Hive – Hbase

Tushar B. Kute,
http://tusharkute.com

Pre-requisites

- Ubuntu 14.04 LTS
- Hadoop 2.6.*
- Java with jdk1.7 onwards

Download and Copy

- Download hive on below path (nearly 93 MB): http://www.apache.org/dyn/closer.cgi/hive/

- Extract the .tar.gz file in Downloads/ and rename it to hive/ and move the folder to **`/usr/lib/`** path:

```
sudo mv Downloads/hive /usr/lib
```

Change the owner

- Provide access to hive path by changing the owners and groups to hduser and hadoop respectively.


```
sudo chown -R hduser:hadoop /usr/lib/hive
```

Configure environment variables

- Configure environment variables in .bashrc file.

```
su - hduser
vim ~/.bashrc
```

- Add following lines at the end of file

```
export HIVE_HOME=/usr/lib/hive/
export PATH=$PATH:$HIVE_HOME/bin
export HADOOP_USER_CLASSPATH_FIRST=true
```

- Apply the changes:

```
source ~/.bashrc
```

tusharkute.com

Make directories

- Create temporary and folder for data warehouse of hive in HDFS as well as change the permissions.

```
hadoop fs -mkdir /tmp
hadoop fs -mkdir -p /user/hive/warehouse
hadoop fs -chmod g+w /tmp
hadoop fs -chmod -R g+w /user/hive/warehouse
```

Configure Hive

- To configure Hive with Hadoop, you need to edit the hive-env.sh file, which is placed in the $HIVE_HOME/conf directory. The following commands redirect to Hive config folder and copy the template file:

```
cd $HIVE_HOME/conf
cp hive-env.sh.template hive-env.sh
```

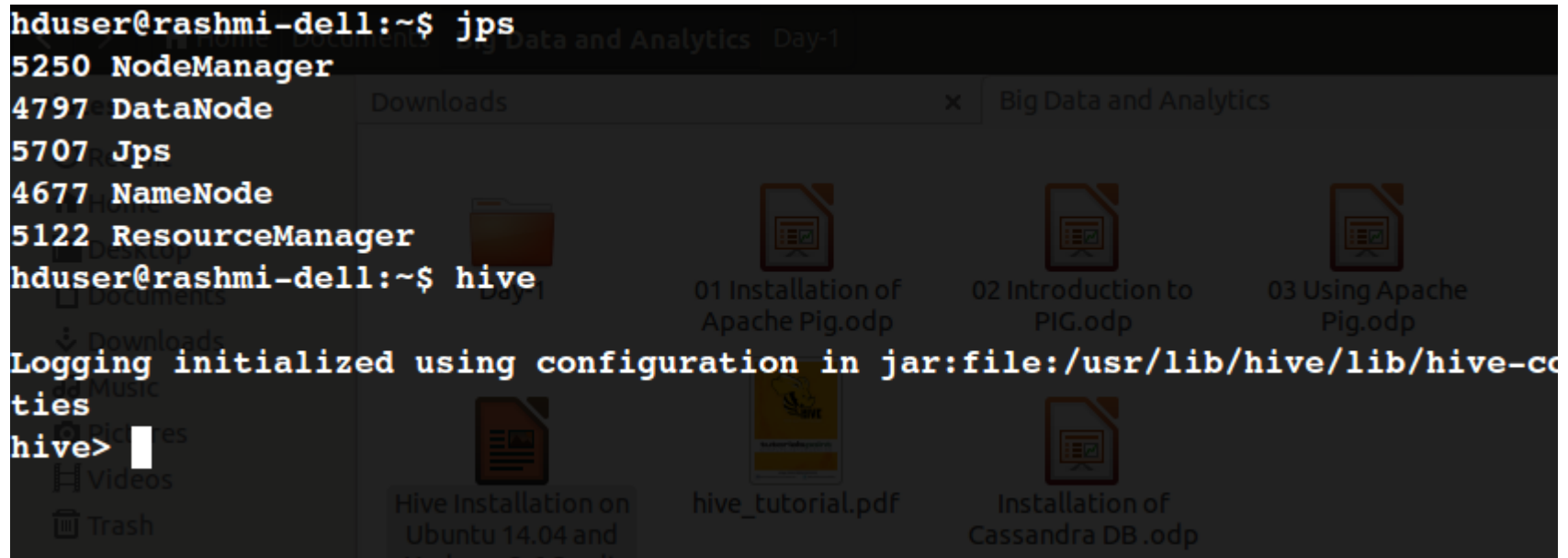- Edit the hive-env.sh file by appending the following line:

```
export HADOOP_HOME=/usr/local/hadoop
```

Run the Hive

- Make sure that Hadoop services are running. Then type-

`hive`

Hadoop Ecosystem

- The Hadoop ecosystem contains different sub-projects (tools) such as Sqoop, Pig, and Hive that are used to help Hadoop modules.

