# Assignment 9.3

**Problem Statement**

**Explain the below concepts with an example in brief.**

- **Nosql Databases**
- **Types of Nosql Databases**
- **CAP Theorem**
- **HBase Architecture**
- **HBase vs RDBMS**

**Solution :**

- **Nosql Databases**

A NoSQL database provides a mechanism for storage and retrieval of data that is modeled by means other than the tabular relations used in relational databases. NoSQL means Not Only SQL, implying that when designing a software solution or product, there are more than one storage mechanism that could be used based on the needs. NoSQL databases emerged as a result of the exponential growth of the Internet and the rise of web applications. Traditional RDBMS has some limitation like scaling-up is expensive and they have rigid data models. NoSQL databases are increasingly used in big data and real-time web applications

Application developers have been frustrated with the impedance mismatch between the relational data structures and the in-memory data structures of the application. Using NoSQL databases allows developers to develop without having to convert in-memory structures to relational structures.

There is also movement away from using databases as integration points in favor of encapsulating databases with applications and integrating using services.

The rise of the web as a platform also created a vital factor change in data storage as the need to support large volumes of data by running on clusters.

Relational databases were not designed to run efficiently on clusters.

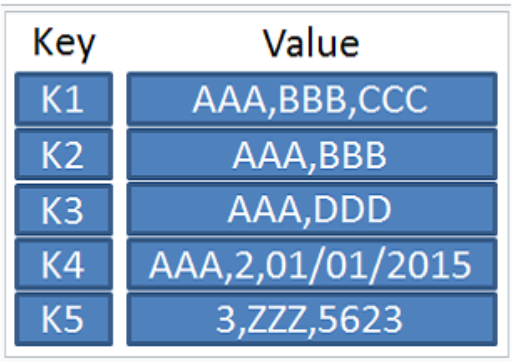
**Main Characteristics:**

- Schema free
- Easy replication support
- Simple API
- Eventually Consistent
- Huge Amount data

- ## Types of Nosql Databases

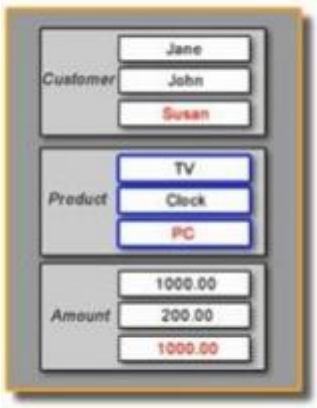
Several different varieties of NoSQL databases have been created to support specific needs and use cases. These fall into four main categories:

**Key-value data stores:** Key-value NoSQL databases emphasize simplicity and are very useful in accelerating an application to support high-speed read and write processing of non-transactional data. Stored values can be any type of binary object (text, video, JSON document, etc.) and are accessed via a key. The application has complete control over what is stored in the value, making this the most flexible NoSQL model. Data is partitioned and replicated across a cluster to get scalability and availability. For this reason, key value stores often do not support transactions. However, they are highly effective at scaling applications that deal with high-velocity, non-transactional data.

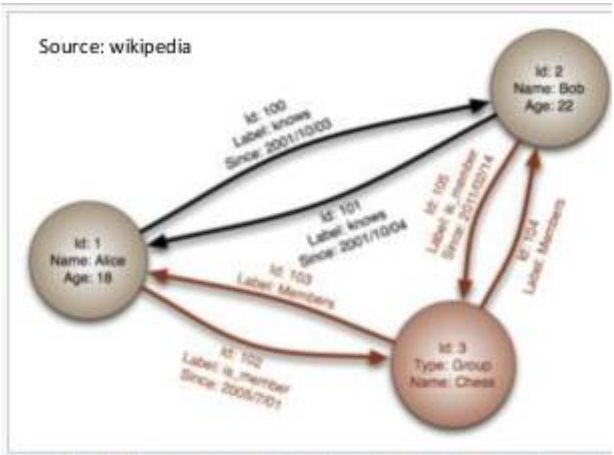



**Document stores:** Document databases typically store self-describing JSON, XML, and BSON documents. They are similar to key-value stores, but in this case, a value is a single document that stores all data related to a specific key. Popular fields in the document can be indexed to provide fast retrieval without knowing the key. Each document can have the same or a different structure.

