IBM DB2 10.5
para Linux, UNIX y Windows

Desarrollo de aplicaciones RDF para servidores de datos IBM

SC11-8357-00

IBM DB2 10.5
para Linux, UNIX y Windows

Desarrollo de aplicaciones RDF para servidores de datos IBM

SC11-8357-00

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 73.

Nota de edición

Este documento contiene información propiedad de IBM. Se proporciona según un acuerdo de licencia y está protegido por la ley de la propiedad intelectual. La información contenida en esta publicación no incluye ninguna garantía de producto, por lo que ninguna declaración proporcionada en este manual deberá interpretarse como tal.

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).

Cuando envía información a IBM, está otorgando a IBM el derecho no exclusivo de utilizar o distribuir la información de cualquier forma que considere adecuada sin incurrir por ello a ninguna obligación para con usted.

© Copyright IBM Corporation 2013.

Contenido

© Copyright IBM Corp. 2013

Parte 1. Desarrollo de la aplicación RDF para servidores de datos de IBM

Resource Description Framework (RDF) es una familia de especificaciones W3 que se puede utilizar como infraestructura estándar de intercambio de datos para la información de modelado. Las aplicaciones pueden almacenar y consultar datos RDF en bases de datos IBM® DB2 10.5 Enterprise Server Edition (DB2 Enterprise Server Edition).

RDF emplea identificadores uniformes de recursos (URI) para crear una relación entre datos como triplete; por ejemplo, en forma de expresiones sujeto-predicado-objeto. Se pueden enlazar, exponer y compartir datos estructurados y semiestructurados entre diferentes aplicaciones utilizando este sencillo modelo.

Un almacenamiento RDF en el servidor de bases de datos DB2 es un conjunto de tablas de usuario contenido en un esquema de base de datos que almacena un conjunto de datos RDF. Cada conjunto de estas tablas lleva asociado un nombre de almacenamiento exclusivo. Cada almacenamiento RDF tiene una tabla que contiene metadatos para el almacenamiento. Esta tabla tiene el mismo nombre que el del almacenamiento.

Puede cargar datos en las tablas de usuario mediante los mandatos del programa de utilidad RDF o las API de Java™. Debe disponer de los permisos de lectura y grabación adecuados en estos conjuntos de tablas. Las API de Java soportadas son las API de estructura JENA. Los programas de utilidad RDF o las API de DB2 están soportados para el software DB2 Versión 9.7 y versiones posteriores.

Las aplicaciones RDF utilizan el lenguaje de consultas SPARQL para recuperar datos en las bases de datos DB2.

No se admite RDF en los entornos de bases de datos particionadas.

© Copyright IBM Corp. 2013

Capítulo 1. Referencias y recursos relacionados con RDF

Hay disponibles muchos recursos para ayudarle a desarrollar aplicaciones RDF que acceden a servidores de datos de IBM.

Tabla 1. Referencias y recursos relacionados con RDF

Recurso RDF	**Enlace de referencia**
RDF Primer	http://www.w3.org/TR/2004/REC-rdf-primer-20040210/
Lenguaje de consultas SPARQL	http://www.w3.org/TR/rdf-sparql-query/
API de gráficos y modelos JENA	http://jena.sourceforge.net/tutorial/RDF_API/
IBM RDF Javadoc	../javadoc/index.html
Parte 1 de la guía de aprendizaje de desarrollo de aplicaciones RDF (creación y mantenimiento de almacenamientos RDF)	http://www.ibm.com/developerworks/data/tutorials/dm-1205rdfdb210/index.html

© Copyright IBM Corp. 2013

Capítulo 2. Tablas de almacenamiento RDF

Un almacenamiento RDF se compone de varias tablas. Estas tablas contienen metadatos sobre el almacenamiento RDF o los datos de usuario.

Tablas de metadatos

Las tablas siguientes contienen metadatos sobre el almacenamiento RDF:

- Una tabla de metadatos que tiene el mismo nombre que el almacenamiento RDF.
- La tabla de metadatos System Predicates, que almacena información sobre predicados RDF que puede utilizar para filtrar con mayor precisión los resultados de una consulta SPARQL.
- La tabla Basic Statistics, que almacena estadísticas sobre los datos del almacenamiento RDF.
- La tabla de estadísticas Top K, que almacena información sobre los datos RDF menos selectivos del almacén.

Tablas de datos

Las tablas siguientes almacenan datos RDF si el valor de los datos no supera una longitud de caracteres determinada:

- La tabla Direct Primary almacena los tripletes de RDF y el gráfico asociado, indexados por sujeto. Los predicados y objetos correspondientes a un sujeto se almacenan en pares de columnas en esta tabla. Un predicado concreto puede aparecer en cualquiera de las tres columnas de esta tabla. El objeto para ese predicado se almacena en la columna de objeto correspondiente del par predicado-objeto.
- La tabla Direct Secondary almacena los tripletes de RDF que comparten el sujeto y el predicado en un gráfico RDF. Estos tripletes sólo tienen un identificador de marcador de posición en la tabla Direct Primary.
- La tabla Reverse Primary almacena los tripletes de RDF y el gráfico asociado, indexados por objeto. Los predicados y sujetos correspondientes a un objeto se almacenan en pares de columnas en esta tabla. Un predicado determinado puede aparecer en cualquiera de las tres columnas de esta tabla, y el sujeto de ese predicado, en la columna de sujeto correspondiente de ese par.
- La tabla Reverse Secondary almacena los tripletes de RDF que comparten el objeto y el predicado en un gráfico RDF. Estos tripletes solo tienen un marcador de posición en la tabla Reverse Primary.
- La tabla Datatypes almacena la correlación de los valores de entero internos para los tipos de datos SPARQL, los tipos de datos definidos por el usuario y los códigos de lenguaje.

Si el valor de un sujeto, predicado, objeto o gráfico RDF supera una longitud de caracteres determinada, las cinco tablas mencionadas anteriormente almacenarán un identificador de marcador de posición solamente. El valor real se almacena en la tabla Long Strings.

Objetos de base de datos administrativa de RDF

RDF dispone tanto de funciones como de un objeto de tarea del planificador para gestionar un almacenamiento RDF.

© Copyright IBM Corp. 2013

Objetos de base de datos administrativa

RDF para DB2 tiene objetos de base de datos administrativa, como se indica a continuación:

- Una UDF externa basada en Java denominada <*nombre_almacenamiento*>_RDF_REGEX da soporte al operador regex en SPARQL. Deben otorgarse los permisos adecuados para utilizar esta UDF.

  En el fixpack 2 y en los fixpacks posteriores de DB2 versión 10.1, <*nombre_almacenamiento*>_RDF_REGEX UDF ya no tiene soporte para la funcionalidad de expresión regular. En su lugar, utilice la función pureXML `fn:matches()` para expresión regular
- Un procedimiento almacenado de SQL denominado <*nombre_almacenamiento*>_T3_STATS recopila las estadísticas básicas y topK para un almacenamiento RDF.
- Se utiliza una tarea de planificador administrativo denominada <*SCHEMANAME*>_<*STORENAME*>_Planificador para planificar el intervalo de las actualizaciones de las estadísticas topK y básicas del almacenamiento.

Capítulo 3. Control de acceso para almacenamientos RDF

Existen dos tipos de control de acceso para los almacenamientos RDF de DB2.

Control de acceso general

Puede utilizar los permisos de nivel de tabla de la base de datos DB2 para controlar el acceso al almacenamiento RDF completo.

Control de acceso de nivel de gráfico RDF

El control de acceso de nivel de gráfico RDF proporciona un control de acceso más preciso en el nivel de gráfico RDF. Puede controlar de forma selectiva los gráficos RDF a los que los usuarios tendrán acceso en el almacenamiento RDF, en lugar de la totalidad del conjunto de datos RDF.

Con el control de acceso de nivel de gráfico RDF, se utilizan los tripletes RDF que haya en un gráfico para determinar si un usuario tiene acceso al gráfico RDF o no. En el momento de la creación del almacenamiento RDF, el usuario debe especificar qué predicados RDF se utilizarán para controlar el acceso al gráfico RDF.

La aplicación del control de acceso durante el tiempo de ejecución (mediante consultas SPARQL) se puede delegar al motor de DB2. Como alternativa, puede utilizarse en el SQL generado por el generador de SQL de almacenamiento RDF de DB2.

Si opta por que el control de acceso lo aplique el motor de DB2, debe utilizar la función de control de acceso preciso del software DB2, para especificar las reglas de control de acceso.

Si opta por que el control de acceso lo aplique el generador de SQL de almacenamiento RDF, la aplicación tiene que pasar las restricciones de forma adicional para aplicarlas en el contexto de `QueryExecution`. En este caso solamente se da soporte a un conjunto limitado de operadores y operandos:

- Creando un almacenamiento RDF con soporte de control de acceso de nivel de gráfico
- Aplicando el control de acceso de nivel de gráfico mediante el generador de SQL de almacenamiento RDF
- Aplicando el control de acceso de nivel de gráfico mediante el motor de DB2

© Copyright IBM Corp. 2013

Capítulo 4. Almacenamientos RDF por omisión y optimizado

Se utilizan dos tipos de almacenamiento RDF con las bases de datos DB2. Uno es el almacenamiento RDF por omisión, mientras que el otro es el almacenamiento RDF optimizado.

Almacenamientos RDF por omisión

Este esquema base se utiliza cuando no se sabe nada sobre los datos RDF que se almacenarán o cuando no se dispone de ejemplos adecuados. Los almacenamientos RDF por omisión utilizan un número por omisión de columnas en las tablas Direct Primary y Reverse Primary. El almacenamiento por omisión se utiliza al comenzar con un conjunto de datos RDF nuevo, sobre el que no se sabe nada. En el almacenamiento por omisión, se utiliza la generación aleatoria (hashing) para determinar las columnas a las que irán los predicados y los objetos en las tablas Direct Primary y Reverse Primary.

Cree un almacenamiento RDF por omisión cuando no tenga datos de ejemplo existentes del conjunto de datos RDF con los que el software DB2 pueda calcular la coexistencia de los predicados.

Almacenamientos RDF optimizados

Si se dispone de suficientes datos representativos del conjunto de datos RDF, se puede crear un esquema más optimizado para las tablas Direct Primary y Reverse Primary. Este esquema optimizado se logra aprovechando que los predicados RDF se correlacionan. Por ejemplo, la edad y el número de la seguridad social coexisten como predicados de Persona, y las oficinas centrales y los ingresos coexisten como predicados de Empresa, pero la edad y los ingresos nunca aparecen juntos en ninguna entidad.

Cree un almacenamiento RDF optimizado cuando tenga datos existentes o datos de ejemplo para el conjunto de datos RDF en con los que DB2 calculará la correlación de predicados para asignar los predicados a las columnas de forma inteligente.

Ventajas de los almacenamientos RDF optimizados

La correlación de predicados se utiliza para reducir de forma drástica, y a veces para eliminar por completo, la aleatoriedad de la función de generación aleatoria (hashing) utilizada en los almacenamientos por omisión. Por lo tanto, en los almacenamientos por omisión, se pueden dar colisiones entre los predicados debido a la falta de información sobre la correlación de predicados, y esto puede provocar que se utilicen más filas en la tabla de las que son necesarias realmente. Las filas de más podrían hacer que las uniones entre tablas sean menos eficientes de lo que deberían ser.

La indexación de los predicados se puede realizar con facilidad, ya que, por lo general, un predicado dado se puede confinar a una única columna. Además, los predicados que no coexistan pueden asignarse a una sola columna, con lo que un único índice de DB2 puede indexar varios predicados.

© Copyright IBM Corp. 2013

Capítulo 5. Vista central de los almacenamientos RDF

Ahora, a partir del fixpack 2 y en los fixpacks posteriores de DB2 versión 10.1, Resource Description Framework (RDF) enumera en una tabla todos los almacenamientos RDF presentes en una determinada base de datos. Consulte la tabla SYSTOOLS.RDFSTORES para ver todos los almacenamientos RDF.

La tabla SYSTOOLS.RDFSTORES se crea la primera vez que se ejecuta la API y el mandato **createrdfstore** o **createrdfsoreandloader** para una base de datos.

Tabla 2. Esquema de la tabla SYSTOOLS.RDFSTORES

Nombre de columna	Tipo de datos	Con posibilidad de nulos	Descripción
STORENAME	VARCHAR(256)	NO	Nombre del almacenamiento RDF
SCHEMANAME	VARCHAR(256)	NO	Nombre del esquema del almacenamiento RDF
STORETABLE	VARCHAR(256)	NO	Nombre de la tabla de metadatos del almacenamiento RDF
Clave primaria		NO	Clave primaria

Para enumerar todos los almacenamientos RDF y sus esquemas correspondientes en una base de datos, ejecute la consulta siguiente:

```
SELECT storeName, schemaName FROM SYSTOOLS.RDFSTORES
```

Se devuelve la salida de ejemplo siguiente:

```
STORENAME       SCHEMANAME
--------------------------
STAFFING   DB2ADMIN
SAMPLSTORE      DB2ADMIN

2 registro(s) seleccionado(s).
```

© Copyright IBM Corp. 2013

Capítulo 6. Configuración de un entorno RDF

Configure su entorno para poder utilizar API y mandatos RDF de DB2.

Emisión de mandatos RDF mediante programas de utilidad de línea de mandatos

Los programas de utilidad de línea de mandatos RDF de DB2 se encuentran en el directorio *<vía_acceso_instalación>*/sqllib/rdf/bin. Inicie los programas de utilidad desde este directorio con un indicador de mandatos de DB2.

Después de instalar el servidor de bases de datos DB2, lleve a cabo las tareas siguientes para usar los programas de utilidad de la línea de mandatos RDF de DB2:

1. Descargue la versión 2.8.5 del paquete ARQ de http://sourceforge.net/projects/jena/files/ARQ/ARQ-2.8.5/"http://sourceforge.net/projects/jena/files/ARQ/ARQ-2.8.5/".

   Copie los archivos JAR desde la carpeta lib del paquete ARQ en el directorio *<vía_acceso_instalación>*/SQLLIB/rdf/lib.

   Nota: Puede omitir la copia de los archivos JAR 'xxx-tests.jar', 'xxx-sources.jar', 'xxx-test-sources.jar'.

   A partir del fixpack 2 de DB2 versión 10.1, use el paquete Apache JENA versión 2.7.3 de http://archive.apache.org/dist/jena/binaries/"http://www.apache.org/dist/jena/binaries/".

   Guarde los archivos JAR de la carpeta lib del paquete Apache JENA en el directorio *<vía_acceso_instalación>*/SQLLIB/rdf/lib.
2. Descargue Commons-logging-1-0-3.jar del proyecto Apache Commons. Coloque este JAR en el directorio *<vía_acceso_instalación>*/SQLLIB/rdf/lib.
3. Abra una línea de mandatos y vaya al directorio *<vía_acceso_instalación>*/SQLLIB/rdf/bin.

   ```
   cd
   "<vía_acceso_instalación>/SQLLIB/rdf/bin"
   ```
4. Añada el controlador JCC de DB2 db2jcc4.jar, del directorio <vía_acceso_instalación>/SQLLIB/java a la variable de entorno de la vía de acceso de la clase, tal como se muestra a:

   ```
   set
   classpath=<vía_acceso_instalación>\SQLLIB\java\db2jcc4.jar;%classpath%
   ```

Ahora puede ejecutar los programas de utilidad de línea de mandatos RDF de DB2 en este indicador de mandatos.

RDF de DB2 en un entorno de desarrollo de aplicaciones

Los archivos JAR RDF de DB2 deben añadirse a la vía de acceso de la aplicación, junto con el archivo JAR siguiente:

- Los archivos JAR que dependen de JENA
- El archivo Commons-logging-1-0-3.jar
- El controlador JCC de DB2 JCC (db2jcc4.jar)

© Copyright IBM Corp. 2013

Los archivos JAR se encuentran en el directorio <*vía_acceso_instalación*>/sqllib/rdf/lib. Incluyen los siguientes archivos JAR:

- rdfstore.jar
- antlr-3.3-java.jar
- wala.jar

RDF con DB2 versión 9.7

Puede instalar el cliente de DB2 Versión 10.1 y utilizarlo con el servidor de bases de datos DB2 Versión 9.7.

