**Assignment - 10.1**

***Task1.1:*** What is NoSQL data base?

***Answer1.1:*** **NoSQL** – stands for **“not only SQL”**, is the Next Generation Databases mostly addressing some of the points: being **non-relational, distributed, open-source** and **horizontally scalable**.

The original intention has been **modern web-scale databases**. The movement began early 2009 and is growing rapidly. Often more characteristics apply such as: **schema-free, easy replication support, simple API, eventually consistent** / **BASE** (not ACID), a **huge amount of data** and more.

***Task1.2:*** How does data get stored in NoSQl database?

***Answer1.2:*** Majorly there are four ways ofstoring data in NoSQL database –

* Key – Value Stores.
* Column – Family Stores.
* Document Database
* Graph Database.

**Key – Value Stores:**

* Examples: Dynamo, Voldemort etc.
* Key - Value is based on a hash table where there is a unique key and a pointer to a particular item of data (value)
* Mappings are usually accompanied by cache mechanisms to maximize performance
* API is typically simple - implementation is often complex
* The values are not queryable

**Column – Family Stores:**

* Big Table, Cassandra, HBase.
* Stores and processes Big Data in real-time
* The database Key points to column families comprising of multiple columns

**Document Database:**

* CouchDB, MongoDB, Lotus Notes, Redis ...
* Documents are addressed in the database via a unique key that represents that document
* The structure of documents can be XML, JSON or BSON formatted, for instance
* In addition to the key, documents can be retrieved with queries within the values

**Graph Database:**

* Neo4J, FlockDB, GraphBase, InfoGrid, ...
* Graph Databases are built with nodes, relationships between nodes (edges) and the properties of nodes
* Nodes represent entities (e.g. "Bob" or "Alice").
* Similar in nature to the objects as in object-oriented programming
* Properties are pertinent information related to nodes (e.g. age: 18)
* Edges connect nodes to nodes or nodes to properties
* Edges represent the relationship between the two nodes.

***Task1.3:*** What is a column family in HBase?

***Answer1.3:*** 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 andcourses: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 ofprintable 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.

***Task1.4:*** How many maximum numbers of columns can be added to HBase table?

***Answer1.4:*** There is no hard limit to number of columns in HBase , we can have more than 1 million columns but usually three column families are recommended ( not more than three).

***Task1.5:*** Why columns are not defined at the time of table creation in HBase?

***Answer1.5:*** **Columns** are arbitrary names (or labels) assigned by the application.  
  
The timestamp is a long identifying (by default) the creation time of the of the cell. Each cell (as opposed to row) is versioned, which makes it interesting to reason about consistency and ACID guarantees (more on that later). No data is ever overwritten or changed in place, instead every "update" creates a new version of the affected set of cells.

**Column families** are declared when a table is created. They define storage attributes such as compression, number of versions to maintain, time to live, and minimum number of versions - among others.

***Task1.6:*** How does data get managed in HBase?

***Answer1.6:*** HBase is a column-oriented database and data is stored in tables. The tables are sorted by RowId. As shown below, HBase has RowId, which is the collection of several column families that are present in the table.

he column families that are present in the schema are key-value pairs. If we observe in detail each column family having a multiple numbers of columns. The column values stored in to disk memory. Each cell of the table has its own Meta data like time stamp and other information.

Coming to HBase the following are the key terms representing table schema

* **Table**: Collection of rows present.
* **Row**: Collection of column families.
* **Column** **Family**: Collection of columns.
* **Column**: Collection of key-value pairs.
* **Namespace**: Logical grouping of tables.
* **Cell**: A {row, column, version} tuple exactly specifies a cell definition in HBase.

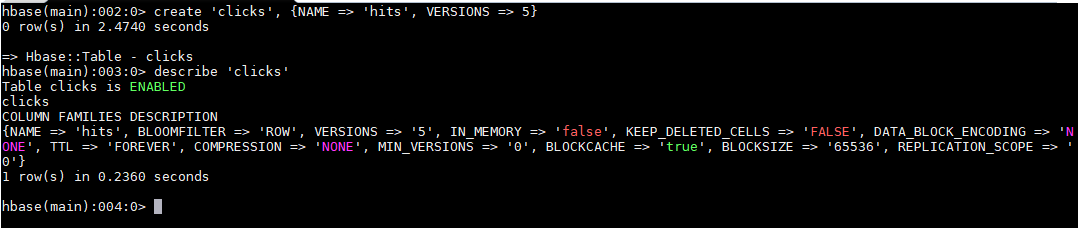
***Task1.7:*** What happens internally when new data gets inserted into HBase table?

***Answer1.7:*** 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).

***Task2.1:*** 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.

***Answer2.1:*** This can be achieved by mentioning VERSION limit in the create table command as mentioned below. Here the no. of versions are limited to 5.

**create 'clicks', {NAME => 'hits', VERSIONS => 5}**



***Task2.2:*** 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.

***Answer2.2:*** As we can see the table is created successfully, now proceed to insert some values using put command as below.

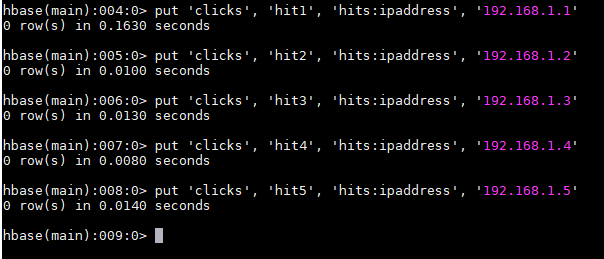
**put 'clicks', 'hit1', 'hits:ipaddress', '192.168.1.1'**

**put 'clicks', 'hit2', 'hits:ipaddress', '192.168.1.2'**

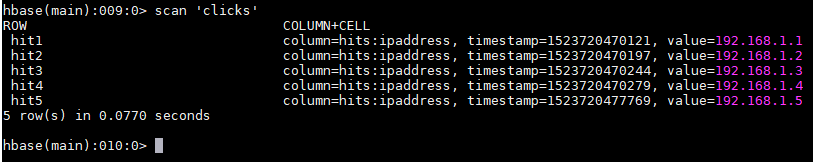
**put 'clicks', 'hit3', 'hits:ipaddress', '192.168.1.3'**

**put 'clicks', 'hit4', 'hits:ipaddress', '192.168.1.4'**

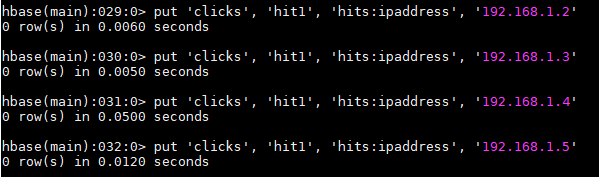
**put 'clicks', 'hit5', 'hits:ipaddress', '192.168.1.5'**



Now scan the table to check the content of the table.



Now update the hits using put commands.



Now as the table has been created with version limit of 5, so we can see here even if ‘n’ no. of times a row is updated, the history holds only last five values. This is received by below mentioned command.

**get 'clicks', 'hit1', {COLUMN=>'hits:ipaddress',VERSIONS=>5}**

