void functionname(Int32 NumParams, Object[] Params)
```

Conceptos relacionados:

- "Diseño de rutinas .NET CLR" en la página 65
- "Estilos de parámetros de rutinas externas" en la página 46
- "Manejo de parámetros en procedimientos PROGRAM TYPE MAIN o PROGRAM TYPE SUB" en *Desarrollo de SQL y rutinas externas*
- "Modalidades de parámetros de procedimiento" en *Desarrollo de SQL y rutinas externas*
- "Áreas reutilizables para funciones externas y métodos" en la página 33

Tareas relacionadas:

- "Ejemplos de funciones CLR .NET en C#" en la página 126
- "Ejemplos de procedimientos CLR .NET en C#" en la página 103
- "Devolución de conjuntos de resultados desde procedimientos .NET CLR" en la página 71
- "Pase de parámetros de tipo estructurado a rutinas externas" en *Desarrollo de SQL y rutinas externas*

Información relacionada:

- "Tipos de datos de SQL soportados para el Proveedor de datos DB2 .NET" en la página 66

Devolución de conjuntos de resultados desde procedimientos .NET CLR

Es posible desarrollar procedimientos CLR que devuelvan conjuntos de resultados a la rutina o aplicación que realiza la llamada. No se pueden devolver conjuntos de resultados desde las funciones de CLR (UDF).

La representación en .NET de un conjunto de resultados es un objeto `DB2DataReader` que se puede devolver desde una de las diversas llamadas de ejecución de un objeto `DB2Command`. Se puede devolver cualquier objeto `DB2DataReader` cuyo método `Close()` no se haya llamado explícitamente antes de la devolución del procedimiento. El orden en que se devuelven los conjuntos de resultados al llamador es el mismo orden en que se han creado las instancias de

los objetos DB2DataReader. No se requieren parámetros adicionales en la definición de función para devolver un conjunto de resultados.

Requisitos previos:

Un conocimiento de cómo se crean las rutinas CLR le ayudará a seguir los pasos del procedimiento siguiente sobre la devolución de resultados desde un procedimiento CLR.

- "Creación de rutinas CLR .NET desde ventanas de mandatos de DB2" en la página 77

Procedimiento:

Para devolver un conjunto de resultados de un procedimiento CLR:

1. En la sentencia CREATE PROCEDURE para la rutina CLR, debe especificar, junto con cualquier otra cláusula apropiada, la cláusula DYNAMIC RESULT SETS con un valor que equivalga al número de conjuntos de resultados que debe devolver el procedimiento.
2. Los marcadores de parámetros no son necesarios en la declaración de un procedimiento para un conjunto de resultados que se va a devolver al llamador.
3. En la implementación en el lenguaje .NET de la rutina CLR, cree un objeto DB2Connection, un objeto DB2Command y un objeto DB2Transaction. El objeto DB2Transaction es responsable de la retrotracción y confirmación de las transacciones de base de datos.
4. Inicialice la propiedad Transaction del objeto DB2Command para el objeto DB2Transaction.
5. Asigne una consulta de serie a la propiedad CommandText del objeto DB2Command que defina el conjunto de resultados que desea devolver.
6. Cree una instancia de un DB2DataReader y asígnele el resultado de la invocación del método ExecuteReader del objeto DB2Command. El conjunto de resultados de la consulta estará incluido en el objeto DB2DataReader.
7. No ejecute el método Close() del objeto DB2DataReader en ningún punto anterior a la devolución del procedimiento al llamador. El objeto DB2DataReader se devolverá todavía abierto como un conjunto de resultados al llamador.

   Cuando se deja abierto más de un DB2DataReader después de la devolución de un procedimiento, los DB2DataReader se devuelven al llamador siguiendo el orden de su creación. Sólo se devolverá al llamador el número de conjuntos de resultados especificado en la sentencia CREATE PROCEDURE.
8. Compile el procedimiento de lenguaje .NET CLR e instale el conjunto en la ubicación especificada por la cláusula EXTERNAL de la sentencia CREATE PROCEDURE. Ejecute la sentencia CREATE PROCEDURE para el procedimiento CLR, si todavía no lo ha hecho.
9. Una vez que haya instalado el conjunto del procedimiento CLR en la ubicación adecuada y haya ejecutado la sentencia CREATE PROCEDURE satisfactoriamente, puede invocar el procedimiento con la sentencia CALL para ver la devolución de conjuntos de resultados al llamador.

   Si desea obtener información sobre la llamada a procedimientos y otros tipos de rutinas, consulte el tema:

   •

Conceptos relacionados:

- "Rutinas de ejecución en el lenguaje común (CLR) .NET" en la página 63

- "Diseño de rutinas .NET CLR" en la página 65
- "Modalidades de parámetros de procedimiento" en *Desarrollo de SQL y rutinas externas*

Tareas relacionadas:

- "Creación de rutinas CLR .NET desde ventanas de mandatos de DB2" en la página 77

Modalidades de seguridad y de ejecución para rutinas CLR

Como administrador de bases de datos o desarrollador de aplicaciones, es posible que desee proteger los activos asociados a las rutinas externas de DB2 frente a manipulaciones no deseadas y restringir las acciones de las rutinas en el momento de la ejecución. Las rutinas de ejecución en el lenguaje común (CLR) .NET de DB2 dan soporte a la especificación de una modalidad de control de ejecución que identifica los tipos de acciones que una rutina puede realizar en el momento de la ejecución. En el momento de la ejecución, DB2 puede detectar si la rutina intenta realizar acciones que quedan fuera del ámbito de su modalidad de control de ejecución especificada, lo que puede resultar de ayuda para determinar si se ha puesto en peligro un activo.

Para establecer la modalidad de control de ejecución de una rutina CLR, especifique la cláusula opcional EXECUTION CONTROL en la sentencia CREATE correspondiente a la rutina. Las modalidades válidas son:

- SAFE
- FILEREAD
- FILEWRITE
- NETWORK
- UNSAFE

Para modificar la modalidad de control de ejecución de una rutina CLR existente, ejecute la sentencia ALTER PROCEDURE o ALTER FUNCTION.

Si no se especifica la cláusula EXECUTION CONTROL para una rutina CLR, por omisión la rutina CLR se ejecuta utilizando la modalidad de control de ejecución más restrictiva: SAFE. Las rutinas que se crean con esta modalidad de control de ejecución sólo pueden acceder a los recursos controlados por el gestor de bases de datos. Las modalidades de control de ejecución menos restrictivas permiten a la rutina acceder a archivos (FILEREAD o FILEWRITE) o efectuar operaciones de red como por ejemplo acceder a una página Web (NETWORK). La modalidad de control de ejecución UNSAFE especifica que no se debe colocar ninguna restricción en el comportamiento de la rutina. Las rutinas definidas con la modalidad de control de ejecución UNSAFE pueden ejecutar código binario.

Estas modalidades representan una jerarquía de acciones permitidas y una modalidad de nivel superior incluye las acciones permitidas por debajo de la misma en la jerarquía. Por ejemplo, la modalidad de control de ejecución NETWORK permite a una rutina acceder a páginas Web en Internet, a archivos de lectura y grabación y a recursos de acceso controlados por el gestor de bases de datos. Se recomienda utilizar la modalidad de control de ejecución más restrictiva posible y evitar utilizar la modalidad UNSAFE.

Si DB2 detecta en el momento de la ejecución que una rutina CLR está intentando una acción que queda fuera del ámbito de su modalidad de control de ejecución, DB2 devuelve un error (SQLSTATE 38501).

La cláusula EXECUTION CONTROL sólo se puede especificar para rutinas CLR LANGUAGE. El ámbito de aplicación de la cláusula EXECUTION CONTROL se limita a la propia rutina CLR .NET y no se aplica a ninguna otra rutina a la que pueda llamar.

Consulte la sintaxis de la sentencia CREATE correspondiente al tipo de rutina adecuado para ver una descripción completa de las modalidades de control de ejecución soportadas.

Conceptos relacionados:

- "Diseño de rutinas .NET CLR" en la página 65

Restricciones de las rutinas CLR .NET

Las restricciones generales de implementación que se aplican a todas las rutinas externas o a determinadas clases de rutinas (procedimiento o UDF) también se aplican a las rutinas CLR. Existen algunas restricciones que son particulares de las rutinas CLR. Estas restricciones se listan aquí.

La sentencia CREATE METHOD con la cláusula LANGUAGE CLR no está soportada:

No se pueden crear métodos externos para tipos estructurados de DB2 que hacen referencia a un conjunto CLR. El uso de una sentencia CREATE METHOD que especifique la cláusula LANGUAGE con el valor CLR no está soportado.

Los procedimientos CLR no se pueden implementar como procedimientos NOT FENCED:

Los procedimientos CLR no se pueden ejecutar como procedimientos no protegidos. La sentencia CREATE PROCEDURE para un procedimiento CLR no puede especificar la cláusula NOT FENCED.

La cláusula EXECUTION CONTROL restringe la lógica que contiene la rutina:

La cláusula EXECUTION CONTROL y el valor asociado determinan los tipos de lógica y las operaciones que pueden ejecutarse en una rutina .NET CLR. Por omisión el valor de la cláusula EXECUTOIN CONTROL se establece en SAFE. Para la lógica de rutina que lee archivos, graba en archivos o que accede a Internet, debe especificarse un valor diferente al valor por omisión para la cláusula EXECUTION CONTROL que sea menos restrictivo.

La precisión decimal máxima es de 29 y la escala decimal máxima es de 28 en una rutina CLR:

El tipo de datos DECIMAL en DB2 se representa con una precisión de 31 dígitos y una escala de 28 dígitos. El tipo de datos System.Decimal de CLR .NET está limitado a una precisión de 29 dígitos y a una escala de 28 dígitos. Por lo tanto, las rutinas CLR externas de DB2 no pueden asignar un valor a un tipo de datos System.Decimal que sea mayor que `(2^96)-1`, que es el valor más alto que puede representarse utilizando una precisión de 29 dígitos y una escala de 28 dígitos. DB2

devolverá un error de tiempo de ejecución (SQLSTATE 22003, SQLCODE -413) si se produce una asignación de este tipo. En el momento de la ejecución de la sentencia CREATE para la rutina, si un parámetro de tipo de datos DECIMAL está definido con una escala mayor que 28, DB2 devolverá un error (SQLSTATE 42613, SQLCODE -628).

Si es necesario que su rutina manipule valores decimales con la precisión y escala máximas soportadas por DB2, puede implementar la rutina externa en un lenguaje de programación distinto, como, por ejemplo, Java.

Tipos de datos no soportados en las rutinas CLR:

Los siguientes tipos de datos de SQL de DB2 no están soportados en las rutinas CLR:

- LONG VARCHAR
- LONG VARCHAR FOR BIT DATA
- LONG GRAPHIC
- DATALINK
- ROWID

Ejecución de una rutina CLR de 32 bits en una instancia de 64 bits:

Las rutinas CLR no se pueden ejecutar en instancias de 64 bits, porque .NET Framework no se puede instalar en sistemas operativos de 64 bits en este momento.

CLR .NET no soportado para la implementación de plug-ins de seguridad:

CLR .NET no está soportado para compilar y enlazar el código fuente de bibliotecas de plug-ins de seguridad.

Conceptos relacionados:

- “Rutinas de ejecución en el lenguaje común (CLR) .NET” en la página 63
- “Diseño de rutinas .NET CLR” en la página 65
- “Parámetros de rutinas .NET CLR” en la página 68

Tareas relacionadas:

- “Creación de rutinas CLR .NET desde ventanas de mandatos de DB2” en la página 77
- “Devolución de conjuntos de resultados desde procedimientos .NET CLR” en la página 71

Creación de rutinas .NET CLR

Creación de rutinas .NET CLR

La creación de rutinas Java incluye lo siguiente:

- Ejecutar la sentencia CREATE que define la rutina en un servidor de base de datos DB2.
- Desarrollar la implementación de la rutina que se corresponde con la definición de la rutina.

A continuación se citan las formas en las que puede crear rutinas Java:

- Mediante las herramientas gráficas que se proporcionan con DB2 Database Development Add-In para Visual Studio .NET 1.2
- Utilizando la Ventana de mandatos de DB2

En general resulta más fácil crear rutinas .NET CLR utilizando DB2 Database Development Add-In para Visual Studio .NET 1.2. Si no puede utilizarse esta opción, la ventana de mandatos de DB2 proporciona un soporte análogo por medio de una interfaz de línea de mandatos.

Requisitos previos:

- Revise la Visión general de la rutina .NET CLR.
- Asegúrese de que tiene acceso a un servidor de DB2 Versión 9, incluyendo instancias y bases de datos.
- Asegúrese de que el sistema operativo está en un nivel de versión soportado por los productos de base de datos DB2.
- Asegúrese de que el software de desarrollo Microsoft .NET esté en un nivel de versión que esté soportado para el desarrollo de las rutinas .NET CLR.
- Autorización para ejecutar la sentencia CREATE PROCEDURE o CREATE FUNCTION.

Restricciones:

Para obtener una lista de las restricciones asociadas con rutinas CLR, consulte:

- "Restricciones de las rutinas CLR .NET" en la página 74

Procedimiento:

Rutinas .NET CLR desde una de las siguientes interfaces:

- Visual Studio .NET cuando también esté instalado IBM DB2 Development Add-In para Microsoft Visual Studio .NET 1.2. Cuando se haya instalado el Add-In, las herramientas gráficas integradas en Visual Studio .NET estarán disponibles para crear rutinas .NET CLR que funcionen en servidores de base de datos DB2.
- Ventanas de mandatos de DB2

Para crear rutinas .NET CLR desde ventanas de mandatos de DB2, consulte:

- Creación de rutinas CLR .NET desde ventanas de mandatos de DB2

Conceptos relacionados:

- "Soporte para el desarrollo de rutinas externas en lenguajes .NET CLR" en la página 64
- "Rutinas de ejecución en el lenguaje común (CLR) .NET" en la página 63

Tareas relacionadas:

- "Creación de rutinas CLR .NET desde ventanas de mandatos de DB2" en la página 77

Información relacionada:

- "Restricciones de las rutinas CLR .NET" en la página 74

Creación de rutinas CLR .NET desde ventanas de mandatos de DB2

Los procedimientos y funciones que hacen referencia a un conjunto de lenguaje intermedio se crean de la misma forma que cualquier rutina externa. Elegirá implementar una rutina externa en un lenguaje .NET si:

- Desea encapsular lógica compleja en una rutina que acceda a la base de datos o que realice una acción fuera de la base de datos.
- Necesita que la lógica encapsulada se invoque desde cualquiera de estos elementos: diversas aplicaciones, el CLP, otra rutina (procedimiento, función (UDF) o método) o un activador.
- Se siente más cómodo al codificar esta lógica en un lenguaje .NET.

Requisitos previos:

- Conocimiento de la implementación de rutinas CLR. Para informarse sobre las rutinas CLR en general y sobre las características CLR, consulte:
  - "Rutinas de ejecución en el lenguaje común (CLR) .NET" en la página 63
- El servidor de bases de datos debe ejecutar un sistema operativo Windows que dé soporte a Microsoft .NET Framework.
- El producto .NET Framework, versión 1.1 o 2.0, debe estar instalado en el servidor. .NET Framework está disponible de forma independiente o formando parte del Kit de desarrollo de software Microsoft .NET Framework 1.1 o del Kit de desarrollo de software .NET Framework 2.0.
- Se deben instalar las versiones siguientes de DB2:
  - Servidor: DB2 8.2 o un release posterior.
  - Cliente: Cualquier cliente que se pueda conectar a una instancia de DB2 8.2 será capaz de invocar una rutina CLR. Es recomendable instalar DB2 Versión 7.2 o un release posterior en el cliente.
- Autorización para ejecutar la sentencia CREATE correspondiente a la rutina externa. Si desea saber cuáles son los privilegios necesarios para ejecutar la sentencia CREATE PROCEDURE o CREATE FUNCTION, consulte los detalles relativos a la sentencia pertinente.

Restricciones:

Para obtener una lista de las restricciones asociadas con rutinas CLR, consulte:

- "Restricciones de las rutinas CLR .NET" en la página 74

Procedimiento:

1. Codifique la lógica de la rutina en cualquier lenguaje CLR soportado.
   - Para obtener información general sobre las rutinas .NET CLR y sus características, consulte los temas referidos en el apartado Requisitos previos.
   - Utilice o importe el conjunto `IBM.Data.DB2` si la rutina ha de ejecutar SQL.
   - Declare las variables del lenguaje principal y los parámetros correctamente utilizando tipos de datos que se correlacionen con tipos de datos de SQL de DB2. Para conocer la correlación entre los tipos de datos de DB2 y .NET:
     - "Tipos de datos de SQL soportados para el Proveedor de datos DB2 .NET" en la página 66
   - Los parámetros y los indicadores de nulo de parámetro se deben declarar utilizando uno de los estilos de parámetros soportados por DB2 y de acuerdo con los requisitos de los parámetros de las rutinas .NET CLR. Asimismo, las

áreas reutilizables para las UDF y la clase DBINFO pasan a las rutinas CLR como parámetros. Si desea más información sobre los parámetros y las declaraciones de prototipo, consulte:

– "Parámetros de rutinas .NET CLR" en la página 68

- Si la rutina es un procedimiento y desea devolver un conjunto de resultados al llamador de la rutina, no es necesario ningún parámetro para el conjunto de resultados. Si desea más información sobre la devolución de conjuntos de resultados desde rutinas CLR:
  – "Devolución de conjuntos de resultados desde procedimientos .NET CLR" en la página 71
- Establezca un valor de retorno de rutina si es necesario. Las funciones escalares CLR requieren que se establezca un valor de retorno antes de la devolución. Las funciones de tabla CLR requieren que se especifique un código de retorno como parámetro de salida para cada invocación de la función de tabla. Los procedimientos CLR no realizan una devolución con un valor de retorno.

2. Cree el código en un conjunto de lenguaje intermedio (IL) para que se ejecute mediante la CLR. A fin de obtener información sobre cómo crear rutinas CLR .NET que accedan a DB2, consulte el enlace relacionado:
   - Creación de rutinas de ejecución en el lenguaje común
3. Copie el conjunto en el *directorio de función* de DB2 del servidor de bases de datos. Es recomendable almacenar conjuntos o bibliotecas asociadas con rutinas de DB2 en el directorio de función. Para conocer más acerca del directorio de función, consulte la cláusula EXTERNAL de una de las sentencias siguientes: CREATE PROCEDURE o CREATE FUNCTION.

   Puede copiar el conjunto en otro directorio del servidor si lo desea, pero, para invocar satisfactoriamente la rutina, debe anotar el nombre de vía de acceso completamente calificado del conjunto porque lo necesitará en el paso siguiente.
4. Ejecute de forma dinámica o estática la sentencia CREATE de lenguaje SQL correspondiente para el tipo de rutina: CREATE PROCEDURE o CREATE FUNCTION.
   - Especifique la cláusula `LANGUAGE` con el valor: `CLR`.
   - Especifique la cláusula `PARAMETER STYLE` con el nombre del estilo de parámetros soportado que se ha implementado en el código de la rutina.
   - Especifique la cláusula `EXTERNAL` con el nombre del conjunto que se ha de asociar con la rutina utilizando uno de los valores siguientes:
     – el nombre de vía de acceso completamente calificado del conjunto de rutinas.
     – el nombre de vía de acceso relativo del conjunto de rutinas en relación con el directorio de función.

     Por omisión, DB2 buscará el conjunto por el nombre en el directorio de función, a menos que se especifique un nombre de vía de acceso completamente calificado o relativo para la biblioteca en la cláusula EXTERNAL.

     Cuando se ejecute la sentencia CREATE, si DB2 no encuentra el conjunto especificado en la cláusula EXTERNAL, se recibirá un error (SQLCODE -20282) con el código de razón 1.
   - Especifique la cláusula DYNAMIC RESULT SETS con un valor entero que equivalga al número máximo de conjuntos de resultados que debe devolver la rutina.

- No es posible especificar la cláusula NOT FENCED para los procedimientos CLR. Por omisión, los procedimientos CLR se ejecutan como procedimientos FENCED.

Para invocar la rutina CLR, consulte el tema ″Invocación de rutinas″.

Conceptos relacionados:

- "Rutinas de ejecución en el lenguaje común (CLR) .NET" en la página 63
- "Parámetros de rutinas .NET CLR" en la página 68
- "Estilos de parámetros de rutinas externas" en la página 46
- "Áreas reutilizables para funciones externas y métodos" en la página 33
- "SQL en rutinas externas" en *Desarrollo de SQL y rutinas externas*

Tareas relacionadas:

- "Devolución de conjuntos de resultados desde procedimientos .NET CLR" en la página 71
- "Creación de rutinas CLR (Common Language Runtime) .NET" en la página 80
- "Creación de rutinas .NET CLR" en la página 75
- "Depuración de rutinas" en *Desarrollo de SQL y rutinas externas*

Información relacionada:

- "Restricciones de las rutinas CLR .NET" en la página 74
- "Tipos de datos de SQL soportados para el Proveedor de datos DB2 .NET" en la página 66

Ejemplos relacionados:

- "SpCreate.db2 -- Creates the external procedures implemented in spserver.cs"
- "SpServer.cs -- C# external code implementation of procedures created in spcat.db2"
- "SpCreate.db2 -- Creates the external procedures implemented in spserver.vb"
- "SpServer.vb -- VB.NET implementation of procedures created in SpCat.db2"

Creación de código de rutinas .NET CLR

Creación de código de rutinas .NET CLR

Una vez se ha escrito el código de implementación de rutina .NET, se debe crear para que el conjunto de rutinas se pueda desplegar y la rutina se pueda invocar. Los pasos a seguir para crear rutinas .NET CLR son parecidos a los necesarios para crear cualquier rutina externa, aunque hay algunas diferencias.

Hay tres formas de crear rutinas .NET CLR:

- Mediante las herramientas gráficas que se proporcionan con DB2 Database Development Add-In para Visual Studio .NET 1.2
- Mediante archivos de proceso por lotes de ejemplo de DB2
- Entrando mandatos desde una ventana de mandatos de DB2

Los scripts de creación de ejemplo de DB2 y los archivos de proceso por lotes correspondientes a rutinas están diseñados para crear rutinas de ejemplo de DB2

(procedimientos y funciones definidas por el usuario), así como rutinas creadas por el usuario para un determinado sistema operativo utilizando los compiladores soportados por omisión.

Hay un conjunto independiente de scripts de creación de ejemplo de DB2 y de archivos de proceso por lotes para C# y Visual Basic. En general, es más fácil crear rutinas de .NET CLR incorporado utilizando las herramientas gráficas o los scripts de creación, que se pueden modificar fácilmente si hace falta; sin embargo, suele resultar útil saber también cómo crear rutinas a partir de los mandatos de Windows de DB2.

Conceptos relacionados:

- "Rutinas de ejecución en el lenguaje común (CLR) .NET" en la página 63
- "Creación de código de rutinas .NET CLR (ejecución en lenguaje común) mediante scripts de creación de ejemplo" en la página 83

Tareas relacionadas:

- "Creación de código de rutinas .NET CLR (Ejecución en el lenguaje común) desde las ventanas de mandatos de DB2" en la página 85

Creación de rutinas CLR (Common Language Runtime) .NET

Los productos de cliente y servidor DB2 proporcionan archivos de proceso por lotes para compilar y enlazar programas .NET DB2. Estos archivos están ubicados en los directorios `sqllib\samples\NET\cs` y `sqllib\samples\.NET\vb`, junto con programas de ejemplo que se pueden crear con esos archivos.

El archivo de proceso por lotes `bldrtn.bat` contiene los mandatos para crear rutinas (procedimientos almacenados y funciones definidas por el usuario) CLR. El archivo de proceso por lotes crea una DLL de conjunto .NET en el servidor. Utiliza dos parámetros como entrada, que están representados dentro del archivo de proceso por lotes por las variables `%1` y `%2`.

El primer parámetro, `%1`, especifica el nombre del archivo fuente. El archivo de proceso por lotes utiliza el nombre del archivo fuente para el nombre de la DLL de conjunto. El segundo parámetro, `%2`, especifica el nombre de la base de datos a la que desea conectarse. Puesto que la DLL de conjunto se debe crear en la misma instancia donde reside la base de datos, no hay parámetros para el ID de usuario ni la contraseña.

Sólo es obligatorio el primer parámetro, el nombre del archivo fuente. El nombre de la base de datos es opcional. Si no se proporciona un nombre de base de datos, el programa utiliza la base de datos por omisión `sample`.

Prerrequisitos:

El servidor de bases de datos debe ejecutar un sistema operativo Windows instalado con Microsoft .NET Framework Versión 1.1 o .NET Framework 2.0. El producto .NET Framework está disponible de forma independiente o como parte de Microsoft .NET Framework Software Development Kit.

Se deben instalar las versiones siguientes de DB2:

Servidor:
DB2 8.2 o posteriores

Cliente:
DB2 7.2 o posteriores

Se debe otorgar autorización para ejecutar la sentencia CREATE para la rutina. Para conocer los privilegios requeridos para ejecutar la sentencia CREATE, vea la sentencia CREATE para el tipo de rutina: CREATE PROCEDURE o CREATE FUNCTION.

Procedimiento:

Los ejemplos siguientes muestran cómo crear las DLL de conjunto de rutinas mediante procedimientos almacenados y funciones definidas por el usuario.

DLL de conjunto de procedimiento almacenado

Para crear la DLL de conjunto `SpServer` a partir del archivo fuente VB .NET, `SpServer.vb`, o a partir del archivo fuente C#, `SpServer.cs`:

1. Escriba el nombre del archivo de proceso por lotes y el nombre del programa (sin la extensión):

   ```
   bldrtn SpServer
   ```

   Si conecta con una base de datos que no sea la base de datos por omisión, `sample`, especifique también el nombre de la base de datos:

   ```
   bldrtn SpServer basedatos
   ```

   El archivo de proceso por lotes copia la DLL de conjunto, `SpServer.dll`, en el directorio `sqllib\function`.
2. A continuación, catalogue las rutinas ejecutando el script `spcat` en el servidor:

   ```
   SpCat
   ```

   Este script conecta con la base de datos "sample", descataloga mediante `SpDrop.db2` las rutinas que se hubieran catalogado previamente, luego las cataloga llamando a `SpCreate.db2`, y finalmente desconecta de la base de datos. También puede llamar a los scripts `SpDrop.db2` y `SpCreate.db2` por separado.
3. A continuación, a menos que sea la primera vez que se ha creado la DLL de conjunto, detenga y rearranque la base de datos para que se reconozca la nueva versión de la DLL de conjunto. Si es necesario, establezca la modalidad de archivo para la DLL de conjunto de forma que la instancia de DB2 pueda acceder a ella.

Una vez creada la DLL de conjunto, `SpServer`, puede crear la aplicación cliente `SpClient`, que la llama.

Puede crear `SpClient` utilizando el archivo de proceso por lotes, `bldapp.bat`.

Para asegurarse de tener los parámetros que necesitará cuando ejecute el ejecutable, puede especificar distintas combinaciones de parámetros, en lugar de aceptar los valores por omisión, en función del número de parámetros que haya entrado:

1. Ningún parámetro. Entre sólo el nombre de programa (para llamar localmente en la instancia del servidor):

   ```
   SpClient
   ```

2. Un parámetro. Entre el nombre del programa más el alias de la base de datos (para llamar a una base de datos distinta de la base de datos sample localmente en la instancia del servidor):

   ```
   SpClient <alias_bd>
   ```

3. Tres parámetros. Entre el nombre del programa más el alias de la base de datos, el ID de usuario y la contraseña (para llamar desde un cliente remoto):

   ```
   SpClient <alias_bd> <IDusuario> <contraseña>
   ```

4. Cinco parámetros. Entre el nombre del programa más el alias de la base de datos, el nombre del servidor, el número de puerto, el ID de usuario y la contraseña (para llamar desde un cliente remoto):

   ```
   SpClient <alias_bd> <servidor> <númeropuerto> <IDusuario> <contraseña>
   ```

La aplicación cliente accede a la DLL de conjunto, SpServer y ejecuta varias rutinas contenidas en la base de datos del servidor. Los datos resultantes se devuelven a la aplicación cliente.

DLL de conjunto de función definida por el usuario

Para crear la DLL de conjunto de función definida por el usuario UDFsrv a partir del archivo fuente VB .NET, UDFsrv.vb, o a partir del archivo fuente C#, UDFsrv.cs:

```
bldrtn UDFsrv
```

Si conecta con una base de datos que no sea la base de datos por omisión, sample, especifique también el nombre de la base de datos:

```
bldrtn UDFsrv basedatos
```

El archivo de proceso por lotes copia la DLL de conjunto de función definida por el usuario, UDFsrv.dll, en el directorio sqllib\function.

Una vez creada UDFsrv, puede crear la aplicación cliente, udfcli, que la llama.

Puede crear UDFcli utilizando el archivo de proceso por lotes, bldapp.bat.

Para asegurarse de tener los parámetros que necesitará cuando ejecute el ejecutable, puede especificar distintas combinaciones de parámetros, en lugar de aceptar los valores por omisión, en función del número de parámetros que haya entrado:

1. Ningún parámetro. Entre sólo el nombre de programa (para llamar localmente en la instancia del servidor):

   ```
   UDFcli
   ```

2. Un parámetro. Entre el nombre del programa más el alias de la base de datos (para llamar a una base de datos distinta de la base de datos sample localmente en la instancia del servidor):

   ```
   UDFcli <alias_bd>
   ```

3. Tres parámetros. Entre el nombre del programa más el alias de la base de datos, el ID de usuario y la contraseña (para llamar desde un cliente remoto):

   ```
   UDFcli <alias_bd> <IDusuario> <contraseña>
   ```

4. Cinco parámetros. Entre el nombre del programa más el alias de la base de datos, el nombre del servidor, el número de puerto, el ID de usuario y la contraseña (para llamar desde un cliente remoto):

   ```
   UDFcli <alias_bd> <servidor> <númpuerto> <IDusuario> <contraseña>
   ```

La aplicación llamadora invoca la función ScalarUDF de la DLL de conjunto udfsrv.

Conceptos relacionados:

- "Rutinas de ejecución en el lenguaje común (CLR) .NET" en la página 63

Ejemplos relacionados:

- "bldrtn.bat -- Builds C# routines (stored procedures and UDFs)"
- "SpServer.cs -- C# external code implementation of procedures created in spcat.db2"
- "SpClient.cs -- Call different types of stored procedures from SpServer.java"
- "UDFcli.cs -- Client application that calls the user-defined functions "
- "UDFsrv.cs -- User-defined scalar functions called by udfcli.cs"
- "bldrtn.bat -- Builds Visual Basic .NET routines (stored procedures and UDFs)"
- "SpServer.vb -- VB.NET implementation of procedures created in SpCat.db2"
- "SpClient.vb -- Call different types of stored procedures from SpServer.java"
- "UDFcli.vb -- Client application that calls the user-defined functions "
- "UDFsrv.vb -- User-defined scalar functions called by udfcli.vb "

Creación de código de rutinas .NET CLR (ejecución en lenguaje común) mediante scripts de creación de ejemplo

El proceso de construir código de rutinas .NET CLR (ejecución en lenguaje común) es una subtarea de la tarea de crear rutinas .NET CLR. Esta tarea se puede realizar de forma rápida y fácil mediante los archivos por lotes de ejemplo de DB2. Los scripts de creación de ejemplo se pueden utilizar para el código fuente con sentencias SQL o sin ellas. Los scripts de creación se encargan de compilar, enlazar y desplegar el ensamblado construido en el `directorio function`.

Como alternativas, puede simplificar la tarea de construir código de rutinas .NET CLR haciéndolo en Visual Studio .NET o siguiendo manualmente los pasos de los scripts de creación de ejemplo de DB2. Consulte:

- Creación de rutinas .NET CLR (ejecución en lenguaje común) en Visual Studio .NET
- Creación de rutinas .NET CLR (ejecución en lenguaje común) utilizando ventanas de mandatos de DB2

Los scripts de creación de ejemplo específicos del lenguaje de programación para construir rutinas .NET CLR en C# y Visual Basic se llaman `bldrtn`. Se encuentran en directorios de DB2 junto con los programas de ejemplo que se pueden crear con ellos, en las siguientes ubicaciones:

- Para C: `sqllib/samples/cs/`
- Para C++: `sqllib/samples/vb/`

Los scripts `bldrtn` se pueden utilizar para construir archivos de código fuente que contengan procedimientos y funciones definidas por el usuario. El script hace lo siguiente:

- Establece una conexión con una base de datos especificada por el usuario
- Compila y enlaza el código fuente para generar un ensamblado con un sufijo de archivo .DLL
- Copia el ensamblado en el directorio function de DB2 en el servidor de base de datos

Los scripts `bldrtn` aceptan dos argumentos:

- El nombre de un archivo de código fuente sin sufijo de archivo
- El nombre de una base de datos con la que se establecerá una conexión

El parámetro correspondiente a la base de datos es opcional. Si no se proporciona un nombre de base de datos, el programa utiliza la base de datos por omisión sample. Puesto que las rutinas se tienen que crear en la misma instancia en la que reside la base de datos, no se necesitan argumentos para ID de usuario y contraseña.

Prerrequisitos:

- Hay que satisfacer los requisitos del sistema operativo y software de desarrollo de las rutinas .NET CLR. Consulte: ″Soporte del desarrollo de rutinas .NET CLR″.
- Archivo de código fuente que contenga una o más implementaciones de rutinas.
- El nombre de la base de datos dentro de la instancia de DB2 actual en la que se van a crear las rutinas.

Procedimiento:

Para crear un archivo de código fuente que contenga una o más implementaciones de código de rutina, siga los pasos siguientes.

1. Abra una ventana de mandatos de DB2.
2. Copie el archivo de código fuente en el mismo directorio que el script `bldrtn`.
3. Si las rutinas se van a crear en la base de datos sample, teclee el nombre del script de creación seguido del nombre del archivo de código fuente sin la extensión de archivo .cs o .vb.

   ```
   bldrtn <nombre-archivo>
   ```

   Si las rutinas se van a crear en otra base de datos, teclee el nombre del script de creación, el nombre del archivo de código fuente sin extensión de archivo y el nombre de la base de datos:

   ```
   bldrtn <nombre-archivo> <nombre-basedatos>
   ```

   El script compila y enlaza el código fuente y produce un ensamblado. Luego, el script copia el ensamblado en el directorio function del servidor de base de datos.
4. Si no es la primera vez que se crea el archivo de código fuente que contiene las implementaciones de rutinas, detenga y vuelva a iniciar la base de datos para asegurarse de que DB2 utiliza la nueva versión de la biblioteca compartida. Lo puede hacer entrando en la línea de mandatos `db2stop` seguido de `db2start`.

Cuando haya creado satisfactoriamente la biblioteca compartida de rutinas y la haya desplegado en el directorio de función en el servidor de bases de datos, debe completar los pasos asociados a la tarea de crear rutinas C y C++.

La creación de rutinas .NET CLR incluye un paso para ejecutar la sentencia CREATE para cada rutina implementada en el archivo de código fuente. Una vez completada la creación de rutinas, puede invocar las rutinas.

Creación de código de rutinas .NET CLR (Ejecución en el lenguaje común) desde las ventanas de mandatos de DB2

La creación de código fuente de rutinas .NET CLR es una subtarea de la creación de rutinas .NET CLR. Esta tarea se puede realizar manualmente desde las ventanas de mandatos de DB2. Se puede seguir el mismo procedimiento, independientemente de si hay o no sentencias de SQL dentro del código de la rutina. Los pasos de la tarea incluyen la compilación de código fuente escrito en un lenguaje de programación soportado por .NET CLR en un conjunto con un sufijo de archivo .DLL.

Como alternativas, puede simplificar la tarea creando código de rutinas .NET CLR haciéndolo en Visual Studio .NET o utilizando scripts de creación de ejemplo de DB2. Consulte:

- Creación de rutinas .NET CLR (Ejecución en el lenguaje común) en Visual Studio .NET
- Creación de rutinas .NET CLR (Ejecución en el lenguaje común) utilizando scripts de creación de ejemplo

Requisitos previos:

- Se han satisfecho los requisitos previos de sistema operativo necesario y software de desarrollo de rutinas .NET CLR. Consulte: ″Soporte del desarrollo de rutinas .NET CLR″.
- El código fuente se ha escrito en un lenguaje de programación de .NET CLR soportado que contiene una o más implementaciones de rutinas .NET CLR.
- El nombre de la base de datos dentro de la instancia de DB2 actual en la que se van a crear las rutinas.
- Las opciones de compilación y enlace específicas del sistema operativo necesarias para crear rutinas .NET CLR.

Procedimiento:

Para crear un archivo de código fuente que contenga una o más implementaciones de código de rutinas .NET CLR siga los pasos siguientes. A continuación se describe un ejemplo que muestra cada uno de los pasos:

1. Abra una ventana de mandatos de DB2.
2. Navegue hasta el directorio que contiene el archivo de código fuente.
3. Establezca una conexión con la base de datos en la que se van a crear las rutinas.
4. Compile el archivo de código fuente.
5. Enlace el archivo de código fuente para generar una biblioteca compartida. Para ello hay que utilizar algunas opciones de compilación y enlace específicas de DB2.
6. Copie el archivo de conjunto con el sufijo de archivo .DLL en el directorio de función de DB2 en el servidor de bases de datos.
7. Si no es la primera vez que se crea el archivo de código fuente que contiene las implementaciones de rutinas, detenga y vuelva a iniciar la base de datos para asegurarse de que DB2 utiliza la nueva versión de la biblioteca compartida. Puede hacerlo emitiendo el mandato db2stop seguido del mandato db2start.

Cuando haya creado y desplegado satisfactoriamente la biblioteca de rutinas, debe completar los pasos asociados a la tarea de crear rutinas .NET CLR. La creación de

rutinas .NET CLR incluye un paso para ejecutar la sentencia CREATE para cada rutina implementada en el archivo de código fuente. Este paso también de debe completar para poder invocar las rutinas.

Ejemplo:

El siguiente ejemplo muestra la recreación de un archivo de código fuente .NET CLR. Se muestran los pasos para un archivo de código Visual Basic denominado `myVBfile.vb` que contiene implementaciones de rutinas y para un archivo de código C# denominado `myCSfile.cs`. Las rutinas se crean en un sistema operativo Windows 2000 y se utiliza Microsoft .NET Framework 1.1 para generar un conjunto de 64 bits.

1. Abra una ventana de mandatos de DB2.
2. Navegue hasta el directorio que contiene el archivo de código fuente.
3. Establezca una conexión con la base de datos en la que se van a crear las rutinas.

   ```
   db2 connect to <nombre-basedatos>
   ```

4. Compile el archivo de código fuente mediante las opciones recomendadas de compilación y enlace (donde $DB2PATH es la vía de acceso de instalación de la instancia de DB2. Sustituya este valor antes de ejecutar el mandato):

   ```
   Ejemplo en C#
   ==================
   csc /out:myCSfile.dll /target:library
       /reference:$DB2PATH%\bin\netf11\IBM.Data.DB2.dll myCSfile.cs

