IBM DB2 10.1
para Linux, UNIX y Windows

Desarrollo de aplicaciones Perl, PHP, Python y Ruby on Rails

SC11-8066-00

IBM DB2 10.1
para Linux, UNIX y Windows

Desarrollo de aplicaciones Perl, PHP, Python y Ruby on Rails

SC11-8066-00

Nota

Antes de utilizar esta información y el producto al que da soporte, lea la información general contenida en el apartado Apéndice B, "Avisos", en la página 85.

Nota de edición

Este manual es la traducción del original en inglés *IBM DB2 10.1 for Linux, UNIX, and Windows Developing Perl, PHP, Python, and Ruby on Rails Applications* (SC27-3876-00).

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

- Para solicitar publicaciones en línea, vaya a IBM Publications Center en http://www.ibm.com/shop/publications/order
- Para encontrar al representante local de IBM que le corresponde, vaya a la sección Worlwide Contacts de IBM Directory en http://www.ibm.com/planetwide/

Para realizar pedidos de publicaciones de DB2 desde DB2 Marketing and Sales, en los EE.UU. o en Canadá, llame al 1-800-IBM-4YOU (426-4968).

Contenido

Capítulo 1. Desarrollo de aplicaciones Perl

Consideraciones sobre la programación en Perl

Perl Database Interface (DBI) es una interfaz de programación de aplicaciones estándar que proporciona acceso de base de datos a las aplicaciones cliente escritas en Perl. La DBI de Perl define un conjunto de funciones, variables y convenciones que ofrecen una interfaz de base de datos independiente de la plataforma.

Puede utilizar IBM® DB2 Database Driver para la DBI de Perl (el controlador DBD::DB2) disponible en http://www.ibm.com/software/data/db2/perl junto con el módulo DBI de Perl disponible en http://www.perl.com para crear aplicaciones de DB2 que usen Perl.

Dado que Perl es un lenguaje interpretado y que el módulo DBI de Perl utiliza SQL dinámico, Perl constituye el lenguaje idóneo para crear y revisar con rapidez los prototipos de aplicaciones de DB2. El módulo DBI de Perl utiliza una interfaz bastante similar a las interfaces CLI y JDBC, lo que le permitirá trasladar fácilmente sus prototipos Perl a CLI y JDBC.

La mayoría de proveedores de bases de datos proporcionan un controlador de base de datos para el módulo DBI de Perl, lo cual significa que también se puede utilizar Perl para crear aplicaciones que accedan a datos de muchos servidores de bases de datos distintos. Por ejemplo, puede escribir una aplicación Perl de DB2 que se conecte a una base de datos Oracle e inserte los datos en una base de datos DB2 utilizando el controlador de base de datos DBD::DB2.

Para obtener información sobre servidores de bases de datos soportados, instrucciones de instalación y requisitos previos, consulte http://www.ibm.com/software/data/db2/perl

Descargas y recursos relacionados de Perl

Hay disponibles varios recursos para ayudarle a desarrollar aplicaciones Perl que acceden a servidores de datos de IBM.

Tabla 1. Descargas y recursos relacionados de Perl.

La primera columna identifica las áreas de descarga. La segunda columna identifica los recursos relacionados.

Descargas	**Recursos relacionados**
Módulo Perl Database Interface (DBI)	http://www.perl.com
Controlador DBD::DB2	http://www.ibm.com/software/data/db2/perl
IBM Data Server Driver Package (controlador DS)	http://www.ibm.com/software/data/support/data-server-clients/index.html
Documentación de la API de DBI	http://search.cpan.org/~timb/DBI/DBI.pm
Nota técnica de DB2 Perl Database Interface for para Linux, UNIX y Windows, que incluye el archivo léame y las instrucciones de instalación	http://www.ibm.com/software/data/db2/perl

Tabla 1. Descargas y recursos relacionados de Perl (continuación).

La primera columna identifica las áreas de descarga. La segunda columna identifica los recursos relacionados.

Descargas	Recursos relacionados
Sistema de notificación de errores de controlador de Perl	http://rt.cpan.org/
Notificación de errores al equipo de código abierto de IBM	opendev@us.ibm.com

Conexiones de bases de datos en Perl

El controlador DBD::DB2 proporciona soporte para las funciones de conexión de base de datos estándar definidas por la API de DBI.

Para permitir que Perl cargue el módulo DBI, debe incluir la línea `use DBI;` en la aplicación:

El módulo DBI carga automáticamente el controlador DBD::DB2 cuando se crea un descriptor de contexto de base de datos utilizando la sentencia **DBI->connect** con la sintaxis indicada:

```
my $dbhandle = DBI->connect('dbi:DB2:dsn', $userID, $password);
```

donde:

$dbhandle
: representa el descriptor de contexto de base de datos devuelto por la sentencia de conexión

dsn

: para las conexiones locales, representa un alias de DB2 catalogado en su directorio de bases de datos DB2

: para las conexiones remotas, representa una serie de conexión completa que incluye el nombre del sistema principal, el número de puerto, el protocolo, el ID de usuario y la contraseña para conectar con el sistema principal remoto

$userID
: representa el ID de usuario utilizado para conectar con la base de datos

$password
: representa la contraseña para el ID de usuario utilizado para conectar con la base de datos

Para obtener más información sobre la API de DBI, consulte http://search.cpan.org/~timb/DBI/DBI.pmhttp://search.cpan.org/~timb/DBI/DBI.pm.

Ejemplo

Ejemplo 1: Conectar con una base de datos en el sistema principal local (el cliente y el servidor se encuentran en la misma estación de trabajo)

```
use DBI;

$DATABASE = 'dbname';
$USERID = 'username';
$PASSWORD = 'password';
```

```
my $dbh = DBI->connect("dbi:DB2:$DATABASE", $USERID, $PASSWORD, {PrintError => 0})
or die "Couldn't connect to database: " . DBI->errstr;

$dbh->disconnect;
```

Ejemplo 2: Conectar con una base de datos en el sistema principal remoto (el cliente y el servidor se encuentran en distintas estaciones de trabajo)

```
use DBI;

$DSN="DATABASE=sample; HOSTNAME=host; PORT=60000; PROTOCOL=TCPIP; UID=username;
PWD=password";

my $dbh = DBI->connect("dbi:DB2:$DSN", $USERID, $PASSWORD, {PrintError => 0})
or die "Couldn't connect to database: " . DBI->errstr;

$dbh->disconnect;
```

Captación de resultados en Perl

El módulo DBI de Perl ofrece métodos para la conexión con una base de datos, la preparación y emisión de sentencias de SQL y la captación de filas de resultados.

Acerca de esta tarea

Este procedimiento capta los resultados de una consulta de SQL.

Restricciones

Puesto que el módulo DBI de Perl sólo soporta SQL dinámico, no se pueden utilizar variables del lenguaje principal en las aplicaciones DB2 en Perl.

Procedimiento

Para captar los resultados:

1. Cree un descriptor de contexto de base de datos estableciendo conexión con la base de datos con la sentencia `DBI->connect`.
2. Cree un descriptor de contexto de sentencia a partir del descriptor de contexto de base de datos. Por ejemplo, puede devolver el descriptor de contexto de sentencia $sth del descriptor de contexto de base de datos llamando al método prepare y pasando una sentencia de SQL como argumento de serie, tal como se muestra en el ejemplo de sentencia de Perl:

   ```
   my $sth = $dbhandle->prepare(
      'SELECT firstnme, lastname
         FROM employee '
      );
   ```

3. Emita la sentencia de SQL llamando al método execute en el descriptor de contexto de sentencia. Una llamada satisfactoria al método execute asocia un conjunto de resultados al descriptor de contexto de sentencia. Por ejemplo, puede ejecutar la sentencia PREPARED en la sentencia de Perl anterior utilizando el ejemplo indicado:

   ```
   #Nota: $rc representa el código de retorno de la llamada a execute
   my $rc = $sth->execute();
   ```

4. Capte una fila del conjunto de resultados asociado al descriptor de contexto de sentencia mediante una llamada al método fetchrow. El DBI de Perl devuelve una fila en forma de matriz con un valor por columna. Por ejemplo, puede devolver todas las filas del descriptor de contexto de sentencia del ejemplo anterior utilizando la sentencia de Perl indicada:

```
while (($firstnme, $lastname) = $sth->fetchrow()) {
   print "$firstnme $lastname\n";
}
```

Ejemplo

El ejemplo muestra cómo conectar con una base de datos y emitir una sentencia SELECT desde una aplicación escrita en Perl.

```
#!/usr/bin/perl
use DBI;

my $database='dbi:DB2:sample';
my $user='';
my $password='';

my $dbh = DBI->connect($database, $user, $password)
   or die "Can't connect to $database: $DBI::errstr";

my $sth = $dbh->prepare(
   q{ SELECT firstnme, lastname
      FROM employee }
   )
   or die "Can't prepare statement: $DBI::errstr";

my $rc = $sth->execute
   or die "Can't execute statement: $DBI::errstr";

print "Query will return $sth->{NUM_OF_FIELDS} fields.\n\n";
print "$sth->{NAME}->[0]: $sth->{NAME}->[1]\n";

while (($firstnme, $lastname) = $sth->fetchrow()) {
   print "$firstnme: $lastname\n";
}

# comprobar si hay problemas que puedan haber cancelado antes la captación
warn $DBI::errstr if $DBI::err;

$sth->finish;
$dbh->disconnect;
```

Marcadores de parámetro en Perl

El módulo DBI de Perl soporta la ejecución de una sentencia preparada que incluye marcadores de parámetro para la entrada de variables. Para incluir un marcador de parámetro en una sentencia de SQL, utilice un signo de interrogación (?) o un carácter de dos puntos seguido de un nombre (`:name`).

El ejemplo de código Perl crea un descriptor de contexto de sentencia que acepta un marcador de parámetro para la cláusula WHERE de una sentencia SELECT. A continuación, el código ejecuta la sentencia dos veces, utilizando los valores de entrada 25000 y 35000 para sustituir el marcador de parámetro.

```
my $sth = $dbhandle->prepare(
   'SELECT firstnme, lastname
      FROM employee
      WHERE salary > ?'
   );

my $rc = $sth->execute(25000);
```

•
•
•

```
my $rc = $sth->execute(35000);
```

Variables SQLSTATE y SQLCODE en Perl

El módulo DBI de Perl proporciona métodos para devolver el SQLSTATE y el SQLCODE asociados con un descriptor de contexto de sentencia o de base de datos del DBI de Perl.

Para devolver el SQLSTATE asociado a un descriptor de contexto de base de datos del DBI de Perl o a un descriptor de contexto de sentencia, llame al método state. Por ejemplo, para devolver el SQLSTATE asociado al descriptor de contexto de base de datos $dbhandle, incluya la sentencia de Perl `my $sqlstate = $dbhandle->state;` en la aplicación:

Para devolver el SQLCODE asociado a un descriptor de contexto de base de datos del DBI de Perl o a un descriptor de contexto de sentencia, llame al método err. Para devolver el mensaje para un SQLCODE asociado a un descriptor de contexto de base de datos del DBI de Perl o a un descriptor de contexto de sentencia, llame al método errstr. Por ejemplo, para devolver el SQLCODE asociado al descriptor de contexto de base de datos $dbhandle, incluya la sentencia de Perl `my $sqlcode = $dbhandle->err;` en la aplicación:

Restricciones de Perl

Se aplican algunas restricciones al soporte que está disponible para el desarrollo de aplicaciones en Perl.

El módulo DBI de Perl sólo soporta SQL dinámico. Cuando tenga que ejecutar una sentencia varias veces, se puede mejorar el rendimiento de las aplicaciones Perl emitiendo una llamada **prepare** para preparar la sentencia.

Perl no da soporte al acceso a bases de datos de varias hebras.

Para obtener información actual sobre las restricciones de la versión del controlador de DBD::DB2 que instale en la estación de trabajo, consulte el archivo CAVEATS contenido en el paquete del controlador de DBD::DB2.

pureXML y Perl

El controlador DBD::DB2 da soporte a DB2 pureXML. El soporte para pureXML ofrece un acceso más directo a los datos a través del controlador DBD::DB2 y ayuda a reducir la lógica de aplicación ofreciendo una comunicación más transparente entre la aplicación y la base de datos.

Con el soporte de pureXML, puede insertar directamente documentos XML en la base de datos DB2. La aplicación ya no necesita analizar los documentos XML porque el analizador de pureXML se ejecuta automáticamente al insertar los datos XML en la base de datos. La posibilidad de gestionar el análisis de los documentos desde el exterior de la aplicación mejora el rendimiento de la aplicación y reduce los esfuerzos de mantenimiento. También resulta fácil la recuperación de datos XML almacenados con el controlador DBD::DB2; puede acceder a los datos utilizando un BLOB o registro.

Para obtener información sobre DB2 Perl Database Interface y sobre cómo descargar el controlador DBD::DB2 más reciente, consulte http://www.ibm.com/software/data/db2/perl.

Ejemplo

El ejemplo es un programa Perl que emplea pureXML:

```
#!/usr/bin/perl
use DBI;
use strict ;

# Utilizar módulo DBD:DB2:
#   para crear una sola tabla DB2 con una columna XML
#   Añadir una fila de datos
#   recuperar los datos XML como un registro o un LOB (basado en $datatype).

# NOTA: la base de datos DB2 SAMPLE debe existir ya.

my $database='dbi:DB2:sample';
my $user='';
my $password='';

my $datatype = "record" ;
#  $datatype = "LOB" ;

my $dbh = DBI->connect($database, $user, $password)
  or die "Can't connect to $database: $DBI::errstr";

# Para el tipo de datos LOB, LongReadLen = 0 -- en la captura inicial
# no se recuperan datos
$dbh->{LongReadLen} = 0 if $datatype eq "LOB" ;

# SQL CREATE TABLE para crear la tabla de prueba
my $stmt = "CREATE TABLE xmlTest (id INTEGER, data XML)";
my $sth = $dbh->prepare($stmt);
$sth->execute();

#insertar una fila de datos en la tabla
insertData() ;

# La sentencia de SQL SELECT devuelve el elemento de teléfono
# de casa de los datos XML
$stmt = qq(
 SELECT XMLQUERY ('
 \$d/*:customerinfo/*:phone[\@type = "home"] '
 passing data as "d")
 FROM xmlTest
) ;


# preparar y ejecutar una sentencia SELECT
$sth = $dbh->prepare($stmt);
$sth->execute();

# Imprimir los datos devueltos de la sentencia select
if($datatype eq "LOB") {
    printLOB() ;
}
else {
 printRecord() ;
}

# Descartar tabla
$stmt = "DROP TABLE xmlTest" ;
$sth = $dbh->prepare($stmt);
$sth->execute();

warn $DBI::errstr if $DBI::err;

$sth->finish;
```

```
$dbh->disconnect;


##############


sub printRecord {
 print "output data as as record\n" ;

 while( my @row = $sth->fetchrow )
 {
  print $row[0] . "\n";
 }

 warn $DBI::errstr if $DBI::err;
}


sub printLOB {
 print "output as Blob data\n" ;

 my $offset = 0;
 my $buff="";
 $sth->fetch();
 while( $buff = $sth->blob_read(1,$offset,1000000)) {
  print $buff;
  $offset+=length($buff);
  $buff="";
 }
 warn $DBI::errstr if $DBI::err;
}


sub insertData {

 # insertar una fila de datos
 my $xmlInfo = qq(\'
 <customerinfo xmlns="http://posample.org" Cid="1011">
   <name>Bill Jones</name>
   <addr country="Canada">
     <street>5 Redwood</street>
     <city>Toronto</city>
     <prov-state>Ontario</prov-state>
     <pcode-zip>M6W 1E9</pcode-zip>
   </addr>
   <phone type="work">416-555-9911</phone>
   <phone type="home">416-555-1212</phone>
 </customerinfo>
 \') ;

 my $catID = 1011 ;

 # Sentencia de SQL para insertar datos.
 my $Sql = qq(
  INSERT INTO xmlTest (id, data)
      VALUES($catID, $xmlInfo )
 );

 $sth = $dbh->prepare( $Sql )
   or die "Can't prepare statement: $DBI::errstr";

 my $rc = $sth->execute
   or die "Can't execute statement: $DBI::errstr";

 # comprobar si hay problemas
 warn $DBI::errstr if $DBI::err;
}
```

Ejecución de programas de ejemplo de Perl

Hay disponibles programas de ejemplo de Perl que muestran cómo crear una aplicación Perl.

Antes de empezar

Antes de ejecutar los programas de ejemplo de Perl, debe instalar la última versión del controlador DB2::DB2 para la DBI de Perl. Para conocer más información sobre cómo obtener el controlador más reciente, consulte http://www.ibm.com/software/data/db2/perl.

Acerca de esta tarea

Los programas de ejemplo de Perl para bases de datos DB2 están disponibles en el directorio `sqllib/samples/perl`.

Procedimiento

Para ejecutar el intérprete de Perl en un programa de ejemplo de Perl desde la línea de mandatos:

Entre el nombre del intérprete y el nombre del programa (incluyendo la extensión del archivo):

- Si está conectando localmente en el servidor:

  ```
  perl dbauth.pl
  ```

- Si está conectando desde un cliente remoto:

  ```
  perl dbauth.pl sample <IDusuario> <contraseña>
  ```

Para algunos programas de ejemplo, tendrá que ejecutar archivos de soporte. Por ejemplo, el programa de ejemplo `tbsel` requiere varias tablas creadas por el script `tbselcreate.db2` del CLP. El script `tbselinit` (UNIX) o el archivo de proceso por lotes `tbselinit.bat` (Windows), en primer lugar llama a `tbseldrop.db2` para eliminar las tablas, en caso de que existan, y luego llama a `tbselcreate.db2` para crearlas. Por consiguiente, para ejecutar el programa de ejemplo `tbsel`, emita los mandatos indicados:

- Si está conectando localmente en el servidor:

  ```
  tbselinit
  perl tbsel.pl
  ```

- Si está conectando desde un cliente remoto:

  ```
  tbselinit
  perl tbsel.pl sample <IDusuario> <contraseña>
  ```

Nota: Para un cliente remoto debe modificar la sentencia de conexión en los archivos `tbselinit` o `tbselinit.bat` para codificar el ID de usuario y la contraseña: `db2 connect to sample user <IDusuario> using <contraseña>`

Ejecución de rutinas desde aplicaciones Perl

Las aplicaciones cliente de DB2 pueden acceder a rutinas (procedimientos almacenados y funciones definidas por el usuario) creadas mediante lenguajes del sistema principal soportados o mediante procedimientos de SQL. Por ejemplo, el programa de ejemplo `spclient.pl` puede acceder a la biblioteca compartida de los procedimientos de SQL spserver, en caso de que exista en la base de datos.

Antes de empezar

Para crear una rutina de lenguaje del sistema principal debe tener configurado el compilador apropiado en el servidor. Los procedimientos de SQL no necesitan compilador. La biblioteca compartida sólo se puede crear en el servidor, y no desde un cliente remoto.

Procedimiento

Para crear procedimientos de SQL en una biblioteca compartida y, a continuación, accesos a los procedimientos desde una aplicación Perl:

1. Cree y catalogue los procedimientos de SQL de la biblioteca. Por ejemplo, vaya al directorio `samples/sqlpl` del servidor y ejecute los mandatos indicados para crear y catalogar los procedimientos de SQL en la biblioteca `spserver`:

   ```
   db2 connect to sample
   db2 -td@ -vf spserver.db2
   ```

2. Vuelva al directorio de muestras `perl` (puede estar en una estación de trabajo cliente remota) y ejecute el intérprete de Perl en el programa cliente para acceder a la biblioteca compartida `spserver`:
   - Si está conectando localmente en el servidor:

     ```
     perl spclient
     ```

   - Si está conectando desde un cliente remoto:

     ```
     perl spclient sample <IDusuario> <contraseña>
     ```

Capítulo 2. Desarrollo de aplicaciones PHP

Desarrollo de aplicaciones PHP para servidores de datos de IBM

PHP: Hypertext Preprocessor (PHP) es un lenguaje de programación interpretado que se utiliza habitualmente para el desarrollo de aplicaciones web. PHP se ha convertido en un lenguaje muy usado en el desarrollo Web porque es fácil de aprender a utilizar, se centra en soluciones prácticas y da soporte a las funciones más utilizadas en aplicaciones web.

PHP es un lenguaje modular que le permite personalizar la funcionalidad disponible mediante el uso de extensiones. Esas extensiones pueden simplificar tareas como la lectura, escritura y manipulación de XML, la creación de clientes y servidores SOAP y el cifrado de comunicaciones entre servidor y navegador. Sin embargo, las extensiones más utilizadas para PHP proporcionan acceso de lectura y escritura a bases de datos, por lo que puede crear fácilmente un sitio web dinámico destinado a bases de datos.

IBM ofrece las extensiones PHP listadas para acceder a bases de datos de servidores de datos de IBM:

ibm_db2
: Una interfaz de programación de aplicaciones (API) de procedimientos que ofrece, además de las operaciones de base de datos de creación, lectura, actualización y grabación normales, un amplio acceso a los metadatos de la base de datos. Puede compilar la extensión ibm_db2 con PHP 4 o con PHP 5. IBM escribe, mantiene y ofrece soporte a esta extensión.

pdo_ibm
: Controlador para la extensión de objetos de datos PHP (PDO) que ofrece acceso a bases de datos de servidores de datos de IBM mediante la interfaz de base de datos orientada a objetos estándar implementada en PHP 5.1.

Estas extensiones se incluyen como parte de la Versión 1.7.0 de IBM Data Server Driver Package (controlador DS). Esta versión, o una versión posterior, recibe el soporte necesario para conectarse con IBM DB2 Versión 9.7 para Linux, UNIX y Windows. Puede comprobar la versión de la extensión ibm_db2 emitiendo el mandato **php --re ibm_db2**.

Las versiones más recientes de ibm_db2 y pdo_ibm también están disponibles a través de la biblioteca PECL (PHP Extension Community Library) en http://pecl.php.net/.

Las aplicaciones PHP pueden acceder a las bases de datos listadas de servidor de datos de IBM:

- IBM DB2 Versión 9.1 para Linux, UNIX y Windows, Fixpack 2 y posterior
- IBM DB2 Universal Database (DB2 UDB) Versión 8 para Linux, UNIX y Windows, Fixpack 15 y posterior
- Conexiones remotas a IBM DB2 para IBM i V5R3
- Conexiones remotas a IBM DB2 para IBM i Versión 5.4 y posteriores
- Conexiones remotas a IBM DB2 para z/OS, Versión 8 y posteriores

Una tercera extensión, Unified ODBC, ha ofrecido históricamente acceso a los sistemas de bases de datos DB2. No obstante, puede utilizar ibm_db2 y pdo_ibm para las aplicaciones nuevas, ya que ofrecen ventajas significativas en cuanto a rendimiento y estabilidad en comparación con Unified ODBC. La API de la extensión ibm_db2 convierte la tarea de transportar una aplicación escrita anteriormente para Unified ODBC en algo casi tan sencillo como realizar un cambio global del nombre de función `odbc_` por `db2_` en el código fuente de la aplicación.

Descargas y recursos relacionados de PHP

Hay disponibles muchos recursos para ayudarle a desarrollar aplicaciones PHP para servidores de datos de IBM.

Tabla 2. Descargas y recursos relacionados de PHP.

Esta tabla muestra los recursos de descarga y el enlace a estos. La fila superior de esta tabla es una celda expandida que contiene el título de la tabla.

