

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 *ctrl+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.

```
484 <property>
485   <!--<name>javax.jdo.option.ConnectionPassword</name>
486   <value>mine</value>-->
487   <name>javax.jdo.option.ConnectionPassword</name>
488   <value>hivepassword</value>
489   <description>password to use against metastore database</description>
490 </property>
```

D: ConnectionDriverName

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

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

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

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

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

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

```
52
53 # Folder containing extra libraries required for hive compilation/execution can be controlled by:
54 # export HIVE_AUX_JARS_PATH=
55
56
57 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:

```
at org.apache.hadoop.util.RunJar.main(RunJar.java:245)
FAILED: ParseException line 1:33 cannot recognize input near '1', 'Alice' in statement
hive> INSERT INTO finance_table VALUES (1, 'Alice'), (2, 'Bob'), (3, 'Charlie');
Query ID = sal_20240910123130_586cfe7b-4d90-4da4-b7f4-f4b6978b595c
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reducers=<number>
Starting Job = job_1725950450473_0001, Tracking URL = http://ubuntu.myquest.virtualbox.org:8088/proxy/application_1725950450473_0001/
Kill Command = /home/sal/hadoop-3.4.0/bin/mapred job -kill job_1725950450473_0001
Hadoop Job Information for Stage-1: number of mappers: 1; number of reducers: 1
2024-09-10 12:31:50,126 Stage-1 map = 0%, reduce = 0%
2024-09-10 12:31:50,524 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.31 sec
2024-09-10 12:32:02,750 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 6.46 sec
MapReduce Total cumulative CPU time: 6 seconds 460 msec
Ended Job = job_1725950450473_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to directory hdfs://localhost:9000/user/hive/warehouse/financials.db/finance_table/.hive-staging_hive_2024-09-10_12-31-36_038_725507672763679201-1/-ext-10000
Loading data to table financials.finance_table
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 6.46 sec HDFS Read: 15656 HDFS Write: 291 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 460 msec
OK
Time taken: 29.561 seconds
hive> CREATE VIEW myview AS SELECT name, id FROM finance_table;
OK
Time taken: 0.356 seconds
hive> SELECT * FROM myview;
OK
Alice      1
Bob        2
Charlie    3
Time taken: 0.224 seconds, Fetched: 3 row(s)
hive> DESCRIBE finance_table;
OK
id          int
name        string
Time taken: 0.103 seconds, Fetched: 2 row(s)
hive> ALTER TABLE finance_table ADD COLUMNS (age INT);
OK
Time taken: 0.149 seconds
hive> hdfs dfs -cat /home/hadoop/pig_output_data/part-n-
sharan@Ubuntu:~$ hdfs dfs -cat /user/hive/warehouse/financials.db/finance_table/
000000_0
1Alice
2Bob
3Charlie
sharan@Ubuntu:~$
```

Result:

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

