IBM Training

Demonstration 1

Setting up Big SQL federation with IBM DB2 10.5 for LUW

At the end of this demonstration, you will be able to:

- Create a federated system to use with DB2 and run queries that span across Big SQL and DB2 tables.

Demonstration 1: Setting up Big SQL federation with IBM DB2 10.5 for LUW

Demonstration 1: Setting up Big SQL federation with IBM DB2 10.5 for LUW

Purpose:
You will set up DB2 as a data source enabling users to send distributed requests to multiple data sources within a single statement. This will involve connecting to DB2 LUW, creating a wrapper to use with DB2, create the server as the definition of the remote DB2 database, map the user id and password at the federated server to the corresponding user id and password at the data source, create a nickname on a remote object and query the remote table from the federated Big SQL server.

Estimated time: 60 minutes

User/Password: **biadmin/biadmin**
root/dalvm3
db2inst1/ibm2blue

Services Password: **ibm2blue**

Task 1. Create and load DB2 tables (if required).

In DB2, you will be creating a table called sls_order_method_dim. This table contains the order method of each product that was sold. You will be joining the data from this table from several tables in Big SQL. Go back to the previous demonstrations to create and load the Big SQL tables.

1. Ensure BigInsights is running via Ambari.
2. Ensure you can access the BigInsights Home page.
3. Ensure you can access the Big SQL page.
   DB2 has been installed on the image. To create a database, you will need to switch user to **db2inst1** with the password **ibm2blue**. If you have a different instance user, switch to that one.
4. Launch a terminal and type in:
   ```
   su db2inst1
   ```
5. As the db2inst1 user, switch to the /home/db2inst1 directory.
   ```
   cd /home/db2inst1
   ```
6. Launch the DB2 CLP:
   ```
   db2
   ```

7. Type in this command to create a database in DB2.

```
create database db2_db
```

8. Connect to the database.

```
connect to db2_db
```

9. Create the table (Note that the statement is provided in a script under /home/biadmin/labfiles/bigsql/Big_SQL_Federation).

```
CREATE TABLE db2inst1.sls_order_method_dim (
order_method_key INT NOT NULL, order_method_code INT NOT
NULL, order_method_en VARCHAR(90) NOT NULL ,
order_method_de VARCHAR(90), order_method_fr
VARCHAR(90), order_method_ja VARCHAR(90),
order_method_cs VARCHAR(90), order_method_da
VARCHAR(90), order_method_el VARCHAR(90),
order_method_es VARCHAR(90), order_method_fi
VARCHAR(90), order_metod_hu VARCHAR(90), order_method_id
VARCHAR(90), order_method_it VARCHAR(90),
order_method_ko VARCHAR(90), order_method_ms
VARCHAR(90), order_method_nl VARCHAR(90),
order_method_no VARCHAR(90), order_method_pl
VARCHAR(90), order_method_pt VARCHAR(90),
order_method_ru VARCHAR(90), order_method_sc
VARCHAR(90), order_method_sv VARCHAR(90),
order_method_tc VARCHAR(90), order_method_th
VARCHAR(90))
```

The DB2_Federation statements.sql file is located in the */home/biadmin/labfiles/bigsql/BIG_SQL_Federation* folder.

10. Once the table has been created, load the data using this command:

```
import from
/home/db2inst1/GOSALESDW.SLS_ORDER_METHOD_DIM.txt of del
modified by coldel0x09 insert into
db2inst1.sls_order_method_dim
```

This dataset should have been previously copied into that directory, if not, modify the path for your environment.

11. Quit from the db2 shell, type:

```
quit
```

Task 2. Setting up the federation environment.

By default, Big SQL federation is not enabled. Enable it using the following steps.

1. Using the same terminal, switch to the **bigsql** user with the password **ibm2blue**.
2. Start Big SQL's CLP
   ```
   db2
   ```
3. Enable the federation. Type in:
   ```
   UPDATE DBM CFG USING FEDERATED YES
   ```
4. Quit the CLP
   ```
   quit
   ```
5. Restart the Big SQL service via Ambari or in the terminal:
   ```
   /usr/ibmpacks/current/bigsql/bigsql/bin/bigsql stop
   /usr/ibmpacks/current/bigsql/bigsql/bin/bigsql start
   ```

Task 3. Setting up the connection for DB2 LUW.

For this task, you will set up the connection from the Big SQL federation server to the existing DB2 server on the same machine.

1. Start the Big SQL CLP (remember you need to do this as the **bigsql** user)
   ```
   db2
   ```
   You must first catalog the remote server in the federated server system directory. The DB2 database was installed on the port 50002.
2. Run this command to create a db2node in the local catalog:
   ```
   CATALOG TCPIP NODE db2node REMOTE localhost SERVER 50002
   ```
   Note: Our data source in the demonstration is installed on the same node. In a real world environment, you will most likely have your data source on a different machine. In that case, you will need install the client on your machine that can connect to the remote data source.
3. Catalog the database from the remote server. Type in this command:
   ```
   CATALOG DATABASE db2_db AS db2db AT NODE db2node
   ```
4. Refresh the local catalog by execute this command
   ```
   terminate
   ```
5. Test the connection before continuing. Start the CLP again:
   ```
   db2
   ```

6. Connect to the remote database (db2) from the local database (big sql)
   ```
   CONNECT TO db2db USER db2inst1 USING ibm2blue
   ```
   If you successfully connected, then the remote data source has been cataloged correctly.
7. Quit the CLP:
   ```
   quit
   ```

Task 4. Creating the wrapper and server for the data source.