Descargas	
Código fuente de PHP completo [1]	http://www.php.net/downloads.php
ibm_db2 y pdo_ibm desde la biblioteca PECL (PHP Extension Community Library)	http://pecl.php.net/
IBM Data Server Driver Package (controlador DS)	http://www.ibm.com/software/data/support/data-server-clients/index.html
Zend Server	http://www.zend.com/en/products/server/downloads
Manual de PHP	http://www.php.net/docs.php
Documentación de la API ibm_db2	http://www.php.net/ibm_db2
Documentación de la API de PDO	http://php.net/manual/en/book.pdo.php
Sitio web de PHP	http://www.php.net/

1. Incluye binarios de Windows. La mayoría de las distribuciones Linux incorporan PHP previamente compilado.

Configuración del entorno PHP

Puede configurar el entorno de PHP en sistemas operativos Linux, UNIX o Windows instalando una versión binaria precompilada de PHP y habilitando el soporte para los servidores de datos de IBM.

Acerca de esta tarea

Para facilitar la instalación y configuración en los sistemas operativos Linux, UNIX o Windows, puede descargar e instalar Zend Server para utilizarlo en sistemas de producción en http://www.zend.com/en/products/server/downloads. En http://www.zend.com/en/products/server/editions encontrará información sobre el paquete.

En Windows, las versiones binarias precompiladas de PHP están disponibles para su descarga en http://www.php.net/downloads.php. La mayoría de las distribuciones Linux incluyen una versión precompilada de PHP. En sistemas operativos UNIX que no incluyen una versión precompilada de PHP, puede compilar su propia versión de PHP.

Para obtener más información sobre la instalación y configuración de PHP, consulte http://www.php.net/manual/en/install.php.

Configuración del entorno de PHP en Windows

Antes de poder conectarse a un servidor de datos de IBM y ejecutar sentencias de SQL, debe configurar el entorno PHP.

Antes de empezar

Debe tener el software necesario instalado en su sistema:

- Apache HTTP Server
- Uno de los tipos de cliente listados: IBM Data Server Driver Package, IBM Data Server Client o IBM Data Server Driver para ODBC y CLI

Acerca de esta tarea

Este procedimiento instala de forma manual una versión binaria precompilada de PHP y habilita soporte para los servidores de datos de IBM en Windows.

Procedimiento

Para configurar el entorno PHP en Windows:

1. Descargue la última versión del paquete comprimido de PHP (en este momento, PHP 5.3.x) de http://windows.php.net/download/ y la recopilación de módulos PECL de http://php.net/releases/index.php. La recopilación de módulos PECL está disponible sólo para las versiones de PHP 5.2.x. Tenga en cuenta que se recomienda la compilación Non Thread Safe de PHP cuando no utiliza PHP como módulo Apache.
2. Extraiga el paquete comprimido de PHP en un directorio de instalación.
3. Extraiga la recopilación de módulos PECL en el subdirectorio `\ext\` del directorio de instalación de PHP.
4. Cree un nuevo archivo denominado `php.ini` en el directorio de instalación realizando una copia del archivo `php.ini-recommended`.
5. Abra el archivo `php.ini` en un editor de texto y añada las líneas en el bloque de código.
   - Para habilitar la extensión PDO y el controlador pdo_ibm:

     ```
     extension=php_pdo.dll
     extension=php_pdo_ibm.dll
     ```

   - Para habilitar la extensión ibm_db2:

     ```
     extension=php_ibm_db2.dll
     ```

6. Si utiliza Apache HTTP Server 2.x., habilite el soporte de PHP añadiendo las siguientes líneas al archivo `httpd.conf`, en el que *phpdir* hace referencia al directorio de instalación de PHP:

   ```
   LoadModule php5_module 'phpdir/php5apache2_2.dll'
   AddType application/x-httpd-php .php
   PHPIniDir 'phpdir'
   ```

7. Vuelva a iniciar Apache HTTP Server para habilitar la configuración modificada.

Resultados

Nota: Si se produce un mensaje DB21085I o SQL09054, se debe realizar una de las tareas listadas:

- Reconstruir PHP en modalidad de 64 bits
- Establecer las variables *PHP_IBM_DB2_LIB* y *PHP_PDO_IBM_LIB* para que utilicen `lib32` en lugar del valor por omisión `lib64`, y actualice *LD_LIBRARY_PATH* para que apunte a `lib32`.

Las extensiones PHP se habrán instalado en su sistema y estarán listos para usar.

Qué hacer a continuación

Conéctese al servidor de datos y empiece a ejecutar sentencias de SQL.

Configuración del entorno PHP en Linux o UNIX

Antes de poder conectarse a un servidor de datos de IBM y ejecutar sentencias de SQL, debe configurar el entorno PHP.

Antes de empezar

DB2 da soporte al acceso de bases de datos para aplicaciones cliente escritas en el lenguaje de programación PHP utilizando la extensión ibm_db2 o el controlador pdo_ibm para la extensión PHP Data Objects (PDO), o bien ambos.

Debe tener el software y los archivos listados instalados en su sistema:

- Apache HTTP Server
- El compilador gcc y los paquetes apache-devel, autoconf, automake, bison, flex, gcc y libxml2-devel.
- Si se conecta desde PHP a un servidor de bases de datos DB2 en una máquina remota, necesita uno de los clientes DB2 en la máquina donde esté instalando o ejecutando PHP, por ejemplo, IBM Data Server Driver Package, IBM Data Server Client o IBM Data Server Driver para ODBC y CLI.
- Si se conecta desde PHP a un servidor de DB2 en una máquina local, no necesita instalar ningún cliente de DB2.

Acerca de esta tarea

Este procedimiento compila e instala PHP de forma manual desde el programa fuente con soporte para DB2 en Linux o UNIX.

Procedimiento

Para configurar el entorno PHP en Linux o UNIX:

1. Descargue la última versión de PHP tarball de http://www.php.net. La última versión de PHP en el momento de escribir este manual es PHP 5.3.x.
2. Descomprima el archivo emitiendo el mandato `tar -xjf php-5.x.x.tar.bz2`.
3. Cambie el directorio al directorio `php-5.x.x` recién creado.
4. Configure el makefile emitiendo el mandato `configure`. Especifique las funciones y extensiones que desea incluir en la versión personalizada de PHP. Un mandato de configuración típico incluye las opciones listadas:

   ```
   ./configure --enable-cli --disable-cgi --with-apxs2=/usr/sbin/apxs2
   --with-zlib --with-pdo-ibm=<sqllib> --with-IBM_DB2=<sqllib>
   ```

   Las opciones de configuración tienen los efectos listados:

   --enable-cli
   Habilita la modalidad de línea de mandatos del acceso de PHP.

--disable-cgi
Inhabilita la modalidad Common Gateway Interface (CGI) del acceso de PHP.

--with-apxs2=/usr/sbin/apxs2
Habilita la modalidad de Apache 2 Dynamic Server Object (DSO) del acceso de PHP.

--with-zlib
Habilita el soporte de compresión de zlib.

--with-pdo-ibm=<sqllib>
Habilita el controlador de pdo_ibm utilizando la biblioteca de CLI para acceder a los sistemas de bases de datos. El valor *<sqllib>* hace referencia al directorio en el que se instala DB2.

Si el código fuente de las extensiones pdo_ibm no está en el directorio `ext/` de la fuente PHP, este distintivo no será válido. Para utilizar `--with-pdo-ibm`, debe tener el directorio pdo_ibm, que contiene el código fuente de pdo_ibm, en el subdirectorio `ext/`.

--with-IBM_DB2=<sqllib>
Habilita el controlador de IBM_DB2 utilizando la biblioteca para acceder a los sistemas de bases de datos. El valor *<sqllib>* hace referencia al directorio en el que se instala DB2.

Si el código fuente de la extensión IBM_DB2 no está en el directorio `ext/` de la fuente PHP, este distintivo no será válido. Para utilizar esta opción de configuración, debe tener el directorio `ibm_db2`, que contiene el código fuente de ibm_db2 en el subdirectorio `ext/`.

5. Compile los archivos emitiendo el mandato `make`.
6. Instale los archivos emitiendo el mandato `make install`. En función de cómo haya configurado el directorio de instalación de PHP utilizando el mandato `configure`, es posible que necesite autorización de root para emitir satisfactoriamente este mandato. Esto instala los archivos ejecutables y actualiza la configuración de Apache HTTP Server para que dé soporte a PHP.
7. Instale la extensión ibm_db2 emitiendo el mandato `pecl install ibm_db2` como usuario con autorización root.

   Este mandato descarga, configura, compila e instala la extensión ibm_db2 para PHP. Se recomienda utilizar la extensión más reciente. No obstante, también puede usar la extensión ibm_db2 que se incluye como parte de los productos DB2.
8. Copie el archivo `php.ini-development` o `php.ini-production` en la vía de acceso del archivo de configuración correspondiente a la nueva instalación de PHP. Para determinar la vía de acceso del archivo de configuración, emita el mandato `php -i` y busque la palabra clave `php.ini`. Cambie el nombre del archivo por `php.ini`.
9. Abra el nuevo archivo `php.ini` con un editor de texto y añada las siguientes líneas, donde *instance* es el nombre de la instancia de DB2 en Linux o UNIX.
   - Para habilitar la extensión pdo_ibm y establecer el entorno:

     ```
     extension=pdo_ibm.so
     PDO_IBM.db2_instance_name=instancia
     ```
   - Para habilitar la extensión ibm_db2 y establecer el entorno de DB2:

     ```
     extension=ibm_db2.so
     ibm_db2.instance_name=instancia
     ```

   Donde puede especificarse la variable de extensión como vía de acceso relativa desde el directorio de extensión, que se especifica en la variable

extension_dir. Por ejemplo, si extension_dir es *$HOME*/usr/php/ext y la extensión se encuentra en *$HOME*/user/sqllib/php32, la entrada tendrá el aspecto siguiente: extension=../../sqllib/php32/ibm_db2_5.2.1.so

10. Antes de ejecutar cualquier script PHP que se conecte a DB2, debe asegurarse de que los controladores de IBM DB2 PHP (PDO_IBM/IBM_DB2) pueden acceder al controlador de CLI libdb2.so, que forma parte de la configuración de su servidor DB2 o de la configuración del cliente de DB2. Si no lo hace, aparecerá un error de faltan las bibliotecas - libdb2.so.1 cuando ejecute el script PHP. En Linux, puede hacerlo añadiendo la carpeta donde reside el archivo libdb2.so en la variable de entorno **LD_LIBRARY_PATH** correspondiente al id de usuario en el que se vaya a ejecutar PHP.

    Cuando se utiliza IBM Data Server Driver Package, libdb2.so se encuentra situado en el directorio odbc_cli_driver/linux/clidriver/lib.

    En la instalación del servidor DB2, libdb2.so se encuentra en el directorio sqllib/lib.
11. Vuelva a iniciar Apache HTTP Server para habilitar la configuración modificada.

Resultados

Nota: Si se produce un mensaje DB21085I o SQL09054, puede realizar una de las tareas listadas:

- Reconstruir PHP en modalidad de 64 bits
- Establecer las variables *PHP_IBM_DB2_LIB* y *PHP_PDO_IBM_LIB* para que utilicen lib32 en lugar del valor por omisión lib64, y actualice *LD_LIBRARY_PATH* para que apunte a lib32.

Desarrollo de aplicaciones en PHP (ibm_db2)

La extensión ibm_db2 proporciona una serie de útiles funciones de PHP para acceder a los datos y manipularlos en una base de datos de IBM Data Server. La extensión incluye funciones para conectarse a una base de datos, ejecutar y preparar sentencias de SQL, recuperar filas de conjuntos de resultados, llamar a procedimientos almacenados, gestionar errores y recuperar metadatos.

Conexión a una base de datos de IBM Data Server en PHP (ibm_db2)

Antes de poder emitir sentencias de SQL para crear, actualizar, suprimir o recuperar datos, debe conectarse a una base de datos de la aplicación PHP. Puede utilizar la API ibm_db2 para conectar con una base de datos de IBM Data Server mediante una conexión catalogada o una conexión TCP/IP directa. Para mejorar el rendimiento, también puede crear una conexión persistente.

Antes de empezar

Antes de conectar con una base de datos de IBM Data Server mediante la extensión ibm_db2, debe configurar el entorno de PHP en el sistema y habilitar la extensión ibm_db2.

Procedimiento

Para devolver un recurso de conexión que puede utilizar para llamar a sentencias de SQL, llame a una de las funciones de conexión listadas:

Tabla 3. Funciones de conexión de ibm_db2

Función	Descripción
db2_connect	Crea una conexión no persistente.
db2_pconnect	Crea una conexión persistente. Permanece abierta una conexión persistente entre peticiones de PHP, lo que permite que las peticiones de script de PHP posteriores reutilicen la conexión si tienen un conjunto de credenciales idéntico.

Los valores de base de datos que proporcione como argumentos para estas funciones pueden especificar un nombre de base de datos catalogado o una serie de conexión de base de datos completa para una conexión TCP/IP directa. Puede especificar argumentos opcionales que controlen cuándo se confirman las transacciones, las mayúsculas/minúsculas de los nombres de columna que se devuelven y el tipo de cursor.
Si el intento de conexión falla, puede recuperar información de diagnóstico llamando a la función db2_conn_error o db2_stmt_errormsg.
Al crear una conexión llamando a la función db2_connect, PHP cierra la conexión con la base de datos cuando se produce uno de los sucesos listados:

- Llama a la función db2_close para la conexión
- Establece el recurso de conexión en NULL
- Finaliza el script de PHP

Cuando crea una conexión llamando a la función db2_pconnect, PHP ignora cualquier llamada a la función db2_close para el recurso de conexión especificado y mantiene abierta la conexión con la base de datos para los scripts de PHP posteriores.
Para obtener más información sobre la API ibm_db2, consulte http://www.php.net/docs.php.

Ejemplo

Conectar con una base de datos catalogada.

```
<?php
$database = "sample";
$user = "db2inst1";
$password = "";

$conn = db2_connect($database, $user, $password);

if($conn) {
echo "Connection succeeded.";
db2_close($conn);
}
else {
echo "Connection failed.";
}
?>
```

Qué hacer a continuación

Si el intento de conexión se realiza correctamente, puede utilizar el recurso de conexión al llamar a funciones de ibm_db2 que ejecutan sentencias de SQL. A continuación, prepare y ejecute las sentencias de SQL.

Contextos fiables en aplicaciones PHP (ibm_db2):

A partir de la Versión 9.5 Fixpack 3 (o posterior), la extensión de ibm_db2 admite contextos fiables mediante el uso de palabras clave de serie de conexión.

Los contextos fiables proporcionan un método para crear aplicaciones de tres niveles más seguras y con mayor rapidez. La identidad del usuario siempre se mantiene para operaciones de auditoría y seguridad. Cuando se necesitan conexiones seguras, los contextos fiables mejoran el rendimiento ya que no es necesario obtener conexiones nuevas.

Ejemplo

Habilitar contextos fiables, alternar usuarios y obtener el ID de usuario actual.

```
<?php

$database = "SAMPLE";
$hostname = "localhost";
$port = 50000;
$authID = "db2inst1";
$auth_pass = "ibmdb2";

$tc_user = "tcuser";
$tc_pass = "tcpassword";

$dsn = "DATABASE=$database;HOSTNAME=$hostname;PORT=$port;PROTOCOL=TCPIP;UID=$authID;PWD=$auth_pass;";
$options = array ("trustedcontext" => DB2_TRUSTED_CONTEXT_ENABLE);

$tc_conn = db2_connect($dsn, "", "", $options);
if($tc_conn) {
 echo "Explicit Trusted Connection succeeded.\n";

 if(db2_get_option($tc_conn, "trustedcontext")) {
  $userBefore = db2_get_option($tc_conn, "trusted_user");

  //Realizar algún trabajo como usuario 1.

  //Cambiando a usuario de confianza.
  $parameters = array("trusted_user" => $tc_user, "trusted_password" => $tcuser_pass);
  $res = db2_set_option ($tc_conn, $parameters, 1);

  $userAfter = db2_get_option($tc_conn, "trusted_user");
  //Realizar más trabajo como usuario de confianza.

  if($userBefore != $userAfter) {
   echo "User has been switched." . "\n";
  }
 }

 db2_close($tc_conn);
}
else {
 echo "Explicit Trusted Connection failed.\n";
}

?>
```

Ejecución de sentencias de SQL en PHP (ibm_db2)

Después de conectarse a una base de datos, utilice las funciones disponibles en la API ibm_db2 para preparar y ejecutar sentencias de SQL. Las sentencias de SQL pueden contener texto estático, expresiones XQuery o marcadores de parámetros que representan la entrada de variables.

Ejecución de una única sentencia de SQL en PHP (ibm_db2):

Para preparar y ejecutar una única sentencia de SQL que no acepta parámetros de entrada, utilice la función db2_exec. Un uso típico de la función db2_exec es establecer el esquema por omisión para la aplicación en un archivo de inclusión común o clase base.

Antes de empezar

Para evitar amenazas a la seguridad por ataques de inyección de SQL, utilice la función db2_exec sólo para ejecutar sentencias de SQL compuestas de series

estáticas. La interpolación de variables de PHP que representan la entrada del usuario en la sentencia de SQL puede exponer la aplicación a ataques de inyección de SQL.

Obtener un recurso de conexión invocando una de las funciones de conexión en la API ibm_db2. Consulte "Conexión a una base de datos de IBM Data Server en PHP (ibm_db2)" en la página 16.

Procedimiento

Para preparar y ejecutar una única sentencia de SQL, llame a la función db2_exec, proporcionando los argumentos listados:

conexión
Un recurso de conexión de base de datos válido que devuelve la función db2_connect o db2_pconnect.

sentencia
Una serie que contiene la sentencia de SQL. Esta serie puede incluir una expresión XQuery a la que llama la función XMLQUERY.

opciones
Opcional: Una matriz asociativa que especifica las opciones de sentencias:

DB2_ATTR_CASE
Para conseguir compatibilidad con sistemas de bases de datos que no sigan el estándar SQL, esta opción establece el caso en los que los nombres de columna se devolverán a la aplicación. Por omisión, el caso de mayúsculas/minúsculas está establecido en DB2_CASE_NATURAL, lo que devuelve nombres de columna tal y como los devuelve la base de datos. Puede establecer este parámetro en DB2_CASE_LOWER para obligar a que los nombres de columna estén en minúsculas, o en DB2_CASE_UPPER para obligar a que los nombres de columna estén en mayúsculas.

DB2_ATTR_CURSOR
Esta opción establece el tipo de cursor que ibm_db2 devuelve para los conjuntos de resultados. Por omisión, ibm_db2 devuelve un cursor de sólo avance (DB2_FORWARD_ONLY) que devuelve la fila siguiente de un conjunto de resultados para cada llamada a db2_fetch_array, db2_fetch_assoc, db2_fetch_both, db2_fetch_object o db2_fetch_row. Puede establecer este parámetro en DB2_SCROLLABLE para solicitar un cursor desplazable de forma que las funciones de captación de ibm_db2 acepten un segundo argumento especificando la posición absoluta de la fila a la que desea acceder en el conjunto de resultados.

Si la llamada de función tiene éxito, devuelve un recurso de sentencia que puede utilizar en llamadas de función posteriores relacionadas con esta consulta.
Si la llamada de función falla (devuelve `False`), puede utilizar la función db2_stmt_error o db2_stmt_errormsg para recuperar información de diagnóstico sobre el error.
Para obtener más información sobre la API ibm_db2, consulte http://www.php.net/docs.php.

Ejemplo

Ejemplo 1: Ejecución de una única sentencia de SQL.

```
<?php
$conn = db2_connect("sample", "db2inst1", "");
$sql = "SELECT * FROM DEPT";
$stmt = db2_exec($conn, $sql);
db2_close($conn);
?>
```

Ejemplo 2: Ejecución de una expresión XQuery.

```
<?php
$xquery = '$doc/customerinfo/phone';
$stmt = db2_exec($conn, "select xmlquery('$xquery'
PASSING INFO AS \"doc\") from customer");?>
```

Qué hacer a continuación

Si la sentencia de SQL ha seleccionado filas utilizando un cursor desplazable, o ha insertado, actualizado o suprimido filas, puede llamar a la función db2_num_rows para devolver el número de filas que la sentencia ha devuelto o a las que ha afectado. Si la sentencia de SQL ha devuelto un conjunto de resultados, puede empezar a captar filas.

Preparación y ejecución de sentencias de SQL con entrada de variables en PHP (ibm_db2):

Para preparar y ejecutar una sentencia de SQL que incluya la entrada de variables, utilice las funciones db2_prepare, db2_bind_param y db2_execute. Preparar una sentencia mejora el rendimiento porque el servidor de bases de datos crea un plan de acceso optimizado para la recuperación de datos que puede reutilizar si se vuelve a ejecutar la sentencia.

Antes de empezar

Obtener un recurso de conexión invocando una de las funciones de conexión en la API ibm_db2. Consulte “Conexión a una base de datos de IBM Data Server en PHP (ibm_db2)” en la página 16.

Procedimiento

Para preparar y ejecutar una sentencia de SQL que incluye marcadores de parámetro:

1. Llame a la función db2_prepare, pasando los argumentos listados:

   conexión
   Un recurso de conexión de base de datos válido que devuelve la función db2_connect o db2_pconnect.

   sentencia
   Una serie que contiene la sentencia de SQL, incluidos los signos de interrogación (?), como marcadores de parámetro para cualquier valor de columna o predicado que requiera entrada de variables. Esta serie puede incluir una expresión XQuery a la que llama la función XMLQUERY. Sólo puede utilizar marcadores de parámetro como lugar reservado para los valores de columna o predicado. El compilador de SQL no puede crear un plan de acceso para una sentencia que usa marcadores de parámetro en lugar de nombres de columna, nombres de tabla u otros identificadores de SQL.

   opciones
   Opcional: Una matriz asociativa que especifica las opciones de sentencias:

DB2_ATTR_CASE
Para conseguir compatibilidad con sistemas de bases de datos que no sigan el estándar SQL, esta opción establece el caso en los que los nombres de columna se devolverán a la aplicación. Por omisión, el caso de mayúsculas/minúsculas está establecido en DB2_CASE_NATURAL, lo que devuelve nombres de columna tal y como los devuelve la base de datos. Puede establecer este parámetro en DB2_CASE_LOWER para obligar a que los nombres de columna estén en minúsculas, o en DB2_CASE_UPPER para obligar a que los nombres de columna estén en mayúsculas.

DB2_ATTR_CURSOR
Esta opción establece el tipo de cursor que ibm_db2 devuelve para los conjuntos de resultados. Por omisión, ibm_db2 devuelve un cursor de sólo avance (DB2_FORWARD_ONLY) que devuelve la fila siguiente de un conjunto de resultados para cada llamada a db2_fetch_array, db2_fetch_assoc, db2_fetch_both, db2_fetch_object o db2_fetch_row. Puede establecer este parámetro en DB2_SCROLLABLE para solicitar un cursor desplazable de forma que las funciones de captación de ibm_db2 acepten un segundo argumento especificando la posición absoluta de la fila a la que desea acceder en el conjunto de resultados.

Si la llamada a la función tiene éxito, devuelve un recurso de descriptor de contexto de sentencia que puede utilizar en posteriores llamadas a función relacionadas con esta consulta.

Si la llamada de función falla (devuelve `False`), puede utilizar la función db2_stmt_error o db2_stmt_errormsg para recuperar información de diagnóstico sobre el error.

2. Opcional: Por cada marcador de parámetro en la serie de SQL, llame a la función db2_bind_param, proporcionando los argumentos listados. Vincular valores de entrada a marcadores de parámetros garantiza que cada valor de entrada se trate como un único parámetro, evitando ataques de inyección de SQL contra la aplicación.

   sentencia
   Una sentencia preparada que devuelve la llamada a la función db2_prepare.

   número_parámetro
   Un entero que representa la posición del marcador de parámetro en la sentencia de SQL.

   nombre-variable
   Una serie que especifica el nombre de la variable PHP que se vinculará al parámetro especificado en *número_parámetro*.

3. Llame a la función db2_execute, pasando los argumentos listados:

   sentencia
   Una sentencia preparada que devuelve la función db2_prepare.

   parámetros
   Opcional: Una matriz que contiene los valores que se deben utilizar en lugar de los marcadores de parámetro, en orden.

