

CISC 525-90-2016 - Unit 7

Hive Usage Report

Nelson Corrocher

August 2016

1 Overview

This exercise is going to show some basic commands used in Hive and how it work. As a reminder, Hive is an interface to MapReduce so a similar output is expected. This exercise includes preparing a basic database in which tables can be created and data loaded. Then some simple select queries are going to be executed and its output exhibited.

The datasets used comes from a certain company. One dataset contains project information while the second table contains employee information. The two tables are joined by the Project Manager ID from the first table to its employee number on the second.

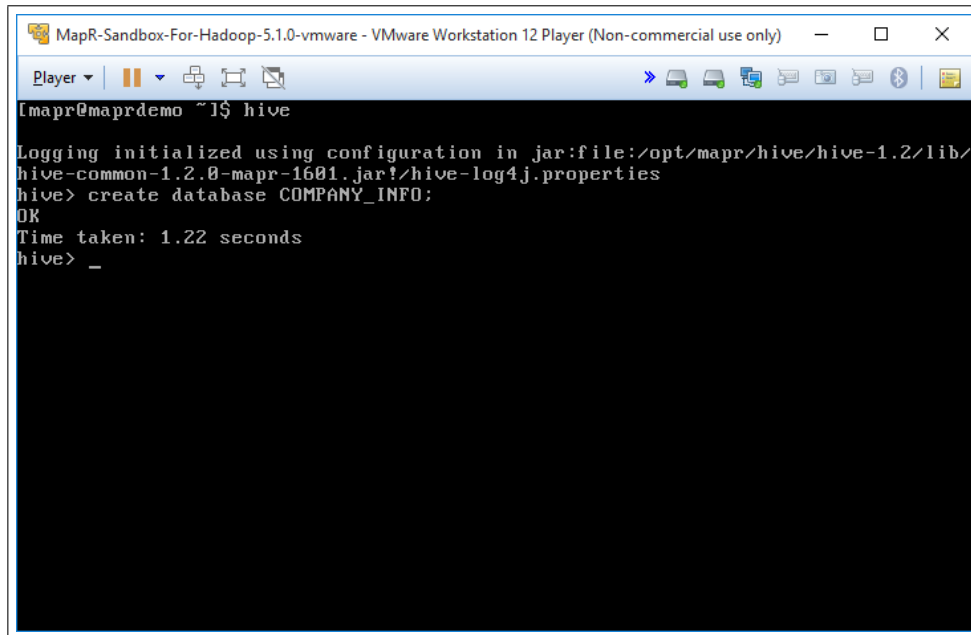
One important note regarding the use of Hive is that its command interface is very frustrating. Instead of using it directly, most Hive SQL code was inputted in a text file and executed through **Hive -f 'filename'** command.

2 Creating the database

A very simple database was created using the following command:

```
CREATE DATABASE COMPANY.INFO;
```

Note, this command was used directly in the console due to its simplicity.



```
[mapr@maprdemo ~]$ hive
Logging initialized using configuration in jar:file:/opt/mapr/hive/hive-1.2/lib/
hive-common-1.2.0-mapr-1601.jar!/hive-log4j.properties
hive> create database COMPANY_INFO;
OK
Time taken: 1.22 seconds
hive> _
```

Figure 1: Database Creation Output

3 Creating the tables

The queries themselves contain the data definition.

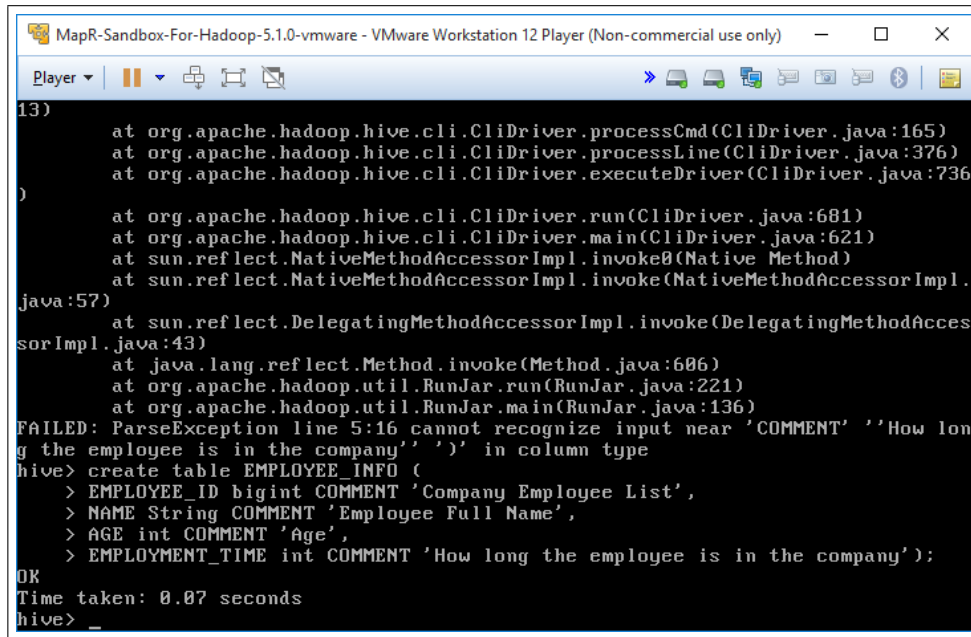
Table for Project Information:

```
CREATE TABLE PROJECT_INFO (
    PROJECT_NUMBER bigint COMMENT 'Project Primary Key',
    PM_ID bigint COMMENT 'Foreign Key in EMPLOYEE_INFO table',
    YTD_REVENUE double COMMENT 'Current Project Revenues')
ROW FORMAT DELIMITED
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n';
```

