NoSQL & CAP Theorem

Dr. Balasundaram A
School of Computer Science & Engineering
VIT, Chennai

Agenda

- What is NoSQL
- CAP Theorem

NoSQL ≠ Not SQL

does not mean

NoSQL

=

It means

Not Only SQL

OR

Not Relational Database

Why NoSQL

- **Large Volume of Data**
- **Dynamic Schemas**
- **Auto-sharding**
- **Replication**
- **Horizontally Scalable**

*** Some Operations can be achieved by Enterprise class RDBMS software but with very Highcost**

Major NoSQL Categories

- **Document databases**
  - pair each key with a complex data structure known as a document.
  - MongoDB
- **Graph databases**
  - store information about networks, such as social connections
  - Neo4j

Contd.

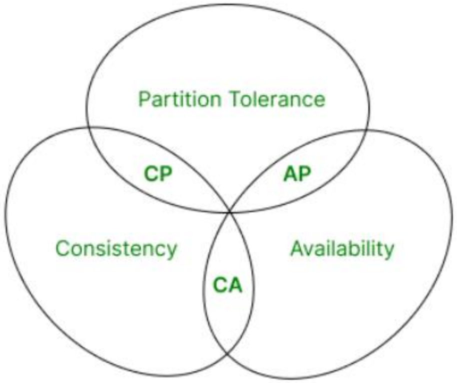
Major NoSQL Categories

- **Key-Value stores**
  - Every single item in the database is stored as an attribute name (or "key"),
  - Riak , Voldemort, Redis
- **Wide-column stores**
  - store data in columns together, instead of row
  - Google’s Bigtable, Cassandra and HBase

CAP Theorem

CAP Theorem

- The CAP theorem emphasizes the limitations that system designers have while addressing distributed data replication.
- It states that only two of the three properties—**consistency, availability, and partition tolerance**—can be concurrently attained by a distributed system.
- Developers must carefully balance these attributes according to their particular application demands because of this underlying restriction.
- Designers may decide which qualities to prioritize to obtain the best performance and reliability for their systems by knowing the CAP theorem.



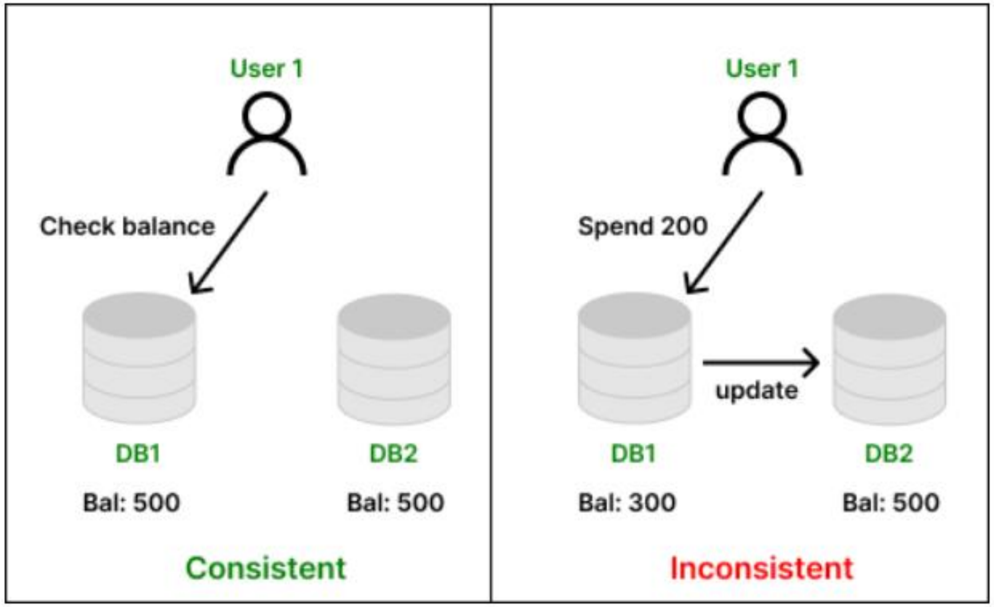
Venn diagram of CAP theorem

CAP Theorem

- **C**onsistency
  - All the servers in the system will have the same data so anyone using the system will get the same copy regardless of which server answers their request.
- **A**vailability
  - The system will always respond to a request (even if it's not the latest data or consistent across the system or just a message saying the system isn't working)
- **P**artition Tolerance
  - The system continues to operate as a whole even if individual servers fail or can't be reached.

