**Assignment 9.1**

**1. What is NoSQL data base?**

A NoSQL (originally referring to "non SQL" or "non relational") database provides a mechanism for [storage](https://en.wikipedia.org/wiki/Computer_data_storage) and [retrieval](https://en.wikipedia.org/wiki/Data_retrieval) of data that is modeled in means other than the tabular relations used in [relational databases](https://en.wikipedia.org/wiki/Relational_database).

Triggered by the needs of [Web 2.0](https://en.wikipedia.org/wiki/Web_2.0) companies such as [Facebook](https://en.wikipedia.org/wiki/Facebook), [Google](https://en.wikipedia.org/wiki/Google), and [Amazon.com](https://en.wikipedia.org/wiki/Amazon.com). NoSQL databases are increasingly used in [big data](https://en.wikipedia.org/wiki/Big_data) and [real-time web](https://en.wikipedia.org/wiki/Real-time_web) applications.

NoSQL systems are also sometimes called "Not only SQL" to emphasize that they may support [SQL](https://en.wikipedia.org/wiki/SQL)-like query languages.

**2. How does data get stored in NoSQl database?**

Data is stored in in single tables as compared to joining multiple tables.

Graph stores are used to store information about networks of data, such as social connections. Graph stores include Neo4J and Giraph. Key-value stores are the simplest NoSQL databases. Every single item in the database is stored as an attribute name (or 'key'), together with its value.

**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 column family qualifier. 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?**

HBase currently does not do well with anything above two or three column families so it is good to keep the number of column families in a schema low. Currently, flushing and compactions are done on a per Region basis so if one column family is carrying the bulk of the data bringing on flushes, the adjacent families will also be flushed though the amount of data they carry is small. When many column families the flushing and compaction interaction can make for a bunch of needless i/o loading

It is good to have one column family if you can in a schemas. Only introduce a second and third column family in the case where data access is usually column scoped; i.e. you query one column family or the other but usually not both at the one time.

**5. Why columns are not defined at the time of table creation in HBase?**

Columns in Apache HBase are grouped into column families. All column members of a column family have the same prefix.

Column exists only when inserted and can have multiple versions, each column identified by it’s key.

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.

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

HBase is built on top of the distributed file system (DFS), which can store large files. HBase provides fast record lookups and updates for large tables.

The ZooKeeper cluster acts as a coordination service for the entire HBase cluster.

HBase contains two primary services:

Master server:

The master server co-ordinates the cluster and performs administrative operations, such as assigning regions and balancing the loads.

Region server:

The region servers do the real work. A subset of the data of each table is handled by each region server. Clients talk to region servers to access data in HBase.

Regions:

Region servers manage a set of regions.

An HBase table is made up of a set of regions. Regions are the basic unit of work in HBase. It is what is used as a split by the map reduce framework. The region contains store objects that correspond to column families. There is one store instance for each column family. Store objects create one or more StoreFiles, which are wrappers around the actual storage file called the HFile.

The region also contains a MemStore, which is in-memory storage and is used as a write cache. Rows are written to the MemStore. The data in the MemStore is ordered. If theMemStore becomes full, it is persisted to an HFile on disk

To improve performance, it is important to get an even distribution of data among regions, which ensures the best parallelism in map tasks.

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

Data can be inserted in to HBase table using put command

Syntax:

*put ‘<table name>’ , ‘row1’, ‘<colfamily : colname>’,’<value>’*

Inserting data using Java API

Data can be inserted into Hbase using the add() method of the put class, which belongs to the org.apache.hadoop.hbase.client package.