   Ejemplo en Visual Basic
   =======================
   vbc /target:library /libpath:$DB2PATH\bin\netf11
       /reference:$DB2PATH\bin\netf11\IBM.Data.DB2.dll
       /reference:System.dll
       /reference:System.Data.dll myVBfile.vb
   ```

   El compilador generará salida si se produce algún error. Este paso genera un archivo de exportación denominado myfile.exp.

5. Copie la biblioteca compartida en el directorio de función de DB2 en el servidor de bases de datos.

   ```
   Ejemplo en C#
   ====================
   rm -f ~HOME/sqllib/function/myCSfile.DLL
   cp myCSfile $HOME/sqllib/function/myCSfile.DLL

   Ejemplo en Visual Basic
   =======================
   rm -f ~HOME/sqllib/function/myVBfile.DLL
   cp myVBfile $HOME/sqllib/function/myVBfile.DLL
   ```

   Este paso asegura que la biblioteca de la rutina está en el directorio por omisión en el que DB2 busca bibliotecas de rutinas. Consulte el tema sobre la creación de rutinas .NET CLR para obtener más información sobre el despliegue de bibliotecas de rutinas.

6. Detenga y vuelva a iniciar la base de datos puesto que se trata de una recreación de un archivo de código fuente de rutina anteriormente creado.

   ```
   db2stop
   db2start
   ```

La creación de rutinas .NET CLR suele resultar más sencilla si se utilizan los scripts de creación de ejemplo específicos del sistema operativo, que también se pueden utilizar como referencia para ver cómo crear rutinas desde la línea de mandatos.

Opciones de compilación y enlace para rutinas de CLR .NET

A continuación se muestran las opciones de compilación y enlace que DB2 recomienda para crear rutinas de Common Language Runtime (CLR) .NET en Windows con el compilador Microsoft Visual Basic .NET o con el compilador Microsoft C#, tal como muestran los archivos de proceso por lotes `samples\.NET\cs\bldrtn.bat` y `samples\.NET\vb\bldrtn.bat`.

Opciones de compilación y enlace para bldrtn
Opciones de compilación y enlace utilizando el compilador Microsoft C#: **csc** El compilador de Microsoft C#. **/out:%1.dll /target:library** Produce como salida la biblioteca de enlace de datos en forma de dll de conjunto de procedimientos almacenados. **/debug** Utilizar el depurador. **/lib: "%DB2PATH%"\bin\netf11** Utilizar la vía de acceso de bibliotecas para .NET Framework versión 1.1. **/reference:IBM.Data.DB2.dll** Utilizar la biblioteca de enlace de datos de DB2 para .NET Framework versión 1.1. Consulte la documentación del compilador para conocer otras opciones de compilador.
Opciones de compilación y enlace utilizando el compilador Microsoft Visual Basic .NET: **vbc** El compilador de Microsoft Visual Basic .NET. **/out:%1.dll /target:library** Produce como salida la biblioteca de enlace de datos en forma de dll de conjunto de procedimientos almacenados. **/debug** Utilizar el depurador. **/libpath:"%DB2PATH%"\bin\netf11** Utilizar la vía de acceso de bibliotecas para .NET Framework versión 1.1. **/reference:IBM.Data.DB2.dll** Utilizar la biblioteca de enlace de datos de DB2 para .NET Framework versión 1.1. **/reference:System.dll** Referencia a la biblioteca de enlace de datos de Microsoft Windows System. **/reference:System.Data.dll** Referencia a la biblioteca de enlace de datos de Microsoft Windows System Data. Consulte la documentación del compilador para conocer otras opciones de compilador.

Tareas relacionadas:

- "Creación de rutinas CLR (Common Language Runtime) .NET" en la página 80

Ejemplos relacionados:

- "bldrtn.bat -- Builds C# routines (stored procedures and UDFs)"
- "bldrtn.bat -- Builds Visual Basic .NET routines (stored procedures and UDFs)"

Depuración de rutinas .NET CLR

Depuración de rutinas .NET CLR

Es posible que sea necesario depurar rutinas .NET CLR si no se puede crear una rutina, invocar una rutina o si al invocar una rutina, ésta no se comporta o ejecuta como se esperaba.

Procedimiento:

Tome en consideración lo siguiente al depurar rutinas .NET CLR:

- Compruebe si se está utilizando un sistema operativo soportado para el desarrollo de la rutina .NET CLR.
- Compruebe si se están utilizando un servidor de base de datos DB2 soportado y un cliente CB2 para el desarrollo de la rutina .NET CLR.
- Compruebe si se está utilizando el software de desarrollo de Microsoft .NET Framework soportado.
- Si ha fallado la creación de la rutina:
  - Compruebe si el usuario tiene la necesaria autorización y privilegios para ejecutar la sentencia CREATE PROCEDURE o CREATE FUNCTION.
- Si ha fallado la invocación de la rutina:
  - Compruebe si el usuario tiene autorización para ejecutar la rutina. Si se ha producido un error (SQLCODE -551, SQLSTATE 42501), esto es probable ya que el invocador carece del privilegio EXECUTE sobre la rutina. El que puede otorgar este privilegio es un usuario con autorización SYSADM, DBADM o bien el definidor de la rutina.
  - Compruebe que la signatura del parámetro de rutina utilizada en la sentencia CREATE para la rutina se corresponde con la signatura del parámetro de rutina de la implementación de la rutina.
  - Compruebe que los tipos de datos utilizados en la implementación de la rutina son compatibles con los tipos de datos especificados en la signatura del parámetro de rutina de la sentencia CREATE.
  - Compruebe que en la implementación de la rutina, sean válidas las palabras clave específicas del lenguaje de .NET CLR utilizadas para indicar el método por el que debe pasarse el parámetro (por valor o referencia).
  - Compruebe que el valor especificado en la cláusula EXTERNAL de la sentencia CREATE PROCEDURE o CREATE FUNCTION se corresponde con la ubicación en la que está ubicado el conjunto .NET CLR que contiene la implementación de la rutina en el sistema de archivos del sistema en el que se ha instalado el servidor de base de datos DB2.
  - Si la rutina es una función, compruebe si todos los tipos de llamada aplicables se han programado correctamente en la implementación de la rutina. Esto es particularmente importante si la rutina se ha definido con la cláusula FINAL CALL.
- Si la rutina no se está comportando como se esperaba:
  - Modifique la rutina de modo que dé salida a la información de diagnóstico en un archivo ubicado en un directorio accesible globalmente. Dar salida a la información de diagnóstico en la pantalla no es posible desde las rutinas .NET CLR. No dirija la salida a archivos que estén en directorios utilizados por bases de datos DB2 o gestores de base de datos DB2.
  - Depure la rutina localmente escribiendo una simple aplicación .NET que invoque de forma directa al punto de entrada de la rutina. Para obtener más

información sobre el modo de depurar características en Microsoft Visual Studio .NET, consulte la documentación del compilador .NET de Microsoft Visual Studio.

Para obtener más información sobre errores comunes relacionados con la invocación y creación de rutinas .NET CLR, consulte:

•

Conceptos relacionados:

- "Rutinas de ejecución en el lenguaje común (CLR) .NET" en la página 63
- "Restricciones de las rutinas externas" en la página 57

Tareas relacionadas:

- "Depuración de las rutinas de CLR .NET desde la línea de mandatos" en *Desarrollo de SQL y rutinas externas*

Información relacionada:

- "Errores relacionados con rutinas CLR .NET" en la página 89
- "Restricciones de las rutinas CLR .NET" en la página 74

Errores relacionados con rutinas CLR .NET

Aunque las rutinas externas comparten una implementación común en general, se pueden producir algunos errores de DB2 específicos de las rutinas CLR. Esta consulta lista los errores de CLR .NET más probables que se pueden encontrar, listados por el SQLCODE o comportamiento, con algunas sugerencias para la depuración. Los errores de DB2 relacionados con rutinas se pueden clasificar del modo siguiente:

Errores de creación de la rutina
Errores que se producen cuando se ejecuta la sentencia CREATE para la rutina.

Errores de ejecución de la rutina
Errores que se producen durante la invocación o ejecución de la rutina.

Independientemente de cuándo DB2 emite un error relacionado con una rutina de DB2, el texto del mensaje de error detalla la causa del error y la acción que el usuario debe emprender para resolver el problema. Hallará información adicional sobre escenarios de los errores de rutinas en el archivo de registro de diagnósticos `db2diag.log`.

Errores de creación de rutinas CLR:

SQLCODE -451, SQLSTATE 42815
Este error se emite tras un intento de ejecutar una sentencia CREATE TYPE que incluye una declaración de método externo especificando la cláusula LANGUAGE con el valor CLR. No se pueden crear métodos externos de DB2 para tipos estructurados que hagan referencia a un conjunto de CLR en este momento. Cambie la cláusula LANGUAGE de forma que especifique un lenguaje soportado para el método e implemente el método en ese lenguaje alternativo.

SQLCODE -449, SQLSTATE 42878
La sentencia CREATE para la rutina CLR contiene una identificación de función o biblioteca de formato no válido en la cláusula EXTERNAL

NAME. Para el lenguaje CLR, el valor de la cláusula EXTERNAL debe tomar el formato específico '<a>:<b>!<c>', del modo siguiente:

- <a> es el archivo de conjunto de CLR en el que está ubicada la clase.
- <b> es la clase en la que reside el método a invocar.
- <c> es el método a invocar.

No se permiten caracteres en blanco iniciales ni de cola entre las comillas simples, identificadores de objeto y caracteres de separación (por ejemplo, ' <a> ! <b> ' no es válido). Sin embargo, los nombres de vía de acceso y archivo pueden contener espacios en blanco si la plataforma lo permite. Para todos los nombres de archivo, el archivo se puede especificar utilizando la forma abreviada del nombre (ejemplo: math.dll) o el nombre de vía de acceso completamente calificado (ejemplo: d:\udfs\math.dll). Si se utiliza la forma abreviada del nombre de archivo, si la plataforma es UNIX o si la rutina es LANGUAGE CLR, el archivo debe residir en el directorio de función. Si la plataforma es Windows y la rutina no es una rutina LANGUAGE CLR, el archivo debe residir en la vía de acceso (PATH) del sistema. Las extensiones de archivo (ejemplos: .a (en UNIX), .dll (en Windows)) siempre se deben incluir en el nombre de archivo.

Errores de ejecución de rutinas CLR:

SQLCODE -20282, SQLSTATE 42724, código de razón 1
No se ha encontrado el conjunto externo especificado por la cláusula EXTERNAL en la sentencia CREATE para la rutina.

- Compruebe si la cláusula EXTERNAL especifica el nombre correcto de conjunto de la rutina y si el conjunto se encuentra en la ubicación especificada. Si la cláusula EXTERNAL no especifica un nombre de vía de acceso completamente calificado para el conjunto deseado, DB2 supone que el nombre de vía de acceso que se proporciona es un nombre de vía de acceso relativo para el conjunto, en relación con el directorio de función de DB2.

SQLCODE -20282, SQLSTATE 42724, código de razón 2
Se ha encontrado un conjunto en la ubicación especificada por la cláusula EXTERNAL de la sentencia CREATE para la rutina, pero, en el conjunto, no se ha encontrado ninguna clase que coincida con la clase especificada en la cláusula EXTERNAL.

- Compruebe si el nombre de conjunto especificado en la cláusula EXTERNAL es el conjunto correcto para la rutina y si existe en la ubicación especificada.
- Compruebe si el nombre de clase especificado en la cláusula EXTERNAL es el nombre de clase correcto y si existe en el conjunto especificado.

SQLCODE -20282, SQLSTATE 42724, código de razón 3
Se ha encontrado un conjunto en la ubicación especificada por la cláusula EXTERNAL de la sentencia CREATE para la rutina, el cual tenía una definición de clase que coincidía correctamente, pero la signatura del método de la rutina no coincide con la signatura de la rutina especificada en la sentencia CREATE para la rutina.

- Compruebe si el nombre de conjunto especificado en la cláusula EXTERNAL es el conjunto correcto para la rutina y si existe en la ubicación especificada.
- Compruebe si el nombre de clase especificado en la cláusula EXTERNAL es el nombre de clase correcto y si existe en el conjunto especificado.

- Compruebe si la implementación del estilo de parámetros coincide con el estilo de parámetros especificado en la sentencia CREATE para la rutina.
- Compruebe si el orden de la implementación de parámetros coincide con el orden de la declaración de parámetros de la sentencia CREATE para la rutina y si se respetan los requisitos adicionales de parámetros del estilo de parámetros.
- Compruebe si los tipos de datos de los parámetros de SQL están correlacionados correctamente con los tipos de datos soportados en CLR .NET.

SQLCODE -4301, SQLSTATE 58004, código de razón 5 ó 6

Se ha producido un error al intentar el inicio o la comunicación de un intérprete de .NET. DB2 no ha podido cargar una biblioteca .NET dependiente [código de razón 5] o ha fallado una llamada al intérprete de .NET [código de razón 6].

- Asegúrese de que la instancia de DB2 está configurada correctamente para ejecutar una función o procedimiento .NET (`mscoree.dll` debe estar presente en la vía de acceso (`PATH`) del sistema). Asegúrese de que `db2clr.dll` está presente en el directorio `sqllib/bin` y de que `IBM.Data.DB2` está instalado en la antememoria del conjunto global. Si no están, asegúrese de que `.NET Framework` versión 1.1, o una versión posterior, se haya instalado en el servidor de bases de datos y de que dicho servidor esté ejecutando DB2 versión 8.2 o un release posterior.

SQLCODE -4302, SQLSTATE 38501

Se ha producido una excepción no controlada durante la ejecución, en la preparación de la ejecución o después de ejecutar la rutina. Puede ser resultado de un error de programación de la lógica de una rutina no controlado o puede ser resultado de un error de proceso interno. Para los errores de este tipo, el rastreo posterior de pila .NET que indica el lugar en el que la excepción no manejada producida se grabará en el db2diag.log.

Este error también puede producirse en el caso de que la rutina haya intentado realizar una acción que no esté incluida dentro del ámbito de las acciones permitidas para la modalidad de ejecución especificada para la rutina. En este caso, se efectuará una entrada en db2diag.log indicando específicamente que la excepción producida debido a una violación de control de ejecución. También se incluirá el rastreo posterior de la pila de excepción que indica el lugar en el que se ha producido la violación.

Determine si se ha comprometido o se ha modificado recientemente el conjunto de la rutina. Si la rutina se ha modificado de forma válida, este problema puede deberse a que la modalidad EXECUTION CONTROL para la rutina ya no esté definida en una modalidad adecuada para la lógica modificada. Si está seguro de que el conjunto no se ha manipulado incorrectamente, puede modificar la modalidad de ejecución de la rutina con la sentencia ALTER PROCEDURE o ALTER FUNCTION, según sea adecuado. Para obtener más información, consulte el tema siguiente:

- "Modalidades de seguridad y de ejecución para rutinas CLR" en la página 73

Conceptos relacionados:

- "Rutinas de ejecución en el lenguaje común (CLR) .NET" en la página 63
- "Autorizaciones y vinculación de rutinas que contienen SQL" en *Desarrollo de SQL y rutinas externas*

- "Gestión de bibliotecas y clases de rutinas externas" en la página 49
- "SQL en rutinas externas" en *Desarrollo de SQL y rutinas externas*

Tareas relacionadas:

- "Creación de rutinas CLR .NET desde ventanas de mandatos de DB2" en la página 77
- "Depuración de rutinas .NET CLR" en la página 88

Migración de rutinas .NET CLR a DB2 9.1

Migración de rutinas .NET CLR

Después de migrar una instancia de DB2 y bases de datos a DB2 Versión 9, deberá migrar las rutinas .NET CLR creadas con anterioridad a la Versión 9 para asegurarse de que siguen funcionando satisfactoriamente y se ejecutan tal y como se esperaba.

Requisitos previos:

- Revise las partes fundamentales de la migración para las rutinas para identificar los cambios clave que pudieran aplicarse a las rutinas .NET CLR.
- Asegúrese de que tiene acceso a un servidor de DB2 Versión 9, incluyendo instancias y bases de datos. Es posible que el servidor de DB2 sea parte del entorno de prueba.
- Asegúrese de que el sistema operativo está en un nivel de versión soportado por los productos de base de datos DB2.
- Asegúrese de que esté instalada una versión soportada del software de .NET Framework en el servidor de DB2.
- Realice las tareas previas a la migración para las rutinas.

Procedimiento:

Para migrar las rutinas .NET CLR a DB2 Versión 9:

1. Si ha identificado cambios en DB2 Versión 9 que afecten a sus rutinas, edite el código de rutina y modifique:
   - La sintaxis de la sentencias SQL
   - Las sentencias SQL que utilicen vistas de catálogos, rutinas y vistas administrativas de SQL
2. Conéctese a la base de datos DB2 Versión 9 en la que definió .NET CLR.
3. Vuelva a crear el código fuente de la rutina .NET CLR utilizando las opciones de compilar y enlazar especificadas en `bldrtn.bat`, el script de ejemplo de DB2 para crear rutinas .NET CLR.
4. Despliegue el conjunto para el servidor de DB2 en la misma ubicación que ha especificado la cláusula EXTERNAL en la definición de rutinas.
5. Pruebe las rutinas .NET CLR. Las rutinas deberían funcionar de modo satisfactorio, sin diferencias de comportamiento entre DB2 UDB Versión 8 y DB2 Versión 9.

Después de migrar las rutinas .NET CLR, realice las tareas posteriores a la migración para las rutinas recomendadas.

Conceptos relacionados:

- "Conceptos básicos de la migración para rutinas" en *Guía de migración*
- "Software de desarrollo .NET soportado" en la página 2
- "Tareas anteriores a la migración para aplicaciones y rutinas de bases de datos" en *Guía de migración*
- "Requisitos del sistema de bases de datos DB2 .NET Data Provider" en la página 7
- "Diseño de rutinas .NET CLR" en la página 65
- "ODBC .NET Data Provider" en la página 181
- "DB2 .NET Data Provider" en la página 7

Tareas relacionadas:

- "Creación de código de rutinas .NET CLR" en la página 79

Información relacionada:

- "Opciones de compilación y enlace para rutinas de CLR .NET" en la página 87

Ejemplos de rutinas .NET CLR

Ejemplos de rutinas .NET CLR

Cuando se desarrollan rutinas .NET CLR, resulta útil consultar ejemplos para ver el aspecto que puede tener la sentencia CREATE y el código de la rutina .NET CLR. Los temas siguientes contienen ejemplos de procedimientos y funciones .NET CLR (incluyen funciones tanto escalares como de tabla):

Procedimientos .NET CLR

- Ejemplos de procedimientos Visual Basic .NET CLR
- Ejemplos de procedimientos C# .NET CLR

Funciones .NET CLR

- Ejemplos de funciones Visual Basic .NET CLR
- Ejemplos de funciones C# .NET CLR

Conceptos relacionados:

- "Rutinas de ejecución en el lenguaje común (CLR) .NET" en la página 63

Tareas relacionadas:

- "Creación de código de rutinas .NET CLR" en la página 79
- "Creación de rutinas .NET CLR" en la página 75
- "Ejemplos de funciones CLR .NET en C#" en la página 126
- "Ejemplos de procedimientos CLR .NET en C#" en la página 103
- "Ejemplos de funciones CLR .NET en Visual Basic" en la página 119
- "Ejemplos de procedimientos CLR .NET en Visual Basic" en la página 94

Información relacionada:

- "Sentencias de SQL soportadas" en *Consulta de SQL, Volumen 2*

Ejemplos de procedimientos CLR .NET en Visual Basic

Una vez comprendidos los conceptos básicos de los procedimientos, también denominados procedimientos almacenados, y los fundamentos de las rutinas de ejecución en el lenguaje común .NET, puede empezar a utilizar procedimientos CLR en sus aplicaciones.

Este tema contiene ejemplos de procedimientos CLR implementados en `Visual Basic`; éstos ilustran los estilos de parámetros soportados, el pase de parámetros, incluida la estructura `dbinfo`, cómo devolver un conjunto de resultados y más información. Para obtener ejemplos de UDF CLR en `Visual Basic`:

- "Ejemplos de funciones CLR .NET en Visual Basic" en la página 119

Requisitos previos:

Antes de trabajar con los ejemplos de procedimientos CLR, puede ser conveniente que lea los temas sobre los conceptos siguientes:

- "Rutinas de ejecución en el lenguaje común (CLR) .NET" en la página 63
- "Creación de rutinas CLR .NET desde ventanas de mandatos de DB2" en la página 77
- "Beneficios del uso de rutinas" en la página 22
- Creación de rutinas .NET de ejecución en el lenguaje común (CLR)

Los ejemplos siguientes utilizan una tabla denominada `EMPLOYEE` que está incluida en la base de datos `SAMPLE`.

Procedimiento:

Utilice los ejemplos siguientes como referencias al crear sus propios procedimientos CLR en Visual Basic:

- "Archivo de código externo Visual Basic"
- "Ejemplo 1: Procedimiento en Visual Basic del estilo de parámetros GENERAL" en la página 95
- "Ejemplo 2: Procedimiento en Visual Basic del estilo de parámetros GENERAL WITH NULLS" en la página 96
- "Ejemplo 3: Procedimiento en Visual Basic del estilo de parámetros SQL" en la página 98
- "Ejemplo 4: Procedimiento en Visual Basic que devuelve un conjunto de resultados" en la página 99
- "Ejemplo 5: Procedimiento en Visual Basic que accede a la estructura dbinfo" en la página 100
- "Ejemplo 6: Procedimiento en Visual Basic del estilo PROGRAM TYPE MAIN" en la página 101

Archivo de código externo Visual Basic:

Los ejemplos muestran una variedad de implementaciones de procedimientos en Visual Basic. Cada ejemplo se compone de dos partes: la sentencia CREATE PROCEDURE y la implementación en código Visual Basic externo del procedimiento desde el cual se puede crear el conjunto asociado.

El archivo fuente en Visual Basic que contiene las implementaciones de procedimientos de los ejemplos siguientes se denomina `gwenVbProc.vb` y tiene el

formato siguiente:

Tabla 5. Formato del archivo de código externo Visual Basic

```
using System;
using System.IO;
using IBM.Data.DB2;

Namespace bizLogic

   Class empOps
              ...
     ' Procedimientos en Visual Basic
              ...
   End Class
End Namespace
```

Las inclusiones del archivo se indican al principio del mismo. La inclusión `IBM.Data.DB2` es necesaria si alguno de los procedimientos del archivo contiene SQL. Existe una declaración de espacio de nombres en este archivo y una clase `empOps` que contiene los procedimientos. El uso de espacios de nombres es opcional. Si se utiliza un espacio de nombres, éste debe aparecer en el nombre de vía de acceso de conjunto proporcionado en la cláusula EXTERNAL de la sentencia CREATE PROCEDURE.

Es importante tener en cuenta el nombre del archivo, el espacio de nombres y el nombre de la clase, que contiene una implementación de procedimiento determinada. Estos nombres son importantes, ya que la cláusula EXTERNAL de la sentencia CREATE PROCEDURE correspondiente a cada procedimiento debe especificar esta información a fin de que DB2 pueda localizar el conjunto y la clase del procedimiento CLR.

Ejemplo 1: Procedimiento en Visual Basic del estilo de parámetros GENERAL:

Este ejemplo muestra lo siguiente:

- Sentencia CREATE PROCEDURE para un procedimiento del estilo de parámetros GENERAL
- Código Visual Basic para un procedimiento del estilo de parámetros GENERAL

Este procedimiento toma un ID de empleado y una cantidad de bonificación actual como entrada. Recupera el nombre y el salario del empleado. Si la cantidad de bonificación actual es cero, se calcula una nueva bonificación basada en el salario del empleado y se devuelve junto con el nombre y apellidos del empleado. Si no se encuentra el empleado, se devuelve una serie vacía.

Tabla 6. Código para crear un procedimiento en Visual Basic del estilo de parámetros GENERAL

```
CREATE PROCEDURE SetEmpBonusGEN(IN empId CHAR(6),
                                INOUT bonus Decimal(9,2),
                                OUT empName VARCHAR(60))
SPECIFIC setEmpBonusGEN
LANGUAGE CLR
PARAMETER STYLE GENERAL
DYNAMIC RESULT SETS 0
FENCED
PROGRAM TYPE SUB
EXTERNAL NAME 'gwenVbProc.dll:bizLogic.empOps!SetEmpBonusGEN'
```

Tabla 6. Código para crear un procedimiento en Visual Basic del estilo de parámetros GENERAL (continuación)

```
Public Shared Sub SetEmpBonusGEN(ByVal empId As String, _
                                 ByRef bonus As Decimal, _
                                 ByRef empName As String)

   Dim salary As Decimal
   Dim myCommand As DB2Command
   Dim myReader As DB2DataReader

   salary = 0

   myCommand = DB2Context.GetCommand()
   myCommand.CommandText = _
             "SELECT FIRSTNME, MIDINIT, LASTNAME, SALARY " _
          + "FROM EMPLOYEE " _
          + "WHERE EMPNO = '" + empId + "'"
   myReader = myCommand.ExecuteReader()

   If myReader.Read()  ' Si se encuentra el registro de empleado
      ' Obtener nombre y apellidos y salario del empleado
      empName = myReader.GetString(0) + " "  _
              + myReader.GetString(1) + ". " _
              + myReader.GetString(2)

      salary = myReader.GetDecimal(3)

      If bonus = 0
         If salary > 75000
            bonus = salary * 0.025
         Else
            bonus = salary * 0.05
         End If
      End If
   Else  ' No se encuentra el empleado
      empName = ""  ' Establecer el parámetro de salida
   End If

   myReader.Close()

End Sub
```

Ejemplo 2: Procedimiento en Visual Basic del estilo de parámetros GENERAL WITH NULLS:
Este ejemplo muestra lo siguiente:

- Sentencia CREATE PROCEDURE para un procedimiento del estilo de parámetros GENERAL WITH NULLS
- Código Visual Basic para un procedimiento del estilo de parámetros GENERAL WITH NULLS

Este procedimiento toma un ID de empleado y una cantidad de bonificación actual como entrada. Si el parámetro de entrada no es nulo, recupera el nombre y salario del empleado. Si la cantidad de bonificación actual es cero, se calcula una nueva bonificación basada en el salario y se devuelve junto con el nombre y apellidos del empleado. Si no se encuentran los datos de empleado, se devuelven un entero y una serie NULL.

Tabla 7. Código para crear un procedimiento en Visual Basic del estilo de parámetros GENERAL WITH NULLS

```
CREATE PROCEDURE SetEmpBonusGENNULL(IN empId CHAR(6),
                                    INOUT bonus Decimal(9,2),
                                    OUT empName VARCHAR(60))
SPECIFIC SetEmpBonusGENNULL
LANGUAGE CLR
PARAMETER STYLE GENERAL WITH NULLS
DYNAMIC RESULT SETS 0
FENCED
PROGRAM TYPE SUB
EXTERNAL NAME 'gwenVbProc.dll:bizLogic.empOps!SetEmpBonusGENNULL'
```

```
  Public Shared Sub SetEmpBonusGENNULL(ByVal empId As String, _
                                       ByRef bonus As Decimal, _
                                       ByRef empName As String, _
                                       byVal nullInds As Int16())

     Dim salary As Decimal
     Dim myCommand As DB2Command
     Dim myReader As DB2DataReader

     salary = 0

     If nullInds(0) = -1   ' Comprobar si la entrada es nula
        nullInds(1) = -1   ' Devolver un valor de bonificación NULL
        empName = ""       ' Establecer el parámetro de salida
        nullInds(2) = -1   ' Devolver un valor empName NULL
        Return
     Else
        myCommand = DB2Context.GetCommand()
        myCommand.CommandText = _
                   "SELECT FIRSTNME, MIDINIT, LASTNAME, SALARY " _
                 + "FROM EMPLOYEE " _
                 + "WHERE EMPNO = '" + empId + "'"

        myReader = myCommand.ExecuteReader()

        If myReader.Read()  ' Si se encuentra el registro de empleado
           ' Obtener nombre y apellidos y salario del empleado
           empName = myReader.GetString(0) + " " _
                   + myReader.GetString(1) + ". " _
                   + myReader.GetString(2)

           salary = myReader.GetDecimal(3)

           If bonus = 0
              If salary > 75000
                 bonus = Salary * 0.025
                 nullInds(1) = 0 'Devolver un valor distinto de NULL
              Else
                 bonus = salary * 0.05
                 nullInds(1) = 0  ' Devolver un valor distinto de NULL
              End If
           Else  'No se encuentra el empleado
              empName = ""      ' Establecer el parámetro de salida
              nullInds(2) = -1  ' Devolver un valor NULL
           End If
        End If

        myReader.Close()

     End If

  End Sub
```

Ejemplo 3: Procedimiento en Visual Basic del estilo de parámetros SQL:
Este ejemplo muestra lo siguiente:

- Sentencia CREATE PROCEDURE para un procedimiento del estilo de parámetros SQL
- Código Visual Basic para un procedimiento del estilo de parámetros SQL

Este procedimiento toma un ID de empleado y una cantidad de bonificación actual como entrada. Recupera el nombre y el salario del empleado. Si la cantidad de bonificación actual es cero, se calcula una nueva bonificación basada en el salario y se devuelve junto con el nombre y apellidos del empleado. Si no se encuentra el empleado, se devuelve una serie vacía.

Tabla 8. Código para crear un procedimiento en Visual Basic del estilo de parámetros SQL con parámetros

```
CREATE PROCEDURE SetEmpBonusSQL(IN empId CHAR(6),
                                INOUT bonus Decimal(9,2),
                                OUT empName VARCHAR(60))
SPECIFIC SetEmpBonusSQL
LANGUAGE CLR
PARAMETER STYLE SQL
DYNAMIC RESULT SETS 0
FENCED
PROGRAM TYPE SUB
EXTERNAL NAME 'gwenVbProc.dll:bizLogic.empOps!SetEmpBonusSQL'
```

Tabla 8. Código para crear un procedimiento en Visual Basic del estilo de parámetros SQL con parámetros (continuación)

```
Public Shared Sub SetEmpBonusSQL(byVal empId As String, _
                                 byRef bonus As Decimal, _
                                 byRef empName As String, _
                                 byVal empIdNullInd As Int16, _
                                 byRef bonusNullInd As Int16, _
                                 byRef empNameNullInd As Int16, _
                                 byRef sqlState As String, _
                                 byVal funcName As String, _
                                 byVal specName As String, _
                                 byRef sqlMessageText As String)

   ' Declarar las variables locales del lenguaje principal
   Dim salary As Decimal
   Dim myCommand As DB2Command
   Dim myReader As DB2DataReader

   salary = 0

   If empIdNullInd = -1   ' Comprobar si la entrada es nula
      bonusNullInd = -1   ' Devolver un valor de bonificación NULL
      empName = ""
      empNameNullInd = -1 ' Devolver un valor empName NULL
   Else
     myCommand = DB2Context.GetCommand()
     myCommand.CommandText = _
                "SELECT FIRSTNME, MIDINIT, LASTNAME, SALARY " _
              + "FROM EMPLOYEE " _
              + " WHERE EMPNO = '" + empId + "'"

      myReader = myCommand.ExecuteReader()

      If myReader.Read()  ' Si se encuentra el registro de empleado
         ' Obtener nombre y apellidos y salario del empleado
         empName = myReader.GetString(0) + " "
                 + myReader.GetString(1) _
                 + ". " +  myReader.GetString(2)
         empNameNullInd = 0
         salary = myReader.GetDecimal(3)

         If bonus = 0
            If salary > 75000
               bonus = salary * 0.025
               bonusNullInd = 0  ' Devolver un valor distinto de NULL
            Else
               bonus = salary * 0.05
               bonusNullInd = 0  ' Devolver un valor distinto de NULL
            End If
         End If
      Else  ' No se encuentra el empleado
         empName = ""            ' Establecer el parámetro de salida
         empNameNullInd = -1     ' Devolver un valor NULL
      End If

      myReader.Close()
   End If

End Sub
```

Ejemplo 4: Procedimiento en Visual Basic del estilo de parámetros GENERAL que devuelve un conjunto de resultados:
Este ejemplo muestra lo siguiente:

- Sentencia CREATE PROCEDURE para un procedimiento en Visual Basic externo que devuelve un conjunto de resultados

- Código Visual Basic para un procedimiento del estilo de parámetros GENERAL que devuelve un conjunto de resultados

Este procedimiento acepta el nombre de una tabla como parámetro. Devuelve un conjunto de resultados que contiene todas las filas de la tabla especificada por el parámetro de entrada. Esto se realiza dejando abierto un `DB2DataReader` para un conjunto de resultados de consulta determinado cuando el procedimiento efectúa la devolución. Específicamente, si no se ejecuta `reader.Close()`, se devolverá el conjunto de resultados.

Tabla 9. Código para crear un procedimiento en Visual Basic que devuelve un conjunto de resultados

```
CREATE PROCEDURE ReturnResultSet(IN tableName VARCHAR(20))
SPECIFIC ReturnResultSet
DYNAMIC RESULT SETS 1
LANGUAGE CLR
PARAMETER STYLE GENERAL
FENCED
PROGRAM TYPE SUB
EXTERNAL NAME 'gwenVbProc.dll:bizLogic.empOps!ReturnResultSet'
```

```
Public Shared Sub ReturnResultSet(byVal tableName As String)

     Dim myCommand As DB2Command
     Dim myReader As DB2DataReader

     myCommand = DB2Context.GetCommand()

     ' Establecer la sentencia de SQL a ejecutar y ejecutarla.
     myCommand.CommandText = "SELECT * FROM " + tableName
     myReader = myCommand.ExecuteReader()

     ' El DB2DataReader contiene el resultado de la consulta.
     ' Este conjunto de resultados se puede devolver con el procedimiento
     ' simplemente NO cerrando el DB2DataReader.
     ' Específicamente, NO ejecutar reader.Close()

   End Sub
```

Ejemplo 5: Procedimiento en Visual Basic del estilo de parámetros SQL que accede a la estructura dbinfo:
Este ejemplo muestra lo siguiente:

- Sentencia CREATE PROCEDURE para un procedimiento que accede a la estructura `dbinfo`
- Código Visual Basic para un procedimiento del estilo de parámetros SQL que accede a la estructura `dbinfo`

Para acceder a la estructura `dbinfo`, se debe especificar la cláusula DBINFO en la sentencia CREATE PROCEDURE. No es necesario ningún parámetro para la estructura `dbinfo` en la sentencia CREATE PROCEDURE; no obstante, se debe crear un parámetro para la misma en el código externo de la rutina. Este procedimiento sólo devuelve el valor del nombre de base de datos actual desde el campo `dbname` de la estructura `dbinfo`.

Tabla 10. Código para crear un procedimiento en Visual Basic que accede a la estructura dbinfo

```
CREATE PROCEDURE ReturnDbName(OUT dbName VARCHAR(20))
SPECIFIC ReturnDbName
LANGUAGE CLR
PARAMETER STYLE SQL
DBINFO
FENCED
PROGRAM TYPE SUB
EXTERNAL NAME 'gwenVbProc.dll:bizLogic.empOps!ReturnDbName'
```

```
  Public Shared Sub ReturnDbName(byRef dbName As String, _
                                 byRef dbNameNullInd As Int16, _
                                 byRef sqlState As String, _
                                 byVal funcName As String, _
                                 byVal specName As String, _
                                 byRef sqlMessageText As String, _
                                 byVal dbinfo As sqludf_dbinfo)

    ' Recuperar el nombre de base de datos actual desde la
    ' estructura dbinfo y devolverlo.
    dbName = dbinfo.dbname
    dbNameNullInd = 0  ' Devolver un valor no nulo