C - Consistency

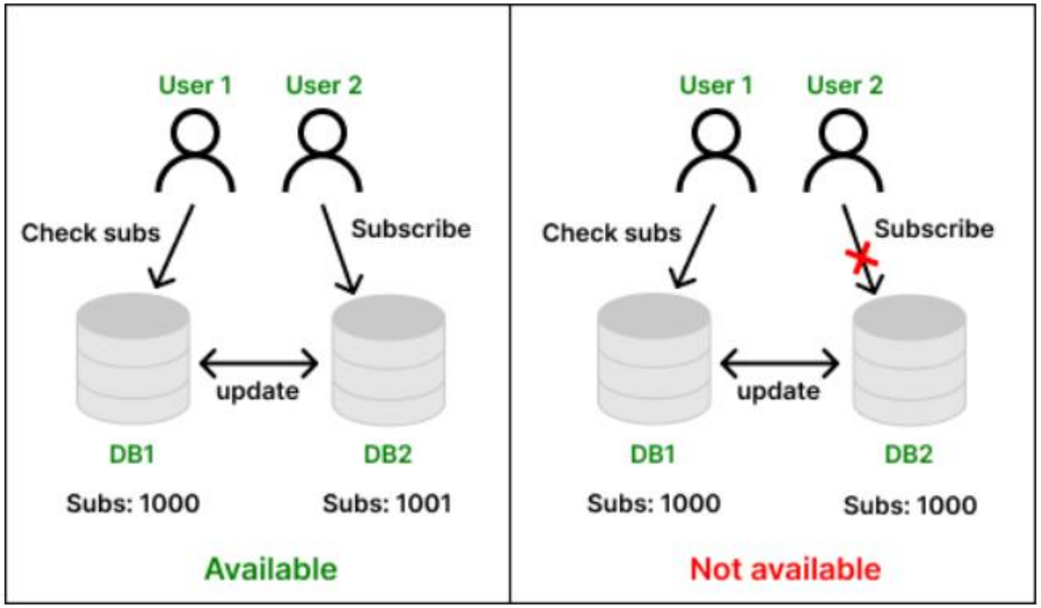
- Consistency means that all the nodes (databases) inside a network will have the same copies of a replicated data item visible for various transactions.
- It guarantees that every node in a distributed cluster returns the same, most recent, and successful write.
- It refers to every client having the same view of the data.



