Assignment 9.1

Problem Statement

1. What is NoSQL data base?

NoSQL (originally referring to "non SQL" or "non relational") database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.

2. How does data get stored in NoSQl database?

There are various NoSQL Databases. Each one uses a different method to store data. Some might use column store, some document, some graph, etc., Each database has its own unique characteristics**.**

There are four ways by which data get stored in NoSQL databases, each with their own specific attributes:

Graph database – Based on graph theory, these databases are designed for data whose relations are well represented as a graph and has elements which are interconnected, with an undetermined number of relations between them. Examples include: Neo4j and Titan.

Key-Value store – we start with this type of database because these are some of the least complex NoSQL options. These databases are designed for storing data in a schema-less way. In a key-value store, all of the data within consists of an indexed key and a value, hence the name. Examples of this type of database include:Cassandra, DyanmoDB, Azure Table Storage (ATS), Riak, BerkeleyDB.

Column store – (also known as wide-column stores) instead of storing data in rows, these databases are designed for storing data tables as sections of columns of data, rather than as rows of data. While this simple description sounds like the inverse of a standard database, wide-column stores offer very high performance and a highly scalable architecture. Examples include: HBase, BigTable and HyperTable.

Document database – expands on the basic idea of key-value stores where “documents” contain more complex in that they contain data and each document is assigned a unique key, which is used to retrieve the document. These are designed for storing, retrieving, and managing document-oriented information, also known as semi-structured data. Examples include: MongoDB and CouchDB.

3. What is a column family in HBase?

HBase is a column-oriented database and the tables in it are sorted by row. The table schema defines only column families, which are the key value pairs. A table have multiple column families and each column family can have any number of columns. Subsequent column values are stored contiguously on the disk. Each cell value of the table has a timestamp.

A column family defines shared features to all columns that are created within them (think of it almost as a sub-table within your larger table).

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 ofthe 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.

Given below is an example schema of table in HBase.

Rowid	Column Family			Column Family			Column Family			Column Family		
	clo1	col 2	col 3	col 1	col 2	col 3	col1	col2	col3	col1	col2	col3
1												
2												
3												

4. How many maximum number of columns can be added to HBase table?

There is no limit on number of columns that get added to an HBase table

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.

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 other. .

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 natively supported on Hadoop. The main characteristics that make Hbase an excellent data management platform are

1. fault tolerance,
2. speed and
3. usability.

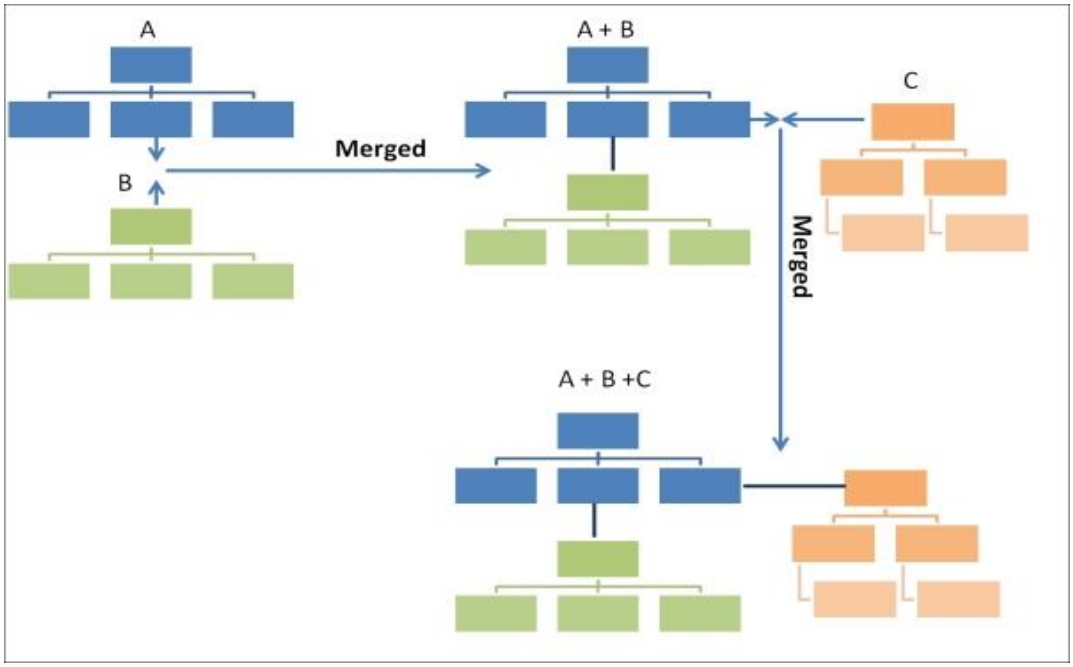
Fault tolerance is provided by automatic fail-over, automatically sharded and load balanced tables, strong consistency in row level operations and replication.