  - **Sqoop**: It is used to import and export data to and fro between HDFS and RDBMS.

  - **Pig**: It is a procedural language platform used to develop a script for MapReduce operations.

  - **Hive**: It is a platform used to develop SQL type scripts to do MapReduce operations.

Database Operations

Hive is a database technology that can define databases and tables to analyze structured data. The theme for structured data analysis is to store the data in a tabular manner, and pass queries to analyze it. This chapter explains how to create Hive database. Hive contains a default database named **default**.

Create Database

- Create Database is a statement used to create a database in Hive.

- A database in Hive is a namespace or a collection of tables. The syntax for this statement is as follows:

```
CREATE DATABASE|SCHEMA [IF NOT EXISTS]
<database name>;
```

Here, IF NOT EXISTS is an optional clause, which notifies the user that a database with the same name already exists. We can use SCHEMA in place of DATABASE in this command.

Create Database

- The following query is executed to create a database named mydb:

  hive> `CREATE DATABASE [IF NOT EXISTS] mydb;`

  or

  hive> `CREATE SCHEMA mydb;`

- The following query is used to verify a databases list:

  hive> `SHOW DATABASES;`

  `default`

  `mydb`

tusharkute.com

Drop Database

- Drop Database is a statement that drops all the tables and deletes the database.

  - Its syntax is as follows:

    ```
    DROP DATABASE StatementDROP
    (DATABASE|SCHEMA) [IF EXISTS]
    database_name [RESTRICT|CASCADE];
    ```

- The following queries are used to drop a database. Let us assume that the database name is mydb.

    ```
    hive> DROP DATABASE IF EXISTS mydb;
    ```

Drop Database

- The following query drops the database using CASCADE. It means dropping respective tables before dropping the database.

  hive> `DROP DATABASE IF EXISTS userdb CASCADE;`

- The following query drops the database using SCHEMA.

  hive> `DROP SCHEMA userdb;`

- This clause was added in Hive 0.6.

Create Table

- Create Table is a statement used to create a table in Hive. The syntax and example are as follows:

- Syntax:

```
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF
NOT EXISTS] [db_name.] table_name

[(col_name data_type [COMMENT
col_comment], ...)]

[COMMENT table_comment]

[ROW FORMAT row_format]

[STORED AS file_format]
```

Create Table : Example

Sr. No.	Field Name	Data type
1	Eid	Int
2	Name	String
3	Salary	Float
4	Designation	String

tusharkute.com

Create Table : Example

- The following query creates a table named employee using the above data.

```
hive> CREATE TABLE IF NOT EXISTS
employee ( eid int, name String,
> salary String, destination String)
> COMMENT 'Employee details'
> ROW FORMAT DELIMITED
> FIELDS TERMINATED BY '\t'
> LINES TERMINATED BY '\n'
> STORED AS TEXTFILE;
```

Alter Table

ALTER TABLE name RENAME TO new_name

ALTER TABLE name ADD COLUMNS (col_spec[, col_spec ...])

ALTER TABLE name DROP [COLUMN] column_name

ALTER TABLE name CHANGE column_name new_name new_type

ALTER TABLE name REPLACE COLUMNS (col_spec[, col_spec ...])

ALTER TABLE employee RENAME TO emp;

The following table contains the fields of **employee** table and it shows the fields to be changed (in bold).

Field Name	Convert from Data Type	Change Field Name	Convert to Data Type
eid	int	eid	int
name	String	**ename**	String
salary	**Float**	salary	**Double**
designation	String	designation	String

Change statement example

- **hive> ALTER TABLE employee CHANGE name ename String;**

- **hive> ALTER TABLE employee CHANGE salary salary Double;**

Add column statement

- **hive>** `ALTER TABLE employee ADD COLUMNS (`

  `> dept STRING COMMENT 'Department name');`

Replace statement

```
hive> ALTER TABLE employee REPLACE COLUMNS
(
> eid INT empid Int,
> ename STRING name String);
```

Drop table statement

- The syntax is as follows:
  - **`DROP TABLE [IF EXISTS] table_name;`**

- The following query drops a table named employee:
  - **`hive> DROP TABLE IF EXISTS employee;`**

Index

- An Index is nothing but a pointer on a particular column of a table.