    ' Si desea devolver un error definido por el usuario en la
    ' SQLCA puede especificar un SQLSTATE de 5 dígitos definido
    ' por el usuario y el texto de la serie de un mensaje de error.
    ' Por ejemplo:
    '
    ' sqlState = "ABCDE"
    ' msg_token = "A user-defined error has occured"
    '
    ' DB2 devolverá estos valores en la SQLCA. Aparecerá
    ' con el formato de un error sqlState normal de
    ' DB2.
  End Sub
```

Ejemplo 6: Procedimiento en Visual Basic con el estilo PROGRAM TYPE MAIN:
Este ejemplo muestra lo siguiente:

- Sentencia CREATE PROCEDURE para un procedimiento que utiliza un estilo de programa principal
- Código Visual Basic para el estilo de parámetros GENERAL WITH NULLS en la utilización de un estilo de programa MAIN

Para implementar una rutina en un estilo de programa principal, se debe especificar la cláusula PROGRAM TYPE en la sentencia CREATE PROCEDURE con el valor MAIN. Se especifican parámetros en la sentencia CREATE PROCEDURE; no obstante, en la implementación del código, los parámetros se pasan a la rutina en un parámetro entero `argc` y una matriz de parámetros `argv`.

Tabla 11. Código para crear un procedimiento en Visual Basic del estilo PROGRAM TYPE MAIN

```
CREATE PROCEDURE MainStyle(IN empId CHAR(6),
                           INOUT bonus Decimal(9,2),
                           OUT empName VARCHAR(60))
SPECIFIC mainStyle
DYNAMIC RESULT SETS 0
LANGUAGE CLR
PARAMETER STYLE GENERAL WITH NULLS
FENCED
PROGRAM TYPE MAIN
EXTERNAL NAME 'gwenVbProc.dll:bizLogic.empOps!Main'
```

Tabla 11. Código para crear un procedimiento en Visual Basic del estilo PROGRAM TYPE MAIN (continuación)

```
Public Shared Sub Main( byVal argc As Int32, _
                        byVal argv As Object())

   Dim myCommand As DB2Command
   Dim myReader As DB2DataReader
   Dim empId As String
   Dim bonus As Decimal
   Dim salary As Decimal
   Dim nullInds As Int16()

   empId = argv(0)  ' argv[0] (IN)    nullInd = argv[3]
   bonus = argv(1)  ' argv[1] (INOUT) nullInd = argv[4]
                    ' argv[2] (OUT)   nullInd = argv[5]
   salary = 0
   nullInds = argv(3)

   If nullInds(0) = -1     ' Comprobar si la entrada empId es nula
      nullInds(1) = -1     ' Devolver un valor de bonificación NULL
      argv(1) = ""         ' Establecer el parámetro de salida empName
      nullInds(2) = -1     ' Devolver un valor empName NULL
      Return
   Else
      ' Si el empleado existe y la bonificación actual es 0,
      ' calcular una nueva bonificación basándose en el salario
      ' del empleado. Devolver nombre y nueva bonificación de éste
      myCommand = DB2Context.GetCommand()
      myCommand.CommandText = _
                "SELECT FIRSTNME, MIDINIT, LASTNAME, SALARY " _
              + " FROM EMPLOYEE " _
              + " WHERE EMPNO = '" + empId + "'"

      myReader = myCommand.ExecuteReader()


      If myReader.Read()  ' Si se encuentra el registro de empleado
         ' Obtener nombre y apellidos y salario del empleado
         argv(2) = myReader.GetString(0) + " " _
                 + myReader.GetString(1) + ". " _
                 + myReader.GetString(2)
         nullInds(2) = 0
         salary = myReader.GetDecimal(3)

         If bonus = 0
            If salary > 75000
               argv(1) = salary * 0.025
               nullInds(1) = 0  ' Devolver un valor distinto de NULL
            Else
               argv(1) = Salary * 0.05
               nullInds(1) = 0  ' Devolver un valor distinto de NULL
            End If
         End If
      Else  ' No se encuentra el empleado
         argv(2) = ""      ' Establecer el parámetro de salida
         nullInds(2) = -1  ' Devolver un valor NULL
      End If

      myReader.Close()
   End If

End Sub
```

Conceptos relacionados:

- "Rutinas de ejecución en el lenguaje común (CLR) .NET" en la página 63

- "Beneficios del uso de rutinas" en la página 22

Tareas relacionadas:
- "Ejemplos de funciones CLR .NET en Visual Basic" en la página 119
- "Creación de rutinas CLR .NET desde ventanas de mandatos de DB2" en la página 77
- "Creación de rutinas CLR (Common Language Runtime) .NET" en la página 80
- "Ejemplos de procedimientos CLR .NET en C#" en la página 103
- "Ejemplos de funciones CLR .NET en C#" en la página 126

Ejemplos de procedimientos CLR .NET en C#

Una vez comprendidos los conceptos básicos de los procedimientos, también denominados procedimientos almacenados, y los fundamentos de las rutinas de ejecución en el lenguaje común .NET, puede empezar a utilizar procedimientos CLR en sus aplicaciones.

Este tema contiene ejemplos de procedimientos CLR implementados en C# que ilustran los estilos de parámetros soportados, el pase de parámetros, incluida la estructura `dbinfo`, cómo devolver un conjunto de resultados y más información. Para obtener ejemplos de UDF CLR en C#:
- "Ejemplos de funciones CLR .NET en C#" en la página 126

Requisitos previos:

Antes de trabajar con los ejemplos de procedimientos CLR, puede ser conveniente que lea los temas sobre los conceptos siguientes:
- "Rutinas de ejecución en el lenguaje común (CLR) .NET" en la página 63
- "Creación de rutinas CLR .NET desde ventanas de mandatos de DB2" en la página 77
- "Beneficios del uso de rutinas" en la página 22
- Creación de rutinas .NET de ejecución en el lenguaje común (CLR)

Los ejemplos siguientes utilizan una tabla denominada `EMPLOYEE` que está incluida en la base de datos `SAMPLE`.

Procedimiento:

Utilice los ejemplos siguientes como referencias al crear sus propios procedimientos CLR en C#:
- "Archivo de código externo C#" en la página 104
- "Ejemplo 1: Procedimiento en C# del estilo de parámetros GENERAL" en la página 104
- "Ejemplo 2: Procedimiento en C# del estilo de parámetros GENERAL WITH NULLS" en la página 105
- "Ejemplo 3: Procedimiento en C# del estilo de parámetros SQL" en la página 107
- "Ejemplo 4: Procedimiento en C# que devuelve un conjunto de resultados" en la página 110
- "Ejemplo 5: Procedimiento en C# que accede a la estructura dbinfo" en la página 110

• "Ejemplo 6: Procedimiento en C# del estilo PROGRAM TYPE MAIN" en la página 111

Archivo de código externo C#:

Los ejemplos muestran una variedad de implementaciones de procedimientos en C#. Cada ejemplo se compone de dos partes: la sentencia CREATE PROCEDURE y la implementación en código C# externo del procedimiento desde el cual se puede crear el conjunto asociado.

El archivo fuente en C# que contiene las implementaciones de procedimientos de los ejemplos siguientes se denomina `gwenProc.cs` y tiene el formato siguiente:

Tabla 12. Formato del archivo de código externo C#

```
using System;
using System.IO;
using IBM.Data.DB2;

namespace bizLogic
{
   class empOps
   {          ...
     // Procedimientos en C#
              ...
   }
}
```

Las inclusiones del archivo se indican al principio del mismo. La inclusión `IBM.Data.DB2` es necesaria si alguno de los procedimientos del archivo contiene SQL. Existe una declaración de espacio de nombres en este archivo y una clase `empOps` que contiene los procedimientos. El uso de espacios de nombres es opcional. Si se utiliza un espacio de nombres, éste debe aparecer en el nombre de vía de acceso de conjunto proporcionado en la cláusula EXTERNAL de la sentencia CREATE PROCEDURE.

Es importante tener en cuenta el nombre del archivo, el espacio de nombres y el nombre de la clase, que contiene una implementación de procedimiento determinada. Estos nombres son importantes, ya que la cláusula EXTERNAL de la sentencia CREATE PROCEDURE correspondiente a cada procedimiento debe especificar esta información a fin de que DB2 pueda localizar el conjunto y la clase del procedimiento CLR.

Ejemplo 1: Procedimiento en C# del estilo de parámetros GENERAL:

Este ejemplo muestra lo siguiente:

- Sentencia CREATE PROCEDURE para un procedimiento del estilo de parámetros GENERAL
- Código C# para un procedimiento del estilo de parámetros GENERAL

Este procedimiento toma un ID de empleado y una cantidad de bonificación actual como entrada. Recupera el nombre y el salario del empleado. Si la cantidad de bonificación actual es cero, se calcula una nueva bonificación basada en el salario del empleado y se devuelve junto con el nombre y apellidos del empleado. Si no se encuentra el empleado, se devuelve una serie vacía.

Tabla 13. Código para crear un procedimiento en C# del estilo de parámetros GENERAL

```
CREATE PROCEDURE setEmpBonusGEN(IN empID CHAR(6), INOUT bonus Decimal(9,2),
                                OUT empName VARCHAR(60))
SPECIFIC SetEmpBonusGEN
LANGUAGE CLR
PARAMETER STYLE GENERAL
MODIFIES SQL DATA
EXECUTION CONTROL SAFE
FENCED
THREADSAFE
DYNAMIC RESULT SETS 0
PROGRAM TYPE SUB
EXTERNAL NAME 'gwenProc.dll:bizLogic.empOps!SetEmpBonusGEN' ;
```

```
public static void SetEmpBonusGEN(    String empID,
                                   ref Decimal bonus,
                                   out String empName)
{
   // Declarar las variables locales
   Decimal salary = 0;

   DB2Command myCommand = DB2Context.GetCommand();
   myCommand.CommandText =
                    "SELECT FIRSTNME, MIDINIT, LASTNAME, SALARY "
                  + "FROM EMPLOYEE "
                  + "WHERE EMPNO = '" + empID + "';

   DB2DataReader reader = myCommand.ExecuteReader();

   if (reader.Read())  // Si se encuentra el registro de empleado
   {
      // Obtener nombre y apellidos y salario del empleado
      empName = reader.GetString(0) + " " +
                reader.GetString(1) + ". " +
                reader.GetString(2);

      salary = reader.GetDecimal(3);

      if (bonus == 0)
      {
         if (salary > 75000)
         {
            bonus = salary * (Decimal)0.025;
         }
         else
         {
            bonus = salary * (Decimal)0.05;
         }
      }
   }
   else  // No se encuentra el empleado
   {
      empName = "";  // Establecer el parámetro de salida
   }

   reader.Close();
}
```

Ejemplo 2: Procedimiento en C# del estilo de parámetros GENERAL WITH NULLS:

Este ejemplo muestra lo siguiente:

- Sentencia CREATE PROCEDURE para un procedimiento del estilo de parámetros GENERAL WITH NULLS

- Código C# para un procedimiento del estilo de parámetros GENERAL WITH NULLS

Este procedimiento toma un ID de empleado y una cantidad de bonificación actual como entrada. Si el parámetro de entrada no es nulo, recupera el nombre y salario del empleado. Si la cantidad de bonificación actual es cero, se calcula una nueva bonificación basada en el salario y se devuelve junto con el nombre y apellidos del empleado. Si no se encuentran los datos de empleado, se devuelven un entero y una serie NULL.

Tabla 14. Código para crear un procedimiento en C# del estilo de parámetros GENERAL WITH NULLS

```
CREATE PROCEDURE SetEmpbonusGENNULL(IN empID CHAR(6),
                                    INOUT bonus Decimal(9,2),
                                    OUT empName VARCHAR(60))
SPECIFIC SetEmpbonusGENNULL
LANGUAGE CLR
PARAMETER STYLE GENERAL WITH NULLS
DYNAMIC RESULT SETS 0
MODIFIES SQL DATA
EXECUTION CONTROL SAFE
FENCED
THREADSAFE
PROGRAM TYPE SUB
EXTERNAL NAME 'gwenProc.dll:bizLogic.empOps!SetEmpBonusGENNULL'
;
```

Tabla 14. Código para crear un procedimiento en C# del estilo de parámetros GENERAL WITH NULLS (continuación)

```
public static void SetEmpBonusGENNULL(    String empID,
                                       ref Decimal bonus,
                                       out String empName,
                                           Int16[] NullInds)
{
   Decimal salary = 0;
   if (NullInds[0] == -1) // Comprobar si la entrada es nula
   {
     NullInds[1] = -1;    // Devolver un valor de bonificación NULL
     empName = "";        // Establecer el valor de salida
     NullInds[2] = -1;    // Devolver un valor empName NULL
   }
   else
   {
      DB2Command myCommand = DB2Context.GetCommand();
     myCommand.CommandText =
                      "SELECT FIRSTNME, MIDINIT, LASTNAME, SALARY "
                    + "FROM EMPLOYEE "
                    + "WHERE EMPNO = '" + empID + "'";
      DB2DataReader reader = myCommand.ExecuteReader();

      if (reader.Read())  // Si se encuentra el registro de empleado
      {
         // Obtener nombre y apellidos y salario del empleado
         empName = reader.GetString(0) + " "
         +
                   reader.GetString(1) + ". " +
                   reader.GetString(2);
         salary = reader.GetDecimal(3);

         if (bonus == 0)
         {
           if (salary > 75000)
           {
              bonus = salary * (Decimal)0.025;
              NullInds[1] = 0; // Devolver un valor distinto de NULL
           }
           else
           {
              bonus = salary * (Decimal)0.05;
              NullInds[1] = 0; // Devolver un valor distinto de NULL
           }
         }
      }
      else  // No se encuentra el empleado
      {
         empName = "*sdq;;         // Establecer el parámetro de salida
         NullInds[2] = -1;     // Devolver un valor NULL
      }

      reader.Close();
   }
}
```

Ejemplo 3: Procedimiento en C# del estilo de parámetros SQL:
Este ejemplo muestra lo siguiente:

- Sentencia CREATE PROCEDURE para un procedimiento del estilo de parámetros SQL
- Código C# para un procedimiento del estilo de parámetros SQL

Este procedimiento toma un ID de empleado y una cantidad de bonificación actual como entrada. Recupera el nombre y el salario del empleado. Si la cantidad de bonificación actual es cero, se calcula una nueva bonificación basada en el salario y se devuelve junto con el nombre y apellidos del empleado. Si no se encuentra el empleado, se devuelve una serie vacía.

Tabla 15. Código para crear un procedimiento en C# del estilo de parámetros SQL con parámetros

```
CREATE PROCEDURE SetEmpbonusSQL(IN empID CHAR(6),
                                INOUT bonus Decimal(9,2),
                                OUT empName VARCHAR(60))
SPECIFIC SetEmpbonusSQL
LANGUAGE CLR
PARAMETER STYLE SQL
DYNAMIC RESULT SETS 0
MODIFIES SQL DATA
FENCED
THREADSAFE
EXECUTION CONTROL SAFE
PROGRAM TYPE SUB
EXTERNAL NAME 'gwenProc.dll:bizLogic.empOps!SetEmpBonusSQL' ;
```

Tabla 15. Código para crear un procedimiento en C# del estilo de parámetros SQL con parámetros (continuación)

```
public static void SetEmpBonusSQL(    String empID,
                                   ref Decimal bonus,
                                   out String empName,
                                       Int16  empIDNullInd,
                                   ref Int16  bonusNullInd,
                                   out Int16  empNameNullInd,
                                   ref string sqlStateate,
                                       string funcName,
                                       string specName,
                                   ref string sqlMessageText)
{
   // Declarar las variables locales del lenguaje principal
   Decimal salary eq; 0;

   if (empIDNullInd == -1) // Comprobar si la entrada es nula
   {
      bonusNullInd = -1;   // Devolver un valor de bonificación NULL
      empName = "";
      empNameNullInd = -1; // Devolver un valor empName NULL
   }
   else
      DB2Command myCommand = DB2Context.GetCommand();
      myCommand.CommandText =
                       "SELECT FIRSTNME, MIDINIT, LASTNAME, SALARY
                       "
                     + "FROM EMPLOYEE "
                     + "WHERE EMPNO = '" + empID + "'";

      DB2DataReader reader = myCommand.ExecuteReader();

      if (reader.Read())  // Si se encuentra el registro de empleado
      {
         // Obtener nombre y apellidos y salario del empleado
         empName = reader.GetString(0) + " "
         +
         reader.GetString(1) + ". " +
         reader.GetString(2);
         empNameNullInd = 0;
         salary = reader.GetDecimal(3);

         if (bonus == 0)
         {
            if (salary > 75000)
            {
               bonus = salary * (Decimal)0.025;
               bonusNullInd = 0;  // Devolver un valor distinto de NULL
            }
            else
            {
               bonus = salary * (Decimal)0.05;
               bonusNullInd = 0;  // Devolver un valor distinto de NULL
            }
         }
      }
      else  // No se encuentra el empleado
      }
         empName = "";         // Establecer el parámetro de salida
         empNameNullInd = -1;  // Devolver un valor NULL
      }

      reader.Close();
   }
}
```

Ejemplo 4: Procedimiento en C# del estilo de parámetros GENERAL que devuelve un conjunto de resultados:
Este ejemplo muestra lo siguiente:

- Sentencia CREATE PROCEDURE para un procedimiento en C# externo que devuelve un conjunto de resultados
- Código C# para un procedimiento del estilo de parámetros GENERAL que devuelve un conjunto de resultados

Este procedimiento acepta el nombre de una tabla como parámetro. Devuelve un conjunto de resultados que contiene todas las filas de la tabla especificada por el parámetro de entrada. Esto se realiza dejando abierto un `DB2DataReader` para un conjunto de resultados de consulta determinado cuando el procedimiento efectúa la devolución. Específicamente, si no se ejecuta `reader.Close()`, se devolverá el conjunto de resultados.

Tabla 16. Código para crear un procedimiento en C# que devuelve un conjunto de resultados

```
CREATE PROCEDURE ReturnResultSet(IN tableName
VARCHAR(20))
SPECIFIC ReturnResultSet
DYNAMIC RESULT SETS 1
LANGUAGE CLR
PARAMETER STYLE GENERAL
FENCED
PROGRAM TYPE SUB
EXTERNAL NAME
'gwenProc.dll:bizLogic.empOps!ReturnResultSet' ;
```

```
public static void ReturnResultSet(string tableName)
{
   DB2Command myCommand = DB2Context.GetCommand();

   // Establecer la sentencia de SQL a ejecutar y ejecutarla.
   myCommand.CommandText = "SELECT * FROM " + tableName;
   DB2DataReader reader = myCommand.ExecuteReader();

   // El DB2DataReader contiene el resultado de la consulta.
   // Este conjunto de resultados se puede devolver con el procedimiento
   // simplemente NO cerrando el DB2DataReader.
   // Específicamente, NO ejecutar reader.Close();
}
```

Ejemplo 5: Procedimiento en C# del estilo de parámetros SQL que accede a la estructura dbinfo:
Este ejemplo muestra lo siguiente:

- Sentencia CREATE PROCEDURE para un procedimiento que accede a la estructura `dbinfo`
- Código C# para un procedimiento del estilo de parámetros SQL que accede a la estructura `dbinfo`

Para acceder a la estructura `dbinfo`, se debe especificar la cláusula DBINFO en la sentencia CREATE PROCEDURE. No es necesario ningún parámetro para la estructura `dbinfo` en la sentencia CREATE PROCEDURE; no obstante, se debe crear un parámetro para la misma en el código externo de la rutina. Este procedimiento sólo devuelve el valor del nombre de base de datos actual desde el campo `dbname` de la estructura `dbinfo`.

Tabla 17. Código para crear un procedimiento en C# que accede a la estructura dbinfo

```
CREATE PROCEDURE ReturnDbName(OUT dbName VARCHAR(20))
SPECIFIC ReturnDbName
DYNAMIC RESULT SETS 0
LANGUAGE CLR
PARAMETER STYLE SQL
FENCED
THREADSAFE
EXECUTION CONTROL SAFE
DBINFO
PROGRAM TYPE SUB
EXTERNAL NAME 'gwenProc.dll:bizLogic.empOps!ReturnDbName'
;
```

```
public static void ReturnDbName(out string dbName,
                                 out Int16  dbNameNullInd,
                                 ref string sqlStateate,
                                     string funcName,
                                     string specName,
                                 ref string sqlMessageText,
                                     sqludf_dbinfo dbinfo)
{
   // Recuperar el nombre de base de datos actual desde la
   // estructura dbinfo y devolverlo.
   // ** Nota ** Los nombres del campo dbinfo son sensibles
   // a las mayúsculas y minúsculas
   dbName = dbinfo.dbname;
   dbNameNullInd = 0;  // Devolver un valor no nulo;

   // Si desea devolver un error definido por el usuario en la
   // SQLCA puede especificar un sqlStateate de 5 dígitos definido
   // por el usuario y el texto de la serie de un mensaje de error.
   // Por ejemplo:
   //
   //   sqlStateate = "ABCDE";
   //   sqlMessageText = "A user-defined error has occured"
   //
   //  DB2 devuelve los valores anteriores al cliente en la
   //  estructura SQLCA. Los valores se utilizan para generar
   //  un error sqlStateate estándar de DB2.
}
```

Ejemplo 6: Procedimiento en C# con el estilo PROGRAM TYPE MAIN:
Este ejemplo muestra lo siguiente:

- Sentencia CREATE PROCEDURE para un procedimiento que utiliza un estilo de programa principal
- Código C# para el estilo de parámetros GENERAL WITH NULLS en la utilización de un estilo de programa MAIN

Para implementar una rutina en un estilo de programa principal, se debe especificar la cláusula PROGRAM TYPE en la sentencia CREATE PROCEDURE con el valor MAIN. Se especifican parámetros en la sentencia CREATE PROCEDURE; no obstante, en la implementación del código, los parámetros se pasan a la rutina en un parámetro entero `argc` y una matriz de parámetros `argv`.

Tabla 18. Código para crear un procedimiento en C# del estilo PROGRAM TYPE MAIN

```
CREATE PROCEDURE MainStyle( IN empID CHAR(6),
                            INOUT bonus Decimal(9,2),
                            OUT empName VARCHAR(60))
SPECIFIC MainStyle
DYNAMIC RESULT SETS 0
LANGUAGE CLR
PARAMETER STYLE GENERAL WITH NULLS
MODIFIES SQL DATA
FENCED
THREADSAFE
EXECUTION CONTROL SAFE
PROGRAM TYPE MAIN
EXTERNAL NAME 'gwenProc.dll:bizLogic.empOps!main' ;
```

Tabla 18. Código para crear un procedimiento en C# del estilo PROGRAM TYPE MAIN (continuación)

```
public static void main(Int32 argc, Object[]
argv)
{
  String empID = (String)argv[0];  // argv[0] tiene nullInd:argv[3]
  Decimal bonus = (Decimal)argv[1]; // argv[1] tiene nullInd:argv[4]
                                    // argv[2] tiene nullInd:argv[5]
  Decimal salary = 0;     Int16[] NullInds =
  (Int16[])argv[3];

  if ((NullInds[0]) == (Int16)(-1)) // Comprobar si empID es nulo
  {
     NullInds[1] = (Int16)(-1);  // Devolver un valor de bonus NULL
     argv[1] = (String)"";       // Establecer el parámetro de salida empName
     NullInds[2] = (Int16)(-1);  // Devolver un valor de empName NULL
     Return;
  }
  else
     DB2Command myCommand = DB2Context.GetCommand();
     myCommand.CommandText =
                     "SELECT FIRSTNME, MIDINIT, LASTNAME, salary "
                   + "FROM EMPLOYEE "
                   + "WHERE EMPNO = '" + empID + "'";

     DB2DataReader reader = myCommand.ExecuteReader();

     if (reader.Read())  // Si se encuentra el registro de empleado
     {
       // Obtener nombre y apellidos y salario del empleado
       argv[2] = (String) (reader.GetString(0) + " " +
                           reader.GetString(1) + ".
                           " +
                           reader.GetString(2));
       NullInds[2] = (Int16)0;
       salary = reader.GetDecimal(3);

       if (bonus == 0)
       {
          if (salary > 75000)
          {
             argv[1] = (Decimal)(salary * (Decimal)0.025);
             NullInds[1] = (Int16)(0);  // Devolver un valor distinto de NULL
          }
          else
          {
             argv[1] = (Decimal)(salary * (Decimal)0.05);
             NullInds[1] = (Int16)(0);  // Devolver un valor distinto de NULL
          }
       }
     }
     else  // No se encuentra el empleado
     {
        argv[2] = (String)("");         // Establecer el parámetro de salida
        NullInds[2] = (Int16)(-1);      // Devolver un valor NULL
     }

     reader.Close();
   }
}
```

Conceptos relacionados:

- "Rutinas de ejecución en el lenguaje común (CLR) .NET" en la página 63
- "Beneficios del uso de rutinas" en la página 22

Tareas relacionadas:

- "Ejemplos de funciones CLR .NET en C#" en la página 126
- "Creación de rutinas CLR .NET desde ventanas de mandatos de DB2" en la página 77
- "Creación de rutinas CLR (Common Language Runtime) .NET" en la página 80
- "Ejemplos de rutinas .NET CLR" en la página 93

Ejemplos relacionados:

- "SpCreate.db2 -- Creates the external procedures implemented in spserver.cs"
- "SpServer.cs -- C# external code implementation of procedures created in spcat.db2"
- "SpCreate.db2 -- Creates the external procedures implemented in spserver.vb"
- "SpServer.vb -- VB.NET implementation of procedures created in SpCat.db2"

Ejemplo: Soporte de XML y XQuery en el procedimiento de C# .NET CLR

Una vez comprendidos los conceptos básicos de los procedimientos y los fundamentos de las rutinas de ejecución en el lenguaje común .NET, XQuery y XML puede empezar a utilizar procedimientos CLR con sus características XML.

El ejemplo que hay a continuación muestra un procedimiento de C# .NET CLR con parámetros del tipo XML así como el modo de actualizar y consultar datos XML.

Requisitos previos:

Antes de trabajar con el ejemplo de procedimientos CLR, puede ser conveniente que lea los temas sobre los conceptos siguientes:

- Rutinas de ejecución en el lenguaje común (CLR)
- Creación de rutinas CLR
- Beneficios del uso de rutinas
- Creación de rutinas .NET de ejecución en el lenguaje común (CLR)

Los ejemplos que hay a continuación utilizan una tabla denominada `xmltable` que se define del siguiente modo:

```
CREATE TABLE xmltable
(
   num INTEGER,
   xdata XML
)

INSERT INTO xmltable VALUES
   (1, XMLPARSE(DOCUMENT '<doc>
                              <type>car</type>
                              <make>Pontiac</make>
                              <model>Sunfire</model>
                              </doc>' PRESERVE WHITESPACE)),
   (2, XMLPARSE(DOCUMENT '<doc>
                              <type>car</type>
                              <make>Mazda</make>
                              <model>Miata</model>
                              </doc>' PRESERVE WHITESPACE)),
   (3, XMLPARSE(DOCUMENT '<doc>
                              <type>person</type>
                              <name>Mary</name>
                              <town>Vancouver</town>
```

```
                            <street>Waterside</street>
                            </doc>' PRESERVE WHITESPACE)),
(4, XMLPARSE(DOCUMENT '<doc>
                            <type>person</type>
                            <name>Mark</name>
                            <town>Edmonton</town>
                            <street>Oak</street>
                            </doc>' PRESERVE WHITESPACE)),
(5, XMLPARSE(DOCUMENT '<doc>
                             <type>animal</type>
                            <name>dog</name>
                            </doc>' PRESERVE WHITESPACE)),
(6, NULL),
(7, XMLPARSE(DOCUMENT '<doc>
                            <type>car</type>
                            <make>Ford</make>
                            <model>Taurus</model>
                            </doc>' PRESERVE WHITESPACE)),
(8, XMLPARSE(DOCUMENT '<doc>
                            <type>person</type>
                            <name>Kim</name>
                            <town>Toronto</town>
                            <street>Elm</street>
                            </doc>' PRESERVE WHITESPACE)),
(9, XMLPARSE(DOCUMENT '<doc>
                            <type>person</type>
                            <name>Bob</name>
                            <town>Toronto</town>
                            <street>Oak</street>
                            </doc>' PRESERVE WHITESPACE)),
(10, XMLPARSE(DOCUMENT '<doc>
                             <type>animal</type>
                             <name>bird</name>
                             </doc>' PRESERVE WHITESPACE))@
```

Procedimiento:

Utilice los ejemplos siguientes como referencias al crear sus propios procedimientos CLR en C#:

- "El archivo de código externo C#"
- "Ejemplo 1: Procedimiento en C# del estilo de parámetros GENERAL con características XML" en la página 116

Archivo de código externo C#:

El ejemplo se compone de dos partes: la sentencia CREATE PROCEDURE y la implementación en código C# externo del procedimiento desde el cual se puede crear el conjunto asociado.

El archivo fuente en C# que contiene las implementaciones de procedimientos de los ejemplos siguientes se denomina `gwenProc.cs` y tiene el formato siguiente:

Tabla 19. Formato del archivo de código externo C#

```
using System;
using System.IO;
using System.Data;
using IBM.Data.DB2;
using IBM.Data.DB2Types;

namespace bizLogic
{
   class empOps
   {          ...
     // Procedimientos en C#
              ...
   }
}
```

Las inclusiones del archivo se indican al principio del mismo. La inclusión `IBM.Data.DB2` es necesaria si alguno de los procedimientos del archivo contiene SQL. La inclusión de `IBM.Data.DB2Types` es necesaria si se va a utilizar alguno de los procedimientos del archivo que contenga parámetros o variables del tipo XML. Existe una declaración de espacio de nombres en este archivo y una clase `empOps` que contiene los procedimientos. El uso de espacios de nombres es opcional. Si se utiliza un espacio de nombres, éste debe aparecer en el nombre de vía de acceso de ensamblaje proporcionado en la cláusula EXTERNAL de la sentencia CREATE PROCEDURE.

Es importante tener en cuenta el nombre del archivo, el espacio de nombres y el nombre de la clase, que contiene una implementación de procedimiento determinada. Estos nombres son importantes, ya que la cláusula EXTERNAL de la sentencia CREATE PROCEDURE correspondiente a cada procedimiento debe especificar esta información a fin de que DB2 pueda localizar el conjunto y la clase del procedimiento CLR.

Ejemplo 1: Procedimiento en C# de estilo de parámetros GENERAL con características XML:

Este ejemplo muestra lo siguiente:

- Sentencia CREATE PROCEDURE para un procedimiento del estilo de parámetros GENERAL
- Código C# para un procedimiento del estilo de parámetros GENERAL con parámetros XML

Este procedimiento toma dos parámetros, un entero `inNum` y `inXML`. Estos valores se insertan en la tabla `xmltable`. A continuación, se recupera un valor XML utilizando XQuery. Otro valor se recupera utilizando SQL. Los valores XML recuperados se asignan a dos parámetros de salida, `outXML1` y `outXML2`. No se devuelve ningún conjunto de resultados.

Tabla 20. Código para crear un procedimiento en C# del estilo de parámetros GENERAL

```
  CREATE PROCEDURE  xmlProc1   ( IN inNUM  INTEGER,
                               IN inXML  XML as CLOB (1K),
                               OUT inXML  XML as CLOB (1K),
                               OUT inXML  XML as CLOB (1K)
                             )
  LANGUAGE CLR
PARAMETER STYLE GENERAL
DYNAMIC RESULT SETS 0
       FENCED
  THREADSAFE
  DETERMINISTIC
NO DBINFO
    MODIFIES SQL DATA
  PROGRAM TYPE SUB
  EXTERNAL NAME 'gwenProc.dll:bizLogic.empOps!xmlProc1' ;
```

```
  //*************************************************************************
  //  Stored Procedure: xmlProc1
  //
  //  Purpose:   insert XML data into XML column
  //
  //  Parameters:
  //
  //   IN:    inNum -- the sequence of XML data to be insert in xmldata table
  //          inXML -- XML data to be inserted
  //   OUT:   outXML1 -- XML data returned - value retrieved using XQuery
  //          outXML2 -- XML data returned - value retrieved using SQL
  //*************************************************************************
```

Tabla 20. Código para crear un procedimiento en C# del estilo de parámetros GENERAL (continuación)

```
public static void xmlProc1 (     int      inNum, DB2Xml   inXML,
                                out DB2Xml   outXML1, out DB2Xml   outXML2 )
{
   // Create new command object from connection context
   DB2Parameter parm;
   DB2Command cmd;
   DB2DataReader reader = null;
   outXML1 = DB2Xml.Null;
   outXML2 = DB2Xml.Null;

   // Insert input XML parameter value into a table
   cmd = DB2Context.GetCommand();
   cmd.CommandText = "INSERT INTO "
                   + "xmltable( num , xdata ) "
                   + "VALUES( ?, ? )";

   parm = cmd.Parameters.Add("@num", DB2Type.Integer );
   parm.Direction = ParameterDirection.Input;
   cmd.Parameters["@num"].Value = inNum;
   parm = cmd.Parameters.Add("@data", DB2Type.Xml);
   parm.Direction = ParameterDirection.Input;
   cmd.Parameters["@data"].Value = inXML ;
   cmd.ExecuteNonQuery();
   cmd.Close();

   // Retrieve XML value using XQuery
                 and assign value to an XML output parameter
   cmd = DB2Context.GetCommand();
   cmd.CommandText = "XQUERY for $x " +
                     "in db2-fn:xmlcolumn(\"XMLTABLE.XDATA\")/doc "+
                     "where $x/make = \'Mazda\' " +
                     "return <carInfo>{$x/make}{$x/model}</carInfo>";
   reader = cmd.ExecuteReader();
   reader.CacheData= true;

   if (reader.Read())
   {  outXML1 = reader.GetDB2Xml(0);  }
   else
   {  outXML1 = DB2Xml.Null;  }

   reader.Close();
   cmd.Close();

   // Retrieve XML value using SQL
                 and assign value to an XML output parameter value
   cmd = DB2Context.GetCommand();
   cmd.CommandText = "SELECT xdata "
                    + "FROM xmltable "
                    + "WHERE num = ?";

   parm = cmd.Parameters.Add("@num", DB2Type.Integer );
   parm.Direction = ParameterDirection.Input;
   cmd.Parameters["@num"].Value = inNum;
   reader = cmd.ExecuteReader();
   reader.CacheData= true;

   if (reader.Read())
   {  outXML2 = reader.GetDB2Xml(0);  }
   else
   {  outXML = DB2Xml.Null;  }

   reader.Close() ;
   cmd.Close();

   return;
}
```

Conceptos relacionados:

- "Rutinas de ejecución en el lenguaje común (CLR) .NET" en la página 63
- "Beneficios del uso de rutinas" en la página 22

Tareas relacionadas:

- "Creación de rutinas CLR (Common Language Runtime) .NET" en la página 80
- "Creación de rutinas CLR .NET desde ventanas de mandatos de DB2" en la página 77

Ejemplos de funciones CLR .NET en Visual Basic

Una vez comprendidos los conceptos básicos de las funciones definidas por el usuario (UDF) y los fundamentos de las rutinas CLR, puede empezar a explotar las UDF CLR en sus aplicaciones y en el entorno de bases de datos. Este tema contiene algunos ejemplos de UDF CLR para poder empezar. Si desea obtener ejemplos de procedimientos CLR en Visual Basic:

- "Ejemplos de procedimientos CLR .NET en Visual Basic" en la página 94

Requisitos previos:

Antes de trabajar con los ejemplos de UDF CLR, puede ser conveniente que lea los temas sobre los conceptos siguientes:

- "Rutinas de ejecución en el lenguaje común (CLR) .NET" en la página 63
- "Creación de rutinas CLR .NET desde ventanas de mandatos de DB2" en la página 77
- "Funciones escalares externas" en la página 26
- Rutinas: Funciones de tabla definidas por el usuario
- Creación de rutinas .NET de ejecución en el lenguaje común (CLR)

Los ejemplos siguientes utilizan una tabla denominada `EMPLOYEE` que está incluida en la base de datos `SAMPLE`.

Procedimiento:

Utilice los ejemplos siguientes como referencias al crear sus propias UDF CLR en Visual Basic:

- "Archivo de código externo Visual Basic"
- "Ejemplo 1: Función de tabla en Visual Basic del estilo de parámetros SQL" en la página 120
- "Ejemplo 2: Función escalar en Visual Basic del estilo de parámetros SQL" en la página 123

Archivo de código externo Visual Basic:

Los ejemplos siguientes muestran una variedad de implementaciones de UDF en Visual Basic. La sentencia CREATE FUNCTION se proporciona para cada UDF con el código fuente Visual Basic correspondiente desde el cual se puede crear el conjunto asociado. El archivo fuente en Visual Basic que contiene las declaraciones de funciones utilizadas en los ejemplos siguientes se denomina `gwenVbUDF.cs` y tiene el formato siguiente:

Tabla 21. Formato del archivo de código externo Visual Basic

```
  using System;
  using System.IO;
  using IBM.Data.DB2;

  Namespace bizLogic

     ...
     ' Definiciones de clases que contienen declaraciones de UDF
     ' y cualquier definición de clase de soporte
     ...