Speed is provided by almost real time lookups, in memory caching and server side processing. Usability is provided by a flexible data model that allows many uses, a simple Java API and ability to export metrics.

The Hbase data model is different from the model provided by relational databases. Hbase is referred to by many terms like a key-value store, column oriented database and versioned map of maps which are correct. The easiest way of visualizing a Hbase data model is a table that has rows and tables. This is the only similarity shared by Hbase model and the relational model.

The internal storage architecture of HBase

The following figure shows the principle algorithm and data structure HBase works on, that is, log-structured merge-tree LSM-tree, and the way of merging, and precedes the explanation:



HBase stores file using LSM-tree, which maintains data in two separate parts that are optimized for underlying storage. This type of data structure depends on two structures, a current and smaller one in memory and a bigger one on the persistent disk, and once the part in memory becomes bigger than a certain limit, it is merged with the bigger structure that is stored on the disk using a merge sort algorithm and a new in-memory tree is created for newer insert requests. It transforms random data access into sequential data access, which improves read performance, and merging is a background process, which does not affect the foreground processing.

Data model operations

The operations that are the basic blocks of data model are Get, Put, Scan, and Delete, using which we can read, write, and delete records from an HBase table.

Get

The Get operation can fetch certain records from an HBase table. It is similar to the select [fields] where RowKey=<Row Key value here> statement in relational databases, where we fetch a row from the table.

The following is the representation of Get:

```
public Result get(Get get)throws IOException
```

In the preceding code, the Get operation can be provided as a single get object out of a list of get objects as get(List<Get> gets). It is specified by Get in the HTableInterface interface given by HBase. The Get operation receives the get parameter, which objects

the data that is to be fetched from the table. It returns data of the particular row, which is specified in the get object as an HBase Result object, and this throws IOException when not able to read.

On HBase shell, it can be used as follows:

```
get 'table name', 'row key',<filters>
```

Put

The Put operation adds a new row of data to a table or updates/overrides a specific row of data. It is executed through HTable.put() or HTable.batch(), which is a batch write operation.

This takes put as parameter. We can see its use as follows:

```
public void put(Put put) throws
InterruptedIOException,RetriesExhaustedWithDetailsException
```

This also takes a single put object or a list of put objects to write a set of values in a table.

Scan

The Scan operation can be used to read multiple rows of data in contrast to Get where we need to specify a set of rows to read data. However, in the case of a scan, we can iterate through a range of rows or all the rows in a table.

Delete

The Delete operation removes a row or a set of rows from a table. It is executed through HTable.delete(). Once a row is set to be deleted, it is marked as tombstone, and once compaction takes place, the row is finally deleted or removed from a table.

```
public void delete (Delete delete) throws IOException
```

delete is specified as an interface of HTableInterface, takes the delete object or delete list as a parameter, and throws IOException if any intermediate exception occurs.

The Delete happens for the following:

Delete: This is for a specific version of a column

Delete column: This is for all versions of a column

Delete family: This is for all columns of a particular column family

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

To create data in an HBase table, the following commands and methods are used:

- **put** command,:

  Using **put** command, you can insert rows into a table.

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

- **add()** method of Java **Put** class, and **put()** method of Java **HTable** class.

  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.

Steps:

1. The client asks Zookeeper the location of .META.
2. The client scans .META. searching for the Region Server responsible to handle the key.
3. The client asks the Region Server to insert/update/delete the specified key/value.
4. The Region Server process the request and dispatch it to the Region responsible to handle key
    1. The operation is written to a Write Ahead Log(WAL)
    2. and the Key Values added to the Store:"MemoStore "