- Creating an index means creating a pointer on a particular column of a table.

-

- hive> CREATE INDEX index_yoj ON TABLE file(yoj)

  > AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler' WITH DEFERRED REBUILD;

Index – Example

```
hive> CREATE INDEX in_salary ON TABLE file(yoj)
    > AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler' WITH DEFERRED REBUILD;
OK
Time taken: 0.485 seconds
hive> show tables;
OK
class
emp_30000
file
file1
file_2010
tushar__file_in_salary__
tushar__file_index_salary__
Time taken: 0.019 seconds, Fetched: 7 row(s)
hive> drop index tushar__file_in_salary__ on file;
OK
Time taken: 0.027 seconds
hive>
```

Drop index

- The following syntax is used to drop an index:

  ```
  DROP INDEX <index_name> ON <table_name>
  ```

- The following query drops an index named index_salary:

  ```
  hive> DROP INDEX index_salary ON employee;
  ```

Select ... order by

- The ORDER BY clause is used to retrieve the details based on one column and sort the result set by ascending or descending order.

- Syntax:

  SELECT [ALL | DISTINCT] select_expr, select_expr, ...

  FROM table_reference

  [WHERE where_condition]

  [GROUP BY col_list]

  [HAVING having_condition]

  [ORDER BY col_list]]

  [LIMIT number];

Select … order by- Example

```
hive> select * from file order by yoj;
Query ID = hduser_20160703164810_7d84d930-f1dd-4ed3-9410-1f09af20a74d
Total jobs = 1
Launching Job 1 out of 1
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.reduces=<number>
Job running in-process (local Hadoop)
2016-07-03 16:48:13,401 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local590275424_0005
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 6000 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
104      Parmeet CS      2010
102      Rajesh  IT      2010
103      Awez    CS      2012
103      Suresh  CS      2012
Time taken: 2.462 seconds, Fetched: 4 row(s)
```

tusharkute
.com

Select... group by

- The GROUP BY clause is used to group all the records in a result set using a particular collection column. It is used to query a group of records.

- Syntax:

    SELECT [ALL | DISTINCT] select_expr, select_expr, ...

    FROM table_reference

    [WHERE where_condition]

    [GROUP BY col_list]

    [HAVING having_condition]

    [ORDER BY col_list]]

    [LIMIT number];

Select… group by – example

```
hive> select dept, count(*) from file group by dept;
Query ID = hduser_20160703165351_da8962c1-3407-49bd-bd57-c463d2aab7ff
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 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.reduces=<number>
Job running in-process (local Hadoop)
2016-07-03 16:53:53,780 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1959421652_0007
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 6300 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
CS      3
IT      1
Time taken: 1.86 seconds, Fetched: 2 row(s)
```

Joins

- JOINS is a clause that is used for combining specific fields from two tables by using values common to each one.

- It is used to combine records from two or more tables in the database.

- It is more or less similar to SQL JOINS.
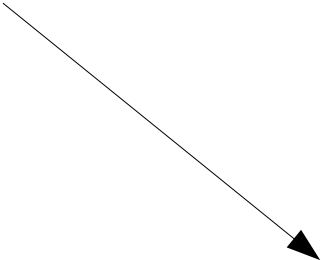
Joins – Examples

```
hive> select * from customer;
OK
1       Kavita  24      Sangvi  34000
2       Chatur  23      Kothrud 35000
3       Fatema  31      Lohgad  20000
4       Rohan   27      Pune Station    22000
Time taken: 0.061 seconds, Fetched: 4 row(s)
```

```
hive> select * from orders;
OK
102     NULL    3       1200
104     NULL    3       3400
105     NULL    4       2150
106     NULL    2       3420
Time taken: 0.057 seconds, Fetched: 4 row(s)
```

Joins – Examples

```
hive> SELECT c.ID, c.NAME, c.AGE, o.AMOUNT
    > FROM CUSTOMER c JOIN ORDERS o
    > ON (c.ID = o.c_id);
Query ID = hduser_20160703175303_ac7c2fcc-c9f2-
Total jobs = 1
```

```
Total MapReduce CPU Time Spent: 0 msec
OK
2        Chatur    23        3420
3        Fatema    31        1200
3        Fatema    31        3400
4        Rohan     27        2150
Time taken: 9.21 seconds, Fetched: 4 row(s)
```
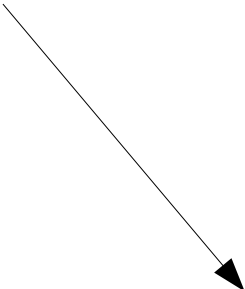
- The HiveQL LEFT OUTER JOIN returns all the rows from the left table, even if there are no matches in the right table.