Consistency problem

A - Availability

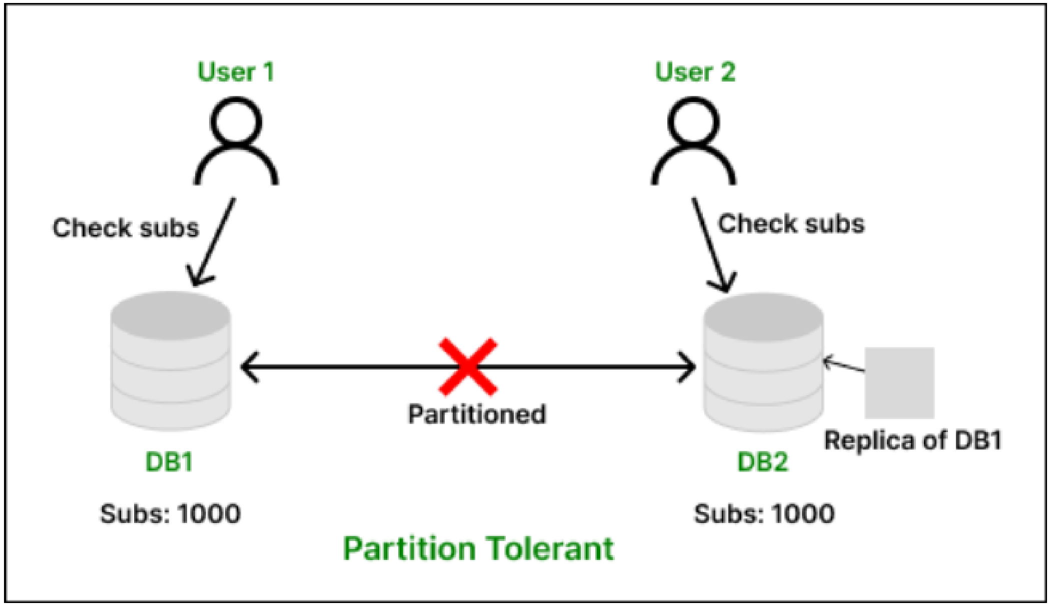
- Availability means that each read or write request for a data item will either be processed successfully or will receive a message that the operation cannot be completed.
- Every non-failing node returns a response for all the read and write requests in a reasonable amount of time. The key word here 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.



