Assignment 17.1

1. **What is NoSQL data base?**
   NoSQL unlike the RDBMS follows dont store data in tabular formats. The data structures used by NoSQL databases (e.g. key-value, wide column, graph, or document) are different from those used by default in relational databases, making some operations faster in NoSQL. NoSQL databases finds its use in big data and real-time web applications. NoSQL Databases are prefered for their simplicity in design and horizontal scaling of cluster nodes to accomodate more capacity and faster execution at comparatively cheaper cost.

2. **How does data get stored in NoSQl database?**
   Key/Value Databases- In this model, data is represented as a collection of key-value pairs. Eg: Voldemort, Redis, Scalaris.

   Columnar Databases- Stores columns of data together than rows. Query performance is often increased as a result, particularly in very large data sets. eg -HBase, Hypertable, Cassandra

   Document Databases -Document-oriented databases are inherently a subclass of the key-value store. A document can have any number of key pair values or key array values. The difference lies in the way the data is processed; in a key-value store the data is considered to be inherently opaque to the database, whereas a document-oriented system relies on internal structure in the document in order to extract metadata that the database engine uses for further optimization. eg -MongoDb, CouchDB

   Graph Databases- data whose relations are well represented as a graph consisting of elements interconnected with a finite number of relations between them. The type of data could be social relations, public transport links, road maps or network topologies. eg - InfoGrid, Neo4j

3. **What is a column family in HBase?**
   Columns in Apache HBase are grouped into column families. All column members of a column family have the same prefix. For example, the columns courses:history and courses:math are both members of the courses column family. The colon character (:) delimits the column family from the . The column family prefix must be composed of printable characters. The qualifying tail, the column family qualifier, can be made of any arbitrary bytes. Column families
   must be declared up front at schema definition time whereas columns do not need to be defined at schema time but can be conjured on the fly while the table is up an running.
   Physically, all column family members are stored together on the filesystem. Because tunings and storage specifications are done at the column family level, it is advised that all column family members have the same general access pattern and size characteristics

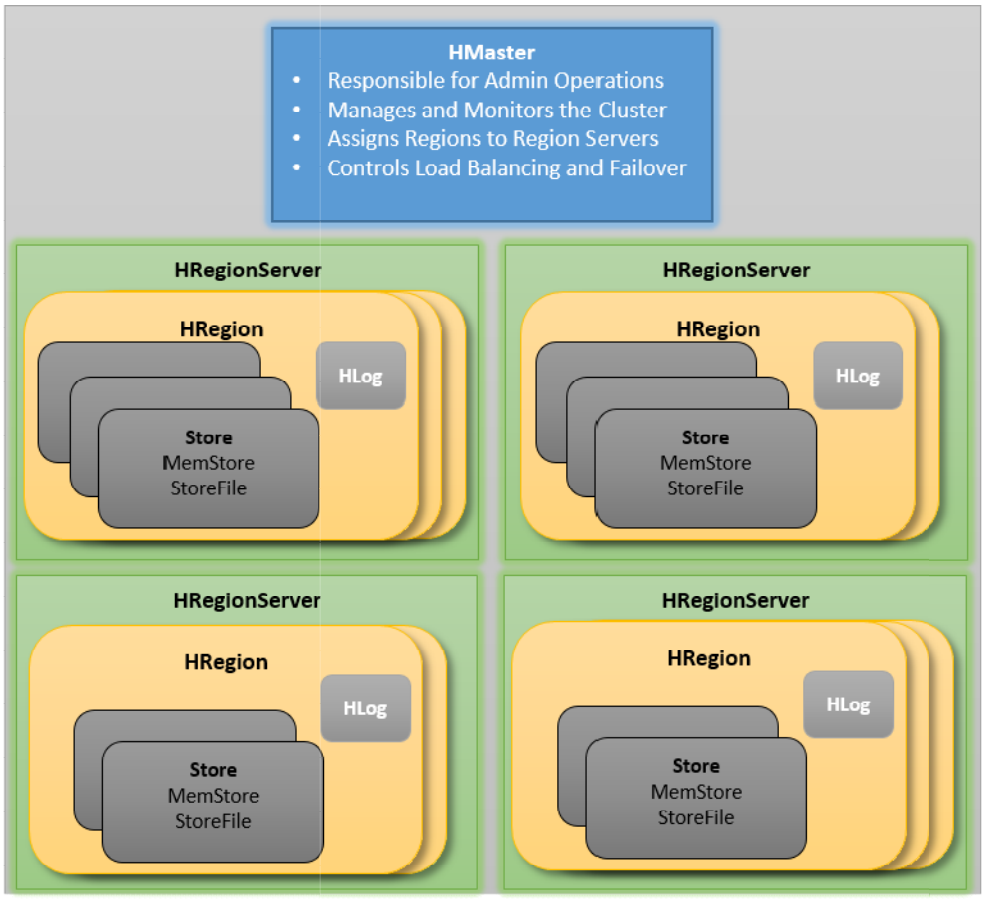
4. **How many maximum number of columns can be added to HBase table?**
   There are no limits to number of columns. Developers can choose either
   1) more rows with less columns
   2) more columns with less rows

