**NoSQL Databases**

A NoSQL (originally referring to "non SQL", "non relational" or "not only SQL"). It provides a mechanism for storage and retrieval of data which is modeled in means other than the tabular relations used in relational databases.

NoSQL is an approach to databases that represents a shift away from traditional relational database management systems (RDBMS) that rely on tables, columns, rows, or schemas to organize and retrieve data. NoSQL databases on the other hand do not rely on these structures and use more flexible data models. As RDBMS have increasingly failed to meet the performance, scalability, and flexibility needs that next-generation, data-intensive applications require, NoSQL databases have been adopted by mainstream enterprises. NoSQL is particularly useful for storing unstructured data, which is growing far more rapidly than structured data and does not fit the relational schemas of RDBMS.

Features of NoSQL:

Generic data model

Heterogeneous containers, including sets, maps, and arrays

Dynamic type discovery and conversion

NoSQL analytics systems support runtime type identification and conversion so that custom business logic can be used to dictate analytic treatment of variation.

Non-relational and De-normalised

Data is stored in single tables as compared to joining multiple tables.

Commodity hardware

Adding more of the economical servers allows NoSQL databases to scale to handle more data.

Highly distributable

Distributed databases can store and process a set of information on more than one device.

**Types of NoSQL Databases**

Document Oriented Databases (MongoDB): Stores documents as a data and allows indexing of documents on the basis of not only its primary identifier but also its properties

Graph Based Databases (Neo4j): A graph database uses graph structures with nodes, edges, and properties to represent and store data.

Column Based Databases (HBase): The column-oriented storage allows data to be stored effectively. It avoids consuming space when storing nulls by simply not storing a column when a value doesn’t exist for that column.

Key Value Databases (Membase): The key of a key/value pair is a unique value in the set and can be easily looked up to access the data.

**CAP Theorem**

*Consistency* - This means that the data in the database remains consistent after the execution of an operation. For example after an update operation, all clients see the same data.

*Availability* - This means that the system is always on (service guarantee availability), no downtime.

*Partition Tolerance* - This means that the system continues to function even if the communication among the servers is unreliable, i.e. the servers may be partitioned into multiple groups that cannot communicate with one another.

**HBase Architecture**

HBase is composed of three types of servers in a master slave type of architecture.

* Region servers serve data for reads and writes.
* HBase Master process handles the Region assignment, DDL (create, delete tables) operations
* Zookeeper maintains a live cluster state.

***Region:***

* HBase Tables are divided horizontally by row key range into “Regions.”
* A region contains all rows in the table between the region’s start key and end key.
* Regions are assigned to the nodes in the cluster, called “Region Servers,” and these serve
* data for reads and writes.
* A region server can serve about 1,000 regions.

***HBase Master***

* Region assignment, DDL (create, delete tables) operations are handled by the HBase Master.
* A master is responsible for:
  + Coordinating the region servers
  + Assigning regions on startup
  + Re-assigning regions for recovery or load balancing
  + Monitoring all RegionServer instances in the cluster (listens for notifications from zookeeper)
* Admin functions
  + Interface for creating, deleting, updating tables

***Zookeeper:***

* HBase uses ZooKeeper as a distributed coordination service to maintain server state in the cluster.
* Zookeeper maintains which servers are alive and available, and provides server failure notification.
* Zookeeper uses consensus to guarantee common shared state. Note that there should be three or five machines for consensus.

**HBase vs RDBMS**

1. Hadoop is not a database.  Hbase or Impala may be considered databases but Hadoop is just a file system (hdfs) with built in redundancy, parallelism.
2. Traditional databases/RDBMS have ACID properties - Atomicity, Consistency, Isolation and Durability.  You get none of these out of the box with Hadoop.  So if you have to for example write code to take money from one bank account and put into another one, you have to (painfully) code all the scenarios like what happens if money is taken out but a failure occurs before it’s moved into another account.
3. Hadoop offers massive scale in processing power and storage at a very low comparable cost to an RDBMS.
4. Hadoop offers tremendous parallel processing capabilities.  You can run jobs in parallel to crunch large volumes of data.
5. Some people argue that traditional databases do not work well with un-structured data, but it’s not as simple as that.  There are many applications built using traditional RDBMS that use a lot of unstructured data or video files or PDFs that I have come across that work well.
6. Typically RDBMS will manage a large chunk of the data in its cache for faster processing while at the same time maintaining read consistency across sessions.  I would argue Hadoop does a better job at using the memory cache to process the data without offering any other items like read consistency.
7. Hive SQL is almost always a magnitude of times slower than SQL you can run in traditional databases.  So if you are thinking SQL in Hive is faster than in a database, you are in for a sad disappointment.  It will not scale at all for complex analytics.
8. Hadoop is very good for parallel processing problems - like finding a set of keywords in a large set of documents (this operation can be parallelized).  However typically RDBMS implementations will be faster for comparable data sets.

In summary

|  |  |
| --- | --- |
| RDBMS | HBASE |
| Row-oriented | Column oriented |
| Fixed schema for the tables | No fixed schema |
| ACID properties followed | Guarantee consistency and partition tolerance |
| Uses SQL to query the database | Uses java client API and Jruby |