CAP THEOREM

The three letters in CAP refer to three desirable properties of distributed systems with replicated data: **consistency** (among replicated copies), **availability** (of the system for read and write operations) and **partition tolerance**.
The CAP theorem states that it is not possible to guarantee all three of the desirable properties – consistency, availability, and partition tolerance at thesame time in a distributed system with data replication.

The theorem states that networked shared-data systems can only strongly support two of the following three properties:

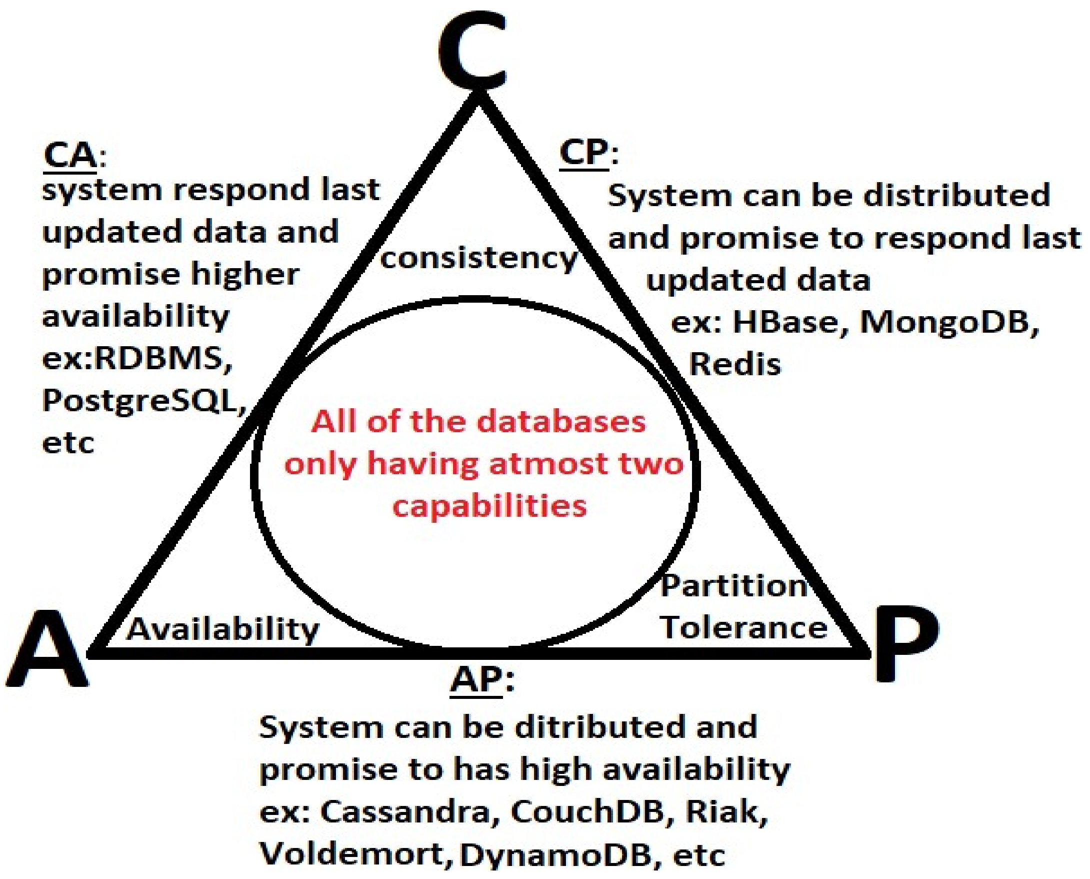
- **Consistency–**
  Consistency means that the nodes will have the same copies of a replicated data item visible for various transactions. A guarantee that every node in a distributed cluster returns the same, most recent and a successful write. Consistency refers to every client having the same view of the data. There are various types of consistency models. Consistency in CAP refers to sequential consistency, a very strong form of consistency.

- **Availability–**
  Availability means that each read or write request for a data item will either be processed successfully or will receive a message that the operation cannot be completed. Every non-failing node returns a response for all the read and write requests in a reasonable amount of time. The key word here is "every". In simple terms, every node (on either side of a network partition) must be able to respond in a reasonable amount of time.

- **Partition Tolerance-** –
  Partition tolerance means that the system can continue operating even if the network connecting the nodes has a fault that results in two or more partitions, where the nodes in each partition can only communicate among each other. That means, the system continues to function and upholds its consistency guarantees in spite of network partitions. Network partitions are a fact of life. Distributed systems guaranteeing partition tolerance can gracefully recover from partitions once the partition heals.

In CAP, the term consistency refers to the consistency of the values in different copies of the same data item in a replicated distributed system. In ACID, it refers to the fact that a transaction will not violate the integrity constraints specified on the database schema.
The CAP theorem states that distributed databases can have at most two of the three properties: consistency, availability, and partition tolerance. As a result, database systems prioritize only two properties at a time.

The following figure represents which database systems prioritize specific properties at a given time:


C
CA:
system respond last updated data and promise higher availability ex:RDBMS, PostgreSQL, etc
CP:
System can be distributed and promise to respond last updated data ex: HBase, MongoDB, Redis
consistency
All of the databases only having atmost two capabilities
Partition Tolerance
Availability
A
P
AP:
System can be ditributed and promise to has high availability ex: Cassandra, CouchDB, Riak, Voldemort, DynamoDB, etc

CAP theorem with databases examples

- **CA(Consistency and Availability)-**

The system prioritizes availability over consistency and can respond with possibly stale data. Example databases: Cassandra, CouchDB, Riak, Voldemort.

- **AP(Availability and Partition Tolerance)-**

The system prioritizes availability over consistency and can respond with possibly stale data. The system can be distributed across multiple nodes and is designed to operate reliably even in the face of network partitions.

Example databases: Amazon DynamoDB, Google Cloud Spanner.

- **CP(Consistency and Partition Tolerance)-**

The system prioritizes consistency over availability and responds with the latest updated data. The system can be distributed across multiple nodes and is designed to operate reliably even in the face of network partitions.

Example databases: Apache HBase, MongoDB, Redis.