   Para obtener más información sobre la API ibm_db2, consulte http://www.php.net/docs.php.

Ejemplo

Preparar y ejecutar una sentencia que incluye entrada de variables.

```
$sql = "SELECT firstnme, lastname FROM employee WHERE bonus > ? AND bonus < ?";
$stmt = db2_prepare($conn, $sql);
if (!$stmt) {
  // Manejar los errores
}

// Vincular explícitamente los parámetros
db2_bind_param($stmt, 1, $_POST['lower']);
db2_bind_param($stmt, 2, $_POST['upper']);

db2_execute($stmt);
// Procesar los resultados

// Invocar de nuevo sentencia preparada usando parámetros vinculados dinámicamente
db2_execute($stmt, array($_POST['lower'], $_POST['upper']));
```

Qué hacer a continuación

Si la sentencia de SQL devuelve uno o varios conjuntos de resultados, puede empezar a captar filas del recurso de sentencia.

Inserción de objetos grandes en PHP (ibm_db2):

Cuando inserta un objeto grande en la base de datos, en lugar de cargar todos los datos del objeto grande en una serie de PHP y entonces pasarlos a la base de datos de IBM Data Server mediante una sentencia INSERT, puede insertar los objetos grandes directamente desde un archivo en el servidor PHP.

Antes de empezar

Obtener un recurso de conexión invocando una de las funciones de conexión en la API ibm_db2.

Procedimiento

Para insertar un objeto grande en la base de datos directamente desde un archivo:

1. Llame a la función db2_prepare para preparar una sentencia INSERT con un marcador de parámetro que represente la columna del objeto grande.
2. Establezca el valor de una variable de PHP en la vía de acceso y el nombre del archivo que contenga los datos del objeto grande. La vía de acceso puede ser relativa o absoluta y está sujeta a los permisos de acceso del archivo ejecutable de PHP.
3. Llame a la función db2_bind_param para vincular el marcador de parámetro a la variable. El tercer argumento de esta función es una serie que representa el nombre de la variable de PHP que conserva el nombre y la vía de acceso del archivo. El cuarto argumento es DB2_PARAM_FILE, que indica a la extensión ibm_db2 que recupere los datos de un archivo.
4. Llame a la función db2_execute para emitir la sentencia INSERT.

Ejemplo

Insertar un objeto grande en la base de datos.

```
$stmt = db2_prepare($conn, "INSERT INTO animal_pictures(picture) VALUES (?)");

$picture = "/opt/albums/spook/grooming.jpg";
$rc = db2_bind_param($stmt, 1, "picture", DB2_PARAM_FILE);
$rc = db2_execute($stmt);
```

Lectura de conjuntos de resultados de consultas

Captación de columnas o columnas de conjuntos de resultados en PHP (ibm_db2):

Después de ejecutar una sentencia que devuelve uno o más conjuntos de resultados, utilice una de las funciones disponibles en la API ibm_db2 para iterar por las filas devueltas de cada conjunto de resultados. Si el conjunto de resultados incluye columnas que contienen datos extremadamente grandes, puede recuperar los datos de columna en columna para evitar que se utilice demasiada memoria.

Antes de empezar

Debe tener un recurso de sentencia que haya devuelto la función db2_exec o db2_execute que tenga uno o varios conjuntos de resultados asociados.

Procedimiento

Para captar datos de un conjunto de resultados:

1. Capte los datos de un conjunto de resultados llamando a una de las funciones de captación.

Tabla 4. Funciones de captación de ibm_db2

Función	Descripción
db2_fetch_array	Devuelve una matriz, indexada por posición de columna, que representa una fila en un conjunto de resultados. Las columnas están indexadas en 0.
db2_fetch_assoc	Devuelve una matriz, indexada por nombre de columna, que representa una fila en un conjunto de resultados.
db2_fetch_both	Devuelve una matriz, indexada tanto por nombre de columna como por posición, que representa una fila en un conjunto de resultados.
db2_fetch_row	Establece el puntero del conjunto de resultados en la siguiente fila o la fila solicitada. Utilice esta función para iterar por un conjunto de resultados.
db2_fetch_object	Devuelve un objeto con propiedades que representa las columnas de la fila captada. Las propiedades del objeto se correlacionan con los nombres de las columnas del conjunto de resultados.

Estas funciones aceptan los argumentos listados:

sentencia
: Un recurso de sentencia válido.

número_fila
: El número de la fila que desea recuperar del conjunto de resultados. La numeración de las filas empieza por 1. Especifique un valor para este parámetro opcional si ha solicitado un cursor desplazable al llamar a la

función db2_exec o db2_prepare. Con el cursor de sólo avance por omisión, cada llamada a un método de captación devuelve la fila siguiente del conjunto de resultados.

2. Opcional: Si ha llamado a la función db2_fetch_row, para cada iteración en el conjunto de resultados, recupere un valor de la columna especificada llamando a la función db2_result. Puede especificar la columna pasando un entero que represente la posición de la columna en la fila (empezando por 0), o una serie que represente el nombre de la columna.
3. Continúe captando filas hasta que la función de captación devuelva False, lo que indicará que habrá llegado al final del conjunto de resultados.

   Para obtener más información sobre la API ibm_db2, consulte http://www.php.net/docs.php.

Ejemplo

Ejemplo 1: Captación de filas de un conjunto de resultados llamando a la función db2_fetch_object

```
<?php
$conn = db2_connect("sample", "db2inst1", "");
$sql = 'SELECT FIRSTNME, LASTNAME FROM EMPLOYEE WHERE EMPNO = ?';
$stmt = db2_prepare($conn, $sql);
db2_execute($stmt, array('000010'));
while ($row = db2_fetch_object($stmt)) {
  print "Name:
  {$row->FIRSTNME} {$row->LASTNAME}

  ";
  }
db2_close($conn);
?>
```

Ejemplo 2: Captación de filas de un conjunto de resultados llamando a la función db2_fetch_row

```
<?php
$conn = db2_connect("sample", "db2inst1", "");
$sql = 'SELECT FIRSTNME, LASTNAME FROM EMPLOYEE WHERE EMPNO = ?';
$stmt = db2_prepare($conn, $sql);
db2_execute($stmt, array('000010'));
while (db2_fetch_row($stmt)) {
  $fname = db2_result($stmt, 0);
  $lname = db2_result($stmt, 'LASTNAME');
  print "
  Name: $fname $lname

  ";
  }
db2_close($conn);
?>
```

Ejemplo 3: Captación de filas de un conjunto de resultados llamando a la función db2_fetch_both

```
<?php
$conn = db2_connect("sample", "db2inst1", "");
$sql = 'SELECT FIRSTNME, LASTNAME FROM EMPLOYEE WHERE EMPNO = ?';
$stmt = db2_prepare($conn, $sql);
db2_execute($stmt, array('000010'));
while ($row = db2_fetch_both($stmt)) {
  print "
  NAME: $row[0] $row[1]

  ";
```

```
  print "
  NAME: " . $row['FIRSTNME'] . " " . $row['LASTNAME'] . "

  ";
  }
db2_close($conn);
?>
```

Qué hacer a continuación

Cuando esté listo para cerrar la conexión con la base de datos, llame a la función db2_close. Si intenta cerrar una conexión persistente creada con db2_pconnect, la petición de cierre devuelve TRUE, y la conexión del cliente de servidor de datos de IBM permanecerá disponible para el siguiente interlocutor.

Captación de objetos grandes en PHP (ibm_db2):

Al captar un objeto grande desde un conjunto de resultados, en lugar de tratar el objeto grande como una serie de PHP, puede guardar los recursos del sistema mediante la captación de objetos grandes directamente en un archivo de su servidor PHP.

Antes de empezar

Obtener un recurso de conexión invocando una de las funciones de conexión en la API ibm_db2.

Procedimiento

Para captar un objeto grande desde la base de datos directamente en un archivo:

1. Cree una variable PHP que represente una corriente. Por ejemplo, asigne a una variable el valor de retorno de una llamada a la función fopen.
2. Cree una sentencia SELECT llamando a la función db2_prepare.
3. Vincule la columna de salida del objeto grande a la variable de PHP que representa a la secuencia llamando a la función db2_bind_param. El tercer argumento de esta función es una serie que representa el nombre de la variable de PHP que conserva el nombre y la vía de acceso del archivo. El cuarto argumento es DB2_PARAM_FILE, que indica a la extensión ibm_db2 que grabe los datos en un archivo.
4. Emita la sentencia de SQL llamando a la función db2_execute.
5. Recupere la fila siguiente del conjunto de resultados llamando a una función de captación de ibm_db2 (por ejemplo, db2_fetch_object).

   Para obtener más información sobre la API ibm_db2, consulte http://www.php.net/docs.php.

Ejemplo

Captar un objeto grande de la base de datos.

```
$stmt = db2_prepare($conn, "SELECT name, picture FROM animal_pictures");
$picture = fopen("/opt/albums/spook/grooming.jpg", "wb");
$rc = db2_bind_param($stmt, 1, "nickname", DB2_CHAR, 32);
$rc = db2_bind_param($stmt, 2, "picture", DB2_PARAM_FILE);
$rc = db2_execute($stmt);
$rc = db2_fetch_object($stmt);
```

Invocación de procedimientos almacenados en PHP (ibm_db2)

Para llamar a un procedimiento almacenado desde una aplicación PHP, prepare y ejecute una sentencia de SQL CALL. El procedimiento que invoca puede incluir parámetros de entrada (IN), parámetro de salida (OUT) y parámetros de entrada y salida.

Antes de empezar

Obtener un recurso de conexión invocando una de las funciones de conexión en la API ibm_db2. Consulte "Conexión a una base de datos de IBM Data Server en PHP (ibm_db2)" en la página 16.

Procedimiento

Para llamar a un procedimiento almacenado:

1. Llame a la función db2_prepare, pasando los argumentos listados:

   conexión
   Un recurso de conexión de base de datos válido que devuelve db2_connect o db2_pconnect.

   sentencia
   Una serie que contiene la sentencia de SQL CALL, incluidos los marcadores de parámetro (?), para los parámetros de entrada o salida

   opciones
   Opcional: Una matriz asociativa que especifica el tipo de cursor que debe devolverse para conjuntos de resultados. Puede utilizar este parámetro para solicitar un cursor desplazable en los servidores de bases de datos que admitan este tipo de cursor. Por omisión, se devuelve un cursor de sólo avance.

2. Por cada marcador de parámetro en la sentencia CALL, llame a la función db2_bind_param, proporcionando los argumentos listados:

   sentencia
   La sentencia preparada que devuelve la llamada a la función db2_prepare.

   número_parámetro
   Un entero que representa la posición del marcador de parámetro en la sentencia de SQL.

   nombre-variable
   El nombre de la variable PHP que se vinculará al parámetro especificado en *número_parámetro*.

   tipo_parámetro
   Una constante que especifica si se vinculará la variable PHP al parámetro de SQL como un parámetro de entrada (DB2_PARAM_INPUT), un parámetro de salida (DB2_PARAM_OUTPUT) o un parámetro que acepta una entrada y devuelve una salida (DB2_PARAM_INPUT_OUTPUT).

   Este paso vincula cada marcador de parámetro al nombre de una variable PHP que contendrá la salida.

3. Llame a la función db2_execute, pasando la sentencia preparada como un argumento.

   Para obtener más información sobre la API ibm_db2, consulte http://www.php.net/docs.php.

Ejemplo

Prepare y ejecute una sentencia de SQL CALL.

```
$sql = 'CALL match_animal(?, ?)';
$stmt = db2_prepare($conn, $sql);

$second_name = "Rickety Ride";
$weight = 0;

db2_bind_param($stmt, 1, "second_name", DB2_PARAM_INOUT);
db2_bind_param($stmt, 2, "weight", DB2_PARAM_OUT);

print "Values of bound parameters _before_ CALL:\n";
print "  1: {$second_name} 2: {$weight}\n";

db2_execute($stmt);

print "Values of bound parameters _after_ CALL:\n";
print "  1: {$second_name} 2: {$weight}\n";
```

Qué hacer a continuación

Si la llamada al procedimiento devuelve uno o varios conjuntos de resultados, puede empezar a captar filas del recurso de sentencia.

Recuperación de varios conjuntos de resultados de un procedimiento almacenado en PHP (ibm_db2):

Cuando una única llamada a un procedimiento almacenado devuelve más de un conjunto de resultados, puede utilizar la función db2_next_result de la API ibm_db2 para recuperar los conjuntos de resultados.

Antes de empezar

Debe tener un recurso de sentencia que haya devuelto la función db2_exec o db2_execute que tenga varios conjuntos de resultados.

Procedimiento

Para recuperar varios conjuntos de resultados:

1. Capte filas del primer conjunto de resultados devuelto por el procedimiento mediante la llamada a una de las funciones de captación ibm_db2, proporcionando el recurso de sentencia como argumento. (El primer conjunto de resultados que se devuelve del procedimiento está asociado con el recurso de sentencia).

Tabla 5. Funciones de captación de ibm_db2

Función	Descripción
db2_fetch_array	Devuelve una matriz, indexada por posición de columna, que representa una fila en un conjunto de resultados. Las columnas están indexadas en 0.
db2_fetch_assoc	Devuelve una matriz, indexada por nombre de columna, que representa una fila en un conjunto de resultados.

Tabla 5. Funciones de captación de ibm_db2 (continuación)

Función	Descripción
db2_fetch_both	Devuelve una matriz, indexada tanto por nombre de columna como por posición, que representa una fila en un conjunto de resultados.
db2_fetch_row	Establece el puntero del conjunto de resultados en la siguiente fila o la fila solicitada. Utilice esta función para iterar por un conjunto de resultados.
db2_fetch_object	Devuelve un objeto con propiedades que representa las columnas de la fila captada. Las propiedades del objeto se correlacionan con los nombres de las columnas del conjunto de resultados.

2. Recupere los conjuntos de resultados siguientes proporcionando el recurso de sentencia original como primer argumento a la función db2_next_result. Puede captar filas del recurso de sentencia hasta que no haya disponibles más filas en el conjunto de resultados.

   La función db2_next_result devuelve False cuando no hay más conjuntos de resultados disponibles o el procedimiento no ha devuelto ningún conjunto de resultados.

   Para obtener más información sobre la API ibm_db2, consulte http://www.php.net/docs.php.

Ejemplo

Recuperar varios conjuntos de resultados de un procedimiento almacenado.

```
$stmt = db2_exec($conn, 'CALL multiResults()');

print "Fetching first result set\n";
while ($row = db2_fetch_array($stmt)) {
    // Trabajar con fila
}

print "\nFetching second result set\n";
$result_2 = db2_next_result($stmt);
if ($result_2) {
  while ($row = db2_fetch_array($result_2)) {
    // Trabajar con fila
  }
}

print "\nFetching third result set\n";
$result_3 = db2_next_result($stmt);
if ($result_3) {
  while ($row = db2_fetch_array($result_3)) {
    // Trabajar con fila
  }
}
```

Qué hacer a continuación

Cuando esté listo para cerrar la conexión con la base de datos, llame a la función db2_close. Si intenta cerrar una conexión persistente creada con db2_pconnect, la petición de cierre devuelve TRUE, y la conexión persistente del cliente de servidor de datos de IBM permanecerá disponible para el siguiente interlocutor.

Modalidades de confirmación en aplicaciones PHP (ibm_db2)

Puede controlar cómo se confirman los grupos de sentencias de SQL especificando una modalidad de confirmación para un recurso de conexión. La extensión ibm_db2 admite dos modalidades de confirmación: la confirmación automática y la confirmación manual.

Debe utilizar un recurso de conexión regular devuelto por la función db2_connect para controlar las transacciones de base de datos en PHP. Las conexiones persistentes siempre utilizan la modalidad Autocommit.

Modalidad de confirmación automática

En modalidad de confirmación automática, cada sentencia de SQL es una transacción completa, que se confirma automáticamente. La modalidad Autocommit le ayuda a evitar problemas de escalas de bloqueo que puedan obstaculizar el rendimiento de aplicaciones Web muy escalables. Por omisión, la extensión ibm_db2 abre cada conexión en modalidad Autocommit.

Puede activar la modalidad de confirmación automática tras inhabilitarla llamando a `db2_autocommit($con, DB2_AUTOCOMMIT_ON)`, donde *con* es un recurso de conexión válido.

Llamar a la función db2_autocommit puede afectar al rendimiento de sus scripts de PHP, porque requiere comunicación adicional entre PHP y el sistema de gestión de bases de datos.

Modalidad de confirmación manual

En modalidad de confirmación manual, la transacción finaliza al llamar a la función db2_commit o db2_rollback. Esto significa que todas las sentencias ejecutadas en la misma conexión entre el inicio de una transacción y la llamada a la función de confirmación o retrotracción se tratan como una única transacción.

La modalidad de confirmación manual es útil si se necesita retrotraer una transacción que contiene una o varias sentencias de SQL. Si emite sentencias de SQL en una transacción y el script finaliza sin confirmar o retrotraer explícitamente la transacción, la extensión ibm_db2 retrotrae automáticamente cualquier trabajo realizado en la transacción.

Puede inhabilitar la modalidad de confirmación automática al crear una conexión de base de datos utilizando la configuración "AUTOCOMMIT" => DB2_AUTOCOMMIT_OFF en la matriz de opciones db2_connect. También puede desactivar la modalidad de confirmación automática para un recurso de conexión existente llamando a `db2_autocommit($con, DB2_AUTOCOMMIT_OFF)`, donde *con* es un recurso de conexión válido.

Para obtener más información sobre la API ibm_db2, consulte http://www.php.net/docs.php.

Ejemplo

Finalizar la transacción cuando se llama a db2_commit o db2_rollback.

```
$conn = db2_connect('SAMPLE', 'db2inst1', 'ibmdb2', array(
  'AUTOCOMMIT' => DB2_AUTOCOMMIT_ON));

// Emitir una o más sentencias SQL en la transacción
$result = db2_exec($conn, 'DELETE FROM TABLE employee');
if ($result === FALSE) {
  print '<p>No se ha podido completar la transacción</p>';
  db2_rollback($conn);
```

```
}
else {
  print '<p>Successfully completed transaction!</p>';
  db2_commit($conn);
}
```

Funciones de manejo de errores en aplicaciones PHP (ibm_db2)

En ocasiones se producen problemas al intentar conectarse a una base de datos o emitir una sentencia de SQL. El nombre de usuario o la contraseña pueden ser incorrectos, puede haberse escrito mal un nombre de tabla o columna o la sentencia de SQL puede no ser válida. La API ibm_db2 proporciona funciones de manejo de errores para ayudarle a recuperarse sin problemas de estas situaciones.

Errores de conexión

Utilice una de las funciones listadas para recuperar información de diagnóstico si falla un intento de conexión.

Tabla 6. Funciones de ibm_db2 para manejar errores de conexión

Función	Descripción
db2_conn_error	Recupera el SQLSTATE devuelto por el último intento de conexión.
db2_conn_errormsg	Recupera un mensaje de error descriptivo adecuado a la anotación cronológica de errores de la aplicación

Errores de SQL

Utilice una de las funciones listadas para recuperar información de diagnóstico si falla un intento de preparar o ejecutar una sentencia de SQL o de captar un resultado de un conjunto de resultados.

Tabla 7. Funciones de ibm_db2 para manejar errores de SQL

Función	Descripción
db2_stmt_error	Recupera el SQLSTATE devuelto por el último intento de preparar o ejecutar una sentencia de SQL o de captar un resultado de un conjunto de resultados.
db2_stmt_errormsg	Recupera un mensaje de error descriptivo adecuado a la anotación cronológica de errores de la aplicación

Para obtener más información sobre la API ibm_db2, consulte http://www.php.net/docs.php.

Consejo: Para evitar las vulneraciones de la seguridad que podrían derivar de la visualización directa del SQLSTATE en bruto devuelto por la base de datos y ofrecer una mejor experiencia general del usuario en la aplicación web, utilice una estructura de conmutación para recuperar de los estados de errores conocidos o devolver mensajes de errores personalizados. Por ejemplo:

```
switch($this->state):
    case '22001':
        // Más datos de los permitidos para la columna definida
        $message = "Ha entrado demasiados caracteres para este valor.";
        break;
```

Ejemplo

Ejemplo 1: Manejo de errores de conexión

```
$connection = db2_connect($database, $user, $password);
if (!$connection) {
    $this->state = db2_conn_error();
    return false;
}
```

Ejemplo 2: Manejo de errores de SQL

```
$stmt = db2_prepare($connection, "DELETE FROM employee
WHERE firstnme = ?");
if (!$stmt) {
    $this->state = db2_stmt_error();
    return false;
}
```

Ejemplo 3: Manejo de errores de SQL que derivan de la ejecución de sentencias preparadas

```
$success = db2_execute($stmt, array('Dan');
if (!$success) {
    $this->state = db2_stmt_error($stmt);
    return $false;
}
```

Funciones de recuperación de metadatos de base de datos en PHP (ibm_db2)

Puede usar las funciones de la API ibm_db2 para recuperar metadatos de las bases de datos atendidas por DB2 Database para Linux, UNIX y Windows, IBM Cloudscape y, a través de DB2 Connect, DB2 para z/OS y DB2 para i.

Algunas clases de aplicaciones como, por ejemplo, las interfaces de administración, deben reflejar de forma dinámica la estructura y los objetos SQL contenidos en bases de datos arbitrarias. Un enfoque para recuperar metadatos sobre una base de datos consiste en emitir sentencias SELECT directamente contra las tablas de catálogos del sistema; sin embargo, el esquema de las tablas de catálogos del sistema podría variar entre versiones de DB2 o el esquema de las tablas de catálogos del sistema en DB2 Database para Linux, UNIX y Windows puede diferir del esquema de las tablas de catálogos del sistema en DB2 para z/OS. En lugar de conservar estas diferencias en el código de su aplicación, lo que conlleva un esfuerzo considerable, puede utilizar las funciones de PHP disponibles en la extensión ibm_db2 para recuperar metadatos de la base de datos.

Antes de llamar a estas funciones, debe instalar el entorno PHP y tener un recurso de conexión que haya devuelto la función db2_connect o db2_pconnect.

Importante: La llamada a funciones de metadatos utiliza una cantidad de espacio considerable. Si es posible, modifique los resultados de las llamadas para su uso en llamadas posteriores.

Tabla 8. Funciones de recuperación de metadatos de ibm_db2

Función	Descripción
db2_client_info	Devuelve un objeto de sólo lectura con información acerca del cliente de servidor de datos de IBM.

Tabla 8. Funciones de recuperación de metadatos de ibm_db2 (continuación)

Función	Descripción
db2_column_privileges	Devuelve un conjunto de resultados que enumera las columnas y los privilegios asociados para una tabla
db2_columns	Devuelve un conjunto de resultados que enumera las columnas y los metadatos asociados para una tabla
db2_foreign_keys	Devuelve un conjunto de resultados que enumera las claves foráneas para una tabla
db2_primary_keys	Devuelve un conjunto de resultados que enumera las claves primarias para una tabla
db2_procedure_columns	Devuelve un conjunto de resultados que enumera los parámetros para uno o varios procedimientos almacenados
db2_procedures	Devuelve una lista de resultados que enumera los procedimientos almacenados registrados en la base de datos
db2_server_info	Devuelve un objeto de sólo lectura con información sobre la configuración y el software del sistema de gestión de bases de datos
db2_special_columns	Devuelve un conjunto de resultados que enumera los identificadores de fila exclusivos para una tabla
db2_statistics	Devuelve un conjunto de resultados que enumera los índices y las estadísticas para una tabla
db2_table_privileges	Devuelve un conjunto de resultados que enumera las tablas y los privilegios asociados en una base de datos

La mayoría de las funciones de recuperación de metadatos de base de datos que ofrece ibm_db2 devuelven conjuntos de resultados con columnas definidas para cada función. Para recuperar filas de los conjuntos de resultados, utilice las funciones de ibm_db2 disponibles para tal fin.