**Wide-column stores:** Wide-column NoSQL databases store data in tables with rows and columns similar to RDBMS, but names and formats of columns can vary from row to row across the table. Wide-column databases group columns of related data together. A query can retrieve related data in a single operation because only the columns associated with the query are retrieved. In an RDBMS, the data would be in different rows stored in different places on disk, requiring multiple disk operations for retrieval.




**Graph stores:** A graph database uses graph structures to store, map, and query relationships. They provide index-free adjacency, so that adjacent elements are linked together without using an index. Neo4j is one of the leading graph databases.



Graph databases employ nodes, properties, and edges.

- **<u>CAP Theorem</u>**
  The CAP theorem, also named Brewer's theorem after computer scientist Eric Brewer, states that it is impossible for a distributed data store to simultaneously provide more than two out of the following three guarantees.

**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.

**case 1:**

Duplicate Copy of same data is maintained on Multiple Machines.
This increases availability, but decreases consistency.

**case 2:**

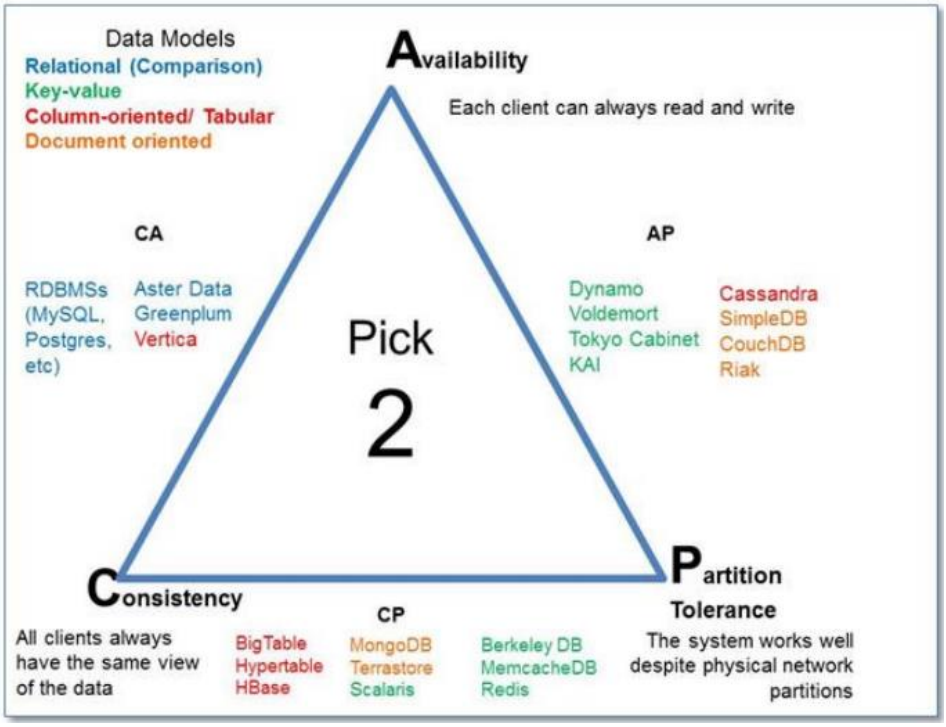
If data on one machine changes, the update propagates to the other Machine, system is inconsistent, but will become eventually consistent.