Table for Employee Information:

```
CREATE TABLE EMPLOYEE_INFO (
    EMPLOYEE_NUMBER bigint COMMENT 'Employee Primary Key',
    FULL_NAME string COMMENT 'Employee Name',
    AGE int COMMENT 'in years',
    EMPLOYMENT_TIME int COMMENT 'in years')
ROW FORMAT DELIMITED
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n';
```

Note: If the 'FIELDS' and 'LINES' statements are omitted, the loading data step fails and fills the table with only NULLs. This was not extensively tested, but is probably because the datasets were exported to CSVs using Excel, and thus, having DOS text format.



```
13)
    at org.apache.hadoop.hive.cli.CliDriver.processCmd(CliDriver.java:165)
    at org.apache.hadoop.hive.cli.CliDriver.processLine(CliDriver.java:376)
    at org.apache.hadoop.hive.cli.CliDriver.executeDriver(CliDriver.java:736)
)
    at org.apache.hadoop.hive.cli.CliDriver.run(CliDriver.java:681)
    at org.apache.hadoop.hive.cli.CliDriver.main(CliDriver.java:621)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccess
sorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:606)
    at org.apache.hadoop.util.RunJar.run(RunJar.java:221)
    at org.apache.hadoop.util.RunJar.main(RunJar.java:136)
FAILED: ParseException line 5:16 cannot recognize input near 'COMMENT' 'How lon
g the employee is in the company'' ' in column type
hive> create table EMPLOYEE_INFO (
  > EMPLOYEE_ID bigint COMMENT 'Company Employee List',
  > NAME String COMMENT 'Employee Full Name',
  > AGE int COMMENT 'Age',
  > EMPLOYMENT_TIME int COMMENT 'How long the employee is in the company');
OK
Time taken: 0.07 seconds
hive> _
```

Figure 2: Table creation output example

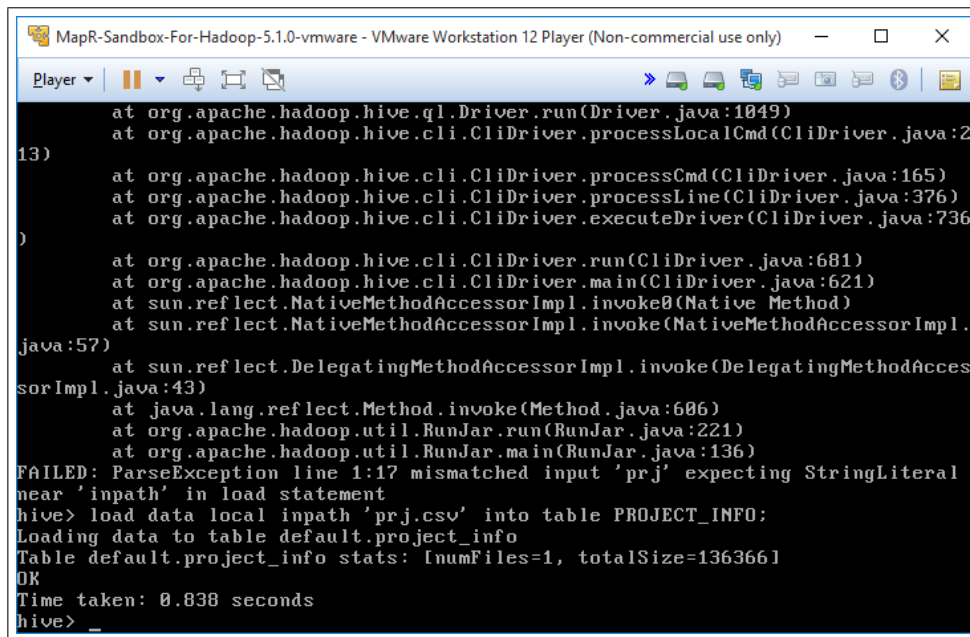
4 Loading the data

Loading data into PROJECT_INFO table:

```
LOAD DATA LOCAL INPATH '/home/mapr/prj.csv' INTO TABLE PROJECT_INFO;
```