Registre las bibliotecas Java externas que se necesitan para el soporte de RDF. El registro se efectúa ejecutando el script 'rdf/bin/registerrdfudf' en el cliente de DB2 versión 10.1. Este script debe emitirse para cada base de datos DB2 versión 9.7 en la que se están creando almacenamientos RDF. Por ejemplo, emita el mandato siguiente:

```
registerrdfudf <nombrebd> <nombreusuario>
```

donde <*nombrebd*> es una base de datos catalogada en el cliente de DB2 local.

Capítulo 7. Creación de un almacenamiento RDF

Cree un almacenamiento RDF por omisión u optimizado, basándose en los requisitos de desarrollo de aplicaciones. Puede optar por crear un almacenamiento RDF primero y cargar los datos más adelante.

Creación de un almacenamiento RDF por omisión

Puede crear un almacenamiento RDF sin disponer de datos RDF existentes. Esto también se conoce como almacenamiento RDF por omisión.

Antes de empezar

Son necesarios los requisitos previos siguientes:

- Asegúrese de que la base de datos tenga un tamaño de página mínimo de 32 KB.
- Asegúrese de que el parámetro de configuración de base de datos **LOGFILSIZ** está establecido en un valor igual o superior a 20000.

```
db2 UPDATE DATABASE CONFIGURATION FOR <NOMBRE_BD>
USING LOGFILSIZ 20000
```

- Asegúrese de que el espacio de tablas SYSTOOLSPACE existe.

```
CREATE TABLESPACE SYSTOOLSPACE IN IBMCATGROUP
MANAGED BY AUTOMATIC STORAGE EXTENTSIZE 4
```

- Asegúrese de que el ID de autorización tiene los privilegios siguientes:
  - Autorización CREATETAB para el esquema de base de datos seleccionado.
  - Autorización CREATE EXTERNAL ROUTINES.
  - Privilegios de actualización para la tabla SYSTOOLS.ADMINTASKSTATUS.

Configure también la siguiente funcionalidad:

- Establezca el planificador de tareas administrativas en "YES".

```
db2set DB2_ATS_ENABLE=YES
```

- Establezca el parámetro de configuración de base de datos **AUTORUNSTATS** en "ON".

```
db2 UPDATE DB CONFIG USING AUTO_MAINT ON AUTO_TBL_MAINT ON AUTO_RUNSTATS ON
```

- Establezca la agrupación de almacenamientos intermedios en "AUTOMATIC" y asigne un tamaño inicial adecuado.

```
db2 alter bufferpool IBMDEFAULTBP IMMEDIATE SIZE 15000 AUTOMATIC
```

Procedimiento

Utilice las instrucciones siguientes para crear un almacenamiento RDF por omisión:

1. Controle los nombres de tablas de base de datos y espacios de tablas que conformarán el almacenamiento RDF. Cree un archivo de propiedades `objectNames.props` que contenga una lista de las tablas de datos de almacenamiento RDF y los nombres y espacios de tablas correspondientes que desea asignar a las tablas.

   El contenido de un archivo de propiedades de ejemplo `objectNames.props` se muestra en el ejemplo siguiente:

© Copyright IBM Corp. 2013

```
direct_primary_hash=<nombre_tabla_usuario>,<espacio_tablas>
direct_secondary=<nombre_tabla_usuario>,<espacio_tablas>
reverse_primary_hash=<nombre_tabla_usuario>,<espacio_tablas>
reverse_secondary=<nombre_tabla_usuario>,<espacio_tablas>
long_strings=<nombre_tabla_usuario>,<espacio_tablas>
basic_stats=<nombre_tabla_usuario>,<espacio_tablas>
topk_stats=<nombre_tabla_usuario>,<espacio_tablas>
system_predicate=<nombre_tabla_usuario>,<espacio_tablas>
data_type=<nombre_tabla_usuario>,<espacio_tablas>
```

Nota: La utilización del archivo `objectNames.props` para establecer los nombres de tabla es opcional. Sin embargo, si decide no utilizar este archivo de propiedades, en su lugar se utilizarán los nombres generados por el sistema.

2. Emita el mandato **createrdfstore**. Especifique la instancia y esquema de base de datos donde desea crear el almacenamiento RDF. Elija también un nombre lógico para el almacenamiento. Este nombre debe ser exclusivo en todos los almacenamientos RDF del esquema de base de datos.

   Por ejemplo, utilice el mandato siguiente para crear un almacenamiento denominado rdfStore1 en la base de datos DB1 y el esquema db2admin para el sistema principal localhost y el puerto 60000:

   ```
   createrdfstore rdfStore1 -host localhost -port 60000 -db DB1
   -user db2admin -password XXX -schema db2admin
   ```

Creación de un almacenamiento RDF optimizado

Creación de un almacenamiento RDF optimizado mediante API

Para crear un almacenamiento optimizado debe disponer de un conjunto de datos triples representativos. Los datos son necesarios para garantizar que la ocurrencia de predicado se pueda calcular adecuadamente.

Antes de empezar

Si no tiene un conjunto de ejemplo representativo de datos triples, cree un almacenamiento por omisión mediante el mandato **createrdfstore**. Utilice el almacenamiento RDF por omisión hasta que haya recopilado suficientes datos triples representativos basándose en qué ocurrencia de predicado se puede calcular.

Si tiene datos suficientes, utilice el método generatePredicateMappings de la clase Java StoreManager para calcular la correlación de predicado de los tripletes en el almacenamiento RDF predeterminado. Guarde la salida de PredicateMappings para esta API.

Al crear un almacenamiento RDF para un entorno de producción, cree un nuevo almacenamiento RDF optimizado vacío utilizando el método createStoreWithPredicateMappings de la clase Java StoreManager. Proporcione la salida de PredicateMappings que ha obtenido anteriormente.

Procedimiento

Para crear un almacén RDF optimizado, grabe un programa que realice dos pasos clave:

1. Calcula la correlación de predicado de los tripletes en el almacén RDF que contiene los datos representativos. Para calcular la correlación de predicado, utilice el método generatePredicateMappings de la clase Java StoreManager . Guarde la salida de la API `PredicateMappings` .

2. Crea el almacén RDF optimizado utilizando el método createStoreWithPredicateMappings de la clase Java StoreManager . Igual que la entrada, proporcione la salida `PredicateMappings` que obtuvo anteriormente.

Ejemplo

El siguiente programa Java muestra cómo crea un almacén RDF optimizado:

```
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import com.ibm.rdf.store.StoreManager;


/**
 * Este programa muestra cómo crear un almacenamiento RDF optimizado
 * proporcionando la información de correlación de predicados de un almacenamiento
 * ya existente. También puede crear almacenes RDF optimizados para entornos
 * de producción después de recopilar un ciclo con suficientes daros representativos
 *  en un almacén predeterminado.
 */
public class CreateOptimizedStore {

 public static void main(String[] args) throws SQLException,
IOException {

  String currentStoreName = "sample";
  String currentStoreSchema = "db2admin";
  Connection currentStoreConn = null;

  /*
   * Conectar con "currentStore" y generar la correlación de
     * predicado para los tripletes que contiene.
   */
  try {
   Class.forName("com.ibm.db2.jcc.DB2Driver");
   currentStoreConn = DriverManager.getConnection(
    "jdbc:db2://localhost:50000/dbrdf", "db2admin",
     "db2admin");
   currentStoreConn.setAutoCommit(false);
  } catch (ClassNotFoundException e1) {
   e1.printStackTrace();
  }

  /* Especificar el archivo de disco donde la correlación de
     * predicado se almacenará.
     */
  String file = "/predicateMappings.nq";
  BufferedOutputStream predicateMappings = new
BufferedOutputStream(new FileOutputStream(file));

  StoreManager.generatePredicateMappings(currentStoreConn,
    currentStoreSchema, currentStoreName,
predicateMappings);

  predicateMappings.close();

  /**
   * Crear un almacenamiento RDF optimizado utilizando la información
     * de correlación de predicado generada anteriormente.
```

```
   */
  String newOptimizedStoreName = "production";
  String newStoreSchema = "db2admin";
  Connection newOptimizedStoreConn =
DriverManager.getConnection(
    "jdbc:db2://localhost:50000/dbrdf",
"db2admin","db2admin");
  BufferedInputStream inputPredicateMappings =
 new BufferedInputStream(
     new FileInputStream(file));

 StoreManager.createStoreWithPredicateMappings(
newOptimizedStoreConn, newStoreSchema,
newOptimizedStoreName, null, inputPredicateMappings);

   }
}
```

Creación de un almacenamiento RDF optimizado mediante mandatos

En el fixpack 2 y en los fixpacks posteriores de DB2 versión 10.1, puede crear un almacenamiento optimizado a partir de un almacenamiento RDF por omisión utilizando los mandatos de RDF.

Procedimiento

Para crear un almacenamiento RDF optimizado desde el indicador de mandatos:

1. Cree un almacenamiento por omisión utilizando el mandato **createrdfstore**.

   ```
   createrdfstore rdfStore1 -db RDFSAMPL
   -user db2admin -password XXX
   ```

2. Añada datos al almacenamiento utilizando las API de JENA o UPDATE de SPARQL. Utilice este almacenamiento por omisión para recopilar un conjunto de datos de tripletes que se pueden utilizar para calcular las apariciones de predicados.
3. Genere las correlaciones de predicados mediante el mandato **genpredicatemappings**.

   ```
   genPredicateMappings MyStore -db RDFSAMPL -user db2admin
   -password db2admin "C:\MyStore_predicate_mappings.txt"
   ```

4. Cree un almacenamiento optimizado mediante el mandato **createrdfstore**, pasando el parámetro **-predicatemappings**.

   Utilice los predicados generados en el paso anterior como entrada para el parámetro **-predicatemappings**.

   ```
   createrdfstore MyOptimizedStore -db RDFSAMPL
   -user db2admin -password XXX
   -predicatemappings "C:\MyStore_predicate_mappings.txt"
   ```

Resultados

Se ha creado el almacenamiento optimizado.

Creación de un almacenamiento RDF optimizado con datos existentes

Puede crear un almacenamiento RDF utilizando datos RDF existentes.

Antes de empezar

Son necesarios los requisitos previos siguientes:

- Asegúrese de que la base de datos tenga un tamaño de página mínimo de 32 KB.
- Asegúrese de que el parámetro de configuración de base de datos **LOGFILSIZ** está establecido en un valor igual o superior a 20000.

  ```
  db2 UPDATE DATABASE CONFIGURATION FOR <NOMBRE_BD>
  USING LOGFILSIZ 20000
  ```

- Asegúrese de que el espacio de tablas SYSTOOLSPACE existe.

  ```
  CREATE TABLESPACE SYSTOOLSPACE IN IBMCATGROUP
  MANAGED BY AUTOMATIC STORAGE EXTENTSIZE 4
  ```

- Asegúrese de que el ID de autorización tiene los privilegios siguientes:
  - Autorización CREATETAB para el esquema de base de datos seleccionado.
  - Autorización CREATE EXTERNAL ROUTINES.
  - Privilegios de actualización para la tabla SYSTOOLS.ADMINTASKSTATUS.

Configure también la siguiente funcionalidad:

- Establezca el planificador de tareas administrativas en "YES".

  ```
  db2set DB2_ATS_ENABLE=YES
  ```

- Establezca el parámetro de configuración de base de datos **AUTORUNSTATS** en "ON".

  ```
  db2 UPDATE DB CONFIG USING AUTO_MAINT ON AUTO_TBL_MAINT ON AUTO_RUNSTATS ON
  ```

- Establezca la agrupación de almacenamientos intermedios en "AUTOMATIC" y asigne un tamaño inicial adecuado.

  ```
  db2 alter bufferpool IBMDEFAULTBP IMMEDIATE SIZE 15000 AUTOMATIC
  ```

En las plataformas Windows, el mandato **createrdfStoreAndLoader** necesita la aplicación CygWin. Para este mandato, se necesita la versión 4.0 o una versión posterior del programa de utilidad Gawk. Asimismo, para este mismo mandato se necesita la versión 8.14 o una versión posterior del programa de utilidad Core. Después de instalar CygWin añada *<directorio_instalación_CygWin>*/bin a la variable de entorno PATH. Si no tiene CygWin en la vía de acceso, verá el siguiente mensaje de error cuando ejecute el mandato:

```
'No se puede
ejecutar el programa "sh": CreateProcess error=2, El sistema no encuentra el
archivo especificado.'
```

En las plataformas Windows, el mandato **createrdfStoreAndLoader** puede invocarse desde un indicador de mandatos CygWin o un indicador de mandatos por omisión. Cuando se utiliza una línea de mandatos de CygWin, ninguna de las vías de acceso de archivo (-rdfdata, -storeloadfile, -storeschemafile, -objectnames) debe contener el prefijo 'cygdrive'. En su lugar, use una vía de acceso de Windows normal como 'C:\....'.

Si alguna de las vías de acceso especificadas contiene un espacio en el nombre de carpeta o de archivo, la serie entera tendrá que estar entre comillas dobles

Procedimiento

1. Exporte los datos existentes en un archivo n-Quad.
2. Controle los nombres de tablas de base de datos y espacios de tablas que conformarán el almacenamiento RDF. Cree un archivo de propiedades

objectNames.props que contenga una lista de las tablas de datos de almacenamiento RDF y los nombres y espacios de tablas correspondientes que desea asignar a las tablas.

El contenido de un archivo de ejemplo objectNames.props se muestra en el ejemplo siguiente:

```
direct_primary_hash=<nombre_tabla_usuario>,<espacio_tablas>
direct_secondary=<nombre_tabla_usuario>,<espacio_tablas>
reverse_primary_hash=<nombre_tabla_usuario>,<espacio_tablas>
reverse_secondary=<nombre_tabla_usuario>,<espacio_tablas>
long_strings=<nombre_tabla_usuario>,<espacio_tablas>
basic_stats=<nombre_tabla_usuario>,<espacio_tablas>
topk_stats=<nombre_tabla_usuario>,<espacio_tablas>
system_predicate=<nombre_tabla_usuario>,<espacio_tablas>
data_type=<nombre_tabla_usuario>,<espacio_tablas>
```

Nota: La utilización del archivo objectNames.props para establecer los nombres de tabla es opcional. Sin embargo, si decide no utilizar este archivo de propiedades, en su lugar se utilizarán los nombres generados por el sistema.

3. Emita el mandato **createrdfstoreandloader**.

   En Windows, este mandato requiere CygWin. Además, el programa de utilidad Gawk necesario para este mandato corresponde a la versión 4.0, mientras que el programa de utilidad base corresponde a la versión 8.14 o posterior.

   Especifique la instancia y esquema de base de datos donde desea crear el almacenamiento RDF. Elija también un nombre lógico para el almacenamiento RDF. Este nombre debe ser exclusivo en todos los almacenamientos RDF del esquema de base de datos.

   Especifique el parámetro **objectnames** en el mandato. Si no especifica el parámetro **objectNames**, en su lugar se utilizarán los nombres de objeto generados por el sistema para las tablas del almacenamiento RDF. Asegúrese de que el directorio de salida ya exista en el sistema de archivos.

   Este mandato crea un almacenamiento RDF optimizado utilizando los datos existentes y también genera archivos de carga de base de datos DB2, que deben utilizarse para cargar los datos en el almacenamiento RDF recientemente creado. Los archivos de carga se crean basándose en la salida del parámetro **storeloadfile**.

   Por ejemplo, emita el mandato siguiente para crear un almacenamiento denominado rdfStore2 en la base de datos DB1 y el esquema db2admin para el sistema principal localhost y el puerto 60000. Especifique myRdfData.nq para el archivo de datos RDF, y load.sql como nombre del archivo de carga del almacenamiento generado.

