HIVE Handson

Prashanth B S[1] , **Karthik**[1], **Akarsha**[1]
[1]Department of Information Science & Engineering, Nitte Meenakshi Institute of Technology, Yelahanka - 560064, Bengaluru

June 22, 2022

1 Working with HIVE

Hive-QL queries are similar to the SQL query statements. Some of the Queries are demonstrated in this section. Start the HADOOP demons and HIVE shell by issuing the following statements,

```
$ cd $HADOOP_HOME/sbin && ./start-all.sh
# Check the deamons of HADOOP
$ hive # Issue the command to start the HIVE shell
```

1.1 Modes, Commands to Create,Use & Delete Database

1.1.1 Modes of Operation

HIVE operates in two modes, They are,

1. **LOCAL Mode**
   - It works only on pseudo distributed mode of Hadoop which is a single Node Hadoop Cluster with limited resource
   - Data size should be small and all operation on the data is Local to the Cluster
   - Query runs faster, since there are no multiple nodes present in the Cluster
2. **Map-Reduce Mode**
   - It works only on Multi Node Hadoop Cluster and operates over large amount of data over the cluster
   - Operates on distributed data in terms of Peta/Zetta bytes
   - Query runs slower as the processing of multi-node takes time

1.1.2 Creating and Using the Database

The queries for creating and using the database are as follows using **hive shell(hive>= $)**,

```
$ show databases ; # Shows all the databases
OK
default
$ create database if not exists studentDB ; # Creating studentDB
$ use studentDB ; # Use the database
$ drop database studentDB ; # To drop Database
```

2 Dealing with the Tables

There are two tables supported by HIVE, they are,

1. **External Table**
   - All data related to the table, stored in the HDFS server. But the tables are not linked fully to the datasource
   - If the table is dropped, metadata of the table gets deleted from the server(Metastore). But the data in the HDFS is retained as it is
   - HIVE does not own the ownership of the data. HIVE does not guarantee the deletion of the data
2. **Internal Table/Managed Table**
   - Tables are fully linked to Datasource
   - Default choice for table in HIVE
   - Security of the data depends on the HIVE user queries
   - If the table is dropped, metadata and data in HDFS will also get removed
   - HIVE owns the data

- The queries for creating the **Internal/Managed** table are as shown below,

```
$ use studentDB;
# Creating a Table with the fields <USN, name, age>
$ create table student(usn int, name string, age int) ;
$ show tables ;# Showing the tables inside studentDB
OK
student
$ desc student ; # Describe the Table
OK
usn     int
name    string
age     int
$ drop table if exists student ; # To drop the table
OK
```

- Data-source for the database can be **.csv, .xlsx, .txt** file for the HIVE database. Here is an example of setting up HIVE table in the format suitable for .csv file.

```
$ use studentDB;
# Create the table suitable for both Normal insert &
# the insert from the .csv file
$ create table student(usn int, name string, age int)
  > row format delimited
  > fields terminated by "," ; # fields are separated by ","
OK
```

- If the data-source for the database is available and is a large-file, then we can use External table for the database. Here is an example for the same,

```
$ use studentDB;
# Create an External table
$ create table studentMentor(mentorid int, mname string, mage int)
  > row format delimited
  > fields terminated by "," # fields are separated by ","
  > stored as textfile; # Specify the format in which the file is stored
OK
```

3 Inserts

Insert operation inserts a record into the database, it can be done in two ways, they are,

1. **Using insert command** : This is similar to the SQL insert command

```
$ insert into student(1, "Rajesh" , 20) ;
# This triggers a mapreduce JOB
OK
Time taken: 45.012 seconds # Observe the Time
$ select * from student ; # To display
OK
1  Rajesh  20
```

2. **Using LOAD function** : This loads the data from the file in a specific format such as .csv, .xlsx etc. This query is way faster than the insert command. But to load the data, we need to create the table in the format of the datasource(CSV). Here is an illustration of the same.

```
# lets say the .CSV file consist of the following fields stored as input.csv
# < Sl.No, CARDname, CustomerName, Amount transacted>
# Stored in /home/user/Desktop/input.csv
# Step 1: Prepare the table in the format of .CSV
$ create table bank(slno int, cname string, custname string, amount int)
  > row format delimited
  > fields terminated by "," ;
# Step 2: Load the data from the csv file
$ load data local inpath '/home/user/Desktop/input.csv' into table bank ;
OK
Time taken: 0.703 seconds # Observe the time
$ select * from bank ;
OK
1  Axis   Prashanth 50000
2  Canara  Karthik 100000
3  SBI   Manoj 2500
4  Axis Ravi  4000
5  ICICI Prashanth 300000
6  SBI   karthik 30000
```

4 Selection, group-by, order-by, & views

4.1 Selection with in-built functions

- Select command is similar to SQL command SELECT which can be combined with inbuilt function such as max, min, sum & avg to retrieve the values of user choice. Here are some examples on the same.

```
# Using the same dataset in/home/user/Desktop/input.csv - bank table
$ select max(amount) from bank ;
OK
300000
$ select avg(amount) from bank ;
OK
81083.33333333333
# Similarly try min(amount), sum(amount)
$ select count(*) from bank ; # Returns number of records
OK
6
```