Availability problem

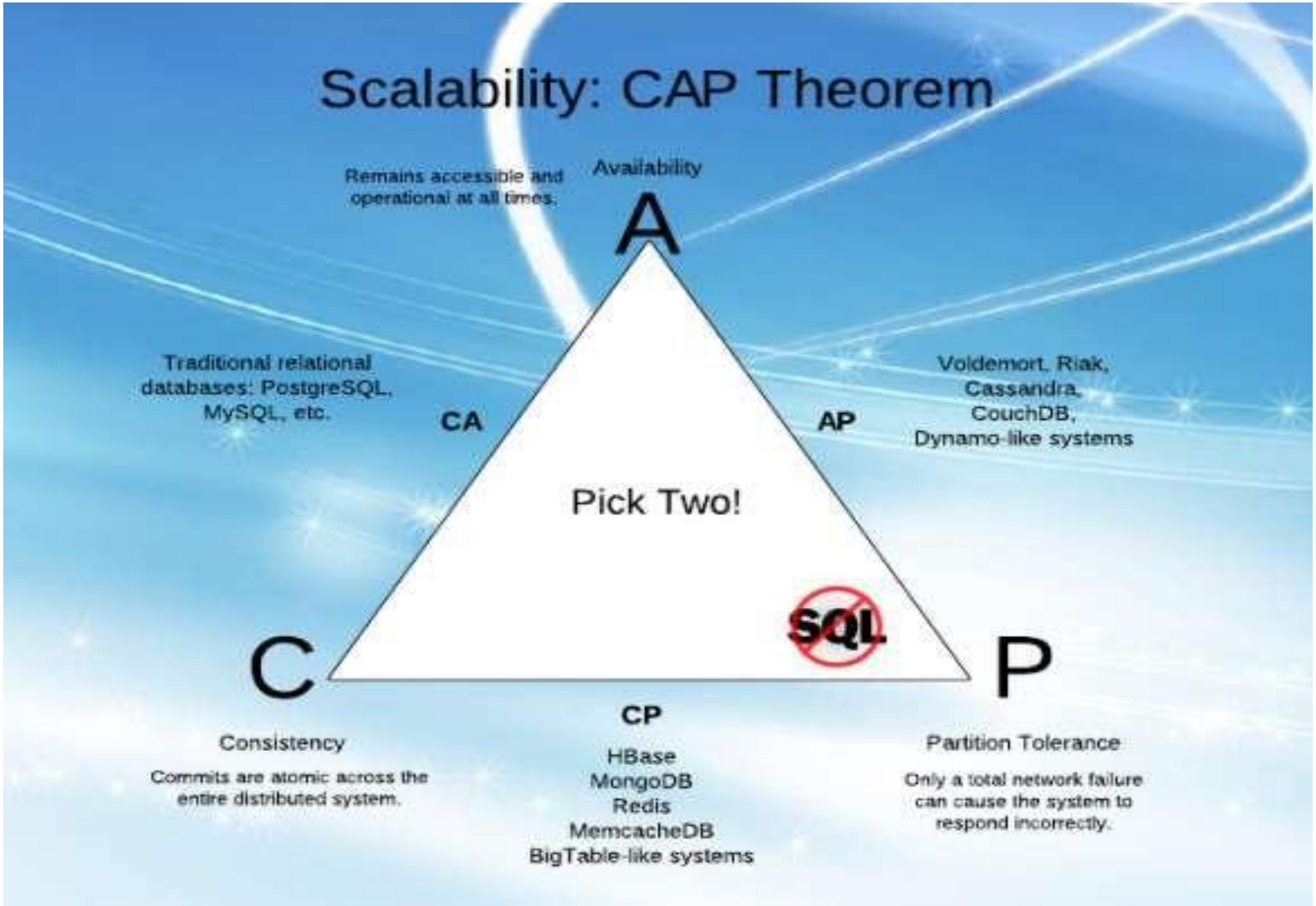
P – Partition Tolerant

- 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.
- The system continues to function and upholds its consistency guarantees in spite of network partitions.
- Distributed systems guaranteeing partition tolerance can gracefully recover from partitions once the partition heals.



Partition Tolerance

CAP Theorem – Trade Offs



Scalability: CAP Theorem

Availability

Remains accessible and operational at all times.

Traditional relational databases: PostgreSQL, MySQL, etc.

Voldemort, Riak, Cassandra, CouchDB, Dynamo-like systems

Pick Two!

Consistency

Commits are atomic across the entire distributed system.

CP

HBase
MongoDB
Redis
MemcacheDB
BigTable-like systems

Partition Tolerance

Only a total network failure can cause the system to respond incorrectly.

Credit: http://architects.dzone.com/articles/better-explaining-cap-theorem

CA – (Consistency & Availability)

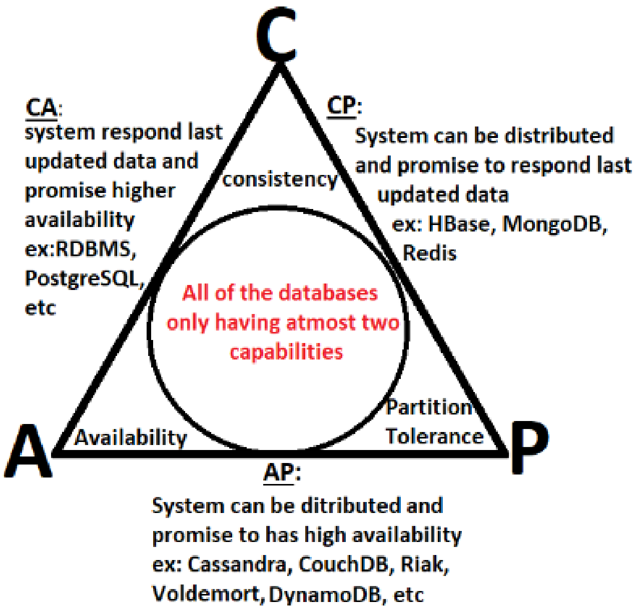
- These types of system always accept the request to view or modify the data sent by the user and they are always responded with data which is consistent among all the database nodes of a big, distributed network.
- Such type of distributed systems is not realizable in real world because when network failure occurs, there are two options:
  - Either send old data which was replicated moments ago before network failure or do not allow user to access the already moments old data.
  - If we choose first option, our system will become Available and if we choose second option our system will become Consistent.
  - Example databases: MySQL, PostgreSQL

AP – (Availability & Partition Tolerant)

- These types of system are distributed in nature, ensuring that the request sent by the user to view or modify the data present in the database nodes are not dropped and are processed in presence of a network partition.
- The system prioritizes availability over consistency and can respond with possibly stale data which was replicated from other nodes before the partition was created due to some technical failure.
- Such design choices are generally used while building social media websites such as Facebook, Instagram, Reddit, etc. and online content websites like YouTube, blog, news, etc. where consistency is usually not required,
- Example databases: Amazon DynamoDB, Google Cloud Spanner.

