ASSIGNMENT 9.3

Problem Statement:

Explain the below concepts with an example in brief.

1. NoSQL Databases
2. Types of NoSQL Databases
3. CAP Theorem
4. HBase Architecture
5. HBase vs RDBMS

Solution:

1. NoSQL Databases:

NoSQL means ‘not SQL’ or ‘not only SQL’. Relational database systems which use SQL are implemented on concepts such as tables or schemas, rows and columns in an organized manner. The problem which such system is that it fails to offer performance, scalability as well as flexibility when input data has no proper structure and size of data is enormous.

NoSQL databases are introduced to overcome these problems. They are specifically designed for ingesting unstructured data and retrieving the same in faster manner than relational database systems.

NoSQL databases are designed to store non-relational data and to provide good scaling options to meet ever growing incoming data requirements. They provide developers many features to choose from and fine tune the system according to their requirements specifically. An important part of a platform or software development is understanding the requirements of how data flows around the system, how it is consumed by the system, which consistency model is of more priority, etc. Relational database systems comes with a standard group of features and there is no provision for flexibility of choosing one feature over the other. Unlike to this, NoSQL databases offer wide variety of choices in this perspective.

For example, **HBase has a schema-less architecture** which allows users not to stick to a fixed column structure and is suitable for datasets in which some columns have blank/null values at times.

2. **Types of NoSQL Databases:**

NoSQL databases can be broadly classified into four types:

- **Wide-column stores:** These type of databases group columns of related data together. They classify data of tables as columns rather than rows which is the case in RDBMS. Wide-column stores are mainly designed query and retrieve data from large volume of data quicker than traditional database systems. Few examples of such databases are Google's BigTable, HBase and Cassandra.

  These type databases are vertically scalable i.e. they define data in terms of column families. A column is a basic unit of wide-column database and a column family is a collection of columns and we can define any number of column families on a table. Each row is distinguished by a unique field termed as row key or row id. Let's see an example:

Row ID	Website		
1	protocol	domain	subdomain
	http://	acadgild.com	www

- **Key-value stores:** These kind of stores offer a simple and easy to use data model that associates a distinct key with a value. Since it is quite simple, it gives high performance and scalability when data size grows larger. Redis, Berkeley DB, Aerospike, MemcacheDB and Riak are few examples of key-value databases. Key-value databases store values in a pair consisting of a key and its corresponding value. Here, each key of a specific table or entity should be unique so as to distinguish between all the other keys that exist in that table; values can be repetitive or distinct based on user data.

  Example:

  Cricket Match Score Card:

  Key Value

  Rohit Sharma 56
  Shikhar Dhavan 18
  Virat Kohli 114

M S Dhoni	44
Yuvraj Singh	75

- **Document databases:** Document databases usually store semi-structured data such as JSON (Java Script Object Notation), XML (eXtended Markup Language) and BSON (Binary JSON). Document stores are typically used in content management and mobile application data handling. Examples include MongoDB, DocumentDB, CouchDB and MarkLogic. Document stores contain data in the form of JSON, XML or BSON, etc. Let's take an example of MongoDB which stores data in the form of JSON.

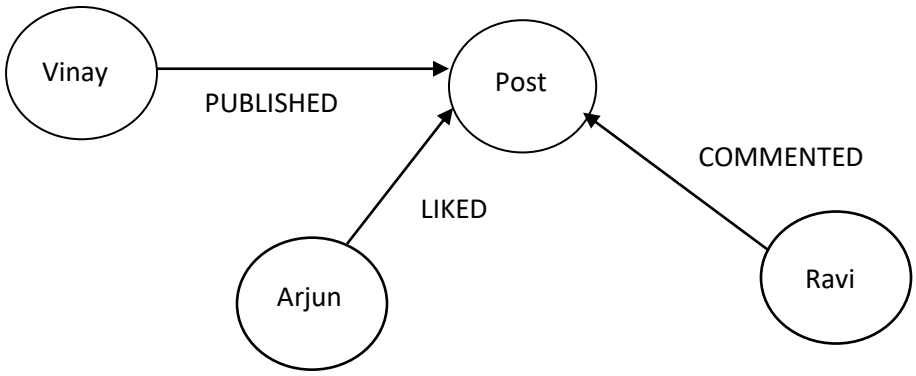
Students Test Scores:

```
{
    "name": "Ajay",
    "class": 10,
    "subjects": ["English", "Science", "Mathematics"],
    "marks": [78, 85, 96]
}
```

In the above example, the fields named 'subjects' and 'marks' hold a collection of values of the same type.