Loading data into EMPLOYEE_INFO table:

```
LOAD DATA LOCAL INPATH '/home/mapr/hr.csv' INTO TABLE EMPLOYEE_INFO;
```



```
at org.apache.hadoop.hive.gl.Driver.run(Driver.java:1049)
at org.apache.hadoop.hive.cli.CliDriver.processLocalCmd(CliDriver.java:2
13)
at org.apache.hadoop.hive.cli.CliDriver.processCmd(CliDriver.java:165)
at org.apache.hadoop.hive.cli.CliDriver.processLine(CliDriver.java:376)
at org.apache.hadoop.hive.cli.CliDriver.executeDriver(CliDriver.java:736
)
at org.apache.hadoop.hive.cli.CliDriver.run(CliDriver.java:681)
at org.apache.hadoop.hive.cli.CliDriver.main(CliDriver.java:621)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.
java:57)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAcces
sorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:606)
at org.apache.hadoop.util.RunJar.run(RunJar.java:221)
at org.apache.hadoop.util.RunJar.main(RunJar.java:136)
FAILED: ParseException line 1:17 mismatched input 'prj' expecting StringLiteral
near 'inpath' in load statement
hive> load data local inpath 'prj.csv' into table PROJECT_INFO;
Loading data to table default.project_info
Table default.project_info stats: [numFiles=1, totalSize=1363661]
OK
Time taken: 0.038 seconds
hive> _
```

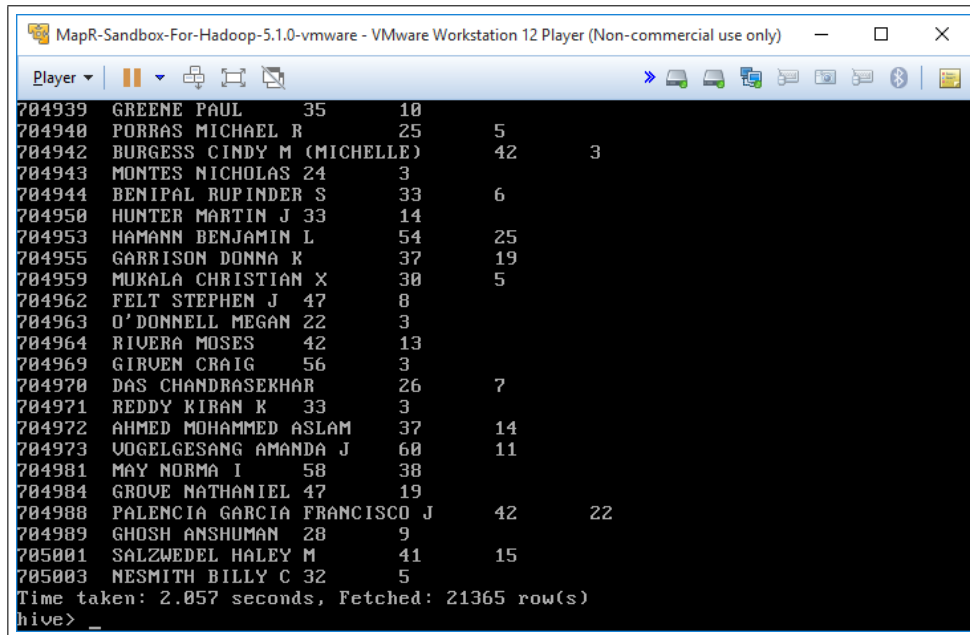
Figure 3: Loading data example

5 Running select queries for analysis

The select queries executed in Hive are compiled in MapReduce and then executed. This can be seen in the command prompt output. The first thing to be noticed is that all but basic select * from table query take a lot of time to run in comparison to regular RDBS.