5. **Why columns are not defined at the time of table creation in HBase?**
   HBase tables are created with column families which generally is not changed. The column keys are specified after the table is up and running. Operations such as tunings and storage specifications are done at the column family level and are hence stored in separate HFiles.

6. **How does data get managed in HBase?**

The HBase Physical Architecture consists of servers in a Master-Slave relationship as shown below. Typically, the HBase cluster has one Master node, called HMaster and multiple Region Servers called HRegionServer. Each Region Server contains multiple Regions – HRegions.

Just like in a Relational Database, data in HBase is stored in Tables and these Tables are stored in Regions. When a Table becomes too big, the Table is partitioned into multiple Regions. These Regions are assigned to Region Servers across the cluster. Each Region Server hosts roughly the same number of Regions.



The HMaster in the HBase is responsible for
Performing Administration
Managing and Monitoring the Cluster
Assigning Regions to the Region Servers
Controlling the Load Balancing and Failover
On the other hand, the HRegionServer perform the following work
Hosting and managing Regions
Splitting the Regions automatically
Handling the read/write requests
Communicating with the Clients directly

Each Region Server contains a Write-Ahead Log (called HLog) and multiple Regions. Each Region in turn is made up of a MemStore and multiple StoreFiles (HFile). The data lives in these StoreFiles in the form of Column Families (explained below). The MemStore holds in-memory modifications to the Store (data).

The mapping of Regions to Region Server is kept in a system table called .META. When trying to read or write data from HBase, the clients read the required Region information from the .META table and directly communicate with the appropriate Region Server. Each Region is identified by the start key (inclusive) and the end key (exclusive)

7. What happens internally when new data gets inserted into HBase table?

You can insert data into Hbase using the **add()** method of the **Put** class. You can save it using the **put()** method of the **HTable** class. These classes belong to the **org.apache.hadoop.hbase.client** package. Below given are the steps to create data in a Table of HBase.

Step 1:Instantiate the Configuration Class

The **Configuration** class adds HBase configuration files to its object. You can create a configuration object using the **create()** method of the **HbaseConfiguration** class as shown below.

```
Configuration conf = HbaseConfiguration.create();
```

Step 2:Instantiate the HTable Class

You have a class called **HTable**, an implementation of Table in HBase. This class is used to communicate with a single HBase table. While instantiating this class, it accepts configuration object and table name as parameters. You can instantiate HTable class as shown below.

```
HTable hTable = new HTable(conf, tableName);
```

Step 3: Instantiate the PutClass

To insert data into an HBase table, the **add()** method and its variants are used. This method belongs to **Put**, therefore instantiate the put class. This class requires the row name you want to insert the data into, in string format. You can instantiate the **Put** class as shown below.

```
Put p = new Put(Bytes.toBytes("row1"));
```

Step 4: Insert Data

The **add()** method of **Put** class is used to insert data. It requires 3 byte arrays representing column family, column qualifier (column name), and the value to be inserted, respectively. Insert data into the HBase table using the add() method as shown below.

```
p.add(Bytes.toBytes("coloumn family "), Bytes.toBytes("column name"),Bytes.toBytes("value"));
```

Step 5: Save the Data in Table

After inserting the required rows, save the changes by adding the put instance to the **put()** method of HTable class as shown below.

```
hTable.put(p);
```

Step 6: Close the HTable Instance

After creating data in the HBase Table, close the **HTable** instance using the **close()** method as shown below.

```
hTable.close();
```

This is how data is created in Hbase