EXP 5 210701195

PRETI SHERINE P

Create tables in Hive and write queries to access the data in the table

Aim:

To create tables in Hive and write queries to access the data in the table.

Procedure:

Step 1: Download the Hive from it's official website. Once the file get downloaded in your */downloads* folder extract this file with below command

tar -xvf apache-hive-2.1.1-bin.tar.gz

Step 2: Once you have downloaded the Hive now Install the latest version of MySQL java connector.

Then create a link between jar file and hive lib folder and copy jar to the lib folder.

sudo ln -s /usr/share/java/mysql-connector-java.jar $HIVE_HOME/lib/mysql-connector-java.jar

Step 3: Now start the Hadoop with the below command:

start-dfs.sh

start-yarn.sh

Step 4: Once your Hadoop gets started we will create Directories for the hive. Implement below commands in your terminal to make directories.

hdfs dfs -mkdir -p /user/hive/warehouse

hdfs dfs -mkdir -p /tmp/hive

Step 5: Now we will change permission for all this directory with below command.

hdfs dfs -chmod 777 /tmp/

hdfs dfs -chmod 777 /user/hive/warehouse

hdfs dfs -chmod 777 /tmp/hive

Step 6: Now we will install MySQL with below command.

sudo apt-get install mysql-server

Step 7: Create the Metastore Database after entering your MySQL terminal, implement all the below commands to do so (use **root** as password for SQL).

mysql> sudo mysql -u root -p

mysql> CREATE DATABASE metastore_db;

mysql> USE metastore_db;

Change the username according to you and path also if it is different.

mysql> SOURCE /home/{user-name}/Documents/apache-hive-2.1.1-bin/scripts/metastore/upgrade/mysql/hive-schema-0.14.0.mysql.sql;

Step 8: Now make hive user and hive password with below command on *mysql* terminal.

mysql> CREATE USER 'hiveusr'@'%' IDENTIFIED BY 'hivepassword';

mysql> GRANT all on *.* to 'hiveusr'@localhost identified by 'hivepassword';

mysql> flush privileges;

Then type exit to quit the MySQL terminal.

mysql> exit

Step 9: Now go to *apache-hive-2.1.1-bin* then go to *conf* folder and rename hive-default.xml.template to *hive-site.xml* and hive-env.sh.template to *hive-env.sh*

Step 10: Now we start configuration for hive so go to *hive-site.xml* and change the following property.(use clrl+f to search property in a file)

A: ConnectionURL

```
<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://localhost/metastore_db?createDatabaseIfNotExist=true</value>
```

```
501  <property>
502    <!--<name>javax.jdo.option.ConnectionURL</name>
503    <value>jdbc:derby:;databaseName=metastore_db;create=true</value>-->
504    <name>javax.jdo.option.ConnectionURL</name>
505              <value>jdbc:mysql://localhost/metastore_db?createDatabaseIfNotExist=true</value>
506    <description>
507      JDBC connect string for a JDBC metastore.
508      To use SSL to encrypt/authenticate the connection, provide database-specific SSL flag in the connection URL.
509      For example, jdbc:postgresql://myhost/db?ssl=true for postgres database.
510    </description>
511  </property>
```

B: ConnectionUserName

```
<name>javax.jdo.option.ConnectionUserName</name>

<value>hiveusr</value>
```

// Change username in value if you change it above.

```
960    <property>
961      <!--<name>javax.jdo.option.ConnectionUserName</name>
962      <value>APP</value>-->
963      <name>javax.jdo.option.ConnectionUserName</name>
964                  <value>hiveusr</value>
965      <description>Username to use against metastore database</description>
966    </property>
```

C: ConnectionPassword

```
<name>javax.jdo.option.ConnectionPassword</name>
```

```
<value>hivepassword</value>
```

// change password in value if you change it above.

```
<property>
  <!--<name>javax.jdo.option.ConnectionPassword</name>
  <value>mine</value>-->
  <name>javax.jdo.option.ConnectionPassword</name>
              <value>hivepassword</value>
  <description>password to use against metastore database</description>
</property>
```