CP – (Consistency & Partition Tolerant)

- These types of system are distributed in nature, ensuring that the request sent by the user to view or modify the data present in the database nodes are dropped instead of responding with inconsistent data in presence of a network partition.
- The system prioritizes consistency over availability and does not allow users to read crucial data from the stored replica which was backed up prior to the occurrence of network partition. Example applications: stock market application, ticket booking application, banking, etc. where problem will arise due to old data present to users of application.
- Example databases: Apache HBase, MongoDB, Redis.

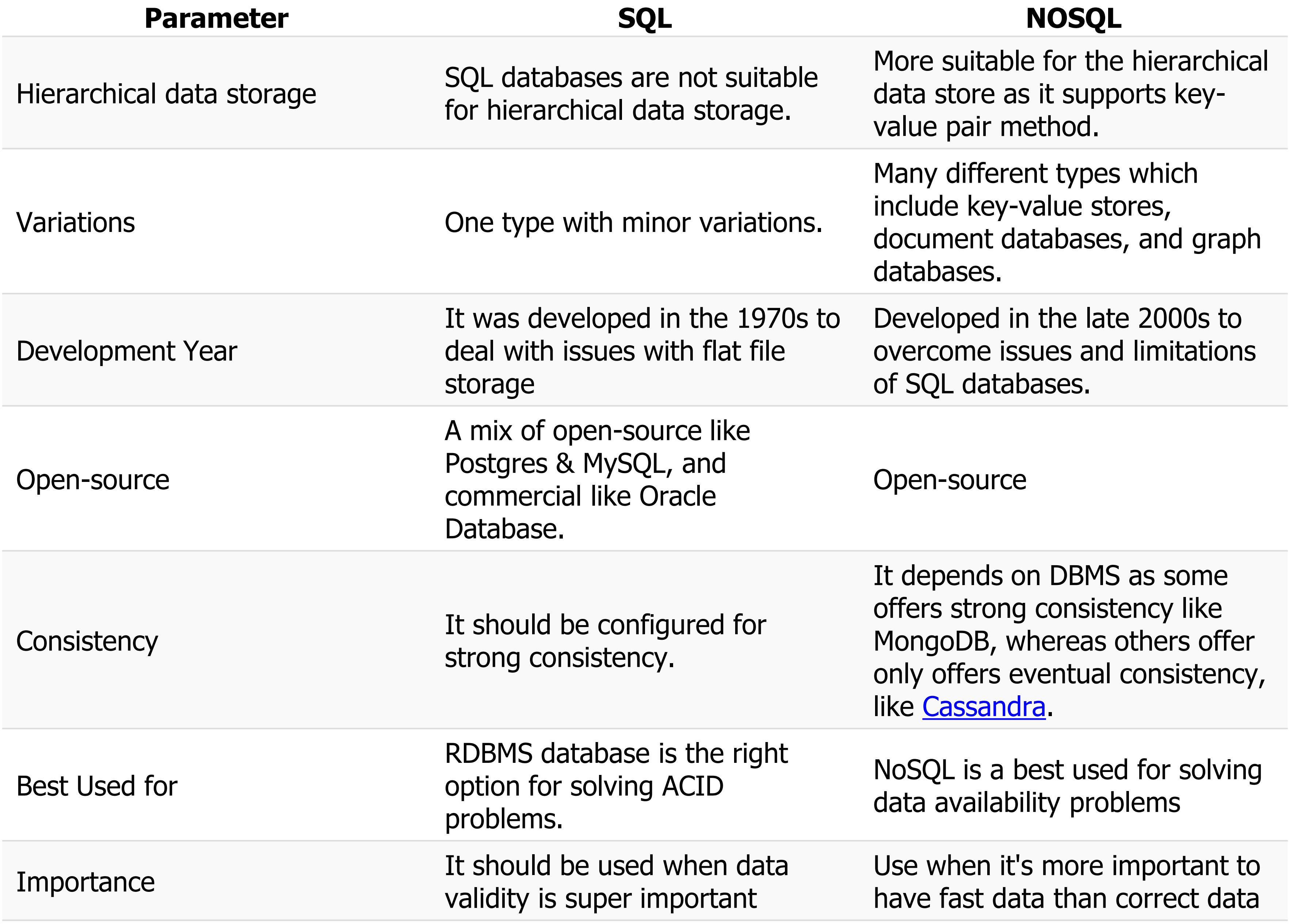

C
CA:
system respond last updated data and promise higher availability ex:RDBMS, PostgreSQL, etc
consistency
CP:
System can be distributed and promise to respond last updated data ex: HBase, MongoDB, Redis
All of the databases only having atmost two capabilities
A
Availability
Partition Tolerance
P
AP: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

CAP diagram

SQL vs NoSQL Difference

Parameter	SQL	NOSQL
Definition	SQL databases are primarily called RDBMS or Relational Databases	NoSQL databases are primarily called as Non-relational or distributed database
Design for	Traditional RDBMS uses SQL syntax and queries to analyze and get the data for further insights. They are used for OLAP systems.	NoSQL database system consists of various kind of database technologies. These databases were developed in response to the demands presented for the development of the modern application.
Query Language	Structured query language (SQL)	No declarative query language
Type	SQL databases are table based databases	NoSQL databases can be document based, key-value pairs, graph databases
Schema	SQL databases have a predefined schema	NoSQL databases use dynamic schema for unstructured data.
Ability to scale	SQL databases are vertically scalable	NoSQL databases are horizontally scalable
Examples	Oracle, Postgres, and MS-SQL.	MongoDB, Redis, Neo4j, Cassandra, Hbase.