- **Graph databases:** Graph databases organize data as nodes, similar to rows in relational databases, and edges represents connections between nodes. Graph databases can evolve over time and usage and users can visualize data in more easier and interactive manner. Some examples of such databases are Neo4j, Titan and IBM Graph.

  In this type data stores, all data of a specific row will be represented as a single node and this node can be connected to another by specifying a relationship between them which is termed as edge. This is mostly useful in depicting social network related data using databases such as Ne04j or Titan. Let's see an example:



3. CAP Theorem:

CAP theorem states that, in a distributed system which is a group of interconnected nodes that communicate and exchange data, we can get only guarantee on two out of three factors:

1. Consistency – It guarantees that the data remains consistent in the database after the execution of an operation. For example, after successful update of some data in a database, all clients or users get the same data.
2. Availability – It guarantees that the data store or the nodes that contain data remain accessible and operational at all times. Every node must be able to complete user requests and respond in a reasonable amount of time to satisfy this criteria.
3. Partition Tolerance: It guarantees that the database will continue to function as expected in spite of partitions in the network. The servers may get partitioned into several groups that can communicate with one another and the system has to guarantee that it will function as intended and provide correct results all the time.

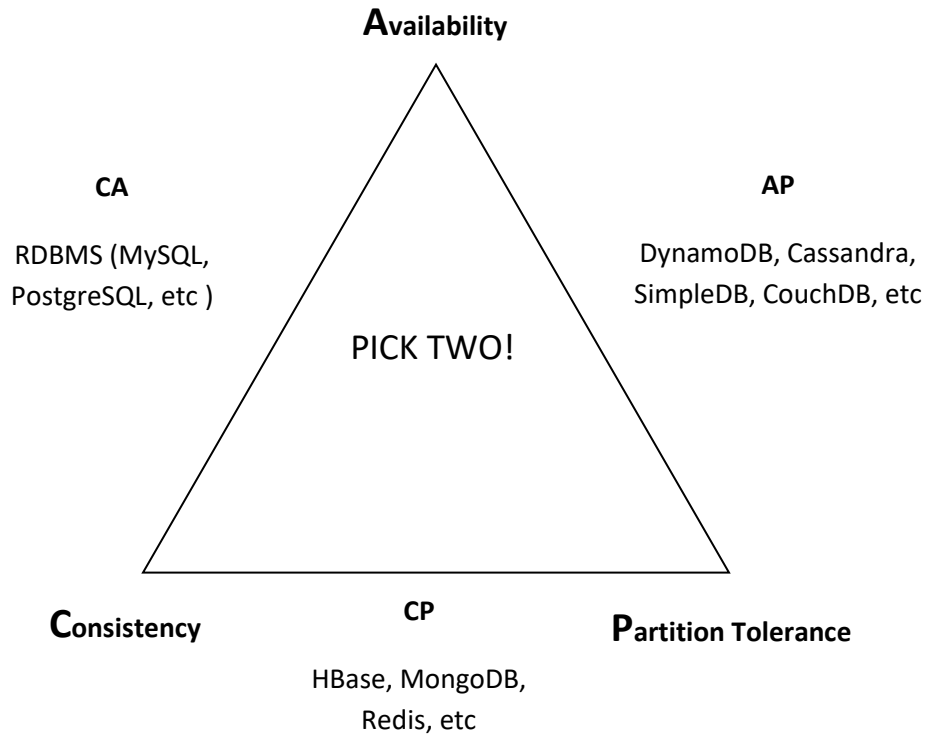


Figure: CAP theorem diagram

The C and A in ACID represent different concepts than C and in A in the CAP theorem.

The CAP theorem categorizes systems into three categories:

- CP (Consistent and Partition Tolerant) - At first glance, the CP category is confusing, i.e., a system that is consistent and partition tolerant but never available. CP is referring to a category of systems where availability is sacrificed only in the case of a network partition.
- CA (Consistent and Available) - CA systems are consistent and available systems in the absence of any network partition. Often a single node's DB servers are categorized as CA systems. Single node DB servers do not need to deal with partition tolerance and are thus considered CA systems. The only hole in this theory is that single node DB systems are not a network of shared data systems and thus do not fall under the preview of CAP.
- AP (Available and Partition Tolerant) - These are systems that are available and partition tolerant but cannot guarantee consistency.

