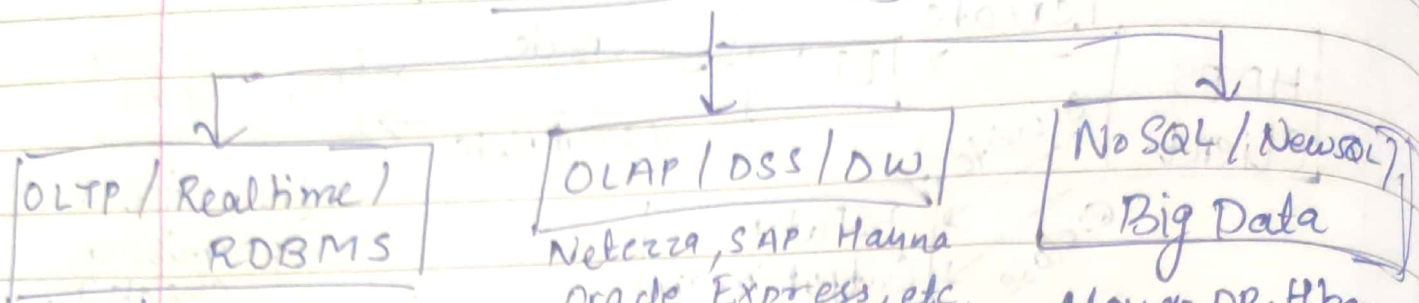
Database Categories



Oracle, MySQL, MS SQL, DB2, etc.

Netezza, SAP Hanna Oracle Express, etc.

Mongo DB, Hbase, Cassandra, couch DB, etc.

* RDBMS to NoSQL

→ RDBMS stores structured data in rows and columns. related to one another.

→ RDBMS follow ACID properties.

Atomicity Consistency Isolated Durable

→ Challenges:

1) Efficiently storing and accessing large amount of data is difficult, even more considering fault tolerance & backups.
2) Manipulating large data sets involves running immensely parallel processes
3) Managing continuously evolving schema & meta-data for semi-structured & un-structured data is difficult.

→ Explosion of social media sites with large data needs
→ Rise of cloud-based solution
→ a shift to dynamically-typed data with frequent schema changes.
→ Open-source community

→ RDBMS not suitable for Big data
→ content is internet
→ RDBMS assumes that data are
  • Dense
  • Largely uniform (structured)
→ Data coming from internet are
  • Massive & sparse
  • Semi-str or Un-str.

→ Thus rise of NoSQL took place.

* NoSQL:
→ A collection of several (related) concepts about data storage & manipulation.
→ A class of products related to large data.
→ Non-relational data storage sys
→ No fixed table schema
→ No joins
→ No multi-document transactions
→ Relaxes one or more ACID properties

* Types of NoSQL :-

1) Document Base:
→ pair each key with a complex ds called document.
→ docs can contain many diff key-value pairs, key-array pairs or even nested docs.
Eg MongoDB, CouchDB, Cloudant

2) Graph store:
→ are used to store info about networks such as social connections.
→ Eg Neo4J, Oracle NoSQL, HyperGraphDB

3) Key-value store :
→ are simplest NoSQL databases.
→ Every single item in the database is stored as an attribute name or key together with its value.
Eg Memcached, Coherence, Redis

4) Wide Column stores :
→ are optimised for queries over large datasets
→ store columns of data together instead of rows
Eg Cassandra, HBase.

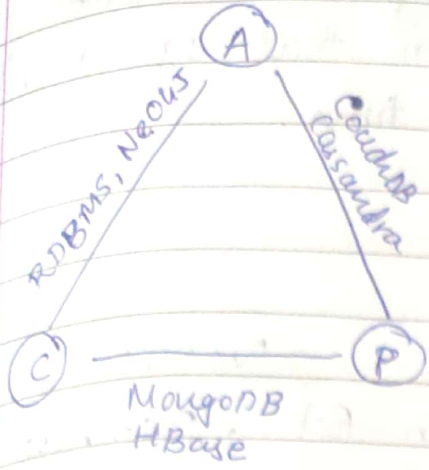
* CAP Theorem :-

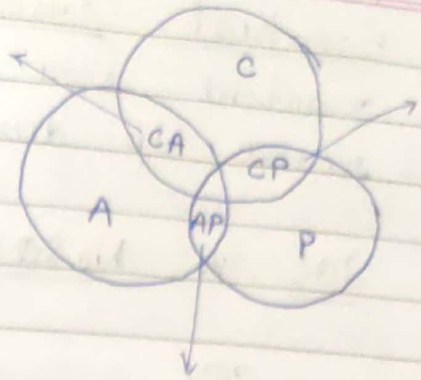
→ It states that there are 3 requirements which exist in a special relation when designing app for distributed archi.

→ Consistency:
→ data in db remains consistent after execution of operation.
Eg After update u all client see same data.

→ Availability:
→ the sys is always on, no downtime

→ Partition Tolerance :
→ sys continues to fn even the commun among the servers is unreliable.
Eg servers may be partitioned into multiple groups that cannot communicate with each other.





→ It is impossible to fulfill all 3 req.
→ CAP provides the basic req for a distributed sys to follow 2 of the 3 req.
→ ∴ all current NoSQL db follow any three CA, AP, CP combinations.

CA → Single site cluster, all nodes are always in contact.
→ when partition occurs, sys blocks

CP → Some data may not be accessible but the rest is still consistent.

AP → Sys is still available under partitioning, but some of the data returned may be inaccurate.

* BASE → sys gives up on consistency.

→ Basically Available : sys does guarantee availability, in terms of CAP theorem.
→ Soft State : state of sys may change over time even without p.ilp. becoz of eventualy consistency model.

→ Eventual Consistency : sys will become consistent over time, given that sys doesn't receive ilp during that time.

★ SQL vs NoSQL

SQL	NoSQL
→ Relational db	Non-Rel, Distributed db
→ " model	Model-less approach
→ Pre defined schema	Dynamic sch
→ Table based db	Doc, graph, wide-column, key-value
→ Vertically scalable	Horizontally
→ Uses SQL	UnQL
→ Not preferred for large data	preferred
→ " " Hierarchical	best fit
→ ACID	CAP
→ support from vendors	from community
→ " complex quering	does not have good support
→ strong consistency	few support strong consi.
Eg MySQL, Oracle, postgreSQL	Eg MongoDB, Cassandra, HBase

★ SQL vs NoSQL vs NewSQL

* MongoDB :

→ Open Source db
→ Agile db → allows schemas to change quickly as app evolve
→ by leveraging in memory computing
→ developed by 10gen, for a wide variety of app
→ Scalability, High Performance & Availability
→ native replication
→ cross platform → Distributed
→ Non-relational → Document oriented.

* Why MongoDB :

→ Open Source → Fast In place updates
→ Full Index Support → Rich Query lang
→ Auto sharding → GridFs → Map Reduce