Lab: Using Hive

|  |  |
| --- | --- |
| **Objective:** | Understand how Hive table data is stored in HDFS. |
| **Successful Outcome:** | You will have created a couple of tables in Hive and learned how data gets associated with a Hive table. |
| **Before you begin:** | Go into terminal on the sandbox. |

1. Review the Data
   1. As root, change directories to the /root/materials/data folder.
   2. Notice there are part-m-0000x files, which are the result of a MapReduce job that formatted the data for use with Hive. View the contents of one of these files:

# more part-m-00000

Notice the data consists of information about visitors to the White House, including the name, date, person being visited, and a comment section.

1. Define a Hive Table
   1. In the data folder, there is a text file named wh\_visits.sql. View its contents. Notice it defines a Hive table named wh\_visits with a schema that matches the data in the part-m-0000x files:

# more wh\_visits.sql

create table wh\_visits (

lname string,

fname string,

time\_of\_arrival string,

appt\_scheduled\_time string,

meeting\_location string,

info\_comment string)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\t' ;

**NOTE**: You cannot use comment or location as column names because those are reserved Hive keywords.

* 1. Run the script with the following command:

# hive -f wh\_visits.sql

* 1. If successful, you should see “OK” in the output along with the time it took to run the query.

1. Verify the Table Creation
   1. Go into <http://localhost:8000/beeswax/tables> to view file you created
   2. Start the Hive Shell:

# hive

hive>

* 1. From the hive> prompt, enter the “show tables” command:

hive> show tables;

You should see wh\_visits in the list of tables.

* 1. Use the describe command to view the details of wh\_visits:

hive> describe wh\_visits;

OK

lname string None

fname string None

time\_of\_arrival string None

appt\_scheduled\_time string None

meeting\_location string None

info\_comment string None

* 1. Try running a query (even though the table is empty):

select \* from wh\_visits;

The query should execute fine, but no result appears.

1. View the Hive Folder Structure
   1. Exit the Hive shell:

hive> exit;

[root@sandbox data]#

* 1. View the contents of the Hive warehouse folder:

# hadoop fs -ls /apps/hive/warehouse

Found 1 items

drwxr-xr-x - root hdfs 0 /apps/hive/warehouse/wh\_visits

Notice there is a folder named wh\_visits. When did this folder get created? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. List the contents of the wh\_visits folder:

# hadoop fs -ls /apps/hive/warehouse/wh\_visits

Notice the folder is empty.

1. Populate the Hive Table
   1. Run the following command to put the local part-m-00000 file into the wh\_visits folder:

# hadoop fs -put part-m-00000 /apps/hive/warehouse/wh\_visits

* 1. From the Hive shell, run the following query:

hive> select \* from wh\_visits;

This time, you should see a couple thousand rows of data. Notice that by simply putting a file into the wh\_visits folder, the table now contains data.

* 1. Notice no MapReduce job was executed to perform the select \* query. Why not? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Drop the Table
   1. Run the following query, which drops the wh\_visits table:

hive> drop table wh\_visits;

* 1. Exit the Hive shell and view the contents of the Hive warehouse folder:

# hadoop fs -ls /apps/hive/warehouse/

Notice that not only has the part-m-00000 file been deleted, but also the wh\_visits folder no longer exists!

1. Create the Table Again
   1. Run wh\_visits.hive again to recreate the wh\_visits table:

# hive -f wh\_visits.hive

1. Use the Hive LOAD DATA Command
   1. Create a new directory in HDFS named whitehouse:

# hadoop fs -mkdir whitehouse

* 1. Put all 5 part-m files into whitehouse:

# hadoop fs -put part-m-\* whitehouse/

* 1. Verify the files are there:

# hadoop fs -ls whitehouse

* 1. From the Hive shell, run the following query:

hive> LOAD DATA INPATH '/user/root/whitehouse/' OVERWRITE INTO TABLE wh\_visits;

* 1. Verify you have now have data in the table:

hive> select \* from wh\_visits limit 10;

You should ten rows of visitors, and no MapReduce is needed.

* 1. Try the following query. Make sure the output looks like first names:

hive> select fname from wh\_visits limit 20;

This time a MapReduce job executed. Why? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. View the Folder Structure
   1. View the contents of the wh\_visits folder:

# hadoop fs -ls /apps/hive/warehouse/wh\_visits

Notice the five part-m files are located in wh\_visits.

* 1. Try viewing the contents of the whitehouse folder:

# hadoop fs -ls whitehouse

Notice the folder is empty. The LOAD DATA command *moved* the files from their original HDFS folder into the Hive warehouse folder; it did not copy them.

**IMPORTANT**: Be careful when you drop a managed table in Hive. Make sure you either have the data backed up somewhere else, or that you no longer want the data.

1. Count the Number of Rows in a Table
   1. Enter the following Hive query, which outputs the number of rows in wh\_visits:

hive> select count(\*) from wh\_visits;

* 1. How many rows are currently in wh\_visits? \_\_\_\_\_\_\_\_\_\_\_\_\_

1. Define an External Table
   1. Drop the wh\_visits table again:

hive> drop table wh\_visits;

* 1. View the contents of external\_table.hive in the /root/labs/data folder:

# more external\_table.hive

create external table wh\_visits (

lname string,

fname string,

time\_of\_arrival string,

appt\_scheduled\_time string,

meeting\_location string,

info\_comment string)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\t'

LOCATION '/user/root/whitehouse/' ;

* 1. Create the whitehouse folder in HDFS again, and put the five part-m files into whitehouse.
  2. Verify that there is not a subfolder of /apps/hive/warehouse named wh\_visits.
  3. Run the query in **external\_table.hive** to create the **wh\_visits** table:

# hive –f external\_table.hive

* 1. Run a query on wh\_visits to verify that the table does actually contain records.
  2. Drop wh\_visits again, but this time notice that the files in the whitehouse folder are not deleted.

**RESULT**: As you just verified, the data for external tables is not deleted when the corresponding table is dropped. Aside from this behavior, managed tables and external tables in Hive are essentially the same.