**ASSIGNMENT 3.7**

**1)What is NoSQL database?**

NoSQL is a class of database management systems (DBMS) that do not follow all of the rules of a relational DBMS and cannot use traditional SQL to query data. The term is somewhat misleading when interpreted as "No SQL," and most translate it as "Not Only SQL," as this type of database is not generally a replacement but, rather, a complementary addition to RDBMSs and SQL.

* Non-relational (NoSQL) databases typically do not enforce a schema. A partition key is generally used to retrieve values, column sets, or semi-structured JSON, XML, or other documents containing related item attributes.
* Non-relational (NoSQL) databases typically do not enforce a schema. A partition key is generally used to retrieve values, column sets, or semi-structured JSON, XML, or other documents containing related item attributes.
* Performance is generally a function of underlying hardware cluster size, network latency, and the calling application.
* Designed to scale “out” using distributed clusters of low-cost hardware to increase throughput without increasing latency.
* NoSQL databases generally offer tools to manage clusters and scaling. Applications are the primary interface to the underlying data.

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

Data storage model of NoSQl database varies based on database type. For example, key-value stores function similarly to SQL databases, but have only two columns ('key' and 'value'), with more complex information sometimes stored as BLOBs within the 'value' columns. Document databases do away with the table-and-row model altogether, storing all relevant data together in single 'document' in JSON, XML, or another format, which can nest values hierarchically.

* 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?**

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.

|  |  |
| --- | --- |
| Logical View of Customer Contact Information in HBase | |
| **Row Key** | **Column Family: {Column Qualifier:Version:Value}** |
| 00001 | CustomerName: {‘FN’: 1383859182496:‘John’, ‘LN’: 1383859182858:‘Smith’, ‘MN’: 1383859183001:’Timothy’, ‘MN’: 1383859182915:’T’} ContactInfo: {‘EA’: 1383859183030:‘John.Smith@xyz.com’, ’SA’: 1383859183073:’1 Hadoop Lane, NY 11111’} |
| 00002 | CustomerName: {‘FN’: 1383859183103:‘Jane’, ‘LN’: 1383859183163:‘Doe’, ContactInfo: { ’SA’: 1383859185577:’7 HBase Ave, CA 22222’} |

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.

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

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.

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

HBase tables are organized by column. Furthermore, the columns are organized in groups called *column families*. When creating an HBase table, you must define the column families before inserting any data. Column families should not be changed often, nor should there be too many of them, so it is important to think carefully about what column families will be useful for your particular data. Each column family, however, can contain a very large number of columns. Columns are named using the format family:qualifier.

Unlike columns in a relational database, which reserve empty space for columns with no values, HBase columns simply don't exist for rows where they have no values. This not only saves space, but means that different rows need not have the same columns; you can use whatever columns you need for your data on a per-row basis.

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

HBase is built on top of the distributed file system (DFS), which can store large files. HBase provides fast record lookups and updates for large tables. The ZooKeeper cluster acts as a coordination service for the entire HBase cluster.

HBase contains two primary services:

**Master server**

The master server co-ordinates the cluster and performs administrative operations, such as assigning regions and balancing the loads.

**Region server**

The region servers do the real work. A subset of the data of each table is handled by each region server. Clients talk to region servers to access data in HBase.

**Regions**

Region servers manage a set of regions.

An HBase table is made up of a set of regions. Regions are the basic unit of work in HBase. It is what is used as a split by the map reduce framework. The region contains store objects that correspond to column families. There is one store instance for each column family. Store objects create one or more *StoreFiles*, which are wrappers around the actual storage file called the *HFile*.

The region also contains a **MemStore**, which is in-memory storage and is used as a write cache. Rows are written to the **MemStore**. The data in the **MemStore** is ordered. If the **MemStore** becomes full, it is persisted to an HFile on disk

To improve performance, it is important to get an even distribution of data among regions, which ensures the best parallelism in map tasks.

**HFiles**

HFiles are the physical representation of data in HBase. Clients do not read HFiles directly but go through region servers to get to the data.

HBase internally puts the data in indexed *StoreFiles* that exist on HDFS for high-speed lookups.

Everything in HBase is stored as bytes and there are no types. There is no schema since each row in HBase can have a different set of columns.

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

**INSERT INTO table-name**

Specifies the HBase table into which data is to be inserted.

**columns**

Specifies the order and column names of those columns into which data is to be inserted. If no column names are specified, data is to be inserted into all columns that were listed, and in the order that was specified, when the named table was created.

**VALUES (expressions)**

*Expressions* supply values for every column. If a column list is specified, the expression list is evaluated to provide values for those columns. NULL is inserted for those columns that are omitted from the column list.

**DISABLE WAL**

Disables write-ahead logging for HBase **writes** and **puts**. Disabling write-ahead logging increases the performance of write operations, but it can result in data loss if the region servers fail. Your client application is responsible for ensuring data consistency when you use this option.