ASSIGNMENT 9.3

Explain the below concepts with an example in brief.

Nosql Databases:

A **NoSQL** (originally referring to "non SQL" or "non relational") database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.NoSQL databases are increasingly used in big data and real-time web applications, NoSQL systems are also sometimes called "Not only SQL" to emphasize that they may support SQL-like query languages.

Motivations for this approach include: simplicity of design, simpler "horizontal" scaling to clusters of machines (which is a problem for relational databases), and finer control over availability. The data structures used by NoSQL databases (e.g. key-value, wide column, graph, or document) are different from those used by default in relational databases, making some operations faster in NoSQL. The particular suitability of a given NoSQL database depends on the problem it must solve. Sometimes the data structures used by NoSQL databases are also viewed as "more flexible" than relational database tables.

Type of Nosql Databases:

There are 4 basic types of NoSQL databases:

1. **Key-Value Store** – It has a Big Hash Table of keys & values

   {Example- Riak, Amazon S3 (Dynamo)}
2. **Document-based Store- It** stores documents made up of tagged elements. {Example- CouchDB}
3. **Column-based Store-** Each storage block contains data from only one column,

   {Example- HBase, Cassandra}
4. **Graph-based**-A network database that uses edges and nodes to represent and store data.

   {Example- Neo4J}

Cap Theorem:

[C] Consistency - All nodes see the same data at the same time.

Simply put, performing a *read* operation will return the value of the most recent *write* operation causing all nodes to return the same data. A system has consistency if a transaction starts with the system in a consistent state, and ends with the system in a consistent state. In this model, a system can (and does) shift into an inconsistent state during a transaction, but the entire transaction gets rolled back if there is an error during any stage in the process.

Typical relational databases are consistent: SQL Server, MySQL, and PostgreSQL.

[A] Availability - Every request gets a response on success/failure.

Achieving availability in a distributed system requires that the system remains operational 100% of the time. Every client gets a response, regardless of the state of any individual node in the system. This metric is trivial to measure: either you can submit read/write commands, or you cannot.

Typical relational databases are also available: SQL Server, MySQL, and PostgreSQL. This means that relational databases exist in the **CA** space - consistency and availability. However, CA is not only reserved for relational databases - some document-oriented tools like ElasticSearch also fall under the CA umbrella.

[P] Partition Tolerance - System continues to work despite message loss or partial failure.

Most people think of their data store as a single node in the network. *“This is our production SQL Server instance”*. Anyone who has run a production instance for more than four minutes, quickly realizes that this creates a *single point of failure*. A system that is partition-tolerant can sustain any amount of network failure that doesn’t result in a failure of the entire network. Data records are sufficiently replicated across combinations of nodes and networks to keep the system up through intermittent outages.

Storage systems that fall under Partition Tolerance with Consistency (CP): MongoDB, Redis, AppFabric Caching, and MemcacheDB. CP systems make for excellent distributed caches since every client gets the same data, and the system is partitioned across network boundaries.

Storage systems that fall under Partition Tolerance with Availability (AP) include DynamoDB, CouchDB, and Cassandra.

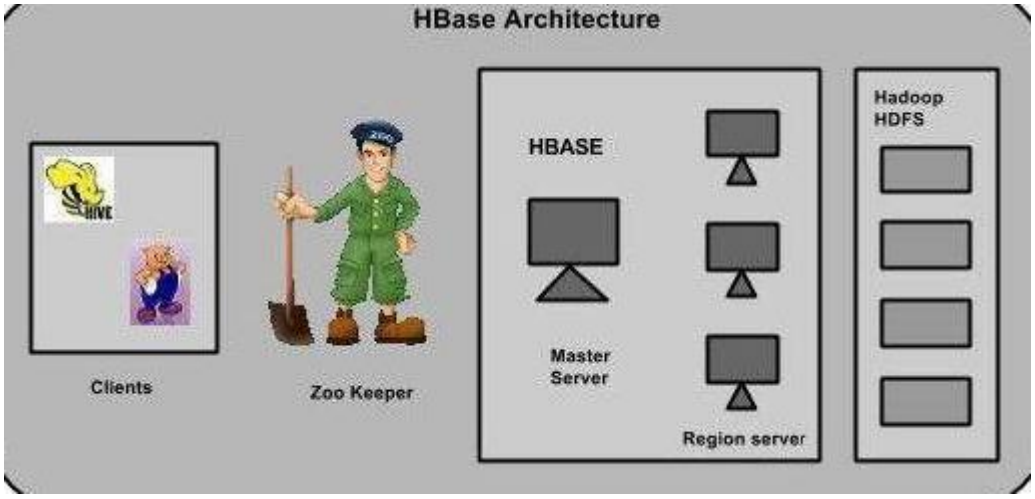
HBase Vs RDBMS:

HBase Vs RDBMS

HBase	RDBMS
A Schema-less ,i.e there is no predefined schema for HBase tables.	RDBMS Tables have Fixed schema, which describes the whole structure of tables.
Built for wide tables. It is horizontally scalable.	It is thin and built for small tables. Hard to scale.
No transactions are there in HBase.	RDBMS is transactional.
It has de-normalized data.	It will have normalized data.
It is good for semi-structured as well as structured data.	It is good for structured data.

HBase Architecture:

In HBase, tables are split into regions and are served by the region servers. Regions are vertically divided by column families into “Stores”. Stores are saved as files in HDFS. Shown below is the architecture of HBase.



HBase has three major components: the client library, a master server, and region servers. Region servers can be added or removed as per requirement.

MasterServer

The master server -

- Assigns regions to the region servers and takes the help of Apache ZooKeeper for this task.

- Handles load balancing of the regions across region servers. It unloads the busy servers and shifts the regions to less occupied servers.
- Maintains the state of the cluster by negotiating the load balancing.
- Is responsible for schema changes and other metadata operations such as creation of tables and column families.

Regions

Regions are nothing but tables that are split up and spread across the region servers.

Region server

The region servers have regions that -

- Communicate with the client and handle data-related operations.
- Handle read and write requests for all the regions under it.
- Decide the size of the region by following the region size thresholds.

The store contains memory store and HFiles. Memstore is just like a cache memory. Anything that is entered into the HBase is stored here initially. Later, the data is transferred and saved in Hfiles as blocks and the memstore is flushed.

Zookeeper

- Zookeeper is an open-source project that provides services like maintaining configuration information, naming, providing distributed synchronization, etc.
- Zookeeper has ephemeral nodes representing different region servers. Master servers use these nodes to discover available servers.
- In addition to availability, the nodes are also used to track server failures or network partitions.
- Clients communicate with region servers via zookeeper.
- In pseudo and standalone modes, HBase itself will take care of zookeeper.