4. HBase architecture:

Apache HBase is an open-source, distributed, versioned, non-relational database.

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 handles Region assignment, DDL (create, alter and drop table) operations.
- Zookeeper maintains a live cluster state.

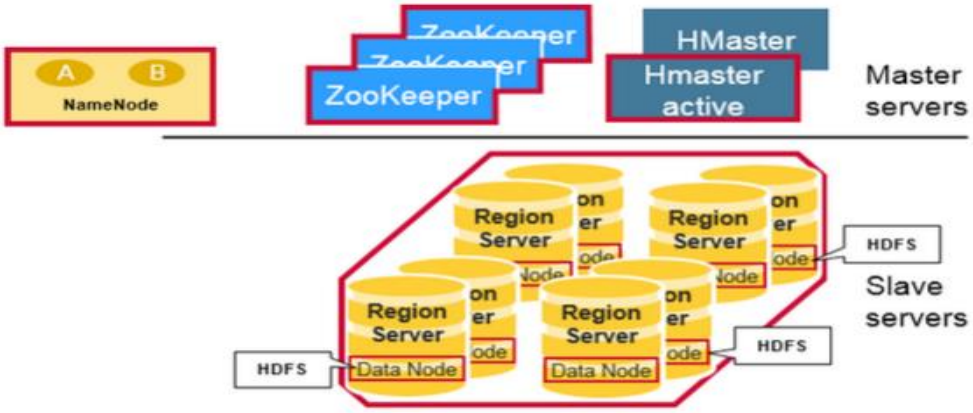
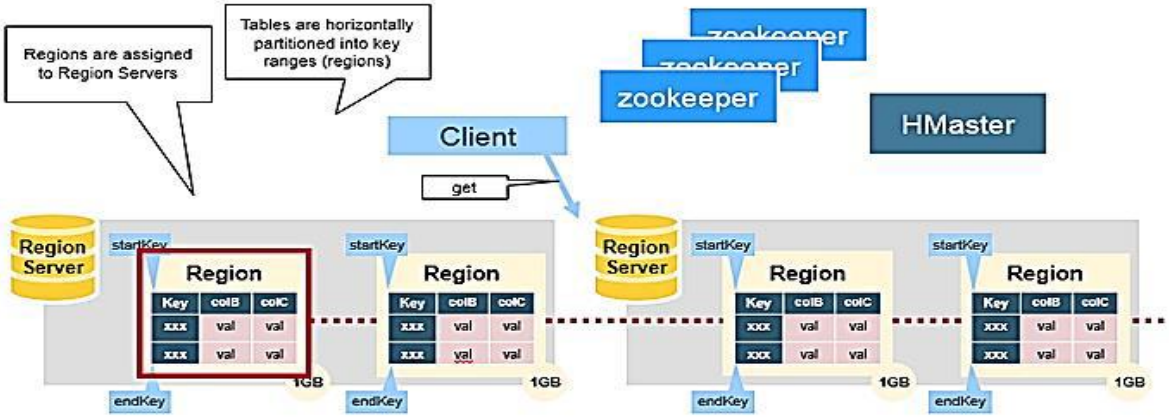


Figure: HBase Architecture

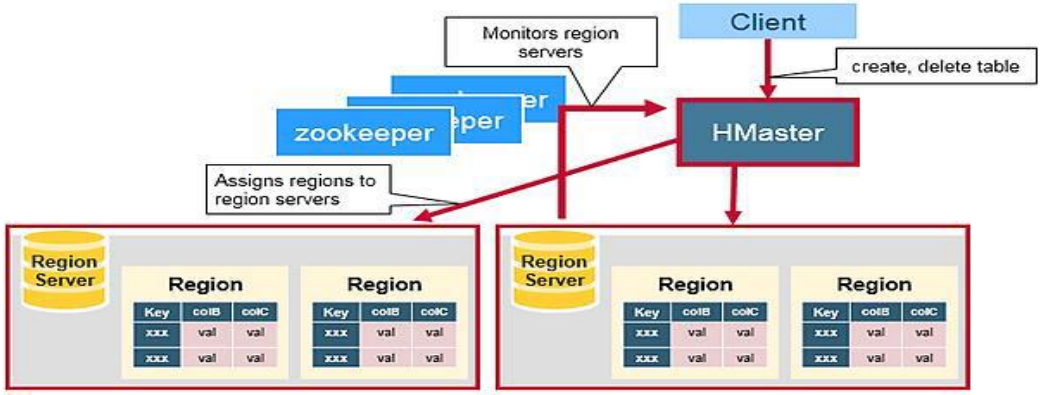
The Hadoop DataNode stores the data that the Region Server is managing. All HBase data is stored in HDFS files. The NameNode maintains metadata information for all the physical data blocks that comprise the files.

