**Nosql Databases**

NoSQL means Not only SQL. NoSQL databases is modeled in a way that it can represent data other than tabular formats, which is maintained in relational databases. It uses different formats to represent data in databases and thus there are different types of NoSQL databases based on their representation format. Most of NoSQL databases leverages availability and speed over consistency. different types of There are many NoSQL databases and their representation formats.

**Types of Nosql Databases**

**Key-Value stores:** It is a schema-less database which contains keys and values. Each key point to a value which is an array of bytes, can be a string, BLOB, XML, etc. e.g. Lamborghini is a key and can point to a value Gallardo, Aventador, Murciélago, Reventón, Diablo, Huracán, Veneno, Centenario etc.

**Key-Value stores databases:** Aerospike, Couchbase, Dynamo, FairCom c-treeACE, FoundationDB, HyperDex, MemcacheDB, MUMPS, Oracle NoSQL Database, OrientDB, Redis, Riak, Berkeley DB.

**Document Oriented:** It follows the same key value pair, but it is semi structured like XML, JSON, BSON. These structures are considered as documents.

**Document Based databases:** Apache CouchDB, Clusterpoint, Couchbase, DocumentDB, HyperDex, IBM Domino, MarkLogic, MongoDB, OrientDB, Qizx, RethinkDB.

**Column Oriented:** In this database, data is stored in cell grouped in column rather than rows. Columns are logically grouped into column families which can be either created during schema definition or at runtime.

These types of databases store all the cell corresponding to a column as continuous disk entry, thus making the access and search much faster.

**Column Based Databases:** HBase, Accumulo, Cassandra, Druid, Vertica

**Graph Oriented:** It is a perfect flexible graphical representation, used unlike SQL. These types of databases easily solve address scalability problems as it contains edges and node which can be extended according to the requirements.

**Graph based databases:** AllegroGraph, ArangoDB, InfiniteGraph, Apache Giraph, MarkLogic, Neo4J, OrientDB, Virtuoso, Stardog

**CAP Theorem**

**CAP** stands for Consistency ,Availability ,Partition-tolerance

Consistency: every read would get you the most recent write.

Availability: every node (if not failed) always executes queries

Partition-tolerance: even if the connections between nodes are down, the other two (A & C) promises, are kept.

HBase maintains Consistency and Availability while Cassandra focuses on Availability and Partition -Tolerance.

**HBase Architecture**

HBase has three major components i.e., HMaster Server, HBase Region Server, Regions and Zookeeper.

**Region**

A region contains all the rows between the start key and the end key assigned to that region. HBase tables can be divided into a number of regions in such a way that all the columns of a column family is stored in one region. Each region contains the rows in a sorted order.

Many regions are assigned to a Region Server, which is responsible for handling, managing, executing reads and writes operations on that set of regions.

So, concluding in a simpler way:

A table can be divided into a number of regions. A Region is a sorted range of rows storing data between a start key and an end key.

A Region has a default size of 256MB which can be configured according to the need.

A Group of regions is served to the clients by a Region Server.

A Region Server can serve approximately 1000 regions to the client.

**HMaster**

HBase HMaster performs DDL operations (create and delete tables) and assigns regions to the Region servers.

It coordinates and manages the Region Server (similar as NameNode manages DataNode in HDFS).

It assigns regions to the Region Servers on startup and re-assigns regions to Region Servers during recovery and load balancing.

It monitors all the Region Server’s instances in the cluster (with the help of Zookeeper) and performs recovery activities whenever any Region Server is down.

It provides an interface for creating, deleting and updating tables.

## ****Meta Table****

The META table is a special HBase catalog table. It maintains a list of all the Regions Servers in the HBase storage system.

META file maintains the table in form of keys and values. Key represents the start key of the region and its id whereas the value contains the path of the Region Server

**Components of Region Server**

WAL: Write Ahead Log (WAL) is a file attached to every Region Server inside the distributed environment. The WAL stores the new data that hasn’t been persisted or committed to the permanent storage. It is used in case of failure to recover the data sets.

Block Cache: Block Cache resides in the top of Region Server. It stores the frequently read data in the memory. If the data in BlockCache is least recently used, then that data is removed from BlockCache.

MemStore: It is the write cache. It stores all the incoming data before committing it to the disk or permanent memory. There is one MemStore for each column family in a regionthere are multiple MemStores for a region because each region contains multiple column families. The data is sorted in lexicographical order before committing it to the disk.

HFile: you can see HFile is stored on HDFS. Thus it stores the actual cells on the disk. MemStore commits the data to HFile when the size of MemStore exceeds.

**HBase vs RDBMS**

|  |  |
| --- | --- |
| **Hbase** | **RDBMS** |
| Column oriented | Row oriented mostly |
| Flexible schema , add column on fly | Fixed schema |
| Good with sparse tables | Not optimized for sparse table |
| Good for un structured and semi structured data | Good for structured data |
| Horizontal scalability- just add hardware | Hard to shard and scale |
|  |  |
|  |  |