End Namespace
```

Las declaraciones de funciones deben estar incluidas en una clase dentro de un archivo de Visual Basic. El uso de espacios de nombres es opcional. Si se utiliza un espacio de nombres, éste debe aparecer en el nombre de vía de acceso de conjunto proporcionado en la cláusula EXTERNAL de la sentencia CREATE PROCEDURE. La inclusión de `IBM.Data.DB2.` es necesaria si la función contiene SQL.

Ejemplo 1: Función de tabla en Visual Basic del estilo de parámetros SQL:
Este ejemplo muestra lo siguiente:

- Sentencia CREATE FUNCTION para una función de tabla del estilo de parámetros SQL
- Código Visual Basic para una función de tabla del estilo de parámetros SQL

Esta función de tabla devuelve una tabla que contiene filas de los datos de empleado que se han creado a partir de una matriz de datos. Existen dos clases asociadas con este ejemplo. La clase `person` representa los empleados y la clase `empOps` contiene la UDF de tabla de rutina que utiliza la clase `person`. La información sobre el salario de los empleados se actualiza basándose en el valor de un parámetro de entrada. La matriz de datos de este ejemplo se crea dentro de la propia función de tabla en la primera llamada de la función de tabla. Dicha matriz también se podría haber creado leyendo datos de un archivo de texto del sistema de archivos. Los valores de datos de la matriz se graban en un área reutilizable para que sea posible acceder a los datos en llamadas subsiguientes de la función de tabla.

En cada llamada de la función de tabla, se lee un registro de la matriz y se genera una fila en la tabla devuelta por la función. La fila se genera en la tabla estableciendo los parámetros de salida de la función de tabla en los valores de fila deseados. Después de que se produzca la llamada final de la función de tabla, se devuelve la tabla de las filas generadas.

Tabla 22. Código para crear una función de tabla en Visual Basic del estilo de parámetros SQL

```
CREATE FUNCTION TableUDF(double)
RETURNS TABLE (name varchar(20),
               job varchar(20),
               salary double)
EXTERNAL NAME 'gwenVbUDF.dll:bizLogic.empOps!TableUDF'
LANGUAGE CLR
PARAMETER STYLE SQL
NOT DETERMINISTIC
FENCED
SCRATCHPAD 10
FINAL CALL
DISALLOW PARALLEL
NO DBINFO
EXECUTION CONTROL SAFE
```

Tabla 22. Código para crear una función de tabla en Visual Basic del estilo de parámetros SQL (continuación)

```
Class Person
' La clase Person es una clase de soporte para
' la función de tabla UDF, tableUDF, siguiente.

  Private name As String
  Private position As String
  Private salary As Int32

  Public Sub New(ByVal newName As String, _
                 ByVal newPosition As String, _
                 ByVal newSalary As Int32)

    name = newName
    position = newPosition
    salary = newSalary
  End Sub

  Public Property GetName() As String
    Get
      Return name
    End Get

    Set (ByVal value As String)
      name = value
    End Set
  End Property

  Public Property GetPosition() As String
    Get
      Return position
    End Get

    Set (ByVal value As String)
      position = value
    End Set
  End Property

  Public Property GetSalary() As Int32
    Get
      Return salary
    End Get

    Set (ByVal value As Int32)
      salary = value
    End Set
  End Property

End Class
```

Tabla 22. Código para crear una función de tabla en Visual Basic del estilo de parámetros SQL (continuación)

```
Class empOps

  Public Shared Sub TableUDF(byVal factor as Double, _
                             byRef name As String, _
                             byRef position As String, _
                             byRef salary As Double, _
                             byVal factorNullInd As Int16, _
                             byRef nameNullInd As Int16, _
                             byRef positionNullInd As Int16, _
                             byRef salaryNullInd As Int16, _
                             byRef sqlState As String, _
                             byVal funcName As String, _
                             byVal specName As String, _
                             byRef sqlMessageText As String, _
                             byVal scratchPad As Byte(), _
                             byVal callType As Int32)

    Dim intRow As Int16

    intRow = 0

    ' Crear una matriz de información del tipo Person
    Dim staff(2) As Person
    staff(0) = New Person("Gwen", "Developer", 10000)
    staff(1) = New Person("Andrew", "Developer", 20000)
    staff(2) = New Person("Liu", "Team Leader", 30000)

    ' Inicializar valores de parámetro de salida e indicadores NULL
    salary = 0
    name = position = ""
    nameNullInd = positionNullInd = salaryNullInd = -1

    Select callType
       Case -2   ' Case SQLUDF_TF_FIRST:
       Case -1   ' Case SQLUDF_TF_OPEN:
         intRow = 1
         scratchPad(0) = intRow  ' Grabar en área reutilizable
       Case 0    ' Case SQLUDF_TF_FETCH:
         intRow = scratchPad(0)
         If intRow > staff.Length
            sqlState = "02000"  ' Devolver un error SQLSTATE
         Else
            ' Generar una fila en la tabla de salida
            ' basada en los datos de la matriz staff.
            name = staff(intRow).GetName()
            position = staff(intRow).GetPosition()
            salary = (staff(intRow).GetSalary()) * factor
            nameNullInd = 0
            positionNullInd = 0
            salaryNullInd = 0
         End If
         intRow = intRow + 1
         scratchPad(0) = intRow  ' Grabar área reutilizable

       Case 1    ' Case SQLUDF_TF_CLOSE:

       Case 2    ' Case SQLUDF_TF_FINAL:
    End Select

  End Sub

End Class
```

Ejemplo 2: Función escalar en Visual Basic del estilo de parámetros SQL:

Este ejemplo muestra lo siguiente:

- Sentencia CREATE FUNCTION para una función escalar del estilo de parámetros SQL
- Código Visual Basic para una función escalar del estilo de parámetros SQL

Esta función escalar devuelve un solo valor de cuenta para cada valor de entrada sobre el que actúa. Para un valor de entrada situado en la nª posición del conjunto de valores de entrada, el valor escalar de salida es el valor n. En cada llamada de la función escalar, donde una llamada está asociada con cada fila o valor del conjunto de filas o valores de entrada, la cuenta aumenta en uno y se devuelve el valor actual de la cuenta. Luego, la cuenta se guarda en el almacenamiento intermedio de memoria del área reutilizable para mantener el valor de cuenta entre cada llamada de la función escalar.

Esta función escalar se puede invocar fácilmente si, por ejemplo, se dispone de una tabla definida del modo siguiente:

```
CREATE TABLE T (i1 INTEGER);
INSERT INTO T VALUES 12, 45, 16, 99;
```

Se puede utilizar una consulta simple como la siguiente para invocar la función escalar:

```
SELECT my_count(i1) as count, i1 FROM T;
```

La salida de una consulta como la indicada sería:

```
COUNT           I1
-----------     ----------
1               12
2               45
3               16
4               99
```

Esta UDF escalar es bastante simple. En lugar de devolver sólo la cuenta de las filas, puede utilizar una función escalar que formatee los datos de una columna existente. Por ejemplo, puede añadir una serie a cada valor de una columna de direcciones, puede crear una serie compleja a partir de una cadena de series de entrada o puede efectuar un cálculo matemático complejo con un conjunto de datos donde deberá almacenar un resultado intermedio.

Tabla 23. Código para crear una función escalar en Visual Basic del estilo de parámetros SQL

```
CREATE FUNCTION mycount(INTEGER)
RETURNS INTEGER
LANGUAGE CLR
PARAMETER STYLE SQL
NO SQL
SCRATCHPAD 10
FINAL CALL
FENCED
EXECUTION CONTROL SAFE
NOT DETERMINISTIC
EXTERNAL NAME 'gwenUDF.dll:bizLogic.empOps!CountUp';
```

Tabla 23. Código para crear una función escalar en Visual Basic del estilo de parámetros SQL (continuación)

```
Class empOps
  Public Shared Sub CountUp(byVal input As Int32, _
                            byRef outCounter As Int32, _
                            byVal nullIndInput As Int16, _
                            byRef nullIndOutCounter As Int16, _
                            byRef sqlState As String, _
                            byVal qualName As String, _
                            byVal specName As String, _
                            byRef sqlMessageText As String, _
                            byVal scratchPad As Byte(), _
                            byVal callType As Int32)

     Dim counter As Int32
     counter = 1

     Select callType
        case -1           ' case SQLUDF_TF_OPEN_CALL
           scratchPad(0) = counter
           outCounter = counter
           nullIndOutCounter = 0
        case 0              'case SQLUDF_TF_FETCH_CALL:
           counter = scratchPad(0)
           counter = counter + 1
           outCounter = counter
           nullIndOutCounter = 0
           scratchPad(0) = counter
        case 1             'case SQLUDF_CLOSE_CALL:
           counter = scratchPad(0)
           outCounter = counter
           nullIndOutCounter = 0
        case Else          ' Nunca se debe entrar aquí
           ' Estos casos no se producirán por las razones siguientes:
           ' Case -2  (SQLUDF_TF_FIRST)     ->No hay FINAL CALL en sent. CREATE
           ' Case 2   (SQLUDF_TF_FINAL)     ->No hay FINAL CALL en sent. CREATE
           ' Case 255 (SQLUDF_TF_FINAL_CRA) ->No se utiliza SQL en la función
           '
           ' * Nota*
           ' ---------
           ' Else es necesario para que durante la compilación
           ' se establezca siempre el parámetro de salida outCounter *
           outCounter = 0
           nullIndOutCounter = -1
     End Select
  End Sub

End Class
```

Conceptos relacionados:

- "Rutinas de ejecución en el lenguaje común (CLR) .NET" en la página 63
- "Funciones escalares externas" en la página 26
- "Funciones de tabla definidas por el usuario" en *Desarrollo de SQL y rutinas externas*

Tareas relacionadas:

- "Ejemplos de procedimientos CLR .NET en Visual Basic" en la página 94
- "Creación de rutinas CLR .NET desde ventanas de mandatos de DB2" en la página 77
- "Creación de rutinas CLR (Common Language Runtime) .NET" en la página 80
- "Ejemplos de rutinas .NET CLR" en la página 93

- "Ejemplos de funciones CLR .NET en C#" en la página 126
- "Ejemplos de procedimientos CLR .NET en C#" en la página 103

Ejemplos de funciones CLR .NET en C#

Una vez comprendidos los conceptos básicos de las funciones definidas por el usuario (UDF) y los fundamentos de las rutinas CLR, puede empezar a explotar las UDF CLR en sus aplicaciones y en el entorno de bases de datos. Este tema contiene algunos ejemplos de UDF CLR para poder empezar. Si desea obtener ejemplos de procedimientos CLR en C#:

- "Ejemplos de procedimientos CLR .NET en C#" en la página 103

Requisitos previos:

Antes de trabajar con los ejemplos de UDF CLR, puede ser conveniente que lea los temas sobre los conceptos siguientes:

- "Rutinas de ejecución en el lenguaje común (CLR) .NET" en la página 63
- "Creación de rutinas CLR .NET desde ventanas de mandatos de DB2" en la página 77
- "Funciones escalares externas" en la página 26
- Rutinas: Funciones de tabla definidas por el usuario
- Creación de rutinas .NET de ejecución en el lenguaje común (CLR)

Los ejemplos siguientes utilizan una tabla denominada `EMPLOYEE` que está incluida en la base de datos `SAMPLE`.

Procedimiento:

Utilice los ejemplos siguientes como referencias al crear sus propias UDF CLR en C#:

- "Archivo de código externo C#"
- "Ejemplo 1: Función de tabla en C# del estilo de parámetros SQL" en la página 127
- "Ejemplo 2: Función escalar en C# del estilo de parámetros SQL" en la página 129

Archivo de código externo C#:

Los ejemplos siguientes muestran una variedad de implementaciones de UDF en C#. La sentencia CREATE FUNCTION se proporciona para cada UDF con el código fuente C# correspondiente desde el cual se puede crear el conjunto asociado. El archivo fuente en C# que contiene las declaraciones de funciones utilizadas en los ejemplos siguientes se denomina `gwenUDF.cs` y tiene el formato siguiente:

Tabla 24. Formato del archivo de código externo C#

```
using System;
using System.IO;
using IBM.Data.DB2;

namespace bizLogic
{
   ...
   // Definiciones de clases que contienen declaraciones de UDF
   // y cualquier definición de clase de soporte
   ...
}
```

Las declaraciones de funciones deben estar incluidas en una clase dentro de un archivo de C#. El uso de espacios de nombres es opcional. Si se utiliza un espacio de nombres, éste debe aparecer en el nombre de vía de acceso de conjunto proporcionado en la cláusula EXTERNAL de la sentencia CREATE PROCEDURE. La inclusión de `IBM.Data.DB2.` es necesaria si la función contiene SQL.

Ejemplo 1: Función de tabla en C# del estilo de parámetros SQL:
Este ejemplo muestra lo siguiente:

- Sentencia CREATE FUNCTION para una función de tabla del estilo de parámetros SQL
- Código C# para una función de tabla del estilo de parámetros SQL

Esta función de tabla devuelve una tabla que contiene filas de los datos de empleado que se han creado a partir de una matriz de datos. Existen dos clases asociadas con este ejemplo. La clase `person` representa los empleados y la clase `empOps` contiene la UDF de tabla de rutina que utiliza la clase `person`. La información sobre el salario de los empleados se actualiza basándose en el valor de un parámetro de entrada. La matriz de datos de este ejemplo se crea dentro de la propia función de tabla en la primera llamada de la función de tabla. Dicha matriz también se podría haber creado leyendo datos de un archivo de texto del sistema de archivos. Los valores de datos de la matriz se graban en un área reutilizable para que sea posible acceder a los datos en llamadas subsiguientes de la función de tabla.

En cada llamada de la función de tabla, se lee un registro de la matriz y se genera una fila en la tabla devuelta por la función. La fila se genera en la tabla estableciendo los parámetros de salida de la función de tabla en los valores de fila deseados. Después de que se produzca la llamada final de la función de tabla, se devuelve la tabla de las filas generadas.

Tabla 25. Código para crear una función de tabla en C# del estilo de parámetros SQL

```
CREATE FUNCTION tableUDF(double)
RETURNS TABLE (name varchar(20),
               job varchar(20),
               salary double)
EXTERNAL NAME 'gwenUDF.dll:bizLogic.empOps!tableUDF'
LANGUAGE CLR
PARAMETER STYLE SQL
NOT DETERMINISTIC
FENCED
THREADSAFE
SCRATCHPAD 10
FINAL CALL
EXECUTION CONTROL SAFE
DISALLOW PARALLEL
NO DBINFO
```

Tabla 25. Código para crear una función de tabla en C# del estilo de parámetros SQL (continuación)

```
// La clase Person es una clase de soporte para
// la función de tabla UDF, tableUDF, siguiente.
class Person
{
     private String name;
     private String position;
     private Int32 salary;

     public Person(String newName, String newPosition, Int32
     newSalary)
     {
        this.name = newName;
        this.position = newPosition;
        this.salary = newSalary;
     }

     public String getName()
     {
        return this.name;
     }

     public String getPosition()
     {
        return this.position;
     }

     public Int32 getSalary()
     {
        return this.salary;
     }
 }
```

Tabla 25. Código para crear una función de tabla en C# del estilo de parámetros SQL (continuación)

```
class empOps
{
  {
   public static void TableUDF( Double factor, out String name,
                     out String position, out Double salary,
                     Int16 factorNullInd, out Int16 nameNullInd,
                     out Int16 positionNullInd, out Int16 salaryNullInd,
                     ref String sqlState, String funcName,
                     String specName, ref String sqlMessageText,
                     Byte[] scratchPad, Int32 callType)
   {

      Int16 intRow = 0;

      // Crear una matriz de información del tipo Person
      Person[] Staff = new
      Person[3];
      Staff[0] = new Person("Gwen", "Developer", 10000);
      Staff[1] = new Person("Andrew", "Developer", 20000);
      Staff[2] = new Person("Liu", "Team Leader", 30000);

      salary = 0;
      name = position = "";
      nameNullInd = positionNullInd = salaryNullInd = -1;

      switch(callType)
      {
         case (-2):  // Case SQLUDF_TF_FIRST:
           break;

         case (-1):  // Case SQLUDF_TF_OPEN:
           intRow = 1;
           scratchPad[0] = (Byte)intRow;  // Grabar en área reutilizable
           break;
         case (0):   // Case SQLUDF_TF_FETCH:
           intRow = (Int16)scratchPad[0];
           if (intRow > Staff.Length)
           {
              sqlState = "02000";  // Devolver un error SQLSTATE
           }
           else
           {
              // Generar una fila en la tabla de salida
              // basada en los datos de la matriz Staff.
              name =
              Staff[intRow-1].getName();
              position = Staff[intRow-1].getPosition();
              salary = (Staff[intRow-1].getSalary[]] * factor;
              nameNullInd = 0;
              positionNullInd = 0;
              salaryNullInd = 0;
           }
           intRow++;
           scratchPad[0] = (Byte)intRow;  // Grabar área reutilizable
           break;

         case (1):   // Case SQLUDF_TF_CLOSE:
           break;

         case (2):   // Case SQLUDF_TF_FINAL:
           break;
      }
   }
}
```

Ejemplo 2: Función escalar en C# del estilo de parámetros SQL:
Este ejemplo muestra lo siguiente:

- Sentencia CREATE FUNCTION para una función escalar del estilo de parámetros SQL

- Código C# para una función escalar del estilo de parámetros SQL

Esta función escalar devuelve un solo valor de cuenta para cada valor de entrada sobre el que actúa. Para un valor de entrada situado en la nª posición del conjunto de valores de entrada, el valor escalar de salida es el valor n. En cada llamada de la función escalar, donde una llamada está asociada con cada fila o valor del conjunto de filas o valores de entrada, la cuenta aumenta en uno y se devuelve el valor actual de la cuenta. Luego, la cuenta se guarda en el almacenamiento intermedio de memoria del área reutilizable para mantener el valor de cuenta entre cada llamada de la función escalar.

Esta función escalar se puede invocar fácilmente si, por ejemplo, se dispone de una tabla definida del modo siguiente:

```
CREATE TABLE T (i1 INTEGER);
INSERT INTO T VALUES 12, 45, 16, 99;
```

Se puede utilizar una consulta simple como la siguiente para invocar la función escalar:

```
SELECT countUp(i1) as count, i1 FROM T;
```

La salida de una consulta como la indicada sería:

```
COUNT           I1
-----------     ----------
1               12
2               45
3               16
4               99
```

Esta UDF escalar es bastante simple. En lugar de devolver sólo la cuenta de las filas, puede utilizar una función escalar que formatee los datos de una columna existente. Por ejemplo, puede añadir una serie a cada valor de una columna de direcciones, puede crear una serie compleja a partir de una cadena de series de entrada o puede efectuar un cálculo matemático complejo con un conjunto de datos donde deberá almacenar un resultado intermedio.

Tabla 26. Código para crear una función escalar en C# del estilo de parámetros SQL

```
CREATE FUNCTION countUp(INTEGER)
RETURNS INTEGER
LANGUAGE CLR
PARAMETER STYLE SQL
SCRATCHPAD 10
FINAL CALL
NO SQL
FENCED
THREADSAFE
NOT DETERMINISTIC
EXECUTION CONTORL SAFE
EXTERNAL NAME 'gwenUDF.dll:bizLogic.empOps!CountUp' ;
```

Tabla 26. Código para crear una función escalar en C# del estilo de parámetros SQL (continuación)

