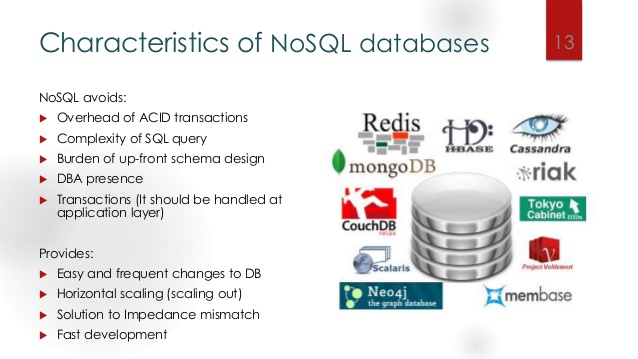
1. **Explain NoSQL Databases?**

* NoSQL database, also called Not Only SQL, is an approach to data management and database design that’s useful for very large set of distributed data. This database system is non-relational, distributed, open-source and horizontally scalable.
* NoSQL, which encompasses a wide range of technologies and architectures, seeks to solve the scalability and big data performance issues that relational databases weren’t designed to address.
* NoSQL does not prohibit SQL. Some NoSQL systems are entirely non-relational, others simply avoid selected relational functionality such as fixed table schemas and join operations.

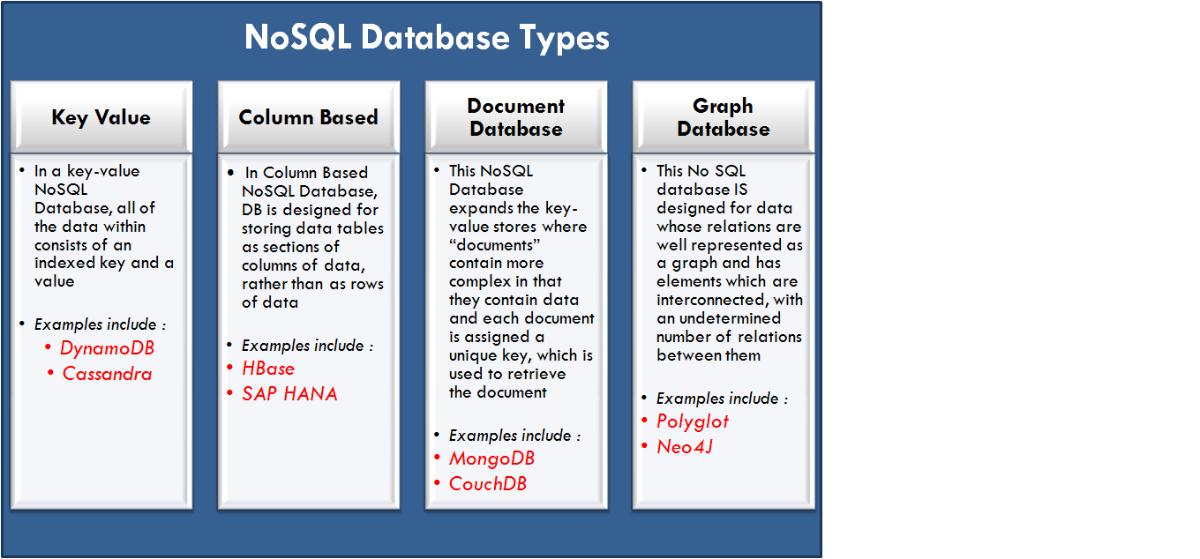
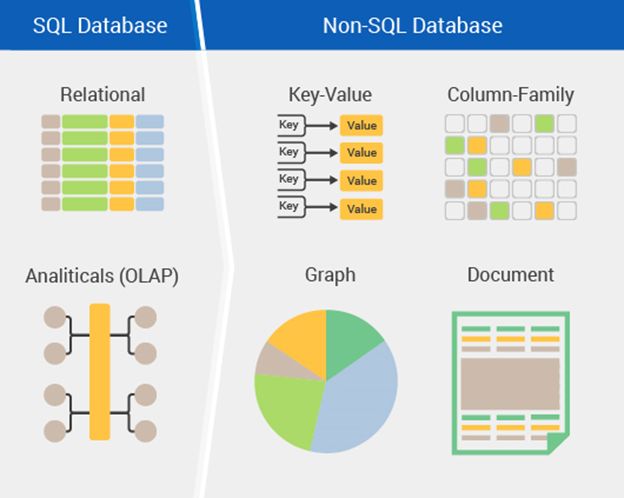
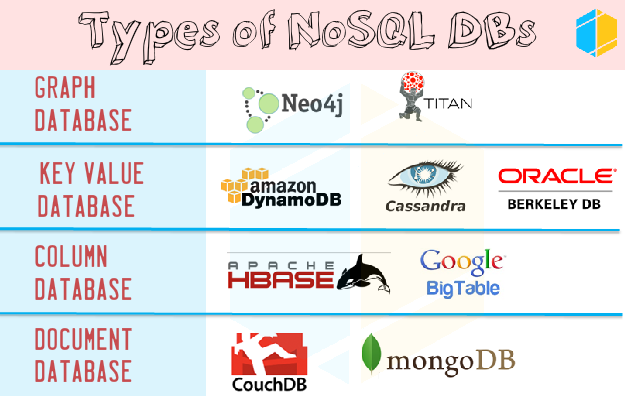
**Features of NoSQL Databases:**