4.2 Group-by, where & Order-by

- The GROUP BY statement groups rows that have the same values into summary rows. Here is an example for the same.

```
# group by clause : Grouping everything based on the cardname
$ select cardname, sum(amount) from bank group by cardname;
Axis  54000
Canara  100000
ICICI 300000
SBI   32500

# where clause : Selecting all the transaction where amount > 10000
$ select cardname, cuser, amount from bank where amount > 10000 ;
OK
Axis    Prashanth  50000
Canara  Karthik 100000
ICICI Prashanth  300000
SBI   karthik 30000

# where clause: Selecting amount in a range<10000, 300000>
$ select cardname, cuser, amount from bank where amount between 10000 and 300000 ;
Axis  Prashanth 50000
Canara  Karthik 100000
ICICI Prashanth  300000
SBI   karthik 30000

# Ascending Sort by column cardname
$ select * from bank order by cardname;
OK
4  Axis    Ravi   4000
1  Axis    Prashanth 50000
2  Canara   Karthik 100000
5  ICICI Prashanth  300000
```

```
6  SBI   karthik 30000
3  SBI   Manoj 2500

# Descending Sort by column cuser
$ select * from bank order by cuser desc ;
6  SBI   karthik 30000
4  Axis    Ravi   4000
5  ICICI Prashanth  300000
1  Axis  Prashanth  50000
3  SBI   Manoj 2500
2  Canara  Karthik 100000
```

- Views are virtual table used for formatting the output of a query to make it more presentable. They do not have storage space and always dependent on the outer query execution.

```
# Creating a view called AXIS card only
$ create view AXIS as select cardname,cuser,amount from bank where cardname='Axis' ;
OK
# Displaying the contents of the VIEW
$ select * from AXIS;
OK
Axis   Prashanth 50000
Axis   Ravi   4000

# To drop the view
$ drop view AXIS;
OK
```

5 Alter

The HQL ALTER TABLE command is used to add, delete or modify columns in an existing table.

```
# Create a DB called as "Sample" with <name, age> as fields

# To change the table name from Sample -> Student use alter
$ alter table Sample rename to student ;
OK

# To rename a column name say name -> sName
$ alter table student change name sName string ;
$ desc student ;
OK
sname                 string
age                   int

# dropping a column is not possible, but we can replace the columns
# column <sname, age> replaced by <sName, id>
$ alter table student replace columns(sName string, id int) ;
$ desc student; # Columns are replaced no
OK
sname                 string
id                    int
```

```
# We can add a new column to the existing table
$ alter table student add columns(age int) ;
$ desc student ; # age column is added
OK
sname                   string
id                      int
age                     int
```

6 JOINS-Inner, Outer Joins

HIVE does supports Joins. By default HIVE provides **Full Outer JOIN**. The other JOINS supported by HIVE are,

1. FULL OUTER JOIN
2. RIGHT OUTER JOIN
3. LEFT OUTER JOIN
4. No inner joins are supported by the HIVE

The joins are summarized in the below diagram 1 for reference. The following segment gives hands

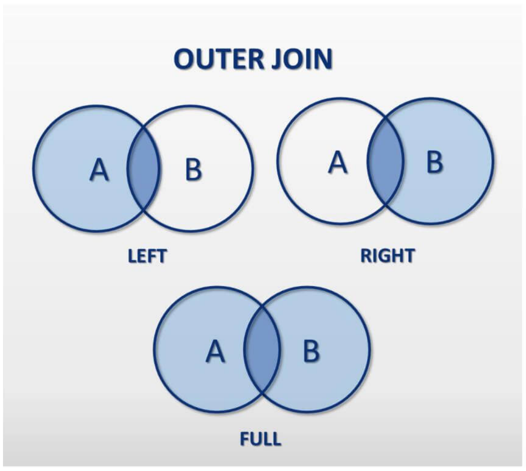

Figure 1: JOINS in HIVE

on session for JOINS,

```
# Prepare dataset customer<id, name, age, salary> and
# order<orderid, custid(refers to id), amount> as cust.csv and order.csv
# Create customer table under SalesDB matching the fields in cust.csv
hive> create table customer(id int, name string, age int, salary int)
    > row format delimited
    > fields terminated by "," ;

# Create orders table under SalesDB matching the fields in order.csv
hive> create table orders(oid string, cid int, amount int)
    > row format delimited
    > fields terminated by "," ;

# Insert values from the CSV file as follows,
hive> load data local inpath '/home/t1da1wav3/Desktop/cust.csv' into table customer ;
hive> load data local inpath '/home/t1da1wav3/Desktop/order.csv' into table orders;

# Performing the Default Outer Join on customer ID , selecting name,age
hive> select c.name, c.age from customer c join
    > orders o on(c.id = o.cid) ;
OK
Rajesh  33
Rajesh  33
Prithvi 32
Prithvi 32
Naveen  31
Rithik  33
Prashanth  30

# Performing the Left Outer Join on customer ID , selecting name,age
hive> select c.name, c.age, o.amount from customer c left outer join
    > orders o on(c.id = o.cid) ;
OK
Prashanth  30 NULL
Rajesh  33 20000
Rajesh  33 30000
Prithvi 32 20000
Prithvi 32 30000
Naveen  31 20000
Rithik  33 20000
Prashanth  30 30000
Naveen  31 NULL

# Performing the Right Outer Join on customer ID , selecting name,age
hive> select c.name, c.age, o.amount from customer c right outer join
    > orders o on(c.id = o.cid) ;
Total MapReduce CPU Time Spent: 2 seconds 720 msec
OK
Prithvi 32 20000
Prithvi 32 30000
Rajesh  33 20000
Rajesh  33 30000
Rithik  33 20000
Prashanth  30 30000
Naveen  31 20000

# Performing the Full Outer Join on customer ID , selecting name,age
hive> select c.name, c.age, o.amount from customer c full outer join
    > orders o on(c.id = o.cid) ;
OK
Prashanth  30 NULL
Rajesh  33 30000
Rajesh  33 20000
Prithvi 32 30000
Prithvi 32 20000
Naveen  31 20000
Rithik  33 20000
Prashanth  30 30000
Naveen  31 NULL
```