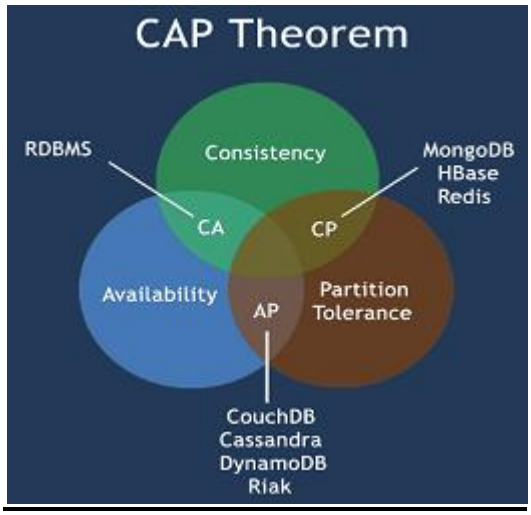
**case 3:**

- If duplicate copy of same data is not maintained, consistency is superior
- But availability decreases.

**case 4:**

If data on one machine changes, the update propagates to the other Machine, system is inconsistent, but will become eventually consistent.




● **HBase Architecture**

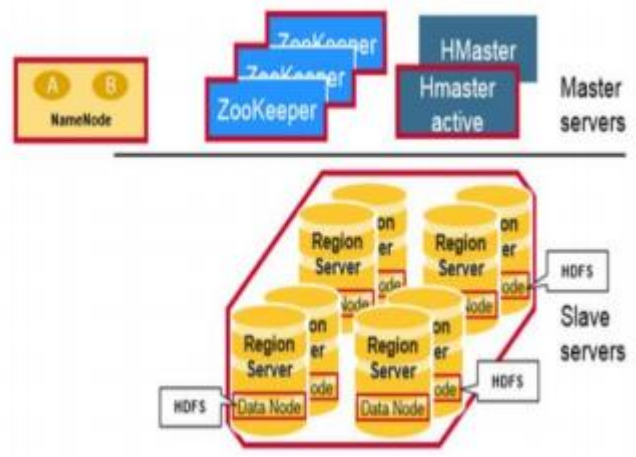
HBase is composed of three types of servers in a master slave type of architecture.

• Region servers serve data for reads and writes.

• HBaseMaster process handles the Region assignment, DDL (create, delete tables) operations

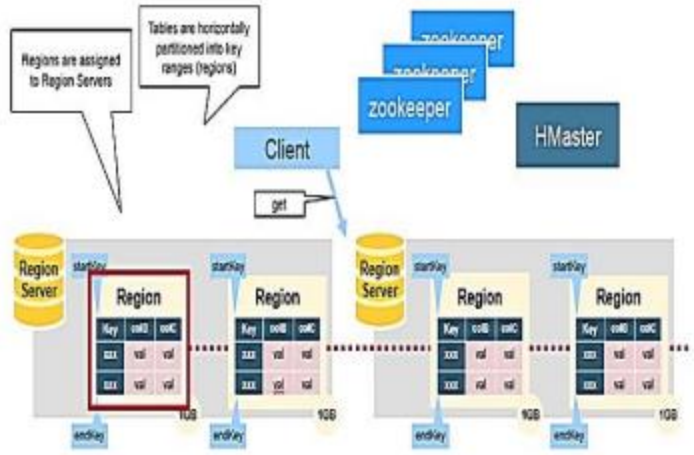
• Zookeeper maintains a live cluster state.

The Hadoop DataNode stores the data that the Region Server is managing. All HBase data is stored in HDFS files.The NameNode maintains metadata information for all the physical data blocks that comprise the files.




