Assignment 9.3

Problem Statement 1 :-

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**

NoSQL database, also called Not Only SQL, is an approach to data management and database design that's useful for very large sets of distributed data. 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 is especially useful when an enterprise needs to access and analyze massive amounts of unstructured data or data that's stored remotely on multiple virtual servers in the cloud.
NoSQL technology was originally created and used by Internet leaders such as Facebook, Google, Amazon and others, who required database management systems that could write and read data anywhere in the world, while scaling and delivering performance across massive data sets and millions of users.

Benefits of NoSQL databases
NoSQL databases provide various important advantages over traditional relational databases. A few core features of NoSQL are listed here, which apply to most NoSQL databases.
Schema agnostic: NoSQL databases are schema agnostic. You aren't required to do a lot on designing your schema before you can store data in NoSQL databases. You can start coding, and store and retrieve data without knowing how the database stores and works internally. If you need advanced functionality, then you can customize the schema manually before indexing

the data. Schema agnosticism may be the most significant difference between NoSQL and relational databases.

Scalability: NoSQL databases support horizontal scaling methodology that makes it easy to add or reduce capacity quickly without tinkering with commodity hardware. This eliminates the tremendous cost and complexity of manual sharding that is necessary when attempting to scale RDBMS.

Performance: Some databases are designed to operate best (or only) with specialized storage and processing hardware. With a NoSQL database, you can increase performance by simply adding cheaper servers, called commodity servers. This helps organizations to continue to deliver reliably fast user experiences with a predictable return on investment for adding resources again, without the overhead associated with manual sharding.

High availability: NoSQL databases are generally designed to ensure high availability and avoid the complexity that comes with a typical RDBMS architecture, which relies on primary and secondary nodes. Some 'distributed' NoSQL databases use a master less architecture that automatically distributes data equally among multiple resources so that the application remains available for both read and write operations, even when one node fails.

Global availability: By automatically replicating data across multiple servers, data centres or cloud resources, distributed NoSQL databases can minimize latency and ensure a consistent application experience wherever users are located. An added benefit is a significantly reduced database management burden of manual RDBMS configuration, freeing operations teams to focus on other business priorities.

- **Types of Nosql Databases**

Several different varieties of NoSQL databases have been created to support specific needs and use cases. These databases can broadly be categorized into four types:-

1. **Key-value store NoSQL database**

From an API perspective, key-value stores are the simplest NoSQL data stores to use. The client can either get the value for the key, assign a value for a key or

delete a key from the data store. The value is a blob that the data store just stores, without caring or knowing what's inside; it's the responsibility of the application to understand what was stored. Since key-value stores always use primary-key access, they generally have great performance and can be easily scaled. The key-value database uses a hash table to store unique keys and pointers (in some databases it's also called the inverted index) with respect to each data value it stores. There are no column type relations in the database; hence, its implementation is easy. Key-value databases give great performance and can be very easily scaled as per business needs.

Use cases: Here are some popular use cases of the key-value databases:

- For storing user session data
- Maintaining schema-less user profiles
- Storing user preferences
- Storing shopping cart data

However key-value databases are not the ideal choice for every use case when:

- We have to query the database by specific data value.
- We need relationships between data values.
- We need to operate on multiple unique keys.
- Our business needs updating a part of the value frequently.

Examples of this database are Redis, MemcacheDB and Riak.

2. **Document store NoSQL database**

Document store NoSQL databases are similar to key-value databases in that there's a key and a value. Data is stored as a value. Its associated key is the unique identifier for that value. The difference is that, in a document database, the value contains structured or semi-structured data. This structured/semi-structured value is referred to as a document and can be in XML, JSON or BSON format.

Use cases: Document store databases are preferable for:

- E-commerce platforms
- Content management systems

- Analytics platforms
- Blogging platforms

Document store NoSQL databases are not the right choice if you have to run complex search queries or if your application requires complex multiple operation transactions.

Examples of document store NoSQL databases are MongoDB, Apache CouchDB and Elasticsearch.

3. **Column store NoSQL database**

In column-oriented NoSQL databases, data is stored in cells grouped in columns of data rather than as rows of data. Columns are logically grouped into column families. Column families can contain a virtually unlimited number of columns that can be created at runtime or while defining the schema. Read and write is done using columns rather than rows. Column families are groups of similar data that is usually accessed together. As an example, we often access customers' names and profile information at the same time, but not the information on their orders.