```
class empOps
{
   public static void CountUp(     Int32 input,
                                out Int32 outCounter,
                                    Int16 inputNullInd,
                                out Int16 outCounterNullInd,
                                ref String sqlState,
                                    String funcName,
                                    String specName,
                                ref String sqlMessageText,
                                    Byte[] scratchPad,
                                    Int32 callType)
   {
      Int32 counter = 1;        switch(callType)
      {
         case -1: // case SQLUDF_FIRST_CALL
           scratchPad[0] = (Byte)counter;
           outCounter = counter;
           outCounterNullInd = 0;
           break;
         case 0:  // case SQLUDF_NORMAL_CALL:
           counter = (Int32)scratchPad[0];
           counter = counter + 1;
           outCounter = counter;
           outCounterNullInd = 0;
           scratchPad[0] =
           (Byte)counter;
           break;
         case 1:  // case SQLUDF_FINAL_CALL:
           counter =
           (Int32)scratchPad[0];
           outCounter = counter;
           outCounterNullInd = 0;
           break;
         default: // Nunca se debe entrar aquí
                  // * Necesario para que durante la compilación se
                  //   establezca siempre el parámetro de salida outCounter *
           outCounter = (Int32)(0);
           outCounterNullInd = -1;
           sqlState="ABCDE";
           sqlMessageText = "Should not get here: Default
           case!";
           break;
      }
   }
}
```

Conceptos relacionados:

- "Rutinas de ejecución en el lenguaje común (CLR) .NET" en la página 63
- "Funciones escalares externas" en la página 26
- "Funciones de tabla definidas por el usuario" en *Desarrollo de SQL y rutinas externas*

Tareas relacionadas:

- "Ejemplos de procedimientos CLR .NET en C#" en la página 103
- "Creación de rutinas CLR .NET desde ventanas de mandatos de DB2" en la página 77
- "Creación de rutinas CLR (Common Language Runtime) .NET" en la página 80

- "Ejemplos de rutinas .NET CLR" en la página 93

Ejemplos relacionados:

- "SpCreate.db2 -- Creates the external procedures implemented in spserver.cs"
- "SpServer.cs -- C# external code implementation of procedures created in spcat.db2"
- "SpCreate.db2 -- Creates the external procedures implemented in spserver.vb"
- "SpServer.vb -- VB.NET implementation of procedures created in SpCat.db2"

Capítulo 6. IBM OLE DB Provider para DB2

IBM OLE DB Provider para DB2

Microsoft OLE DB es un conjunto de interfaces OLE/COM que proporciona a las aplicaciones un acceso uniforme a datos almacenados en distintas fuentes de información. La arquitectura OLE DB define a los consumidores de OLE DB y a los proveedores de OLE DB. Un consumidor de OLE DB puede ser cualquier sistema o aplicación que utiliza interfaces OLE DB; un proveedor de OLE DB es un componente que expone las interfaces OLE DB.

IBM OLE DB Provider para DB2 permite que DB2 actúe como un gestor de recursos para el proveedor de OLE DB. Este soporte ofrece a las aplicaciones basadas en OLE DB la posibilidad de extraer o consultar datos de DB2 mediante la interfaz OLE. IBM OLE DB Provider para DB2, cuyo nombre de proveedor es IBMDADB2, permite a los consumidores de OLE DB acceder a datos en un servidor de bases de datos de DB2. Si DB2 Connect está instalado, estos

consumidores de OLE DB también podrán acceder a datos contenidos en sistemas principales DBMS, tales como DB2 para MVS, DB2 para VM/VSE o SQL/400.

IBM OLE DB Provider para DB2 ofrece las siguientes funciones:

- Nivel de soporte 0 de la especificación de proveedor de OLE DB, incluidas algunas interfaces adicionales de nivel 1.
- Una implantación del proveedor de hebras libres, que permite a la aplicación crear componentes en una hebra y utilizar dichos componentes en cualquier otra hebra.
- Un Servicio de búsqueda de errores que devuelve mensajes de error de DB2.

Tenga en cuenta que IBM OLE DB Provider reside en el cliente, y es distinto de las funciones de tabla OLE DB, que también reciben soporte de sistemas de bases de datos de DB2.

Las siguientes secciones de este documento describen la implantación específica de IBM OLE DB Provider para DB2. Para obtener más información sobre la especificación OLE DB 2.0 de Microsoft, consulte el manual ″Microsoft OLE DB 2.0 Programmer's Reference and Data Access SDK″, publicado por Microsoft Press.

Cumplimiento de las versiones:

IBM OLE DB Provider para DB2 se ajusta a la Versión 2.7 de la especificación OLE DB de Microsoft.

Requisitos del sistema:

Consulte la carta de presentación correspondiente a IBM OLE DB Provider para Servidores DB2 para ver qué sistemas operativos Windows tienen soporte.

Para instalar IBM OLE DB Provider para DB2, debe ejecutar primero uno de los sistemas operativos soportados mencionados anteriormente. También necesita instalar el Cliente DB2. Este cliente incluye Microsoft Data Access Components (MDAC).

Información relacionada:

- "Soporte de IBM OLE DB para componentes e interfaces de OLE DB" en la página 142

Tipos de aplicaciones soportados por IBM OLE DB Provider para DB2

Con IBM OLE DB Provider para DB2, puede crear los siguientes tipos de aplicaciones:

- Aplicaciones ADO, que incluyen:
  - Aplicaciones Microsoft Visual Studio C++
  - Aplicaciones Microsoft Visual Basic
- Aplicaciones ADO.NET que hacen uso de OLE DB .NET Data Provider
- Aplicaciones C/C++ que acceden a IBMDADB2 directamente mediante las interfaces OLE DB, incluidas aplicaciones ATL cuyos Objetos de consumidor de acceso a datos se han generado mediante ATL COM AppWizard.

Servicios OLE DB

Las secciones siguientes describen los servicios OLE DB.

Modelo de hebra soportado por IBM OLE DB Provider

IBM OLE DB2 Provider para DB2 da soporte al modelo con hebras libre. Esto permite a las aplicaciones crear componentes en una hebra y utilizar estos componentes en cualquier otra hebra.

Manipulación de objetos grandes con IBM OLE DB Provider

Para obtener y establecer datos como objetos de almacenamiento (`DBTYPE_IUNKNOWN`) con el proveedor de IBMDADB2, utilice la interfaz de ISequentialStream de la forma siguiente:

- Para vincular un objeto de almacenamiento a un parámetro, DBOBJECT en la estructura DBBINDING sólo puede contener el valor `STGM_READ` para el campo dwFlag. IBMDADB2 ejecutará el método Read de la interfaz ISequentialStream del objeto vinculado.
- Para obtener datos de un objeto de almacenamiento, la aplicación debe ejecutar el método Read en la interfaz ISequentialStream del objeto de almacenamiento.
- Cuando se obtienen datos, el valor de la parte de longitud es la longitud de los datos reales, no la longitud del puntero IUnknown.

Conjuntos de filas de esquema soportados por IBM OLE DB Provider

La tabla siguiente muestra los conjuntos de filas de esquema soportados por `IDBSchemaRowset`. Las columnas no soportadas se establecerán en `nulo` eb los conjuntos de filas.

Tabla 27. Conjuntos de filas soportados por IBM OLE DB Provider para DB2

GUID soportados	Restricciones soportadas	Columnas soportadas	Notas
DBSCHEMA _COLUMN_PRIVILEGES	COLUMN_NAME TABLE_NAME TABLE_SCHEMA	COLUMN_NAME GRANTEE GRANTOR IS_GRANTABLE PRIVILEGE_TYPE TABLE_NAME TABLE_SCHEMA	
DBSCHEMA_COLUMNS	COLUMN_NAME TABLE_NAME TABLE_SCHEMA	CHARACTER_MAXIMUM_LENGTH CHARACTER_OCTET_LENGTH COLUMN_DEFAULT COLUMN_FLAGS COLUMN_HASDEFAULT COLUMN_NAME DATA_TYPE DESCRIPTION IS_NULLABLE NUMERIC_PRECISION NUMERIC_SCALE ORDINAL_POSITION TABLE_NAME TABLE_SCHEMA	

Tabla 27. Conjuntos de filas soportados por IBM OLE DB Provider para DB2 (continuación)

GUID soportados	Restricciones soportadas	Columnas soportadas	Notas
DBSCHEMA_FOREIGN_KEYS	FK_TABLE_NAME FK_TABLE_SCHEMA PK_TABLE_NAME PK_TABLE_SCHEMA	DEFERRABILITY DELETE_RULE FK_COLUMN_NAME FK_NAME FK_TABLE_NAME FK_TABLE_SCHEMA ORDINAL PK_COLUMN_NAME PK_NAME PK_TABLE_NAME PK_TABLE_SCHEMA UPDATE_RULE	Se debe especificar al menos una de las siguientes restricciones: PK_TABLE_NAME o FK_TABLE_NAME No se permiten caracteres comodín "%".
DBSCHEMA_INDEXES	TABLE_NAME TABLE_SCHEMA	CARDINALITY CLUSTERED COLLATION COLUMN_NAME INDEX_NAME INDEX_SCHEMA ORDINAL_POSITION PAGES TABLE_NAME TABLE_SCHEMA TYPE UNIQUE	No se permite ningún orden de clasificación. El orden de clasificación, si se especifica, se pasará por alto.
DBSCHEMA_PRIMARY_KEYS	TABLE_NAME TABLE_SCHEMA	COLUMN_NAME ORDINAL PK_NAME TABLE_NAME TABLE_SCHEMA	Se debe especificar al menos la siguiente restricción: TABLE_NAME No se permiten caracteres comodín "%".
DBSCHEMA _PROCEDURE_PARAMETERS	PARAMETER_NAME PROCEDURE_NAME PROCEDURE_SCHEMA	CHARACTER_MAXIMUM_LENGTH CHARACTER_OCTET_LENGTH DATA_TYPE DESCRIPTION IS_NULLABLE NUMERIC_PRECISION NUMERIC_SCALE ORDINAL_POSITION PARAMETER_DEFAULT PARAMETER_HASDEFAULT PARAMETER_NAME PARAMETER_TYPE PROCEDURE_NAME PROCEDURE_SCHEMA TYPE_NAME	
DBSCHEMA_PROCEDURES	PROCEDURE_NAME PROCEDURE_SCHEMA	DESCRIPTION PROCEDURE_NAME PROCEDURE_SCHEMA PROCEDURE_TYPE	
DBSCHEMA_PROVIDER_TYPES	DATA_TYPE BEST_MATCH	AUTO_UNIQUE_VALUE BEST_MATCH CASE_SENSITIVE CREATE_PARAMS COLUMN_SIZE DATA_TYPE FIXED_PREC_SCALE IS_FIXEDLENGTH IS_LONG IS_NULLABLE LITERAL_PREFIX LITERAL_SUFFIX LOCAL_TYPE_NAME MINIMUM_SCALE MAXIMUM_SCALE SEARCHABLE TYPE_NAME UNSIGNED_ATTRIBUTE	

Tabla 27. Conjuntos de filas soportados por IBM OLE DB Provider para DB2 (continuación)

GUID soportados	Restricciones soportadas	Columnas soportadas	Notas
DBSCHEMA_STATISTICS	TABLE_NAME TABLE_SCHEMA	CARDINALITY TABLE_NAME TABLE_SCHEMA	No se permite ningún orden de clasificación. El orden de clasificación, si se especifica, se pasará por alto.
DBSCHEMA _TABLE_PRIVILEGES	TABLE_NAME TABLE_SCHEMA	GRANTEE GRANTOR IS_GRANTABLE PRIVILEGE_TYPE TABLE_NAME TABLE_SCHEMA	
DBSCHEMA_TABLES	TABLE_NAME TABLE_SCHEMA TABLE_TYPE	DESCRIPTION TABLE_NAME TABLE_SCHEMA TABLE_TYPE	

Habilitación automática de servicios OLE DB por parte de IBM OLE DB Provider

Por omisión, IBM OLE DB Provider para DB2 habilita automáticamente todos los servicios OLE DB, añadiendo una entrada de registro `OLEDB_SERVICES` bajo el ID de clase (CLSID) del proveedor con el valor DWORD de `0xFFFFFFFF`. El significado de este valor es el siguiente:

Tabla 28. Servicios OLE DB

Servicios habilitados	Valor de DWORD
Todos los servicios (valor por omisión)	`0xFFFFFFFF`
Todos excepto agrupación y AutoEnlistment	`0xFFFFFFFC`
Todos excepto cursor del cliente	`0xFFFFFFFB`
Todos excepto agrupación, alistamiento y cursor	`0xFFFFFFF8`
Ningún servicio	`0x000000000`

Servicios de datos

Las secciones siguientes describen consideraciones sobre los servicios de datos.

Modalidades de cursor soportadas para IBM OLE DB Provider

IBM OLE DB Provider para DB2 da soporte nativo a cursores de sólo lectura, de sólo avance, desplazables y actualizables y actualizables y desplazables.

Correlaciones de tipos de datos entre DB2 y OLE DB

IBM OLE DB Provider para DB2 da soporte a las correlaciones de tipos de datos entre tipos de datos DB2 y tipos de datos OLE DB. La tabla siguiente proporciona una lista completa de correlaciones soportadas y nombres disponibles para indicar los tipos de datos de columnas y parámetros.

Tabla 29. Correlaciones de tipos de datos DB2 y tipos de datos OLE DB

Tipos de datos DB2	Indicadores de tipos de datos OLE DB	Nombres de tipos estándar de OLE DB	Nombres específicos de DB2
SMALLINT	DBTYPE_I2	"DBTYPE_I2"	"SMALLINT"
INTEGER	DBTYPE_I4	"DBTYPE_I4"	"INTEGER" o "INT"

Tabla 29. Correlaciones de tipos de datos DB2 y tipos de datos OLE DB (continuación)

Tipos de datos DB2	Indicadores de tipos de datos OLE DB	Nombres de tipos estándar de OLE DB	Nombres específicos de DB2
BIGINT	DBTYPE_I8	"DBTYPE_I8"	"BIGINT"
REAL	DBTYPE_R4	"DBTYPE_R4"	"REAL"
FLOAT	DBTYPE_R8	"DBTYPE_R8"	"FLOAT"
DOUBLE	DBTYPE_R8	"DBTYPE_R8"	"DOUBLE" o "DOUBLE PRECISION"
DECIMAL	DBTYPE_NUMERIC	"DBTYPE_NUMERIC"	"DEC" o "DECIMAL"
NUMERIC	DBTYPE_NUMERIC	"DBTYPE_NUMERIC"	"NUM" o "NUMERIC"
DATE	DBTYPE_DBDATE	"DBTYPE_DBDATE"	"DATE"
TIME	DBTYPE_DBTIME	"DBTYPE_DBTIME"	"TIME"
TIMESTAMP	DBTYPE_DBTIMESTAMP	"DBTYPE_DBTIMESTAMP"	"TIMESTAMP"
CHAR	DBTYPE_STR	"DBTYPE_CHAR"	"CHAR" o "CHARACTER"
VARCHAR	DBTYPE_STR	"DBTYPE_VARCHAR"	"VARCHAR"
LONG VARCHAR	DBTYPE_STR	"DBTYPE_LONGVARCHAR"	"LONG VARCHAR"
CLOB	DBTYPE_STR y DBCOLUMNFLAGS_ISLONG o DBPARAMFLAGS_ISLONG	"DBTYPE_CHAR" "DBTYPE_VARCHAR" "DBTYPE_LONGVARCHAR" y DBCOLUMNFLAGS_ISLONG o DBPARAMFLAGS_ISLONG	"CLOB"
GRAPHIC	DBTYPE_WSTR	"DBTYPE_WCHAR"	"GRAPHIC"
VARGRAPHIC	DBTYPE_WSTR	"DBTYPE_WVARCHAR"	"VARGRAPHIC"
LONG VARGRAPHIC	DBTYPE_WSTR	"DBTYPE_WLONGVARCHAR"	"LONG VARGRAPHIC"
DBCLOB	DBTYPE_WSTR y DBCOLUMNFLAGS_ISLONG o DBPARAMFLAGS_ISLONG	"DBTYPE_WCHAR" "DBTYPE_WVARCHAR" "DBTYPE_WLONGVARCHAR" y DBCOLUMNFLAGS_ISLONG o DBPARAMFLAGS_ISLONG	"DBCLOB"
CHAR(n) FOR BIT DATA	DBTYPE_BYTES	"DBTYPE_BINARY"	
VARCHAR(n) FOR BIT DATA	DBTYPE_BYTES	"DBTYPE_VARBINARY"	
LONG VARCHAR FOR BIT DATA	DBTYPE_BYTES	"DBTYPE_LONGVARBINARY"	
BLOB	DBTYPE_BYTES y DBCOLUMNFLAGS_ISLONG o DBPARAMFLAGS_ISLONG	"DBTYPE_BINARY" "DBTYPE_VARBINARY" "DBTYPE_LONGVARBINARY" y DBCOLUMNFLAGS_ISLONG o DBPARAMFLAGS_ISLONG	"BLOB"
DATA LINK	DBTYPE_STR	"DBTYPE_CHAR"	"DATA LINK"

Conversión de datos para establecer datos de tipos OLE DB a tipos DB2

IBM OLE DB Provider para DB2 da soporte a las conversiones de datos para establecer datos de tipos OLE DB a tipos DB2. Tenga en cuenta que es posible que se trunquen datos en algunos casos, en función de los tipos y del valor de los datos.

Tabla 30. Conversiones de datos de tipos OLE DB a tipos DB2

	Tipos de datos DB2																					
																		For Bit Data				
Indicador de tipo OLE DB	**SMALLINT**	**INTEGER**	**BIGINT**	**REAL**	**FLOAT DOUBLE**	**DECIMAL NUMERIC**	**DATE**	**TIME**	**TIMESTAMP**	**CHAR**	**VARCHAR**	**LONG VARCHAR**	**CLOB**	**GRAPHIC**	**VARGRAPHIC**	**LONG VARGRAPHIC**	**DBCLOB**	**CHAR**	**VARCHAR**	**LONG VARCHAR**	**BLOB**	**DATALINK**
DBTYPE_EMPTY																						
DBTYPE_NULL																						
DBTYPE_RESERVED																						
DBTYPE_I1	X	X	X	X	X	X				X	X											
DBTYPE_I2	X	X	X	X	X	X				X	X											
DBTYPE_I4	X	X	X	X	X	X				X	X											
DBTYPE_I8	X	X	X	X	X	X				X	X											
DBTYPE_UI1	X	X	X	X	X	X				X	X											
DBTYPE_UI2	X	X	X	X	X	X				X	X											
DBTYPE_UI4	X	X	X	X	X	X				X	X											
DBTYPE_UI8	X	X	X	X	X	X				X	X											
DBTYPE_R4	X	X	X	X	X	X				X	X											
DBTYPE_R8	X	X	X	X	X	X				X	X											
DBTYPE_CY																						
DBTYPE_DECIMAL	X	X	X	X	X	X				X	X											
DBTYPE_NUMERIC	X	X	X	X	X	X				X	X											
DBTYPE_DATE																						
DBTYPE_BOOL	X	X	X	X	X	X				X	X											
DBTYPE_BYTES			X			X				X	X	X				X		X	X	X		
DBTYPE_BSTR – por determinar																						
DBTYPE_STR	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X		X	X	X		X
DBTYPE_WSTR														X	X	X						
DBTYPE_VARIANT – por determinar																						
DBTYPE_IDISPATCH																						
DBTYPE_IUNKNOWN										X	X	X	X	X	X	X	X	X	X	X		
DBTYPE_GUID																						
DBTYPE_ERROR																						
DBTYPE_BYREF																						
DBTYPE_ARRAY																						
DBTYPE_VECTOR																						
DBTYPE_UDT																						
DBTYPE_DBDATE							X		X	X	X											

Tabla 30. Conversiones de datos de tipos OLE DB a tipos DB2 (continuación)

	Tipos de datos DB2																					
																		For Bit Data				
Indicador de tipo OLE DB	SMALLINT	INTEGER	BIGINT	REAL	FLOAT DOUBLE	DECIMAL NUMERIC	DATE	TIME	TIMESTAMP	CHAR	VARCHAR	LONG VARCHAR	CLOB	GRAPHIC	VARGRAPHIC	LONG VARGRAPHIC	DBCLOB	CHAR	VARCHAR	LONG VARCHAR	BLOB	DATALINK
DBTYPE_DBTIME								X	X	X	X											
DBTYPE_DBTIMESTAMP							X	X	X	X	X											
DBTYPE_FILETIME																						
DBTYPE_PROP_VARIANT																						
DBTYPE_HCHAPTER																						
DBTYPE_VARNUMERIC																						

Información relacionada:

- "Conversión de datos para establecer datos de tipos DB2 en tipos OLE DB" en la página 140

Conversión de datos para establecer datos de tipos DB2 en tipos OLE DB

Para obtener datos, IBM OLE DB Provider permite conversiones de datos de tipos DB2 a tipos OLE DB. Tenga en cuenta que es posible que se trunquen datos en algunos casos, en función de los tipos y del valor de los datos.

Tabla 31. Conversiones de datos de tipos DB2 a tipos OLE DB

	Tipos de datos DB2																					
																		For Bit Data				
Indicador de tipo OLE DB	SMALLINT	INTEGER	BIGINT	REAL	FLOAT DOUBLE	DECIMAL NUMERIC	DATE	TIME	TIMESTAMP	CHAR	VARCHAR	LONG VARCHAR	CLOB	GRAPHIC	VARGRAPHIC	LONG VARGRAPHIC	DBCLOB	CHAR	VARCHAR	LONG VARCHAR	BLOB	DATALINK
DBTYPE_EMPTY																						
DBTYPE_NULL																						
DBTYPE_RESERVED																						
DBTYPE_I1	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X

Tabla 31. Conversiones de datos de tipos DB2 a tipos OLE DB (continuación)

	Tipos de datos DB2																						
																			For Bit Data				
Indicador de tipo OLE DB	**SMALLINT**	**INTEGER**	**BIGINT**	**REAL**	**FLOAT DOUBLE**	**DECIMAL NUMERIC**	**DATE**	**TIME**	**TIMESTAMP**	**CHAR**	**VARCHAR**	**LONG VARCHAR**	**CLOB**	**GRAPHIC**	**VARGRAPHIC**	**LONG VARGRAPHIC**	**DBCLOB**	**CHAR**	**VARCHAR**	**LONG VARCHAR**	**BLOB**	**DATALINK**
DBTYPE_I2	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_I4	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_I8	X	X	X	X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_UI1	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_UI2	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_UI4	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_UI8	X	X	X	X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_R4	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_R8	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_CY	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_DECIMAL	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_NUMERIC	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_DATE	X	X		X	X		X	X	X	X	X	X		X	X	X						X
DBTYPE_BOOL	X	X		X	X	X				X	X	X		X	X	X		X	X	X		X
DBTYPE_BYTES	X	X		X	X	X	X	X	X	X	X	X		X	X	X		X	X	X		X
DBTYPE_BSTR	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X		X	X	X		X
DBTYPE_STR	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X		X	X	X		X
DBTYPE_WSTR	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X		X	X	X		X
DBTYPE_VARIANT	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X		X	X	X		X
DBTYPE_IDISPATCH																						
DBTYPE_IUNKNOWN	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
DBTYPE_GUID										X	X	X		X	X	X		X	X	X		X
DBTYPE_ERROR																						
DBTYPE_BYREF																						
DBTYPE_ARRAY																						
DBTYPE_VECTOR																						
DBTYPE_UDT																						
DBTYPE_DBDATE							X	X	X	X	X	X		X	X	X		X	X	X		X
DBTYPE_DBTIME							X	X	X	X	X	X		X	X	X						X
DBTYPE_DBTIMESTAMP							X	X	X	X	X	X		X	X	X		X	X	X		X
DBTYPE_FILETIME			X				X	X	X	X	X	X		X	X	X		X	X	X		X
DBTYPE_PROP_VARIANT	X	X	X	X	X					X	X	X		X	X	X		X	X	X		X
DBTYPE_HCHAPTER																						
DBTYPE_VARNUMERIC																						

Tabla 31. Conversiones de datos de tipos DB2 a tipos OLE DB (continuación)

Indicador de tipo OLE DB	Tipos de datos DB2																				
	SMALLINT	INTEGER	BIGINT	REAL	FLOAT DOUBLE	DECIMAL NUMERIC	DATE	TIME	TIMESTAMP	CHAR	VARCHAR	LONG VARCHAR	CLOB	GRAPHIC	VARGRAPHIC	LONG VARGRAPHIC	DBCLOB	For Bit Data			BLOB	DATALINK
																		CHAR	VARCHAR	LONG VARCHAR		

Nota: Cuando la aplicación realiza una operación ISequentialStream::Read para obtener los datos del objeto de almacenamiento, el formato de los datos devueltos depende del tipo de datos de la columna:

- Para tipos de datos binarios y que no sean de tipo carácter, los datos de la columna se exponen como una secuencia de bytes que representan dichos valores en el sistema operativo.
- Para tipos de datos de carácter, primero los datos se convierten a DBTYPE_STR.
- Para DBCLOB, primero los datos se convierten a DBTYPE_WCHAR.

Información relacionada:

- "Conversión de datos para establecer datos de tipos OLE DB a tipos DB2" en la página 138

Restricciones de IBM OLE DB Provider

A continuación se describen las restricciones de IBM OLE DB Provider:

- IBMDADB2 da soporte al ámbito de transacciones de confirmación automática y controladas por el usuario con la interfaz `ITransactionLocal`. El ámbito de transacciones de confirmación automática es el ámbito por omisión. Las transacciones anidadas no reciben soporte.
- `RestartPosition` no recibe soporte si el texto del mandato contiene parámetros.
- IBMDADB2 no pone entre comillas nombres de tablas que se pasan a través de parámetros de DBID, que son los parámetros que utiliza la interfaz `IOpenRowset`. En su lugar, el consumidor de OLE DB debe añadir comillas a los nombres de tablas cuando sea necesario.

Soporte de IBM OLE DB para componentes e interfaces de OLE DB

La tabla siguiente lista los componentes y las interfaces de OLE DB que reciben soporte de IBM OLE DB Provider para DB2 y de Microsoft OLE DB Provider para ODBC.

Tabla 32. Comparación de interfaces y componentes de OLE DB soportados por IBM OLE DB Provider para DB2 y Microsoft OLE DB Provider para ODBC

	Interfaz	DB2	ODBC Provider
BLOB			
	ISequentialStream	Sí	Sí
Mandato			
	IAccessor	Sí	Sí

Tabla 32. Comparación de interfaces y componentes de OLE DB soportados por IBM OLE DB Provider para DB2 y Microsoft OLE DB Provider para ODBC (continuación)

	Interfaz	**DB2**	**ODBC Provider**
	ICommand	Sí	Sí
	ICommandPersist	No	No
	ICommandPrepare	Sí	Sí
	ICommandProperties	Sí	Sí
	ICommandText	Sí	Sí
	ICommandWithParameters	Sí	Sí
	IColumnsInfo	Sí	Sí
	IColumnsRowset	Sí	Sí
	IConvertType	Sí	Sí
	ISupportErrorInfo	Sí	Sí
Fuente de datos			
	IConnectionPoint	No	Sí
	IDBAsynchNotify (consumer)	No	No
	IDBAsynchStatus	No	No
	IDBConnectionPointContainer	No	Sí
	IDBCreateSession	Sí	Sí
	IDBDataSourceAdmin	No	No
	IDBInfo	Sí	Sí
	IDBInitialize	Sí	Sí
	IDBProperties	Sí	Sí
	IPersist	Sí	No
	IPersistFile	Sí	Sí
	ISupportErrorInfo	Sí	Sí
Enumerador			
	IDBInitialize	Sí	Sí
	IDBProperties	Sí	Sí
	IParseDisplayName	Sí	No
	ISourcesRowset	Sí	Sí
	ISupportErrorInfo	Sí	Sí
Servicio de búsqueda de errores			
	IErrorLookUp	Sí	Sí
Objeto Error			
	IErrorInfo	Sí	No
	IErrorRecords	Sí	No
	ISQLErrorInfo (custom)	Sí	No
Varios resultados			
	IMultipleResults	Sí	Sí
	ISupportErrorInfo	Sí	Sí
Conjunto de filas			

Tabla 32. Comparación de interfaces y componentes de OLE DB soportados por IBM OLE DB Provider para DB2 y Microsoft OLE DB Provider para ODBC (continuación)

	Interfaz	**DB2**	**ODBC Provider**
	IAccessor	Sí	Sí
	IColumnsRowset	Sí	Sí
	IColumnsInfo	Sí	Sí
	IConvertType	Sí	Sí
	IChapteredRowset	No	No
	IConnectionPointContainer	Sí	Sí
	IDBAsynchStatus	No	No
	IParentRowset	No	No
	IRowset	Sí	Sí
	IRowsetChange	Sí	Sí
	IRowsetChapterMember	No	No
	IRowsetFind	No	No
	IRowsetIdentity	Sí	Sí
	IRowsetIndex	No	No
	IRowsetInfo	Sí	Sí
	IRowsetLocate	Sí	Sí
	IRowsetNotify (consumer)	Sí	No
	IRowsetRefresh	Componente Servicio de cursor	Sí
	IRowsetResynch	Componente Servicio de cursor	Sí
	IRowsetScroll	Sí[1]	Sí
	IRowsetUpdate	Componente Servicio de cursor	Sí
	IRowsetView	No	No
	ISupportErrorInfo	Sí	Sí
Notas: 1. Los valores a devolver son aproximaciones. Las filas suprimidas no se pasarán por alto.			
Sesión			
	IAlterIndex	No	No
	IAlterTable	No	No
	IDBCreateCommand	Sí	Sí
	IDBSchemaRowset	Sí	Sí
	IGetDataSource	Sí	Sí
	IIndexDefinition	No	No
	IOpenRowset	Sí	Sí
	ISessionProperties	Sí	Sí
	ISupportErrorInfo	Sí	Sí
	ITableDefinition	No	No
	ITableDefinitionWithConstraints	No	No
	ITransaction	Sí	Sí
	ITransactionJoin	Sí	Sí

Tabla 32. Comparación de interfaces y componentes de OLE DB soportados por IBM OLE DB Provider para DB2 y Microsoft OLE DB Provider para ODBC (continuación)

	Interfaz	DB2	ODBC Provider
	ITransactionLocal	Sí	Sí
	ITransactionObject	No	No
	ITransactionOptions	No	Sí
Objetos Vista			
	IViewChapter	No	No
	IViewFilter	No	No
	IViewRowset	No	No
	IViewSort	No	No

Soporte de IBM OLE DB Provider para propiedades de OLE DB

La tabla siguiente muestra las propiedades de OLE DB que reciben soporte de IBM OLE DB Provider para DB2:

Tabla 33. Propiedades soportadas por IBM OLE DB Provider para DB2

Grupo de propiedades	Conjunto de propiedades	Propiedades	Valor por omisión	R/W
Fuente de datos	DBPROPSET_DATASOURCE	DBPROP_MULTIPLECONNECTIONS	VARIANT_FALSE	R
		DBPROP_RESETDATASOURCE	DBPROPVAL_RD_RESETALL	R/W
Fuente de datos DB2	DBPROPSET_DB2DATASOURCE	DB2PROP_REPORTISLONGFORLONGTYPES	VARIANT_FALSE	R/W
		DB2PROP_RETURNCHARASWCHAR	VARIANT_TRUE	R/W
		DB2PROP_SORTBYORDINAL	VARIANT_FALSE	R/W
Información de fuente de datos	DBPROPSET _DATASOURCEINFO	DBPROP_ACTIVESESSIONS	0	R
		DBPROP_ASYNCTXNABORT	VARIANT_FALSE	R
		DBPROP_ASYNCTXNCOMMIT	VARIANT_FALSE	R
		DBPROP_BYREFACCESSORS	VARIANT_FALSE	R
		DBPROP_COLUMNDEFINITION	DBPROPVAL_CD_NOTNULL	R
		DBPROP_CONCATNULLBEHAVIOR	DBPROPVAL_CB_NULL	R
		DBPROP_CONNECTIONSTATUS	DBPROPVAL_CS_INITIALIZED	R
		DBPROP_DATASOURCENAME	N/D	R
		DBPROP_DATASOURCEREADONLY	VARIANT_FALSE	R
		DBPROP_DBMSNAME	N/D	R
		DBPROP_DBMSVER	N/D	R
		DBPROP_DSOTHREADMODEL	DBPROPVAL_RT_FREETHREAD	R
		DBPROP_GROUPBY	DBPROPVAL_GB_CONTAINS_SELECT	R
		DBPROP_IDENTIFIERCASE	DBPROPVAL_IC_UPPER	R
		DBPROP_MAXINDEXSIZE	0	R
		DBPROP_MAXROWSIZE	0	R
		DBPROP_MAXROWSIZEINCLUDESBLOB	VARIANT_TRUE	R
		DBPROP_MAXTABLEINSELECT	0	R
		DBPROP_MULTIPLEPARAMSETS	VARIANT_FALSE	R
		DBPROP_MULTIPLERESULTS	DBPROPVAL_MR_SUPPORTED	R
		DBPROP_MULTIPLESTORAGEOBJECTS	VARIANT_TRUE	R
		DBPROP_MULTITABLEUPDATE	VARIANT_FALSE	R
		DBPROP_NULLCOLLATION	DBPROPVAL_NC_LOW	R
		DBPROP_OLEOBJECTS	DBPROPVAL_OO_BLOB	R
		DBPROP_ORDERBYCOLUMNSINSELECT	VARIANT_FALSE	R
		DBPROP _OUTPUTPARAMETERAVAILABILITY	DBPROPVAL_OA_ATEXECUTE	R

Tabla 33. Propiedades soportadas por IBM OLE DB Provider para DB2 (continuación)

Grupo de propiedades	Conjunto de propiedades	Propiedades	Valor por omisión	R/W
		DBPROP_PERSISTENTIDTYPE	DBPROPVAL_PT_NAME	R
		DBPROP_PREPAREABORTBEHAVIOR	DBPROPVAL_CB_DELETE	R
		DBPROP_PROCEDURETERM	"PROCEDIMIENTO ALMACENADO"	R
		DBPROP_PROVIDERFRIENDLYNAME	"IBM OLE DB Provider para DB2"	R
		DBPROP_PROVIDERNAME	"IBMDADB2.DLL"	R
		DBPROP_PROVIDEROLEDBVER	"02.7"	R
		DBPROP_PROVIDERVER	N/D	R
		DBPROP_QUOTEIDENTIFIERCASE	DBPROPVAL_IC_SENSITIVE	R
		DBPROP _ROWSETCONVERSIONSONCOMMAND	VARIANT_TRUE	R
		DBPROP_SCHEMATERM	"ESQUEMA"	R
		DBPROP_SCHEMAUSAGE	DBPROPVAL_SU_DML_STATEMENTS | DBPROPVAL_SU_TABLE_DEFINITION | DBPROPVAL_SU_INDEX_DEFINITION | DBPROPVAL_SU_PRIVILEGE_DEFINITION	R
		DBPROP_SQLSUPPORT	DBPROPVAL_SQL_ODBC_EXTENDED | DBPROPVAL_SQL_ESCAPECLAUSES | DBPROPVAL_SQL_ANSI92_ENTRY	R
		DBPROP_SERVERNAME	N/D	R
		DBPROP_STRUCTUREDSTORAGE	DBPROPVAL_SS_ISEQUENTIALSTREAM	R
		DBPROP_SUBQUERIES	DBPROPVAL_SQ_CORRELATEDSUBQUERIES | DBPROPVAL_SQ_COMPARISON | DBPROPVAL_SQ_EXISTS | DBPROPVAL_SQ_IN | DBPROPVAL_SQ_QUANTIFIED |	R
		DBPROP_SUPPORTEDTXNDDL	DBPROPVAL_TC_ALL	R
		DBPROP_SUPPORTEDTXNISOLEVELS	DBPROPVAL_TI_CURSORSTABILITY | DBPROPVAL_TI_READCOMMITTED | DBPROPVAL_TI_READUNCOMMITTED | DBPROPVAL_TI_SERIALIZABLE |	R
		DBPROP_SUPPORTEDTXNISORETAIN	DBPROPVAL_TR_COMMIT_DC | DBPROPVAL_TR_ABORT_NO |	R
		DBPROP_TABLETERM	"TABLA"	R
		DBPROP_USERNAME	N/D	R
Inicialización	DBPROPSET_DBINIT	DBPROP_AUTH_PASSWORD	N/D	R/W
		DBPROP_INIT_TIMEOUT (1)	0	R/W
		DBPROP_AUTH_PERSIST _SENSITIVE_AUTHINFO	VARIANT_FALSE	R/W
		DBPROP_AUTH_USERID	N/D	R/W
		DBPROP_INIT_DATASOURCE	N/D	R/W
		DBPROP_INIT_HWND	N/D	R/W
		DBPROP_INIT_MODE	DB_MODE_READWRITE	R/W
		DBPROP_INIT_OLEDBSERVICES	0xFFFFFFFF	R/W
		DBPROP_INIT_PROMPT	DBPROMPT_NOPROMPT	R/W
		DBPROP_INIT_PROVIDERSTRING	N/D	R/W
Conjunto de filas	DBPROPSET_ROWSET	DBPROP_ABORTPRESERVE	VARIANT_FALSE	R
		DBPROP_ACCESSORDER	DBPROPVAL_AO_RANDOM	R
		DBPROP_BLOCKINGSTORAGEOBJECTS	VARIANT_FALSE	R
		DBPROP_BOOKMARKS	VARIANT_FALSE	R/W
		DBPROP_BOOKMARKSKIPPED	VARIANT_FALSE	R
		DBPROP_BOOKMARKTYPE	DBPROPVAL_BMK_NUMERIC	R
		DBPROP_CACHEDEFERRED	VARIANT_FALSE	R/W
		DBPROP_CANFETCHBACKWARDS	VARIANT_FALSE	R/W
		DBPROP_CANHOLDROWS	VARIANT_FALSE	R
		DBPROP_CANSCROLLBACKWARDS	VARIANT_FALSE	R/W
		DBPROP_CHANGEINSERTEDROWS	VARIANT_FALSE	R
		DBPROP_COMMITPRESERVE	VARIANT_TRUE	R/W
		DBPROP_COMMANDTIMEOUT	0	R/W
		DBPROP_DEFERRED	VARIANT_FALSE	R
		DBPROP_IAccessor	VARIANT_TRUE	R
		DBPROP_IColumnsInfo	VARIANT_TRUE	R
		DBPROP_IColumnsRowset	VARIANT_TRUE	R/W

Tabla 33. Propiedades soportadas por IBM OLE DB Provider para DB2 (continuación)

Grupo de propiedades	Conjunto de propiedades	Propiedades	Valor por omisión	R/W
		DBPROP_IConvertType	VARIANT_TRUE	R
		DBPROP_IMultipleResults	VARIANT_FALSE	R/W
		DBPROP_IRowset	VARIANT_TRUE	R
		DBPROP_IRowChange	VARIANT_FALSE	R/W
		DBPROP_IRowsetFind	VARIANT_FALSE	R
		DBPROP_IRowsetIdentity	VARIANT_TRUE	R
		DBPROP_IRowsetInfo	VARIANT_TRUE	R
		DBPROP_IRowsetLocate	VARIANT_FALSE	R/W
		DBPROP_IRowsetScroll	VARIANT_FALSE	R/W
		DBPROP_IRowsetUpdate	VARIANT_FALSE	R
		DBPROP_ISequentialStream	VARIANT_TRUE	R
		DBPROP_ISupportErrorInfo	VARIANT_TRUE	R
		DBPROP_LITERALBOOKMARKS	VARIANT_FALSE	R
		DBPROP_LITERALIDENTITY	VARIANT_TRUE	R
		DBPROP_LOCKMODE	DBPROPVAL_LM_SINGLEROW	R/W
		DBPROP_MAXOPENROWS	32767	R
		DBPROP_MAXROWS	0	R/W
		DBPROP_NOTIFICATIONGRANULARITY	DBPROPVAL_NT_SINGLEROW	R/W
		DBPROP_NOTIFICATION PHASES	DBPROPVAL_NP_OKTODO DBPROPBAL_NP_ABOUTTODO DBPROPVAL_NP_SYNCHAFTER DBPROPVAL_NP_FAILEDTODO DBPROPVAL_NP_DIDEVENT	R
		DBPROP_NOTIFYROWSETRELEASE	DBPROPVAL_NP_OKTODO DBPROPVAL_NP_ABOUTTODO	R
		DBPROP _NOTIFYROWSETFETCHPOSITIONCHANGE	DBPROPVAL_NP_OKTODO DBPROPVAL_NP_ABOUTTODO	R
		DBPROP_NOTIFYCOLUMNSET	DBPROPVAL_NP_OKTODO DBPROPVAL_NP_ABOUTTODO	R
		DBPROP_NOTIFYROWDELETE	DBPROPVAL_NP_OKTODO DBPROPVAL_NP_ABOUTTODO	R
		DBPROP_NOTIFYROWINSERT	DBPROPVAL_NP_OKTODO DBPROPVAL_NP_ABOUTTODO	R
		DBPROP_ORDEREDBOOKMARKS	VARIANT_FALSE	R
		DBPROP_OTHERINSERT	VARIANT_FALSE	R
		DBPROP_OTHERUPDATEDELETE	VARIANT_FALSE	R/W
		DBPROP_OWNINSERT	VARIANT_FALSE	R
		DBPROP_OWNUPDATEDELETE	VARIANT_FALSE	R
		DBPROP_QUICKRESTART	VARIANT_FALSE	R/W
		DBPROP_REMOVEDELETED	VARIANT_FALSE	R/W
		DBPROP_ROWTHREADMODEL	DBPROPVAL_RT_FREETHREAD	R
		DBPROP_SERVERCURSOR	VARIANT_TRUE	R
		DBPROP_SERVERDATAONINSERT	VARIANT_FALSE	R
		DBPROP_UNIQUEROWS	VARIANT_FALSE	R/W
		DBPROP_UPDATABILITY	0	R/W
Conjunto de filas	DBPROPSET_DB2ROWSET	DBPROP_ISLONGMINLENGTH	32000	R/W
Sesión	DBPROPSET_SESSION	DBPROP_SESS_AUTOCOMMITISOLEVELS	DBPROPVAL_TI_CURSORSTABILITY	R/W

Notas:

1. El tiempo de espera excedido sólo puede aplicarse al utilizar el protocolo TCP/IP para conectarse al servidor. El tiempo de espera excedido sólo se aplica durante la conexión de sock de TCP/IP. Si la conexión de sock se completa antes de que caduque el tiempo de espera especificado, el tiempo de espera no podrá aplicarse al resto del proceso de inicialización. Si se utiliza la característica de redireccionamiento-cliente se doblará el tiempo de espera excedido. En general, cuando se habilita el redireccionamiento, dicho redireccionamiento de cliente determinará el comportamiento del tiempo de espera excedido de la conexión.

Conexiones a fuentes de datos mediante IBM OLE DB Provider

Los ejemplos siguientes muestran cómo conectar con una fuente de datos de DB2 utilizando IBM OLE DB Provider para DB2:

Ejemplo 1: Aplicación Visual Basic que utiliza ADO:

```
Dim db As ADODB.Connection
Set db = New ADODB.Connection
db.Provider = "IBMDADB2"
db.CursorLocation = adUseClient
...
```

Ejemplo 2: Aplicación C/C++ que utiliza IDBPromptInitialize y Data Links:

```
// Crear DataLinks
hr = CoCreateInstance (
     CLSID_DataLinks,
     NULL,
     CLSCTX_INPROC_SERVER,
     IID_IDBPromptInitialize,
     (void**)&pIDBPromptInitialize);

// Invocar la UI DataLinks para seleccionar el proveedor y la fuente de datos
hr = pIDBPromptInitialize->PromptDataSource (
     NULL,
     GetDesktopWindow(),
     DBPROMPTOPTIONS_PROPERTYSHEET,
     0,
     NULL,
     NULL,
     IID_IDBInitialize,
     (IUnknown**)&pIDBInitialize);
```

Ejemplo 3: Aplicación C/C++ que utiliza IDataInitialize y el componente de servicio:

```
hr = CoCreateInstance (
     CLSID_MSDAINITIALIZE,
     NULL,
     CLSCTX_INPROC_SERVER,
     IID_IDataInitialize,
     (void**)&pIDataInitialize);

hr = pIDataInitialize->CreateDBInstance(
     CLSID_IBMDADB2, // ClassID de IBMDADB2
     NULL,
     CLSCTX_INPROC_SERVER,
     NULL,
     IID_IDBInitialize,
     (IUnknown**)&pIDBInitialize);
```

Aplicaciones ADO

Las siguientes secciones describen consideraciones sobre aplicaciones ADO.

Palabras clave de series de conexión de ADO

Para especificar palabras clave de series de conexión de ADO (Objetos de datos ActiveX), especifique la palabra clave con el formato `palabra clave=`*`valor`* en la serie (conexión) del proveedor. Delimite varias palabras clave con un punto y coma (;).

La tabla siguiente describe las palabras clave a las que da soporte IBM OLE DB Provider para DB2:

Tabla 34. Palabras clave soportadas por IBM OLE DB Provider para DB2

Palabra clave	Valor	Significado
DSN	Nombre del alias de la base de datos	El alias de la base de datos DB2 en el directorio de bases de datos.
UID	ID de usuario	El ID de usuario que se utiliza para conectar con el servidor DB2.
PWD	Contraseña de UID	Contraseña correspondiente al ID de usuario utilizado para conectar con el servidor DB2.

Hay otras palabras clave de configuración de CLI de DB2 que también afectan al comportamiento de IBM OLE DB Provider.

Información relacionada:

- "CLI/ODBC configuration keywords listing by category" en *Call Level Interface Guide and Reference, Volume 1*

Conexiones a fuentes de datos con aplicaciones ADO Visual Basic

Para conectar con una fuente de datos DB2 utilizando IBM OLE DB Provider para DB2, especifique el nombre del proveedor de IBMDADB2.

Conceptos relacionados:

- "Conexiones a fuentes de datos mediante IBM OLE DB Provider" en la página 148

Tareas relacionadas:

- "Creación de aplicaciones ADO con Visual Basic" en la página 155

Cursores desplazables actualizables en aplicaciones ADO

IBM OLE DB Provider para DB2 da soporte nativo a cursores de sólo lectura, de sólo avance, desplazables de sólo lectura, así como actualizables y desplazables. Una aplicación ADO que desea acceder a cursores desplazables actualizables puede establecer la ubicación del cursor en `adUseClient` o `adUseServer`. Si establece la ubicación del cursor en `adUseServer`, hace que el cursor se materialice en el servidor.

Limitaciones para aplicaciones ADO

A continuación se describen las limitaciones de las aplicaciones ADO:

- Las aplicaciones ADO que llaman a procedimientos almacenados deben crear y vincular de forma explícita sus parámetros. El método `Parameters.Refresh` para generar parámetros automáticamente no está soportado para DB2 Server para VSE y VM.
- No se da soporte a valores por omisión de parámetros.
- Al insertar una fila nueva utilizando un cursor desplazable en el lado del servidor, utilice el método AddNew() con los argumentos Fieldlist y Values. Esto

es más eficaz que invocar AddNew() sin argumentos a continuación de llamadas Update() para cada columna. Cada llamada AddNew() y Update() es una petición independiente para el servidor y por tanto, es menos eficaz que una única llamada a AddNew().

- Las filas recién insertadas no pueden actualizarse con un cursor desplazable en el extremo del servidor.
- Las tablas con datos largos, LOB, o columnas Datalink no pueden actualizarse al utilizar un cursor desplazable en el extremo del servidor.

Soporte de IBM OLE DB Provider para propiedades y métodos ADO

IBM OLE DB Provider da soporte a los siguientes métodos y propiedades ADO:

Tabla 35. Métodos y propiedades ADO soportados por IBM OLE DB Provider para DB2

ADO	Método/Propiedad	Interfaz/Propiedad OLE DB	Soporte de IBM OLE DB
Métodos de mandato	Cancel	ICommand	Sí
	CreateParameter		Sí
	Execute		Sí
Propiedades de mandato	ActiveConnection	(Específico de ADO)	
	Command Text	ICommandText	Sí
	Command Timeout	ICommandProperties::SetProperties DBPROP_COMMANDTIMEOUT	Sí
	CommandType	(Específico de ADO)	
	Prepared	ICommandPrepare	Sí
	State	(Específico de ADO)	
Grupo de mandatos	Parameters	ICommandWithParameter DBSCHEMA _PROCEDURE_PARAMETERS	Sí
	Properties	ICommandProperties IDBProperties	Sí
Métodos de conexión	BeginTrans CommitTrans RollbackTrans	ITransactionLocal	Sí (pero no anidado) Sí (pero no anidado) Sí (pero no anidado)
	Execute	ICommand IOpenRowset	Sí
	Open	IDBCreateSession IDBInitialize	Sí
	OpenSchema adSchemaColumnPrivileges adSchemaColumns adSchemaForeignKeys adSchemaIndexes adSchemaPrimaryKeys adSchemaProcedureParam adSchemaProcedures adSchemaProviderType adSchemaStatistics adSchemaTablePrivileges adSchemaTables	IDBSchemaRowset	 Sí Sí Sí Sí Sí Sí Sí Sí Sí Sí Sí Sí
	Cancel		Sí

Tabla 35. Métodos y propiedades ADO soportados por IBM OLE DB Provider para DB2 (continuación)

ADO	Método/Propiedad	Interfaz/Propiedad OLE DB	Soporte de IBM OLE DB
Propiedades de conexión	Attributes adXactCommitRetaining adXactRollbackRetaining	ITransactionLocal	 Sí Sí
	CommandTimeout	ICommandProperties DBPROP_COMMAND_TIMEOUT	Sí
	ConnectionString	(Específico de ADO)	
	ConnectionTimeout	IDBProperties DBPROP_INIT_TIMEOUT	No
	CursorLocation: adUseClient adUseNone adUseServer	 (Utilizar Servicio de cursor de OLE DB) (No utilizado)	 Sí No Sí
	DefaultDataBase	IDBProperties DBPROP_CURRENTCATALOG	No
	IsolationLevel	ITransactionLocal DBPROP_SESS _AUTOCOMMITISOLEVELS	Sí
	Mode adModeRead adModeReadWrite adModeShareDenyNone adModeShareDenyRead adModeShareDenyWrite adModeShareExclusive adModeUnknown adModeWrite	IDBProperties DBPROP_INIT_MODE	 No Sí No No No No No No
	Provider	ISourceRowset::GetSourceRowset	Sí
	State	(Específico de ADO)	
	Version	(Específico de ADO)	
Grupos de conexiones	Errors	IErrorRecords	Sí
	Properties	IDBProperties	Sí
Propiedades de error	Description NativeError Number Source SQLState	IErrorRecords	Sí Sí Sí Sí Sí
	HelpContext HelpFile		No No
Métodos de campo	AppendChunk GetChunk	ISequentialStream	Sí Sí
Propiedades de campo	Actual Size	IAccessor IRowset	Sí
	Attributes DataFormat DefinedSize Name NumericScale Precision Type	IColumnInfo	 Sí Sí Sí Sí Sí Sí
	OriginalValue	IRowsetUpdate	Sí (Servicio de cursor)
	UnderlyingValue	IRowsetRefresh IRowsetResynch	Sí (Servicio de cursor) Sí (Servicio de cursor)

Tabla 35. Métodos y propiedades ADO soportados por IBM OLE DB Provider para DB2 (continuación)

ADO	Método/Propiedad	Interfaz/Propiedad OLE DB	Soporte de IBM OLE DB
	Value	IAccessor IRowset	Sí
Grupo de campos	Properties	IDBProperties IRowsetInfo	Sí
Métodos de parámetro	AppendChunk	ISequentialStream	Sí
	Attributes Direction Name NumericScale Precision Scale Size Type	ICommandWithParameter DBSCHEMA _PROCEDURE_PARAMETERS	 Sí No Sí Sí Sí Sí Sí
	Value	IAccessor ICommand	Sí
Grupo de parámetros	Properties		Sí
Métodos RecordSet	AddNew	IRowsetChange	Sí
	Cancel		Sí
	CancelBatch	IRowsetUpdate::Undo	Sí (Servicio de cursor)
	CancelUpdate		Sí (Servicio de cursor)
	Clone	IRowsetLocate	Sí
	Close	IAccessor IRowset	Sí
	CompareBookmarks		No
	Delete	IRowsetChange	Sí
	GetRows	IAccessor IRowset	Sí
	Move	IRowset IRowsetLocate	Sí
	MoveFirst	IRowset IRowsetLocate	Sí
	MoveNext	IRowset IRowsetLocate	Sí
	MoveLast	IRowsetLocate	Sí
	MovePrevious	IRowsetLocate	Sí
	NextRecordSet	IMultipleResults	Sí
	Open	ICommand IOpenRowset	Sí
	Requery	ICommand IOpenRowset	Sí
	Resync	IRowsetRefresh	Sí (Servicio de cursor)
	Supports	IRowsetInfo	Sí
	Update UpdateBatch	IRowsetChange IRowsetUpdate	Sí Sí (Servicio de cursor)
Propiedades de RecordSet	AbsolutePage	IRowsetLocate IRowsetScroll	Sí Sí[1]
	AbsolutePosition	IRowsetLocate IRowsetScroll	Sí Sí[1]

Tabla 35. Métodos y propiedades ADO soportados por IBM OLE DB Provider para DB2 (continuación)

ADO	Método/Propiedad	Interfaz/Propiedad OLE DB	Soporte de IBM OLE DB
	ActiveConnection	IDBCreateSession IDBInitialize	Sí
	BOF	(Específico de ADO)	
	Bookmark	IAccessor IRowsetLocate	Sí
	CacheSize	cRows in IRowsetLocate IRowset	Sí
	CursorType adOpenDynamic adOpenForwardOnly adOpenKeySet adOpenStatic	ICommandProperties	 No Sí Sí Sí
	EditMode	IRowsetUpdate	Sí (Servicio de cursor)
	EOF	(Específico de ADO)	
	Filter	IRowsetLocate IRowsetView IRowsetUpdate IViewChapter IViewFilter	No
	LockType	ICommandProperties	Sí
	MarshallOption		No
	MaxRecords	ICommandProperties IOpenRowset	Sí
	PageCount	IRowsetScroll	Sí[1]
	PageSize	(Específico de ADO)	
	Sort	(Específico de ADO)	
	Source	(Específico de ADO)	
	State	(Específico de ADO)	
	Status	IRowsetUpdate	Sí (Servicio de cursor)
Notas: 1. Los valores a devolver son aproximaciones. Las filas suprimidas no se pasarán por alto.			
Grupo de RecordSet	Fields	IColumnInfo	Sí
	Properties	IDBProperties IRowsetInfo::GetProperties	Sí

Funciones de tabla de base de datos OLE DB (base de datos de enlace e integración de objetos)

DB2 da soporte a funciones de tabla de base de datos OLE. Para estas funciones, no es necesario crear aplicaciones, salvo la creación de la DLL `CREATE FUNCTION`. DB2 proporciona archivos de ejemplo para funciones de tabla de base de datos OLE en el directorio `sqllib\samples\oledb`. Estos archivos son archivos del Procesador de Línea de Mandatos (CLP). Se pueden crear siguiendo estos pasos:

1. db2 connect to nombre_basedatos
2. db2 -t -v -f *nombre_archivo*.db2
3. db2 terminate

donde `nombre_basedatos` es la base de datos a la que se está conectando y *nombre_archivo* es el nombre del archivo de CLP, con la extensión `.db2`.

Estos mandatos se deben ejecutar en una ventana de mandatos de DB2.

Conceptos relacionados:

- "Funciones de tabla de OLE DB definidas por el usuario" en *Desarrollo de SQL y rutinas externas*

Información relacionada:

- "Ejemplos de funciones de tabla de Object Linking and Embedding Database (OLE DB)" en *Temas de ejemplo*

Automatización del enlace e integración de objetos (OLE) con Visual Basic

Puede implementar funciones definidas por el usuario y procedimientos almacenados de OLE Automation en cualquier lenguaje, pues OLE es independiente del lenguaje. Para hacer esto, se exponen métodos de servidores de OLE Automation, y los métodos se registran en DB2 como UDF (funciones definidas por el usuario). Determinadas versiones de los entornos siguientes de desarrollo de aplicaciones dan soporte al desarrollo de servidores de OLE Automation: Microsoft Visual Basic, Microsoft Visual C++, Microsoft Visual J++, Microsoft FoxPro, Borland Delphi, Powersoft PowerBuilder y Micro Focus COBOL. Además, los objetos de beans Java que están envueltos debidamente para OLE, por ejemplo mediante Microsoft Visual J++, se pueden acceder utilizando OLE Automation.

Consulte la documentación del entorno apropiado de desarrollo de aplicaciones para obtener más información sobre el desarrollo de servidores de OLE Automation.

UDF (funciones definidas por el usuario) y procedimientos almacenados de OLE Automation

Microsoft Visual Basic da soporte a la creación de servidores de OLE Automation. En Visual Basic se crea una nueva clase de objeto añadiendo un módulo de clase al proyecto Visual Basic. Los métodos se crean añadiendo subprocedimientos públicos al módulo de clase. Estos procedimientos públicos se pueden registrar en DB2 como UDF (funciones definidas por el usuario) y procedimientos almacenados de OLE Automation. Para obtener más información sobre la creación de servidores OLE, consulte el manual de Microsoft Visual Basic, *Creating OLE Servers, Microsoft Corporation, 1995*, y los programas de ejemplo de OLE proporcionados por Microsoft Visual Basic.

DB2 proporciona ejemplos incorporados de funciones definidas por el usuario y procedimientos almacenados de OLE Automation en Microsoft Visual Basic, en el directorio `sqllib\samples\ole\msvb`. Para obtener información sobre la creación y ejecución de los ejemplos de funciones definidas por el usuario y procedimientos almacenados de OLE Automation, consulte el archivo `README` situado en `sqllib\samples\ole`.

Conceptos relacionados:

- "Diseño de rutinas de automatización de OLE" en *Desarrollo de SQL y rutinas externas*
- "Rutinas de automatización de OLE en BASIC y C++" en *Desarrollo de SQL y rutinas externas*

Información relacionada:

- "Ejemplos de Object Linking and Embedding (OLE)" en *Temas de ejemplo*

Automatización del enlace e integración de objetos (OLE) con Visual C++

Puede implementar funciones definidas por el usuario y procedimientos almacenados de OLE Automation en cualquier lenguaje, pues OLE es independiente del lenguaje. Para hacer esto, se exponen métodos de servidores de OLE Automation, y los métodos se registran en DB2 como UDF(funciones definidas por el usuario). Determinadas versiones de los entornos siguientes de desarrollo de aplicaciones dan soporte al desarrollo de servidores de OLE Automation: Microsoft Visual Basic, Microsoft Visual C++, Microsoft Visual J++, Microsoft FoxPro, Borland Delphi, Powersoft PowerBuilder y Micro Focus COBOL. Además, los objetos de beans Java que están envueltos debidamente para OLE, por ejemplo mediante Microsoft Visual J++, se pueden acceder utilizando OLE Automation.

Consulte la documentación del entorno apropiado de desarrollo de aplicaciones para obtener más información sobre el desarrollo de servidores de OLE Automation.

UDF (funciones definidas por el usuario) y procedimientos almacenados de OLE Automation

Microsoft Visual C++ da soporte a la creación de servidores de OLE Automation. Los servidores se pueden implementar utilizando Microsoft Foundation Classes y el asistente de la aplicación Microsoft Foundation Class, o bien se pueden implementar como aplicaciones Win32. Los servidores pueden ser una DLL o un EXE. Consulte la documentación de Microsoft Visual C++ y los ejemplos de OLE proporcionados por Microsoft Visual C++ para obtener más información.

DB2 proporciona ejemplos independientes de UDF (funciones definidas por el usuario) y procedimientos almacenados de OLE Automation en Microsoft Visual C++, en el directorio `sqllib\samples\ole\msvc`. Para obtener información sobre la creación y ejecución de los ejemplos de funciones definidas por el usuario y procedimientos almacenados de OLE Automation, consulte el archivo `README` situado en `sqllib\samples\ole`.

Conceptos relacionados:

- "Diseño de rutinas de automatización de OLE" en *Desarrollo de SQL y rutinas externas*
- "Rutinas de automatización de OLE en BASIC y C++" en *Desarrollo de SQL y rutinas externas*

Información relacionada:

- "Ejemplos de Object Linking and Embedding (OLE)" en *Temas de ejemplo*

Creación de aplicaciones ADO con Visual Basic

ActiveX Data Objects (ADO) le permite escribir una aplicación para acceder y manejar datos de un servidor de bases de datos mediante un proveedor de bases

de datos OLE. Las ventajas principales de ADO son una alta velocidad, la facilidad de manejo, una baja actividad de la memoria y una escasa ocupación de espacio de disco.

Los programas ADO de ejemplo creados con Visual Basic se encuentran en el directorio `sqllib\samples\VB\ADO`.

Nota: Para ejecutar los programas de ejemplo ADO para DB2, son recomendables estas versiones (o versiones posteriores) de los componentes siguientes:

1. Visual Basic 6.0 Professional Edition
2. Microsoft Data Access 2.7 SDK (instalado opcionalmente con DB2 Versión 8)
3. Service Pack 5 de Visual Basic, que puede obtener en http://msdn.microsoft.com/vstudio/sp/vs6sp5/vbfixes.asp.
4. El Service Pack más reciente de Visual Studio, que se puede obtener en http://msdn.microsoft.com/vstudio/.

Procedimiento:

Puede utilizar cualquiera de estos dos proveedores compatibles con ODBC:

- Proveedor IBM OLE DB para DB2
- Proveedor Microsoft OLE DB para ODBC

Utilización del proveedor IBM OLE DB para DB2

Los clientes de DB2 Versión 8.2 en sistemas operativos Windows instalarán opcionalmente IBMDADB2, que es el proveedor compatible IBM OLE DB 2.0 para DB2. El proveedor expone interfaces para los usuarios que desean acceder a datos de una base de datos DB2. El proveedor IBM OLE DB para DB2 da soporte a los tipos de aplicación ADO siguientes:

- Microsoft Active Server Pages (ASP)
- Aplicaciones Microsoft Visual Studio C++ y Visual Basic
- Microsoft Visual Interdev

Para conocer detalles sobre estos tipos de aplicaciones, consulte la documentación de ADO.

Para acceder a un servidor DB2 utilizando el proveedor IBM OLE DB para DB2, la aplicación Visual Basic debe especificar la palabra clave PROVIDER en la cadena de conexión de ADO, de esta manera:

```
Dim c1 As ADODB.Connection
Dim c1str As String
c1str = "Provider=ibmdadb2; DSN=aliasDB2; UID=IDusuario; PWD=contraseña"
c1.Open c1str
...
```

donde `aliasDB2` es el alias de la base de datos DB2 que está catalogada en el directorio de bases de datos DB2.

Nota: Si utiliza el proveedor IBM OLE DB para DB2, no es necesario que efectúe el paso de catalogación ODBC para la fuente de datos. Este paso es necesario cuando utiliza el proveedor OLE DB para ODBC.

Utilización del proveedor Microsoft OLE DB para ODBC

Para utilizar ADO con el proveedor Microsoft OLE DB y Visual Basic, necesita establecer una referencia que apunte a la biblioteca de tipos ADO. Siga estos pasos:

1. Seleccione ″Referencias″ en el menú Proyecto.
2. Seleccione la casilla correspondiente a ″Microsoft ActiveX Data Objects <número_versión> Library″
3. Pulse Bien.

donde <número_versión> es la versión actual de la biblioteca ADO.

Una vez efectuados los pasos anteriores, podrá acceder a los objetos, métodos y propiedades de ADO utilizando el Navegador de Objetos VBA y el Editor IDE.

Establezca una conexión:

```
Dim db As Connection
Set db = New Connection
```

Defina los cursores del extremo cliente proporcionados por la biblioteca de cursores local:

```
db.CursorLocation = adUseClient
```

y defina el proveedor para que ADO utilice el controlador Microsoft ODBC.

Acceso a la base de datos ″sample″ con ADO

Un programa Visual Basic completo incluye formularios y otros elementos gráficos, y el usuario necesita visualizar el programa dentro del entorno Visual Basic. Después de conectarse a la base de datos `sample` de DB2 utilizando el proveedor IBM OLE DB o el proveedor Microsoft OLE DB, tal como se describió anteriormente, puede utilizar los siguientes mandatos de Visual Basic como parte de un programa para acceder a la base de datos.

Abra la base de datos `sample` sin especificar un ID de usuario ni una contraseña; es decir, utilice el usuario actual:

```
db.Open "SAMPLE"
```

Cree un conjunto de registros:

```
Set adoPrimaryRS = New Recordset
```

Utilice una sentencia SELECT para llenar el conjunto de registros:

```
adoPrimaryRS.Open "select EMPNO,LASTNAME,FIRSTNME,MIDINIT,EDLEVEL,JOB
from EMPLOYEE Order by EMPNO", db
```

A partir de este punto, el programador puede utilizar los métodos ADO para acceder a los datos, por ejemplo, para pasar al conjunto de registros siguiente:

```
adoPrimaryRS.MoveNext
```

Suprimir el registro actual del conjunto de registros:

```
adoPrimaryRS.Delete
```

Además, el programador puede hacer lo siguiente para acceder a un campo individual:

```
Dim Text1 as String
Text1 = adoPrimaryRS!LASTNAME
```

Conceptos relacionados:

- "Tipos de aplicaciones soportados por IBM OLE DB Provider para DB2" en la página 134
- "Conexiones a fuentes de datos con aplicaciones ADO Visual Basic" en la página 149
- "Soporte de transacciones distribuidas COM+ e IBM OLE DB Provider" en la página 162
- "Restricciones de IBM OLE DB Provider" en la página 142
- "Manipulación de objetos grandes con IBM OLE DB Provider" en la página 135
- "Habilitación automática de servicios OLE DB por parte de IBM OLE DB Provider" en la página 137
- "IBM OLE DB Provider para DB2" en la página 133

Información relacionada:

- "Soporte de IBM OLE DB Provider para propiedades y métodos ADO" en la página 150
- "Soporte de IBM OLE DB para componentes e interfaces de OLE DB" en la página 142
- "Soporte de IBM OLE DB Provider para propiedades de OLE DB" en la página 145
- "Ejemplos de Visual Basic" en *Temas de ejemplos*

Creación de aplicaciones RDO con Visual Basic

Remote Data Objects (RDO) proporciona un modelo de información para acceder a fuentes de datos remotas a través de ODBC. RDO ofrece un conjunto de objetos que hacen más fácil el conectar con una base de datos, ejecutar consultas y procedimientos almacenados, manejar los resultados y confirmar los cambios en el servidor. RDO está pensado específicamente para acceder a fuentes remotas de datos relacionales ODBC, facilita el uso de ODBC sin ser necesario un código de aplicación complejo y es un medio básico para acceder a una base de datos relacional que se expone con un controlador ODBC. RDO implementa una capa de código sobre la API de ODBC (Open Database Connectivity) y el gestor de controladores; esta capa establece conexiones, crea conjuntos y cursores del resultado y ejecuta procedimientos complejos con un mínimo de recursos de la estación de trabajo.

DB2 proporciona programas de ejemplo de RDO para Visual Basic en el directorio `sqllib\samples\VB`.

Procedimiento:

Para utilizar RDO con Microsoft Visual Basic, es necesario establecer una referencia con el proyecto de Visual Basic. Siga estos pasos:

1. Seleccione ″Referencias″ en el menú Proyecto.
2. Seleccione la casilla correspondiente a ″Microsoft Remote Data Object <número_versión>″
3. Pulse Bien.

donde <número_versión> es la versión actual de ADO.

Un programa Visual Basic completo incluye formularios y otros elementos gráficos, y el usuario necesita visualizar el programa dentro del entorno Visual Basic. A continuación siguen mandatos de Visual Basic que forman parte de un programa

DB2 que conecta con la base de datos sample, abre un conjunto de registros que selecciona todas las columnas de la tabla EMPLOYEE y luego muestra los nombres de los empleados en una ventana de mensajes, uno a uno:

```
Dim rdoEn As rdoEngine
Dim rdoEv As rdoEnvironment
Dim rdoCn As rdoConnection
Dim Cnct$
Dim rdoRS As rdoResultset
Dim SQLQueryDB As String
```

Asignar la cadena de conexión:

```
Cnct$ = "DSN=SAMPLE;UID=;PWD=;"
```

Definir el entorno RDO:

```
Set rdoEn = rdoEngine
Set rdoEv = rdoEn.rdoEnvironments(0)
```

Conectar con la base de datos:

```
Set rdoCn = rdoEv.OpenConnection("", , , Cnct$)
```

Asignar la sentencia SELECT para el conjunto de registros:

```
SQLQueryDB = "SELECT * FROM EMPLOYEE"
```

Abrir el conjunto de registros y ejecutar la consulta:

```
Set rdoRS = rdoCn.OpenResultset(SQLQueryDB)
```

Mientras no se alcance el final del conjunto de registros, abrir la ventana de mensajes y mostrar para cada empleado el apellido (LASTNAME) y el nombre (FIRSTNME) a partir de la tabla:

```
While Not rdoRS.EOF
MsgBox rdoRS!LASTNAME & ", " & rdoRS!FIRSTNME
```

Pasar a la fila siguiente del conjunto de registros:

```
rdoRS.MoveNext
Wend
```

Cerrar el programa:

```
rdoRS.Close
rdoCn.Close
rdoEv.Close
```

Información relacionada:

- "Ejemplos de Visual Basic" en *Temas de ejemplos*

Creación de aplicaciones ADO con Visual C++

ActiveX Data Objects (ADO) le permite escribir una aplicación para acceder y manejar datos de un servidor de bases de datos mediante un proveedor de bases de datos OLE. Las ventajas principales de ADO son una alta velocidad, la facilidad de manejo, una baja actividad de la memoria y una escasa ocupación de espacio de disco.

DB2 proporciona programas de ejemplo ADO para Visual C++ en el directorio sqllib\samples\VC.

Procedimiento:

Puede utilizar cualquiera de estos dos proveedores compatibles con ODBC:

- Proveedor IBM OLE DB para DB2
- Proveedor Microsoft OLE DB para ODBC

Utilización del proveedor IBM OLE DB para DB2

Los clientes de DB2 Versión 8.2 en sistemas operativos Windows instalarán opcionalmente IBMDADB2, que es el proveedor compatible IBM OLE DB 2.0 para DB2. El proveedor expone interfaces para los usuarios que desean acceder a datos de una base de datos DB2. El proveedor IBM OLE DB para DB2 da soporte a los tipos de aplicación ADO siguientes:

- Microsoft Active Server Pages (ASP)
- Aplicaciones Microsoft Visual Studio C++ y Visual Basic
- Microsoft Visual Interdev

Para conocer detalles sobre estos tipos de aplicaciones, consulte la documentación de ADO.

Utilización del proveedor Microsoft OLE DB para ODBC

Los programas ADO para DB2 que hacen uso del proveedor Microsoft OLE DB y Visual C++ se pueden compilar del mismo modo que los programas C++ normales, una vez hecho el cambio siguiente.

Para que el programa fuente C++ se ejecute como un programa ADO, debe colocar la siguiente sentencia de importación al comienzo del archivo del programa fuente:

```
#import "C:\program files\common files\system\ado\msado<número_versión>.dll" \
 no_namespace \
 rename( "EOF", "adoEOF")
```

donde `<número_versión>` es el número de versión de la biblioteca ADO.

Cuando se compila el programa, el usuario debe verificar que `msado<número_versión>.dll` está en la vía especificada. Como alternativa, puede añadir `C:\program files\common files\system\ado` a la variable de entorno `LIBPATH` y luego colocar esta sentencia de importación más corta en el archivo fuente:

```
#import <msado<número_versión>.dll> \
 no_namespace \
 rename( "EOF", "adoEOF")
```

Este es el método utilizado en el programa de ejemplo de DB2, `BLOBAccess.dsp`.

Mediante esta sentencia de importación, el programa DB2 podrá acceder a la biblioteca ADO. Ahora puede compilar el programa Visual C++ tal como haría con cualquier otro programa. Si también está utilizando otra interfaz de programación, tal como las API de DB2 o la CLI de DB2, consulte el apartado apropiado para obtener más información sobre la creación del programa.

Conceptos relacionados:

- "Tipos de aplicaciones soportados por IBM OLE DB Provider para DB2" en la página 134
- "Compilación y enlace de aplicaciones C/C++ e IBM OLE DB Provider" en la página 161

- "Conexiones con fuentes de datos en aplicaciones C/C++ mediante IBM OLE DB Provider" en la página 161
- "Restricciones de IBM OLE DB Provider" en la página 142
- "Manipulación de objetos grandes con IBM OLE DB Provider" en la página 135
- "Habilitación automática de servicios OLE DB por parte de IBM OLE DB Provider" en la página 137
- "IBM OLE DB Provider para DB2" en la página 133

Información relacionada:

- "Conversión de datos para establecer datos de tipos DB2 en tipos OLE DB" en la página 140
- "Conversión de datos para establecer datos de tipos OLE DB a tipos DB2" en la página 138
- "Correlaciones de tipos de datos entre DB2 y OLE DB" en la página 137
- "Soporte de IBM OLE DB Provider para propiedades y métodos ADO" en la página 150
- "Soporte de IBM OLE DB para componentes e interfaces de OLE DB" en la página 142
- "Soporte de IBM OLE DB Provider para propiedades de OLE DB" en la página 145
- "Ejemplos de Visual C++" en *Temas de ejemplos*

Aplicaciones C y C++

Las secciones siguientes describen consideraciones sobre aplicaciones C y C++.

Compilación y enlace de aplicaciones C/C++ e IBM OLE DB Provider

Las aplicaciones C/C++ que utilizan la constante CLSID_IBMDADB2 deben incluir el archivo `ibmdadb2.h`, que se encuentra en el directorio `SQLLIB\include`. Estas aplicaciones deben definir `DBINITCONSTANTS` antes de la sentencia include. El ejemplo siguiente muestra la secuencia correcta de directrices de preprocesador C/C++:

```
#define DBINITCONSTANTS
#include "ibmdadb2.h"
```

Conexiones con fuentes de datos en aplicaciones C/C++ mediante IBM OLE DB Provider

Para conectar con una fuente de datos DB2 utilizando IBM OLE DB Provider para DB2 en una aplicación C/C++, utilice una de las dos interfaces principales de OLE DB, `IDBPromptInitialize` o `IDataInitialize`, o bien llame a la API CoCreateInstance de COM. El componente OLE DB Service expone la interfaz `IDataInitialize` y el componente Data Links expone la interfaz `IDBPromptInitialize`.

Conceptos relacionados:

- "Conexiones a fuentes de datos mediante IBM OLE DB Provider" en la página 148

Tareas relacionadas:

- “Creación de aplicaciones ADO con Visual C++” en la página 159

Transacciones distribuidas MTS y COM+

Las secciones siguientes describen consideraciones sobre las transacciones distribuidas MTS y COM+.

Soporte de transacciones distribuidas COM+ e IBM OLE DB Provider

Las aplicaciones OLE DB que se ejecutan en un entorno Microsoft Component Services (COM+) en Windows 2000 o XP pueden utilizar la interfaz `ITransactionJoin` para participar en transacciones distribuidas con varios servidores de bases de datos DB2 Database para Linux, UNIX y Windows, de sistema principal y iSeries, así como otros gestores de recursos que cumplan con las especificaciones COM+.

Requisitos previos:

Para poder utilizar el soporte de transacciones distribuidas COM+ que ofrece IBM OLE DB Provider para DB2, asegúrese de que el servidor cumpla los siguientes requisitos previos.

Nota: Estos requisitos sólo afectan a máquinas basadas en Windows en la que se están instalados los cliente DB2.

- Windows 2000 con Service Pack 3 o posterior
- Windows XP

Conceptos relacionados:

- “Soporte ligeramente agrupado con Microsoft Component Services (COM+)” en la página 164
- “Microsoft Component Services (COM+) como gestor de transacciones” en la página 163

Habilitación del soporte de COM+ en aplicaciones de bases de datos C/C++

Para ejecutar una aplicación C o C++ en la modalidad transaccional COM+, puede crear la instancia de fuente de datos IBMDADB2 mediante la interfaz `DataLink`. También podría utilizar `CoCreateInstance`, obtener un objeto de sesión y utilizar `JoinTransaction`. Consulte la descripción sobre cómo conectar una aplicación C o C++ a una fuente de datos para obtener más información.

Para ejecutar una aplicación ADO en modalidad transaccional COM+, consulte la descripción sobre cómo conectar una aplicación C o C++ a una fuente de datos.

Para utilizar un componente de un paquete COM+ en modalidad transaccional, establezca para la propiedad Transactions del componente uno de los siguientes valores:

- “Required”
- “Required New”
- “Supported”

Para obtener información sobre estos valores, consulte la documentación de COM+.

Conceptos relacionados:

- "Soporte ligeramente agrupado con Microsoft Component Services (COM+)" en la página 164
- "Microsoft Component Services (COM+) como gestor de transacciones" en la página 163

Microsoft Component Services (COM+) como gestor de transacciones

Microsoft Component Services (COM+) como gestor de transacciones

Los sistemas de bases de datos DB2 se puede integrar por completo con Microsoft Component Services (COM+) en Windows 2000 y Windows XP para coordinar la confirmación en dos fases con varios servidores de bases de datos DB2 Database para Linux, UNIX y Windows, zSeries e iSeries, así como con otros gestores de recursos compatibles con las especificaciones COM+.

Requisitos necesarios:

Para utilizar el soporte de transacciones distribuidas COM+, asegúrese de que se cumplen los siguientes requisitos para la máquina Windows en la que está instalado el cliente DB2:

- Windows 2000: Service Pack 3 o posterior

Para aplicaciones CLI de DB2 que utilizan COM+:

- No cambie el valor por omisión del atributo de entorno SQL_ATTR_CONNECTION_POOLING CLI (el valor por omisión es SQL_CP_OFF)
- La instalación del controlador DB2 ODBC en sistemas operativos Windows añadirá automáticamente una nueva palabra clave al registro:

```
HKEY_LOCAL_MACHINE\software\ODBC\odbcinit.ini\IBM DB2 ODBC DRIVER - <Nombre copia DB2>:
  Nombre del valor de la palabra clave: CPTimeout
  Tipo de datos: REG_SZ
  Valor: 60
```

Servidores de bases de datos DB2 soportados:

Los siguientes servidores reciben soporte para la actualización de varios sitios mediante transacciones coordinadas COM+:

- DB2 Enterprise Server Edition (ESE)

  Nota: Las transacciones globales ligeramente agrupadas para COM+ no reciben soporte en entornos de proceso paralelo masivo (MPP). Las transacciones globales ligeramente agrupadas existen cuando cada uno de los procesos de aplicaciones accede a gestores de recursos como si fueran una transacción global separada; sin embargo, estos procesos de aplicaciones están bajo la coordinación del gestor de transacciones. Cada proceso de aplicación tendrá su propia ramificación de transacción dentro de un gestor de recursos. Cuando alguno de los procesos de aplicaciones, gestores de transacciones o gestores de recursos solicita una confirmación o retrotracción, las ramificaciones de la transacción finalizan

simultáneamente. Es responsabilidad de la aplicación asegurar que no se produzca un punto muerto de recurso entre las ramificaciones.

(Las transacciones globales estrechamente agrupadas existen cuando varios procesos de aplicaciones hacen turnos para hacer un trabajo bajo la misma ramificación de transacción en un gestor de recursos. Para el gestor de recursos, los dos procesos de aplicaciones son una sola entidad. El gestor de recursos debe asegurarse de que no se produzca un punto muerto dentro de la ramificación de la transacción.)

- DB2 Universal Database para z/OS
- DB2 Universal Database para iSeries
- DB2 Server para VSE y VM

Conceptos relacionados:

- "Soporte ligeramente agrupado con Microsoft Component Services (COM+)" en la página 164
- "Tiempo de espera de transacciones de Microsoft Component Services (COM+)" en la página 165
- "Soporte de transacciones distribuidas COM+ e IBM OLE DB Provider" en la página 162
- "X/Open distributed transaction processing model" en *Administration Guide: Planning*
- "Agrupación de conexiones ODBC y ADO con Microsoft Component Services (COM+)" en la página 166

Tareas relacionadas:

- "Instalación de un producto de servidor de DB2 Connect (Windows)" en *Guía rápida de iniciación para servidores DB2 Connect*

Información relacionada:

- "Connection attributes (CLI) list" en *Call Level Interface Guide and Reference, Volume 2*
- "Environment attributes (CLI) list" en *Call Level Interface Guide and Reference, Volume 2*
- "SQLSetConnectAttr function (CLI) - Set connection attributes" en *Call Level Interface Guide and Reference, Volume 2*
- "Oferta de productos DB2 Connect" en *Guía del usuario de DB2 Connect*

Soporte ligeramente agrupado con Microsoft Component Services (COM+)

Existen transacciones globales ligeramente agrupadas cuando cada uno de los procesos de aplicación accede a gestores de recursos como si estuviera en una transacción global separada; sin embargo, estos procesos de aplicación están bajo la coordinación del gestor de transacciones. Cada proceso de aplicación tendrá su propia ramificación de transacción dentro de un gestor de recursos. Cuando alguno de los procesos de aplicaciones, gestores de transacciones o gestores de recursos solicita una confirmación o retrotracción, las ramificaciones de la transacción finalizan simultáneamente. Es responsabilidad de la aplicación asegurar que no se produzca un punto muerto de recurso entre las ramificaciones.

DB2 Versión 9 da soporte a transacciones globales ligeramente agrupadas para objetos COM+, sin tiempo de espera de bloqueo ni punto muerto, con las siguientes restricciones:

- El lenguaje de definición de datos (DDL) recibe soporte si se ejecuta en una sola ramificación mientras no hay ninguna otra transacción ligeramente agrupada activa. Si una ramificación ligeramente agrupada se intenta iniciar mientras una ramificación que ejecuta DDL está activa, la ramificación ligeramente agrupada se rechaza. Por el contrario, si hay al menos una transacción ligeramente agrupada activa, cualquier intento de ejecutar DDL en otra ramificación se rechaza.
- Las transacciones globales ligeramente agrupadas no reciben soporte en entornos de proceso paralelo masivo (MPP). En un entorno MPP, cada transacción global se trata de forma aislada, donde no se puede producir ningún punto muerto ni tiempo de espera excedido.
- No puede disponer de varias conexiones en la misma fuente de datos en una única transacción de confirmación en dos fases.
- El proceso de puntos de control y las sentencias de SQL se ejecutan en serie entre varias conexiones.
- Cuando se ha realizado una retrotracción implícita en una conexión, todas las ramificaciones de otras conexiones que están relacionadas con la transacción ligeramente agrupada devolverán el mensaje SQL0998N, con código de razón 225 y subcódigo 4: "Sólo se permiten retrotracciones para esta transacción".

Conceptos relacionados:

- "Microsoft Component Services (COM+) como gestor de transacciones" en la página 163
- "Tiempo de espera de transacciones de Microsoft Component Services (COM+)" en la página 165
- "Soporte de transacciones distribuidas COM+ e IBM OLE DB Provider" en la página 162
- "Agrupación de conexiones ODBC y ADO con Microsoft Component Services (COM+)" en la página 166
- "X/Open distributed transaction processing model" en *Administration Guide: Planning*

Tiempo de espera de transacciones de Microsoft Component Services (COM+)

El tiempo de espera de transacciones puede establecerse por medio de la herramienta siguiente cuando se utiliza COM+: (Microsoft Windows 2000 y XP): Servicios de componentes, situado bajo Herramientas administrativas del Panel de control de Windows.

Si una transacción tarda más en ejecutarse que el valor del tiempo de espera de la transacción (el valor por omisión es 60 segundos), COM+ emitirá de forma asíncrona una terminación anormal para todos los gestores de recursos involucrados y la transacción completa terminará anormalmente.

La terminación anormal se convierte en una solicitud de retrotracción de DB2 en el servidor. La solicitud de retrotracción se serializa en la conexión en servidores que no sean DB2 para z/OS ni DB2 para iSeries, para asegurar la integridad de los datos en el servidor de bases de datos. Cuando el servidor es DB2 para z/OS o DB2 para iSeries, la conexión se debe definir con la opción INTERRUPT_ENABLED en la entrada de catálogo de DCS para que, cuando se

produzca un tiempo de espera excedido, la conexión procedente del servidor DB2 Connect con el servidor z/OS o iSeries se desconecte, forzando una retrotracción en el servidor z/OS o iSeries.

Como resultado:

- Si la conexión está desocupada, la retrotracción se ejecuta de inmediato.
- Si se está procesando una sentencia de SQL de larga ejecución, la solicitud de retrotracción espera hasta que finalice la sentencia de SQL.

Conceptos relacionados:

- "Soporte ligeramente agrupado con Microsoft Component Services (COM+)" en la página 164
- "Microsoft Component Services (COM+) como gestor de transacciones" en la página 163
- "Soporte de transacciones distribuidas COM+ e IBM OLE DB Provider" en la página 162
- "Agrupación de conexiones ODBC y ADO con Microsoft Component Services (COM+)" en la página 166
- "Valores del directorio de DCS" en *Guía del usuario de DB2 Connect*
- "X/Open distributed transaction processing model" en *Administration Guide: Planning*

Agrupación de conexiones ODBC y ADO con Microsoft Component Services (COM+)

La agrupación de conexiones permite a una aplicación utilizar una conexión de una agrupación de conexiones, de modo que la conexión no se tiene que restablecer cada vez que se desea utilizar. Cuando una conexión se crea y se coloca en una agrupación, una aplicación puede reutilizar dicha conexión sin tener que realizar el proceso completo de conexión. La conexión se agrupa cuando la aplicación se desconecta de la fuente de datos y se otorga a una nueva conexión cuyos atributos son los mismos.

Agrupación de conexiones ODBC:

La agrupación de conexiones es una característica de ODBC Driver Manager desde ODBC 2.x. Con la última versión de ODBC Driver Manager (versión 3.5) disponible como parte de la descarga de Microsoft Data Access Components (MDAC), la agrupación de conexiones tiene algunos cambios de configuración y un nuevo comportamiento para las conexiones ODBC de objetos COM+ transaccionales.

ODBC Driver Manager 3.5 necesita que el controlador ODBC registre una nueva palabra clave en el registro para permitir que se active la agrupación de conexiones. La palabra clave es:

```
Nombre clave: SOFTWARE\ODBC\ODBCINST.INI\IBM DB2 ODBC DRIVER - <Nombre copia DB2>
Nombre: CPTimeout
Tipo: REG_SZ
Datos: 60
```

El controlador DB2 ODBC correspondiente al sistema operativo Windows da soporte completo a la agrupación de conexiones; por lo tanto, esta palabra clave se registra.

El valor por omisión, 60, significa que la conexión se agrupará durante 60 segundos antes de que se desconecte.

En un entorno con mucha actividad, es mejor aumentar el valor CPTimeout a un número mayor pare evitar demasiadas conexiones y desconexiones físicas, puesto que estas consumen gran cantidad de recursos del sistema, incluidos recursos de memoria del sistema y de pila de comunicaciones.

Además, para asegurar que se utiliza la misma conexión entre objetos de la misma transacción en una máquina de varios procesadores, debe desactivar el soporte de "agrupación múltiple por procesador". Para ello, copie el siguiente valor de registro en un archivo denominado `odbcpool.reg`, guárdelo como un archivo de texto plano y emita el mandato **odbcpool.reg**. El sistema operativo Windows importará este valor de registro.

```
REGEDIT4

[HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBCINST.INI\ODBC Connection Pooling]
"NumberOfPools"="1"
```

Sin esta palabra clave establecida en 1, COM+ puede agrupar conexiones correspondientes a la misma transacción en distintas agrupaciones, por lo que puede no reutilizar la misma conexión.

Agrupación de conexiones ADO:

Si los objetos MTS o COM+ utilizan ADO para acceder a la base de datos, debe desactivar la agrupación de recursos OLE DB de modo que el proveedor Microsoft OLE DB para ODBC (MSDASQL) no interfiera con la agrupación de conexiones ODBC. Esta característica se inicializa en `OFF` en ADO 2.0, pero se inicializa en `ON` en ADO 2.1. Para desactivar la agrupación de recursos OLE DB, copie las siguientes líneas en un archivo denominado `oledb.reg`, guárdelo como un archivo de texto plano y emita el mandato **oledb.reg**. El sistema operativo Windows importará estos valores de registro.