Las funciones db2_client_info y db2_server_info devuelven directamente un único objeto con propiedades de sólo lectura. Las propiedades de estos objetos pueden utilizarse para crear una aplicación que tenga un comportamiento distinto en función del sistema de gestión de bases de datos al que se conecte. Por ejemplo, en lugar de codificar un límite del denominador común inferior para todos los sistemas de gestión de bases de datos posibles, una aplicación de administración de bases de datos basada en Web creada en la extensión ibm_db2 podría utilizar la propiedad `db2_server_info()->MAX_COL_NAME_LEN` para visualizar de forma dinámica los campos de texto para denominar las columnas con longitudes máximas que se correspondan con la longitud máxima de los nombres de columna en el sistema de gestión de bases de datos al que está conectado.

Para obtener más información sobre la API ibm_db2, consulte http://www.php.net/docs.php.

Ejemplo

Ejemplo 1: Visualizar una lista de las columnas y los privilegios asociados para una tabla

```
<?php
$conn = db2_connect('sample', 'db2inst1', 'ibmdb2');

if($conn) {
 $stmt = db2_column_privileges($conn, NULL, NULL, 'DEPARTMENT');
 $row = db2_fetch_array($stmt);
 print $row[2] . "\n";
 print $row[3] . "\n";
 print $row[7];
 db2_close($conn);
}
else {
 echo db2_conn_errormsg();
 printf("Connection failed\n\n");
}
?>
```

Ejemplo 2: Visualizar una lista de las claves primarias para una tabla

```
<?php
$conn = db2_connect('sample', 'db2inst1', 'ibmdb2');

if($conn) {
 $stmt = db2_primary_keys($conn, NULL, NULL, 'DEPARTMENT');
 while ($row = db2_fetch_array($stmt)) {
  echo "TABLE_NAME:\t" . $row[2] . "\n";
  echo "COLUMN_NAME:\t" . $row[3] . "\n";
  echo "KEY_SEQ:\t" . $row[4] . "\n";
 }

 db2_close($conn);
}
else {
 echo db2_conn_errormsg();
 printf("Connection failed\n\n");
}
?>
```

Ejemplo 3: Visualizar una lista de parámetros para uno o varios procedimientos almacenados

```
<?php
$conn = db2_connect('sample', 'db2inst1', 'ibmdb2');

if($conn) {
 $stmt = db2_procedures($conn, NULL, 'SYS%', '%%');

 $row = db2_fetch_assoc($stmt);
 var_dump($row);

 db2_close($conn);
}
else {
 echo "Connection failed.\n";
}
?>
```

Ejemplo 4: Visualizar una lista de los índices y las estadísticas para una tabla

```
<?php
$conn = db2_connect('sample', 'db2inst1', 'ibmdb2');
```

```
if($conn) {
 echo "Test DEPARTMENT table:\n";
 $result = db2_statistics($conn, NULL, NULL, "EMPLOYEE", 1);
 while ($row = db2_fetch_assoc($result)) {
  var_dump($row);
 }

 echo "Test non-existent table:\n";
 $result = db2_statistics($conn,NULL,NULL,"NON_EXISTENT_TABLE",1);
 $row = db2_fetch_array($result);
 if ($row) {
  echo "Non-Empty\n";
 } else {
  echo "Empty\n";
 }

 db2_close($conn);
}
else {
 echo 'no connection: ' . db2_conn_errormsg();
}
?>
```

Ejemplo 5: Visualizar una lista de tablas y los privilegios asociados de la base de datos

```
<?php
$conn = db2_connect('sample', 'db2inst1', 'ibmdb2');

if($conn) {
 $stmt = db2_table_privileges($conn, NULL, "%%", "DEPARTMENT");
 while ($row = db2_fetch_assoc($stmt)) {
  var_dump($row);
 }
 db2_close($conn);
}
else {
 echo db2_conn_errormsg();
 printf("Connection failed\n\n");
}
?>
```

Desarrollo de aplicaciones en PHP (PDO)

La extensión PDO_IBM proporciona una serie de útiles funciones de PHP para acceder a los datos y manipularlos a través de una interfaz de base de datos orientada a objetos estándar incorporada en PHP 5.1. La extensión incluye funciones para conectarse a una base de datos, ejecutar y preparar sentencias de SQL, recuperar filas de conjuntos de resultados, gestionar transacciones, llamar a procedimientos almacenados, gestionar errores y recuperar metadatos.

Conexión a una base de datos de IBM Data Server con PHP (PDO)

Antes de poder emitir sentencias de SQL para crear, actualizar, suprimir o recuperar datos, debe conectarse a una base de datos. Puede utilizar la interfaz de objetos de datos PHP (PDO) para PHP para conectar con una base de datos de IBM Data Server mediante una conexión catalogada o una conexión TCP/IP directa. Para mejorar el rendimiento, también puede crear una conexión persistente.

Antes de empezar

Debe configurar el entorno de PHP 5.1 (o posterior) en el sistema y habilitar las extensiones PDO y PDO_IBM.

Acerca de esta tarea

Este procedimiento devuelve un objeto de conexión a una base de datos de IBM Data Server. Esta conexión continúa abierta hasta que establece el objeto de PDO en NULL o hasta que finaliza el script de PHP.

Procedimiento

Para conectar con una base de datos de IBM Data Server:

1. Cree una conexión a la base de datos llamando al constructor de PDO dentro de un bloque `try{}`. Pase un valor *DSN* que especifique `ibm:` para la extensión PDO_IBM, seguido de un nombre de base de datos catalogada o de toda una serie de conexión de base de datos para una conexión TCP/IP directa.
   - (Windows): Por omisión, la extensión PDO_IBM utiliza el agrupamiento de conexiones para minimizar los recursos de conexión y mejorar el rendimiento de las conexiones.
   - (Linux y UNIX): Para crear una conexión persistente, proporcione `array(PDO::ATTR_PERSISTENT => TRUE)` como el (cuarto) argumento *opciones_controlador* al constructor de PDO.
2. Opcional: Establezca las opciones de manejo de errores para la conexión PDO en el cuarto argumento al constructor de PDO:
   - Por omisión, PDO establece un mensaje de error que se puede recuperar mediante `PDO::errorInfo()` y un SQLCODE que se puede recuperar mediante `PDO::errorCode()` cuando se produce cualquier error; para solicitar esta modalidad de forma explícita, establezca `PDO::ATTR_ERRMODE => PDO::ERRMODE_SILENT`
   - Para emitir un `E_WARNING` de PHP cuando se produzca cualquier error, además de establecer el mensaje de error y SQLCODE, establezca `PDO::ATTR_ERRMODE => PDO::ERRMODE_WARNING`
   - Para emitir una excepción de PHP cuando se produzca cualquier error, establezca `PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION`
3. Detecte cualquier excepción emitida por el bloque `try{}` en un bloque `catch {}` correspondiente.

   Para obtener más información sobre la API de PDO, consulte http://php.net/manual/en/book.pdo.php.

Ejemplo

Conexión a una base de datos de IBM Data Server en una conexión persistente.

```
try {
  $connection = new PDO("ibm:SAMPLE", "db2inst1", "ibmdb2", array(
    PDO::ATTR_PERSISTENT => TRUE,
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION)
  );
}
catch (Exception $e) {
  echo($e->getMessage());
}
```

Qué hacer a continuación

A continuación, prepare y ejecute las sentencias de SQL.

Ejecución de sentencias de SQL en PHP (PDO)

Después de conectarse a una base de datos, utilice los métodos disponibles en la API de PDO para preparar y ejecutar sentencias de SQL. Las sentencias de SQL pueden contener texto estático o marcadores de parámetros que representan la entrada de variables.

Ejecución de una única sentencia de SQL en PHP (PDO):

Para preparar y ejecutar una única sentencia de SQL que no acepta parámetros de entrada, utilice el método PDO::exec o PDO::query. Emplee el método PDO::exec para ejecutar una sentencia que no devuelve conjuntos de resultados. Utilice el método PDO::query para ejecutar una sentencia que devuelve uno o varios conjuntos de resultados.

Antes de empezar

Importante: Para evitar amenazas a la seguridad por ataques de inyección de SQL, utilice el método PDO::exec o PDO::query sólo para ejecutar sentencias de SQL compuestas de series estáticas. La interpolación de variables de PHP que representan la entrada del usuario en la sentencia de SQL puede exponer la aplicación a ataques de inyección de SQL.

Obtener un objeto de conexión mediante una llamada al constructor de PDO.

Procedimiento

Para preparar y ejecutar una única sentencia de SQL que no acepta parámetros de entrada, llame a uno de los métodos listados:

- Para ejecutar una sentencia de SQL que no devuelve conjuntos de resultados, llame al método PDO::exec en el objeto de conexión PDO, proporcionando una serie que contenga la sentencia de SQL. Por ejemplo, un uso típico de PDO::exec es establecer el esquema por omisión para la aplicación en un archivo de inclusión común o clase base.

  Si la sentencia de SQL se completa con éxito (inserta, modifica o suprime filas satisfactoriamente), el método PDO::execdevuelve un valor entero que representa el número de filas que se han insertado, modificado o suprimido.

  Para determinar si el método PDO::exec falló (devolvió FALSE o 0), utilice el operador === para probar estrictamente el valor devuelto contra FALSE.
- Para ejecutar una sentencia de SQL que devuelve uno o varios conjuntos de resultados, llame al método PDO::query en el objeto de conexión PDO, proporcionando una serie que contenga la sentencia de SQL. Por ejemplo, es posible que desee llamar a este método para ejecutar una sentencia SELECT estática.

  Si la llamada de método se realiza con éxito, devuelve un recurso PDOStatement que puede utilizar en llamadas de método posteriores.

Si la llamada de método falla (devuelve FALSE), puede utilizar el método PDO::errorCode y PDO::errorInfo para recuperar información de diagnóstico sobre el error.
Para obtener más información sobre la API de PDO, consulte http://php.net/manual/en/book.pdo.php.

Ejemplo

Ejemplo 1: Llamar al método PDO::exec para determinar el esquema por omisión de su aplicación

```
$conn = new PDO('ibm:SAMPLE', 'db2inst1', 'ibmdb2');
$result = $conn->exec('SET SCHEMA myapp');
if ($result === FALSE) {
  print "Failed to set schema: " . $conn->errorMsg();
}
```

Ejemplo 2: Llamar al método PDO::query para emitir una sentencia de SQL SELECT

```
$conn = new PDO('ibm:SAMPLE', 'db2inst1', 'ibmdb2');
$result = $conn->query('SELECT firstnme, lastname FROM employee');
if (!$result) {
  print "<p>Could not retrieve employee list: " . $conn->errorMsg(). "</p>";
}
while ($row = $conn->fetch()) {
  print "<p>Name: {$row[0] $row[1]}</p>";
}
```

Qué hacer a continuación

Si llamó al método PDO::query para crear un objeto PDOStatement, puede empezar a recuperar filas del objeto llamando al método PDOStatement::fetch o PDOStatement::fetchAll.

Preparación y ejecución de sentencias de SQL en PHP (PDO):

Para preparar y ejecutar una sentencia de SQL que incluya la entrada de variables, utilice los métodos PDO::prepare, PDOStatement::bindParam y PDOStatement::execute. Preparar una sentencia mejora el rendimiento porque el servidor de bases de datos crea un plan de acceso optimizado para la recuperación de datos que puede reutilizar si se vuelve a ejecutar la sentencia.

Antes de empezar

Obtener un objeto de conexión mediante una llamada al constructor de PDO. Consulte “Conexión a una base de datos de IBM Data Server con PHP (PDO)” en la página 34.

Procedimiento

Para preparar y ejecutar una sentencia de SQL que incluye marcadores de parámetro:

1. Llame al método PDO::prepare, pasando los argumentos listados:

   sentencia
   Una serie que contiene la sentencia de SQL, incluidos los signos de interrogación (?), o variables con nombre (`:name`), como marcadores de parámetro para cualquier valor de columna o predicado que requiera entrada de variables. Sólo puede utilizar marcadores de parámetro como lugar reservado para los valores de columna o predicado. El compilador de SQL no puede crear un plan de acceso para una sentencia que usa marcadores de parámetro en lugar de nombres de columna, nombres de tabla u otros identificadores de SQL. No puede utilizar marcadores de parámetro con signo de interrogación (?) y marcadores de parámetro con nombre (`:name`) en la misma sentencia de SQL.

opciones_controlador
Opcional: Una matriz que contiene opciones de sentencias:

PDO::ATTR_CURSOR
Esta opción establece el tipo de cursor que PDO devuelve para los conjuntos de resultados. Por omisión, PDO devuelve un cursor de sólo avance (PDO::CURSOR_FWDONLY), que devuelve la fila siguiente de un conjunto de resultados para cada llamada a `PDOStatement::fetch()`. Puede establecer este parámetro en PDO::CURSOR_SCROLL para solicitar un cursor desplazable.

Si la llamada de función tiene éxito, devuelve un objeto PDOStatement que puede utilizar en llamadas de método posteriores relacionadas con esta consulta.

Si la llamada de función falla (devuelve `False`), puede utilizar el método PDO::errorCode o PDO::errorInfo para recuperar información de diagnóstico sobre el error.

2. Opcional: Por cada marcador de parámetro en la serie de SQL, llame al método PDOStatement::bindParam, proporcionando los argumentos listados. Vincular valores de entrada a marcadores de parámetros garantiza que cada valor de entrada se trate como un único parámetro, evitando ataques de inyección de SQL contra la aplicación.

   parámetro
   Un identificador de parámetro. Para los marcadores de parámetro con signo de interrogación (?)se trata de un entero que representa la posición indexada 1 del parámetro en la sentencia de SQL. Para marcadores de parámetro con nombre (`:name`), se trata de una serie que representa el nombre de parámetro.

   variable
   Valor que se debe utilizar en lugar del marcador de parámetro

3. Llame al método PDOStatement::execute, opcionalmente pasando una matriz que contiene los valores que se deben utilizar en lugar de los marcadores de parámetro, bien en orden para marcadores de parámetro de signo de interrogación, bien como una matriz asociativa `:name => value` para marcadores de parámetro con nombre.

   Para obtener más información sobre la API de PDO, consulte http://php.net/manual/en/book.pdo.php.

Ejemplo

Preparar y ejecutar una sentencia que incluye entrada de variables.

```
$sql = "SELECT firstnme, lastname FROM employee WHERE bonus > ? AND bonus < ?";
$stmt = $conn->prepare($sql);
if (!$stmt) {
  // Manejar los errores
}

// Vincular explícitamente los parámetros
$stmt->bindParam(1, $_POST['lower']);
$stmt->bindParam(2, $_POST['upper']);

$stmt->execute($stmt);

// Invocar de nuevo la sentencia utilizando parámetros vinculados dinámicamente
$stmt->execute($stmt, array($_POST['lower'], $_POST['upper']));
```

Qué hacer a continuación

Si la sentencia de SQL devuelve uno o varios conjuntos de resultados, puede empezar a captar filas del recurso de sentencia llamando al método PDOStatement::fetch o PDOStatement::fetchAll.

Inserción de objetos grandes en PHP (PDO):

Cuando inserta un objeto grande en la base de datos, en lugar de cargar todos los datos del objeto grande en una serie de PHP y entonces pasarlos a la base de datos de IBM Data Server mediante una sentencia INSERT, puede insertar los objetos grandes directamente desde un archivo en el servidor PHP.

Antes de empezar

Obtener un objeto de conexión mediante una llamada al constructor de PDO.

Procedimiento

Para insertar un objeto grande en la base de datos directamente desde un archivo:

1. Llame al método PDO::prepare para crear un objeto PDOStatement a partir de una sentencia INSERT con un marcador de parámetro que represente la columna del objeto grande.
2. Cree una variable de PHP que represente una secuencia (por ejemplo, asignando el valor devuelto por la función `fopen` a la variable).
3. Llame al método PDOStatement::bindParam, pasando los argumentos listados para vincular el marcador de parámetro a la variable de PHP que represente la secuencia de datos para el objeto grande:

   parámetro
   Un identificador de parámetro. Para los marcadores de parámetro con signo de interrogación (?)se trata de un entero que representa la posición indexada 1 del parámetro en la sentencia de SQL. Para marcadores de parámetro con nombre (`:name`), se trata de una serie que representa el nombre de parámetro.

   variable
   Valor que se debe utilizar en lugar del marcador de parámetro

   data_type
   La constante de PHP, `PDO::PARAM_LOB`, que indica a la extensión PDO que debe recuperar los datos de un archivo.
4. Llame al método PDOStatement::execute para emitir la sentencia INSERT.

Ejemplo

Insertar un objeto grande en la base de datos.

```
$stmt = $conn->prepare("INSERT INTO animal_pictures(picture) VALUES (?)");
$picture = fopen("/opt/albums/spook/grooming.jpg", "rb");
$stmt->bindParam(1, $picture, PDO::PARAM_LOB);
$stmt->execute();
```

Lectura de conjuntos de resultados de consultas

Captación de filas o columnas de conjuntos de resultados en PHP (PDO):

Después de ejecutar una sentencia que devuelve uno o más conjuntos de resultados, utilice uno de los métodos disponibles en la API de PDO para iterar por las filas devueltas. La API de PDO ofrece también métodos para captar una única columna de una o varias filas en el conjunto de resultados.

Antes de empezar

Debe tener un recurso de sentencia que haya devuelto el método PDO::query o PDOStatement::execute que tenga asociados uno o varios conjuntos de resultados.

Procedimiento

Para captar datos de un conjunto de resultados:

1. Capte los datos de un conjunto de resultados llamando a uno de los métodos de captación:
   - Para devolver una única fila de un conjunto de resultados como una matriz u objeto, llame al método PDOStatement::fetch.
   - Para devolver todas las filas del conjunto de resultados como una matriz de matrices u objetos, llame al método PDOStatement::fetchAll.

   Por omisión, PDO devuelve cada fila como una matriz indexada por el nombre de columna y la posición de columna indexada 0 en la fila. Para solicitar un estilo de devolución diferente, especifique una de las constantes `PDO::FETCH_*` como primer parámetro al llamar al método PDOStatement::fetch:

   `PDO::FETCH_ASSOC`
   Devuelve una matriz indexada por nombre de columna tal y como se ha devuelto en el conjunto de resultados.

   `PDO::FETCH_BOTH` (valor por omisión)
   Devuelve una matriz indexada tanto por nombre de columna como por número de columna indexado 0 en el conjunto de resultados

   `PDO::FETCH_BOUND`
   Devuelve TRUE y asigna los valores de las columnas del conjunto de resultados a las variables de PHP a las que estaban vinculadas con el método `PDOStatement::bindParam`.

   `PDO::FETCH_CLASS`
   Devuelve una nueva instancia de la clase solicitada, correlacionando las columnas del conjunto de resultados con las propiedades con nombre de la clase.

   `PDO::FETCH_INTO`
   Actualiza una instancia existente de la clase solicitada, correlacionando las columnas del conjunto de resultados con las propiedades de la clase.

   `PDO::FETCH_LAZY`
   Combina `PDO::FETCH_BOTH` y `PDO::FETCH_OBJ`, creando los nombres de variable de objeto conforme se accede a los mismos.

   `PDO::FETCH_NUM`
   Devuelve una matriz indexada por número de columna tal y como se ha devuelto en el conjunto de resultados, empezando en la columna 0.

   `PDO::FETCH_OBJ`
   Devuelve un objeto anónimo con nombres de propiedad que se corresponden con los nombres de columna devueltos en el conjunto de resultados.

Si ha solicitado un cursor desplazable al llamar al método PDO::query o PDOStatement::execute, puede pasar los parámetros opcionales listados que controlan qué filas se devuelven al emisor:

- Una de las constantes `PDO::FETCH_ORI_*` que representa la orientación de captación de la petición de captación:

  `PDO::FETCH_ORI_NEXT` (valor por omisión)
  Capta la siguiente fila del conjunto de resultados.

  `PDO::FETCH_ORI_PRIOR`
  Capta la fila anterior del conjunto de resultados.

  `PDO::FETCH_ORI_FIRST`
  Capta la primera fila del conjunto de resultados.

  `PDO::FETCH_ORI_LAST`
  Capta la última fila del conjunto de resultados.

  `PDO::FETCH_ORI_ABS`
  Capta la fila absoluta del conjunto de resultados. Requiere un entero positivo como tercer argumento para el método `PDOStatement::fetch`.

  `PDO::FETCH_ORI_REL`
  capta la fila relativa del conjunto de resultados. Requiere un entero positivo o negativo como tercer argumento para el método `PDOStatement::fetch`.

- Entero que solicita la fila absoluta o relativa del conjunto de resultados, correspondiente a la orientación de captación solicitada en el segundo argumento al método `PDOStatement::fetch`.

2. Opcional: Capte una única columna de una o varias filas de un conjunto de resultados llamando a uno de los métodos listados:
   - Para devolver una sola columna desde una sola fila del conjunto de resultados:

     Llame al método PDOStatement::fetchColumn especificando la columna que desee recuperar como el primer argumento del método. Los números de columna empiezan por el 0. Si no especifica ninguna columna, PDOStatement::fetchColumn devuelve la primera columna de la fila.
   - Para devolver una matriz que contenga una sola columna de todas las filas restantes del conjunto de resultados:

     Llame al método PDOStatement::fetchAll pasando la constante PDO::FETCH_COLUMN como el primer argumento y la columna que desea recuperar como el segundo argumento. Los números de columna empiezan por el 0. Si no especifica ninguna columna, al llamar a `PDOStatement::fetchAll(PDO::FETCH_COLUMN)` se devuelve la primera columna de la fila.

   Para obtener más información sobre la API de PDO, consulte http://php.net/manual/en/book.pdo.php.

Ejemplo

Devolver una matriz indexada por número de columna.

```
$stmt = $conn->query("SELECT firstnme, lastname FROM employee");
while ($row = $stmt->fetch(PDO::FETCH_NUM)) {
  print "Name: <p>{$row[0] $row[1]}</p>";
}
```

Qué hacer a continuación

Cuando esté listo para cerrar la conexión a la base de datos, establezca el objeto PDO en NULL. La conexión se cierra automáticamente cuando finaliza el script de PHP.

Captación de objetos grandes en PHP (PDO):

Al captar un objeto grande desde un conjunto de resultados, en lugar de tratar el objeto grande como una serie de PHP, puede guardar los recursos del sistema mediante la captación de objetos grandes directamente en un archivo de su servidor PHP.

Antes de empezar

Obtener un objeto de conexión mediante una llamada al constructor de PDO.

Procedimiento

Para captar un objeto grande desde la base de datos directamente en un archivo:

1. Cree una variable PHP que represente una corriente. Por ejemplo, asigne a una variable el valor de retorno de una llamada a la función fopen.
2. Cree un objeto PDOStatement a partir de una sentencia de SQL llamando al método PDO::prepare.
3. Vincule la columna de salida del objeto grande a la variable de PHP que representa a la secuencia llamando al método PDOStatement::bindColumn. El segundo argumento es una serie que representa el nombre de la variable de PHP que conserva el nombre y la vía de acceso del archivo. El tercer argumento es una constante de PHP, PDO::PARAM_LOB, que indica a la extensión PDO que grabe los datos en un archivo. Debe llamar al método PDOStatement::bindColumn para asignar una variable de PHP distinta para cada columna del conjunto de resultados.
4. Emita la sentencia de SQL llamando al método PDOStatement::execute.
5. Llame a PDOStatement::fetch(PDO::FETCH_BOUND) para recuperar la fila siguiente del conjunto de resultados, vinculando la salida de la columna a las variables de PHP que asoció cuando llamó al método PDOStatement::bindColumn.

