HBase Assignment 1 :

1) What is NoSQL data base?

NoSQL means Not Only SQL, implying that when designing a software solution or product, there are more than one storage mechanism that could be used based on the needs. NoSQL was a hashtag (#nosql) choosen for a meetup to discuss these new databases. The most important result of the rise of NoSQL is Polyglot Persistence. NoSQL does not have a prescriptive definition but we can make a set of common observations, such as:

- Not using the relational model
- Running well on clusters
- Mostly open-source
- Built for the 21st century web estates
- Schema-less

2) How does data get stored in NoSQl database?

Several different varieties of NoSQL databases have been created to support specific needs and use cases. These fall into four main categories:

- **Document databases** pair each key with a complex data structure known as a document. Documents can contain many different key-value pairs, or key-array pairs, or even nested documents.
- **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. Examples of key-value stores are Riak and Berkeley DB. Some key-value stores, such as Redis, allow each value to have a type, such as 'integer', which adds functionality.
- **Wide-column stores** such as Cassandra and HBase are optimized for queries over large datasets, and store columns of data together, instead of rows.

3) What is a column family in HBase?

An HBase table contains column families, which are the logical and physical grouping of columns. There are column qualifiers inside of a column family, which are the columns. Column families contain columns with time stamped versions. Columns only exist

when they are inserted, which makes HBase a sparse database. An HBase column can be specified by using the following format:

hbase-family:hbase-column-name

Columns in Apache HBase are grouped into column families. All column members of a column family have the same prefix. Each column value is identified by a key. The row key is the implicit primary key. Rows are sorted by the row key.

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 name . 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's not really a limit. Here are some things to consider:

Lock granularity

When you do an operation within a row, the RegionServer code briefly holds a lock on that row while applying the mutation.

On the plus side, this means that you can act atomically on several columns - concurrent readers will either see the entire update or won't see the update at all. They shouldn't (barring one or two bugs we're still stomping on) see a partial update.

On the minus side, this means that the throughput of write operations within a single row is limited (probably a few hundred per second).

We're currently working on some optimizations for specific cases like increment so that multiple incrementers can "line up" behind the lock and then batch their addition together into a single transaction.

Region distribution

The unit of load balancing and distribution is the region, and a row will never be split across regions. So, no matter how hot a row is, it will always be served by a single server. If the data were split across many rows, you could force a split in between two hot rows to distribute load between two hosts.

Bugs

In prior versions of HBase there were some bugs where we would accidentally load or deserialize an entire row into RAM. So if your row is very large (100s of MBs) you may have run into serious performance issues, OOMEs, etc. I think most of these bugs are since squashed, and the RS does a smart job of only loading the necessary columns, but it's something to be aware of.

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

In order to support scaling out columns are not defined during table creation. 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 and running .

The HBase Put is used to insert data. It requires 3 byte arrays representing column family, column qualifier (column name), and the value to be inserted, respectively. The syntax is as below.

```
put '<HBase_table_name>','row_key','<colfamily:colname>','<value>'
```

If the column name doesnt exist in the column family, the put command will create the column name in the column family.

6) How does data get managed in HBase?

Hbase commands can be broadly divided into five categories. The commands operate in a similar way to those in relational databases. Security commands are used to GRANT, REVOKE and show USER_PERMISSION. Cluster replication commands are used to manage a cluster. Some cluster management activities are: ADD_PEER, REMOVE_PEER, DISABLE_PEER and STOP_REPLICATION.

Data manipulation commands include COUNT, DELETE, DELETEALL and SCAN. Some table management commands are: ALTER, CREATE, DESCRIBE, DROP, and DROPALL. Some general commands are VERSION and STATUS. This is a very small list of commands available for managing data in Hbase.

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.

Data in Hbase is organized into tables. Any characters that are legal in file paths are used to name tables. Tables are further organized into rows that store data. Each row is identified by a unique row key which does not belong to any data type but is stored as a bytearray. Column families are further used to group data in rows. Column families define the physical structure of data so they are defined upfront and their modification is difficult. Each row in a table has same column families.

Data in a column family is addressed using a column qualifier. It is not necessary to specify column qualifiers in advance and there is no consistency requirement between rows. No data types are specified for column qualifiers, as such they are just stored as bytearrays. A unique combination of row key, column family and column qualifier forms a cell. Data contained in a cell is referred to as cell value. There is no concept of data type when referring to cell values and they are stored as bytearrays. Versioning happens to cell values using a timestamp of when the cell was written.

Tables in Hbase have several properties that need to be understood for one to come up with an effective data model. Indexing and sorting only happens on the row key. The concept of data types is absent and everything is stored as bytearray. Only row level atomicity is enforced so multi row transactions are not supported.

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

Puts and Deletes are collected into an in-memory structure called the MemStore. Before the MemStore is update the changes are written to a Write Ahead Log (WAL) to enable recovery in case a server crashes.

When it reaches a certain size the MemStore is flushed to disk into StoreFile. Periodically StoreFiles are compacted into fewer StoreFiles.

For reading and writing HBase employs Log Structured Mergetrees, which is just a fancy way of saying that reading and compacting in HBase is performing a merge sort (a scan looks at the heads of all StoreFiles and the Memstore and picks the smallest element first, in case of a Scan it is returned to the client, in case of a compacted it is written to the new StoreFile).