**Regions**

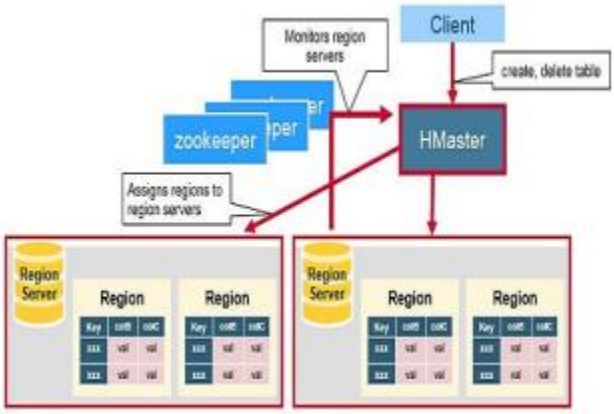
HBase Tables are divided horizontally by row key range into “Regions.” A region contains all rows in the table between the region’s start key and end key.• Regions are assigned to the nodes in the cluster, called “Region Servers,” and these serve data for reads and writes. A region server can serve about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




**HBaseHMaster :**Region assignment, DDL (create, delete tables) operations are handled by the HBase Master.A master is responsible for:

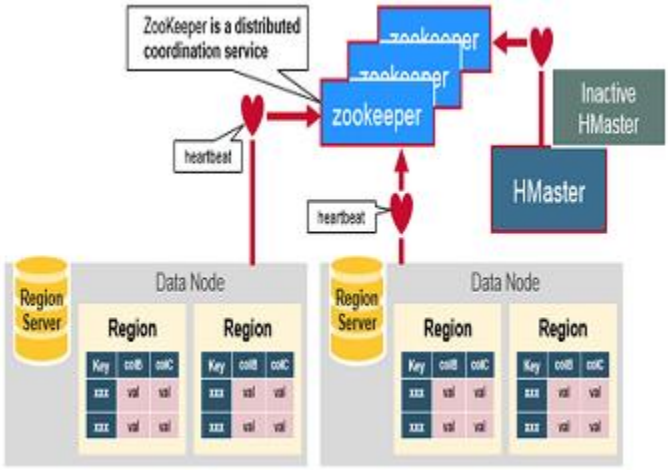
- Coordinating the region servers
- Assigning regions on startup
- Re-assigning regions for recovery or load balancing
- Monitoring all RegionServer instances in the cluster (listens for notifications from zookeeper)

- Interface for creating, deleting, updating tables




**ZooKeeper: The Coordinator**

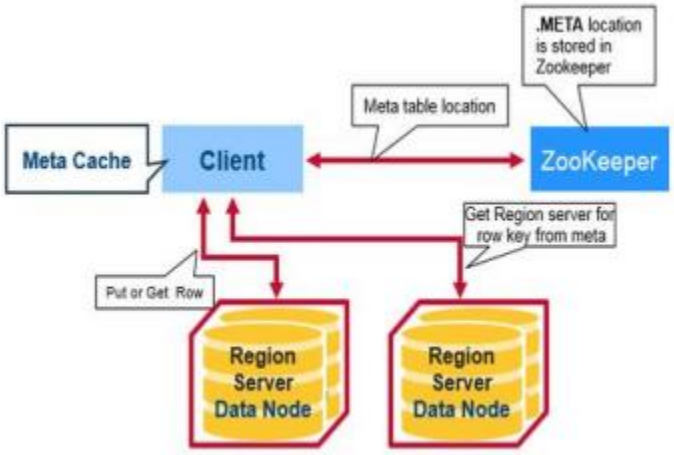
HBase uses ZooKeeper as a distributed coordination service to maintain server state in the cluster. Zookeeper maintains which servers are alive and available, and provides server failure notification. Zookeeper uses consensus to guarantee common shared state. Note that there should be three or five machines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