   ```
   createrdfstoreandloader rdfStore2 -host localhost -port 60000 -db DB1
   -user db2admin -password XXX -schema db2admin
   -rdfdatafile ./myRdfData.nq -storeloadfile ./rdfLoader/load.sql
   ```

4. Abra una ventana de indicador de mandatos de DB2 y conéctese a la instancia y esquema de base de datos donde se ha creado el almacenamiento RDF.
5. Ejecute el archivo ./rdfLoader/load.sql.

   Se cargarán los datos de este archivo generado al almacenamiento RDF.

   Nota: No utilice el argumento **-t** cuando ejecute el script SQL, dado que dicho script se genera con una nueva línea como separador de mandatos.

Creación de un almacén RDF mediante el control de acceso a nivel de gráfico

Puede crear un almacén RDF que utiliza el control de acceso a nivel de gráfico.

Antes de empezar

Determine los predicados de RDF cuyos tripletes se utilizan para controlar el acceso al gráfico RDF. Por ejemplo, puede utilizar los predicados ContextId (http://myapp.net/xmlns/CONTEXTID) y AppId (http://myapp.net/xmlns/APPID). Los filtros que utiliza para el acceso de control al gráfico también se conocen como *predicados de filtro*.

Determine los tipos de datos de DB2 para los valores de objeto RDF en estos predicados. Actualmente solo se da soporte al tipo de datos VARCHAR de DB2.

Procedimiento

Para crear un almacén RDF que utiliza el acceso de control a nivel de gráfico, grabe un programa que utilice el método createStore() de la clase StoreManager . Este método toma un argumento de propiedades. Especifique los predicados de filtro utilizando el siguiente formato de propiedades: `<RDF_PREDICATE> = <DB2_DATATYPE>`.

Ejemplo

El siguiente programa Java le muestra cómo utilizar la clase StoreManager para crear un almacén RDF con control de acceso a nivel de gráfico:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

import com.ibm.rdf.store.StoreManager;

public class CreateStoreGraphAccessControl {

 /* El acceso al gráfico se controla en función a
   * los siguientes dos predicados RDF.
  */
private final static String APPID =
"http://myapp.net/xmlns/APPID";
 private final static String CONTEXTID =
"http://myapp.net/xmlns/CONTEXTID";

 /**
  * El tipo de datos DB2 para estos predicados se asigna.
  */
 private final static String APPID_DATATYPE = "VARCHAR(60)";
 private final static String CONTEXTID_DATATYPE = "VARCHAR(60)";



/*
  * Crear un archivo java.util.properties que lista estas dos
   * propiedades y sus tipos de datos, en las que
   * propertyName es el predicado de RDF y
   * propertyValue es el tipo de datos para el predicado de RDF.
  */
 private static Properties filterPredicateProps = new
Properties();
```

```
 static {
  filterPredicateProps.setProperty(APPID, APPID_DATATYPE);
  filterPredicateProps.setProperty(CONTEXTID,
CONTEXTID_DATATYPE);
 }


 public static void main(String[] args) throws SQLException {

  Connection conn = null;

  // Obtener una conexión con la base de datos DB2
  try {
   Class.forName("com.ibm.db2.jcc.DB2Driver");
   conn = DriverManager.getConnection(
     "jdbc:db2://localhost:50000/dbrdf",
"db2admin", "db2admin");
  } catch (ClassNotFoundException e1) {
   e1.printStackTrace();
  }

  /*
   *  Crear el almacenamiento con los predicados de control de acceso.
  */
  StoreManager.createStore(conn, "db2admin",
"SampleAccessControl", null,
   filterPredicateProps);
 }
}
```

Capítulo 8. Modificación de datos en un almacenamiento RDF

Los datos de un almacenamiento RDF se pueden modificar mediante API de JENA u operaciones de actualización de SPARQL.

Modificación de datos en un almacenamiento RDF

Trabajar con datos en un almacenamiento RDF utilizando APIs de JENA.

Procedimiento

Para modificar datos en un almacenamiento RDF, puede utilizar el siguiente programa de muestra.

Modifique tripletes y gráficos en un almacenamiento RDF utilizando APIs de JENA como se muestra en el siguiente ejemplo.

Ejemplo

El programa siguiente muestra cómo modificar tripletes y gráficos en un almacenamiento RDF mediante API de JENA:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import com.hp.hpl.jena.graph.Graph;
import com.hp.hpl.jena.graph.Node;
import com.hp.hpl.jena.graph.Triple;
import com.hp.hpl.jena.query.Dataset;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.vocabulary.VCARD;

import com.ibm.rdf.store.Store;
import com.ibm.rdf.store.StoreManager;
import com.ibm.rdf.store.jena.RdfStoreFactory;

public class RDFStoreSampleInsert {

 public static void main(String[] args) throws SQLException {

  Connection conn = null;
  Store store = null;
  String storeName = "sample";
  String schema = "db2admin";

  try {
   Class.forName("com.ibm.db2.jcc.DB2Driver");
   conn = DriverManager.getConnection(
     "jdbc:db2://localhost:50000/dbrdf", "db2admin",
     "db2admin");
   conn.setAutoCommit(false);
  } catch (ClassNotFoundException e1) {
   e1.printStackTrace();
  }


  // Crear un almacenamiento o un conjunto de datos.
```

© Copyright IBM Corp. 2013

```
 store = StoreManager.createStore(conn, schema, storeName, null);

 // Si el almacenamiento ya existe, conectar con él.
 //store = StoreManager.connectStore(conn, schema, storeName);

  // Suprimir el almacenamiento si es necesario.
 //StoreManager.deleteStore(conn, schema, storeName);



 /*
  * Por lo general, retenga el objeto "store". De lo contrario, necesita
    * una consulta para saber con qué conjunto de tablas debe
    * trabajar. El objeto Store no mantiene una referencia con la conexión
    * que se pasa a los métodos StoreManager. Por lo tanto, en la API,
    * debe pasar una conexión de nuevo en los métodos de RDFStoreFactory. Puede
    * utilizar los demás objetos (como un conjunto de datos, un gráfico o un modelo)
    * como objetos ligeros. Es decir, crear un objeto para cada solicitud.
  */

 // Añadir un gráfico con nombre completo al almacenamiento.
 addNamedGraph(store,conn);


 // Eliminar un gráfico con nombre completo.
 removeNamedGraph(store, conn);

 // Añadir un triplete al gráfico predeterminado.
 addTripleToDefaultGraph(store, conn);

 // Añadir un triplete utilizando la interfaz de gráfico JENA.
 addTripleViaGraphInterface(store, conn);


 // Suprimir un almacenamiento.
 StoreManager.deleteStore(conn, schema, storeName);

 conn.commit();
}

public static void addNamedGraph(Store store, Connection conn) {

 // Conectar con un NamedModel en el almacenamiento.
 Model storeModel = RdfStoreFactory.connectNamedModel(store, conn,
   "http://graph1");

 // Crear un modelo en memoria con algunos datos.
 Model m = getMemModelWithSomeTriples();

 // Añadir un gráfico completo a rdfstore.
 storeModel.begin();
 storeModel.add(m);
 storeModel.commit();

 storeModel.close();

}

public static void removeNamedGraph(Store store, Connection conn) {

 Model storeModel = RdfStoreFactory.connectNamedModel(store, conn,
   "http://graph1");

 storeModel.begin();
 storeModel.removeAll();
 storeModel.commit();
```

```
}

public static void addTripleToDefaultGraph(Store store, Connection conn) {

 Dataset ds = RdfStoreFactory.connectDataset(store, conn);
 Model m = ds.getDefaultModel();

 // Añadir información utilizando el objeto de modelo thye.
 m.begin();

 String personURI = "http://somewhere/JohnSmith";
 String fullName = "John Smith";
 Resource johnSmith = m.createResource(personURI);
 johnSmith.addProperty(VCARD.FN, fullName);

 m.commit();
 m.close();
}

public static void addTripleViaGraphInterface(Store store, Connection conn) {

 Graph g = RdfStoreFactory.connectNamedGraph(store, conn,
   "http://graph2");

 Node s = Node.createURI("http://sub1");
 Node p = Node.createURI("http://pred1");
 Node v = Node.createLiteral("val1");

 g.add(new Triple(s, p, v));
 g.close();
}

private static Model getMemModelWithSomeTriples() {

 Model m = ModelFactory.createDefaultModel();

 Node s = Node.createURI("somesubject");
 Node p = Node.createURI("somepredicate");
 Node v = Node.createURI("AnObject");
 Triple t = Triple.create(s, p, v);
 m.add(m.asStatement(t));

 s = Node.createURI("someothersubject");
 p = Node.createURI("someotherpredicate");
 v = Node.createURI("AnotherObject");
 t = Triple.create(s, p, v);
 m.add(m.asStatement(t));

 return m;
}
}
```

Soporte para SPARQL UPDATE

A partir del fixpack 2 y los fixpacks posteriores de DB2 versión 10.1, se da soporte a SPARQL UPDATE versión 1.1. SPARQL UPDATE versión 1.1 da soporte a dos categorías de operaciones de actualización en un almacén de gráficos.

Actualización de gráficos de SPARQL

En el fixpack 2 y los fixpacks posteriores de DB2 versión 10.1, se da soporte a los mandatos de actualización de gráficos de SPARQL. Estos mandatos ayudan a añadir o eliminar tripletes de gráficos de un almacén de gráficos.

Los siguientes mandatos de actualización de gráficos tienen soporte:

INSERT DATA
Añade en el gráfico de destino, tripletes especificados en la consulta. Crea un gráfico de destino, si este no existe.

INSERT WHERE
Añade tripletes correlacionando el patrón de la condición `WHERE` de la consulta con el gráfico de destino. Crea un gráfico de destino, si este no existe.

DELETE DATA
Elimina los tripletes que se especifican en la consulta. Suprimir tripletes que no están presentes en un almacenamiento RDF o un gráfico no tiene ningún efecto y produce un resultado correcto.

DELETE WHERE
Elimine los tripletes correlacionando el patrón que se especifica en la cláusula `WHERE` de la consulta. Suprimir tripletes que no están presentes en un almacenamiento RDF o un gráfico no tiene ningún efecto y produce un resultado correcto.

LOAD
Lee un documento RDF a partir de un identificador de recursos internacionalizados (IRI) e inserta sus tripletes en un gráfico especificado. Crea un gráfico de destino, si este no existe.

CLEAR
Elimina todos los tripletes de un gráfico especificado.

Gestión de gráficos SPARQL

En el fixpack 2 y en los fixpacks posteriores de DB2 versión 10.1, se da soporte a los mandatos de gestión de gráficos SPARQL que crean y suprimen gráficos de un almacén de gráficos. Los mandatos también proporcionan atajos para llevar a cabo operaciones de actualización de gráficos que se ejecutan a menudo durante la gestión de gráficos para añadir, mover o copiar gráficos.

Se da soporte a los siguientes mandatos de actualización:

CREATE
Crea un gráfico en el almacén de gráficos. El gráfico no es permanente ya que RDF de DB2 no guarda los gráficos vacíos.

DROP Elimina los gráficos especificados del almacén de gráficos.

COPY Inserta todos los datos de un gráfico de entrada en un gráfico de destino. Los datos del gráfico de entrada no se ven afectados. Los datos del gráfico de destino, si hay, se eliminan antes de efectuar la inserción.

MOVE
Mueve todos los datos de un gráfico de entrada a un gráfico de destino. El gráfico de entrada se elimina después de la inserción. Los datos del gráfico de destino, si hay, se eliminan antes de efectuar la inserción.

ADD Inserta todos los datos de un gráfico de entrada en un gráfico de destino. Los datos del gráfico de entrada no se ven afectados. Los datos iniciales del gráfico de destino, si hay, se mantienen intactos.

Modificación de un almacenamiento RDF mediante API de SPARQL UPDATE

En el fixpack 2 y en los fixpacks posteriores de DB2 versión 10.1, puede actualizar los datos de un almacenamiento de datos RDF mediante API UPDATE con soporte de SPARQL versión 1.1.

El programa siguiente muestra cómo modificar un almacenamiento RDF ejecutando API UPDATE de SPARQL.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import com.hp.hpl.jena.graph.Node;
import com.hp.hpl.jena.query.Dataset;
import com.hp.hpl.jena.sparql.core.Quad;
import com.hp.hpl.jena.sparql.modify.request.QuadDataAcc;
import com.hp.hpl.jena.sparql.modify.request.UpdateDataInsert;
import com.hp.hpl.jena.sparql.util.NodeFactory;
import com.hp.hpl.jena.update.UpdateAction;
import com.ibm.rdf.store.Store;
import com.ibm.rdf.store.StoreManager;
import com.ibm.rdf.store.jena.RdfStoreFactory;
/**
* Programa de ejemplo para usar actualizaciones de SPARQL
*/
public class RDFStoreUpdateSample {
 public static void main(String[] args) throws SQLException
  // Crear la conexión
  Connection conn = null;
  Store store = null;
  String storeName = "staffing";
  String schema = "db2admin";
  try {
   Class.forName("com.ibm.db2.jcc.DB2Driver");
   conn = DriverManager.getConnection( "jdbc:db2://localhost:50000/RDFDB",
     "db2admin", "db2admin");
   conn.setAutoCommit(false);  }
  catch (ClassNotFoundException e1) {
   e1.printStackTrace();
  }
  // Conectar con el almacenamiento
  store = StoreManager.connectStore(conn, schema, storeName);


  // Crear el conjunto de datos
  Dataset ds = RdfStoreFactory.connectDataset(store, conn);
  // Actualizar conjunto de datos analizando la sentencia SPARQL UPDATE
  // updateByParsingSPARQLUpdates(ds);
  // Actualizar conjunto de datos compilando objetos de actualización
  // updateByBuildingUpdateObjects(ds);
  ds.close();
  conn.commit();
}

/**
* Actualizar analizando actualización SPARQL
*
* @param ds
* @param graphNode
*/
private static void updateByParsingSPARQLUpdates(Dataset ds) {
 String update = "INSERT DATA
 { GRAPH <http://example/bookStore>
  { <http://example/book1> <http://example.org/ns#price>  100 } }";
 //Execute update via UpdateAction
```

```
 UpdateAction.parseExecute(update, ds);
}
/**
* Actualizar creando objetos de actualización
*
* @param ds
* @param graphNode
*/
 private static void updateByBuildingUpdateObjects(Dataset ds) {
  // Nodo de gráfico
  Node graphNode = NodeFactory.parseNode("http://example/book2>");
  Node p = NodeFactory.parseNode("<http://example.org/ns#price>");
  Node o = NodeFactory.parseNode("1000");
  Quad quad = new Quad(graphNode, s, p, o);
  Node s2 = NodeFactory.parseNode("<http://example/book3>");
  Node o2 = NodeFactory.parseNode("2000");
  Quad quad2 = new Quad(graphNode, s2, p, o2);
  //Crear datos de cuarteto para añadir al almacenamiento
  QuadDataAcc acc = new QuadDataAcc();
  acc.addQuad(quad);
  acc.addQuad(quad2);
  //Crear el objeto de actualización
  UpdateDataInsert insert = new UpdateDataInsert(acc);
  //Ejecutar la actualización mediante UpdateAction
  UpdateAction.execute(insert, ds);
 }
}
```

Capítulo 9. Consulta de un almacenamiento RDF

Utilice SPARQL para consultar los datos de los almacenamientos Resource Description Framework (RDF) de DB2.

SPARQL para RDF versión 1.0 tiene soporte. También tiene soporte el subconjunto de funciones siguientes de SPARQL versión 1.1:

- `AVG`
- `COALESCE`
- `COUNT`
- `GROUP BY`
- `HAVING`
- `MAX`
- `MIN`
- Expresiones `SELECT`
- `STRSTARTS`
- `STRENDS`
- `Subconsultas`
- `SUM`

A partir del fixpack 2 de DB2 versión 10.1, las funciones de SPARQL versión 1.1 también tienen soporte:

- Soporte de UPDATE para el lenguaje de consulta de SPARQL.
- Soporte del protocolo de HTTP de almacén de gráficos para el lenguaje de consulta de SPARQL.

Consultas de RDF y API

El lenguaje de consultas SPARQL se utiliza para modificar los datos de las bases de datos DB2, mientras que las API de estructura JENA proporcionan la interfaz de programación. Existen algunas limitaciones para los almacenamientos RDF de DB2.

Soporte para las consultas SPARQL

Cuando se trabaja con datos RDF, deben tenerse en cuenta las diversas restricciones y limitaciones sintácticas y semánticas que se aplican a SPARQL.

Límites sobre las longitudes de las URI
Una implementación RDF de base de datos de DB2 puede almacenar URI de cualquier longitud. Sin embargos, solamente se utilizan los 2000 primeros caracteres para las operaciones de comparación.

Límites sobre las longitudes de los literales
Una implementación RDF de base de datos de DB2 almacena literales de cualquier longitud. Sin embargo, solamente se utilizan los 2000 primeros caracteres para las operaciones de comparación y otras operaciones, como STRSTARTS y STRENDS.

Operador DATATYPE en una expresión FILTER
El soporte se amplía para el operador SPARQL `DATATYPE` en una expresión FILTER.

© Copyright IBM Corp. 2013

Constantes en una expresión FILTER
El soporte se amplía para constantes en una expresión FILTER.

Menos unario en una expresión FILTER
No se da soporte a las expresiones de filtro con un menos unario en las variables.