- This means, if the ON clause matches 0 (zero) records in the right table, the JOIN still returns a row in the result, but with NULL in each column from the right table.

- A LEFT JOIN returns all the values from the left table, plus the matched values from the right table, or NULL in case of no matching JOIN predicate.

Left outer join

```
hive> select c.ID, c.NAME, o.AMOUNT
    > FROM CUSTOMER c
    > LEFT OUTER JOIN ORDERS o
    > ON (c.ID = o.C_ID);
```
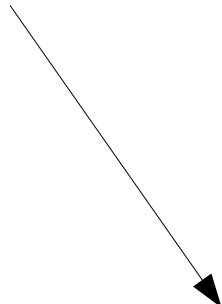
```
MapReduce Jobs Launched:
Stage-Stage-3:  HDFS Read: 106 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
1       Kavita  NULL
2       Chatur  3420
3       Fatema  1200
3       Fatema  3400
4       Rohan   2150
Time taken: 11.194 seconds, Fetched: 5 row(s)
```

tusharkute
.com

Right outer join

- The HiveQL RIGHT OUTER JOIN returns all the rows from the right table, even if there are no matches in the left table.

- If the ON clause matches 0 (zero) records in the left table, the JOIN still returns a row in the result, but with NULL in each column from the left table.

- A RIGHT JOIN returns all the values from the right table, plus the matched values from the left table, or NULL in case of no matching join predicate.
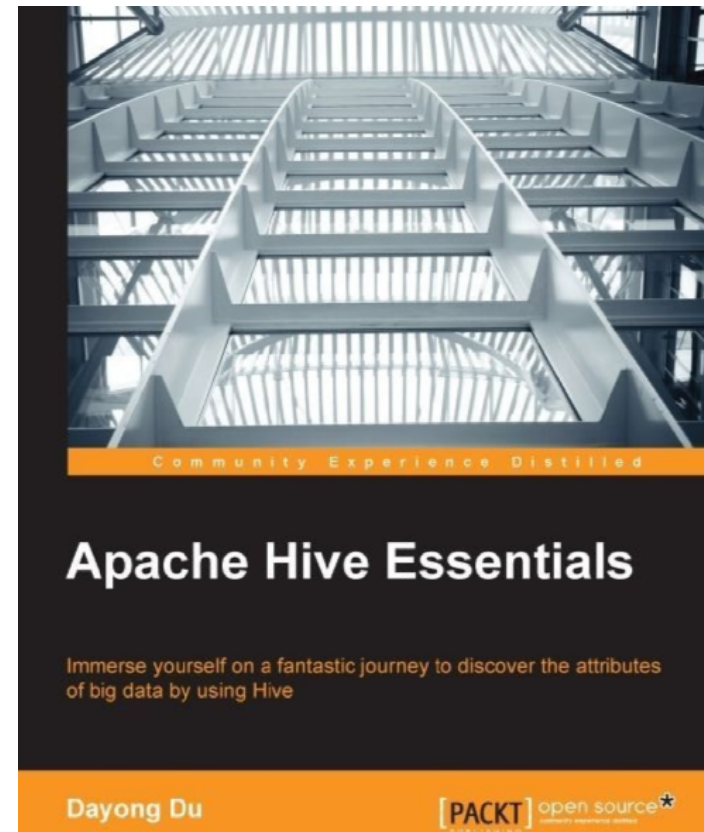
Right outer join – Example

```
hive> select c.ID, c.NAME, o.AMOUNT
    > FROM CUSTOMER c
    > RIGHT OUTER JOIN ORDERS o
    > ON (c.ID = o.C_ID);
```

```
Stage-Stage-3:  HDFS Read: 162 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
3        Fatema   1200
3        Fatema   3400
4        Rohan    2150
2        Chatur   3420
Time taken: 18.488 seconds, Fetched: 4 row(s)
```

References

Data Warehouse and Query Language for Hadoop

Programming **Hive**

Jason Rutherglen,
Dean Wampler &
Edward Capriolo

O'REILLY®

Community Experience Distilled

Apache Hive Essentials

Immerse yourself on a fantastic journey to discover the attributes of big data by using Hive

Dayong Du

[PACKT] open source*

Thank you

Web Resources
http://mitu.co.in
http://tusharkute.com

Blogs
http://digitallocha.blogspot.in
http://kyamputar.blogspot.in

tushar@tusharkute.com