BIG DATA

Open CV - Assignment 5

1) **Is it possible to use a script file to run HIVE queries? If so, include an example to back up your claim.**

Hive scripts are used to execute a set of *Hive commands* collectively. This helps in reducing the time and effort invested in writing and executing each command manually. This blog is a step by step guide to write your first *Hive script* and executing it.
Hive supports scripting from Hive 0.10.0 and above versions. Cloudera distribution for *hadoop (CDH4)* quick VM comes with pre-installed *Hive 0.10.0* (CDH3 Demo VM uses Hive 0.90 and hence, cannot run Hive Scripts).

Execute the following steps to create your first Hive Script:

- **Writing a script:**

Open a terminal in your *Cloudera CDH4 distribution* and give the below command to create a Hive Script.
command: **gedit sample.sql**
The *Hive script file* should be saved with **.sql** extension to enable the execution.
Edit the file and write few Hive commands that will be executed using this script.
In this sample script, we will create a table, describe it, load the data into the table and retrieve the data from this table.

- **Create a table 'product' in Hive:**

command: **create table product (productid: int, productname: string, price: float, category: string) rows format delimited fields terminated by ',' ;**
Here { productid, productname, price, category} are the columns in the 'product' table.
"*Fields* terminated by ',' " indicates that the *columns* in the input file are separated by the ',' delimiter. You can use other delimiters also. For example, the records in an input file can be separated by a *new line ('*
') character.

- **Describe the Table :**

command: **describe product;**

- **Load the data into the Table:**

To *load* the data into the table, create an input file which contains the records that needs to be inserted into the table.
command: **sudo gedit input.txt**
Create few records in the **input text** file
Command: **load data local inpath '/home/cloudera/input.txt' into table product;**

- **Retrieving the data:**

To *retrieve* the data use select command.

2) With a practical example, learn how to delete DBPROPERTY in Hive.

There is no **way to delete** or "unset" a **DBPROPERTY**. Hence, we cannot delete the **DBPROPERTY in Hive.**

3) What exactly is an index in HIVE, and how do we make one?

Indexes are **a pointer or reference to a record in a table as in relational databases**. Indexing is a relatively new feature in Hive. In Hive, the index table is different than the main table. Indexes facilitate in making query execution or search operation faster.

There are two types of indexing in Hive:

- **Bitmap Indexing:** This is used with columns having a few distinct values. It is known to store both the indexed column's value and the list of rows as a bitmap. From Hive V0.8.0 onwards, the bitmap index handler is built-in in Hive.
- **Compact Indexing:** This type of indexing is known to store the column value and storage blockid.

Example: **Create an Index**
The general syntax for creating an index for a column of a table

```
CREATE INDEX index_name
ON TABLE base_table_name (col_name, ...)
AS index_type
[WITH DEFERRED REBUILD] [IDXPROPERTIES (property_name=property_value, ...)] [IN TABLE index_table_name] [ [ ROW FORMAT ...] STORED AS ...
| STORED BY ... ] [LOCATION hdfs_path] [TBLPROPERTIES (...)] [COMMENT "index comment"]
```

Here,

- index_name will be the name of the table's index name.
- Base_table_name and the columns in bracket is the table for which index is to be created.
- Index_type will specify the type of indexing to use.

4) Describe the architecture of HBase in detail.

HBase architecture has 3 main components: HMaster, Region Server, Zookeeper.

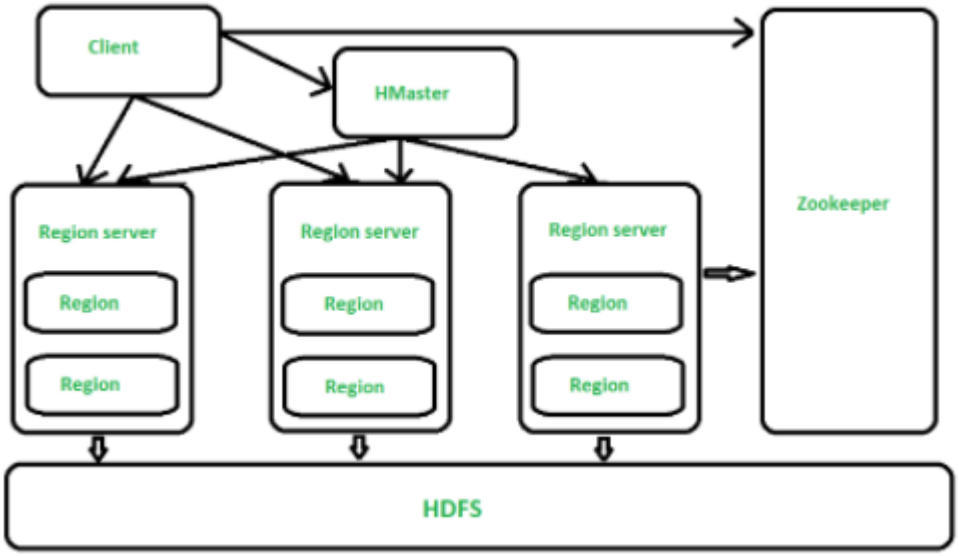

Figure – Architecture of HBase

All the 3 components are described below:

- **HMaster –**
  The implementation of Master Server in HBase is HMaster. It is a process in which regions are assigned to region server as well as DDL (create, delete table) operations. It monitor all Region Server instances present in the cluster. In a distributed environment, Master runs several background threads. HMaster has many features like controlling load balancing, failover etc.

- **Region Server –**
  HBase Tables are divided horizontally by row key range into Regions. **Regions** are the basic building elements of HBase cluster that consists of the distribution of tables and are comprised of Column families. Region Server runs on HDFS DataNode which is present in Hadoop cluster. Regions of Region Server are responsible for several things, like handling, managing, executing as well as reads and writes HBase operations on that set of regions. The default size of a region is 256 MB.

- **Zookeeper –**
  It is like a coordinator in HBase. It provides services like maintaining configuration information, naming, providing distributed synchronization, server failure notification etc. Clients communicate with region servers via zookeeper.

5) **How to read and store a 1 GB CSV file into MongoDB using Spark. Create 1 GB of data or gather it from the internet and show how it was implemented.**

First read the csv as pyspark dataframe.

```
from pyspark import SparkConf,SparkContext
from pyspark.sql import SQLContext

sc = SparkContext(conf = conf)
sql = SQLContext(sc)

df = sql.read.csv("cal.csv", header=True, mode="DROPMALFORMED")
```

Then write it to mongodb,

```
df.write.format('com.mongodb.spark.sql.DefaultSource').mode('append')\
    .option('database',NAME).option('collection',COLLECTION_MONGODB).save()
```

Specify the NAME and COLLECTION_MONGODB as created by you.

Also, you need to give conf and packages alongwith spark-submit according to your version,

```
/bin/spark-submit --conf
"spark.mongodb.inuri=mongodb://127.0.0.1/DATABASE.COLLECTION_NAME?readPreferenc
e=primaryPreferred"
        --conf
"spark.mongodb.output.uri=mongodb://127.0.0.1/DATABASE.COLLECTION_NAME"
        --packages org.mongodb.spark:mongo-spark-connector_2.11:2.2.0
        tester.py
```

Specify COLLECTION_NAME and DATABASE above. tester.py assuming name of the code file.