```
FILTER ( -?v = -10 )
```

La expresión devuelve una RdfStoreException, con el identificador de error DB255001E y el código de error de SQL -104.

Operadores DISTINCT * o REDUCED * de una expresión SELECT
El soporte se amplía para operadores `DISTINCT *` o `REDUCED *` en una expresión SELECT.

Operador de tipo de datos de una expresión SELECT
El soporte se amplía para el operador de tipo de datos SPARQL en una expresión SELECT.

Secuencia de escape de punto en expresiones regulares
La secuencia de escape de punto en las coincidencias de patrón de expresiones regulares tiene limitaciones, donde la expresión no coincide de forma adecuada con el carácter de punto.

```
FILTER regex(?val, "example\\.com")
```

El ejemplo de código anterior no coincide con la serie "example.com" como se esperaba.

Limitación de secuencia de escape de barra inclinada invertida doble
Las secuencias de escape con una barra inclinada invertida doble en las series no se interpretan correctamente. No existe ninguna solución alternativa.

Cygwin y el mandato `createrdfstoreandloader` (Windows)
Cuando se ejecuta el mandato **`createrdfstoreandloader`** mediante Cygwin en plataformas Windows, Cygwin se bloquea en vez de visualizar mensajes de error o de aviso. Por ello, ejecute el mandato **`createrdfstoreandloader`** únicamente en plataformas Linux o UNIX. A continuación, utilice los archivos de SQL y de carga de DB2 generados para la carga a un servidor DB2 en la plataforma Windows.

Soporte de la API del modelo JENA

La implementación de DB2 de la API del modelo JENA está limitada en cuanto a cómo se pueden utilizar las API `Model.read()` y `Model.add(Model)`.

Tripletes duplicados cuando se utiliza la API `Model.read()`
Si la fuente de entrada contiene tripletes duplicados, puede que los duplicados no se eliminen, porque la implementación de la biblioteca JENA de la API `Model.read()` utiliza la carga masiva en lotes de 1000 tripletes. El almacenamiento RDF de DB2 no filtra los tripletes duplicados en estos lotes.

Como método alternativo, lea siempre la fuente de entrada en un modelo JENA en memoria y, a continuación, añada el modelo en memoria al almacenamiento de DB2 mediante la API `Model.add(model)`.

Tripletes duplicados cuando se utiliza la API `Model.add(Model)`
La API `Model.add(Model)` presupone que el gráfico que se añade no existe en el conjunto de datos. Si el gráfico existe y está añadiendo tripletes duplicados, el triplete duplicado no se elimina.

El enfoque sugerido para un almacenamiento RDF de DB2 se indica a continuación:

1. La primera vez que añada un gráfico, utilice el método `Model.add(Model)`.
2. Si desea añadir o actualizar tripletes para ese gráfico existente, utilice las funciones siguientes:
   - `model.add(s,p,v)`
   - `model.add(statement)`
   - `graph.add(Triple)`
   - `Resource.addXX()`

   Recuerde: Si está añadiendo un triplete que ya existe, se devuelve un mensaje de error.

Ahora, en el fixpack 2 o los fixpacks posteriores de DB2, versión 10.1, se han eliminado las dos limitaciones anteriores en la implementación de producto DB2 de la API de JENA.

Ejecución de consultas SPARQL

Puede consultar los datos almacenados en un almacenamiento RDF.

Ejemplo

El programa siguiente muestra cómo consultar datos en un almacenamiento RDF mediante el lenguaje de consultas SPARQL.

```
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import com.hp.hpl.jena.query.Dataset;
import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecution;
import com.hp.hpl.jena.query.QuerySolution;
import com.hp.hpl.jena.query.ResultSet;
import com.hp.hpl.jena.rdf.model.Model;
import com.ibm.rdf.store.Store;
import com.ibm.rdf.store.StoreManager;
import com.ibm.rdf.store.exception.RdfStoreException;
import com.ibm.rdf.store.jena.RdfStoreFactory;
import com.ibm.rdf.store.jena.RdfStoreQueryExecutionFactory;
import com.ibm.rdf.store.jena.RdfStoreQueryFactory;

public class RDFStoreSampleQuery   {


 public static void main(String[] args) throws SQLException, IOException {

  Connection conn = null;
  Store store = null;
  String storeName = "sample";
  String schema = "db2admin";

  // Obtener una conexión con la base de datos DB2
  try {
   Class.forName("com.ibm.db2.jcc.DB2Driver");
   conn = DriverManager.getConnection(
     "jdbc:db2://localhost:50000/dbrdf", "db2admin",
     "db2admin");
```

```
} catch (ClassNotFoundException e1) {
 e1.printStackTrace();
}


try {

 /* Conectar con el almacenamiento RDF necesario en el esquema especificado. */
 store = StoreManager.connectStore(conn, schema, storeName);

 /* Esto va a ser nuestra consulta SPARQL, por ejemplo, seleccionar tripletes
  en el gráfico por omisión donde el objeto es <ibm.com>
  */
     String query = "SELECT * WHERE { ?s ?p
    <https://www.ibm.com> }";

 /* Crear el objeto Query para la serie SPARQL. */
 Query q = RdfStoreQueryFactory.create(query);

 /* Obtener la interfaz de conjunto de datos del almacenamiento RDF. */
 Dataset ds = RdfStoreFactory.connectDataset(store, conn);

 /* Crear un objeto QueryExecution proporcionando la consulta para ejecutar
  y el conjunto de datos en el que se ejecutará. */
 QueryExecution qe = RdfStoreQueryExecutionFactory.create(q, ds);

 long rows = 0;
 Model m = null;

 /* En función del tipo de consulta SPARQL, llamar al método de ejecución
   adecuado. */
 if (q.isSelectType()) {
  ResultSet rs = qe.execSelect();
  while (rs.hasNext()) {
   QuerySolution qs = rs.next();
   System.out.println(qs);
   System.out.println();
   rows++;
  }
 }
 else if ( q.isDescribeType() ) {
   m = qe.execDescribe();
   m.write(System.out, "N-TRIPLE");
 }
 else if ( q.isAskType() ) {
  System.out.println(qe.execAsk());

 }
 else if (q.isConstructType()) {
  m = qe.execConstruct();
  m.write(System.out, "N-TRIPLE");
 }


 /* Cerrar el objeto QueryExecution. Esto es necesario para garantizar que
  ninguna sentencia JDBC tiene fugas. */
 qe.close();

 // Mostrar el número de filas devueltas
 if ( m != null ) {
  System.out.println("Número de filas  : " + m.size());
  m.close();
 }
 else {
  System.out.println("Número de filas  : " + rows);
 }
```

```
  }
  catch(RdfStoreException e) {

   e.printStackTrace();
  }
  catch (Exception e) {
   e.printStackTrace();
  }

 }

}
```

Creación de una unión de todos los gráficos con nombres

Puede establecer el gráfico por omisión de modo que sea la unión de todos los gráficos con nombre del conjunto de datos para una consulta SPARQL. Esta característica se aplica a las consultas solamente. No afecta al almacenamiento ni cambia la carga.

Los dos programas siguientes muestran cómo establecer el gráfico por omisión como la unión de todos los gráficos con nombre en el conjunto de datos para una consulta SPARQL.

Ejemplo

1. En el programa Java de ejemplo siguiente, se establece el gráfico por omisión para todas las consultas de un objeto de almacenamiento:

```
import java.sql.Connection;

import com.hp.hpl.jena.query.Dataset;
import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecution;
import com.ibm.rdf.store.Store;
import com.ibm.rdf.store.StoreManager;
import com.ibm.rdf.store.Symbols;
import com.ibm.rdf.store.jena.RdfStoreFactory;
import com.ibm.rdf.store.jena.RdfStoreQueryExecutionFactory;
import com.ibm.rdf.store.jena.RdfStoreQueryFactory;


public class UnionDefaultGraph {

 public static void setPerQuery() {

  Connection conn = null;
  Store store = null;

  // obtener una conexión con la base de datos DB2
  //conn = DriverManager.getConnection(...);

  store = StoreManager.connectStore(conn, "db2admin", "Sample");

  //Establecer el gráfico predeterminado como unión de todos los gráficos con
  //nombre del conjunto de datos   para todas las consultas en el objeto de almacén
  store.getContext().set(Symbols.unionDefaultGraph, true);

  // crear la consulta
  String query = "SELECT * WHERE { ?s ?p <https://www.ibm.com> }";
  Query q = RdfStoreQueryFactory.create(query);
  Dataset ds = RdfStoreFactory.connectDataset(store, conn);
   QueryExecution qe = RdfStoreQueryExecutionFactory.create(q, ds);

  // continuar para ejecutar la consulta
```

```
 }

}
```

2. En el ejemplo de programa Java siguiente, se establece el gráfico por omisión consulta a consulta:

```
import java.sql.Connection;

import com.hp.hpl.jena.query.Dataset;
import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecution;
import com.ibm.rdf.store.Store;
import com.ibm.rdf.store.StoreManager;
import com.ibm.rdf.store.Symbols;
import com.ibm.rdf.store.jena.RdfStoreFactory;
import com.ibm.rdf.store.jena.RdfStoreQueryExecutionFactory;
import com.ibm.rdf.store.jena.RdfStoreQueryFactory;


public class UnionDefaultGraph {

 public static void setPerQuery() {

  Connection conn = null;
  Store store = null;

  // obtener una conexión con la base de datos DB2
  //conn = DriverManager.getConnection(...);


store = StoreManager.connectStore(conn, "db2admin",
"Sample");
  String query = "SELECT * WHERE { ?s ?p <https://www.ibm.com> }";
  Query q = RdfStoreQueryFactory.create(query);
  Dataset ds = RdfStoreFactory.connectDataset(store, conn);
  QueryExecution qe = RdfStoreQueryExecutionFactory.create(q,
ds);

  /* Establecer el gráfico por omisión como la unión de todos los gráficos con nombre
    * del conjunto de datos solamente para esta consulta.
   */
  qe.getContext().set(Symbols.unionDefaultGraph, true);

  // Continuar para ejecutar la consulta.

 }

}
```

Registro de manejadores DESCRIBE personalizados

Use el mecanismo definido por ARQ para personalizar cómo se manejarán las consultas DESCRIBE.

Con el almacenamiento RDF de DB2 ya viene un manejador DESCRIBE por omisión. Proporciona una descripción de un nivel de profundidad de los recursos seleccionados.

Cuando implemente sus propios manejadores DESCRIBE, asegúrese de que minimiza el número de llamadas realizadas al servidor de bases de datos DB2.

Ejemplo

El siguiente programa muestra cómo registrar e implementar su propio manejador DESCRIBE para el almacenamiento RDF de DB2.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;

import com.hp.hpl.jena.graph.Node;
import com.hp.hpl.jena.graph.Triple;
import com.hp.hpl.jena.query.Dataset;
import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecution;
import com.hp.hpl.jena.rdf.model.AnonId;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.Property;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.rdf.model.ResourceFactory;
import com.hp.hpl.jena.sparql.core.describe.DescribeHandler;
import com.hp.hpl.jena.sparql.core.describe.DescribeHandlerFactory;
import com.hp.hpl.jena.sparql.core.describe.DescribeHandlerRegistry;
import com.hp.hpl.jena.sparql.util.Context;
import com.ibm.rdf.store.Store;
import com.ibm.rdf.store.StoreManager;
import com.ibm.rdf.store.jena.RdfStoreFactory;
import com.ibm.rdf.store.jena.RdfStoreQueryExecutionFactory;
import com.ibm.rdf.store.jena.RdfStoreQueryFactory;

public class DescribeTest {

 /**
  * @param args
  * @throws ClassNotFoundException
  * @throws SQLException
  */

 public static void main(String[] args) throws ClassNotFoundException,SQLException {
  if (args.length != 5) {
   System.err.print("Invalid arguments.\n");
   printUsage();
   System.exit(0);
  }

    /* Nota: Asegúrese de que el manejador de descripciones de DB2 por omisión
también se elimina.
     * Utilice las API de ARQ para eliminar los manejadores de descripciones
registrados por omisión.
     * Si no lo hace, cada recurso se ejecuta a través de varios manejadores
     * de descripciones, lo que provocaría mucha actividad general innecesaria.
     */

  /*
   * Ahora se registra un nuevo DescribeHandler (MyDescribeHandler)
   */
  DescribeHandlerRegistry.get().add(new DescribeHandlerFactory() {
   public DescribeHandler create() {
    return new MyDescribeHandler();
   }
  });

  /*
   * Creación de conexión de base de datos y objeto de almacenamiento.
```

```
 */
 Store store = null;
 Connection conn = null;

 Class.forName("com.ibm.db2.jcc.DB2Driver");

 String datasetName = args[0];
 String url = args[1];
 String schema = args[2];
 String username = args[3];
 String passwd = args[4];

 conn = DriverManager.getConnection(url, username, passwd);

 if (StoreManager.checkStoreExists(conn, schema, datasetName)) {
  store = StoreManager.connectStore(conn, schema, datasetName);
 } else {
  store = StoreManager.createStore(conn, schema, datasetName, null);
 }

 /*
  * Creación de conjunto de datos con datos de prueba.
  */
 Dataset ds = RdfStoreFactory.connectDataset(store, conn);

 ds.getDefaultModel().removeAll();
 ds.getDefaultModel().add(getInputData());

 /*
  * Ejecutando una consulta DESCRIBE SPARQL.
  */
 String sparql = "DESCRIBE <http://example.com/x>";

 Query query = RdfStoreQueryFactory.create(sparql);

 QueryExecution qe = RdfStoreQueryExecutionFactory.create(query, ds);

 Model m = qe.execDescribe();

 m.write(System.out, "N-TRIPLES");

 qe.close();

 conn.close();
}

private static void printUsage() {
 System.out.println("Uso correcto: ");
 System.out.println("java DescribeTest <NOMBRE_CONJUNTO_DATOS>");
 System.out.println(" <URL> <ESQUEMA> <NOMBRE_USUARIO> <CONTRASEÑA>");
}

// Creando datos de entrada.
private static Model getInputData() {
 Model input = ModelFactory.createDefaultModel();

 Resource iris[] = {
   ResourceFactory.createResource("http://example.com/w"),
   ResourceFactory.createResource("http://example.com/x"),
   ResourceFactory.createResource("http://example.com/y"),
   ResourceFactory.createResource("http://example.com/z") };

 Property p = ResourceFactory.createProperty("http://example.com/p");
 Node o = Node.createAnon(new AnonId("AnonID"));

 for (int i = 0; i < iris.length - 1; i++) {
  input.add(input.asStatement(Triple.create(iris[i].asNode(), p
```

```
      .asNode(), iris[i + 1].asNode())));
   }

   input.add(input.asStatement(Triple.create(iris[iris.length - 1]
     .asNode(), p.asNode(), o)));

   return input;
  }
}

/*
 * Implementación de ejemplo de DescribeHandler.
  */
class MyDescribeHandler implements DescribeHandler {

 /*
  * Establecer para mantener un seguimiento de todos los recursos únicos
  * necesarios para DESCRIBE.
  */
 private Set <Resource> resources;
      private Model accumulator;

 // Variables de campo restantes

 public void start(Model accumulator, Context ctx) {
  resources = new HashSet <Resource>();
  this.accumulator = accumulator;
  // Otra declaración de objeto según sea necesario.
 }

 public void describe(Resource resource) {
  resources.add(resource);
 }

 public void finish() {
  /*
   * Implementar su propia lógica de descripciones.
     * Añadir los nuevos tripletes al objeto de modelo 'accumulator'.
     * Es mejor evitar varias llamadas a la base de datos, por lo tanto
   * estructure la lógica de acuerdo con esto.
     * Si necesita FullClosure, utilice las API
   * com.ibm.rdf.store.internal.jena.impl.DB2Closure.closure(),
   * en lugar de com.hp.hpl.jena.sparql.util.Closure.closure().
   */


 }
}
```

Aplicación del control de acceso de nivel de gráfico mediante el servidor de bases de datos DB2

Puede asegurarse de que una consulta SPARQL puede acceder solamente a gráficos RDF específicos haciendo que sea el motor de DB2 el que aplique el control de acceso.

