**Assignment 9.3**

**Problem Statement:**

Explain the following in brief with an example.

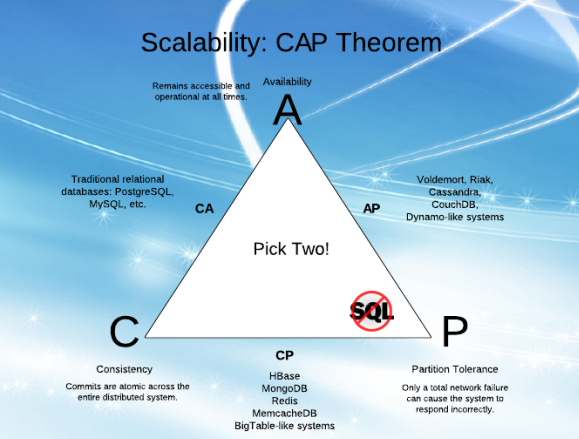
**Nosql Databases:**

* NoSQL encompasses a wide variety of different database technologies that were developed in response to the demands presented in building modern applications:
* Developers are working with applications that create massive volumes of new, rapidly changing data types — structured, semi-structured, unstructured and polymorphic data.
* Long gone is the twelve-to-eighteen month waterfall development cycle. Now small teams work in agile sprints, iterating quickly and pushing code every week or two, some even multiple times every day.
* Applications that once served a finite audience are now delivered as services that must be always-on, accessible from many different devices and scaled globally to millions of users.
* Organizations are now turning to scale-out architectures using open source software, commodity servers and cloud computing instead of large monolithic servers and storage infrastructure.
* Relational databases were not designed to cope with the scale and agility challenges that face modern applications, nor were they built to take advantage of the commodity storage and processing power available today.

**Types of Nosql Databases:**

* **Document databases** pair each key with a complex data structure known as a document. Documents can contain many different key-value pairs, or key-array pairs, or even nested documents.
* **Graph stores** are used to store information about networks of data, such as social connections. Graph stores include Neo4J and Giraph.
* **Key-value stores** are the simplest NoSQL databases. Every single item in the database is stored as an attribute name (or 'key'), together with its value. Examples of key-value stores are Riak and Berkeley DB. Some key-value stores, such as Redis, allow each value to have a type, such as 'integer', which adds functionality.
* **Wide-column stores** such as Cassandra and HBase are optimized for queries over large datasets, and store columns of data together, instead of rows.

**CAP Theorem:**



|  |  |
| --- | --- |
|  | Consistency means that data is the same across the cluster, so you can read or write to/from any node and get the same data.  Availability means the ability to access the cluster even if a node in the cluster goes down.  Partition Tolerance means that the cluster continues to function even if there is a "partition" (communications break) between two nodes (both nodes are up, but can't communicate).  In order to get both availability and partition tolerance, you have to give up consistency. Consider if you have two nodes, X and Y, in a master-master setup. Now, there is a break between network comms in X and Y, so they can't synch updates. At this point you can either:  A) Allow the nodes to get out of sync (giving up consistency), or  B) Consider the cluster to be "down" (giving up availability)  All the combinations available are:   * **CA** - data is consistent between all nodes - as long as all nodes are online - and you can read/write from any node and be sure that the data is the same, but if you ever develop a partition between nodes, the data will be out of sync (and won't re-sync once the partition is resolved). * **CP** - data is consistent between all nodes, and maintains partition tolerance (preventing data desync) by becoming unavailable when a node goes down. * **AP** - nodes remain online even if they can't communicate with each other and will resync data once the partition is resolved, but you aren't guaranteed that all nodes will have the same data (either during or after the partition).   **HBase Architecture:**    HBase provides low-latency random reads and writes on top of HDFS. In HBase, tables are dynamically distributed by the system whenever they become too large to handle (Auto Sharding). 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.  HBase can be run in a multiple master setup, wherein there is only single active master at a time. HBase tables are partitioned into multiple regions with every region storing multiple table’s rows.  **Components of Apache HBase Architecture**  HBase architecture has 3 important components- HMaster, Region Server and ZooKeeper.  **HMaster:**   * HBase HMaster is a lightweight process that assigns regions to region servers in the Hadoop cluster for load balancing. Responsibilities of HMaster – * Manages and Monitors the Hadoop Cluster * Performs Administration (Interface for creating, updating and deleting tables.) * Controlling the failover * DDL operations are handled by the HMaster * Whenever a client wants to change the schema and change any of the metadata operations, HMaster is responsible for all these operations.   **Region Server:**   * These are the worker nodes which handle read, write, update, and delete requests from clients. Region Server process, runs on every node in the hadoop cluster. Region Server runs on HDFS DataNode and consists of the following components – * Block Cache – This is the read cache. Most frequently read data is stored in the read cache and whenever the block cache is full, recently used data is evicted. * MemStore- This is the write cache and stores new data that is not yet written to the disk. Every column family in a region has a MemStore. * Write Ahead Log (WAL) is a file that stores new data that is not persisted to permanent storage. * HFile is the actual storage file that stores the rows as sorted key values on a disk.   **Zookeeper:**  HBase uses ZooKeeper as a distributed coordination service for region assignments and to recover any region server crashes by loading them onto other region servers that are functioning. ZooKeeper is a centralized monitoring server that maintains configuration information and provides distributed synchronization. Whenever a client wants to communicate with regions, they have to approach Zookeeper first. HMaster and Region servers are registered with ZooKeeper service, client needs to access ZooKeeper quorum in order to connect with region servers and HMaster. In case of node failure within an HBase cluster, ZKquoram will trigger error messages and start repairing failed nodes.  ZooKeeper service keeps track of all the region servers that are there in an HBase cluster- tracking information about how many region servers are there and which region servers are holding which DataNode. HMaster contacts  ZooKeeper to get the details of region servers. Various services that Zookeeper provides include –   * Establishing client communication with region servers. * Tracking server failure and network partitions. * Maintain Configuration Information * Provides ephemeral nodes, which represent different region servers.   **HBase vs RDBMS** |