CAP Theorem

Consistency Availability Partition tolerance

Not all the three would be true at the same time.

CP, AP, AC - are the only possible combinations

Availability: Every request received by a non-failing node in the system must

return a response.

Consistency: In a consistent system, once a client writes a value to any server

and gets a response, it expects to get that value (or a fresher value) back

from any server it reads from.

Partition tolerance: In case of network failure, there should be no impact.

If a node fails and it is partitioned from the clusters, that is partition.

For a distributed system, the system will continue to work unless there is a

total network failure. A few nodes can fail and the system keeps going.

CP, AP, CA:

When a network partition occurs, either of the two hold true but not both.

Either the DB would be consistent, i.e., reads and writes are consistent,

but the nodes are not available to take new requests

or the nodes are available but the messages are not consistent.

If the system is not distributed, then it need not have partition tolerance, because there is a single node and no concept of partition.

so both C and A are valid.

CP - mongoDB, redis

AP - cassandra

AP - MSSql, mariaDB

Database system designed with ACID guarantees (RDBMS) usually chooses consistency over availability

whereas system Designed with BASE guarantees, availability.

Relational databases strongly follow ACID principles:

Atomic: Everything in a transaction succeeds or the entire transaction is

rolled back.

Consistent: A transaction cannot leave the database in an inconsistent state.

Isolated: Transactions cannot interfere with each other.

Durable: Completed transactions persist, even when servers restart etc.

NoSQL databases follow BASE principles:

Basically Available, Soft State, Eventual consistency

Couch DB is eventually consistent in exchange for high availability.

CouchDB prioritizes availability, while MongoDB prioritizes consistency

https://medium.com/system-design-blog/cap-theorem-1455ce5fc0a0

Database Sharding

Graphical user interface, application

Description automatically generated

Document – Horizontal scaling

Relational – Vertical scaling

Because of all sorts of table relations

* For **cache**— use a **key-value DB**.
* For **graph-**like data — use a **graph DB**.
* If you tend to query on **subsets of columns** /features — use **column DB.**
* For all other use cases — **Relational** or **Document DB**.