Procedimiento

La tabla de metadatos de predicados del sistema contiene los predicados RDF. Estos metadatos se especifican para forzar la aplicación del control de acceso de nivel de gráfico durante la creación del almacenamiento. También almacena los nombres de las columnas en las tablas Direct Primary y Reverse Primary que contienen los valores para estos predicados.

Emita la siguiente consulta:

```
"select * from System_predicates_table>"
```

Salida de ejemplo:

```
ENTRY_ID   COLNAME        MAPNAME
=====================================================
1          SYSPRED_0      http://myapp.net/xmlns/APPID
1          SYSPRED_1      http://myapp.net/xmlns/CONTEXTID
```

Aquí, `SYSPRED_0` es la columna correspondiente al predicado http://myapp.net/xmlns/APPID y `SYSPRED_1` es la columna correspondiente al predicado http://myapp.net/xmlns/CONTEXTID en las tablas Direct Primary y Reverse Primary.
Puede utilizar las funciones del control de acceso preciso del software DB2 para definir las restricciones de `ROW LEVEL` según sus requisitos en las tablas Direct Primary y Reverse Primary utilizando estas columnas.

Aplicación del control de acceso de nivel de gráfico mediante el generador de SQL de almacenamiento RDF

Puede controlar el acceso a determinados gráficos RDF. El acceso se puede controlar haciendo que el generador de SQL de almacenamiento RDF aplique los filtros adecuados en el SQL que genera.

Determine los valores de los predicados RDF basándose en los gráficos RDF que contengan tripletes. Los valores determinan a qué gráficos se accede.

Decida si la comprobación de control de acceso debe realizarse sobre un solo valor o sobre un valor cualquiera de un conjunto de valores. Si la comprobación de control de acceso se realiza sobre un único valor, cree un objeto `QueryFilterPredicateEquals` que devuelva dicho valor. Por el contrario, si la comprobación de control de acceso se realiza sobre un valor cualquiera de un conjunto de valores, cree un objeto `QueryFilterPredicateMember` que devuelva el conjunto de valores. Repita este proceso para cada predicado de filtro de control de acceso.

Establezca los objetos en el contexto `QueryExecution` de la consulta SPARQL que se ejecuta.

Ejemplo

El programa Java siguiente muestra cómo se controla el acceso a gráficos mediante el generador de SQL del almacenamiento RDF.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import com.hp.hpl.jena.query.Dataset;
import com.hp.hpl.jena.query.QueryExecution;
import com.hp.hpl.jena.query.QuerySolution;
import com.hp.hpl.jena.query.ResultSet;
import com.hp.hpl.jena.rdf.model.Literal;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.Property;
```

```
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.rdf.model.ResourceFactory;
import com.ibm.rdf.store.Store;
import com.ibm.rdf.store.StoreManager;
import com.ibm.rdf.store.Symbols;
import com.ibm.rdf.store.jena.RdfStoreFactory;
import com.ibm.rdf.store.jena.RdfStoreQueryExecutionFactory;
import com.ibm.rdf.store.query.filter.QueryFilterPredicate;
import com.ibm.rdf.store.query.filter.QueryFilterPredicateEquals;
import com.ibm.rdf.store.query.filter.QueryFilterPredicateMember;
import com.ibm.rdf.store.query.filter.QueryFilterPredicateProvider;

public class QueryStoreGraphAccessControl {

 /* Objetos de propiedad para los dos predicados RDF basados en los
  * tripletes con los que se controla el acceso al gráfico.
  */
 private final static Property APPID =
ModelFactory.createDefaultModel()
.createProperty("http://myapp.net/xmlns/APPID");

 private final static Property CONTEXTID =
ModelFactory.createDefaultModel()
   .createProperty("http://myapp.net/xmlns/CONTEXTID");


 public static void main(String[] args) throws SQLException {

  Connection conn = null;
  Store store = null;

  // obtener una conexión con la base de datos DB2
  try {
   Class.forName("com.ibm.db2.jcc.DB2Driver");
   conn = DriverManager.getConnection(
     "jdbc:db2://localhost:50000/dbrdf",
"db2admin","db2admin");
  } catch (ClassNotFoundException e1) {
   e1.printStackTrace();
  }

  /* Conectar con el almacenamiento de acceso controlado. */
  store = StoreManager.connectStore(conn, "db2admin",
"SampleAccessControl");
  Dataset ds = RdfStoreFactory.connectDataset(store, conn);

  // Insertar datos para consultar
  insertDataForQuerying(ds);

  // Consultar y asegurarse de que el control de acceso se ha aplicado
  QueryWithGraphAccessControl(ds);

 }


 private static void QueryWithGraphAccessControl(Dataset ds) {

  //Crear el valor de filtro para APPID.
  final QueryFilterPredicateEquals appIdFilter =
   new QueryFilterPredicateEquals() {
    public Literal getValue() {
     return ModelFactory.createDefaultModel()
       .createLiteral("App1");
    }
   };

  //Crear el valor de filtro para contextID.
```

```
  final QueryFilterPredicateMember ctxIdFilter = new
QueryFilterPredicateMember() {
    public List <> getValues() {
      List<Literal> a = new ArrayList<Literal>();
      a.add(ModelFactory.createDefaultModel()
       .createLiteral("Context1"));
      a.add(ModelFactory.createDefaultModel()
       .createLiteral("Context2"));
      return a;
    }
 };

  // Crear el objeto QueryExecution.
  QueryExecution qe = RdfStoreQueryExecutionFactory.create(
   "select ?who where { ?who <http://pre/test.3> ?x }",
ds);

  // Establecer los valores de filtro de control de acceso para esta
consulta.
  qe.getContext().set(Symbols. queryFilterPredicates,
new QueryFilterPredicateProvider() {
    public QueryFilterPredicate getQueryFilterPredicate(
     Property filterProperty) {
    if (filterProperty.equals(APPID) ) {
     return appIdFilter;
    }
    else if ( filterProperty.equals(CONTEXTID)) {
     return ctxIdFilter;
    }
    else
     return null;

   }

   });

  // Establecer el gráfico por omisión como unión de todos los
gráficos con nombre.
  qe.getContext().set(Symbols.unionDefaultGraph, true);

  /* Ejecutar SPARQL. Debe tenerse en cuenta que solamente coincide Model1
     * y los tripletes de Model2 no se devuelven */
  ResultSet rs = qe.execSelect();
  while (rs.hasNext()) {
   QuerySolution qs = rs.next();
   System.out.println(qs.toString());
  }
  qe.close();


 }

 private static void insertDataForQuerying(Dataset ds) {

  // Añadir tripletes a graph1.
  ds.getNamedModel("Model1").add(getMemModel());

  // Añadir tripletes de predicado de filtro en el gráfico existente.
  ds.getNamedModel("Model1").add(
ResourceFactory.createResource(
      "http://res1"), APPID, "App1");

ds.getNamedModel("Model1").add(
ResourceFactory.createResource(
     "http://res1"), CONTEXTID, "Context1");

  // Añadir tripletes a graph2.
```

```
  ds.getNamedModel("Model2").add(getMemModel());

  // Añadir tripletes de predicado de filtro en el gráfico existente.
  ds.getNamedModel("Model2").add(
ResourceFactory.createResource(
      "http://res2"), APPID, "App2");
ds.getNamedModel("Model2").add(
ResourceFactory.createResource(
     "http://res2"), CONTEXTID, "Context2");

 }

 private static Model getMemModel() {private static Model getMemModel() {
  String sub = "http://sub/";
  String pre = "http://pre/test.";
  String obj = "http://obj/";

  Model m = ModelFactory.createDefaultModel();

  long TRIPLE_COUNT = 12;
  for (int i = 0; m.size() < TRIPLE_COUNT; i++) {
   Resource s = ResourceFactory.createResource(sub + (i % 3));
   Property p = ResourceFactory.createProperty(pre + (i % 9));
   Literal o = ResourceFactory.createPlainLiteral(obj + (i % 4));
   m.add(ResourceFactory.createStatement(s, p, o));
  }

  return m;
 }


}
```

Capítulo 10. Configuración del protocolo de almacén de gráficos de SPARQL versión 1.1 y de SPARQL en HTTP

En el fixpack 2 y en los fixpacks posteriores de DB2 versión 10.1, RDF de DB2 admite el protocolo de HTTP de almacén de gráficos de SPARQL versión 1.1. Este protocolo requiere Apache JENA Fuseki versión 0.2.4. Para usar la API REST de SPARQL necesita configurar el entorno Fuseki.

Antes de empezar

Para configurar el entorno Fuseki:

1. Descargue el archivo `jena-fuseki-0.2.4-distribution.zip` desde http://archive.apache.org/dist/jena/binaries/“http://archive.apache.org/dist/jena/binaries/”.
2. Extraiga el archivo en el sistema local.

Procedimiento

1. Abra una ventana del indicador de mandatos y vaya al directorio *<dir instalación Fuseki>*`/jena-fuseki-0.2.4`.
2. Abra el archivo `config.ttl` y añada `db2rdf` como prefijo

```
@prefix :        <#> .
@prefix fuseki:  <http://jena.apache.org/fuseki#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix tdb:     <http://jena.hpl.hp.com/2008/tdb#> .
@prefix ja:      <http://jena.hpl.hp.com/2005/11/Assembler#> .

@prefix db2rdf:  <http://rdfstore.ibm.com/IM/fuseki/configuration#>
```

3. Añada el servicio DB2 RDF al archivo `config.ttl`. Añada este servicio a la sección del archivo donde están registrados los servicios restantes.

```
fuseki:services (
     <#service1>
     <#service2>
      <#serviceDB2RDF_staffing>
   ) .
```

   Puede registrar varios servicios. Cada uno de ellos consulta diferentes conjuntos de datos RDF de DB2.

4. Añada la configuración siguiente al archivo `config.ttl` para inicializar el espacio de nombres RDF. Esta configuración registra el ensamblador que crea el `DB2Dataset`. La configuración también registra los motores `DB2QueryEngine` y `DB2UpdateEngine`.

```
# DB2
[] ja:loadClass "com.ibm.rdf.store.jena.DB2" .
db2rdf:DB2Dataset  rdfs:subClassOf  ja:RDFDataset .
```

5. Añada todos estos detalles sobre el servicio RDF de DB2 al final del archivo `config.ttl`.

```
# Service DB2 Staffing store
<#serviceDB2RDF_staffing>
rdf:type fuseki:Service ;
rdfs:label "SPARQL against DB2 RDF store" ;
fuseki:name "staffing" ;
fuseki:serviceQuery "sparql" ;
fuseki:serviceQuery "query" ;
```

© Copyright IBM Corp. 2013

```
fuseki:serviceUpdate "update" ;
fuseki:serviceUpload "upload" ;
fuseki:serviceReadWriteGraphStore "data" ;
fuseki:serviceReadGraphStore "get" ;
fuseki:serviceReadGraphStore "" ;
fuseki:dataset <#db2_dataset_read> ;
.

<#db2_dataset_read> rdf:type db2rdf:DB2Dataset ;

# especificar el almacenamiento RDF/conjunto de datos y el esquema
db2rdf:store "staffing" ;
db2rdf:schema "db2admin" ;

# Detalles de la base de datos. Especifique un jdbcConnectString
# con nombre de usuario y contraseña o especifique un jndiDataSource
db2rdf:jdbcConnectString "jdbc:db2://localhost:50000/RDFSAMPL" ;
db2rdf:user "db2admin" ;
db2rdf:password "db2admin" .

#db2rdf:jndiDataSource "jdbc/DB2RDFDS" .
```

6. Ejecute los mandatos siguientes desde la línea de mandatos:

```
SET CLASSPATH=./fuseki-server.jar;<DB2_FOLDER>/rdf/lib/rdfstore.jar;
<DB2_FOLDER>/rdf/lib/wala.jar;<DB2_FOLDER>/rdf/lib/antlr-3.3-java.jar;
<DB2_FOLDER>/rdf/lib/commons-logging-1-0-3.jar;<DB2_FOLDER>/java/db2jcc4.jar;
%CLASSPATH%;

java org.apache.jena.fuseki.FusekiCmd --config config.ttl
```

Resultados

1. Inicie el navegador y cargue el URL `localhost:3030`. La página Fuseki se carga en la ventana del navegador.
2. Pulse el hiperenlace `Control Panel` en la ventana del navegador y seleccione `Dataset` en la lista desplegable. La lista desplegable contiene todos los conjuntos de datos que figuran en el archivo `config.ttl`. Seleccione el conjunto de datos que ha configurado.
3. Ejecute una consulta SPARQL usando las opciones de la GUI. La primera sección se usa para consultar el conjunto de datos con el lenguaje de consultas SPARQL. La segunda sección sirve para modificar el conjunto de datos con actualizaciones SPARQL. La tercera sección sirve para cargar datos en los gráficos del conjunto de datos.

Capítulo 11. Mantenimiento de un almacenamiento RDF

Cree un almacenamiento RDF de DB2 por omisión u optimizado y cargue datos RDF en él. Realice tareas de mantenimiento en el almacenamiento RDF para conseguir un rendimiento óptimo de las consultas.

Actualización de estadísticas en un almacenamiento RDF

Puede actualizar las estadísticas para un almacenamiento RDF.

Acerca de esta tarea

Para garantizar un óptimo rendimiento de las consultas de rendimiento en un almacenamiento RDF, se mantiene una combinación de estadísticas de base de datos DB2 y estadísticas específicas de almacenamiento RDF para un almacenamiento RDF.

Como parte del proceso de creación del almacenamiento RDF, se crean perfiles de mandato **RUNSTATS** de DB2 para las tablas de datos del almacenamiento RDF. Compruebe que el parámetro **AUTORUNSTATS** está habilitado para que DB2 actualice automáticamente las estadísticas de distribución de las tablas.

En el caso de que haya realizado operaciones concretas que hayan actualizado grandes cantidades de datos en el almacenamiento RDF, es mejor invocar manualmente la recopilación de estadísticas de DB2, en lugar de esperar a que se lleven a cabo las actualizaciones automáticas. Para invocar manualmente la recopilación de estadísticas de DB2, invoque el siguiente mandato para cada una de las tablas de datos en el almacenamiento RDF:

```
db2 RUNSTATS ON <nombre_tabla> USE PROFILE
```

Las tablas de datos son direct_primary, direct_secondary, reverse_primary, reverse_secondary y long_strings.

Para garantizar que las consultas SPARQL generan consultas SQL eficientes, es importante que se almacenen los tripletes de RDF más habituales por sujeto, predicado y objeto RDF en la tabla Topk_Statsdel almacenamiento RDF. Para recopilar automáticamente información sobre los tripletes RDF más habituales, como parte de la creación del almacenamiento RDF, se crea y se registra una tarea administrativa de DB2 con el planificador de tareas administrativas de DB2.

No obstante, el intervalo de esta tarea no se establece automáticamente y, por lo tanto, no se ejecutará hasta que lo establezca. Utilice el parámetro **schedule** del mandato **SETSTATSSCHEDULE**. Este parámetro acepta una serie de formato CRON estándar para representar el intervalo en el que se debe invocar la recopilación de estadísticas. Generalmente, no es necesario establecer la frecuencia de esta tarea en un valor menor a una hora.

Procedimiento

- Utilice el mandato siguiente para establecer la planificación para la recopilación de estadísticas cada hora a la hora en punto para el almacenamiento rdfStore2 en la base de datos DB1.

© Copyright IBM Corp. 2013

```
setstatsschedule rdfStore2 -host localhost -port 60000
-db DB1 -user db2admin -password XXX
-schema db2admin -schedule "*/59 * * * *"
```

- En el caso de que haya realizado operaciones concretas que hayan actualizado grandes cantidades de datos en el almacenamiento RDF, recopile manualmente estas estadísticas en lugar de esperar al planificador. Para invocar esto manualmente, ejecute el mandato **updaterdfstorestats**.

  Por ejemplo, utilice el mandato siguiente para actualizar las estadísticas del almacenamiento rdfstore2 en la base de datos DB1 y el esquema db2admin para el sistema principal localhost y el puerto 60000:

  ```
  updaterdfstorestats rdfstore2 -host localhost -port 60000 -db DB1
  -user db2admin -password XXX
  -schema db2admin
  ```

Conversión de un almacenamiento por omisión en un almacenamiento RDF optimizado

Si ha empezado con un almacenamiento RDF por omisión, puede utilizar el mandato **reorgcheckrdfstore** para verificar si el almacenamiento debe optimizarse. Puede hacer esto para un almacenamiento optimizado en el que la correlación de predicados haya cambiado de forma significativa. A continuación, utilice los mandatos **reorgrdfstore** y **reorgswitchrdfstore** para pasar a un almacenamiento RDF optimizado.

En primer lugar, verifique si un almacenamiento RDF necesita reorganizarse. A continuación, cree tablas reorganizadas para un almacenamiento RDF. Por último, pase a las tablas reorganizadas.

Verificación de si un almacenamiento RDF debe reorganizarse

Si la correlación de predicados de los datos RDF ha cambiado y se han insertado grandes cantidades de datos con este cambio, el número y la longitud de columnas de las tablas del almacenamiento, así como la asignación de predicados a las columnas, podrían dejar de ser óptimos para los datos.

Si estos valores ya no son óptimos, el rendimiento de las consultas e inserciones del almacenamiento RDF podría verse afectado de forma negativa.

Procedimiento

Para determinar si un almacenamiento necesita una reorganización, emita el mandato **reorgcheckrdfstore**. Por ejemplo, para el almacenamiento myRdfStore de la base de datos DB1, emita el mandato siguiente:

```
reorgcheckrdfstore myRdfStore -db DB1
-user db2admin -password db2admin
```

Si las tablas no necesitan reorganizarse, se mostrará el siguiente mensaje:

```
No reorganization is required for store myRdfStore.
```

Si alguna tabla necesita reorganizarse, la salida enumera las tablas, tal como se muestra en el ejemplo siguiente:

```
TABLENAME                REORG    ColsRequired    ColSize
-------------------------------------------------------------
direct_primary_hash       true               5        118
reverse_primary_hash      true               3        118
-------------------------------------------------------------
```

Qué hacer a continuación

Utilice el mandato **reorgrdfstore** para crear tablas reorganizadas. Para obtener información detallada, consulte el tema sobre la creación de tablas reorganizadas en un almacenamiento RDF.

Creación de tablas reorganizadas para un almacenamiento RDF

Si el mandato **reorgcheckrdfstore** indica que debe reorganizar tablas, utilice el mandato **reorgrdfstore** para crear y llenar las tablas reorganizadas.

Antes de empezar

Asegúrese de que no se realicen actualizaciones en el almacenamiento RDF mientras el mandato **reorgrdfstore** se esté ejecutando. Para ello, cambie todas las tablas del almacenamiento RDF a modo de sólo lectura.

Procedimiento

Para reorganizar tablas en un almacenamiento RDF, emita el mandato **reorgrdfstore**. Por ejemplo, para el almacenamiento myRdfStore de la base de datos DB1, emita el mandato siguiente para crear una nueva tabla reorganizada para la tabla direct_primary_hash:

```
reorgrdfstore myRdfStore -db DB1
-user db2admin -password db2admin -table direct_primary_hash -newtablenamesprefix reorgd
```

El nombre de la nueva tabla es reorgd_*nombre_tabla_original*, ya que se ha especificado reorgd como prefijo para el nombre de la nueva tabla en el mandato. El valor *nombre_tabla_original* representa el nombre que se ha establecido para la tabla direct_primary_hash en el archivo de propiedades `objectNames.props`.

Resultados

El tiempo para reorganizar tablas depende de la cantidad de datos que haya en el almacenamiento.

Si no se necesita reorganizar la tabla que especifique en el parámetro **table**, un mensaje lo indicará.

Qué hacer a continuación

Cambie el almacenamiento para que utilice las nuevas tablas reorganizadas. Para obtener información detallada, consulte el tema en el que se explica cómo conmutar a las tablas reorganizadas en un almacenamiento RDF.

Cómo conmutar a las tablas reorganizadas en un almacenamiento RDF

El mandato **reorgrdfstore** crea tablas reorganizadas, pero el almacenamiento RDF no utiliza esas tablas nuevas hasta que se emita el mandato **reorgswitchrdfstore**.

Antes de empezar

Asegúrese de que todos los clientes del almacenamiento RDF estén desconectados.

Procedimiento

Para actualizar un almacenamiento RDF con tablas reorganizadas, emita el mandato **reorgswitchrdfstore**. Por ejemplo, para el almacenamiento myRdfStore de la base de datos DB1, emita el mandato siguiente:

```
reorgswitchrdfstore myRdfStore -db DB1
-user db2admin -password db2admin
```

Resultados

El mandato **reorgswitchrdfstore** renombra las tablas originales como old_*nombre_tabla_original*, y las tablas reorganizadas utilizan los nombres originales.

Cuando se conmuta un almacenamiento RDF a un almacenamiento reorganizado, las tablas no se renombran. El almacenamiento se modificará para que empiece a utilizar las nuevas tablas, y las tablas antiguas permanecerán inalteradas.

Qué hacer a continuación

Vuelva a conectar los clientes al almacenamiento.

Si es necesario, descarte las tablas antiguas.

Capítulo 12. Mandatos RDF

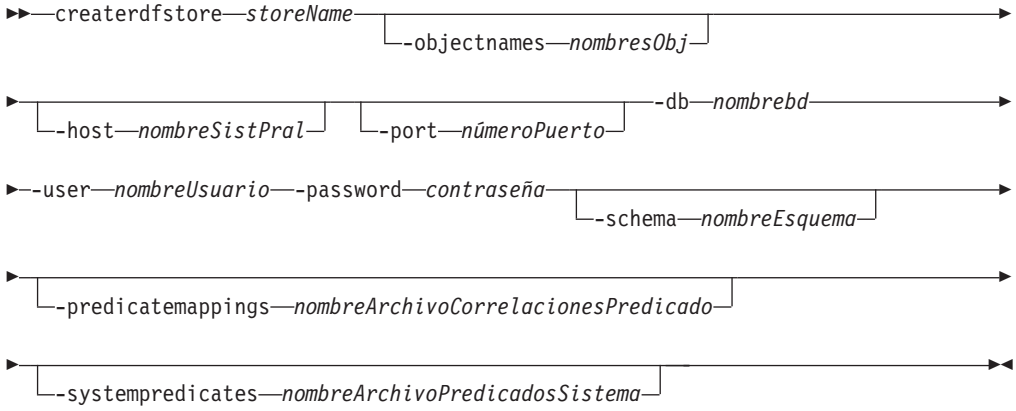
Los mandatos de RDF proporcionan un amplio control de usuario, además de la capacidad de personalización. Puede utilizar estos mandatos para ejecutar diversas tareas relacionadas con la creación, administración y consulta de almacenamientos.

createrdfstore, mandato

El mandato **createrdfstore** crea un almacenamiento RDF vacío sin datos.

Para crear un almacenamiento RDF optimizado que utilice datos existentes, ejecute el mandato **createrdfstoreandloader**.

Sintaxis del mandato



Parámetros del mandato

-storename *nombreAlmacenamiento*
Especifica un nombre para el almacenamiento RDF. Asegúrese de que el nombre cumpla las normas para los nombres de tabla de base de datos DB2.

-objectnames *nombresObj*
Especifica un archivo de propiedades de Java que enumera los nombres de las tablas de almacenamiento RDF. Este archivo puede ser cualquier nombre de archivo válido, pero es necesario que tenga la extensión ".properties".

Si no especifica el parámetro **objectNames**, en su lugar se utilizarán los nombres de tabla generados por el sistema.

-host *nombreSistPral*
Especifica el sistema principal donde se encuentra la base de datos.

-port *númeroPuerto*
Especifica el número de puerto de la base de datos.

-db *nombrebd*
Especifica la base de datos con la que se establece la conexión. El tamaño de página de base de datos mínimo es de 32 KB.

-user *nombreUsuario*
Especifica el nombre de autorización utilizado para establecer la conexión.

© Copyright IBM Corp. 2013

-password *contraseña*
Especifica la contraseña utilizada para establecer la conexión.

-schema *nombreEsquema*
Especifica el esquema de base de datos en el que se creará el almacenamiento RDF.

-predicatemappings *nombreArchivoCorrelacionesPredicado*
En el fixpack 2 y en los fixpacks posteriores de DB2 versión 10.1, especifica la vía de acceso del archivo que contiene las correlaciones de predicado que deben utilizarse en el almacenamiento. Las correlaciones se producen entre los predicados y las columnas que tienen asignadas. Las correlaciones se calculan basándose en las ocurrencias de predicados.

-systempredicates *NombreArchivoPredicadossistema*
En el fixpack 2 y en los fixpacks posteriores de DB2 versión 10.1, especifica el archivo de propiedades que contiene los predicados de filtro que se aplicarán a la consulta. Estos predicados del sistema se almacenan en un almacenamiento RDF para permitir el control de acceso de nivel de gráfico.

Ejemplo

Ejemplo 1: El mandato siguiente crea un almacenamiento denominado rdfStore1 en la base de datos DB1 con el puerto 60000 y el esquema db2admin en el sistema principal localhost:

```
createrdfstore rdfStore1 -host localhost -port 60000 -db DB1
-user db2admin -password XXX -schema db2admin
```

Ejemplo 2: El mandato siguiente crea un almacenamiento denominado rdfstore2 en la base de datos DB1 con el puerto 60000 y el esquema db2admin utilizando los predicados del sistema del archivo syspreds.props y las correlaciones de predicado del archivo `predicatemappings.nq`.

```
createrdfstore rdfStore1 -host localhost -port 60000 -db DB1
-user db2admin -password XXX -schema db2admin
-predicatemappings predicatemappings.nq -systempredicates syspreds.props
```

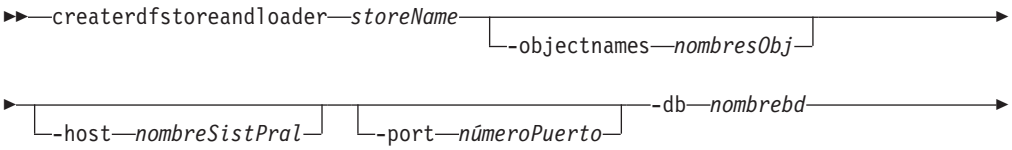
Notas de uso

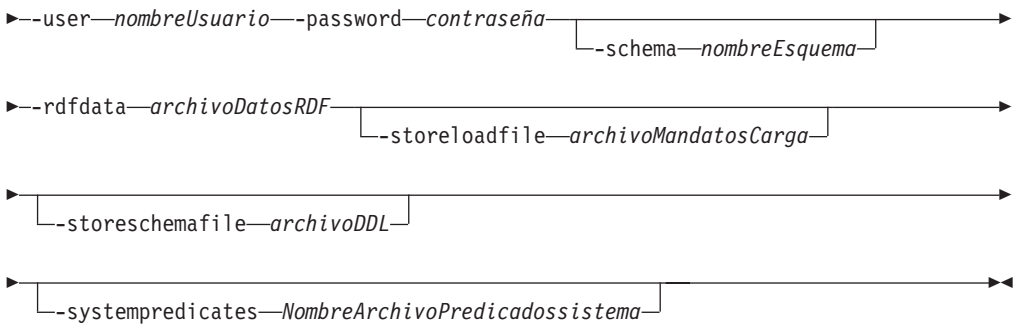
Los nombres de mandato y de parámetro deben escribirse en minúsculas.

Mandato createrdfstoreandloader

El mandato **createrdfstoreandloader** analiza los datos RDF y crea un almacenamiento RDF vacío cuyo esquema se optimiza para los datos RDF existentes. Este mandato también genera los archivos de carga y los mandatos para cargar el almacenamiento desde estos archivos de carga.

Sintaxis del mandato





Parámetros del mandato

-storename *nombreAlmacenamiento*
Especifica un nombre para el almacenamiento RDF. Asegúrese de que el nombre cumpla las normas para los nombres de tabla de servidor de bases de datos DB2 y que sea exclusivo en el esquema de base de datos.

-objectnames *nombresObj*
Especifica un archivo de propiedades de Java que enumera los nombres de las tablas de almacenamiento RDF. Este archivo puede ser cualquier nombre de archivo válido, pero es necesario que tenga la extensión ".properties".

Si no especifica el parámetro **objectNames**, en su lugar se utilizarán los nombres de tabla generados por el sistema.

-host *nombreSistPral*
Especifica el sistema principal donde se encuentra la base de datos.

-port *númeroPuerto*
Especifica el número de puerto de la base de datos.

-db *nombrebd*
Especifica la base de datos con la que se establece la conexión. El tamaño de página de base de datos mínimo es de 32 KB.

-user *nombreUsuario*
Especifica el nombre de autorización utilizado para establecer la conexión.

-password *contraseña*
Especifica la contraseña utilizada para establecer la conexión.

-schema *nombreEsquema*
Especifica el esquema de base de datos en el que se creará el almacenamiento RDF.

-rdfdata *archivoDatosRDF*
Especifica un archivo con los datos RDF a partir de los cuales se crea el esquema de almacenamiento RDF optimizado. Asimismo, se crean archivos de carga basándose en los datos RDF de este archivo.

-storeloadfile *archivoMandatosCarga*
Especifica la vía de acceso de salida del archivo con los mandatos para cargar datos en el almacenamiento RDF. Puede ejecutar este archivo mediante el procesador de línea de mandatos (CLP) de DB2.

Si no especifica el parámetro **-storeloadfile**, se creará un archivo denominado `loadCommands.sql` en la carpeta actual.

Los archivos de carga se crean en la carpeta donde se crea el archivo con los mandatos.

-storeschemafile *archivoDDL*
Especifica la vía de acceso de salida y el archivo donde se generan los scripts de DDL para el almacenamiento.

Si no especifica el parámetro **-storeschemafile**, no se creará el archivo de scripts de DDL.

-systempredicates *NombreArchivoPredicadossistema*
En el fixpack 2 y en los fixpacks posteriores de DB2 versión 10.1, especifica el archivo de propiedades que contiene los predicados de filtro que se aplicarán a la consulta. Estos predicados del sistema se almacenan en un almacenamiento RDF de DB2 RDF para permitir el control de acceso de nivel de gráfico.

Ejemplo

El mandato siguiente crea un almacenamiento denominado rdfStore1 en la base de datos DB1, con el puerto 60000 y el esquema db2admin en el sistema principal localhost. El archivo de datos RDF de entrada es `myRdfData.nq`, y el nombre del archivo generado de carga de almacenamiento es `load.sql` en el directorio `./rdfLoader/`.