The first query executed was a simple test to see if the data was correctly imported. This SELECT statements was relatively fast and executed in 2.88 seconds. As mentioned in class, this happens because Hive directly extracts the data from table in these very simple queries.

```
SELECT * FROM EMPLOYEE.INFO;
```



```
704939 GREENE PAUL 35 10
704940 PORRAS MICHAEL R 25 5
704942 BURGESS CINDY M (MICHELLE) 42 3
704943 MONTES NICHOLAS 24 3
704944 BENIPAL RUPINDER S 33 6
704950 HUNTER MARTIN J 33 14
704953 HAMANN BENJAMIN L 54 25
704955 GARRISON DONNA K 37 19
704959 MUKALA CHRISTIAN X 30 5
704962 FELT STEPHEN J 47 8
704963 O'DONNELL MEGAN 22 3
704964 RIVERA MOSES 42 13
704969 GIRVEN CRAIG 56 3
704970 DAS CHANDRASEKHAR 26 7
704971 REDDY KIRAN K 33 3
704972 AHMED MOHAMMED ASLAM 37 14
704973 VOGELGESANG AMANDA J 60 11
704981 MAY NORMA I 58 38
704984 GROVE NATHANIEL 47 19
704988 PALENCIA GARCIA FRANCISCO J 42 22
704989 GHOSH ANSHUMAN 28 9
705001 SALZWEDEL HALEY M 41 15
705003 NESMITH BILLY C 32 5
Time taken: 2.057 seconds, Fetched: 21365 row(s)
hive>
```

Figure 4: Running a simple query

The next query was just a `SELECT SUM` to test the results. One can notice that the execution time increased by a lot, due to Hive converting the query to MapReduce code and then running it in Hadoop.

```
SELECT SUM(YTD.REVENUE) FROM PROJECT INFO;
```

```

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.reduces=<number>
Starting Job = job_1471208667639_0002, Tracking URL = http://maprdemo:8088/proxy
/application_1471208667639_0002/
Kill Command = /opt/mapr/hadoop/hadoop-2.7.0/bin/hadoop job -kill job_147120866
7639_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2016-08-14 14:24:32,253 Stage-1 map = 0%, reduce = 0%
2016-08-14 14:24:39,940 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.91 se
c
2016-08-14 14:24:48,637 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 3.32
sec
MapReduce Total cumulative CPU time: 3 seconds 320 msec
Ended Job = job_1471208667639_0002
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 3.32 sec MAPRFS Read: 0 MAP
RFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 320 msec
OK
3.3078380317999995E8
Time taken: 34.399 seconds, Fetched: 1 row(s)
[mapr@maprdemo ~]$ _

```

Figure 5: Running an aggregation query

The final test was to execute a JOIN query on the two tables. The time taken was not much higher than the SUM Query. This could imply that the overhead from Hadoop MapReduce framework is what really takes a lot of time in these queries.

```

SELECT
    b.EMPLOYEENUMBER,
    b.FULL_NAME,
    sum(a.YTD.REVENUE)
FROM
    PROJECT_INFO a JOIN EMPLOYEE_INFO b
    ON (a.PM.ID = b.EMPLOYEENUMBER)
WHERE b.EMPLOYMENT.TIME < 10
GROUP BY
    b.EMPLOYEENUMBER,
    b.FULLNAME;

```

```
701364 THOMPSON ROBIN 262227.9
701487 PARENTE WILLIAM J 1104981.85
701552 SMITH CHRISTOPHER R 787076.03
701735 POWLEY KEITH L 242682.43
701758 MASTER SAMEET 355156.85
701786 LESTER MARK C 372446.3
701819 ACKERMAN STEVEN 978002.1100000002
701839 JENNINGS MICHAEL 940384.37
702290 RODRIGUEZ REBECCA 127420.12999999999
702438 PLOCH ERIC 1177910.9299999997
702467 MELDRUM COLIN B 52229.48
702522 BYRNE ESCRIBANO ALYSON M 331647.57999999996
702559 BURNS CHRISTOPHER L 461406.3
702664 AFIFY MOHSEN E 544094.0
702923 LYONS JENNIFER W 503917.9800000001
703006 GUSIC DONALD 3070106.5
703081 PAL SHAUMIK 39654.03
703180 UEACH JESSICA L 150394.34999999998
703204 HOLM THOMAS F 214912.08
703669 CURRY WAYNE C 230825.12
703768 GRANTHAM GEORGE L 801706.89
704169 SCHMIDT RODGER P 77483.73
704449 YOST SCOTT ALAN 426251.0
Time taken: 37.626 seconds, Fetched: 519 row(s)
lmapr@maprdemo ~]$_
```

Figure 6: Running a join query

6 Conclusion

The command line interface for Hive is bad and very difficult to use for everything but the most simple queries. A better approach is to write the code in a text file and execute it through **Hive -f 'filename'**. The feel of Hive SQL is very similar to SQL, although it would be interesting to see if and how nested queries would work. Finally, the overhead of MapReduce framework is significant in comparison to regular RDBS. The real benefits of using Hive couldn't be tested due the environment having only one node and only test data.

References

Only the slides from class.

Total words (excluding title and references): 797 words