```
REGEDIT4

[HKEY_CLASSES_ROOT\CLSID\{c8b522cb-5cf3-11ce-ade5-00aa0044773d}]
@="MSDASQL"
"OLEDB_SERVICES"=dword:fffffffc
```

Conceptos relacionados:

- "Soporte ligeramente agrupado con Microsoft Component Services (COM+)" en la página 164
- "Microsoft Component Services (COM+) como gestor de transacciones" en la página 163
- "Tiempo de espera de transacciones de Microsoft Component Services (COM+)" en la página 165
- "Soporte de transacciones distribuidas COM+ e IBM OLE DB Provider" en la página 162
- "X/Open distributed transaction processing model" en *Administration Guide: Planning*

Resolución de problemas de un proyecto de transacción emparejada débilmente de Visual Basic

Una característica del entorno integrado de Visual Basic es que cada vez que se compila una biblioteca enlazada por datos (dll), Visual Basic crea un nuevo

identificador exclusivo globalmente (GUID) para la misma, y lo registra en el registro de Windows. Una vez que se crea la dll del proyecto, se registra con el coordinador de transacciones distribuidas (DTC) utilizando el GUID. Si se reconstruye la dll del proyecto, Visual Basic le asigna un nuevo GUID y también lo registra en el registro de Windows. Por consiguiente el GUID que utiliza el DTC para hacer referencia a la dll del proyecto ahora está desfasado, y en el registro de Windows existen dos entradas distintas para el proyecto.

Procedimiento:

Para evitar este problema, después de crear el proyecto por primera vez debe modificar las propiedades del mismo:

1. En la pestaña Componente, seleccione ″Compatibilidad binaria″ y luego utilice el botón ″...″ para buscar la vía de acceso completa para la dll que acaba de crear.
2. A continuación, pulse el botón ″Aceptar″ y guarde el proyecto de inmediato.

Ahora, cada vez que recompile la dll del proyecto ésta conservará el mismo GUID. Sin embargo, si cambia la interfaz con la dll (por ejemplo, añadiendo o eliminando métodos, o cambiando los parámetros de los métodos existentes), tendrá que utilizar un nuevo GUID.

Si en el registro de Windows ya existen varias entradas de GUID, haga lo siguiente:

1. Busque el nombre del proyecto con el editor del registro.
2. Elimine todas las apariciones del nombre de proyecto que no sean la hallada en la lista de proyectos recientes de Visual Basic.

La información de conexión, incluidos el ID de usuario y la contraseña, debe ser idéntica en el DTC y en la aplicación Visual Basic para que se produzca la transacción emparejada débilmente. Normalmente, para acceder directamente a una dll se utiliza los métodos públicos de la misma. Sin embargo, cuando el DTC está implicado, éste encapsula la dll e intercepta todas las llamadas de entrada para los métodos, así como los resultados de salida de dichos métodos. De este modo, el DTC puede indicar cuándo la actividad de base de datos de uno de estos métodos se debe emparejar débilmente con la actividad de otra base de datos del mismo objeto, o con la actividad de base de datos de un objeto distinto.

Para evitar cualquier problema derivado de esta situación, haga lo siguiente:

1. En las Propiedades de aplicación del DTC, bajo la pestaña ″Identidad″, seleccione ″Este usuario:″.
2. Utilice el botón Examinar para buscar el ID del usuario que va a ejecutar este proyecto.
3. Utilice el mismo ID de usuario y la misma contraseña en la serie de conexión del proyecto.

Si en la serie de conexión de la aplicación Visual Basic utiliza el mismo ID de usuario y la misma contraseña que ha utilizado para iniciar una sesión en el sistema, no es necesario que realice este paso adicional.

Para asegurarse que el proyecto de transacción emparejada débilmente está funcionando correctamente, puede hacer lo siguiente:

1. Observe el archivo de salida creado por el ejecutable y confirme que se están produciendo actualizaciones en la base de datos.
2. Examine un rastreo de la CLI para ENLIST_IN_DTC.

Si falla cualquiera de estas pruebas, la dll no está bien registrada con el DTC y no se están produciendo transacciones emparejadas débilmente,

Tareas relacionadas:

- "Creación de aplicaciones ADO con Visual Basic" en la página 155
- "Creación de transacciones emparejadas débilmente con Visual Basic" en la página 169

Información relacionada:

- "Ejemplos de Visual Basic" en *Temas de ejemplos*

Creación de transacciones emparejadas débilmente con Visual Basic

XA brinda dos manera de que las hebras de aplicaciones de control puedan participar en una sola transacción global XA: emparejamiento fuerte y emparejamiento débil. El proyecto de ejemplo, `LCTransTest`, muestra transacciones XA emparejadas débilmente. Los archivos de ejemplo están ubicados en el directorio `sqllib\samples\VB\MTS`.

Procedimiento:

Para crear y ejecutar el ejemplo de transacciones emparejadas débilmente, realice los pasos siguientes:

1. **Cree el proyecto LCTransTest.vbp**
   a. Abra el proyecto ″LCTransTest.vbp″ realizando una doble pulsación sobre el mismo.
   b. Si recibe el mensaje de error: ″No se puede establecer el componente con versión compatible X:\...\LCTransTest.dll″, pulse ″Bien″ para continuar.
   c. Compile el proyecto. Vaya a ″Archivo″ -> ″Crear LCTransTest.dll″ y luego pulse ″Bien″.
   d. Para resolver el problema de incompatibilidad de versión, pulse con el botón derecho del ratón el proyecto ″LCTransTest (LCTransTest.vbp)″ ubicado en el panel superior derecho. A continuación, seleccione ″Propiedades de LCTransTest″. En la ventana ″LCTransTest - Propiedades del proyecto″. Pulse la pestaña ″Componente″. En la sección ″Compatibilidad de versión″, seleccione ″Compatibilidad binaria″.
   e. Guarde el proyecto. Vaya a ″Archivo″ -> ″Guardar proyecto″.
   f. Cierre el proyecto.
2. **Cree el proyecto Main.vbp**
   a. Abra el proyecto ″Main.vbp″ realizando una doble pulsación sobre el mismo.
   b. Es probable que reciba el mensaje de aviso: ″No se ha podido crear la referencia: X:\...\LCTransTest.dll″, pulse ″Bien″ para continuar.
   c. Vaya a ″Proyecto″ -> ″Referencias″ (en la barra de herramientas).
   d. En la ventana ″Referencias - main.vbp″, asegúrese que el recuadro ″Microsoft ActiveX Data Objects 2.7 Library″ esté seleccionado. Vaya a ″Examinar...″. Busque el proyecto LCTransTest.dll que ha generado en el paso 1 y pulse ″Abrir″ para añadir esta referencia. Pulse ″Bien″ en la ventana ″Referencias - main.vbp″.
   e. Compile el proyecto. Vaya a ″Archivo″ -> ″Crear main.exe″ y luego pulse ″Bien″.

f. Guarde el proyecto. Vaya a ″Archivo″ -> ″Guardar proyecto″.

g. Cierre el proyecto.

3. **Otros valores**

   a. En Windows, vaya a ″Inicio″ -> ″Configuración″ -> ″Panel de control″ -> ″Herramientas administrativas″ -> ″Servicios de componentes″.

   b. En la ventana ″Servicios de componentes″, expanda ″Servicios de componentes″ en el panel de la izquierda hasta que vea ″Aplicaciones COM+″.

   c. Pulse con el botón derecho del ratón ″Aplicaciones COM+″, seleccione ″Nuevo″ -> ″Aplicación″.

   d. En la ventana emergente, ″Asistente para instalación de aplicación COM″, pulse ″Siguiente″.

   e. Seleccione ″Crear una aplicación vacía″.

   f. Entre ″LCTransTest″ como nombre de la nueva aplicación. Mantenga ″Tipo de activación″ como ″Aplicación de servidor″. Pulse ″Siguiente″.

   g. Pulse ″Siguiente″ y luego ″Finalizar″.

   h. Expanda ″LCTransTest″, pulse con el botón derecho del ratón ″Componentes″. Vaya a ″Nuevo″ -> ″Componentes″ -> seleccione ″Importar componentes que ya están registrados″ -> pulse ″LCTransTest.TestClass″ -> ″Siguiente″ -> ″Finalizar″.

   i. Expanda ″Componentes″. Pulse con el botón derecho del ratón ″LCTransTest.TestClass″. Vaya a ″Propiedades″. En la pestaña ″Transacción″, seleccione ″Necesario″ para ″Soporte de transacción″

   j. Reinicie Microsoft Distributed Transaction Coordinator pulsando con el botón derecho del ratón ″Servicios de componentes″ -> ″Sistemas″ -> ″Mi PC″. Seleccione ″Detener MS DTC″. Espere hasta que se detenga y, a continuación, pulse con el botón derecho del ratón ″Mi PC″ -> ″Iniciar MS DTC″ para reiniciar DTC.

4. **Para ejecutar el ejemplo en modalidad de depuración**

   a. Abra LCTransTest.vbp.

   b. En LCTransTest, asegúrese que en ″propiedades del proyecto*″, bajo la pestaña ″Depuración″, está seleccionado ″Esperar a que se creen componentes″. (Lo debe estar por omisión.)

   c. Coloque un punto de interrupción en la línea ″con1.Open connString″ (colocando el cursor sobre dicha línea y pulsando F9).

   d. Pulse F5 (o seleccione ″Iniciar″ en el menú desplegable ″Ejecutar″). Cuando se cargue esta dll, y se ejecute este método, el depurador detendrá la ejecución en ese punto de interrupción.

   e. Abra main.vbp.

   f. En main.vbp, defina los argumentos de la línea de mandatos. En Propiedades del proyecto, bajo la pestaña ″Crear″, en el recuadro ″Argumentos de la línea de mandatos″, escriba lo siguiente:

      ```
      provider=ibmdadb2;dsn=<nombbd>;uid=<IDusuario>;pwd=<ctrsña> <nombarch>
      ```

      donde <nombrebd> es el nombre de una base de datos que tenga y <nombrearchivo> es el nombre de un archivo (incluida la vía de acceso) que contendrá información de salida. Por ejemplo, C:\lctoutput.txt. A continuación, pulse Bien.

   g. En Visual Basic, puede utilizar ″Debug ->Step Into″ <F8> o ″Debug->Step Over″ <despl + F8> para ejecutar el ejecutable de línea de código en línea de código.

h. Cuando llegue a la línea que llama a ″transTest.RunTest″ en el ejecutable principal, e intente evitarla, aflorará la otra ventana de Visual Basic (el proyecto LCTransTest que ha abierto) y se le detendrá en el punto de interrupción que ha colocado ahí. A continuación puede utilizar ″Step Into″ o ″Step Over″ para avanzar por el método RunTest de línea de código en línea de código.

Tareas relacionadas:

- “Creación de aplicaciones ADO con Visual Basic” en la página 155
- “Resolución de problemas de un proyecto de transacción emparejada débilmente de Visual Basic” en la página 167

Información relacionada:

- “Ejemplos de Visual Basic” en *Temas de ejemplos*

Capítulo 7. OLE DB .NET Data Provider

OLE DB .NET Data Provider

OLE DB .NET Data Provider utiliza el Controlador IBM DB2 OLE DB, al que se hace referencia en un objeto `ConnectionString` como `IBMDADB2`. Las palabras clave de la serie de conexión soportadas por OLE DB .NET Data Provider son las misma que las soportadas por IBM OLE DB Provider para DB2. Además, OLE DB .NET Data Provider tiene las mismas restricciones que IBM DB2 OLE DB Provider. Existen restricciones adicionales para OLE DB .NET Data Provider, que se describen en el tema: Restricciones de OLE DB .NET Data Provider.

Para utilizar OLE DB .NET Data Provider, debe tener instalado .NET Framework Versión 1.1 o Versión 2.0.

Para DB2 Universal Database para AS/400 e iSeries, es necesario el siguiente arreglo de programa en el servidor: APAR ii13348.

Estas son las palabras clave de conexión soportadas para OLE DB .NET Data Provider:

Tabla 36. Palabras clave de **ConnectionString** *para OLE DB .NET Data Provider*

Palabra clave	Value	Significado
`PROVIDER`	`IBMDADB2`	Especifica el IBM OLE DB Provider para DB2 (obligatorio)
`DSN` o `Data Source`	alias de base de datos	El alias de la base de datos DB2 tal como está catalogado en el directorio de bases de datos.
`UID`	ID de usuario	El ID de usuario que se utiliza para conectar con el servidor DB2
`PWD`	contraseña	La contraseña correspondiente al ID de usuario utilizado para conectar con el servidor DB2

El ejemplo siguiente crea una conexión `OleDbConnection` para conectar con la base de datos SAMPLE:

```
[Visual Basic .NET]
Dim con As New OleDbConnection("Provider=IBMDADB2;" +
    "Data Source=sample;UID=userid;PWD=password;")
con.Open()

[C#]
OleDbConnection con = new OleDbConnection("Provider=IBMDADB2;" +
                "Data Source=sample;UID=userid;PWD=password;" );
con.Open()
```

Restricciones de OLE DB .NET Data Provider

La tabla siguiente muestra las restricciones de utilización de IBM OLE DB .NET Data Provider:

Tabla 37. Restricciones de IBM OLE DB .NET Data Provider

Clase o función	Descripción de la restricción	Servidores DB2 afectados
Corrientes de caracteres ASCII	No puede utilizar corrientes de caracteres ASCII con `OleDbParameters` si utiliza `DbType.AnsiString` o `DbType.AnsiStringFixedLength`. OLE DB .NET Data Provider emitirá la excepción siguiente: `"Specified cast is not valid" (la conversión de datos especificada no es válida)` **Solución:** Utilice `DbType.Binary` en lugar de utilizar `DbType.AnsiString` o `DbType.AnsiStringFixedLength`.	Todos
`ADORecord`	`ADORecord` no está soportado.	Todos

`ADORecordSet` y `Timestamp`	Tal como está documentado en MSDN, la variable `ADORecordSet` toma el valor 1 segundo. Como consecuencia, las fracciones de segundo se pierden cuando una columna `Timestamp` de DB2 se almacena en un `ADORecordSet`. Similarmente, después de llenar un `DataSet` de un `ADORecordSet`, las columnas `Timestamp` de `DataSet` no tendrán fracciones de segundo.
	
	Solución:
	
	Esta solución solo es efectiva para DB2 Universal Database para Linux, UNIX y Windows, Versión 8.1, FixPak 4 o posterior. Para evitar la pérdida de fracciones de segundo, puede definir la siguiente palabra clave de CLI:
	`MAPTIMESTAMPDESCRIBE = 2`
	
	Esta palabra clave describe `Timestamp` como WCHAR(26). Para definir la palabra clave, ejecute el mandato siguiente desde una ventana de mandatos de DB2:
	`db2 update cli cfg for section common using MAPTIMESTAMPDESCRIBE 2`
		Todos

`Capítulos`	Los capítulos no están soportados.	Todos
Información de claves	OLE DB .NET Data Provider no puede obtener información de claves al abrir un `IDataReader` al mismo tiempo.	DB2 para VM/VSE

Tabla 37. Restricciones de IBM OLE DB .NET Data Provider (continuación)

Clase o función	Descripción de la restricción	Servidores DB2 afectados
Información de claves procedente de procedimientos almacenados	OLE DB .NET Data Provider puede recuperar información de claves referente a un conjunto de resultados devuelto por un procedimiento almacenado solo desde DB2 Database para Linux, UNIX y Windows. Esto es debido a que los servidores DB2 para plataformas que no sean Linux, UNIX y Windows no devuelven información descriptiva ampliada para los conjuntos de resultados abiertos en el procedimiento almacenado. Para recuperar información de claves de un conjunto de resultados devuelto por un procedimiento almacenado en DB2 Database para Linux, UNIX y Windows, es necesario definir la siguiente variable de registro en el servidor DB2: `db2set DB2_APM_PERFORMANCE=8` El definir esta variable de registro de DB2 del servidor hará que los metadatos del conjunto de resultados estén disponibles en el servidor durante más tiempo, lo cual permite que OLE DB recupere satisfactoriamente la información de claves. Sin embargo, dependiendo de la carga de trabajo del servidor, puede que los metadatos no estén disponibles el tiempo suficiente para que OLE DB Provider consulte la información. Por tanto, no es seguro que la información de claves esté siempre disponible para los conjuntos de resultados devueltos por un procedimiento almacenado. Para recuperar cualquier información de claves sobre una sentencia CALL, la aplicación debe ejecutar la sentencia CALL. La invocación de `OleDbDataAdapter.FillSchema()` o `OleDbCommand.ExecuteReader(CommandBehavior.SchemaOnly | CommandBehavior.KeyInfo)` no ejecutará realmente la llamada de procedimiento almacenado. Por tanto, el usuario no obtiene ninguna información de claves para el conjunto de resultados que debe ser devuelto por el procedimiento almacenado.	Todos

Información de claves de sentencias de SQL de proceso por lotes	Cuando se utilizan sentencias de SQL de proceso por lotes que devuelven varios resultados, el método `FillSchema()` intenta recuperar información de esquemas solo para la primera sentencia de SQL contenida en la lista de sentencias de SQL de proceso por lotes. Si esta sentencia no devuelve un conjunto de resultados, no se crea ninguna tabla. Por ejemplo:
	`[C#]`
	`cmd.CommandText = "INSERT INTO ORG(C1) VALUES(1000); SELECT C1 FROM ORG;";`
	`da = new OleDbDataAdapter(cmd);`
	`da.FillSchema(ds, SchemaType.Source);`
	
	No se creará ninguna tabla en el archivo, pues la primera sentencia de SQL del lote es una sentencia INSERT, que no devuelve un conjunto de resultados.
		Todos

Tabla 37. Restricciones de IBM OLE DB .NET Data Provider (continuación)

Clase o función	Descripción de la restricción	Servidores DB2 afectados
`OleDbCommandBuilder`	Las sentencias UPDATE, DELETE e INSERT generadas automáticamente por `OleDbCommandBuilder` son incorrectas si la sentencia SELECT contiene cualquier columna de los tipos de datos siguientes: • CLOB • BLOB • DBCLOB • LONG VARCHAR • LONG VARCHAR FOR BIT DATA • LONG VARGRAPHIC Si se conecta a un servidor DB2 que no sea DB2 Database para Linux, UNIX y Windows, este problema también es debido a columnas con los tipos de datos siguientes: • VARCHAR[1] • VARCHAR FOR BIT DATA[1] • VARGRAPHIC[1] • REAL • FLOAT o DOUBLE • TIMESTAMP **Notas:** 1. Las columnas con estos tipos de datos son aplicables si están definidas para ser valores VARCHAR mayores que 254 bytes, valores VARCHAR FOR BIT DATA mayores que 254 bytes o valores VARGRAPHICs mayores que 127 bytes. Esta condición solo es válida si se conecta a un servidor DB2 que no sea DB2 Database para Linux, UNIX y Windows. `OleDbCommandBuilder` genera sentencias de SQL que utilizan todas las columnas seleccionadas en una comparación de igualdad de la cláusula WHERE, pero los tipos de datos listados anteriormente no se pueden utilizar en una comparación de igualdad. **Nota:** Observe que esta restricción no afectará al método `IDbDataAdapter.Update()` que depende de `OleDbCommandBuilder` para generar automáticamente las sentencias UPDATE, DELETE e INSERT. La operación UPDATE no es efectiva si la sentencia generada contiene cualquiera de los tipos de datos listados anteriormente. **Solución:** Es necesario que elimine explícitamente en la cláusula WHERE de la sentencia de SQL generada todas las columnas que tengan cualquiera de los tipos de datos listados anteriormente. Es recomendable que codifique sus propias sentencias UPDATE, DELETE e INSERT.	Todos
`OleDbCommandBuilder. DeriveParameters`	La distinción entre mayúsculas y minúsculas es importante al utilizar `DeriveParameters()`. El nombre de procedimiento almacenado especificado en `OleDbCommand.CommandText` necesita estar escrito exactamente tal como está almacenado en la tablas de catálogo del sistema DB2. Para ver cómo están almacenados los nombres de procedimiento almacenado, ejecute OpenSchema(OleDbSchemaGuid.Procedures) sin proporcionar la restricción de nombre del procedimiento. Esto devolverá todos los nombres de procedimiento almacenado. Por omisión, DB2 almacena los nombres de procedimiento almacenado en letras mayúsculas, por lo que generalmente deberá especificar el nombre del procedimiento almacenado en mayúsculas.	Todos
`OleDbCommandBuilder. DeriveParameters`	El método `OleDbCommandBuilder.DeriveParameters()` no incluye el parámetro `ReturnValue` en el objeto `OleDbParameterCollection` generado. Por omisión, SqlClient y DB2 .NET Data Provider añaden el parámetro con `ParameterDirection.ReturnValue` al objeto `ParameterCollection` generado.	Todos

Tabla 37. Restricciones de IBM OLE DB .NET Data Provider (continuación)

Clase o función	Descripción de la restricción	Servidores DB2 afectados
`OleDbCommandBuilder. DeriveParameters`	El método `OleDbCommandBuilder.DeriveParameters()` no es efectivo para procedimientos almacenados sobrecargados. Si existen varios procedimientos almacenados denominados "MYPROC", cada uno de los cuales acepta un número diferente de parámetros o tipos diferentes de parámetros, `OleDbCommandBuilder.DeriveParameters()` recupera todos los parámetros de todos los procedimientos almacenados sobrecargados.	Todos
`OleDbCommandBuilder. DeriveParameters`	Si la aplicación no autoriza un procedimiento almacenado con un esquema, `DeriveParameters()` devuelve todos los parámetros para ese nombre de procedimiento. Por tanto, si existen varios esquemas para el mismo nombre de procedimiento, `DeriveParameters()` devuelve todos los parámetros para todos los procedimientos del mismo nombre.	Todos
`OleDbConnection. ChangeDatabase`	El método `OleDbConnection.ChangeDatabase()` no está soportado.	Todos
`OleDbConnection. ConnectionString`	El uso de caracteres no imprimibles, tales como '\b', '\a' o '\O' en la serie de conexión hará que se emita una excepción. Existen restricciones para las palabras clave siguientes: **`Data Source`** La fuente de datos es el nombre de la base de datos, no el servidor. Puede especificar la palabra clave SERVER, pero no será tenida en cuenta por el proveedor IBMDADB2. **`Initial Catalog` y `Connect Timeout`** Estas palabras clave no están soportadas. En general, OLE DB .NET Data Provider no tiene en cuenta ninguna de las palabras clave no reconocidas y no soportadas. Pero si especifica estas palabras clave, se emite la excepción siguiente: `La operación de OLE DB de varios pasos ha generado errores. Examine cada valor de estado de OLE DB, si está disponible. No se ha realizado ninguna acción.` **`ConnectionTimeout`** `ConnectionTimeout` es de solo lectura.	Todos

`OleDbConnection. GetOleDbSchemaTable`	Los valores de restricción distinguen entre mayúsculas y minúsculas, y necesitan estar escritos exactamente tal como están especificados los objetos de base de datos almacenados en las tablas de catálogo del sistema. Por omisión, esos valores se almacenan en mayúsculas.
	
	Por ejemplo, si ha creado una tabla de la manera siguiente:
	`CREATE TABLE abc(c1 SMALLINT)`
	
	DB2 almacenará el nombre de tabla en mayúsculas ("ABC") en el catálogo del sistema. Por tanto, debe utilizar "ABC" como valor de restricción. Por ejemplo:
	`schemaTable = con.GetOleDbSchemaTable(OleDbSchemaGuid.Tables, new object[] { null, null, "ABC", "TABLE" });`
	
	Solución:
	
	Si necesita que se distinga entre mayúsculas y minúsculas o necesita especificar espacios en blanco en las definiciones de datos, encierre las definiciones entre comillas. Por ejemplo:
	`cmd.CommandText = "create table \"Mi Tabla\"(c1 int)";`
	`cmd.ExecuteNonQuery();`
	`tablename = "\"Mi Tabla\"";`
	`schemaTable = con.GetOleDbSchemaTable(OleDbSchemaGuid.Tables, new object[] { null, null, tablename, "TABLE" });`
		Todos

Tabla 37. Restricciones de IBM OLE DB .NET Data Provider (continuación)

Clase o función	Descripción de la restricción	Servidores DB2 afectados
`OleDbDataAdapter` y `DataColumnMapping`	El nombre de la columna fuente distingue entre mayúsculas y minúsculas. Debe estar escrita exactamente tal como está almacenada en los catálogos de DB2, que, por omisión, es en mayúsculas. Por ejemplo: `colMap = new DataColumnMapping("EMPNO", "Employee ID");`	Todos
`OleDbDataReader.GetSchemaTable`	OLE DB .NET Data Provider no puede recuperar información descriptiva ampliada procedente de servidores que no devuelven esa clase de información. Si se conecta a un servidor que no da soporte a información descriptiva ampliada (los servidores afectados), las columnas siguientes de la tabla de metadatos devuelta desde `IDataReader.GetSchemaTable()` no son válidas: • `IsReadOnly` • `IsUnique` • `IsAutoIncrement` • `BaseSchemaName` • `BaseCatalogName`	DB2 para OS/390, versión 7 o inferior DB2 para OS/400 DB2 para VM/VSE
Procedimientos almacenados: sin nombres de columna para conjuntos de resultados	El servidor DB2 para OS/390 versión 6.1 no devuelve nombres de columna para conjuntos de resultados devueltos desde un procedimiento almacenado. OLE DB .NET Data Provider correlaciona estas columnas sin nombre con su número de posición ordinal (por ejemplo, ″1″, ″2″ ″3″). Esto es diferente de la correlación documentada en MSDN: `"Columna1", "Columna2", "Columna3"`.	DB2 para OS/390 versión 6.1

Agrupación de conexiones en aplicaciones de OLE DB .NET Data Provider

OLE DB .NET Data Provider agrupa automáticamente las conexiones utilizando el servicio de agrupación de sesiones de OLE DB. Se pueden utilizar argumentos de la serie de conexión para habilitar o inhabilitar servicios de OLE DB, incluida la agrupación de conexiones. Por ejemplo, la siguiente serie de conexión inhabilita la agrupación de sesiones y el enlistamiento automático de transacciones de OLE DB.

```
Provider=IBMDADB2;OLE DB Services=-4;Data Source=SAMPLE;
```

La tabla siguiente describe los atributos de la serie de conexión de ADO que puede utilizar para definir los servicios de OLE DB:

Tabla 38. Definición de servicios de OLE DB mediante atributos de la serie de conexión de ADO

Servicios habilitados	Valor en la serie de conexión
Todos los servicios (valor por omisión)	″OLE DB Services = -1;″
Todos los servicios excepto la agrupación	″OLE DB Services = -2;″
Todos los servicios excepto la agrupación y el enlistamiento automático	″OLE DB Services = -4;″
Todos los servicios excepto el cursor de cliente	″OLE DB Services = -5;″
Todos los servicios excepto el cursor de cliente y la agrupación	″OLE DB Services = -6;″
Ningún servicio	″OLE DB Services = 0;″

Para obtener más información sobre la agrupación de sesiones o de recursos de OLE DB, así como sobre la inhabilitación del servicio de agrupación mediante la

modificación de los valores por omisión del servicio proveedor de OLE DB, consulte el manual OLE DB Programmer's Reference que se encuentra en la biblioteca de MSDN situada en:

`http://msdn.microsoft.com/library`

Columnas de tiempo en aplicaciones de OLE DB .NET Data Provider

Las secciones siguientes describen implementar columnas de tiempo en aplicaciones de OLE DB .NET Data Provider.

Inserción marcadores de parámetros:

Por ejemplo, suponga que desea insertar un valor de tiempo en una columna de Tiempo:

```
command.CommandText = "insert into mytable(c1) values( ? )";
```

donde la columna c1 es una columna de Tiempo. He aquí dos métodos para vincular un valor de tiempo al marcador de parámetros:

Utilizando `OleDbParameter.OleDbType = OleDbType.DBTime`

Debido a que OleDbType.DBTime se correlaciona con un objeto TimeSpan, debe proporcionar un objeto TimeSpan como valor de parámetro. El valor de parámetro no puede ser un objeto String ni DateTime; debe ser un objeto TimeSpan. Por ejemplo:

```
p1.OleDbType = OleDbType.DBTime;
p1.Value = TimeSpan.Parse("0.11:20:30");
rowsAffected = cmd.ExecuteNonQuery();
```

El formato de TimeSpan es una serie de caracteres de la forma: ″[-]d.hh:mm:ss.ff″, tal como está documentado en el documentación de MSDN.

Utilizando `OleDbParameter.OleDbType = OleDbType.DateTime`

Esta especificación hará que OLE DB .NET Data Provider convierta el valor de parámetro en un objeto DateTime, en lugar de un objeto TimeSpan. Como consecuencia, el valor de parámetro puede ser cualquier serie/objeto válido que se pueda convertir en un objeto DateTime. Esto significa que valores tales como ″11:20:30″ serán efectivos. El valor también puede ser un objeto DateTime. El valor no puede ser un objeto TimeSpan, pues no se puede convertir en un objeto DateTime. TimeSpan no implementa IConvertible.

Por ejemplo:

```
p1.OleDbType = OleDbType.DBTimeStamp;
p1.Value = "11:20:30";
rowsAffected = cmd.ExecuteNonQuery();
```

Recuperación:

Para recuperar una columna de tiempo necesita utilizar el método `IDataRecord.GetValue()` o `OleDbDataReader.GetTimeSpan()`.

Por ejemplo:

```
TimeSpan ts1 = ((OleDbDataReader)reader).GetTimeSpan( 0 );
TimeSpan ts2 = (TimeSpan) reader.GetValue( 0 );
```

Objetos ADORecordset en aplicaciones de OLE DB .NET Data Provider

Las consideraciones siguientes tratan sobre la utilización de objetos ADORecordset.

- La clase adDBTime de tipo ADO se correlaciona con la clase DateTime de .NET Framework. OleDbType.DBTime corresponde a un objeto TimeSpan.
- No puede asignar un objeto TimeSpan al campo Time de un objeto ADORecordset. Esto es debido a que el campo Time del objeto ADORecordset espera recibir un objeto DateTime. Cuando asigna un objeto TimeSpan a un objeto ADORecordset, recibe el mensaje siguiente:

  ```
  La signatura de tipo del método no es compatible con Interop.
  ```

  Solo puede llenar el campo Time con un objeto DateTime, o con un objeto String que se pueda convertir en un objeto DateTime.
- Cuando llena un DataSet con un ADORecordset utilizando OleDbDataAdapter, el campo Time de ADORecordset se convierte en una columna TimeSpan de DataSet.
- Los Recordsets no almacenan claves primarias ni restricciones. Por tanto, no se añade ninguna información de claves al llenar un DataSet a partir de un Recordset utilizando MissingSchemaAction.AddWithKey.

Capítulo 8. ODBC .NET Data Provider

ODBC .NET Data Provider

ODBC .NET Data Provider efectúa llamadas de ODBC a una fuente de datos DB2 utilizando el controlador de CLI de DB2. Por tanto, las palabras de serie de conexión soportadas por ODBC .NET Data Provider son las mismas que las soportadas por el controlador de CLI de DB2. Además, ODBC DB .NET Data Provider tiene las mismas restricciones que el controlador de CLI de DB2. Existen restricciones adicionales para ODBC .NET Data Provider, que se describen en el tema: Restricciones de ODBC .NET Data Provider.

Para utilizar ODBC .NET Data Provider, debe tener instalado .NET Framework Versión 1.1 o Versión 2.0. Para DB2 Universal Database para AS/400 e iSeries, es necesario el siguiente arreglo para el servidor: APAR II13348.

Estas son las palabras clave de conexión soportadas para ODBC .NET Data Provider:

Tabla 39. Palabras clave de **ConnectionString** *para ODBC .NET Data Provider*

Palabra clave	Value	Significado
DSN	alias de base de datos	El alias de la base de datos DB2 tal como está catalogado en el directorio de bases de datos.
UID	ID de usuario	El ID de usuario que se utiliza para conectar con el servidor DB2
PWD	contraseña	La contraseña correspondiente al ID de usuario utilizado para conectar con el servidor DB2

El ejemplo siguiente crea una conexión `OdbcConnection` para conectar con la base de datos SAMPLE:

```
[Visual Basic .NET]
Dim con As New OdbcConnection("DSN=sample;UID=userid;PWD=password;")
con.Open()