```
createrdfstoreandloader rdfStore1 -host localhost -port 60000 -db DB1
-user db2admin -password XXX -schema db2admin
-rdfdatafile ./myRdfData.nq -storeloadfile ./rdfLoader/load.sql
```

Notas de uso

Los nombres de mandato y de parámetro deben escribirse en minúsculas.

El directorio desde el que se ejecuta el mandato no debe contener ningún espacio en su vía de acceso. Los parámetros **storeloadfile** y **storeschemafile** tampoco deben contener espacios en su vía de acceso. En las plataformas Windows, si alguna vía de acceso especificada contiene un espacio en el nombre de la carpeta o del archivo, la serie entera deberá estar entre comillas.

En las plataformas Windows, el mandato **createrdfStoreAndLoader** necesita la aplicación CygWin. Para este mandato, se necesita la versión 4.0 o una versión posterior del programa de utilidad Gawk. Asimismo, para este mismo mandato se necesita la versión 8.14 o una versión posterior del programa de utilidad Core. Después de instalar CygWin, añada *<directorio_instalación_CygWin>*`/bin` a la variable de entorno PATH. Si no tiene CygWin en la vía de acceso, se visualizará el siguiente mensaje de error cuando ejecute el mandato:

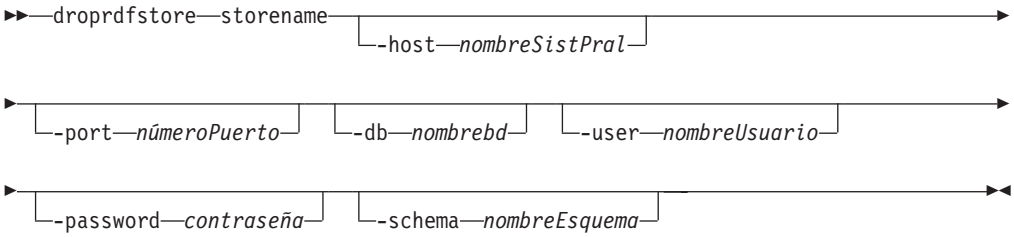
```
'No se puede ejecutar el programa "sh": CreateProcess error=2, El sistema no
encuentra el archivo especificado.'
```

En las plataformas Windows, puede iniciar el mandato **createrdfStoreAndLoader** desde la línea de mandatos CygWin o desde la vía de mandatos por omisión. Cuando se utiliza una línea de mandatos de CygWin, ninguna de las vías de acceso de archivo (-rdfdata, -storeloadfile, -storeschemafile, -objectnames) debe contener el prefijo 'cygdrive'. En su lugar, utilice una vía de acceso de Windows como `C:\.....`.

Mandato droprdfstore

El mandato **droprdfstore** elimina un almacenamiento RDF existente.

Sintaxis del mandato



Parámetros del mandato

-storename
Especifica un nombre que identifica el almacenamiento triple en una base de datos o esquema.

-host *nombreSistPral*
Especifica el sistema principal donde se encuentra la base de datos.

-port *númeroPuerto*
Especifica el número de puerto de la base de datos.

-db *nombrebd*
Especifica la base de datos con la que se establece la conexión.

-user *nombreUsuario*
Especifica el nombre de autorización utilizado para establecer la conexión.

-password *contraseña*
Especifica la contraseña utilizada para establecer la conexión.

-schema *nombreEsquema*
Especifica el esquema de base de datos en el que se ubicará RDF.

Ejemplo

Emita el mandato **droprdfstore** para eliminar un almacenamiento denominado rdfStore4 en la base de datos DB1 y el esquema db2admin para el sistema principal localhost y el puerto 60000.

```
droprdfstore rdfStore4 -host localhost -port 60000 -db DB1
-user db2admin -password XXX
-schema db2admin
```

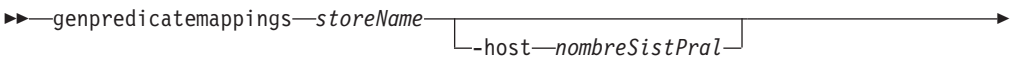
Notas de uso

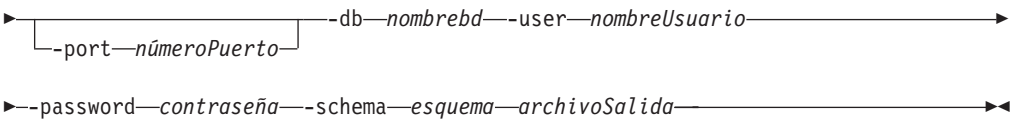
- Los nombres de los mandatos y parámetros deben emitirse en minúsculas.

Mandato genpredicatemappings

En el fixpack 2 y en los fixpacks posteriores de DB2 versión 10.1, el mandato **genpredicatemappings** genera correlaciones de predicado basadas en la correlación de predicado de un almacenamiento RDF.

Sintaxis del mandato





Parámetros del mandato

storeName
Especifica el almacenamiento RDF.

-host *nombreSistPral*
Especifica el sistema principal donde se encuentra la base de datos.

-port *númeroPuerto*
Especifica el número de puerto de la base de datos.

-db *nombrebd*
Especifica la base de datos con la que se establece la conexión.

-user *nombreUsuario*
Especifica el nombre de autorización utilizado para establecer la conexión.

-password *contraseña*
Especifica la contraseña utilizada para establecer la conexión.

-schema *nombreEsquema*
Especifica el esquema de base de datos para el almacenamiento RDF.

archivoSalida
Especifica la vía de acceso y el nombre del archivo en el que se graban las correlaciones. Si no se especifica un archivo de salida, esta se grabará en la consola.

Ejemplo

El mandato siguiente genera correlaciones de predicado para un almacenamiento RDF denominado MyStore y graba la salida en un archivo:

```
genpredicatemappings MyStore -db RDFSAMPL -user db2admin
-password db2admin "C:\MyStore_predicate_mappings.txt"
```

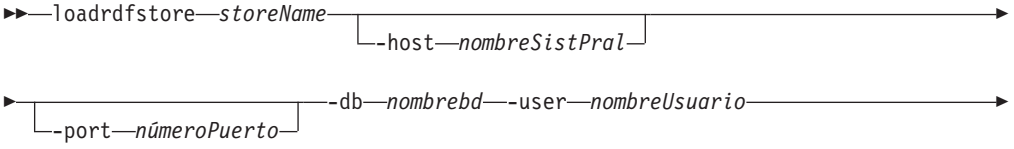
Notas de uso

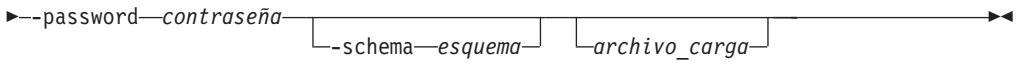
Los nombres de mandato y de parámetro deben escribirse en minúsculas.

Mandato loadrdfstore

En el fixpack 2 y los fixpacks posteriores de DB2 versión 10.1, el mandato **loadrdfstore** carga tripletes en un almacenamiento RDF ya existente.

Sintaxis del mandato





Parámetros del mandato

storeName
Especifica el almacenamiento RDF en el que se efectuará la consulta.

-host *nombreSistPral*
Especifica el sistema principal donde se encuentra la base de datos.

-port *númeroPuerto*
Especifica el número de puerto de la base de datos.

-db *nombrebd*
Especifica la base de datos con la que se establece la conexión.

-user *nombreUsuario*
Especifica el nombre de autorización utilizado para establecer la conexión.

-password *contraseña*
Especifica la contraseña utilizada para establecer la conexión.

-schema *nombreEsquema*
Especifica el esquema de base de datos para el almacenamiento RDF.

archivo_carga
Especifica el archivo que contiene los tripletes que se cargarán. El archivo puede ser de tipo nquad, ntriple o rdfxml. Los archivos ntriple y rdfxml se cargan únicamente en el gráfico por omisión.

- La extensión de los archivos nquad es .nq.
- La extensión de los archivos ntriples es .nt.
- La extensión de los archivos rdfxml es .rdf o .xml.

Ejemplo

El mandato siguiente carga un archivo que contiene tripletes para un almacenamiento RDF denominado myStore en la base de datos RDFSAMPL.

```
loadrdfstore myStore -db RDFSAMPL -user db2admin
-password db2admin -schema db2admin c:\simple.nq
```

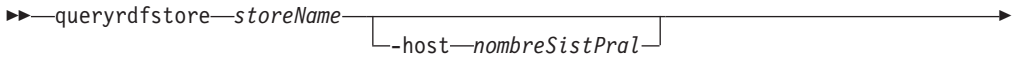
Notas de uso

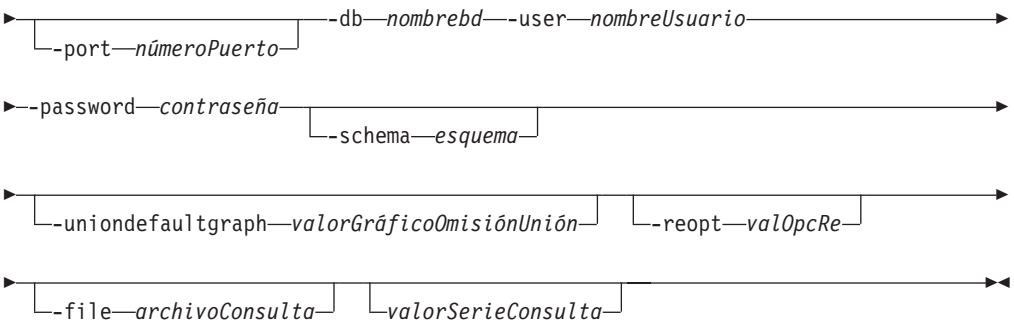
Los nombres de mandato y de parámetro deben escribirse en minúsculas.

Mandato queryrdfstore

En el fixpack 2 y en los fixpacks posteriores de DB2 versión 10.1, puede utilizar el mandato **queryrdfstore** para consultar un almacenamiento RDF desde la línea de mandatos. Esta consulta puede ejecutarse desde un archivo o especificándola en línea como argumento del mandato **queryrdfstore**.

Sintaxis del mandato





Parámetros del mandato

storeName
Especifica el almacenamiento RDF en el que se efectuará la consulta.

-host *nombreSistPral*
Especifica el sistema principal donde se encuentra la base de datos.

-port *númeroPuerto*
Especifica el número de puerto de la base de datos.

-db *nombrebd*
Especifica la base de datos con la que se establece la conexión.

-user *nombreUsuario*
Especifica el nombre de autorización utilizado para establecer la conexión.

-password *contraseña*
Especifica la contraseña utilizada para establecer la conexión.

-schema *nombreEsquema*
Especifica el esquema de base de datos para el almacenamiento RDF.

-uniondefaultgraph*valorGráficoOmisiónUnión*
Especifica si se está utilizando un gráfico por omisión de unión. El valor puede ser `true` o `false`.

-reopt*valorOpcRe*
Especifica las opciones de repetición para ejecutar la consulta. Las opciones son `once`, `always` o `none`. El valor por omisión es `none`.

-file*archivoConsulta*
Especifica el archivo que contiene la consulta SPARQL.

valorSerieConsulta
Especifica la consulta SPARQL ejecutada como una serie.

Ejemplo

Ejemplo 1: el mandato siguiente especifica una consulta de tripletes en un almacenamiento RDF denominado myStore y en una base de datos RDFSAMPL de un sistema local, con el parámetro **unionDefaultGraph** establecido en `true`. Puede consultar todos los tripletes de un almacenamiento RDF denominado myStore en una base de datos RDFSAMPL del sistema local, con el parámetro **unionDefaultGraph** establecido en `true`.

```
queryrdfstore myStore -db RDFSAMPL -user db2admin
-password db2admin -schema db2admin
-uniondefaultgraph true "select * where {?s ?p ?v}"
```

Ejemplo 2: el mandato siguiente especifica el uso de la consulta dentro de un archivo de texto.

```
queryrdfstore myStore -db RDFSAMPL -user db2admin
-password db2admin -schema db2admin
-uniondefaultgraph true -file "C:\query.txt"
```

Notas de uso

Los nombres de mandato y de parámetro deben escribirse en minúsculas.

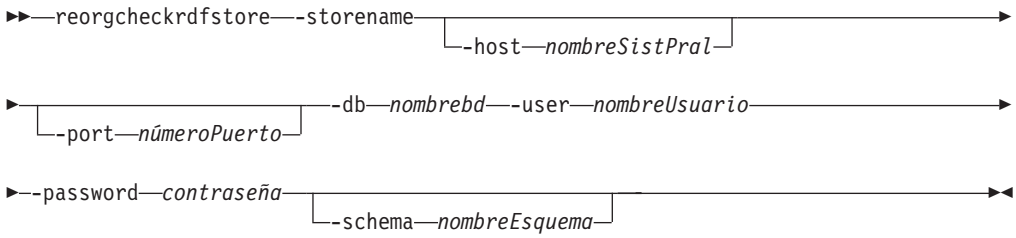
Puede especificar la consulta dentro de un archivo o como parámetro del mandato, pero no ambas cosas a la vez.

Mandato reorgcheckrdfstore

El mandato **reorgcheckrdfstore** comprueba si se debe reorganizar el almacenamiento RDF.

El mandato **reorgcheckrdfstore** identifica si no son óptimos el número de columnas o las longitudes de columna en una o varias tablas de almacenamiento RDF. Tras ejecutar el mandato **reorgcheckrdfstore**, emita el mandato **reorgrdfstore** para reorganizar las tablas identificadas y así mejorar el rendimiento de las consultas.

Sintaxis del mandato



Parámetros del mandato

-storename
Especifica el nombre de un almacenamiento triple en una base de datos o esquema.

-host *nombreSistPral*
Especifica el sistema principal donde se encuentra la base de datos.

-port *númeroPuerto*
Especifica el número de puerto de la base de datos.

-db *nombrebd*
Especifica la base de datos con la que se establece la conexión.

-user *nombreUsuario*
Especifica el nombre de autorización utilizado para establecer la conexión.

-password *contraseña*
Especifica la contraseña utilizada para establecer la conexión.

-schema *nombreEsquema*
Especifica el esquema de base de datos en el que se ubicará el almacenamiento RDF.

Ejemplo

El mandato siguiente comprueba si debe reorganizarse un almacenamiento denominado rdfStore3. El almacenamiento se encuentra en la base de datos DB1, con el puerto 60000 y el esquema db2admin en el sistema principal localhost.

