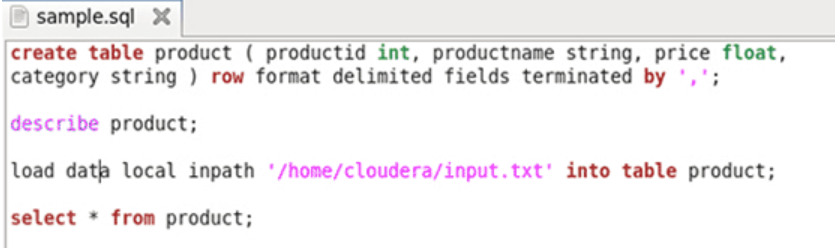
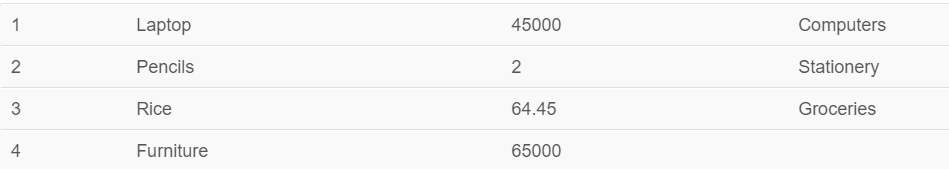
**Big Data Assignment – 2**

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

Yes, it is possible to use a script file for running Hive queries. In general, we use the scripts to execute a set of statements at once. Hive Scripts are used pretty much in the same way. It will reduce the time and effort we put on to writing and executing each command manually. Steps for using a script file to run Hive queries:

* **Step 1: Create a Hive Script file**
  + Command: gedit sample.sql
* **Step 2: Prepare the Queries**
  + Open the sample.sql file in edit mode and write the queries
  + In this script, a table will be created, described and data will be loaded and retrieved from the table.
  + The script should be like as it is shown in the below image.
    - 
  + Create a file called input.txt with data as shown below:
    - 1, Laptop, 45000, Computers
    - 2, Pencils, 2, Stationery
    - 3, Rice, 64.45, Grocery
    - 4, Furniture, 65000
* **Step 3: Running the Hive Script**
  + The following is the command to run the Hive script:
    - Command: hive –f /home/cloudera/sample.sql
  + Once all the commands are executed successfully the following output will be shown:
    - 

1. **With a practical example, learn how to delete DBPROPERTY in Hive.**

There is no way to delete or “unset” a DBPROPERTY in Hive. However, it is possible to set the key-value pairs in the DBPROPERTIES associated with a database using the ALTER DATABASE command. No other metadata about the database can be changed, including its name and directory location.

**Command:** ALTER DATABASE financials SET DBPROPERTIES ('edited-by' = 'Joe Dba')

**Note:** What is DBPROPERTY?

* The DB properties are the details about the database created by the user. Suppose the name of the user, the type of the database and the tables it has, the date on which the database is created etc. This makes the other user easy the recognize the database and use it according to the requirement.

1. **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. However, storing indexes require disk space and creating an index involves cost. So, the use of indexes may not always be of any benefit. “EXPLAIN” query must be checked to evaluate the benefit through a query execution plan. Indexing in hive makes large dataset analysis relatively quicker by better query performance on operations.

**Why do we need index?**

* With the petabytes of data that needs to be analyzed, querying Hive tables with millions of records and hundreds of columns becomes time-consuming. Indexing a table helps in performing any operation faster. First, the index of the column is checked and then the operation is performed on that column only. Without an index, queries involving filtering with the “WHERE” clause would load an entire table and then process all the rows. Indexing in Hive is **present only for ORC file format**, as it has a built-in index with it.

**Types of indexing?**

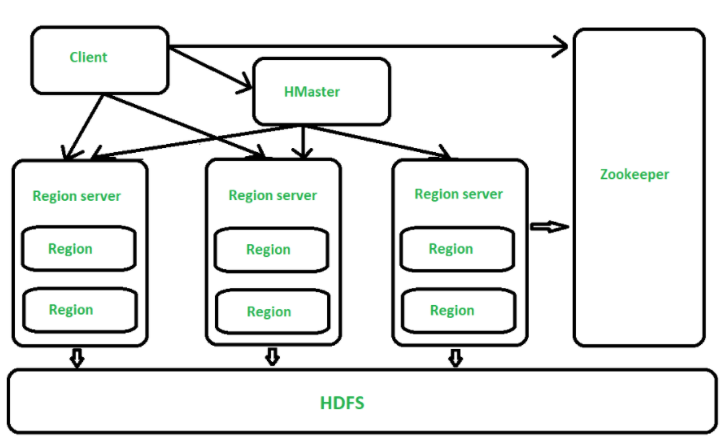
* **Bitmap Indexing:** This is used with columns having a few distinct values. It is known to store both the indexed column’s value and the list of rows as a bitmap. From Hive V0.8.0 onwards, the bitmap index handler is built-in in Hive.
* **Compact Indexing:** This type of indexing is known to store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