Ejemplo

Capte un objeto grande desde la base de datos directamente en un archivo.

```
$stmt = $conn->prepare("SELECT name, picture FROM animal_pictures");
$picture = fopen("/opt/albums/spook/grooming.jpg", "wb");
$stmt->bindColumn('NAME', $nickname, PDO::PARAM_STR, 32);
$stmt->bindColumn('PICTURE', $picture, PDO::PARAM_LOB);
$stmt->execute();
$stmt->fetch(PDO::FETCH_BOUND);
```

Invocación de procedimientos almacenados en PHP (PDO)

Para llamar a un procedimiento almacenado desde una aplicación PHP, ejecute una sentencia de SQL CALL. El procedimiento que invoca puede incluir parámetros de entrada (IN), parámetro de salida (OUT) y parámetros de entrada y salida.

Antes de empezar

Obtener un objeto de conexión mediante una llamada al constructor de PDO.

Acerca de esta tarea

Este procedimiento prepara y ejecuta una sentencia de SQL CALL. Para obtener más información, consulte también el tema sobre preparación y ejecución de sentencias de SQL.

Procedimiento

Para llamar a un procedimiento almacenado:

1. Llame al método PDO::prepare para preparar una sentencia CALL con marcadores de parámetro que representen los parámetros OUT e INOUT.
2. Para cada marcado de parámetro de la sentencia CALL, llame al método PDOStatement::bindParam para vincular cada marcador de parámetro con el nombre de la variable de PHP que contendrá el valor de salida del parámetro después de emitir la sentencia CALL. Para los parámetros INOUT, el valor de la variable de PHP se pasa como el valor de entrada del parámetro cuando se emite la sentencia CALL.
   a. Establezca el tercer parámetro, *data_type*, en una de las constantes de PDO::PARAM_* que especifica el tipo de datos que se está vinculando:

      PDO::PARAM_NULL
      Representa el tipo de datos SQL NULL.

      PDO::PARAM_INT
      Representa los tipos enteros SQL.

      PDO::PARAM_LOB
      Representa los tipos de objetos grandes SQL.

      PDO::PARAM_STR
      Representa los tipos de datos de caracteres SQL.

      Para un parámetro INOUT, utilice el operador basado en bits OR para añadir `PDO::PARAM_INPUT_OUTPUT` al tipo de datos que se está vinculando.
   b. Establezca el cuarto parámetro, *longitud*, en la longitud máxima esperada del valor de salida.
3. Llame al método PDOStatement::execute, pasando la sentencia preparada como un argumento.

   Para obtener más información sobre la API de PDO, consulte http://php.net/manual/en/book.pdo.php.

Ejemplo

Prepare y ejecute una sentencia de SQL CALL.

```
$sql = 'CALL match_animal(?, ?)';
$stmt = $conn->prepare($sql);

$second_name = "Rickety Ride";
$weight = 0;

$stmt->bindParam(1, $second_name, PDO::PARAM_STR|PDO::PARAM_INPUT_OUTPUT, 32);
$stmt->bindParam(2, $weight, PDO::PARAM_INT, 10);

print "Values of bound parameters _before_ CALL:\n";
print "  1: {$second_name} 2: {$weight}\n";

$stmt->execute();
```

```
print "Values of bound parameters _after_ CALL:\n";
print "  1: {$second_name} 2: {$weight}\n";
```

Recuperación de varios conjuntos de resultados de un procedimiento almacenado en PHP (PDO):

Cuando una única llamada a un procedimiento almacenado devuelve más de un conjunto de resultados, puede utilizar el método PDOStatement::nextRow de la API de PDO para recuperar los conjuntos de resultados.

Antes de empezar

Debe tener un objeto PDOStatement que haya devuelto la llamada a un procedimiento almacenado con el método PDO::query o PDOStatement::execute.

Procedimiento

Para recuperar varios conjuntos de resultados:

1. Capte filas del primer conjunto de resultados devuelto por el procedimiento mediante la llamada a uno de los métodos de captación de PDO. (El primer conjunto de resultados que se devuelve del procedimiento está asociado con el objeto PDOStatement que devuelve la sentencia CALL).
   - Para devolver una única fila de un conjunto de resultados como una matriz u objeto, llame al método PDOStatement::fetch.
   - Para devolver todas las filas del conjunto de resultados como una matriz de matrices u objetos, llame al método PDOStatement::fetchAll.

   Capte filas desde el objeto PDOStatement hasta que no haya disponibles más filas en el primer conjunto de resultados.
2. Recupere los conjuntos de resultados siguientes llamando al método PDOStatement::nextRowset para devolver el conjunto de resultados siguiente. Puede captar filas desde el objeto PDOStatement hasta que no haya disponibles más filas en el conjunto de resultados.

   El método PDOStatement::nextRowset devuelve False cuando no hay más conjuntos de resultados disponibles o cuando el procedimiento no ha devuelto ningún conjunto de resultados.

   Para obtener más información sobre la API de PDO, consulte http://php.net/manual/en/book.pdo.php.

Ejemplo

Recuperar varios conjuntos de resultados de un procedimiento almacenado.

```
$sql = 'CALL multiple_results()';
$stmt = $conn->query($sql);
do {
   $rows = $stmt->fetchAll(PDO::FETCH_NUM);
   if ($rows) {
       print_r($rows);
   }
} while ($stmt->nextRowset());
```

Qué hacer a continuación

Cuando esté listo para cerrar la conexión a la base de datos, establezca el objeto PDO en NULL. La conexión se cierra automáticamente cuando finaliza el script de PHP.

Modalidades de confirmación en PHP (PDO)

Puede controlar cómo se confirman los grupos de sentencias de SQL especificando una modalidad de confirmación para un recurso de conexión. La extensión PDO admite dos modalidades de confirmación: la confirmación automática y la confirmación manual.

Modalidad de confirmación automática

En modalidad de confirmación automática, cada sentencia de SQL es una transacción completa, que se confirma automáticamente. La modalidad Autocommit le ayuda a evitar problemas de escalas de bloqueo que puedan obstaculizar el rendimiento de aplicaciones Web muy escalables. Por omisión, la extensión PDO abre cada conexión en modalidad de confirmación automática.

Modalidad de confirmación manual

En modalidad de confirmación manual, la transacción comienza al llamar al método PDO::beginTransaction, y finaliza al llamar al método PDO::commit o PDO::rollBack. Esto significa que las sentencias ejecutadas (en la misma conexión) entre el inicio de una transacción y la llamada al método de confirmación o retrotracción se tratan como una única transacción.

La modalidad de confirmación manual es útil si se necesita retrotraer una transacción que contiene una o varias sentencias de SQL. Si emite sentencias SQL en una transacción y el script finaliza sin confirmar o retrotraer explícitamente la transacción, PDO automáticamente retrotrae cualquier trabajo realizado en la transacción.

Después de confirmar o retrotraer la transacción, PDO restablece automáticamente la conexión de base de datos a la modalidad Autocommit.

Para obtener más información sobre la API de PDO, consulte http://php.net/manual/en/book.pdo.php.

Ejemplo

Finalizar la transacción cuando se llama a PDO::commit o PDO::rollBack.

```
$conn = new PDO('ibm:SAMPLE', 'db2inst1', 'ibmdb2', array(
  PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION));
  // PDO::ERRMODE_EXCEPTION significa que un error SQL emite una excepción
try {
  // Emitir estas sentencias SQL en una transacción en un bloque try{}
  $conn->beginTransaction();

  // Una o más sentencias SQL

  $conn->commit();
}
catch (Exception $e) {
  // Si algo ha emitido una excepción en nuestro bloque de transacciones
  // de sentencias, retrotraer cualquier trabajo realizado en la transacción
  print '<p>No se ha podido completar la transacción</p>';
  $conn->rollBack();
}
```

Manejo de errores y avisos en PHP (PDO)

En ocasiones se producen problemas al intentar conectarse a una base de datos o emitir una sentencia de SQL. Es posible que la contraseña para la conexión sea incorrecta, que la tabla a la que hizo referencia en una sentencia SELECT no exista o que la sentencia de SQL no sea válida. PDO proporciona métodos de manejo de errores para ayudarle a recuperarse sin problemas de estas situaciones.

Antes de empezar

Debe configurar el entorno de PHP en el sistema y habilitar las extensiones PDO y PDO_IBM.

Acerca de esta tarea

PDO le ofrece la opción de manejar errores como avisos, errores o excepciones. Sin embargo, cuando crea un nuevo objeto de conexión de PDO, éste siempre emite un objeto PDOException si se produce un error. Si no capta la excepción, PHP imprime un rastreo de la información del error, que puede mostrar las credenciales de conexión de la base de datos, incluido su nombre de usuario y contraseña.

Este procedimiento capta un objeto PDOException y gestiona el error asociado.

Procedimiento

1. Para captar un objeto PDOException y gestionar el error asociado:
   a. Transmita la llamada al constructor de PDO en un bloque `try`.
   b. A continuación del bloque `try`, incluya un bloque `catch` que capte el objeto PDOException.
   c. Recupere el mensaje de error asociado al error invocando el método Exception::getMessage en el objeto PDOException.
2. Para recuperar el SQLSTATE asociado a un objeto PDO o PDOStatement, invoque el método errorCode sobre el objeto.
3. Para recuperar la matriz asociada a un objeto PDO o PDOStatement, invoque el método errorInfo sobre el objeto. La matriz contiene una serie que representa el SQLSTATE como el primer elemento, un entero que representa el código de error SQL o CLI como el segundo elemento y una serie que contiene el texto completo del mensaje de error como el tercer elemento.

   Para obtener más información sobre la API de PDO, consulte http://php.net/manual/en/book.pdo.php.

Capítulo 3. Desarrollo de aplicaciones Python

Desarrollo de aplicaciones Python, SQLAlchemy y Django Framework para servidores de datos de IBM

Python es un lenguaje de creación de scripts general y de primer nivel muy adecuado para el rápido desarrollo de las aplicaciones. Python hace hincapié en la legibilidad del código y admite una variedad de paradigmas de programación, que incluye la programación de procedimientos, orientada a objetos, orientada a aspectos, funcional y la metaprogramación. Python Software Foundation administra el lenguaje Python.

Las extensiones están disponibles para acceder a las bases de datos de IBM Data Server de una aplicación Python:

ibm_db
IBM define esta API, que proporciona el mejor soporte para características avanzadas. Además de emitir consultas de SQL, llamar a procedimientos almacenados y utilizar el soporte pureXML, puede acceder a información de metadatos.

ibm_db_dbi
Esta API implementa Python Database API Specification 2.0. Dado que la API ibm_db_dbi se ajusta a la especificación, no ofrece algunas de las características avanzadas que soporta la API ibm_db. Si tiene una aplicación con un controlador que admita Python Database API Specification v2.0, puede cambiar fácilmente a ibm_db. Las API ibm_db e ibm_db_dbi se empaquetan conjuntamente.

ibm_db_sa
Este adaptador soporta SQLAlchemy, que ofrece un modo flexible para acceder a los servidores de datos de IBM. SQLAlchemy es un popular kit de herramientas de código abierto y correlacionador relacional de objetos (ORM) SQL Python.

ibm_db_django
Este adaptador proporciona acceso a servidores de datos de IBM desde Django. Django es una conocida infraestructura web que se utiliza para crear aplicaciones web elegantes y de alto rendimiento de forma rápida.

Las aplicaciones Python pueden acceder a los servidores de datos de IBM listados:

- IBM DB2 Versión 9.1 para Linux, UNIX y Windows, Fixpack 2 y posterior
- IBM DB2 Universal Database (DB2 UDB) Versión 8 para Linux, UNIX y Windows, Fixpack 15 y posterior
- Conexiones remotas a IBM DB2 para IBM i V5R3, con PTF SI27358 (incluye SI27250)
- Conexiones remotas a IBM DB2 para IBM i 5.4 y posteriores, con PTF SI27256
- Conexiones remotas a IBM DB2 para z/OS, Versión 8 y Versión 9
- IBM Informix Dynamic Server v11.10 y posterior

Descargas y recursos relacionados de Python

Hay disponibles muchos recursos para ayudarle a desarrollar aplicaciones de Python para servidores de datos de IBM.

Tabla 9. Descargas y recursos relacionados de Python.

Esta tabla muestra los recursos de descarga y el enlace a estos. La fila superior de esta tabla es una celda expandida que contiene el título de la tabla.

Descargas	
Python (incluye archivos binarios de Windows. La mayoría de las distribuciones Linux incorporan Python previamente compilado.)	http://www.python.org/download/
SQLAlchemy	http://www.sqlalchemy.org/download.html
Django	http://www.djangoproject.com/download/
Extensiones ibm_db e ibm_db_dbi (incluido el código fuente)	http://pypi.python.org/pypi/ibm_db/
	http://code.google.com/p/ibm-db/downloads/list
Adaptador ibm_db_sa para SQLAlchemy 0.4	http://code.google.com/p/ibm-db/downloads/list
	http://pypi.python.org/pypi/ibm_db_sa
Adaptador ibm_db_django para Django 1.0.x y 1.1	http://code.google.com/p/ibm-db/downloads/list
	http://pypi.python.org/pypi/ibm_db_django
Programa setuptools	http://pypi.python.org/pypi/setuptools
IBM Data Server Driver Package (controlador DS)	http://www.ibm.com/software/data/support/data-server-clients/index.html
Documentación de la API	
Documentación de la API ibm_db	http://code.google.com/p/ibm-db/wiki/APIs
Python Database API Specification v2.0	http://www.python.org/dev/peps/pep-0249/
Documentación de SQLAlchemy	
Pasos rápidos de iniciación para ibm_db_sa	http://code.google.com/p/ibm-db/wiki/README
Documentación de SQLAlchemy	http://www.sqlalchemy.org/docs/index.html
Documentación de Django	
Pasos de iniciación para ibm_db_django	http://code.google.com/p/ibm-db/wiki/ibm_db_django_README
Documentación de Django	http://www.djangoproject.com
Recursos adicionales	
Sitio web del lenguaje de programación Python	http://www.python.org/
Sitio web del kit de herramientas y el correlacionador relacional de objetos de SQL de Python	http://www.sqlalchemy.org/

Configuración del entorno Python para servidores de datos IBM

Antes de poder conectarse a un servidor de datos de IBM y ejecutar sentencias de SQL, debe configurar el entorno Python instalando el paquete ibm_db (Python) y, opcionalmente, los paquetes ibm_db_sa (SQLAlchemy) o ibm_db_django (Django) en el sistema.

Antes de empezar

Debe tener el software listado instalado en su sistema:

- Python 2.5 o posterior, excluyendo Python 3.X. para sistemas operativos Linux; también necesita el paquete python2.5-dev.
- setuptools, un programa disponible en http://pypi.python.org/pypi/setuptools. Este programa se puede utilizar para descargar, compilar, instalar, actualizar y desinstalar paquetes Python.
- Si se conecta desde Python a un servidor de bases de datos DB2 en una máquina remota, necesita uno de los clientes listados de DB2 en la máquina donde esté instalando o ejecutando Python: IBM Data Server Driver Package, IBM Data Server Client o IBM Data Server Driver para ODBC y CLI.
- Si se conecta desde Python a un servidor de DB2 en una máquina local, no necesita instalar ningún cliente de base de datos DB2.

Procedimiento

Para configurar el entorno Python:

1. Configure su entorno Linux o Windows usando uno de los métodos listados:

   Para instalar los controladores de DB2 Python y los adaptadores de estructura más recientes, consígalos de la comunidad. Las versiones de comunidad siempre contienen arreglos actualizados; en ocasiones los más recientes pueden no estar disponibles en los controladores o adaptadores incluidos en la instalación de DB2.

   - Si tiene acceso a Internet, emita uno de los mandatos listados:
     - Para instalar ibm_db: `easy_install ibm_db`
     - Para instalar tanto ibm_db_sa como ibm_db: `easy_install ibm_db_sa`
     - Para instalar ibm_db_django: `easy_install ibm_db_django`

     Este paso instala archivos egg bajo el directorio `site-packages`, donde se instala setuptools.
   - Si no tiene acceso a Internet, copie el archivo egg correspondiente a su sistema de http://code.google.com/p/ibm-db/downloads/list y emita el mandato `easy_install`:

     ```
     easy_install archivo_egg
     ```

     donde *archivo_egg* es la vía de acceso al archivo egg. Por ejemplo, emita el mandato `easy_install`:

     ```
     easy_install /home/user/ibm_db-xx-py2.5-linux-i386.egg
     ```
   - Para realizar la instalación desde el código fuente de la comunidad, descárguelo de http://pypi.python.org/pypi/ibm_db/. A continuación compile e instale el controlador. Las instrucciones para compilar e instalar el controlador se encuentran en el archivo `README` que se incluye con el código fuente del controlador.

- Si desea instalar los adaptadores de estructura y los controladores de DB2 Python que están disponibles en la imagen de DB2, vaya al directorio del archivo egg de la instalación de DB2, por ejemplo: `/home/user/sqllib/python64/`. Luego emita uno de los mandatos listados:
  - Para instalar ibm_db: `easy_install nombre_archivo_egg_ibm_db`, por ejemplo: `easy_install ibm_db-xx-py2.5-linux-x86_64.egg`
  - Para instalar ibm_db_sa: `easy_install nombre_archivo_egg_ibm_db_sa`; por ejemplo: `easy_install ibm_db_sa-xx-py2.5.egg`
  - Para instalar ibm_db_django: `easy_install nombre_archivo_egg_ibm_db_django`, por ejemplo: `easy_install ibm_db_django-xx-py2.5.egg`

  Este paso instala archivos egg bajo el directorio `site-packages`, donde se instala setuptools.

2. Cree una variable de entorno denominada **PYTHONPATH** y establézcala en la vía de acceso donde está instalado el archivo ibm_db egg, tal como se indica en los ejemplos listados:
   - En sistemas operativos Windows: `PYTHONPATH=`*`vía_acceso_instalación_setuptools`*`\site-packages\ibm_db-xx.egg`
   - En Linux (shell BASH): `export PYTHONPATH=`*`vía_acceso_instalación_setuptools`*`/site-packages/ibm_db-xx.egg`
3. Antes de ejecutar cualquier script Python que se conecte a DB2, debe asegurarse de que el controlador de IBM DB2 Python puede acceder al controlador de CLI denominado `libdb2.so`, que forma parte de la configuración de su servidor DB2 o de la configuración del cliente de DB2. Si no lo hace, aparecerá un error de `'faltan las bibliotecas - libdb2.so.1'` cuando ejecute el programa Python. En Linux, debe asegurarse de que el controlador Python de IBM DB2 pueda acceder al controlador de CLI; para ello, añada la vía de acceso del archivo `libdb2.so` a la variable de entorno **LD_LIBRARY_PATH**.

   Cuando se utiliza IBM Data Server Driver Package, `libdb2.so` se encuentra situado en el directorio `odbc_cli_driver/linux/clidriver/lib`.

   En la instalación del servidor DB2, `libdb2.so` se encuentra en el directorio `sqllib/lib`.
4. Para probar la instalación, en el indicador de mandatos escriba `python` para iniciar el intérprete de Python y escriba un código similar al que se muestra en los ejemplos siguientes:
   - Para probar ibm_db:

     ```
     import ibm_db
     ibm_db_conn = ibm_db.connect('dsn=database', 'user', 'password')
     import ibm_db_dbi
     conn = ibm_db_dbi.Connection(ibm_db_conn)
     conn.tables('SYSCAT', '%')
     ```

   - Para probar ibm_db_sa:

     ```
     import sqlalchemy
     from sqlalchemy import *
     import ibm_db_sa.ibm_db_sa
     db2 = sqlalchemy.create_engine('ibm_db_sa://user:password@host.name.com:50000/database')
     metadata = MetaData()
     users = Table('users', metadata,
     Column('user_id', Integer, primary_key = True),
     Column('user_name', String(16), nullable = False),
     Column('email_address', String(60), key='email'),
     Column('password', String(20), nullable = False)
     )
     metadata.bind = db2
     metadata.create_all()
     users_table = Table('users', metadata, autoload=True, autoload_with=db2)
     users_table
     ```

- Para probar ibm_db_django:
  a. Cree un proyecto de Django nuevo:
     ```
     django-admin.py startproject myproj
     ```
  b. Edite el archivo settings.py para configurar el acceso a DB2. Utilice cualquier editor disponible en el sistema. Un ejemplo en Unix es:
     ```
     $ cd myproj
     $ vi settings.py

     DATABASE_ENGINE    = 'ibm_db_django'
     DATABASE_NAME      = 'mydb'
     DATABASE_USER      = 'db2inst1'
     DATABASE_PASSWORD  = 'ibmdb2'
     DATABASE_HOST      = 'localhost'
     DATABASE_PORT      = '50000'
     ```
  c. Añada las líneas listadas en la sección INSTALLED_APPS de la tupla del archivo settings.py.
     ```
     'django.contrib.flatpages',
     'django.contrib.redirects',
     'django.contrib.comments',
     'django.contrib.admin',
     ```
  d. Ejecute una suite de pruebas para confirmar que la configuración es correcta:
     ```
     python manage.py test
     ```

Resultados

Los paquetes de Python se habrán instalado en su sistema y estarán listos para usar.

Qué hacer a continuación

Conéctese al servidor de datos y empiece a emitir sentencias de SQL.

Desarrollo de aplicaciones en Python con ibm_db

La API ibm_db ofrece una gran variedad de útiles funciones de Python para acceder a los datos de una base de datos de IBM Data Server y manipularlos, incluidas funciones para la conexión a una base de datos, la preparación y la emisión de sentencias de SQL, la recuperación de filas de conjuntos de resultados, la llamada a procedimientos almacenados, la confirmación y retrotracción de transacciones, el manejo de errores y la recuperación de metadatos.

Conexión a una base de datos de IBM Data Server en Python

Antes de poder ejecutar sentencias de SQL para crear, actualizar, suprimir o recuperar datos, debe conectarse a una base de datos. Puede utilizar la API ibm_db para conectarse a una base de datos mediante una conexión catalogada o no catalogada. Para mejorar el rendimiento, también puede crear una conexión persistente.

Antes de empezar

- Configure el entorno de Python.
- Emita el mandato `import ibm_db` desde el script de Python.

Procedimiento

Para devolver un recurso de conexión que puede utilizar para llamar a sentencias de SQL, llame a una de las funciones listadas:

Tabla 10. Funciones de conexión de ibm_db

Función	Descripción
ibm_db.connect	Crea una conexión no persistente.
ibm_db.pconnect	Crea una conexión persistente. Una conexión persistente continúa abierta tras la petición inicial del script de Python, lo que permite que las peticiones de Python posteriores reutilicen la conexión si su conjunto de credenciales es idéntico.

El valor de base de datos que proporcione como argumento a estas funciones puede ser un nombre de base de datos catalogado o una serie de conexión de base de datos completa para una conexión TCP/IP directa. Puede especificar argumentos opcionales que controlen la temporización de confirmación de las transacciones, las mayúsculas/minúsculas de los nombres de columna que se devuelven y el tipo de cursor.
Si el intento de conexión falla, puede recuperar información de diagnóstico llamando a la función ibm_db.conn_error o ibm_db.conn_errormsg.
Para obtener más información sobre la API ibm_db, consulte http://code.google.com/p/ibm-db/wiki/APIs.

Ejemplo

Ejemplo 1: Conexión con una base de datos local o catalogada

Enfoque 1:

```
import ibm_db
conn = ibm_db.connect("dsn=name","username","password")
```

Enfoque 2:

```
import ibm_db
conn = ibm_db.connect("name","username","password")
```

Ejemplo 2: Conexión con una base de datos catalogada o no catalogada

```
import ibm_db
ibm_db.connect("DATABASE=name;HOSTNAME=host;PORT=60000;PROTOCOL=TCPIP;UID=username;
                PWD=password;", "", "")
```

Qué hacer a continuación

Si el intento de conexión se realiza correctamente, puede utilizar el recurso de conexión al llamar a funciones de ibm_db que ejecutan sentencias de SQL. A continuación, prepare y ejecute las sentencias de SQL.

