# PACELC Theorem

We cannot avoid partition in a distributed system, therefore, according to the CAP Theorem, a distributed system should choose between Consistency or Availability.

**ACID** (Atomicity, Consistency, Isolation, Durability) databases, such as RDBMS like MySQL, Oracle and Microsoft SQL Server, chose consistency (*refute response if it cannot check with peers*).

While **BASE** (Basically Available, Soft-state, Evetnually consistent) databases, such as NoSQL databases like MongoDB, Cassandra and Redis, chose availability (*respond with local data without ensuring it is the latest with its peers*).

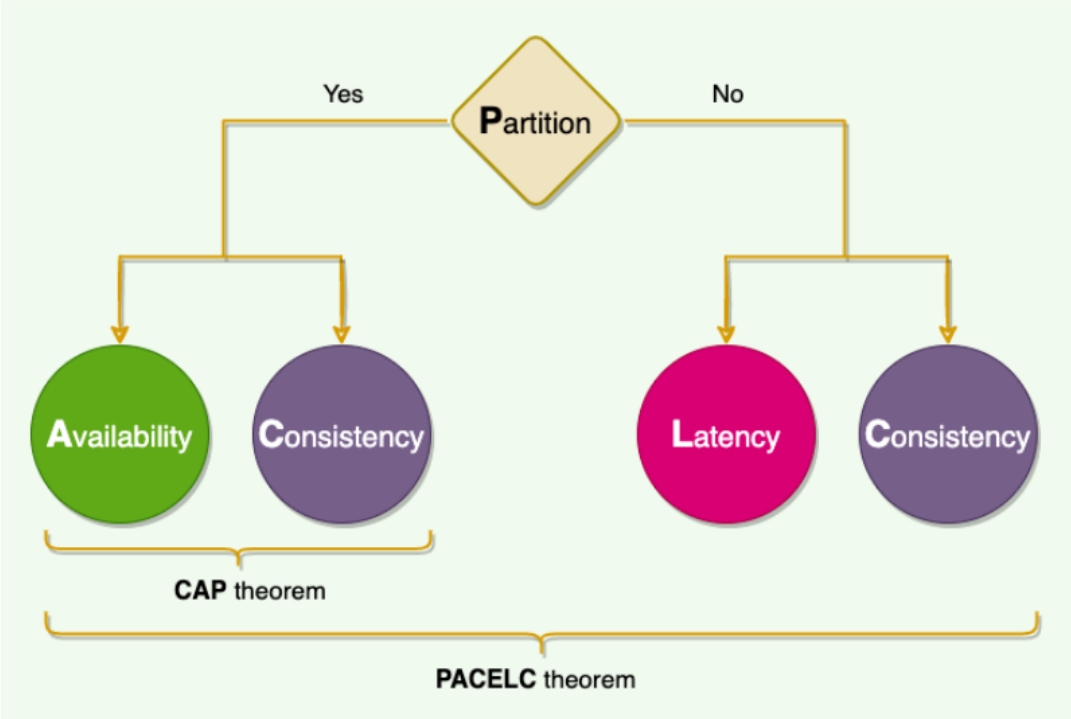
One place where the CAP Theorem is silent is what happens when there is no network partition.

What choices does a distributed system have when there is no partition?

❗ The **PACELC Theorem** states that in a system that replicates data:

If there is a partition (P), a distributed system can tradeoff between availability (A) and consistency (C).

Otherwise, if there is no partition, a system can tradeoff between latency (L) and consistency (C).




PACELC theorem

The first part of the theorem (PAC) is the same as the CAP Theorem, and the ELC is the extension.

The whole thesis is assuming we maintain high availability by replication.

So, when there is a failure, CAP Theorem prevails.

But if not, we still have to consider the tradeoff between consistency and latency of a replicated system.

## Examples

- Dynamo and Cassandra are PA/EL systems — they choose availability over consistency when a partition occurs; otherwise they choose lower latency.
- BigTable and HBase are PC/EC systems — they will always choose consistency, giving up availability and lower latency.
- MongoDB can be considered PA/EC (default configuration) — MongoDB works in a primary-secondaries configuration; in the default configuration, all writes and reads are performed on the primary.

  As all replication is done asynchronously (from primary to secondaries), when

there is a network partition in which the primary is lost or becomes isolated on the minority side, there is a chance of losing data this is unreplicated to secondaries. Hence, there is a loss of consistency during partitions.

Therefore, it can be concluded that in the case of a network partition, MongoDB chooses availability, but otherwise guarantees consistency.

Alternately, when MongoDB is configured to write on majority replicas and read from the primary, it could be categorised as PC/EC.