* **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.

**Examples of NoSQL Databases are: HBase, Cassandra, mongoDB etc.**



1. **Explain Types of NoSQL Databases?**

****

1. **Explain CAP Theorem**

CAP stands for:

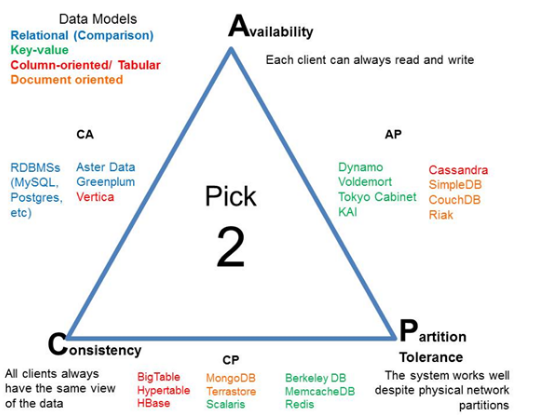
**Consistency** - This means that the data in the database remains consistent after the execution of an operation. For example after an update operation, all clients see the same data.

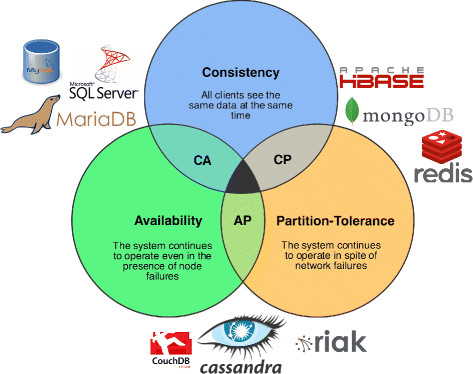
**Availability** - This means that the system is always on (service guarantee availability), no downtime.

**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.

In other words, the CAP theorem states that in the presence of a network partition, one has to choose between consistency and availability. Note that consistency as defined in the CAP theorem is quite different from the consistency guaranteed in [ACID](https://en.wikipedia.org/wiki/ACID) [database transactions](https://en.wikipedia.org/wiki/Database_transaction).

* No distributed system is safe from network failures, thus network partitioning generally has to be tolerated. In the presence of a partition, one is then left with two options: consistency or availability.
* When choosing consistency over availability, the system will return an error or a time-out if particular information cannot be guaranteed to be up to date due to network partitioning.
* When choosing availability over consistency, the system will always process the query and try to return the most recent available version of the information, even if it cannot guarantee it is up to date due to network partitioning.
* In the absence of network failure – that is, when the distributed system is running normally – both availability and consistency can be satisfied.
* CAP is frequently misunderstood as if one had to choose to abandon one of the three guarantees at all times. In fact, the choice is really between consistency and availability only when a network partition or failure happens; at all other times, no trade-off has to be made.
* Database systems designed with traditional ACID guarantees in mind such as RDBMS choose consistency over availability, whereas systems designed around the BASE philosophy, common in the NoSQL movement for example, choose availability over consistency.
* The PACELC theorem builds on CAP by stating that even in the absence of partitioning, another trade-off between latency and consistency occurs.