Ejecución de sentencias de SQL en Python

Después de conectarse a una base de datos, utilice las funciones disponibles en la API ibm_db para preparar y ejecutar sentencias de SQL. Las sentencias de SQL pueden contener texto estático, expresiones XQuery o marcadores de parámetros que representan la entrada de variables.

Preparación y ejecución de una única sentencia de SQL en Python:

Para preparar y ejecutar una única sentencia de SQL, utilice la función ibm_db.exec_immediate. Para evitar amenazas a la seguridad por ataques de inyección de SQL, utilice la función ibm_db.exec_immediate sólo para ejecutar sentencias de SQL compuestas de series estáticas. La interpolación de variables de Python que representan la entrada del usuario en la sentencia de SQL puede exponer la aplicación a ataques de inyección de SQL.

Antes de empezar

Obtener un recurso de conexión invocando una de las funciones de conexión en la API ibm_db. Consulte "Conexión a una base de datos de IBM Data Server en Python" en la página 51.

Procedimiento

Para preparar y ejecutar una única sentencia de SQL, llame a la función ibm_db.exec_immediate, proporcionando los argumentos listados:

conexión
Un recurso de conexión de base de datos válido que devuelve la función ibm_db.connect o ibm_db.pconnect.

sentencia
Una serie que contiene la sentencia de SQL. Esta serie puede incluir una expresión XQuery a la que llama la función XMLQUERY.

opciones
Opcional: Un diccionario que especifica el tipo de cursor que debe devolverse para conjuntos de resultados. Puede utilizar este parámetro para solicitar un cursor desplazable para servidores de bases de datos que admitan este tipo de cursor. Por omisión, se devuelve un cursor de sólo avance.

Si la llamada de función falla (devuelve `False`), puede utilizar la función ibm_db.stmt_error o ibm_db.stmt_errormsg para recuperar información de diagnóstico sobre el error.
Si la llamada de función tiene éxito, puede utilizar la función ibm_db.num_rows para devolver el número de filas que la sentencia de SQL ha devuelto o a las que ha afectado. Si la sentencia de SQL devuelve un conjunto de resultados, puede empezar a captar las filas.
Para obtener más información sobre la API ibm_db, consulte http://code.google.com/p/ibm-db/wiki/APIs.

Ejemplo

Ejemplo 1: Ejecutar una única sentencia de SQL

```
import ibm_db
conn = ibm_db.connect("dsn=name","username","password")
stmt = ibm_db.exec_immediate(conn, "UPDATE employee SET bonus = '1000' WHERE job =
'MANAGER'")
print "Number of affected rows: ", ibm_db.num_rows(stmt)
```

Ejemplo 2: Ejecutar una expresión XQuery

```
import ibm_db
conn = ibm_db.connect("dsn=name","username","password")
if conn:
    sql = "SELECT XMLSERIALIZE(XMLQUERY('for $i in $t/address where
    $i/city = \"Olathe\" return <zip>{$i/zip/text()}</zip>'
    passing c.xmlcol as \"t\") AS CLOB(32k)) FROM xml_test c WHERE id = 1"
    stmt = ibm_db.exec_immediate(conn, sql)
    result = ibm_db.fetch_both(stmt)
    while( result ):
        print "Result from XMLSerialize and XMLQuery:", result[0]
        result = ibm_db.fetch_both(stmt)
```

Qué hacer a continuación

Si la sentencia de SQL devuelve uno o varios conjuntos de resultados, puede empezar a captar filas del recurso de sentencia.

Preparación y ejecución de sentencias de SQL con entrada de variables en Python:

Para preparar y ejecutar una sentencia de SQL que incluya la entrada de variables, utilice las funciones ibm_db.prepare, ibm_db.bind_param e ibm_db.execute. Preparar una sentencia mejora el rendimiento porque el servidor de bases de datos crea un plan de acceso optimizado para la recuperación de datos que puede reutilizar si se vuelve a ejecutar la sentencia.

Antes de empezar

Obtener un recurso de conexión invocando una de las funciones de conexión en la API ibm_db. Consulte "Conexión a una base de datos de IBM Data Server en Python" en la página 51.

Procedimiento

Para preparar y ejecutar una sentencia de SQL que incluye marcadores de parámetro:

1. Llame a la función ibm_db.prepare, pasando los argumentos listados:

   conexión
   Un recurso de conexión de base de datos válido que devuelve la función ibm_db.connect o ibm_db.pconnect.

   sentencia
   Una serie que contiene la sentencia de SQL, incluidos los signos de interrogación (?), como marcadores de parámetro para los valores de columna o predicado que requieran entrada de variables. Esta serie puede incluir una expresión XQuery a la que llama la función XMLQUERY.

   opciones
   Opcional: Un diccionario que especifica el tipo de cursor que debe devolverse para conjuntos de resultados. Puede utilizar este parámetro para solicitar un cursor desplazable para servidores de bases de datos que admitan este tipo de cursor. Por omisión, se devuelve un cursor de sólo avance.

   Si la llamada de función tiene éxito, devuelve un recurso de descriptor de contexto de sentencia que puede utilizar en llamadas de función posteriores relacionadas con la consulta.

   Si la llamada de función falla (devuelve `False`), puede utilizar la función ibm_db.stmt_error o ibm_db.stmt_errormsg para recuperar información de diagnóstico sobre el error.
2. Opcional: Por cada marcador de parámetro en la serie de SQL, llame a la función ibm_db.bind_param, proporcionando los argumentos listados. Vincular valores de entrada a marcadores de parámetros garantiza que cada valor de entrada se trate como un único parámetro, evitando ataques de inyección de SQL.

   sentencia
   La sentencia preparada que devuelve la llamada a la función ibm_db.prepare.

número_parámetro
Un entero que representa la posición del marcador de parámetro en la sentencia de SQL.

variable
Valor que se debe utilizar en lugar del marcador de parámetro.

3. Llame a la función ibm_db.execute, pasando los argumentos listados:

sentencia
Una sentencia preparada que se devuelve de ibm_db.prepare.

parámetros
Una tupla de parámetros de entrada que coinciden con los marcadores de parámetros contenidos en la sentencia preparada.

Para obtener más información sobre la API ibm_db, consulte http://code.google.com/p/ibm-db/wiki/APIs.

Ejemplo

Preparar y ejecutar una sentencia que incluye entrada de variables.

```
import ibm_db
conn = ibm_db.connect("dsn=name","username","password")
sql = "SELECT EMPNO, LASTNAME FROM EMPLOYEE WHERE EMPNO > ? AND EMPNO < ?"
stmt = ibm_db.prepare(conn, sql)
max = 50
min = 0
# Vincular explícitamente los parámetros
ibm_db.bind_param(stmt, 1, min)
ibm_db.bind_param(stmt, 2, max)
ibm_db.execute(stmt)
# Procesar resultados

# Invocar de nuevo sentencia preparada usando parámetros vinculados dinámicamente
param = max, min,
ibm_db.execute(stmt, param)
```

Qué hacer a continuación

Si la sentencia de SQL devuelve uno o varios conjuntos de resultados, puede empezar a captar filas del recurso de sentencia.

Captación de filas o columnas de conjuntos de resultados en Python

Después de ejecutar una sentencia que devuelve uno o más conjuntos de resultados, utilice una de las funciones disponibles en la API ibm_db para iterar por las filas devueltas. Si el conjunto de resultados incluye columnas que contienen datos extremadamente grandes (por ejemplo, datos BLOB o CLOB), puede recuperar los datos de columna en columna para evitar que se utilice demasiada memoria.

Antes de empezar

Debe tener un recurso de sentencia que haya devuelto la función ibm_db.exec_immediate o ibm_db.execute que tenga asociados uno o varios conjuntos de resultados.

Procedimiento

Para captar datos de un conjunto de resultados:

1. Capte los datos de un conjunto de resultados llamando a una de las funciones de captación.

Tabla 11. Funciones de captación de ibm_db

Función	Descripción
ibm_db.fetch_tuple	Devuelve una tupla, indexado por posición de columna, que representa una fila en un conjunto de resultados. Las columnas están indexadas en 0.
ibm_db.fetch_assoc	Devuelve un diccionario, indexado por nombre de columna, que representa una fila en un conjunto de resultados.
ibm_db.fetch_both	Devuelve un diccionario, indexado tanto por nombre de columna como por posición, que representa una fila en un conjunto de resultados.
ibm_db.fetch_row	Establece el puntero del conjunto de resultados en la siguiente fila o la fila solicitada. Utilice esta función para iterar por un conjunto de resultados.

Estas funciones aceptan los argumentos listados:

sentencia
Un recurso de sentencia válido.

número_fila
El número de la fila que desea recuperar del conjunto de resultados. Especifique un valor para este parámetro si ha solicitado un cursor desplazable al llamar a la función ibm_db.exec_immediate o ibm_db.prepare. Con el cursor de sólo avance por omisión, cada llamada a un método de captación devuelve la fila siguiente del conjunto de resultados.

2. Opcional: Si ha llamado a la función ibm_db.fetch_row, para cada iteración de todo el conjunto de resultados, recupere un valor de una columna especificada llamando a la función ibm_db.result. Puede especificar la columna pasando un entero que represente la posición de la columna en la fila (empezando por 0) o pasando una serie que represente el nombre de la columna.
3. Continúe captando filas hasta que el método de captación devuelva un valor falso, lo que indicará que habrá llegado al final del conjunto de resultados.

Para obtener más información sobre la API ibm_db, consulte http://code.google.com/p/ibm-db/wiki/APIs.

Ejemplo

Ejemplo 1: Captación de filas de un conjunto de resultados llamando a la función ibm_db.fetch_both

```
import ibm_db

conn = ibm_db.connect( "dsn=name", "username", "password" )
sql = "SELECT * FROM EMPLOYEE"
stmt = ibm_db.exec_immediate(conn, sql)
dictionary = ibm_db.fetch_both(stmt)
while dictionary != False:
    print "The ID is : ",  dictionary["EMPNO"]
    print "The Name is : ", dictionary[1]
    dictionary = ibm_db.fetch_both(stmt)
```

Ejemplo 2: Captación de filas de un conjunto de resultados llamando a la función ibm_db.fetch_tuple

```
import ibm_db

conn = ibm_db.connect( "dsn=name", "username", "password" )
sql = "SELECT * FROM EMPLOYEE"
stmt = ibm_db.exec_immediate(conn, sql)
tuple = ibm_db.fetch_tuple(stmt)
while tuple != False:
    print "The ID is : ", tuple[0]
    print "The name is : ", tuple[1]
    tuple = ibm_db.fetch_tuple(stmt)
```

Ejemplo 3: Captación de filas de un conjunto de resultados llamando a la función ibm_db.fetch_assoc

```
import ibm_db

conn = ibm_db.connect( "dsn=name", "username", "password" )
sql = "SELECT * FROM EMPLOYEE"
stmt = ibm_db.exec_immediate(conn, sql)
dictionary = ibm_db.fetch_assoc(stmt)
while dictionary != False:
    print "The ID is : ", dictionary["EMPNO"]
    print "The name is : ", dictionary["FIRSTNME"]
    dictionary = ibm_db.fetch_assoc(stmt)
```

Ejemplo 4: Captación de columnas de un conjunto de resultados

```
import ibm_db

conn = ibm_db.connect( "dsn=name", "username", "password" )
sql = "SELECT * FROM EMPLOYEE
stmt = ibm_db.exec_immediate(conn, sql)
while ibm_db.fetch_row(stmt) != False:
    print "The Employee number is : ",  ibm_db.result(stmt, 0)
    print "The Name is : ", ibm_db.result(stmt, "NAME")
```

Qué hacer a continuación

Cuando esté listo para cerrar la conexión a la base de datos, llame a la función ibm_db.close. Si intenta cerrar una conexión persistente creada con ibm_db.pconnect, la petición de cierre devuelve True, y la conexión permanecerá disponible para el siguiente interlocutor.

Invocación de procedimientos almacenados en Python

Para llamar a un procedimiento almacenado desde una aplicación Python, prepare y ejecute una sentencia SQL CALL. El procedimiento que invoca puede incluir parámetros de entrada (IN), parámetro de salida (OUT) y parámetros de entrada y salida.

Antes de empezar

Obtener un recurso de conexión invocando una de las funciones de conexión en la API ibm_db. Consulte "Conexión a una base de datos de IBM Data Server en Python" en la página 51.

Procedimiento

Para llamar a un procedimiento almacenado:

1. Llame a la función ibm_db.prepare, pasando los argumentos listados:

conexión
Un recurso de conexión de base de datos válido que devuelve ibm_db.connect o ibm_db.pconnect.

sentencia
Una serie que contiene la sentencia de SQL CALL, incluidos los marcadores de parámetro (?), para los parámetros de entrada o salida.

opciones
Opcional: Un diccionario que especifica el tipo de cursor que debe devolverse para conjuntos de resultados. Puede utilizar este parámetro para solicitar un cursor desplazable para servidores de bases de datos que admitan este tipo de cursor. Por omisión, se devuelve un cursor de sólo avance.

2. Por cada marcador de parámetro en la sentencia CALL, llame a la función ibm_db.bind_param, proporcionando los argumentos listados:

   sentencia
   La sentencia preparada que devuelve la llamada a la función ibm_db.prepare.

   número_parámetro
   Un entero que representa la posición del marcador de parámetro en la sentencia de SQL.

   variable
   El nombre de la variable Python que contendrá la salida.

   tipo_parámetro
   Una constante que especifica si se vinculará la variable de Python al parámetro de SQL como un parámetro de entrada (SQL_PARAM_INPUT), un parámetro de salida (SQL_PARAM_OUTPUT) o un parámetro que acepta una entrada y devuelve una salida (SQL_PARAM_INPUT_OUTPUT).

   Este paso vincula cada marcador de parámetro al nombre de una variable de Python que contendrá la salida.

3. Llame a la función ibm_db.execute, pasando la sentencia preparada como un argumento.

   Para obtener más información sobre la API ibm_db, consulte http://code.google.com/p/ibm-db/wiki/APIs.

Ejemplo

Prepare y ejecute una sentencia de SQL CALL.

```
import ibm_db

conn = ibm_db.connect("dsn=sample","username","password")
if conn:
      sql = 'CALL match_animal(?, ?, ?)'
      stmt = ibm_db.prepare(conn, sql)

      name = "Peaches"
      second_name = "Rickety Ride"
      weight = 0
      ibm_db.bind_param(stmt, 1, name, ibm_db.SQL_PARAM_INPUT)
      ibm_db.bind_param(stmt, 2, second_name, ibm_db.SQL_PARAM_INPUT_OUTPUT)
      ibm_db.bind_param(stmt, 3, weight, ibm_db.SQL_PARAM_OUTPUT)

      print "Values of bound parameters _before_ CALL:"
      print "  1: %s 2: %s 3: %d\n" % (name, second_name, weight)
```

```
if ibm_db.execute(stmt):
  print "Values of bound parameters _after_ CALL:"
  print "  1: %s 2: %s 3: %d\n" % (name, second_name, weight)
```

Qué hacer a continuación

Si la llamada al procedimiento devuelve uno o varios conjuntos de resultados, puede empezar a captar filas del recurso de sentencia.

Recuperación de varios conjuntos de resultados de un procedimiento almacenado en Python

Cuando una única llamada a un procedimiento almacenado devuelve más de un conjunto de resultados, puede utilizar la función ibm_db.next_result de la API ibm_db para recuperar los conjuntos de resultados.

Antes de empezar

Debe disponer de un recurso de sentencia que haya devuelto la función ibm_db.exec_immediate o ibm_db.execute que tenga varios conjuntos de resultados.

Procedimiento

Para recuperar varios conjuntos de resultados:

1. Captar filas del primer conjunto de resultados devuelto por el procedimiento mediante la llamada a una de las funciones listadas de captación ibm_db, proporcionando el recurso de sentencia como argumento. (El primer conjunto de resultados que se devuelve del procedimiento está asociado con el recurso de sentencia).

Tabla 12. Funciones de captación de ibm_db

Función	Descripción
ibm_db.fetch_tuple	Devuelve una tupla, indexado por posición de columna, que representa una fila en un conjunto de resultados. Las columnas están indexadas en 0.
ibm_db.fetch_assoc	Devuelve un diccionario, indexado por nombre de columna, que representa una fila en un conjunto de resultados.
ibm_db.fetch_both	Devuelve un diccionario, indexado tanto por nombre de columna como por posición, que representa una fila en un conjunto de resultados.
ibm_db.fetch_row	Establece el puntero del conjunto de resultados en la siguiente fila o la fila solicitada. Utilice esta función para iterar por un conjunto de resultados.

2. Recuperar los conjuntos de resultados siguientes proporcionando el recurso de sentencia original como primer argumento a la función ibm_db.next_result. Puede captar filas del recurso de sentencia hasta que no haya disponibles más filas en el conjunto de resultados.

   La función ibm_db.next_result devuelve False cuando no hay más conjuntos de resultados disponibles o el procedimiento no ha devuelto ningún conjunto de resultados.

Para obtener más información sobre la API ibm_db, consulte http://code.google.com/p/ibm-db/wiki/APIs.

Ejemplo

Recuperar varios conjuntos de resultados de un procedimiento almacenado.

```
import ibm_db
conn = ibm_db.connect( "dsn=sample", "user", "password" )
if conn:
    sql = 'CALL sp_multi()'
    stmt = ibm_db.exec_immediate(conn, sql)
    row = ibm_db.fetch_assoc(stmt)
    while row != False :
        print "The value returned : ", row
        row = ibm_db.fetch_assoc(stmt)

    stmt1 = ibm_db.next_result(stmt)
    while stmt1 != False:
        row = ibm_db.fetch_assoc(stmt1)
        while row != False :
            print "The value returned : ", row
            row = ibm_db.fetch_assoc(stmt1)
        stmt1 = ibm_db.next_result(stmt)
```

Qué hacer a continuación

Cuando esté listo para cerrar la conexión a la base de datos, llame a la función ibm_db.close. Si intenta cerrar una conexión persistente creada con ibm_db.pconnect, la petición de cierre devuelve True, y la conexión del cliente de servidor de datos de IBM permanecerá disponible para el siguiente interlocutor.

Modalidades de confirmación en aplicaciones Python

Puede controlar cómo se confirman los grupos de sentencias de SQL especificando una modalidad de confirmación para un recurso de conexión. La API ibm_db admite dos modalidades de confirmación: la confirmación automática y la confirmación manual.

Modalidad de confirmación automática

En modalidad de confirmación automática, cada sentencia de SQL es una transacción completa, que se confirma automáticamente. La modalidad Autocommit le ayuda a evitar problemas de escalas de bloqueo que puedan obstaculizar el rendimiento de aplicaciones web muy escalables. Por omisión, la API ibm_db abre cada conexión en modalidad de confirmación automática.

Puede activar la modalidad de confirmación automática tras inhabilitarla llamando a `ibm_db.autocommit(`*conn*`, ibm_db.SQL_AUTOCOMMIT_ON)`, donde *con* es un recurso de conexión válido.

Llamar a la función ibm_db.autocommit puede afectar al rendimiento de sus scripts de Python, porque requiere comunicación adicional entre Python y el sistema de gestión de bases de datos.

Modalidad de confirmación manual

En modalidad de confirmación manual, la transacción finaliza al llamar a la función ibm_db.commit o ibm_db.rollback. Esto significa que todas las sentencias ejecutadas en la misma conexión entre el inicio de una transacción y la llamada a la función de confirmación o retrotracción se tratan como una única transacción.

La modalidad de confirmación manual es útil si se necesita retrotraer una transacción que contiene una o varias sentencias de SQL. Si ejecuta sentencias de SQL en una transacción y el script finaliza sin confirmar o retrotraer explícitamente la transacción, la extensión ibm_db retrotrae automáticamente cualquier trabajo realizado en la transacción.

Puede inhabilitar la modalidad de confirmación manual al crear una conexión de base de datos utilizando el valor de { `ibm_db.SQL_ATTR_AUTOCOMMIT: ibm_db.SQL_AUTOCOMMIT_OFF` } en la matriz de opciones ibm_db.connect o ibm_db.pconnect. También puede desactivar la modalidad de confirmación automática para un recurso de conexión llamando a `ibm_db.autocommit(`*con*`, ibm_db.SQL_AUTOCOMMIT_OFF)`, donde *con* es un recurso de conexión válido.

Para obtener más información sobre la API ibm_db, consulte http://code.google.com/p/ibm-db/wiki/APIs.

Ejemplo

Inhabilitar la modalidad de confirmación automática y finalizar la transacción cuando se llama a ibm_db.commit o ibm_db.rollback.

```
import ibm_db

array = { ibm_db.SQL_ATTR_AUTOCOMMIT : ibm_db.SQL_AUTOCOMMIT_OFF }
conn = ibm_db.pconnect("dsn=SAMPLE", "user", "password", array)
sql = "DELETE FROM EMPLOYEE"
try:
   stmt = ibm_db.exec_immediate(conn, sql)
except:
   print "Transaction couldn't be completed."
   ibm_db.rollback(conn)
else:
   ibm_db.commit(conn)
   print "Transaction complete."
```

Funciones de manejo de errores en Python

En ocasiones se producen problemas al intentar conectarse a una base de datos o emitir una sentencia de SQL. El nombre de usuario o la contraseña pueden ser incorrectos, puede haberse escrito mal un nombre de tabla o columna o la sentencia de SQL puede no ser válida. La API ibm_db proporciona funciones de manejo de errores para ayudarle a recuperarse sin problemas de estas situaciones.

Errores de conexión

Utilice una de las funciones listadas para recuperar información de diagnóstico si falla un intento de conexión.

Tabla 13. Funciones de ibm_db para manejar errores de conexión

Función	Descripción
ibm_db.conn_error	Recupera el SQLSTATE devuelto por el último intento de conexión.
ibm_db. conn_errormsg	Recupera un mensaje de error descriptivo adecuado a la anotación cronológica de errores de la aplicación

Errores de SQL

Utilice una de las funciones listadas para recuperar información de diagnóstico si falla un intento de preparar o ejecutar una sentencia de SQL o de captar un resultado de un conjunto de resultados.

Tabla 14. Funciones de ibm_db para manejar errores de SQL

Función	Descripción
ibm_db.stmt_error	Recupera el SQLSTATE devuelto por el último intento de preparar o ejecutar una sentencia de SQL o de captar un resultado de un conjunto de resultados.
ibm_db.stmt_errormsg	Recupera un mensaje de error descriptivo adecuado a la anotación cronológica de errores de la aplicación

Para obtener más información sobre la API ibm_db, consulte http://code.google.com/p/ibm-db/wiki/APIs.

Ejemplo

Ejemplo 1: Manejo de errores de conexión

```
import ibm_db
try:
    conn = ibm_db.connect("dsn=sample","user","password")
except:
    print "no connection:", ibm_db.conn_errormsg()
else:
    print "The connection was successful"
```

Ejemplo 2: Manejo de errores de SQL

```
import ibm_db
conn = ibm_db.connect( "dsn=sample", "user", "password")
sql = "DELETE FROM EMPLOYEE"
try:
   stmt = ibm_db.exec_immediate(conn, sql)
except:
   print "Transaction couldn't be completed:" , ibm_db.stmt_errormsg()
else:
   print "Transaction complete."
```

Funciones de recuperación de metadatos de base de datos en Python

Puede utilizar funciones en la API ibm_db para recuperar metadatos para bases de datos de IBM.

Antes de llamar a estas funciones, debe instalar el entorno Python, emitir `import_db` en su script de Python y obtener un recurso de conexión llamando a la función ibm_db.connect o ibm_db.pconnect.

Importante: La llamada a funciones de metadatos utiliza una cantidad de espacio considerable. Si es posible, modifique los resultados de las llamadas para su uso en llamadas posteriores.