Best suited for	An ideal choice for the complex query intensive environment.	It is not good fit complex queries.

SQL vs NoSQL Difference



Parameter	SQL	NOSQL
Hierarchical data storage	SQL databases are not suitable for hierarchical data storage.	More suitable for the hierarchical data store as it supports key-value pair method.
Variations	One type with minor variations.	Many different types which include key-value stores, document databases, and graph databases.
Development Year	It was developed in the 1970s to deal with issues with flat file storage	Developed in the late 2000s to overcome issues and limitations of SQL databases.
Open-source	A mix of open-source like Postgres & MySQL, and commercial like Oracle Database.	Open-source
Consistency	It should be configured for strong consistency.	It depends on DBMS as some offers strong consistency like MongoDB, whereas others offer only offers eventual consistency, like Cassandra.
Best Used for	RDBMS database is the right option for solving ACID problems.	NoSQL is a best used for solving data availability problems
Importance	It should be used when data validity is super important	Use when it's more important to have fast data than correct data

SQL vs NoSQL Difference

Parameter	SQL	NOSQL
Best option	When you need to support dynamic queries	Use when you need to scale based on changing requirements
Hardware	Specialized DB hardware (Oracle Exadata, etc.)	Commodity hardware
Network	Highly available network (Infiniband, Fabric Path, etc.)	Commodity network (Ethernet, etc.)
Storage Type	Highly Available Storage (SAN, RAID, etc.)	Commodity drives storage (standard HDDs, JBOD)
Best features	Cross-platform support, Secure and free	Easy to use, High performance, and Flexible tool.
Top Companies Using	Hootsuite, CircleCI, Gauges	Airbnb, Uber, Kickstarter
ACID vs. BASE Model	ACID(Atomicity, Consistency, Isolation, and Durability) is a standard for RDBMS	Base (Basically Available, Soft state, Eventually Consistent) is a model of many NoSQL systems

Source: https://www.guru99.com/sql-vs-nosql.html

Conclusion

☺ **SQL**

Works great, but can’t fit for large dataset

☹ **NoSQL**

Works great, but can’t fit for all Use-cases

☺ **SQL + NoSQL**