```
reorgcheckrdfstore rdfStore3 -host localhost -port 60000 -db DB1
-user db2admin -password XXX
-schema db2admin
```

En la salida del mandato **reorgcheckrdfstore** únicamente se listan las tablas que precisan reorganizarse. La salida contiene las columnas de datos siguientes:

tablename
Nombre lógico de la tabla de almacenamiento RDF.

reorg Valor true o false que indica si la tabla precisa reorganizarse.

colsrequired
Número requerido de columnas.

colsize
Longitud óptima de columna.

Notas de uso

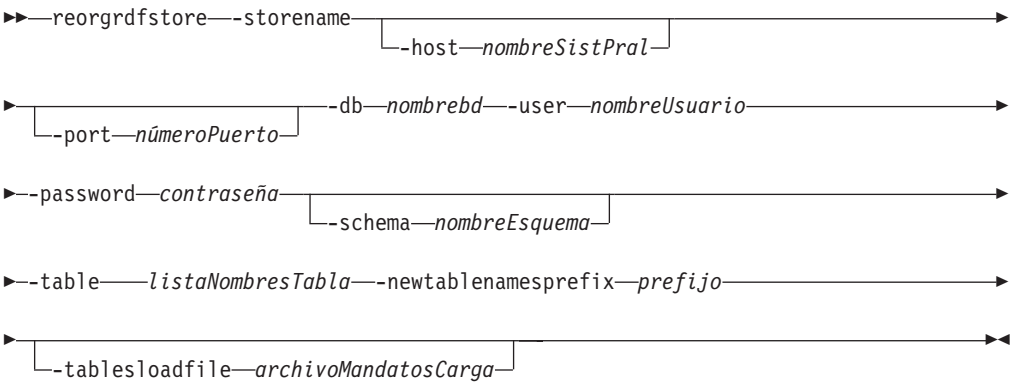
Los nombres de mandato y de parámetro deben escribirse en minúsculas.

Mandato reorgrdfstore

El mandato **reorgrdfstore** crea nuevas tablas reorganizadas con el número y la longitud de columna óptimos para un almacenamiento RDF basado en los datos existentes en el almacenamiento. De forma opcional, el mandato también puede generar archivos con el DDL de las nuevas tablas y los mandatos de carga para cargar datos en las tablas reorganizadas.

Este mandato puede tardar un poco en ejecutarse en función de la cantidad de datos que se necesita descargar, la creación de archivos de carga y la carga de datos en las tablas reorganizadas. Mientras el mandato se ejecuta, las tablas de datos de almacenamiento RDF deberían definirse como de sólo lectura.

Sintaxis del mandato



Parámetros del mandato

-storename
Especifica el nombre del almacenamiento triple en una base de datos o esquema.

-host *nombreSistPral*
Especifica el sistema principal donde se encuentra la base de datos.

-port *númeroPuerto*
Especifica el número de puerto de la base de datos.

-db *nombrebd*
Especifica la base de datos con la que se establece la conexión.

-user *nombreUsuario*
Especifica el nombre de autorización utilizado para establecer la conexión.

-password *contraseña*
Especifica la contraseña utilizada para establecer la conexión.

-schema *nombreEsquema*
Especifica el esquema de base de datos en el que se ubicará el almacenamiento RDF.

-table *listaNombresTabla*
Especifica la lista de nombres lógicos de las tablas que se deben reorganizar. Utilice el carácter de barra vertical para separar varios nombres de tabla. La lista debe estar encerrada entre comillas dobles.

-newtablenamesprefix *prefijo*
Especifica un prefijo para las nuevas tablas.

-tablesloadfile *archivoMandatosCarga*
Especifica la vía de acceso de salida del archivo que contiene el DDL de las nuevas tablas y los mandatos para cargar datos en las tablas reorganizadas.

Si no especifica el parámetro **-tablesloadfile**, se creará un archivo denominado `loadCommands.sql` en la carpeta actual.

Los archivos de carga se crean en la carpeta donde se ha creado el archivo de mandatos de carga. Estos archivos se pueden ejecutar mediante una ventana de CLP.

Ejemplo

El mandato siguiente reorganiza las tablas de un almacenamiento denominado rdfStore3 en la base de datos DB1, con el puerto 60000 y el esquema db2admin en el sistema principal localhost para las tablas direct_primary_hash y reverse_primary_hash, utilizando el prefijo reorg para los nombres de las nuevas tablas:

```
reorgrdfstore rdfStore3 -host localhost -port 60000
-db DB1 -user db2admin -password XXX
-schema db2admin -table "direct_primary_hash|reverse_primary_hash"
-newtablenamesprefix reorg
```

Notas de uso

- El mandato **reorgrdfstore** debe emitirse en la máquina en la que estén ubicados el servidor de bases de datos DB2 y el almacenamiento RDF.
- Los nombres de mandato y de parámetro deben escribirse en minúsculas.
- Cuando el mandato finalice correctamente, las tablas se cargarán de forma automática. Este archivo se puede ejecutar mediante una ventana de CLP.

- Cuando el mandato finaliza, el almacenamiento continúa utilizando las tablas antiguas y deberá cambiarse para que empiece a utilizar las nuevas tablas. Para utilizar las tablas reorganizadas, emita el mandato **reorgswitchrdfstore**.

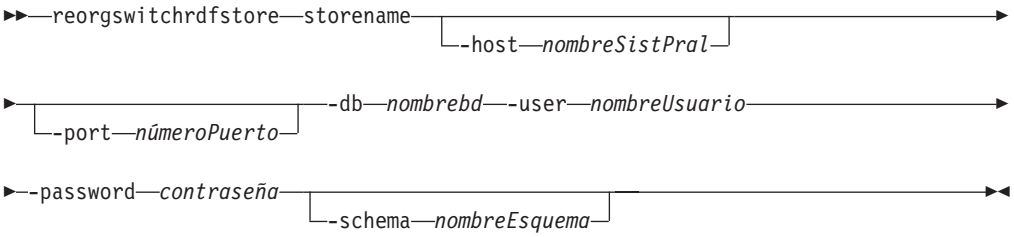
Mandato reorgswitchrdfstore

El mandato **reorgswitchrdfstore** conmuta un almacenamiento RDF para que utilice las tablas recientemente reorganizadas que se han creado mediante el mandato **reorgrdfstore**.

Todos los clientes de este almacenamiento RDF deben desconectarse antes de realizar esta operación, y volverse a conectar después de que el mandato haya finalizado.

El tiempo de finalización no depende de la cantidad de datos que haya en el almacenamiento.

Sintaxis del mandato



Parámetros del mandato

-storename
Especifica un nombre que identifica el almacenamiento triple en una base de datos o esquema.

-host *nombreSistPral*
Especifica el sistema principal donde se crea un almacenamiento.

-port *númeroPuerto*
Especifica el número de puerto.

-db *nombrebd*
Especifica la base de datos con la que se establece la conexión.

-user *nombreUsuario*
Especifica el nombre de autorización utilizado para establecer la conexión.

-password *contraseña*
Especifica la contraseña utilizada para establecer la conexión.

-schema *nombreEsquema*
Especifica el esquema de base de datos en el que se ubicará el almacenamiento RDF.

Ejemplo

Emita el mandato reorgswitchrdfstore para que conmute a las tablas reorganizadas el almacenamiento denominado rdfStore3 en la base de datos DB1 y el esquema db2admin para el sistema principal localhost y el puerto 60000.

```
reorgswitchrdfstore rdfStore3 -host localhost -port 60000 -db DB1
-user db2admin -password XXX
-schema db2admin
```

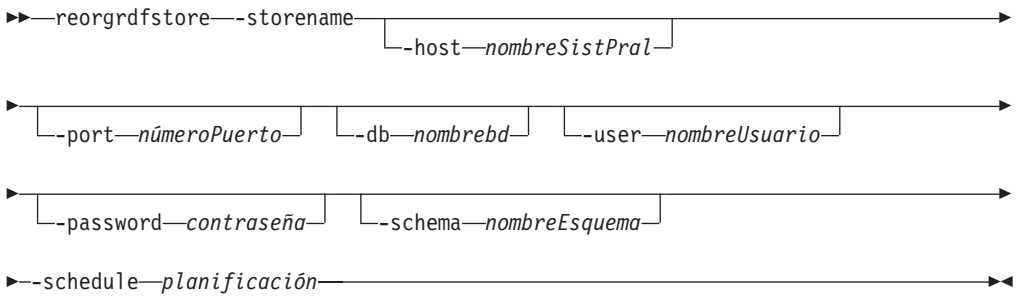
Notas de uso

- Los nombres de los mandatos y parámetros deben emitirse en minúsculas.

Mandato setstatsschedule

El mandato **setstatsschedule** planifica la actualización automática de estadísticas del almacenamiento RDF.

Sintaxis del mandato



Parámetros del mandato

-storename
Especifica el nombre del almacenamiento triple en una base de datos o esquema.

-host *nombreSistPral*
Especifica el sistema principal donde se encuentra la base de datos.

-port *númeroPuerto*
Especifica el número de puerto de la base de datos.

-db *nombrebd*
Especifica la base de datos con la que se establece la conexión.

-user *nombreUsuario*
Especifica el nombre de autorización utilizado para establecer la conexión.

-password *contraseña*
Especifica la contraseña utilizada para establecer la conexión.

-schema *nombreEsquema*
Especifica el esquema de base de datos en el que se ubicará el almacenamiento RDF.

-schedule *planificación*
Especifica la planificación de actualizaciones de estadísticas en formato CRON de UNIX. Este parámetro debe especificarse entre comillas dobles.

Ejemplo

El mandato siguiente planifica las actualizaciones automáticas de las estadísticas para que se ejecuten en el minuto 15 de cada hora para un almacenamiento denominado RDFStore en la base de datos RDFDB, con el puerto 60000 y el esquema db2admin en el sistema principal localhost:

```
setStatsSchedule RDFStore -host localhost -port 60000
-db RDFDB -user db2admin -password XXX
-schema db2admin -schedule "15 * * * *"
```

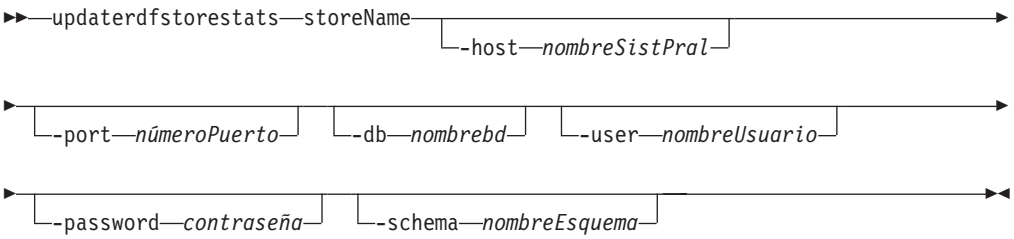
Notas de uso

- Los nombres de mandato y de parámetro deben escribirse en minúsculas.

Mandato updaterdfstorestats

El mandato **updaterdfstorestats** actualiza las estadísticas para que reflejen los datos actuales en un almacenamiento RDF.

Sintaxis del mandato



Parámetros del mandato

-storename
Especifica un nombre para el almacenamiento. El nombre debe ser exclusivo en esa base de datos o esquema.

-host *nombreSistPral*
Especifica el sistema principal donde se encuentra la base de datos.

-port *númeroPuerto*
Especifica el número de puerto de la base de datos.

-db *nombrebd*
Especifica la base de datos con la que se establece la conexión.

-user *nombreUsuario*
Especifica el nombre de autorización utilizado para establecer la conexión.

-password *contraseña*
Especifica la contraseña utilizada para establecer la conexión.

-schema *nombreEsquema*
Especifica el esquema de base de datos en el que se ubicará el almacenamiento RDF.

Ejemplo

El mandato siguiente actualiza las estadísticas para un almacenamiento denominado rdfStore3 en la base de datos DB1, con el puerto 60000 y el esquema db2admin en el sistema principal localhost:

```
updaterdfstorestats rdfStore3 -host localhost -port 60000 -db DB1
-user db2admin -password XXX
-schema db2admin
```

Notas de uso

- Los nombres de mandato y de parámetro deben escribirse en minúsculas.

Parte 2. Apéndices

© Copyright IBM Corp. 2013

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.

© Copyright IBM Corp. 2013

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=swg27009474.

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 3. Información técnica de DB2

Nombre	Número de documento	Copia impresa disponible	Fecha de disponibilidad
Consulta de las API administrativas	SC27-5506-00	Sí	28 de julio de 2013
Rutinas y vistas administrativas	SC27-5507-00	No	28 de julio de 2013
Call Level Interface Guide and Reference Volume 1	SC27-5511-00	Sí	28 de julio de 2013
Call Level Interface Guide and Reference Volume 2	SC27-5512-00	Sí	28 de julio de 2013
Consulta de mandatos	SC27-5508-00	Sí	28 de julio de 2013
Database Administration Concepts and Configuration Reference	SC27-4546-00	Sí	28 de julio de 2013
Data Movement Utilities Guide and Reference	SC27-5528-00	Sí	28 de julio de 2013
Database Monitoring Guide and Reference	SC27-4547-00	Sí	28 de julio de 2013
Data Recovery and High Availability Guide and Reference	SC27-5529-00	Sí	28 de julio de 2013
Database Security Guide	SC27-5530-00	Sí	28 de julio de 2013
Guía y consulta de DB2 Workload Management	SC27-5520-00	Sí	28 de julio de 2013
Developing ADO.NET and OLE DB Applications	SC27-4549-00	Sí	28 de julio de 2013
Developing Embedded SQL Applications	SC27-4550-00	Sí	28 de julio de 2013
Desarrollo de aplicaciones Java	SC27-5503-00	Sí	28 de julio de 2013

Tabla 3. Información técnica de DB2 (continuación)

Nombre	Número de documento	Copia impresa disponible	Fecha de disponibilidad
Desarrollo de aplicaciones Perl, PHP, Python y Ruby on Rails	SC11-8358-00	No	28 de julio de 2013
Desarrollo de aplicaciones RDF para servidores de datos IBM	SC11-8357-00	Sí	28 de julio de 2013
Developing User-defined Routines (SQL and External)	SC27-5501-00	Sí	28 de julio de 2013
Getting Started with Database Application Development	GI13-2084-00	Sí	28 de julio de 2013
Iniciación a la instalación y administración de DB2 en Linux y Windows	GI13-2085-00	Sí	28 de julio de 2013
Globalization Guide	SC27-5531-00	Sí	28 de julio de 2013
Instalación de servidores DB2	GC27-5514-00	Sí	28 de julio de 2013
Instalación de clientes de IBM Data Server	GC27-5515-00	No	28 de julio de 2013
Consulta de mensajes Volumen 1	SC27-5523-00	No	28 de julio de 2013
Consulta de mensajes Volumen 2	SC27-5524-00	No	28 de julio de 2013
Net Search Extender Guía de administración y del usuario	SC27-5526-00	No	28 de julio de 2013
Partitioning and Clustering Guide	SC27-5532-00	Sí	28 de julio de 2013
pureXML Guide	SC27-5521-00	Sí	28 de julio de 2013
Spatial Extender Guía del usuario y manual de consulta	SC27-5525-00	No	28 de julio de 2013
SQL Procedural Languages: Application Enablement and Support	SC27-5502-00	Sí	28 de julio de 2013
Consulta de SQL - Volumen 1	SC27-5509-00	Sí	28 de julio de 2013
Consulta de SQL - Volumen 2	SC27-5510-00	Sí	28 de julio de 2013
Guía de Text Search	SC27-5527-00	Sí	28 de julio de 2013
Troubleshooting and Tuning Database Performance	SC27-4548-00	Sí	28 de julio de 2013
Actualización a DB2 Versión 10.5	SC27-5513-00	Sí	28 de julio de 2013

Tabla 3. Información técnica de DB2 (continuación)

Nombre	Número de documento	Copia impresa disponible	Fecha de disponibilidad
Novedades en DB2 Versión 10.5	SC27-5519-00	Sí	28 de julio de 2013
XQuery Reference	SC27-5522-00	No	28 de julio de 2013

Tabla 4. Información técnica específica de DB2 Connect

Nombre	Número de documento	Copia impresa disponible	Fecha de disponibilidad
DB2 Connect Instalación y configuración de DB2 Connect Personal Edition	SC27-5516-00	Sí	28 de julio de 2013
DB2 Connect Instalación y configuración de servidores DB2 Connect	SC27-5517-00	Sí	28 de julio de 2013
Guía del usuario de DB2 Connect	SC27-5518-00	Sí	28 de julio de 2013

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://pic.dhe.ibm.com/infocenter/db2luw/v10r1.

Para los temas de DB2 versión 9.8, el URL del *Centro de información de DB2* es http://pic.dhe.ibm.com/infocenter/db2luw/v9r8/.

Para los temas de DB2 versión 9.7, el URL del *Centro de información de DB2* es http://pic.dhe.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.

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, no se sigan correctamente las instrucciones anteriores.

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.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokio 103-8510, 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

© Copyright IBM Corp. 2013

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

© Copyright IBM Corp. 2013

S

T

U

Impreso en España

SC11-8357-00

Spine information:

IBM DB2 10.5 para Linux, UNIX y Windows

Desarrollo de aplicaciones RDF para servidores de datos IBM