**Assignment9.1**

**Problem Statement:**

**What is NoSQL data base?**

A 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. NoSQL systems are also sometimes called "Not only SQL" to emphasize that they may support SQL-like query languages. Motivations for this approach include: simplicity of design, simpler "horizontal" scaling to clusters of machines (which is a problem for relational databases),and finer control over availability. The data structures used by NoSQL databases (e.g. key-value, wide column, graph, or document) are different from those used by default in relational databases, making some operations faster in NoSQL. The particular suitability of a given NoSQL database depends on the problem it must solve. Sometimes the data structures used by NoSQL databases are also viewed as "more flexible" than relational database tables.

**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.

**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.

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

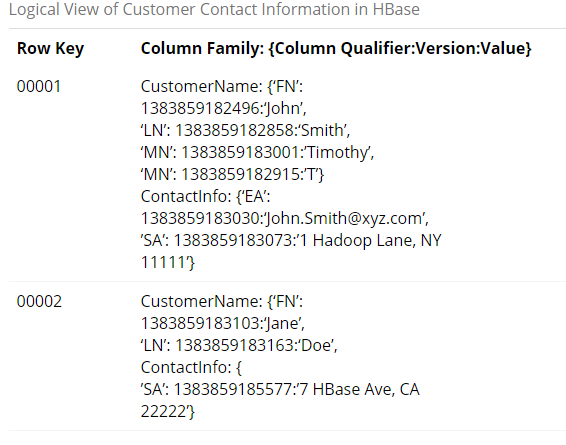
Agile sprints, quick schema iteration, and frequent code pushes

Object-oriented programming that is easy to use and flexible

Geographically distributed scale-out architecture instead of expensive, monolithic architecture

**What is a column family in HBase?**

In the HBase data model columns are grouped into *column families*, which must be defined up front during table creation. Column families are stored together on disk, which is why HBase is referred to as a column-oriented data store.



The table shows two column families: CustomerName and ContactInfo. When creating a table in HBase, the developer or administrator is required to define one or more column families using printable characters.

Generally, column families remain fixed throughout the lifetime of an HBase table but new column families can be added by using administrative commands. The official recommendation for the number of column families per table is three or less.

In addition, you should store data with similar access patterns in the same column family — you wouldn’t want a customer’s middle name stored in a separate column family from the first or last name because you generally access all name data at the same time.

Column families are grouped together on disk, so grouping data with similar access patterns reduces overall disk access and increases performance.

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

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). Depending on your data access patterns, you should consider wide table vs tall table layout.

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

There is really no such thing as a nested table in hbase ... sometimes it is called nested entity. The main idea is really a column family. A single column family contains one or more columns. Column families must be defined at table creation time but columns can be added dynamically after table creation (if an insert statement states a column that does not exist for a column family it will create it). Column families thus can be seen as holding an array of information that may have different lengths among rows (keys). You do not have to use it that way: you can always use identical columns for each column family. Another feature of column families is that they are written to their own files. Thus queries read only the column families holding the columns in the queries. This allows you to design very wide tables (hundreds of columns) and read only a subset of columns for each query (resulting in faster performance). Also, column families can have different properties, e.g one can be compressed and others not. Thus the general rule is to group columns that will be queried together into the same column family and allow the number of columns in a column family to be dynamic among records if you wish.

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

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.

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

When you put data into HBase, a timestamp is required. The timestamp can be generated automatically by the RegionServer or can be supplied by you. The timestamp must be unique per version of a given cell, because the timestamp identifies the version. To modify a previous version of a cell, for instance, you would issue a Put with a different value for the data itself, but the same timestamp.

HBase's behavior regarding versions is highly configurable. The maximum number of versions defaults to 1 in CDH 5, and 3 in previous versions. You can change the default value for HBase by configuring hbase.column.max.version in hbase-site.xml, either using an advanced configuration snippet if you use Cloudera Manager, or by editing the file directly otherwise.

You can also configure the maximum and minimum number of versions to keep for a given column, or specify a default time-to-live (TTL), which is the number of seconds before a version is deleted. The following examples all use alter statements in HBase Shell to create new column families with the given characteristics, but you can use the same syntax when creating a new table or to alter an existing column family. This is only a fraction of the options you can specify for a given column family. 