The simplest and foundational unit of horizontal scalability in HBase is a **Region**. A continuous, sorted set of rows that are stored together is referred to as a region (subset of table data). HBase architecture has a single **HBase master node** (HMaster) and several slaves i.e. **region servers**. Each region server (slave) serves a set of regions, and a region can be served only by a single region server. Whenever a client sends a write request, HMaster receives the request and forwards it to the corresponding region server.

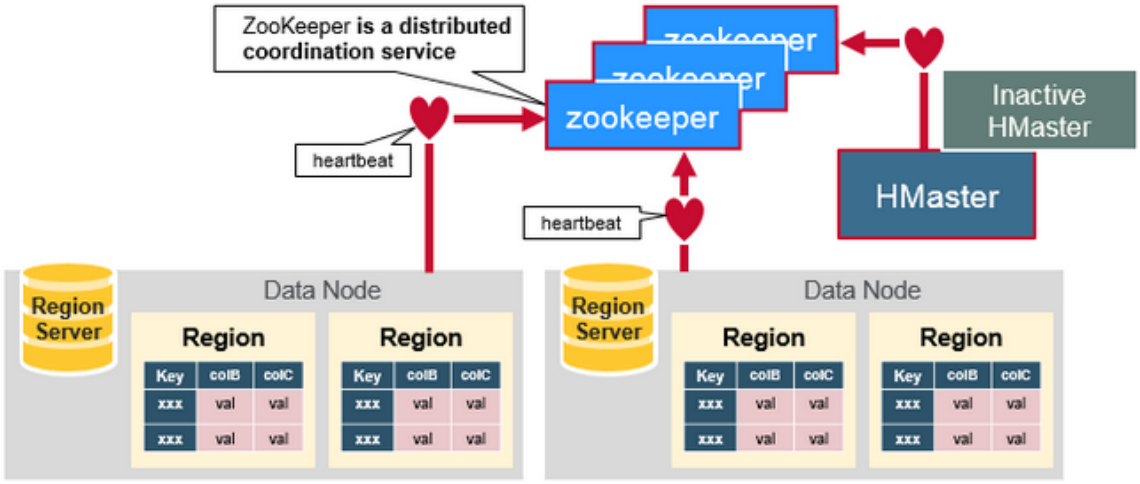
Regions: HBase tables are divided horizontally by row key range into 'Regions'. A region contains all rows in the table between the region's start key and end key.



HBase HMaster: HMaster handles region assignment, DDL operations. It is responsible for coordinating the region servers, assigning regions on startup, reassigning in case of regions recovery and monitoring all RegionServer instances in the cluster.



Zookeeper: HBase uses Zookeeper as a distributed coordination service to maintain server state in 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. The figure below depicts zookeeper's functionality.



5. HBase vs RDBMS

HBase	RDBMS
HBase is a schema-less or non-relational data model that doesn't have a concept of fixed columns schema.	A relation in RDBMS is identified by its schema that has fixed number of columns which can't be changed once it gets created.
HBase is vertically scalable i.e. we can add any number of columns under different column families. It's build for wide tables.	RDBMS is built for small tables and is hard to scale. It's scalable only in row level.
HBase doesn't support normalization of data.	RDBMS is designed to support normalization.
HBase can guarantee consistency and partition tolerance.	An RDBMS table guarantee ACID properties.
Transactions are not supported in HBase.	RDBMS supports transactions over a database
HBase is suitable for structured as well as semi structured data.	RDBMS is only suitable for structured data.
HBase uses Java client API and JRuby to perform all database related operations.	RDBMS offers SQL (Structured Query Language) to create schema and query data.