[C#]
OdbcConnection con = new OdbcConnection("DSN=sample;UID=userid;PWD=password;");
con.Open()
```

Restricciones de ODBC .NET Data Provider

La tabla siguiente muestra las restricciones de utilización de IBM ODBC .NET Data Provider:

Tabla 40. Restricciones de IBM ODBC .NET Data Provider

Clase o función	Descripción de la restricción	Servidores DB2 afectados
Corrientes de caracteres ASCII	No puede utilizar corrientes de caracteres ASCII con `OdbcParameters` si utiliza `DbType.AnsiString` o `DbType.AnsiStringFixedLength`. ODBC .NET Data Provider emitirá la excepción siguiente: `"Specified cast is not valid" (la conversión de tipo de datos especificada no es válida)` **Solución:** Utilice `DbType.Binary` en lugar de utilizar `DbType.AnsiString` o `DbType.AnsiStringFixedLength`.	Todos

`Command.Prepare`	Antes de ejecutar un mandato (`Command.ExecuteNonQuery` o `Command.ExecuteReader`), debe ejecutar explícitamente `OdbcCommand.Prepare()` si `CommandText` ha cambiado desde la última preparación. Si no ejecuta `OdbcCommand.Prepare()` de nuevo, ODBC .NET Data Provider ejecutará el `CommandText` preparado anteriormente.
	
	Por ejemplo:
	`[C#]`
	`command.CommandText="select CLOB('ABC') from table1";`
	`command.Prepare();`
	`command.ExecuteReader();`
	`command.CommandText="select CLOB('XYZ') from table2";`
	`command.ExecuteReader(); // Esto finaliza ejecutando de nuevo`
	`// la primera sentencia`
		Todos

CommandBehavior. SequentialAccess	Si utiliza `IDataReader.GetChars()` para leer con un lector creado con `CommandBehavior.SequentialAccess`, debe asignar un almacenamiento intermedio que sea lo suficiente grande para contener la columna completa. De lo contrario, recibirá la excepción siguiente:
	`Requested range extends past the end of the array.`
	`at System.Runtime.InteropServices.Marshal.Copy(Int32 source,`
	`Char[] destination, Int32 startIndex, Int32 length)`
	`at System.Data.Odbc.OdbcDataReader.GetChars(Int32 i,`
	`Int64 dataIndex, Char[] buffer, Int32 bufferIndex, Int32 length)`
	`at OleRestrict.TestGetCharsAndBufferSize(IDbConnection con)`
	
	El ejemplo siguiente muestra cómo asignar un almacenamiento intermedio apropiado:
	`CREATE TABLE myTable(c0 int, c1 CLOB(10K))`
	`SELECT c1 FROM myTable;`
	
	`[C#]`
	`cmd.CommandText = "SELECT c1 from myTable";`
	`IDataReader reader = cmd.ExecuteReader(CommandBehavior.SequentialAccess);`
	
	`Int32 iChunkSize = 10;`
	`Int32 iBufferSize = 10;`
	`Int32 iFieldOffset = 0;`
	
	`Char[] buffer = new Char[ iBufferSize ];`
	
	`reader.Read();`
	`reader.GetChars(0, iFieldOffset, buffer, 0, iChunkSize);`
	
	La llamada a `GetChars()` emitirá la excepción siguiente:
	`"Requested range extends past the end of the array"`
	
	Para evitar que `GetChars()` emita la excepción anterior, asigne a `BufferSize` un valor igual al tamaño de la columna, de esta manera:
	`Int32 iBufferSize = 10000;`
	
	Observe que el valor de 10000 para `iBufferSize` corresponde al valor de 10K asignado a la columna CLOB `c1`.
		Todos

Tabla 40. Restricciones de IBM ODBC .NET Data Provider (continuación)

Clase o función	Descripción de la restricción	Servidores DB2 afectados

CommandBehavior. SequentialAccess	ODBC .NET Data Provider emite la excepción siguiente cuando no existen más datos para leer al utilizar `OdbcDataReader.GetChars()`:
	`NO_DATA - no error information available`
	`at System.Data.Odbc.OdbcConnection.HandleError(HandleRef hrHandle, SQL_HANDLE hType, RETCODE retcode)`
	`at System.Data.Odbc.OdbcDataReader.GetData(Int32 i, SQL_C sqlctype, Int32 cb)`
	`at System.Data.Odbc.OdbcDataReader.GetChars(Int32 i, Int64 dataIndex, Char[] buffer, Int32 bufferIndex, Int32 length)`
		Todos

CommandBehavior. SequentialAccess	No podrá utilizar tamaños de bloques de datos grandes, tales como el representado por el valor 5000, al utilizar `OdbcDataReader.GetChars()`. Cuando intente utilizar un tamaño de bloque de datos grande, ODBC .NET Data Provider emitirá esta excepción:
	`Object reference not set to an instance of an object.`
	`at System.Runtime.InteropServices.Marshal.Copy(Int32 source, Char[] destination, Int32 startIndex, Int32 length)`
	`at System.Data.Odbc.OdbcDataReader.GetChars(Int32 i, Int64 dataIndex, Char[] buffer, Int32 bufferIndex, Int32 length)`
	`at OleRestrict.TestGetCharsAndBufferSize(IDbConnection con)`
		Todos

agrupación de conexiones	ODBC .NET Data Provider no controla la agrupación de conexiones. La agrupación de conexiones es gestionada por el Gestor de controladores ODBC. Para obtener más información sobre la agrupación de conexiones, consulte el manual ODBC Programmer's Reference de la biblioteca de MSDN situada en `http://msdn.microsoft.com/library`	Todos
DataColumnMapping	El nombre de la columna fuente debe coincidir exactamente con el nombre utilizado en las tablas de catálogo del sistema, que por omisión se escribe en mayúsculas.	Todos

Columnas decimales	No se pueden utilizar marcadores de parámetros para las columnas decimales.
	
	Generalmente utilizará `OdbcType.Decimal` para un `OdbcParameter` si el `SQLType` de destino es una columna `Decimal`; sin embargo, cuando ODBC .NET Data Provider lee el `OdbcType.Decimal`, vincula el parámetro utilizando el tipo C de `SQL_C_WCHAR` y `SQLType` de `SQL_VARCHAR`, lo cual no es válido.
	
	Por ejemplo:
	`[C#]`
	`cmd.CommandText = "SELECT dec_col FROM MYTABLE WHERE dec_col > ? ";`
	`OdbcParameter p1 = cmd.CreateParameter();`
	`p1.DbType = DbType.Decimal;`
	`p1.Value = 10.0;`
	`cmd.Parameters.Add(p1);`
	`IDataReader rdr = cmd.ExecuteReader();`
	
	Obtendrá una excepción:
	`ERROR [07006] [IBM][CLI Driver][SQLDS/VM] SQL0301N The value of input host variable or parameter number "" cannot be used because of its data type. SQLSTATE=07006`
	
	Solución:
	
	En lugar de utilizar valores de `OdbcParameter`, utilice literales exclusivamente.
		DB2 para VM/VSE

Tabla 40. Restricciones de IBM ODBC .NET Data Provider (continuación)

Clase o función	Descripción de la restricción	Servidores DB2 afectados

Información de claves	El nombre de esquema utilizado para calificar el nombre de tabla (por ejemplo, `MYSCHEMA.MYTABLE`) debe coincidir con el ID de usuario de la conexión. ODBC .NET Data Provider no puede recuperar ninguna información de claves en la que el esquema especificado sea diferente del ID de usuario de la conexión.
	
	Por ejemplo:
	`CREATE TABLE USERID2.TABLE1(c1 INT NOT NULL PRIMARY KEY);`
	
	`[C#]`
	`// Conectar como usuario bob`
	`odbcCon = new OdbcConnection("DSN=sample;UID=bob;PWD=mypassword");`
	
	`OdbcCommand cmd = odbcCon.CreateCommand();`
	
	`// Seleccionar de la tabla con esquema USERID2`
	`cmd.CommandText="SELECT * FROM USERID2.TABLE1";`
	
	`// Error - no se recupera información de claves`
	`da.FillSchema(ds, SchemaType.Source);`
	
	`// Error - SchemaTable no tiene ninguna clave primaria`
	`cmd.ExecuteReader(CommandBehavior.KeyInfo)`
	
	`// Emite una excepción debido a la falta de claves primarias`
	`cbuilder.GetUpdateCommand();`
		Todos

Información de claves	ODBC .NET Data Provider no puede obtener información de claves al abrir un `IDataReader` al mismo tiempo. Cuando ODBC .NET Data Provider abre un `IDataReader`, se abre un cursor en el servidor. Si se solicita información de claves, se invocará `SQLPrimaryKeys()` o `SQLStatistic()` para obtener la información de claves, pero estas funciones de esquema abrirán otro cursor. Debido a que DB2 para VM/VSE no da soporte a la retención de cursor, se cierra el primer cursor. Como consecuencia, las llamadas de `IDataReader.Read()` al `IDataReader` producen la excepción siguiente:
	`System.Data.Odbc.OdbcException: ERROR [HY010] [IBM][CLI Driver]`
	`CLI0125E Function sequence error. SQLSTATE=HY010`
	
	Solución:
	
	Primero deberá recuperar la información de claves y luego recuperar los datos.
	
	Por ejemplo:
	`[C#]`
	`OdbcCommand cmd = odbcCon.CreateCommand();`
	`OdbcDataAdapter da = new OdbcDataAdapter(cmd);`
	
	`cmd.CommandText = "SELECT * FROM MYTABLE";`
	
	`// Utilizar FillSchema para recuperar solo la información de esquema`
	`da.FillSchema(ds, SchemaType.Source);`
	`// Utilizar FillSchema para recuperar solo la información de esquema`
	`da.Fill(ds);`
		DB2 para VM/VSE

Información de claves	Los nombres utilizados en sus sentencias de SQL para especificar objetos de base de datos deben coincidir exactamente con los nombres utilizados para esos objetos en las tablas de catálogo del sistema. Por omisión, los nombres de los objetos de base de datos se guardan en mayúsculas en las tablas de catálogo del sistema. Generalmente, pues, deberá utilizar letras mayúsculas. ODBC .NET Data Provider examina sentencias de SQL para recuperar nombres de objetos de base de datos y los pasa a funciones de esquema, tales como `SQLPrimaryKeys` y `SQLStatistics`, que emiten consultas para estos objetos en las tablas de catálogo del sistema. Los nombres utilizados para especificar los objetos de base de datos deben coincidir exactamente con los nombres correspondientes que están almacenados en las tablas de catálogo del sistema; de lo contrario se devuelve un conjunto de resultados vacío.	DB2 para OS/390 DB2 para OS/400 DB2 para VM/VSE

Tabla 40. Restricciones de IBM ODBC .NET Data Provider (continuación)

Clase o función	Descripción de la restricción	Servidores DB2 afectados
Información de claves para sentencias de SQL de proceso por lotes distintas de SELECT	ODBC .NET Data Provider no puede recuperar ninguna información de claves para una sentencia de proceso por lotes que no comience por "SELECT".	DB2 para OS/390 DB2 para OS/400 DB2 para VM/VSE

Columnas LOB	ODBC .NET Data Provider no da soporte a los tipos de datos LOB. Como consecuencia, cada vez que el servidor DB2 devuelva un SQL_CLOB (-99), SQL_BLOB (-98) o SQL_DBCLOB (-350), ODBC .NET Data Provider emitirá la excepción siguiente:
	`"Unknown SQL type - -98"  (para una columna Blob)`
	`"Unknown SQL type - -99"  (para una columna Clob)`
	`"Unknown SQL type - -350"  (para una columna DbClob)`
	
	Cualquier método que acceda directa o indirectamente a columnas LOB fallará.
	
	Solución:
	
	Asigne el valor 1 a la palabra clave `LongDataCompat` de CLI/ODBC. Esto obligará al controlador CLI de DB2 a correlacionar los tipos de datos siguientes con tipos de datos que DBC .NET Data Provider reconoce, de esta manera:
	• SQL_CLOB con SQL_LONGVARCHAR
	• SQL_BLOB con SQL_LONGVARBINARY
	• SQL_DBCLOB con SQL_WLONGVARCHAR
	
	Para definir la palabra clave `LongDataCompat`, ejecute el mandato de DB2 siguiente desde una ventana de mandatos de DB2 en la máquina cliente:
	`db2 update cli cfg for section common using longdatacompat 1`
	
	Puede también definir esta palabra clave en su aplicación, utilizando la serie de conexión, de esta manera:
	`[C#]`
	`OdbcConnection con =`
	`  new OdbcConnection("DSN=SAMPLE;UID=uid;PWD=mypwd;LONGDATACOMPAT=1;");`
	
	Para obtener una lista de todas las palabras clave de CLI/ODBC, consulte el tema: UID CLI/ODBC configuration keyword en el manual DB2 CLI Guide and Reference.
		Todos

`OdbcCommand.Cancel`	La ejecución de sentencias después de ejecutar `OdbcCommand.Cancel` puede producir la excepción siguiente: `"ERROR [24000] [Microsoft][ODBC Driver Manager] Invalid cursor state"`	Todos

`OdbcCommandBuilder`	`OdbcCommandBuilder` no puede generar mandatos para servidores que no dan soporte a caracteres de escape. Cuando `OdbcCommandBuilder` genera mandatos, primero realiza una llamada a `SQLGetInfo`, solicitando el atributo `SQL_SEARCH_PATTERN_ESCAPE`. Si el servidor no soporta caracteres de escape, se devolverá una serie vacía, lo que hace que ODBC .NET Data Provider emita la siguiente excepción:
	`El índice estaba fuera de los límites de la matriz.`
	`  at System.Data.Odbc.OdbcConnection.get_EscapeChar()`
	`  at System.Data.Odbc.OdbcDataReader.GetTableNameFromCommandText()`
	`  at System.Data.Odbc.OdbcDataReader.BuildMetaDataInfo()`
	`  at System.Data.Odbc.OdbcDataReader.GetSchemaTable()`
	`  at System.Data.Common.CommandBuilder.BuildCache(Boolean closeConnection)`
	`  at System.Data.Odbc.OdbcCommandBuilder.GetUpdateCommand()`
		DB2 para OS/390, sólo servidores DBCS; DB2 para VM/VSE, sólo servidores DBCS

Tabla 40. Restricciones de IBM ODBC .NET Data Provider (continuación)

Clase o función	Descripción de la restricción	Servidores DB2 afectados

OdbcCommandBuilder	La distinción entre letras mayúsculas y minúsculas es importante cuando se utiliza OdbcCommandBuilder para generar automáticamente sentencias UPDATE, DELETE e INSERT. Por omisión, DB2 almacena la información de esquema (como nombres de tabla y nombres de columna) en letras mayúsculas en las tablas de catálogo del sistema, a menos que se hayan creado explícitamente con discriminación de mayúsculas y minúsculas (encerrando entre comillas los nombres de objetos de base de datos en el momento de su creación). Las sentencias de SQL que defina deben utilizar nombres escritos exactamente tal como están almacenados en los catálogos (por omisión, los nombres se almacenan en mayúsculas).
	
	Por ejemplo, si ha creado una tabla utilizando la sentencia siguiente:
	`"db2 create table mytable (c1 int) "`
	
	DB2 almacenará el nombre de tabla ″mytable″ como ″MYTABLE″ en las tablas de catálogo del sistema.
	
	El ejemplo de código siguiente muestra la utilización apropiada de la clase OdbcCommandBuilder:
	`[C#]`
	`OdbcCommand cmd = odbcCon.CreateCommand();`
	`cmd.CommandText  = "SELECT * FROM MYTABLE";`
	`OdbcDataAdapter da = new OdbcDataAdapter(cmd);`
	`OdbcCommandBuilder cb = new OdbcCommandBuilder(da);`
	`OdbcCommand updateCmd = cb.GetUpdateCommand();`
	
	En este ejemplo, si no especifica el nombre de tabla en mayúsculas, obtendrá la excepción siguiente:
	`"La generación de SQL dinámico para UpdateCommand no está soportada para un SelectCommand que no devuelve ninguna información de columnas de clave."`
		Todos

OdbcCommandBuilder	Los mandatos generados por OdbcCommandBuilder son incorrectos cuando la sentencia SELECT contiene los tipos de datos de columna siguientes:
	`REAL`
	`FLOAT o DOUBLE`
	`TIMESTAMP`
	
	Estos tipos de datos no se pueden utilizar en la cláusula WHERE de sentencias SELECT.
		DB2 para OS/390
		DB2 para OS/400
		DB2 para VM/VSE

OdbcCommandBuilder. DeriveParameters	El método DeriveParameters() se correlaciona con SQLProcedureColumns y utiliza la propiedad CommandText para el nombre del procedimiento almacenado. Debido a que CommandText no contiene el nombre del procedimiento almacenado (utilizando la sintaxis completa de la llamada de ODBC), SQLProcedureColumns se invoca con el nombre de procedimiento identificado de acuerdo con la sintaxis de la llamada de ODBC. Por ejemplo: `"{ CALL myProc(?) }"` Esto producirá un conjunto de resultados vacío, cuando no se encuentren columnas para el procedimiento.	Todos
OdbcCommandBuilder. DeriveParameters	Para utilizar DeriveParameters(), especifique el nombre de procedimiento almacenado en CommandText (por ejemplo, cmd.CommandText = "MYPROC"). El nombre del procedimiento almacenado debe estar escrito exactamente tal como está almacenado en las tablas de catálogo del sistema. DeriveParameters() devolverá todos los parámetros para ese nombre de procedimiento que encuentre en las tablas de catálogo del sistema. Recuerde que debe restaurar la sintaxis completa de llamada de ODBC para CommandText antes de ejecutar la sentencia.	Todos
OdbcCommandBuilder. DeriveParameters	El parámetro ReturnValue no se devuelve para ODBC .NET Data Provider.	Todos

Tabla 40. Restricciones de IBM ODBC .NET Data Provider (continuación)

Clase o función	Descripción de la restricción	Servidores DB2 afectados
OdbcCommandBuilder. DeriveParameters	`DeriveParameters()` no da soporte a nombres de procedimiento almacenado totalmente calificados. Por ejemplo, la invocación de `DeriveParameters()` para `CommandText = "MYSCHEMA.MYPROC"` fallará. En este caso, no se devuelve ningún parámetro.	Todos
OdbcCommandBuilder. DeriveParameters	`DeriveParameters()` no es efectivo para procedimientos almacenados sobrecargados. `SQLProcedureColumns` devolverá todos los parámetros para todas las versiones del procedimiento almacenado.	Todos
`OdbcConnection. ChangeDatabase`	El método `OdbcConnection.ChangeDatabase()` no está soportado.	Todos
`OdbcConnection. ConnectionString`	• La palabra clave `Server` no se tiene en cuenta. • La palabra clave `Connect Timeout` no se tiene en cuenta. La CLI de DB2 no da soporte a los tiempos de espera de conexión, por lo que definir esta propiedad no afectará al controlador. • Las palabras clave de agrupación de conexiones no se tiene en cuenta. Específicamente, esto afecta a las palabras clave siguientes: `Pooling`, `Min Pool Size`, `Max Pool Size`, `Connection Lifetime` y `Connection Reset`.	Todos
OdbcDataReader. GetSchemaTable	ODBC .NET Data Provider no puede recuperar información descriptiva ampliada procedente de servidores que no devuelven esa clase de información. Por tanto, si se conecta a un servidor que no da soporte a información descriptiva ampliada (los servidores afectados), las columnas siguientes de la tabla de metadatos devuelta por `IDataReader.GetSchemaTable()` no son válidas: • `IsReadOnly` • `IsUnique` • `IsAutoIncrement` • `BaseSchemaName` • `BaseCatalogName`	DB2 para OS/390, versión 7 o inferior DB2 para OS/400 DB2 para VM/VSE

Procedimientos almacenados	Para llamar a un procedimiento almacenado debe especificar la sintaxis completa de llamada de ODBC.
	
	Por ejemplo, para llamar al procedimiento almacenado `MYPROC` que hace uso de un VARCHAR(10) como parámetro de entrada:
	`[C#]`
	`OdbcCommand cmd = odbcCon.CreateCommand();`
	`cmd.CommandType = CommandType.Text;`
	`cmd.CommandText = "{ CALL MYPROC(?) }"`
	`OdbcParameter p1 = cmd.CreateParameter();`
	`p1.Value = "Joe";`
	`p1.OdbcType = OdbcType.NVarChar;`
	`cmd.Parameters.Add(p1);`
	`cmd.ExecuteNonQuery();`
	
	Nota: Observe que debe utilizar la sintaxis completa de llamada de ODBC aunque esté utilizando `CommandType.StoredProcedure`. Esto está documentado en MSDN, en el tema sobre la propiedad OdbcCommand.CommandText.
		Todos

Procedimientos almacenados: sin nombres de columna para conjuntos de resultados	El servidor DB2 para OS/390 versión 6.1 no devuelve nombres de columna para conjuntos de resultados devueltos desde un procedimiento almacenado. ODBC .NET Data Provider correlaciona estas columnas sin nombre con su número de posición ordinal (por ejemplo, ″1″, ″2″ ″3″). Esto es diferente de la correlación documentada en MSDN: `"Columna1", "Columna2", "Columna3"`.	DB2 para OS/390 versión 6.1
Promoción de índice exclusivo a clave primaria	ODBC .NET Data Provider promociona los índices exclusivos anulables a claves primarias. Esto es contrario a la documentación de MSDN, que establece que los índices exclusivos anulables no se deben promocionar a claves primarias.	Todos

Apéndice A. Información técnica sobre DB2 Database

Visión general de la información técnica de DB2

La información técnica de DB2 está disponible a través de las herramientas y los métodos siguientes:

- Centro de información de DB2
  - Temas
  - Ayuda para herramientas de DB2
  - Programas de ejemplo
  - Guías de aprendizaje
- Manuales de DB2
  - Archivos PDF (descargables)
  - Archivos PDF (del CD en PDF de DB2)
  - manuales en copia impresa
- Ayuda de línea de mandatos
  - Ayuda de mandatos
  - Ayuda de mensajes
- Programas de ejemplo

IBM proporciona periódicamente actualizaciones de la documentación. Si accede a la versión en línea del Centro de información de DB2 en ibm.com, no es necesario que instale las actualizaciones de la documentación porque IBM mantiene actualizada esta versión. Si ha instalado el Centro de información de DB2, es recomendable instalar las actualizaciones de la documentación. Las actualizaciones de la documentación permiten actualizar la información que instaló desde el CD del Centro de información de *DB2* o que descargó de Passport Advantage a medida que información nueva pasa a estar disponible.

Nota: Los temas del Centro de información de DB2 se actualizan con más frecuencia que los manuales en PDF o impresos. Para obtener la información más actualizada, instale las actualizaciones de la documentación cuando estén disponibles, o consulte el Centro de información de DB2 en ibm.com.

Puede acceder a información técnica adicional de DB2 como, por ejemplo, notas técnicas, White papers y Redbooks en línea en el sitio ibm.com. Acceda al sitio de la biblioteca de software de gestión de información de DB2 en http://www.ibm.com/software/data/sw-library/.

Comentarios sobre la documentación

Agradecemos los comentarios sobre la documentación de DB2. Si tiene sugerencias sobre cómo podemos mejorar la documentación de DB2, envíe un correo electrónico a db2docs@ca.ibm.com. El personal encargado de la documentación de DB2 lee todos los comentarios de los usuarios, pero no puede responder directamente a cada uno. Proporcione ejemplos específicos siempre que sea posible de manera que podamos comprender mejor sus problemas. Si realiza comentarios sobre un tema o archivo de ayuda determinado, incluya el título del tema y el URL.

No utilice esta dirección de correo electrónico para contactar con el Servicio al cliente de DB2. Si tiene un problema técnico de DB2 que no está tratado por la documentación, consulte al centro local de servicio técnico de IBM para obtener ayuda.

Conceptos relacionados:

- "Características del Centro de información de DB2" en *Centro de información de DB2 en línea*
- "Archivos de ejemplo" en *Temas de ejemplo*

Tareas relacionadas:

- "Invocación de ayuda de mandatos desde el procesador de línea de mandatos" en *Consulta de mandatos*
- "Invocación de ayuda de mensajes desde el procesador de línea de mandatos" en *Consulta de mandatos*
- "Actualización del Centro de información de DB2 instalado en el sistema o en un servidor de intranet" en la página 195

Información relacionada:

- "Biblioteca técnica de DB2 en formato PDF" en la página 190

Biblioteca técnica de DB2 en formato PDF

Las tablas siguientes describen la biblioteca de DB2 que está disponible en el Centro de publicaciones de IBM en www.ibm.com/shop/publications/order.

Aunque las tablas identifican los manuales en copia impresa disponibles, puede que dichos manuales no estén disponibles en su país o región.

La información de estos manuales es fundamental para todos los usuarios de DB2; esta información le resultará útil tanto si es un programador o un administrador de bases de datos como si trabaja con DB2 Connect u otros productos de DB2.

Tabla 41. Información técnica de DB2

Nombre	Número de documento	Copia impresa disponible
Administration Guide: Implementation	SC10-4221	Sí
Administration Guide: Planning	SC10-4223	Sí
Consulta de las API administrativas	SC11-3192	Sí
Vistas y rutinas administrativas SQL	SC11-3194	No
Call Level Interface Guide and Reference, Volume 1	SC10-4224	Sí
Call Level Interface Guide and Reference, Volume 2	SC10-4225	Sí
Consulta de mandatos	SC11-3179	No
Data Movement Utilities Guide and Reference	SC10-4227	Sí
Data Recovery and High Availability Guide and Reference	SC10-4228	Sí

Tabla 41. Información técnica de DB2 (continuación)

Nombre	Número de documento	Copia impresa disponible
Desarrollo de aplicaciones ADO.NET y OLE DB	SC11-3178	Sí
Desarrollo de aplicaciones de SQL incorporado	SC11-3190	Sí
Desarrollo de SQL y rutinas externas	SC11-3381	No
Desarrollo de aplicaciones Java	SC11-3189	Sí
Desarrollo de aplicaciones Perl y PHP	SC11-3187	No
Iniciación al desarrollo de aplicaciones de bases de datos	SC11-3188	Sí
Iniciación a la instalación y administración de DB2 en Linux y Windows	GC11-3195	Sí
Consulta de mensajes Volumen 1	SC11-3184	No
Consulta de mensajes Volumen 2	SC11-3198	No
Guía de migración	GC11-3196	Sí
Net Search Extender Guía de administración y del usuario **Nota:** El HTML para este documento no se instala desde el CD de documentación HTML.	SH10-9290	Sí
Performance Guide	SC10-4222	Sí
Query Patroller Administration and User's Guide	GC10-4241	Sí
Guía rápida de iniciación para clientes DB2	GC11-3182	No
Guía rápida de iniciación para servidores DB2	GC11-3181	Sí
Spatial Extender y Geodetic Data Management Feature Guía del usuario y manual de consulta	SC11-3229	Sí
Guía de SQL	SC11-3191	Sí
Consulta de SQL, Volumen 1	SC11-3180	Sí
Consulta de SQL, Volumen 2	SC11-3193	Sí
System Monitor Guide and Reference	SC10-4251	Sí
Troubleshooting Guide	GC10-4240	No
Guía de aprendizaje de Visual Explain	SC11-3357	No
Novedades	SC11-3185	Sí
XML Extender Administración y programación	SC11-3230-00	Sí
XML Guide	SC10-4254	Sí
XQuery Reference	SC18-9796	Sí

Tabla 42. Información técnica específica de DB2 Connect

Nombre	Número de documento	Copia impresa disponible
DB2 Connect Guía del usuario	SC11-3197	Sí
Quick Beginnings for DB2 Connect Personal Edition	GC10-4244	Sí
Guía rápida de iniciación para servidores DB2 Connect	GC11-3183	Sí

Tabla 43. Información técnica de integración de la información de WebSphere

Nombre	Número de documento	Copia impresa disponible
WebSphere Information Integration: Administration Guide for Federated Systems	SC19-1020	Sí
WebSphere Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC19-1018	Sí
WebSphere Information Integration: Configuration Guide for Federated Data Sources	SC19-1034	No
WebSphere Information Integration: SQL Replication Guide and Reference	SC19-1030	Sí

Nota: Las Notas de release de DB2 proporcionan información adicional específica para el release del producto y el nivel de fixpack. Para obtener más información, consulte los enlaces relacionados.

Conceptos relacionados:

- "Visión general de la información técnica de DB2" en la página 189
- "Acerca de las notas del release" en *Notas del release*

Tareas relacionadas:

- "Pedido de manuales de DB2 en copia impresa" en la página 192

Pedido de manuales de DB2 en copia impresa

Si necesita manuales de DB2 en copia impresa, puede comprarlos en línea en varios, pero no en todos los países o regiones. Siempre puede hacer pedidos de manuales de DB2 en copia impresa a través del representante local de IBM. Recuerde que algunos manuales en copia software del CD *Documentación en PDF de DB2* no están disponibles en copia impresa. Por ejemplo, ningún volumen de *Consulta de mensajes de DB2* está disponible como manual impreso.

Las versiones impresas de muchos de los manuales de DB2 disponibles en el CD de la Documentación PDF de DB2 se pueden solicitar a IBM por una cantidad. Dependiendo desde dónde realice el pedido, podrá solicitar manuales en línea, desde el Centro de publicaciones de IBM. Si la realización de pedidos en línea no está disponible en su país o región, siempre puede hacer pedidos de manuales de

DB2 en copia impresa al representante local de IBM. Tenga en cuenta que no todos los manuales del CD de la Documentación PDF de DB2 están disponibles en copia impresa.

Nota: La documentación más actualizada y completa de DB2 se mantiene en el Centro de información de DB2 en el sitio http://publib.boulder.ibm.com/infocenter/db2help/.

Procedimiento:

Para hacer pedidos de manuales de DB2 en copia impresa:

- Para averiguar si puede hacer pedidos de manuales de DB2 en copia impresa en línea en su país o región, consulte el Centro de publicaciones de IBM en el sitio http://www.ibm.com/shop/publications/order. Debe seleccionar un país, región o idioma para poder acceder a la información sobre pedidos de publicaciones y, a continuación, seguir las instrucciones sobre pedidos para su localidad.
- Para hacer pedidos de manuales de DB2 en copia impresa a través del representante local de IBM:
  - Localice la información de contacto de su representante local desde uno de los siguientes sitios Web:
    - El directorio de IBM de contactos en todo el mundo en el sitio www.ibm.com/planetwide
    - El sitio Web de publicaciones de IBM en el sitio http://www.ibm.com/shop/publications/order. Tendrá que seleccionar su país, región o idioma para acceder a la página de presentación de las publicaciones apropiadas para su localidad. Desde esta página, siga el enlace "Acerca de este sitio".
  - Cuando llame, indique que desea hacer un pedido de una publicación de DB2.
  - Proporciónele al representante los títulos y los números de documento de los manuales que desee solicitar .

Conceptos relacionados:

- "Visión general de la información técnica de DB2" en la página 189

Información relacionada:

- "Biblioteca técnica de DB2 en formato PDF" en la página 190

Visualización de la ayuda para estados de SQL desde el procesador de línea de mandatos

DB2 devuelve un valor de SQLSTATE para las condiciones que pueden ser el resultado de una sentencia de SQL. La ayuda de SQLSTATE explica los significados de los estados de SQL y los códigos de las clases de estados de SQL.

Procedimiento:

Para invocar la ayuda para estados de SQL, abra el procesador de línea de mandatos y entre:

```
? sqlstate o ? código de clase
```

donde *sqlstate* representa un estado de SQL válido de cinco dígitos y *código de clase* representa los dos primeros dígitos del estado de SQL.

Por ejemplo, `? 08003` visualiza la ayuda para el estado de SQL 08003, y `? 08` visualiza la ayuda para el código de clase 08.

Tareas relacionadas:

- "Invocación de ayuda de mandatos desde el procesador de línea de mandatos" en *Consulta de mandatos*
- "Invocación de ayuda de mensajes desde el procesador de línea de mandatos" en *Consulta de mandatos*

Acceso a diferentes versiones del Centro de información de DB2

Para obtener los temas de DB2 Versión 9, la URL del Centro de información de DB2 es http://publib.boulder.ibm.com/infocenter/db2luw/v9/.

Para obtener los temas de DB2 Versión 8, vaya a la URL del Centro de información Versión 8 en: http://publib.boulder.ibm.com/infocenter/db2luw/v8/.

Tareas relacionadas:

- "Setting up access to DB2 contextual help and documentation" en *Administration Guide: Implementation*

Visualización de temas en el idioma preferido en el Centro de información de DB2

El Centro de información de DB2 intenta visualizar los temas en el idioma especificado en las preferencias del navegador. Si un tema no se ha traducido al idioma preferido, el Centro de información de DB2 visualiza dicho tema en inglés.

Procedimiento:

Para visualizar temas en su idioma preferido en el navegador Internet Explorer:

1. En Internet Explorer, pulse en el botón **Herramientas** —> **Opciones de Internet** —> **Idiomas...**. Se abrirá la ventana Preferencias de idioma.
2. Asegúrese de que su idioma preferido esté especificado como la primera entrada de la lista de idiomas.
   - Para añadir un nuevo idioma a la lista, pulse el botón **Agregar...**.

     Nota: La adición de un idioma no garantiza que el sistema tenga los fonts necesarios para visualizar los temas en el idioma preferido.
   - Para mover un idioma hacia el principio de la lista, seleccione el idioma y pulse el botón **Subir** hasta que el idioma esté en primer lugar en la lista de idiomas.
3. Limpie la antememoria del navegador y, a continuación, renueve la página para visualizar el Centro de información de DB2 en su idioma preferido.

Para visualizar temas en su idioma preferido en un navegador Firefox o Mozilla:

1. Seleccione el botón **Herramientas** —> **Opciones** —> **Idiomas**. Se visualizará el panel Idiomas en la ventana Preferencias.
2. Asegúrese de que su idioma preferido esté especificado como la primera entrada de la lista de idiomas.

- Para añadir un nuevo idioma a la lista, pulse el botón **Añadir...** a fin de seleccionar un idioma en la ventana Añadir idiomas.
- Para mover un idioma hacia el principio de la lista, seleccione el idioma y pulse el botón **Subir** hasta que el idioma esté en primer lugar en la lista de idiomas.

3. Limpie la antememoria del navegador y, a continuación, renueve la página para visualizar el Centro de información de DB2 en su idioma preferido.

En algunas combinaciones de navegador y sistema operativo, puede que también tenga que cambiar los valores regionales del sistema operativo al entorno local y al idioma de su elección.

Conceptos relacionados:

- "Visión general de la información técnica de DB2" en la página 189

Actualización del Centro de información de DB2 instalado en el sistema o en un servidor de intranet

Si ha instalado localmente un Centro de información de DB2, puede descargar temas actualizados. El valor de 'Última actualización' que se encuentra la final de la mayoría de los temas indica el nivel actual de ese tema.

Para determinar si hay una actualización disponible para todo el Centro de información de DB2, busque el valor de 'Última actualización' en la página Web inicial del Centro de información. Compare el valor contenido en la página Web inicial instalada localmente con la fecha de la actualización descargable más reciente contenida en http://www.ibm.com/software/data/db2/udb/support/icupdate.html. Puede actualizar el Centro de información instalado localmente si está disponible una actualización descargable más reciente.

Para actualizar el Centro de información de DB2 instalado localmente debe:

1. Detener el Centro de información de DB2 en el sistema, y reiniciar el Centro de información en modalidad autónoma. La ejecución del Centro de información en modalidad autónoma impide que otros usuarios de la red accedan al Centro de información, y permite descargar y aplicar actualizaciones.
2. Utilice la función Actualizar para determinar si hay paquetes de actualización disponibles en IBM.

   Nota: También existen actualizaciones en CD. Para conocer detalles sobre cómo configurar el Centro de información para instalar actualizaciones desde CD, vea los enlaces correspondientes.

   Si hay paquetes de actualización disponibles, utilice la función Actualizar para descargar los paquetes. (La función actualizar sólo está disponible en modalidad autónoma.)
3. Detenga el Centro de información autónomo y reinicie el servicio Centro de información de DB2 en el sistema.

Procedimiento:

Para actualizar el Centro de información de DB2 instalado en el sistema o en el servidor de intranet:

1. Detenga el servicio Centro de información de DB2.

- En Windows, pulse en **Inicio → Panel de control → Herramientas administrativas → Servicios**. Después, pulse con el botón derecho del ratón en el servicio **Centro de información de DB2** y seleccione **Detener**.
- En Linux, especifique el mandato siguiente:

  ```
  /etc/init.d/db2icdv9 stop
  ```

2. Inicie el Centro de información en modalidad autónoma.
   - En Windows:
     a. Abra una ventana de mandatos.
     b. Navegue a la vía de acceso en la que está instalado el Centro de información. Por omisión, el Centro de información de DB2 está instalado en el directorio `C:\Archivos de programa\IBM\Centro de información de DB2\Versión 9`.
     c. Ejecute el archivo `help_start.bat` utilizando la vía de acceso completamente calificada para el Centro de información de DB2:

        ```
        <directorio de Centro de información de DB2>\doc\bin\help_start.bat
        ```
   - En Linux:
     a. Vaya hasta la vía de acceso en la que está instalado el Centro de información. Por omisión, el Centro de información de DB2 está instalado en el directorio `/opt/ibm/db2ic/V9`.
     b. Ejecute el script `help_start` utilizando la vía de acceso totalmente calificada del Centro de información de DB2:

        ```
        <directorio del Centro de información de DB2>/doc/bin/help_start
        ```

   Se inicia el navegador Web por omisión de los sistemas para visualizar el Centro de información autónomo.
3. Pulse en el botón Actualizar ( ). En la derecha del panel del Centro de información, pulse en **Buscar actualizaciones**. Se visualiza una lista de actualizaciones para la documentación existente.
4. Para iniciar el proceso de descarga, compruebe las selecciones que desea descargar, después pulse en **Instalar actualizaciones**.
5. Cuando finalice el proceso de descarga e instalación, pulse en **Finalizar**.
6. Detenga el Centro de información autónomo.
   - En Windows, ejecute el archivo `help_end.bat` utilizando la vía de acceso completamente calificada para el Centro de información de DB2:

     ```
     <directorio de Centro de información de DB2>\doc\bin\help_end.bat
     ```

     Nota: El archivo help_end de proceso por lotes contiene los mandatos necesarios para concluir sin peligro los procesos que se iniciaron mediante el archivo help_start de proceso por lotes. No utilice `Control-C` ni ningún otro método para concluir `help_start.bat`.
   - En Linux, ejecute el script `help_end` utilizando la vía de acceso totalmente calificada del Centro de información de DB2:

     ```
     <directorio del Centro de información de DB2>/doc/bin/help_end
     ```

     Nota: El script help_end contiene los mandatos necesarios para concluir sin peligro los procesos que se iniciaron mediante el script help_start. No utilice ningún otro método para concluir el script `help_start`.
7. Reinicie el servicio Centro de información de DB2.
   - En Windows, pulse en **Inicio → Panel de control → Herramientas administrativas → Servicios**. Después, pulse con el botón derecho del ratón en el servicio **Centro de información de DB2** y seleccione **Inicio**.

- En Linux, especifique el mandato siguiente:
  ```
  /etc/init.d/db2icdv9 start
  ```

El Centro de información de DB2 actualizado visualiza los temas nuevos y actualizados.

Conceptos relacionados:

- "Opciones de instalación del Centro de información de DB2" en *Guía rápida de iniciación para servidores DB2*

Tareas relacionadas:

- "Instalación del Centro de información de DB2 utilizando el asistente de instalación de DB2 (Linux)" en *Guía rápida de iniciación para servidores DB2*
- "Instalación del Centro de información de DB2 mediante el Asistente de instalación de DB2 (Windows)" en *Guía rápida de iniciación para servidores DB2*

Guías de aprendizaje de DB2

Las guías de aprendizaje de DB2 le ayudan a conocer diversos aspectos de productos DB2. Se proporcionan instrucciones paso a paso a través de lecciones.

Antes de comenzar:

Puede ver la versión XHTML de la guía de aprendizaje desde el Centro de información en el sitio http://publib.boulder.ibm.com/infocenter/db2help/.

Algunas lecciones utilizan datos o código de ejemplo. Consulte la guía de aprendizaje para obtener una descripción de los prerrequisitos para las tareas específicas.

Guías de aprendizaje de DB2:

Para ver la guía de aprendizaje, pulse el título.

Almacén de datos XML nativos
: Configure una base de datos DB2 para almacenar datos XML y realizar operaciones básicas con el almacén de datos XML nativos.

Guía de aprendizaje de Visual Explain
: Analizar, optimizar y ajustar sentencias de SQL para obtener un mejor rendimiento al utilizar Visual Explain.

Conceptos relacionados:

- "Visual Explain overview" en *Administration Guide: Implementation*

Información de resolución de problemas de DB2

Existe una gran variedad de información para la resolución y determinación de problemas para ayudarle en la utilización de productos DB2.

Documentación de DB2
: Puede encontrar información sobre la resolución de problemas en la publicación DB2 Troubleshooting Guide o en la sección Soporte y resolución de problemas del Centro de información de DB2. En ellas encontrará información sobre cómo aislar e identificar problemas

utilizando herramientas y programas de utilidad de diagnóstico de DB2, soluciones a algunos de los problemas más habituales y otros consejos sobre cómo solucionar problemas que podría encontrar en los productos DB2.

Sitio Web de soporte técnico de DB2
Consulte el sitio Web de soporte técnico de DB2 si tiene problemas y desea obtener ayuda para encontrar las causas y soluciones posibles. El sitio de soporte técnico tiene enlaces a las publicaciones más recientes de DB2, notas técnicas, Informes autorizados de análisis del programa (APAR o arreglos de defectos), fix packs y otros recursos. Puede buscar en esta base de conocimiento para encontrar posibles soluciones a los problemas.

Acceda al sitio Web de soporte técnico de DB2 en la dirección http://www.ibm.com/software/data/db2/udb/support.html

Conceptos relacionados:

- "Introduction to problem determination" en *Troubleshooting Guide*
- "Visión general de la información técnica de DB2" en la página 189

Términos y condiciones

Los permisos para utilizar estas publicaciones se otorgan sujetos a los siguientes términos y condiciones.

Uso personal: Puede reproducir estas publicaciones para su uso personal, no comercial, siempre y cuando se mantengan los avisos sobre la propiedad. No puede distribuir, visualizar o realizar trabajos derivados de estas publicaciones, o de partes de las mismas, sin el consentimiento expreso de IBM.

Uso comercial: Puede reproducir, distribuir y visualizar estas publicaciones únicamente dentro de su empresa, siempre y cuando se mantengan todos los avisos sobre la propiedad. No puede realizar trabajos derivativos de estas publicaciones, ni reproducirlas, distribuirlas o visualizarlas, ni de partes de las mismas fuera de su empresa, sin el consentimiento expreso de IBM.

Excepto lo expresamente concedido en este permiso, no se conceden otros permisos, licencias ni derechos, explícitos o implícitos, sobre las publicaciones ni sobre ninguna información, datos, software u otra propiedad intelectual contenida en el mismo.

IBM se reserva el derecho de retirar los permisos aquí concedidos cuando, a su discreción, el uso de las publicaciones sea en detrimento de su interés o cuando, según determine IBM, las instrucciones anteriores no se cumplan correctamente.

No puede descargar, exportar ni volver a exportar esta información excepto en el caso de cumplimiento total con todas las leyes y regulaciones vigentes, incluyendo todas las leyes y regulaciones sobre exportación de los Estados Unidos.

IBM NO GARANTIZA EL CONTENIDO DE ESTAS PUBLICACIONES. LAS PUBLICACIONES SE PROPORCIONAN ″TAL CUAL″ Y SIN GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUYENDO PERO SIN LIMITARSE A LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN, NO VULNERACIÓN E IDONEIDAD PARA UN FIN DETERMINADO.

Apéndice B. 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. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implicar que sólo se pueda utilizar dicho producto, programa o servicio de IBM. En su lugar se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no vulnere 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
EE.UU.

Para realizar consultas sobre licencias referentes a información de doble byte (DBCS), puede ponerse en contacto con el Departamento de Propiedad Intelectual de IBM de su país/región o escribir a:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokio 106, Japón

El párrafo siguiente no es aplicable al Reino Unido ni a ningún país/región en donde 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, COMERCIALIZACIÓN 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 cambios en los productos y 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 estos sitios Web no forma parte de la información del presente producto IBM y el usuario es responsable de la utilización de dichos sitios.

IBM puede utilizar o distribuir cualquier información que se le facilite de la manera que 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 éste) y (ii) el uso mutuo de la información intercambiada, deben ponerse en contacto con:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADÁ

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 documento y todo el material bajo licencia asociado a él, los proporciona IBM según los términos del Acuerdo de Cliente de IBM, el Acuerdo Internacional de Programas Bajo Licencia de IBM o cualquier acuerdo equivalente entre el usuario e IBM.

Los datos de rendimiento contenidos en este documento se obtuvieron en un entorno controlado. Por lo tanto, los resultados obtenidos en otros entornos operativos pueden variar significativamente. Algunas mediciones pueden haberse realizado 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 ninguna otra afirmación referente a productos que no son de IBM. Las preguntas sobre las prestaciones de productos que no son de 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.

Este manual 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 nombres 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 fortuita.

LICENCIA DE COPYRIGHT:

Este manual puede contener programas de aplicaciones de ejemplo escritos en lenguaje fuente, que muestran técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo como desee, sin pago alguno a IBM con la intención de desarrollar, utilizar, comercializar o distribuir programas de aplicaciones 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 lo tanto, IBM no puede asegurar ni implicar la fiabilidad, utilidad o función de estos programas.

Cada copia o parte 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 proceden de programas de ejemplo de IBM Corp. © Copyright IBM Corp. *_entre el o los años_*. Reservados todos los derechos.

Marcas registradas

Los nombres de empresas, productos o servicios identificados en la biblioteca de documentación de DB2 Versión 9 pueden ser marcas registradas o marcas de servicios de International Business Machines Corporation o de otras empresas. La información sobre marcas registradas de IBM Corporation en los Estados Unidos y/o en otros países está ubicada en http://www.ibm.com/legal/copytrade.shtml.

Los términos siguientes son marcas registradas de otras empresas y se han utilizado como mínimo en uno de los documentos de la biblioteca de documentación de DB2:

Microsoft, Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en los Estados Unidos y/o en otros países.

Intel, Itanium, Pentium y Xeon son marcas registradas de Intel Corporation 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.

Linux es una marca registrada de Linus Torvalds en los Estados Unidos y/o en otros países.

Otros nombres de empresas, productos o servicios, pueden ser marcas registradas o marcas de servicio de otras empresas.

Índice

Caracteres Especiales

A

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

Cómo ponerse en contacto con IBM

Para ponerse en contacto con IBM en su país o región, consulte IBM Directory of Worldwide Contacts en el sitio http://www.ibm.com/planetwide

Para obtener más información sobre productos DB2, vaya a http://www.ibm.com/software/data/db2/.

SC11-3178-00

Spine information:

IBM DB2 **DB2 Versión 9** **Desarrollo de aplicaciones ADO.NET y OLE DB**