****

****

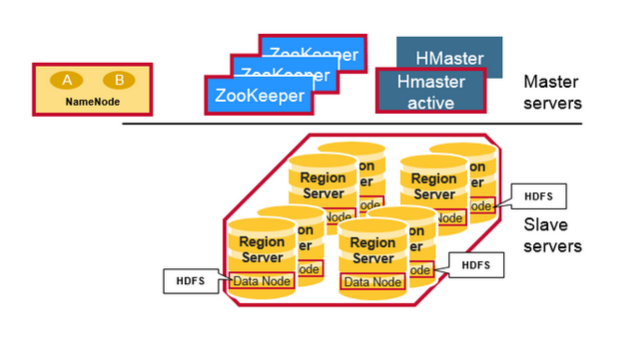
1. **Explain HBase Architecture**

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

* Region servers serve data for reads and writes. When accessing data, clients communicate with HBase RegionServers directly.
* HBase Master handles Region assignment, DDL (create, delete tables) operations.
* Zookeeper, which is part of HDFS, maintains a live cluster state.

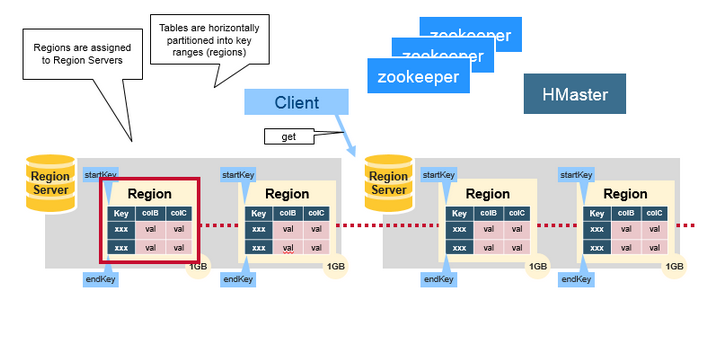
The Hadoop DataNode stores the data that the Region Server is managing. All HBase data is stored in HDFS files. Region Servers are collocated with the HDFS DataNodes, which enable data locality (putting the data close to where it is needed) for the data served by the RegionServers. HBase data is local when it is written, but when a region is moved, it is not local until compaction.

The NameNode maintains metadata information for all the physical data blocks that comprise the files.

****

**Regions**

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.

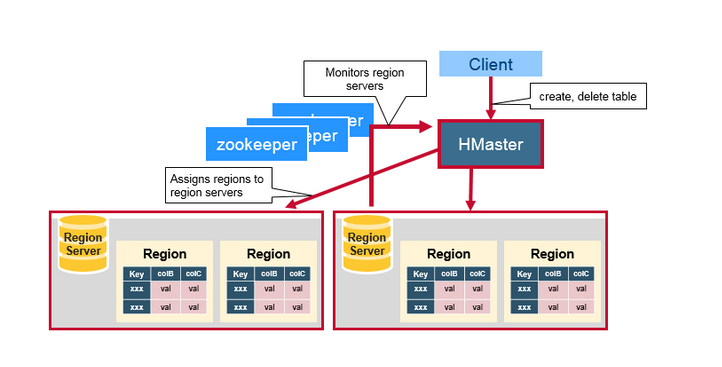


**HBase HMaster**

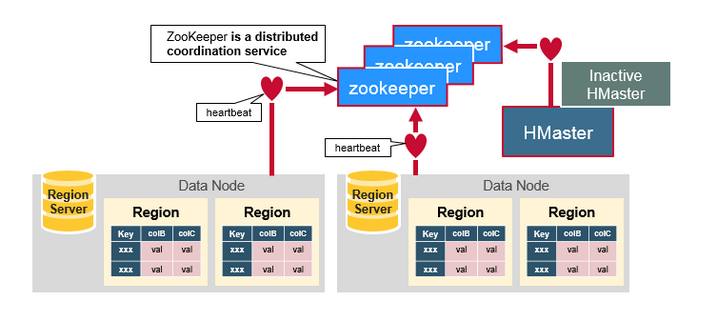
Region assignment, DDL (create, delete tables) operations are handled by the HBase Master.