A wrapper is associated with a single library file consisting of routines that are used to access the data source. They can be written as C++ or Java applications. One wrapper is required for each data source type, regardless of the number of data sources of that particular type. If the data source supports SQL, the queries are submitted in SQL. If the data sources support a different language, then the queries are translated into the native query language or API calls.

1. Start the Big SQL CLP (as the bigsql user)
   ```
   db2
   ```
2. To create a wrapper and a remote server, you will need to be within a local Big SQL database. Connect to the bigsql database. Type in:
   ```
   connect to bigsql
   ```
3. Execute this command in the CLP to create a wrapper for the DB2 data source:
   ```
   CREATE WRAPPER db2wrapper LIBRARY 'libdb2drda.so'
   ```
4. Once the wrapper has been created, define the remote server next:
   ```
   CREATE SERVER db2server TYPE DB2/UDB VERSION 10.5
   WRAPPER db2wrapper AUTHORIZATION "db2inst1" PASSWORD
   "ibm2blue" OPTIONS (DBNAME 'DB2DB', COLLATING_SEQUENCE
   'N', PUSHDOWN 'Y')
   ```
   Note: The user id and password should be escaped to preserve the case sensitivity. Use double quotes for this purpose.

Task 5. Mapping and creating a nickname.

1. Once the remote server has been defined, you need to map the local user id and password to the corresponding user id and password on the remote server. While still connected to the bigsql server, run this command:
   ```
   CREATE USER MAPPING FOR bigsql SERVER db2server
   OPTIONS(REMOTE_AUTHID 'db2inst1', REMOTE_PASSWORD
   'ibm2blue')
   ```

2. With the id and password mapped, you can now create the nickname of the remote database object. Type in the command to create a nickname for the remote table within the DB2 data source:

```
CREATE NICKNAME bigsql.order_method FOR
db2server.db2inst1.sls_order_method_dim
```

Once the nickname has been created, you can use it as you would any other database object (local or remote). In the next section, you will run a query that joins tables from both Big SQL and DB2.

Task 6. Querying tables from both Big SQL and DB2.

For this task, it may be easier to use SQL Editor in the Big SQL console. However, you can still use JSqsh if are comfortable with it.

1. You will drop and recreate the bigsql.sls_product_dim table to ensure we have a clean copy to work with.
2. Drop the table:

```
drop table bigsql.sls_product_dim
```

3. Create the table:

```
-- product dimension table
CREATE HADOOP TABLE IF NOT EXISTS sls_product_dim
( product_key      INT NOT NULL
, product_line_code  INT NOT NULL
, product_type_key   INT NOT NULL
, product_type_code  INT NOT NULL
, product_number   INT NOT NULL
, base_product_key   INT NOT NULL
, base_product_number     INT NOT NULL
, product_color_code INT
, product_size_code  INT
, product_brand_key  INT NOT NULL
, product_brand_code INT NOT NULL
, product_image    VARCHAR(60)
, introduction_date  TIMESTAMP
, discontinued_date  TIMESTAMP
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
;
```

4. Load the data:

```
load hadoop using file url
'file:///usr/ibmpacks/bigsql/4.0/bigsql/samples/data/GOS
ALESDW.SLS_PRODUCT_DIM.txt' with SOURCE PROPERTIES
('field.delimiter'='\t') INTO TABLE SLS_PRODUCT_DIM
overwrite;
```

5. Take a moment to look over this query before issuing:

```
SELECT pnumb.product_name, sales.quantity,
meth.order_method_en
FROM
sls_sales_fact sales,
sls_product_dim prod,
sls_product_lookup pnumb,
order_method meth
WHERE
pnumb.product_language='EN'
AND sales.product_key=prod.product_key
AND prod.product_number=pnumb.product_number
AND meth.order_method_key=sales.order_method_key;
```

   The method selects three columns, *product_name*, *quantity*, and *order_method_en*. The third column (*order_method_en*) comes from the DB2 data source. Notice how the *order_method* table appears as a regular table. A user would not be able to tell that it is from a remote data source.

6. Launch the Big SQL console from BigInsights Home via Ambari.
7. Log in to the Big SQL console using **bigsql/ibm2blue**.
8. Inspect the results to see that the tables were joined and the columns were selected.

PRODUCT_NAME	QUANTITY	ORDER_METHOD_EN
Course Pro Putter	587	Telephone
Blue Steel Max Putter	214	Telephone
Course Pro Gloves	576	Telephone
Glacier Deluxe	129	Sales visit
BugShield Natural	1776	Sales visit

9. Go ahead and close any applications and windows that you have opened.

   This concludes the demo. You have successfully set up DB2 as a data source to be used in the same query and joined together with Big SQL tables.

Results:
You setup DB2 as a data source enabling users to send distributed requests to multiple data sources within a single statement. This involved connecting to DB2 LUW, creating a wrapper to use with DB2, created the server as the definition of the remote DB2 database, mapped the user id and password at the federated server to the corresponding user id and password at the data source, created a nickname on a remote object and queried the remote table from the federated Big SQL server.