Tabla 15. Funciones de recuperación de metadatos de ibm_db

Función	Descripción
ibm_db.client_info	Devuelve un objeto de sólo lectura con información acerca del cliente de servidor de datos de IBM.
ibm_db.column_privileges	Devuelve un conjunto de resultados que enumera las columnas y los privilegios asociados para una tabla
ibm_db.columns	Devuelve un conjunto de resultados que enumera las columnas y los metadatos asociados para una tabla
ibm_db.foreign_keys	Devuelve un conjunto de resultados que enumera las claves foráneas para una tabla
ibm_db.primary_keys	Devuelve un conjunto de resultados que enumera las claves primarias para una tabla
ibm_db.procedure_columns	Devuelve un conjunto de resultados que enumera los parámetros para uno o varios procedimientos almacenados
ibm_db.procedures	Devuelve una lista de resultados que enumera los procedimientos almacenados registrados en una base de datos
ibm_db.server_info	Devuelve un objeto de sólo lectura con información acerca de IBM Data Server
ibm_db.special_columns	Devuelve un conjunto de resultados que enumera las columnas de identificador exclusivo de fila para una tabla
ibm_db.statistics	Devuelve un conjunto de resultados que enumera el índice y las estadísticas para una tabla
ibm_db.table_privileges	Devuelve un conjunto de resultados que enumera las tablas de una base de datos y los privilegios asociados

Para obtener más información sobre la API ibm_db, consulte http://code.google.com/p/ibm-db/wiki/APIs.

Ejemplo

Ejemplo 1: Mostrar información sobre IBM Data Server Client

```
import ibm_db

conn = ibm_db.connect("dsn=sample", "user", "password")
client = ibm_db.client_info(conn)

if client:
    print "DRIVER_NAME: string(%d) \"%s\"" % (len(client.DRIVER_NAME),
           client.DRIVER_NAME)
    print "DRIVER_VER: string(%d) \"%s\"" % (len(client.DRIVER_VER),
           client.DRIVER_VER)
    print "DATA_SOURCE_NAME: string(%d) \"%s\"" % (len(client.DATA_SOURCE_NAME),
           client.DATA_SOURCE_NAME)
    print "DRIVER_ODBC_VER: string(%d) \"%s\"" % (len(client.DRIVER_ODBC_VER),
           client.DRIVER_ODBC_VER)
    print "ODBC_VER: string(%d) \"%s\"" % (len(client.ODBC_VER), client.ODBC_VER)
    print "ODBC_SQL_CONFORMANCE: string(%d) \"%s\"" %
          (len(client.ODBC_SQL_CONFORMANCE), client.ODBC_SQL_CONFORMANCE)
    print "APPL_CODEPAGE: int(%s)" % client.APPL_CODEPAGE
    print "CONN_CODEPAGE: int(%s)" % client.CONN_CODEPAGE
    ibm_db.close(conn)
else:
    print "Error."
```

Ejemplo 2: Mostrar información sobre IBM Data Server

```
import ibm_db

conn = ibm_db.connect("dsn=sample", "user", "password")
server = ibm_db.server_info(conn)

if server:
    print "DBMS_NAME: string(%d) \"%s\"" % (len(server.DBMS_NAME), server.DBMS_NAME)
    print "DBMS_VER: string(%d) \"%s\"" % (len(server.DBMS_VER), server.DBMS_VER)
    print "DB_NAME: string(%d) \"%s\"" % (len(server.DB_NAME), server.DB_NAME)
    ibm_db.close(conn)
else:
    print "Error."
```

Capítulo 4. Desarrollo de aplicaciones Ruby on Rails

Controlador IBM_DB Ruby y adaptador Rails

Con la introducción del soporte de la infraestructura Ruby on Rails, las aplicaciones Rails ya pueden acceder a los datos de los servidores de datos de IBM.

Conjuntamente conocido como IBM_DB Ruby, el controlador IBM_DB Ruby y adaptador Rails permiten que las aplicaciones Ruby accedan a los sistemas de gestión de bases de datos listados:

- DB2 para Linux, UNIX y Windows Versión 9 y posterior
- DB2 Universal Database (DB2 UDB) Versión 8 para Linux, UNIX y Windows
- DB2 UDB Versión 5, Release 1 (y posterior) para AS/400 e iSeries, mediante DB2 Connect
- DB2 para z/OS, Versión 8 y Versión 9, mediante DB2 Connect
- Informix Dynamic Server, Versión 11.10 y posterior

  Nota: Las aplicaciones cliente deben usar IBM Data Server Driver Versión 9.5 o posterior al acceder a Informix Dynamic Server Versión 11.10. No se da soporte a las versiones anteriores. Las aplicaciones cliente también deben utilizar IBM Data Server Runtime Client o IBM Data Server Client.

El controlador IBM_DB Ruby puede utilizarse para conectarse y acceder a datos desde los servidores de datos de IBM mencionados anteriormente. El adaptador IBM_DB Ruby permite que cualquier aplicación Rails respaldada por una base de datos interactúe con servidores de datos de IBM.

Para obtener más información sobre los proyectos de IBM Ruby y la comunidad de código abierto RubyForge, consulte http://rubyforge.org/projects/rubyibm/

Para obtener una lista de requisitos de instalación para productos de base de datos de DB2, consulte http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.qb.server.doc/doc/r0025127.html

Para obtener una lista de requisitos de instalación para IBM Informix Dynamic Server, consulte http://publib.boulder.ibm.com/infocenter/idshelp/v111/topic/com.ibm.expr.doc/ids_in_004x.html

Para obtener información sobre cómo descargar IBM Data Server Driver Package (controlador DS), consulte http://www.ibm.com/software/data/support/data-server-clients/index.html.

Iniciación a los servidores de datos de IBM en Rails

Para empezar a desarrollar las aplicaciones de Ruby on Rails con los servidores de datos de IBM, es necesario configurar el entorno de Rails con los servidores de datos de IBM. Para comenzar, puede descargar la versión gratuita de DB2 y empezar a desarrollar las aplicaciones de Rails usando DB2.

Antes de empezar

Para asegurarse de que los valores numéricos entre comillas se gestionan correctamente, es necesario utilizar la Versión 9.1 Fixpack 2 (o posterior) de uno de los tipos de cliente listados: IBM Data Server Driver Package, IBM Data Server Client o IBM Data Server Driver para ODBC y CLI.

Procedimiento

Para configurar el entorno y comenzar a utilizar IBM_DB:

1. Descargue e instale DB2 o IBM Informix desde http://www.ibm.com/software/data/servers/.
2. Descargue e instale la versión más reciente de Ruby desde http://www.ruby-lang.org/en/downloads/.
3. Instale la gema Rails y sus dependencias emitiendo el mandato gem install:
   ```
   gem install rails —-include-dependencies
   ```

Qué hacer a continuación

Ya está listo para instalar el controlador IBM_DB Ruby y el adaptador Rails como una gema. Si lo desea, también puede configurar un entorno de desarrollo integrado (IDE) para Rails.

Configuración de un entorno de desarrollo integrado para Rails

Rails no requiere formatos de archivo ni entornos de desarrollo integrado (IDE) especiales; puede comenzar con un indicador de línea de mandatos y un editor de texto. No obstante, ahora hay disponibles varios IDE con soporte de Rails, por ejemplo, RadRails, que es un entorno de Rails para Eclipse.

Acerca de esta tarea

Para obtener más información acerca de RadRails, consulte http://www.radrails.org/.

Procedimiento

Para configurar un IDE basado en Eclipse para el desarrollo de Ruby on Rails (RoR):

1. Instale Eclipse desde http://www.eclipse.org/downloads/.
2. Instale los plug-ins de Eclipse desde los sitios de actualización remota de Eclipse:
   a. Ruby Development Tools desde http://rubyeclipse.sourceforge.net/download.rdt.html
   b. La característica IDE de RubyRails desde http://radrails.sourceforge.net/update
   c. El plug-in Subclipse desde http://subclipse.tigris.org/update

Instalación del adaptador y controlador IBM_DB como una gema Ruby

Ruby Gems es la infraestructura estándar de empaquetado e instalación para bibliotecas y aplicaciones en el entorno de ejecución de Ruby. Se denomina gema a un solo archivo por cada paquete que es compatible con el formato de paquete.

Este paquete se distribuye y almacena en un depósito central, lo que permite el despliegue simultáneo de varias versiones de la misma biblioteca o aplicación.

Acerca de esta tarea

Si se conecta desde Ruby a un servidor de bases de datos DB2 en una máquina remota, necesita uno de los clientes listados de DB2 en la máquina donde esté instalando o ejecutando Ruby: IBM Data Server Driver Package, IBM Data Server Runtime Client o IBM Data Server Client.

Si se conecta desde Ruby/Ruby on Rails a un servidor de DB2 en una máquina local, no necesita instalar ningún cliente de base de datos DB2.

De forma similar a la gestión de paquetes y los paquetes (.rpm, .deb) que se utilizan en las distribuciones Linux, estas gemas también se pueden consultar, instalar, desinstalar y manejar mediante el programa de utilidad de usuarios finales de gemas.

El programa de utilidad de gemas puede consultar de forma transparente el depósito central remoto de RubyForge, así como buscar e instalar cualquiera de los numerosos programas de utilidad que hay disponibles. Cuando se ha instalado la gema IBM_DB, esta funcionalidad está accesible inmediatamente desde cualquier script (o aplicación) en el entorno de ejecución de Ruby, mediante:

```
require 'ibm_db'
```

o en Windows:

```
require 'mswin32/ibm_db'
```

Procedimiento

Para instalar el adaptador y controlador IBM_DB como una gema Ruby:

1. En las plataformas Linux, UNIX y Mac OS X, establezca las variables de entorno y, opcionalmente, especifique la ubicación del perfil de DB2:
   a. Emita los mandatos `export` para establecer las variables de entorno **IBM_DB_INCLUDE** e **IBM_DB_LIB**:

      ```
      $ export IBM_DB_INCLUDE=DB2HOME/include
      $ export IBM_DB_LIB=DB2HOME/lib
      ```

      donde *DB2HOME* es el directorio donde está instalado el servidor de datos de IBM. Por ejemplo:

      ```
      $ export IBM_DB_INCLUDE=/home/db2inst1/sqllib/include
      $ export IBM_DB_LIB=/home/db2inst1/sqllib/lib
      ```

      Si está utilizando ibm_db 1.0.0 o anterior, en lugar de establecer IBM_DB_INCLUDE, debe definir la variable de entorno **IBM_DB_DIR** con el valor de *DB2HOME*.

      Más información acerca de la definición de variables de entorno:

      En función de la arquitectura para la que está instalado el servidor de datos de IBM, el directorio `lib` bajo *DB2HOME* es un enlace a `lib32` o bien a `lib64`. Puede definir **IBM_DB_LIB** según la arquitectura para la que está compilado Ruby. Para una arquitectura de 32 bits, establezca **IBM_DB_LIB** en el directorio `lib32` bajo *DB2HOME*. Para una arquitectura de 64 bits, establezca **IBM_DB_LIB** en el directorio `lib64` bajo *DB2HOME*.

Nota:

En IBM Data Server Driver Package, *DB2HOME* hace referencia al directorio donde se desempaqueta el paquete del cliente, por ejemplo, el directorio `odbc_cli_driver/linux/clidriver`.

En la instalación del servidor de DB2, *DB2HOME* hace referencia al directorio `sqllib` debajo de la instancia de DB2.

2. En todas las plataformas soportadas, emita el mandato gem install para instalar el adaptador y controlador IBM_DB:

   ```
   $ gem install ibm_db
   ```

3. Antes de ejecutar un script ruby que se conecte a DB2, debe asegurarse de que el controlador de IBM DB2 Ruby puede acceder al controlador de CLI libdb2.so, que forma parte de la configuración de su servidor de DB2 o de la configuración del cliente de DB2. Si no lo hace, aparecerá un error de `faltan las bibliotecas - libdb2.so.1` cuando ejecute el programa Ruby. Puede hacerlo añadiendo la carpeta donde reside el archivo libdb2.so en la variable de entorno **LD_LIBRARY_PATH** de las plataformas Linux o AIX correspondiente al id de usuario en el que se vaya a ejecutar Ruby.

   Cuando se utiliza IBM Data Server Driver Package, el archivo `libdb2.so` se encuentra situado en el directorio `odbc_cli_driver/linux/clidriver/lib`.

   En la instalación del servidor DB2, libdb2.so se encuentra en el directorio `sqllib/lib/`.

   Nota: Para una arquitectura de 32 bits, establezca **LD_LIBRARY_PATH** en el directorio `lib32` bajo *DB2HOME*. Para una arquitectura de 64 bits, establezca **LD_LIBRARY_PATH** en el directorio `lib64` bajo *DB2HOME*.

Resultados

La gema IBM_DB ya está instalada en la estación de trabajo.

Verificación de la instalación de la gema IBM_DB con DB2 Express-C

Para verificar la instalación de la gema IBM_DB con DB2 Express-C, tiene que conectar con la base de datos, emitir una sentencia SELECT y, a continuación, captar la primera fila del conjunto de resultados.

Procedimiento

Utilice los mandatos `gem` para instalar y comprobar la instalación de la gema IBM_DB con Ruby-1.8.6 nivel de parche 111 en un sistema operativo Windows o Linux. También se muestra la salida de los mandatos.

- Para realizar la instalación, emita el mandato **gem install ibm_db**. Por ejemplo:

  ```
  D:\>gem install ibm_db
  Seleccione cuál es la gema que instalará para la plataforma (i386-mswin32)
  1. ibm_db 1.0.1 (ruby)
  2. ibm_db 1.0.1 (mswin32)
  2. ibm_db 1.0.0 (ruby)
  3. ibm_db 1.0.0 (mswin32)
  4. Eludir esta gema
  5. Cancelar la instalación
  > 2
  Se ha instalado satisfactoriamente ibm_db-1.0.0-mswin32
  Instalando la documentación de ri para ibm_db-1.0.0-mswin32...
  Instalando la documentación de RDoc para ibm_db-1.0.0-mswin32...
  ```

Nota: Los ejemplos de este tema incluyen información de versión para mostrar la instalación. No obstante, al ejecutar la instalación, puede escoger entre las dos versiones más recientes disponibles de la gema.
La gema IBM_DB ya está instalada en la máquina.

- Para verificar la instalación, ejecute los mandatos listados.

  Puede realizar este proceso para comprobar la instalación en IBM Informix, DB2 Database para Linux, UNIX y Windows, IBM DB2 para IBM i y DB2 para z/OS. Puede utilizar DB2 Connect para acceder a servidores de datos de IBM DB2 para IBM i y DB2 para z/OS.

```
C:\>irb
irb(main):001:0> require 'mswin32/ibm_db' (si está utilizando una plataforma basada en Linux
emita require 'ibm_db')
=>true
irb(main):002:0> conn = IBM_DB::connect
'devdb','username','password' (donde 'devdb' es la base de datos catalogada en
el directorio de base de datos del cliente)
=> #<IBM_DB::Connection:0x2dddf40>
irb(main):003:0> stmt = IBM_DB::exec conn,'select * from cars'
=> #<IBM_DB::Statement:0x2beaabc>
irb(main):004:0> IBM_DB::fetch_assoc stmt (captará la primera fila del
conjunto de resultados)
```

Qué hacer a continuación

Si estos mandatos se ejecutan satisfactoriamente, la gema se habrá instalado correctamente y podrá empezar a construir aplicaciones de Rails.

Verificación de la instalación con servidores de datos de IBM en aplicaciones de Rails

Para verificar que el adaptador y el controlador IBM_DB se han instalado correctamente, compruebe el acceso del controlador IBM_DB conectando con un servidor de datos de IBM y emitiendo una sentencia SELECT. A continuación, compruebe el acceso del adaptador IBM_DB construyendo y ejecutando una aplicación Rails de ejemplo.

Procedimiento

Para verificar la instalación:

1. Instale la última versión de la gema IBM_DB.
2. Pruebe el acceso del controlador IBM_DB.

   Por ejemplo, para probar el acceso a un servidor de datos i5 mediante el controlador IBM_DB (y DB2 Connect e IBM Data Server Driver para ODBC y CLI subyacentes):

```
D:\ws\RoR\TeamRoom>irb
irb(main):001:0> require 'mswin32/ibm_db'
=> true
irb(main):002:0> conn = IBM_DB::connect 'testdb', 'user', 'pass'
=> #<IBM_DB::Connection:0x2f79d40>
irb(main):003:0> stmt = IBM_DB::exec conn, 'select * from qsys2.qsqptabl'
=> #<IBM_DB::Statement:0x2f762f8>
irb(main):004:0> IBM_DB::fetch_assoc stmt
```

3. Pruebe el acceso del adaptador IBM_DB.

   Para probar el acceso a un servidor de datos de IBM a través del adaptador IBM_DB, siga los pasos que se indican a continuación para construir una aplicación Rails de ejemplo.

   a. Cree una aplicación Rails nueva emitiendo el mandato listado:

```
C:\>rails newapp --database=ibm_db
create
create app/controllers
create app/helpers
create app/models
create app/views/layouts
create config/environments
create config/initializers
create db
[......]
create log/server.log
create log/production.log
create log/development.log
create log/test.log
```

b. Cambie al directorio que se acaba de crear, `newapp`:

```
C:\>cd newapp
```

c. Configure las conexiones de la aplicación Rails editando el archivo `database.yml`. Para obtener más información, consulte "Configuración de conexiones de aplicaciones de servidores de datos de IBM" en la página 71.

Si está utilizando una versión anterior a Rails 2.0, tendrá que registrar el adaptador IBM_DB en la lista de adaptadores de conexión de la infraestructura Rails añadiendo manualmente `ibm_db` a la lista de adaptadores de conexión de `<RubyHome>\gems\1.8\gems\activerecord-1.15.6\lib\active_record.rb` en torno a la línea 77:

```
RAILS_CONNECTION_ADAPTERS = %w( mysql postgresql sqlite firebird
sqlserver db2 oracle sybase openbase frontbase ibm_db )
```

d. Cree un modelo y un andamio emitiendo el mandato `ruby`:

```
C:\>ruby script/generate scaffold Tool name:string model_num:integer
exists app/models/
exists app/controllers/
[....]
create db/migrate
create db/migrate/20080716103959_create_tools.rb
```

e. Emita el mandato `migrate` de Rails para crear la tabla (Tools) en la base de datos (devdb):

```
C:\ >rake db:migrate
(in C:/ruby trials/newapp)
== 20080716111617 CreateTools: migrating
=====================================
-- create_table(:tools)
-> 0.5320s
== 20080716111617 CreateTools: migrated (0.5320s)
```

La aplicación de Rails puede acceder ahora a la tabla Tools y ejecutar operaciones en ella.

f. Emita el mandato `ruby` para probar la aplicación:

```
C:\ruby trials\newapp>ruby script/console
Loading development environment (Rails 2.1.0)
>> tool = Tool.new
=> #<Tool id: nil, name: nil, model_num: nil, created_at: nil,
updated_at: nil>
>> tool.name = 'chistel'
=> "chistel"
>> tool.model_num = '007'
=> "007"
>> tool.save
=> true
>> Tool.find :all
=> [#<Tool id: 100, name: "chistel", model_num: 7, created_at:
"2008-07-16 11:29:35", updated_at: "2008-07-16 11:29:35">]
>>
```

Configuración de conexiones de aplicaciones de servidores de datos de IBM

Las conexiones de base de datos se configuran para una aplicación Rails especificando los detalles de la conexión en el archivo `database.yml`.

Procedimiento

Para configurar conexiones de servidor de datos de sistema principal para una aplicación de Rails:

Edite los detalles de la configuración de bases de datos en *vía_acceso_aplicación_rails*`\config\database.yml` y especifique los atributos de conexión listados:

```
# El adaptador IBM_DB requiere el controlador Ruby nativo (ibm_db)
    # para los servidores de datos de IBM (ibm_db.so).
    # +config+ hash pasado como contenido del argumento del inicializador:
    # == parámetros obligatorios
    #   adapter:     'ibm_db'        // nombre de Adaptador IBM_DB
    #   username:    'db2user'       // usuario de servidor de datos (database)
    #   password:    'secret'        // contraseña de servidor de datos (database)
    #   database:    'DEVDB'         // nombre de base de datos remota (o alias de entrada de catálogo)
    # == opcional (muy recomendado para fines de auditoría y supervisión de servidores de datos)
    #   schema:      'rails123'      // calificador de espacio de nombres
    #   account:     'tester'        // cuenta de SO (estación de trabajo cliente)
    #   app_user:    'test11'        // usuario de aplicación autentificado
    #   application: 'rtests'        // nombre de aplicación
    #   workstation: 'plato'         // nombre de estación de trabajo cliente
    # == conexión TCP/IP remota (necesario cuando no hay ninguna entrada de catálogo de base de datos
    # == local disponible)
    #   host:        'Socrates'      // nombre de host totalmente calificado o dirección IP
    #   port:        '50000'         // número de puerto TCP/IP de servidor de datos
    #
    # Cuando no se especifique el esquema, se utilizará el valor de nombre de usuario en su lugar.
```

Nota: Los cambios en la información de conexión en este archivo se aplican cuando se inicializa el entorno de Rails durante el arranque del servidor. Los cambios realizados después de la inicialización no afectarán a las conexiones que se han creado.

Schema, account, app_user, application y workstation no están soportados para IBM Informix.

Controlador IBM Ruby y contextos fiables

El controlador IBM_DB Ruby admite contextos fiables mediante la utilización de palabras clave de serie.

Los contextos fiables proporcionan un método para crear aplicaciones de tres niveles más seguras y con mayor rapidez. La identidad del usuario siempre se mantiene para operaciones de auditoría y seguridad. Cuando requiere conexiones seguras, los contextos fiables mejoran el rendimiento ya que no es necesario obtener conexiones nuevas.

Ejemplo

El ejemplo establece una conexión fiable y cambia al usuario de la misma conexión.

```
def trusted_connection(database,hostname,port,auth_user,auth_pass,tc_user,tc_pass)
  dsn = "DATABASE=#{database};HOSTNAME=#{hostname};PORT=#{port};PROTOCOL=TCPIP;UID=#{auth_user};
         PWD=#{auth_pass};"
  conn_options = {IBM_DB::SQL_ATTR_USE_TRUSTED_CONTEXT => IBM_DB::SQL_TRUE}
  tc_options = {IBM_DB::SQL_ATTR_TRUSTED_CONTEXT_USERID => tc_user,
                IBM_DB::SQL_ATTR_TRUSTED_CONTEXT_PASSWORD => tc_pass}
  tc_conn = IBM_DB.connect dsn, '', '', conn_options
  if tc_conn
```

```
    puts "Trusted connection established successfully."
    val = IBM_DB.get_option tc_conn, IBM_DB::SQL_ATTR_USE_TRUSTED_CONTEXT, 1
    if val
      userBefore = IBM_DB.get_option tc_conn, IBM_DB::SQL_ATTR_TRUSTED_CONTEXT_USERID, 1
      #do some work as user 1
      #....
      #....
      #switch the user
      result = IBM_DB.set_option tc_conn, tc_options, 1
      userAfter = IBM_DB.get_option tc_conn, IBM_DB::SQL_ATTR_TRUSTED_CONTEXT_USERID, 1
      if userBefore != userAfter
        puts "User has been switched."
        #do some work as user 2
        #....
        #....
      end
    end
    IBM_DB.close tc_conn
  else
    puts "Attempt to connect failed due to: #{IBM_DB.conn_errormsg}"
  end
end
```

Dependencias y consecuencias del adaptador IBM_DB Rails

El adaptador IBM_DB (ibm_db_adapter.rb) tiene una dependencia directa del controlador IBM_DB, que utiliza IBM Data Server Driver para ODBC y CLI para conectarse a los servidores de datos de IBM. La interfaz de nivel de llamada (CLI) de IBM es una interfaz de SQL llamable para servidores de datos de IBM que son compatibles con ODBC (Open Database Connectivity).