D: ConnectionDriverName

```
<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

<description>MySQL JDBC driver class</description>
```

```
<property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <!--<value>org.apache.derby.jdbc.EmbeddedDriver</value>-->
  <value>com.mysql.jdbc.Driver</value>
  <!--description>Driver class name for a JDBC metastore</description>-->
            <description>MySQL JDBC driver class</description>
</property>
```

Step 11: Now open *hive-env.sh* and append your hadoop path inside it.

export HADOOP_HOME=/home/dikshant/Documents/hadoop

```

# Folder containing extra ibraries required for hive compilation/execution can be controlled by:
# export HIVE_AUX_JARS_PATH=


export HADOOP_HOME=/home/dikshant/Documents/hadoop
```

Step 12: Also replace the below values in hive-site.xml (search property with ctrl+f and enter the name inside search box)

A: Replace this properties

```
<property>

  <name>hive.exec.local.scratchdir</name>

  <value>${system:java.io.tmpdir}/${system:user.name}</value>

  <description>Local scratch space for Hive jobs</description>

</property>
```

With this property

```
<property>
  <name>hive.exec.local.scratchdir</name>
  <value>/tmp/${user.name}</value>
  <description>Local scratch space for Hive jobs</description>
</property>
```

B: Replace this property

```
<property>
  <name>hive.downloaded.resources.dir</name>
  <value>${system:java.io.tmpdir}/${hive.session.id}_resources</value>
  <description>Temporary local directory for added resources in the remote file system.</description>
</property>
```

With these properties

```
<property>
  <name>hive.downloaded.resources.dir</name>
  <value>/tmp/${user.name}_resources</value>
  <description>Temporary local directory for added resources in the remote file system.</description>
</property>
```

Step 13: Now the most important part is to set path for Hive in our *.bashrc* file, so open *.bashrc* with below command.

sudo gedit ~/.bashrc

Copy the Hive path shown in the below image and update it according to your hive path (if different).

```
#Hive Path
export HIVE_HOME=/home/dikshant/Documents/apache-hive-2.1.1-bin
export PATH=$PATH:$HIVE_HOME/bin
```

source ~/.bashrc

Step 14: Now run this below command to initialize schema for MySQL database.

schematool -initSchema -dbType mysql

Step 15: that's it now run hive shell by typing hive in terminal.

hive

Step 16:

Create a Database
Create a new database in Hive:
hive>CREATE DATABASE financials;

Step 17:

Switch to the newly created database:

hive>use financials;

Step 18:

Create a simple table in your database:

hive>CREATE TABLE finance_table(id INT, name STRING);

Step 19:

You can insert sample data into the table:

hive>INSERT INTO finance_tableVALUES (1, 'Alice'), (2, 'Bob'), (3, 'Charlie');

Step 20:

Use SQL-like queries to retrieve data from your table:

hive>CREATE VIEW myview AS SELECT name, id FROM finance_table;

Step 21:

To see the data in the view, you would
need to query the view
hive>SELECT*FROM myview;

Step 22:

To exit the Hive
CLI, simply type:
hive>quit;

OUTPUT:

```
hadoop@sanjay-VirtualBox:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoop/apache-hive-3.1.2-bin/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBi
nder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/share/hadoop/common/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLogge
rBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = 35fe72d8-3ed1-4428-8590-2c88743dedc7

Logging initialized using configuration in jar:file:/home/hadoop/apache-hive-3.1.2-bin/lib/hive-common-3.1.2.jar!/hive-log4j2.prope
rties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.
e. spark, tez) or using Hive 1.X releases.
Hive Session ID = adccd81f-8176-49ba-bda6-f65011e1f514
hive> show databases;
OK
default
Time taken: 0.484 seconds, Fetched: 1 row(s)
```

Result:

Thus to create tables in Hive and write queries to access the data in the table is executed successfully.