A master is responsible for:

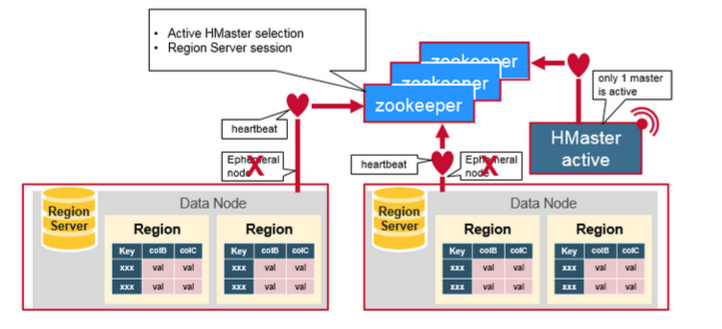
* Coordinating the region servers
  + Assigning regions on startup , re-assigning regions for recovery or load balancing
  + Monitoring all RegionServer instances in the cluster (listens for notifications from zookeeper)
* Admin functions
  + Interface for creating, deleting, updating tables



## ZooKeeper: The Coordinator

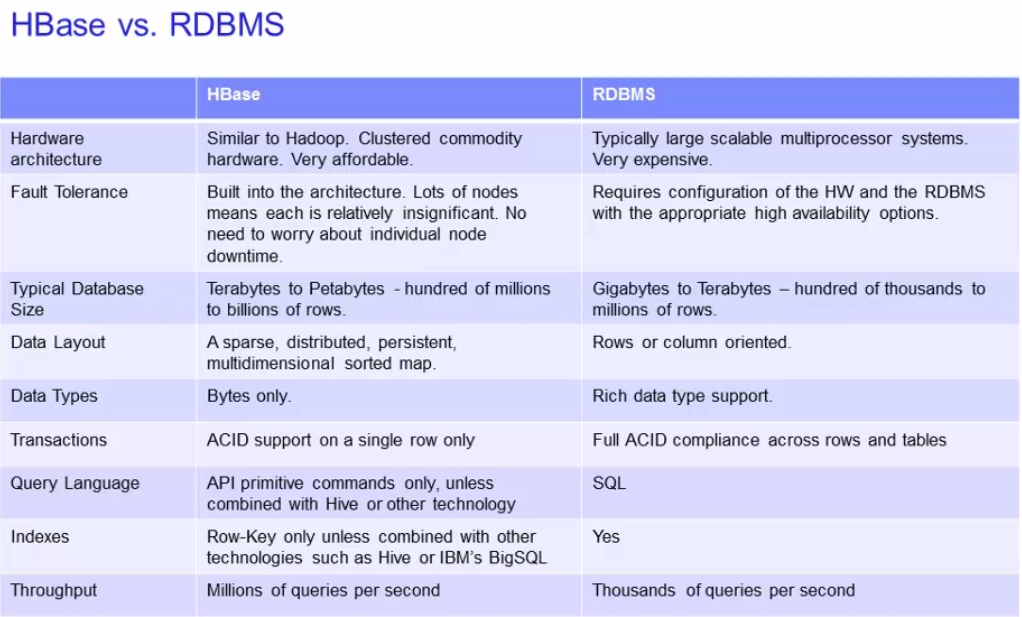
HBase uses ZooKeeper as a distributed coordination service to maintain server state in the cluster. Zookeeper maintains which servers are alive and available, and provides server failure notification. Zookeeper uses consensus to guarantee common shared state. Note that there should be three or five machines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

## How do the Components Work Together?

Zookeeper is used to coordinate shared state information for members of distributed systems. Region servers and the active HMaster connect with a session to ZooKeeper. The ZooKeeper maintains ephemeral nodes for active sessions via heartbeats.

Each Region Server creates an ephemeral node. The HMaster monitors these nodes to discover available region servers, and it also monitors these nodes for server failures. HMasters vie to create an ephemeral node. Zookeeper determines the first one and uses it to make sure that only one master is active. The active HMaster sends heartbeats to Zookeeper, and the inactive HMaster listens for notifications of the active HMaster failure.

If a region server or the active HMaster fails to send a heartbeat, the session is expired and the corresponding ephemeral node is deleted. Listeners for updates will be notified of the deleted nodes. The active HMaster listens for region servers, and will recover region servers on failure. The Inactive HMaster listens for active HMaster failure, and if an active HMaster fails, the inactive HMaster becomes active.

1. **HBASE vs. RDBMS**