The main advantages of storing data in columns over relational DBMS are fast search/access and data aggregation. Relational databases store a single row as a continuous disk entry. Different rows are stored in different places on the disk while columnar databases store all the cells corresponding to a column as a continuous disk entry, thus making the search/access faster.

Each column family can be compared to a container of rows in an RDBMS table, where the key identifies the row and the row consists of multiple columns. The difference is that various rows do not have to have the same columns, and columns can be added to any row at any time without having to add them to other rows.

Use cases: Developers mainly use column databases in:

- Content management systems
- Blogging platforms
- Systems that maintain counters

- Services that have expiring usage
- Systems that require heavy write requests (like log aggregators)

Column store databases should be avoided if you have to use complex querying or if your querying patterns frequently change. Also avoid them if you don't have an established database requirement, a trend which we are beginning to see in new systems.

Examples of column store NoSQL databases are Cassandra and Apache Hadoop Hbase.

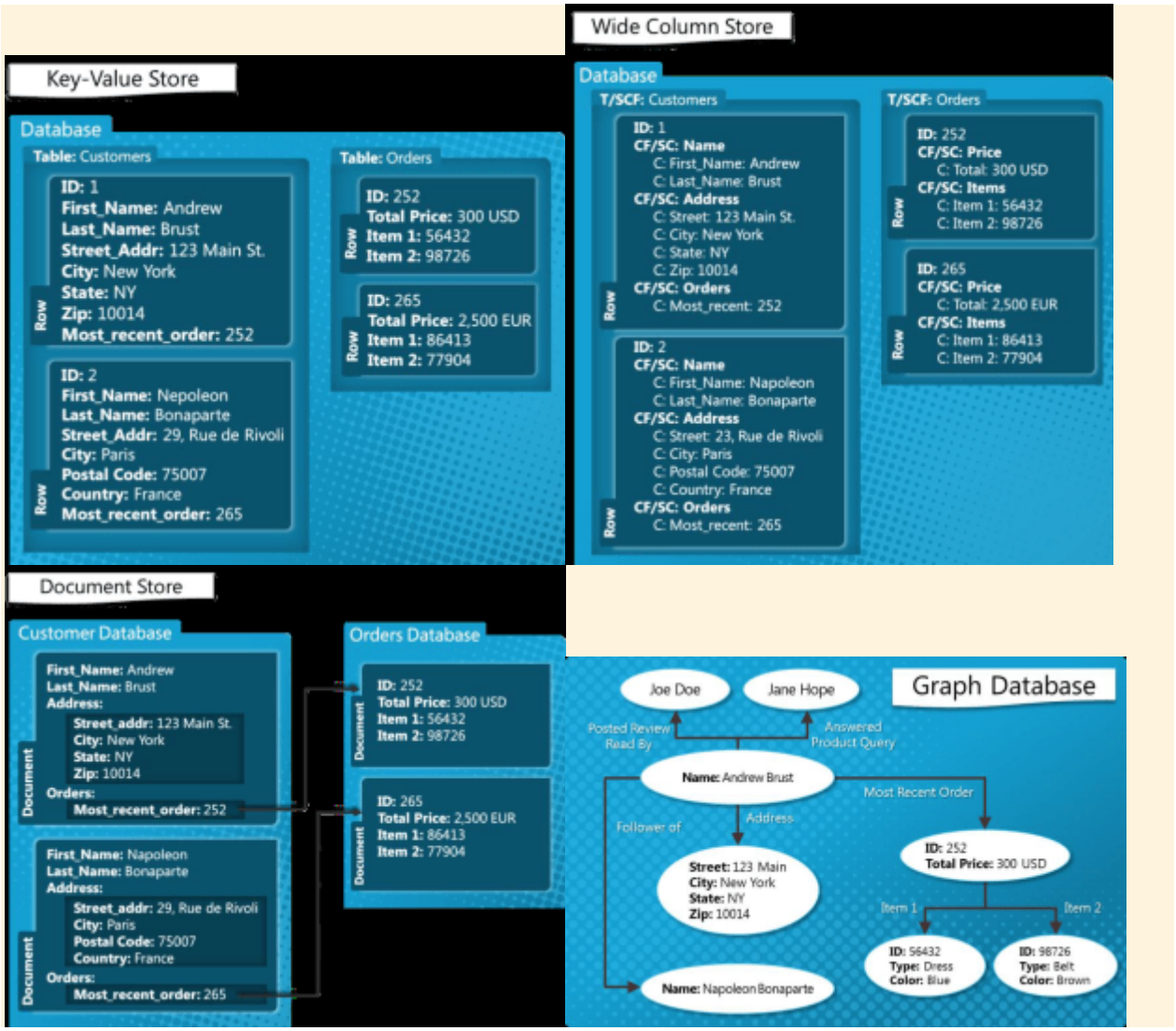
4. **Graph base NoSQL database**

Graph databases are basically built upon the Entity – Attribute – Value model. Entities are also known as nodes, which have properties. It is a very flexible way to describe how data relates to other data. Nodes store data about each entity in the database, relationships describe a relationship between nodes, and a property is simply the node on the opposite end of the relationship. Whereas a traditional database stores a description of each possible relationship in foreign key fields or junction tables, graph databases allow for virtually any relationship to be defined on-the-fly.

Use cases: Graph base NoSQL databases are usually used in:

- Fraud detection
- Graph based search
- Network and IT operations
- Social networks, etc.

Examples of graph base NoSQL databases are Neo4j, ArangoDB and OrientDB.

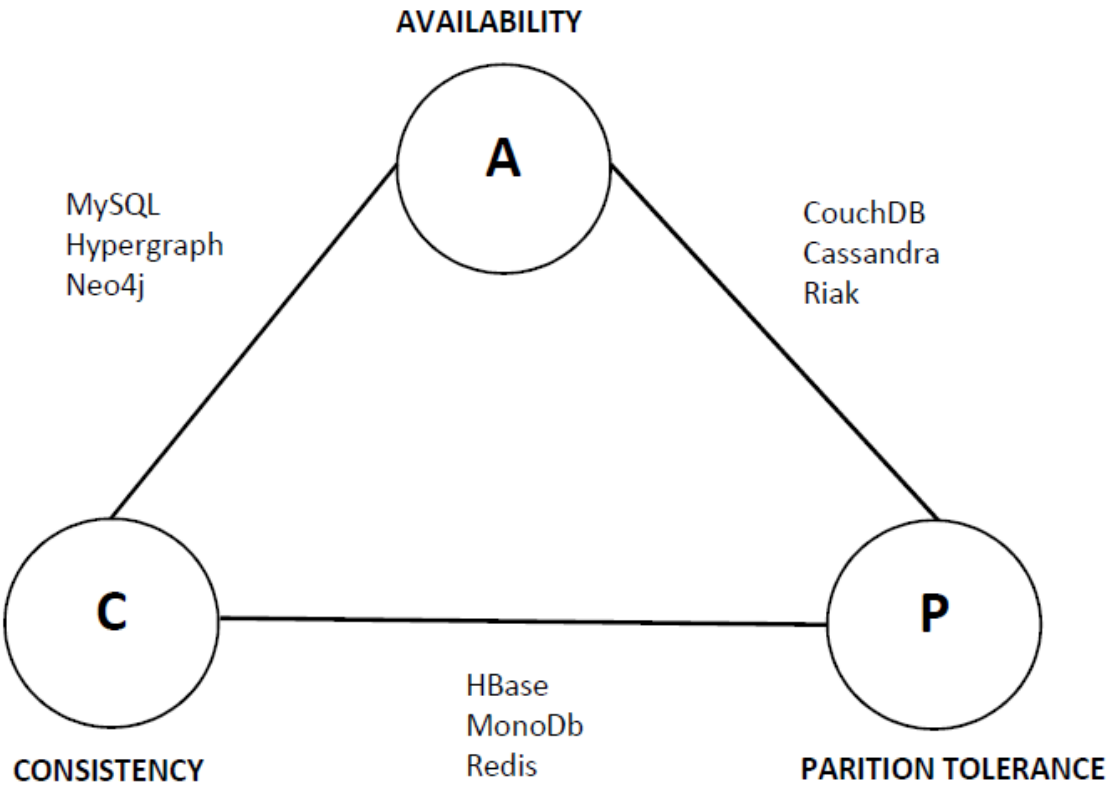


- **CAP Theorem:-**

Consistency: All the servers in the system will have the same data so anyone accessing the system will get the same copy regardless of which server answer's the request.