**Creating an Index?**

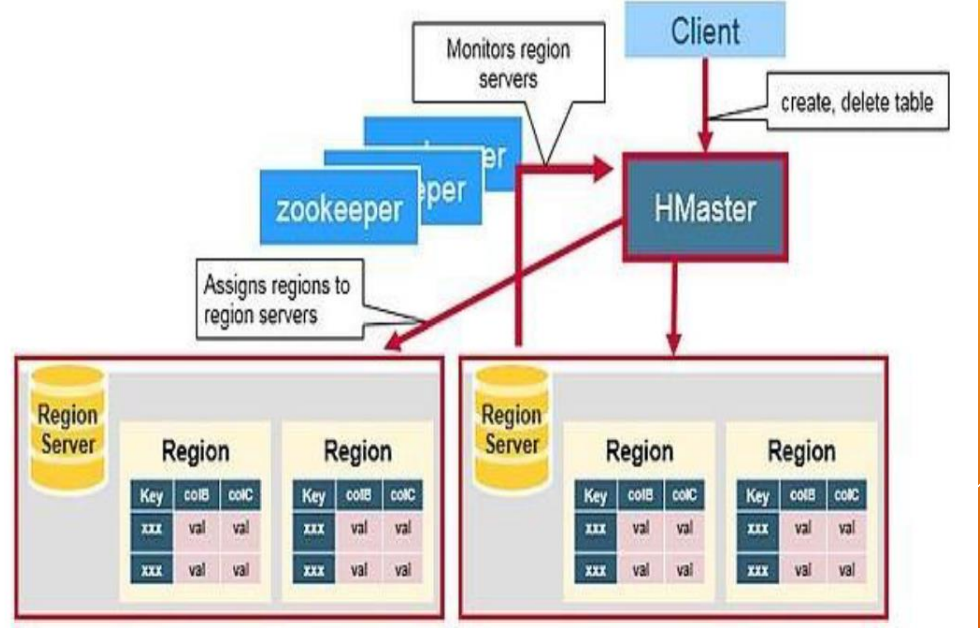
* The general syntax for creating an index for a column of a table
  + **CREATE INDEX index\_name**
  + **ON TABLE base\_table\_name (col\_name, ...)**
  + **AS index\_type**
  + **[WITH DEFERRED REBUILD] [IDXPROPERTIES (property\_name=property\_value, ...)] [IN TABLE index\_table\_name] [ [ ROW FORMAT ...] 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. If we want to use the built-in compact index handler, below clause will replace index\_type
    - org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler
  + If we want to create a bitmap index, then index\_type will be “bitmap”. It specifies the java class for implementing indexing.

1. **Describe the architecture of HBase in detail.**

HBase has three major components: a master server, region servers and the zookeeper.

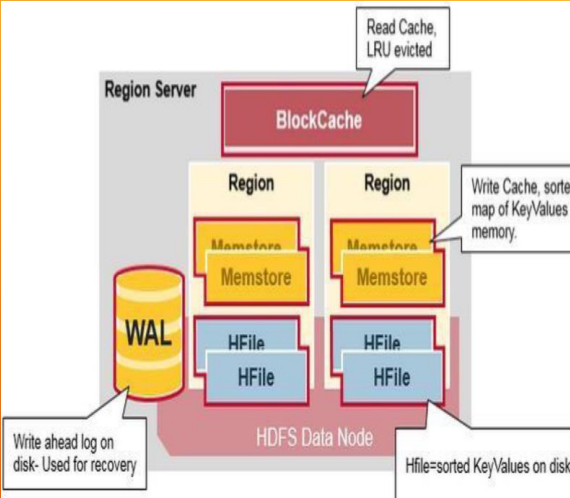


**Master Server:**

* Assigns regions to the region servers and takes the help of Apache ZooKeeper for this task.
* Handles load balancing of the regions across region servers. It unloads the busy servers and shifts the regions to less occupied servers.
* Maintains the state of the cluster by negotiating the load balancing.
* Is responsible for schema changes and other metadata operations such as creation of tables and column families.
* 

**Regions:** Regions are nothing but tables that are split up and spread across the region servers.

**Region Servers:** The region servers have regions that -

* Communicate with the client and handle data-related operations.
* Handle read and write requests for all the regions under it.
* Decide the size of the region by following the region size thresholds.
* When we take a deeper look into the region server, it contains regions and stores as shown below:
  + 
  + **WAL**:
    - Write Ahead Log is a file on the distributed file system. The WAL is used to store new data that hasn't yet been persisted to permanent storage; it is used for recovery in the case of failure.
  + **BlockCache**
    - It is the read cache. It stores frequently read data in memory. Least Recently Used data is evicted when full.
  + **MemStore**
    - It is the write cache. It stores new data which has not yet been written to disk. It is sorted before writing to disk. There is one MemStore per column family per region.
  + **Hfiles**
    - They store the rows as sorted KeyValues on disk

**Zookeeper:**

* Zookeeper is an open-source project that provides services like maintaining configuration information, naming, providing distributed synchronization, etc.
* Zookeeper has ephemeral nodes representing different region servers. Master servers use these nodes to discover available servers.
* In addition to availability, the nodes are also used to track server failures or network partitions.
* Clients communicate with region servers via zookeeper.
* In pseudo and standalone modes, HBase itself will take care of zookeeper.

1. **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.**

The reading and storing of data from CSV file to MongoDB can be achieved using PySpark. The first step is to prepare an csv file with 1 GB data. Then use the following code to read the CSV file as a PySpark dataframe:

from pyspark import SparkConf,SparkContext

from pyspark.sql import SQLContext

sc = SparkContext(conf = conf)

sql = SQLContext(sc)

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

Now the data from the csv file is loaded into the dataframe. Next step is to load the data to the MongoDB system. This can be achieved using the following code:

df.write.format('com.mongodb.spark.sql.DefaultSource').mode('append')

.option('database',NAME).option('collection',COLLECTION\_MONGODB).save()

Here, we need to specify the NAME and COLLECTION\_MONGODB as per our need. Here NAME is “cities” and COLLECTION\_MONGODB is “con”. Also, you need to give conf and packages along with spark-submit according to your version,

pyspark --conf "spark.mongodb.inuri=mongodb://127.0.0.1/DATABASE.COLLECTION\_NAME?readPreference=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

Once the operation is completed it can be verified in MongoDB using the following code:

mongo

show dbs

use cities

show collections

db.con.find().pretty()