**HBASE ASSIGNMENT 10**

**Task 1**

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

NoSQL is an approach to [database](https://searchsqlserver.techtarget.com/definition/database) design that can accomodate a wide variety of data models, including key-value, document, columnar and graph formats. NoSQL, which stand for "not only [SQL](https://searchsqlserver.techtarget.com/definition/SQL)," is an alternative to traditional relational databases in which data is placed in tables and data [schema](https://searchsqlserver.techtarget.com/definition/schema) is carefully designed before the database is built. NoSQL databases are especially useful for working with large sets of distributed data.

The Benefits of NoSQL

When compared to relational databases, NoSQL databases are [more scalable and provide superior performance,](https://www.mongodb.com/scale) and their data model addresses several issues that the relational model is not designed to address:

* Large volumes of rapidly changing structured, semi-structured, and unstructured data
* Agile sprints, quick schema iteration, and frequent code pushes
* Object-oriented programming that is easy to use and flexible
* Geographically distributed scale-out architecture instead of expensive, monolithic architecture

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

* 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. Examples are [ArangoDB](http://www.arangoDB.com/), [OrientDB](http://orientdb.com), [gunDB](http://gunDB.io),MongoDB.
* 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 ,Hadoop 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?**

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 on 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 to number of columns in Hbase table.

*But HBase currently does not do well with anything above two or three column families so keep the number of column families in your 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 (To be addressed by changing flushing and compaction to work on a per column family basis).*

*Try to make do with one column family if you can in your 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**?

*Column families are specified when a table is created. They should be carefully designed before a table is created since it would be either impossible or difficult to change them later.*

*Column families’ names are strings that are composed of characters that are safe to use in file system paths.*

*All columns in a column family are stored and sorted together in the same HFile.*

*Column families group columns together physically and logically and they are usually used for a performance reason. A column family has a set of parameters that specify its storage (e.g., caching, compression, etc.). All tuning and storage specifications are done at the column family level. It is important that all column family members have the same or similar access pattern and sizes.*

*Some shortcomings in the current HBase implementation do not properly support large number of column families in a single table. That number should be in low tens. Most of the time up to three column families should work fine without any significant performance drawback. Ideally you should go with a single column family. The column family names should be as small as possible, preferably one character.*

*A column family can have an arbitrary number of columns denoted by a column qualifier which is like a column’s label. For example:*

*Tables are declared up front at schema definition time. Row keys are arrays of bytes and they are lexicographically sorted with the lowest order appearing first.*

*HBASE returns the latest version of data by default but you can ask for multiple versions in your query. HBase returns data sorted first by the row key values, then by column family, column qualifier and finally by the timestamp value, with the most recent data returned first.*

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

*Hbase is natively supported on Hadoop and it is the subject of this tutorial. The main characteristics that make Hbase an excellent data management platform are fault tolerance, speed and 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.*

*Hbase can run standalone on the local file system but this set up does not guarantee durability. Edits will be lost when daemons are not cleanly started and stopped. Such a set up is not suitable in a production environment but it provides a way of exploring how the database functions. Alternatively Hbase can be installed on a single or multi node cluster and use HDFS. This set up requires a working set up of Hadoop. If you have not yet installed Hadoop please refer to the setting up Hadoop tutorial. Type hadoop version at the terminal to check Hadoop is correctly installed.*

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

There is a special HBase Catalog table called the META table, which holds the location of the regions in the cluster. ZooKeeper stores the location of the META table.

This is what happens the first time a client reads or writes to HBase:

1. The client gets the Region server that hosts the META table from ZooKeeper.
2. The client will query the .META. server to get the region server corresponding to the row key it wants to access. The client caches this information along with the META table location.
3. It will get the Row from the corresponding Region Server.

For future reads, the client uses the cache to retrieve the META location and previously read row keys. Over time, it does not need to query the META table, unless there is a miss because a region has moved; then it will re-query and update the cache

When the client issues a Put request, the first step is to write the data to the write-ahead log, the WAL:

-Edits are appended to the end of the WAL file that is stored on disk.

- The WAL is used to recover not-yet-persisted data in case a server crashes.

Once the data is written to the WAL, it is placed in the MemStore. Then, the put request acknowledgement returns to the client.

**Task 2**

1. **Create an HBase table named 'clicks' with a column family 'hits' such that it should be**

**able to store last 5 values of qualifiers inside 'hits' column family.**

2. Add few records in the table and update some of them. Use IP Address as row-key. Scan

the table to view if all the previous versions are getting displayed.