Availability: The system will always respond to a request.

Partition Tolerance: The system continues to operate as a whole even if individuals' server fail or can't be reached

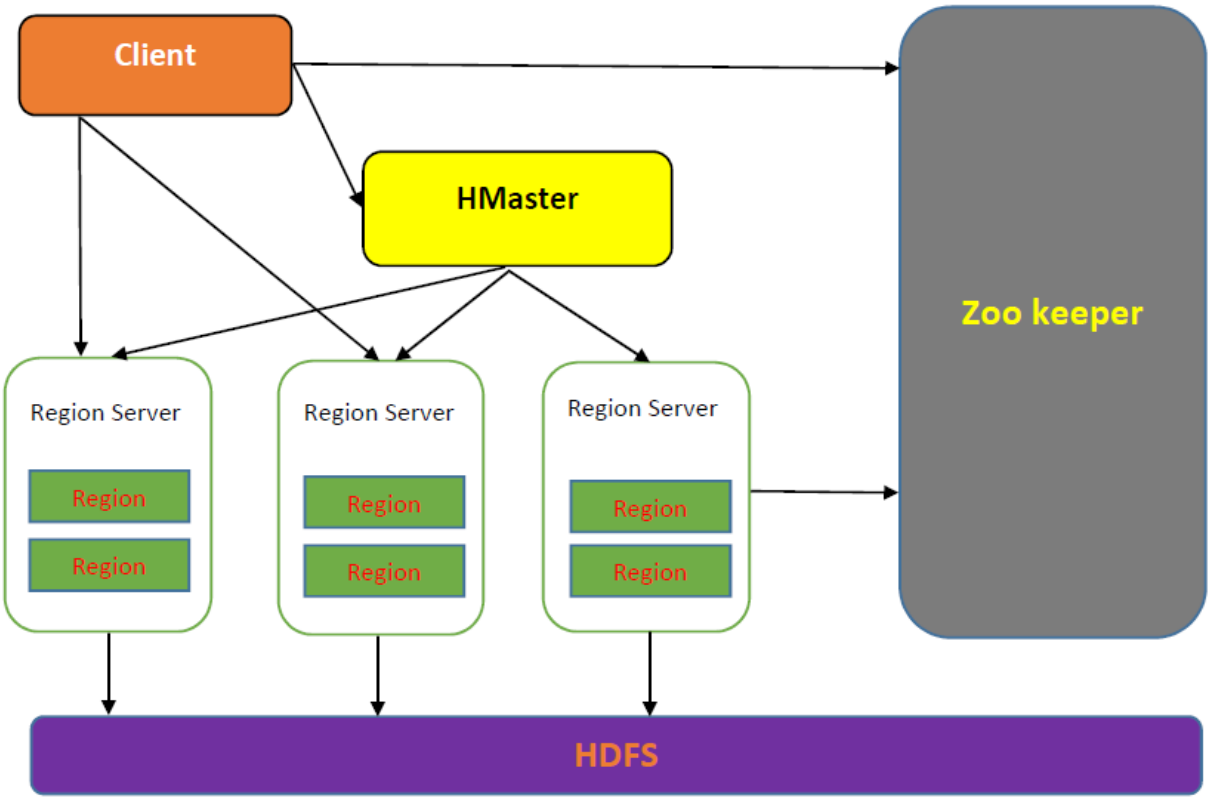


- **HBase Architecture:-**

HBase is composed of three types of servers in a master slave type of architecture. Region servers serve data for reads and writes. When accessing data, clients communicate with HBase Region Servers directly. Region assignment, DDL (create, delete tables) operations are handled by the HBase Master process. Zookeeper, which is part of HDFS, maintains a live cluster state.

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 Region Server instances in the cluster (listens for notifications from zookeeper)

- Admin functions
  - Interface for creating, deleting, and 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.

- **HBase vs. RDBMS**

RDBMS	HBASE
RDBMS is row-oriented databases	HBase is a distributed, column-oriented data storage system
RDBMS tables have fixed-schema	Hbase tables do not have fixed-schema
RDBMS tables guarantee ACID properties	Hbase tables guarantee consistency and partition tolerance
RDBMS uses SQL (Structured query Langauge) to query the data	Hbase uses Java client API and Jruby

Submitted By:-

Hardik Kaushik