**HBaseFirst Read**

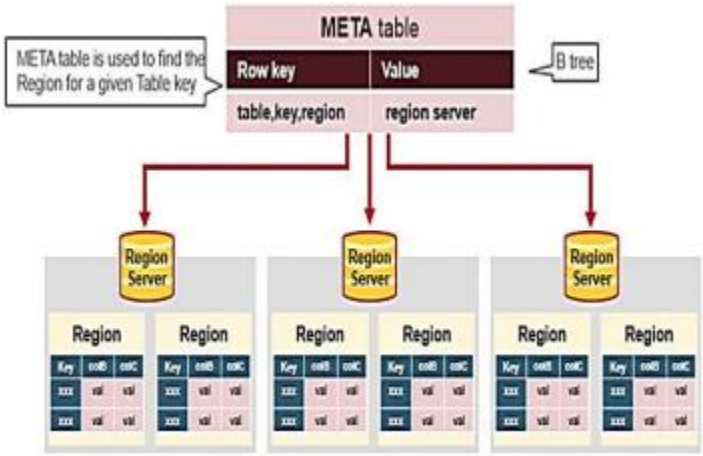
There is a special HBaseCatalog table called the META table, which holds the location of the regions in the cluster. ZooKeeper stores the location of the META table. The client gets the Region server that hosts the META table from ZooKeeper. The client will query the .META. server to get the region server corresponding to the row key it wants to access. The client caches this information along with the META table location. It will get the row from the corresponding Region Server.
For future reads, the client uses the cache to retrieve the META location and previously read row keys. Over time, it does not need to query the META table,

unless there is a miss because a region has moved; then it will re-query and update the cache.




**HBaseMeta Table**

This META table is an HBase table that keeps a list of all regions in the system.
The .META. table is like a b tree.The .META. table structure is as follows:
Key: region start key, region id
Values: RegionServer




**Region Server Components**

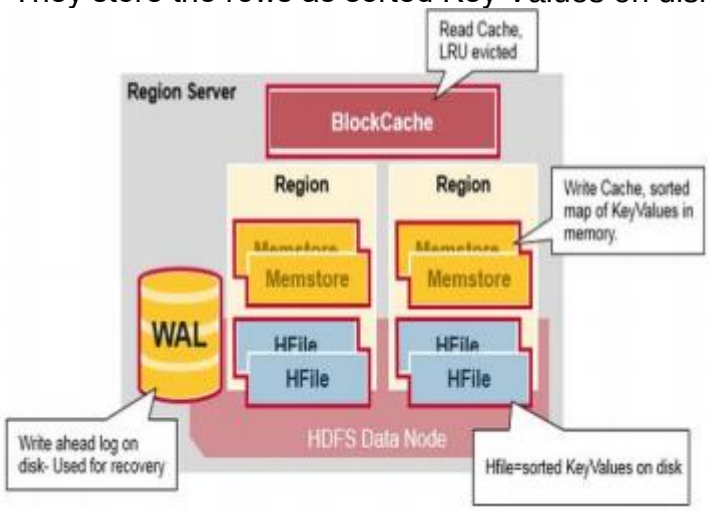
Region Server runs on an HDFS data node and has the following components:

**WAL**
• Write Ahead Log is a file on the distributed file system. The WAL is used to store new data that hasn't yet been persisted to permanent storage; it is used for recovery in the case of failure.

**BlockCache**
• It is the read cache. It stores frequently read data in memory. Least Recently Used data is evicted when full.

**MemStore**
• It is the write cache. It stores new data which has not yet been written to disk. It is sorted before writing to disk. There is one MemStoreper column family per region.

**Hfiles**
• They store the rows as sorted Key Values on disk.




## ● HBase vs RDBMS

| | HBASE | RDBMS |
|---|---|---|
| 1 | HBase is a distributed, column-oriented data storage system | RDBMS is row-oriented databases |
| 2 | Hbase tables guarantee consistency and partition tolerance | RDBMS tables guarantee ACID properties |
| 3 | Hbase tables do not have fixed-schema add column on the fly | RDBMS tables have fixed-schema |
| 4 | Hbase uses Java client API and Jruby | RDBMS uses SQL (Structured query Langauge) to query the data |
| 5 | Joins using MapReduce-not optimized | Optimized for Joins |
| 6 | Tight Integration with MR | No integration |
| 7 | Good for Semi-structured data as well as Un-structured data | Good for Structured Data |