Esta dependencia tiene varias ramificaciones para el adaptador y controlador IBM_DB.

- Es necesario realizar la instalación de IBM Data Server Driver para ODBC y CLI, que cumple el requisito de IBM_DB.

  IBM Data Server Driver para ODBC y CLI viene incluido en la instalación de una base de datos DB2 completa, o puede obtenerse por separado

  Nota: IBM Data Server Driver para ODBC y CLI se incluye en los paquetes de cliente listados:

  – IBM Data Server Client
  – IBM Data Server Runtime Client
  – IBM Data Server Driver Package
- El comportamiento del controlador puede modificarse fuera de una aplicación Rails mediante palabras clave CLI.

  Determinados comportamientos transaccionales pueden modificarse fuera de la aplicación Rails mediante estas palabras clave CLI. Por ejemplo, pueden utilizarse palabras clave CLI para establecer el esquema actual o modificar elementos transaccionales tales como la desactivación del comportamiento de confirmación automática. Para obtener más información sobre las palabras clave CLI, consulte los enlaces listados:

  Para la Versión 9: http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.apdv.cli.doc/doc/r0007964.htm

  Para la Versión 9.5: http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.apdv.cli.doc/doc/r0007964.html

  Para la Versión 9.7: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.apdv.cli.doc/doc/r0007964.html
- Cualquier recopilación de diagnósticos requiere el rastreo del controlador CLI.

  Dado que todas las peticiones realizadas a través de IBM_DB se implementan mediante IBM Data Server Driver para ODBC y CLI, el recurso de rastreo de CLI puede identificar los problemas de las aplicaciones que utilicen el adaptador y controlador IBM_DB.

  Un rastreo de CLI captura todas las llamadas de API efectuadas por una aplicación a IBM Data Server Driver para ODBC y CLI (incluidos todos los

parámetros de entrada) y captura todos los valores devueltos desde el controlador a la aplicación. Es un rastreo de interfaz que captura cómo interactúa una aplicación con IBM Data Server Driver para ODBC y CLI y proporciona información sobre el funcionamiento interno del controlador.

El controlador IBM_DB y el adaptador Rails no están soportados en JRuby

El adaptador IBM_DB no está soportado en JRuby.

El adaptador IBM_DB no está soportado en JRuby porque (como se indica en el Wiki de JRuby, "Getting Started") muchas gemas funcionarán correctamente en JRuby, aunque algunas gemas construyen bibliotecas C nativas como parte de su proceso de instalación. Estas gemas no funcionarán en JRuby salvo que cuenten también con un equivalente Java para la biblioteca nativa. Para obtener más información, consulte http://kenai.com/projects/jruby/pages/GettingStarted.

El adaptador IBM_DB se basa en el controlador IBM_DB Ruby (extensión C) y en IBM Data Server Driver para ODBC y CLI para acceder a bases de datos en servidores de datos de IBM. Como método alternativo, puede utilizar la implementación normal de C de Ruby o bien utilizar JDBC_adapter para acceder a las bases de datos.

Adaptador ActiveRecord-JDBC frente a adaptador IBM_DB

La actualización 0.6.0 y posterior de la gema IBM_DB ofrece una gestión ligeramente distinta del uso de comillas en los valores numéricos necesarios.

Aunque la versión anterior del adaptador intentaba filtrar el uso de comillas en valores numéricos para cumplir las expectativas de los servidores de datos DB2 en plataformas diferentes, la nueva implementación sustituye la solución alternativa por un arreglo permanente en el cliente de servidor de datos de IBM. Esto no sólo habilita servidores de datos de IBM entre plataformas, sino que proporciona una gestión más fiable de todas las API de Rails que puedan escapar al filtrado anterior. La solución alternativa proporcionada por la versión anterior del adaptador es, por su propia naturaleza, muy frágil, a causa de los desarrollos fluidos de los componentes de la infraestructura de Rails (ActiveRecord). También se sabe que ciertas API de Rails consiguen escapar del filtrado de estos métodos superados, por lo que la solución alternativa utilizada en el adaptador ActiveRecord-JDBC puede necesitar que se gestionen algunos casos adicionales.

El entorno de ejecución de JRuby no se beneficia de este mismo arreglo, a causa de su interacción interna específica con los servidores de datos. IBM DB2 para IBM i no muestra este problema (arreglado en las versiones V5R3 y V5R4) y lo mismo ocurre respecto a IBM Informix. Por el momento, hasta que JRuby y el adaptador ActiveRecord-JDBC maduren, la mejor alternativa consiste en utilizar el "Ruby clásico" (implementación de C) y el adaptador/controlador IBM_DB. También se puede tener en cuenta un arreglo en el adaptador ActiveRecord-JDBC, que puede emular la gestión anterior que proporcionaba el adaptador IBM_DB.

Consideraciones acerca del tamaño de almacenamiento dinámico con DB2 en Rails

Las aplicaciones Rails en DB2 requieren que el parámetro de configuración de base de datos **applheapsz** esté establecido en valores superiores a 1024.

Debe establecer este parámetro para cada base de datos para la que se ejecutarán aplicaciones Rails en DB2. Utilice el mandato `db2 update db cfg` para actualizar el parámetro **applheapsz**:

```
db2 update db cfg for nombre_basedatos using APPLHEAPSZ 1024
```

Para activar este parámetro, debe reiniciar la instancia de DB2.

Apéndice A. Visión general de la información técnica de DB2

La información técnica de DB2 está disponible en diversos formatos a los que se puede acceder de varias maneras.

La información técnica de DB2 está disponible a través de las herramientas y los métodos siguientes:

- DB2Centro de información
  - Temas (Tareas, concepto y temas de consulta)
  - Programas de ejemplo
  - Guías de aprendizaje
- Manuales de DB2
  - Archivos PDF (descargables)
  - Archivos PDF (desde el DVD con PDF de DB2)
  - Manuales en copia impresa
- Ayuda de la línea de mandatos
  - Ayuda de mandatos
  - Ayuda de mensajes

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 conforme pasen a estar 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, documentos técnicos y publicaciones IBM 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 responderlos directamente. 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 Soporte 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.

Biblioteca técnica de DB2 en copia impresa o 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/e-business/linkweb/publications/servlet/pbi.wss. Los manuales de DB2 Versión 10.1 en inglés y las versiones traducidas en formato PDF se pueden descargar del sitio web www.ibm.com/support/docview.wss?rs=71&uid=swg2700947.

Aunque las tablas identifican los manuales en copia impresa disponibles, puede que dichos manuales no estén disponibles en su país o región.

El número de documento se incrementa cada vez que se actualiza un manual. Asegúrese de que lee la versión más reciente de los manuales, tal como aparece a continuación:

Nota: El *Centro de información de DB2* se actualiza con más frecuencia que los manuales en PDF o impresos.

Tabla 16. Información técnica de DB2

Nombre	Número de documento	Copia impresa disponible	Última actualización
Consulta de las API administrativas	SC11-8067-00	Sí	Abril de 2012
Rutinas y vistas administrativas	SC11-8068-00	No	Abril de 2012
Call Level Interface Guide and Reference Volume 1	SC27-3866-00	Sí	Abril de 2012
Call Level Interface Guide and Reference Volume 2	SC27-3867-00	Sí	Abril de 2012
Consulta de mandatos	SC11-8069-00	Sí	Abril de 2012
Database Administration Concepts and Configuration Reference	SC27-3871-00	Sí	Abril de 2012
Data Movement Utilities Guide and Reference	SC27-3869-00	Sí	Abril de 2012
Database Monitoring Guide and Reference	SC27-3887-00	Sí	Abril de 2012
Data Recovery and High Availability Guide and Reference	SC27-3870-00	Sí	Abril de 2012
Database Security Guide	SC27-3872-00	Sí	Abril de 2012
Guía y consulta de DB2 Workload Management	SC11-8079-00	Sí	Abril de 2012
Developing ADO.NET and OLE DB Applications	SC27-3873-00	Sí	Abril de 2012
Developing Embedded SQL Applications	SC27-3874-00	Sí	Abril de 2012
Desarrollo de aplicaciones Java	SC11-8065-00	Sí	Abril de 2012

Tabla 16. Información técnica de DB2 (continuación)

Nombre	Número de documento	Copia impresa disponible	Última actualización
Desarrollo de aplicaciones Perl, PHP, Python y Ruby on Rails	SC11-8066-00	No	Abril de 2012
Developing User-defined Routines (SQL and External)	SC27-3877-00	Sí	Abril de 2012
Getting Started with Database Application Development	GI13-2046-00	Sí	Abril de 2012
Iniciación a la instalación y administración de DB2 en Linux y Windows	GI13-1946-00	Sí	Abril de 2012
Globalization Guide	SC27-3878-00	Sí	Abril de 2012
Instalación de servidores DB2	GC11-8073-00	Sí	Abril de 2012
Instalación de clientes de IBM Data Server	GC11-8074-00	No	Abril de 2012
Consulta de mensajes Volumen 1	SC11-8079-00	No	Abril de 2012
Consulta de mensajes Volumen 2	SC11-8080-00	No	Abril de 2012
Net Search Extender Guía de administración y del usuario	SC11-8082-00	No	Abril de 2012
Partitioning and Clustering Guide	SC27-3882-00	Sí	Abril de 2012
pureXML Guide	SC27-3892-00	Sí	Abril de 2012
Spatial Extender Guía del usuario y manual de consulta	SC11-8081-00	No	Abril de 2012
SQL Procedural Languages: Application Enablement and Support	SC27-3896-00	Sí	Abril de 2012
Consulta de SQL - Volumen 1	SC11-8070-00	Sí	Abril de 2012
Consulta de SQL - Volumen 2	SC11-8071-00	Sí	Abril de 2012
Guía de Text Search	SC11-8083-00	Sí	Abril de 2012
Troubleshooting and Tuning Database Performance	SC27-3889-00	Sí	Abril de 2012
Actualización a DB2 Versión 10.1	SC11-8072-00	Sí	Abril de 2012
Novedades en DB2 Versión 10.1	SC11-8078-00	Sí	Abril de 2012
XQuery Reference	SC27-3893-00	No	Abril de 2012

Tabla 17. Información técnica específica de DB2 Connect

Nombre	Número de documento	Copia impresa disponible	Última actualización
DB2 Connect Instalación y configuración de DB2 Connect Personal Edition	SC11-8075-00	Sí	Abril de 2012
DB2 Connect Instalación y configuración de servidores DB2 Connect	SC11-8076-00	Sí	Abril de 2012
Guía del usuario de DB2 Connect	SC11-8077-00	Sí	Abril de 2012

Visualización de la ayuda para estados de SQL desde el procesador de línea de mandatos

Los productos DB2 devuelven 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 iniciar 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.

Acceso a diferentes versiones del Centro de información de DB2

La documentación correspondiente a otras versiones de los productos DB2 se encuentra en otros centros de información en ibm.com.

Acerca de esta tarea

Para los temas de DB2 Versión 10.1, el URL del *Centro de información de DB2* es http://publib.boulder.ibm.com/infocenter/db2luw/v10r1.

Para los temas de DB2 Versión 9.8, el URL del *Centro de información de DB2* es http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/.

Para los temas de DB2 Versión 9.7, el URL del *Centro de información de DB2* es http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/.

Para los temas de DB2 Versión 9.5, el URL del *Centro de información de DB2* es http://publib.boulder.ibm.com/infocenter/db2luw/v9r5.

Para los temas de DB2 Versión 9.1, el URL del *Centro de información de DB2* es http://publib.boulder.ibm.com/infocenter/db2luw/v9/.

Para los temas de DB2 Versión 8, vaya al URL del *Centro de información de DB2* en el sitio: http://publib.boulder.ibm.com/infocenter/db2luw/v8/.

Actualización del Centro de información de DB2 instalado en el sistema o en el servidor de intranet

El Centro de información de DB2 instalado en local se debe actualizar periódicamente.

Antes de empezar

Ya debe haber un Centro de información de DB2 Versión 10.1 instalado. Para obtener información adicional, consulte el tema "Instalación del Centro de información de DB2 utilizando el Asistente de instalación de DB2" en la publicación *Instalación de servidores DB2*. Todos los requisitos previos y las restricciones aplicables a la instalación del Centro de información se aplican también a la actualización del Centro de información.

Acerca de esta tarea

Un Centro de información de DB2 existente se puede actualizar automática o manualmente:

- Las actualizaciones automáticas actualizan las funciones y los idiomas del Centro de información existentes. Una ventaja de las actualizaciones automáticas es que el Centro de información deja de estar disponible durante un período de tiempo más breve a cuando se realiza la actualización manual. Además, la ejecución de las actualizaciones automáticas se puede configurar como parte de otros trabajos de proceso por lotes que se ejecutan periódicamente.
- Las actualizaciones manuales se pueden utilizar para actualizar las funciones y los idiomas existentes del Centro de información. Las actualizaciones automáticas reducen el tiempo de inactividad durante el proceso de actualización. Sin embargo, debe utilizar el proceso manual cuando desee añadir funciones o idiomas. Por ejemplo, un Centro de información en local se instaló inicialmente tanto en inglés como en francés, y ahora se desea instalar el idioma alemán. Con la actualización manual, se instalará el alemán y se actualizarán además las funciones y los idiomas existentes del Centro de información. No obstante, la actualización manual requiere que el usuario detenga, actualice y reinicie manualmente el Centro de información. El Centro de información no está disponible durante todo el proceso de actualización. En el proceso de actualización automática, el Centro de información incurre en una interrupción de servicio para reiniciar el Centro de información solo después de la actualización.

Este tema detalla el proceso de las actualizaciones automáticas. Para conocer las instrucciones para la actualización manual, consulte el tema "Actualización manual del Centro de información de DB2 instalado en el sistema o en el servidor de intranet".

Procedimiento

Para actualizar automáticamente el Centro de información de DB2 instalado en el sistema o en el servidor de Intranet:

1. En sistemas operativos Linux,

a. Navegue hasta la vía de acceso en la que está instalado el Centro de información. Por omisión, el Centro de información de DB2 se instala en el directorio `/opt/ibm/db2ic/V10.1`.
b. Navegue desde el directorio de instalación al directorio `doc/bin`.
c. Ejecute el script `update-ic`:
   ```
   update-ic
   ```

2. En sistemas operativos Windows,
   a. Abra una ventana de mandatos.
   b. Navegue hasta la vía de acceso en la que está instalado el Centro de información. Por omisión, el Centro de información de DB2 se instala en el directorio `<Archivos de programa>\IBM\DB2 Information Center\Versión 10.1`, siendo `<Archivos de programa>` la ubicación del directorio Archivos de programa.
   c. Navegue desde el directorio de instalación al directorio `doc\bin`.
   d. Ejecute el archivo `update-ic.bat`:
      ```
      update-ic.bat
      ```

Resultados

El Centro de información de DB2 se reinicia automáticamente. Si hay actualizaciones disponibles, el Centro de información muestra los temas nuevos y actualizados. Si no había actualizaciones del Centro de información disponibles, se añade un mensaje al archivo de anotaciones cronológicas. El archivo de anotaciones cronológicas está ubicado en el directorio `doc\eclipse\configuration`. El nombre del archivo de anotaciones cronológicas es un número generado aleatoriamente. Por ejemplo, `1239053440785.log`.

Actualización manual del Centro de información de DB2 instalado en el sistema o en el servidor de intranet

Si ha instalado localmente el Centro de información de DB2 localmente, puede obtener e instalar actualizaciones de la documentación de IBM.

Acerca de esta tarea

Para actualizar manualmente el *Centro de información de DB2* instalado localmente es preciso que:

1. Detenga el *Centro de información de DB2* en el sistema, y reinicie 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 al usuario aplicar las actualizaciones. La versión de estación de trabajo del Centro de información de DB2 siempre se ejecuta en modalidad autónoma.
2. Utilice la función Actualizar para ver qué actualizaciones están disponibles. Si hay actualizaciones que debe instalar, puede utilizar la función Actualizar para obtenerlas y actualizarlas.

   Nota: Si su entorno requiere la instalación de actualizaciones del *Centro de información de DB2* en una máquina no conectada a Internet, duplique el sitio de actualizaciones en un sistema de archivos local utilizando una máquina que esté conectada a Internet y tenga instalado el *Centro de información de DB2*. Si muchos usuarios en la red van a instalar las actualizaciones de la documentación, puede reducir el tiempo necesario para realizar las

actualizaciones duplicando también el sitio de actualizaciones localmente y creando un proxy para el sitio de actualizaciones.
Si hay paquetes de actualización disponibles, utilice la característica Actualizar para obtener los paquetes. Sin embargo, la característica Actualizar sólo está disponible en modalidad autónoma.

3. Detenga el Centro de información autónomo y reinicie el *Centro de información de DB2* en su equipo.

Nota: En Windows 2008 y Windows Vista (y posterior), los mandatos listados más abajo deben ejecutarse como administrador. Para abrir un indicador de mandatos o una herramienta gráfica con privilegios de administrador completos, pulse con el botón derecho del ratón el atajo y, a continuación, seleccione **Ejecutar como administrador**.

Procedimiento

Para actualizar el *Centro de información de DB2* instalado en el sistema o en el servidor de Intranet:

1. Detenga el *Centro de información de DB2*.
   - En Windows, pulse **Inicio** > **Panel de control** > **Herramientas administrativas** > **Servicios**. A continuación, 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/db2icdv10 stop
     ```
2. Inicie el Centro de información en modalidad autónoma.
   - En Windows:
     a. Abra una ventana de mandatos.
     b. Navegue hasta la vía de acceso en la que está instalado el Centro de información. Por omisión, el *Centro de información de DB2* se instala en el directorio *Archivos_de_programa*\IBM\DB2 Information Center\Versión 10.1, siendo *Archivos_de_programa* la ubicación del directorio Archivos de programa.
     c. Navegue desde el directorio de instalación al directorio doc\bin.
     d. Ejecute el archivo help_start.bat:
        ```
        help_start.bat
        ```
   - En Linux:
     a. Navegue hasta la vía de acceso en la que está instalado el Centro de información. Por omisión, el *Centro de información de DB2* se instala en el directorio /opt/ibm/db2ic/V10.1.
     b. Navegue desde el directorio de instalación al directorio doc/bin.
     c. Ejecute el script help_start:
        ```
        help_start
        ```

   Se abre el navegador Web por omisión de los sistemas para visualizar el Centro de información autónomo.
3. Pulse en el botón **Actualizar** (). (JavaScript debe estar habilitado en el navegador.) 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 instalación, compruebe las selecciones que desee instalar y, a continuación, pulse **Instalar actualizaciones**.
5. Cuando finalice el proceso de instalación, pulse **Finalizar**.

6. Detenga el Centro de información autónomo:
   - En Windows, navegue hasta el directorio doc\bin del directorio de instalación y ejecute el archivo help_end.bat:
     ```
     help_end.bat
     ```
     Nota: El archivo help_end de proceso por lotes contiene los mandatos necesarios para detener 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 detener help_start.bat.
   - En Linux, navegue hasta el directorio doc/bin del directorio de instalación y ejecute el script help_end:
     ```
     help_end
     ```
     Nota: El script help_end contiene los mandatos necesarios para detener sin peligro los procesos que se iniciaron mediante el script help_start. No utilice ningún otro método para detener el script help_start.
7. Reinicie el *Centro de información de DB2*.
   - En Windows, pulse **Inicio** > **Panel de control** > **Herramientas administrativas** > **Servicios**. A continuación, pulse con el botón derecho del ratón en el servicio **Centro de información de DB2** y seleccione **Iniciar**.
   - En Linux, especifique el mandato siguiente:
     ```
     /etc/init.d/db2icdv10 start
     ```

Resultados

El *Centro de información de DB2* actualizado muestra los temas nuevos y actualizados.

Guías de aprendizaje de DB2

Las guías de aprendizaje de DB2 le ayudan a conocer diversos aspectos de productos de base de datos 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/db2luw/v10r1/.

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.

"pureXML" en *pureXML Guide*
: Configure una base de datos DB2 para almacenar datos XML y realizar operaciones básicas con el almacén de datos XML nativos.

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 de base de datos DB2.

Documentación de DB2

Puede encontrar información sobre la resolución de problemas en la publicación *Troubleshooting and Tuning Database Performance* o en la sección sobre conceptos fundamentales sobre bases de datos del *Centro de información de DB2*, que contiene:

- Información sobre cómo aislar e identificar problemas con programas de utilidad y herramientas de diagnóstico de DB2.
- Soluciones a algunos de los problemas más comunes.
- Consejo para ayudarle a resolver problemas que podría encontrar en los productos de base de datos DB2

Portal de Soporte de IBM

Consulte el portal de soporte de IBM 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), fixpacks y otros recursos. Puede buscar en esta base de conocimiento para encontrar posibles soluciones a los problemas.

Acceda al portal de Soporte de IBM en el sitio http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_for_Linux,_UNIX_and_Windows

Términos y condiciones

Los permisos para utilizar estas publicaciones se otorgan sujetos a los siguientes términos y condiciones.

Aplicación: Además de las condiciones de uso del sitio web de IBM, se aplican estos 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.

Derechos: 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.

Marcas registradas de IBM: IBM, el logotipo de IBM e ibm.com son marcas registradas de International Business Machines Corp., que se han registrado en muchas otras jurisdicciones. Otros nombres de productos y servicios pueden ser marcas registradas de IBM o de otras empresas. Puede consultarse en línea una lista actualizada de las marcas registradas de IBM en la web en www.ibm.com/legal/copytrade.shtml.

Apéndice B. Avisos

Esta información ha sido desarrollada para productos y servicios que se ofrecen en Estados Unidos de América. La información acerca de productos que no son IBM se basa en la información disponible cuando se publicó este documento por primera vez y está sujeta a cambio.

Es posible que IBM no comercialice en otros 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 juegos de caracteres de doble byte (DBCS), puede ponerse en contacto con el Departamento de Propiedad Intelectual de IBM de su país o escribir a:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 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 de esos sitios web no forma parte de la información del presente producto de IBM y la utilización de esos sitios web se realiza bajo la responsabilidad del usuario.

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
U59/3600
3600 Steeles Avenue East
Markham, Ontario L3R 9Z7
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 contiene 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. Los programas de ejemplo se proporcionan "TAL CUAL", sin ningún tipo de garantía. IBM no se hará responsable de los daños derivados de la utilización que haga el usuario de los programas de ejemplo.

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

IBM, el logotipo de IBM e ibm.com son marcas registradas de International Business Machines Corp., que se han registrado en muchas otras jurisdicciones. Otros nombres de productos y servicios pueden ser marcas registradas de IBM o de otras empresas. La lista actual de marcas registradas de IBM está disponible en la web, en “Copyright and trademark information”, en la dirección www.ibm.com/legal/copytrade.shtml.

Los siguientes términos son marcas registradas de otras empresas.

- Linux es una marca registrada de Linus Torvalds en los Estados Unidos y/o en otros países.
- Java y todos los logotipos y marcas registradas basadas en Java son marcas registradas de Oracle, sus filiales o ambos.
- UNIX es una marca registrada de The Open Group en los Estados Unidos y/o en otros países.
- Intel, el logotipo de Intel, Intel Inside, el logotipo de Intel Inside, Celeron, Intel SpeedStep, Itanium y Pentium son marcas registradas de Intel Corporation o de sus empresas subsidiarias en Estados Unidos y en otros países.
- Microsoft, Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en los Estados Unidos y/o en otros países.

Otros nombres de empresas, productos o servicios, pueden ser marcas registradas o marcas de servicio de otras empresas.

Índice

E

F

G

I

J

M

N

O

P

R

S

T

V

SC11-8066-00

Spine information:

IBM DB2 10.1 para Linux, UNIX y Windows

Desarrollo de aplicaciones Perl, PHP, Python y Ruby on Rails