ACID

A transaction is a collection of instructions. To maintain the integrity of a database, all transactions must obey ACID properties.

Atomicity

A transaction is an atomic unit; hence, all the instructions within a transaction will successfully execute, or none of them will execute. If any of the instructions fail, the entire transaction should abort and rollback.

Consistency

A database is initially in a consistent state, and it should remain consistent after every transaction. Suppose that the transaction fails after first sub-transaction and the transaction is not rolled back; then, the database will be inconsistent.

Isolation

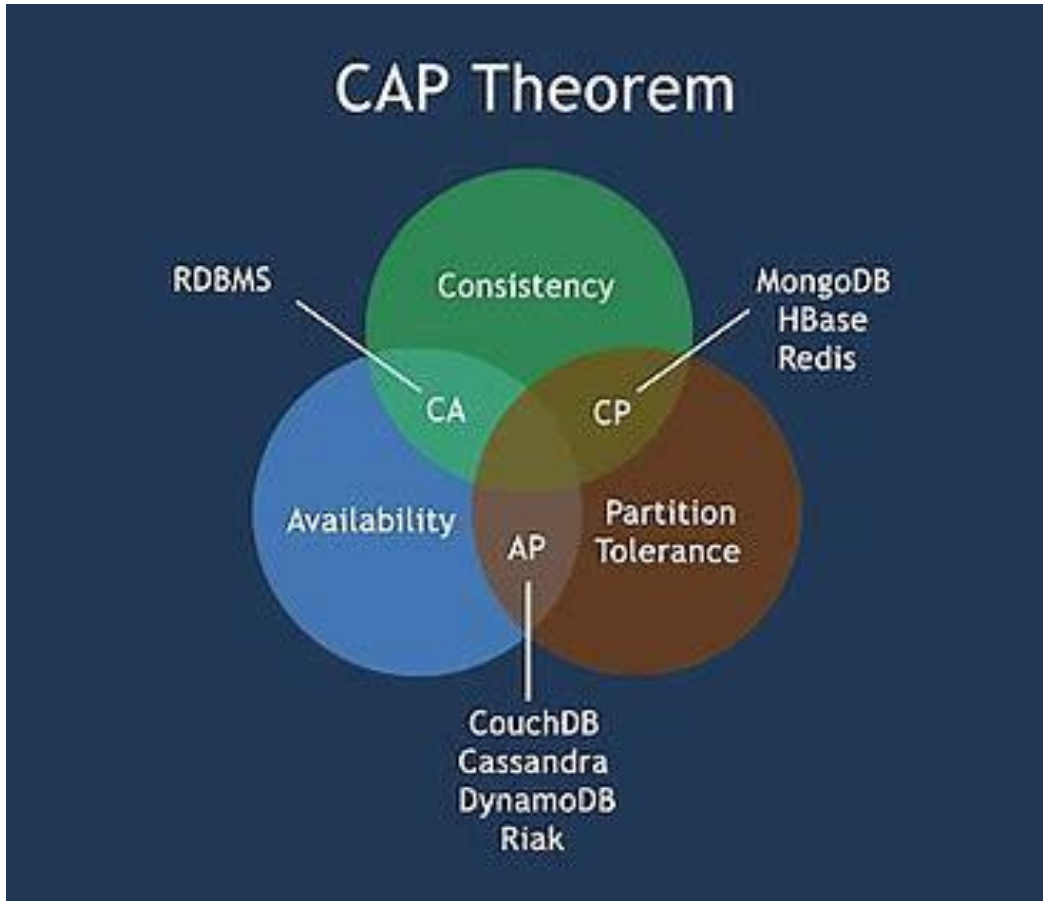
If the multiple transactions are running concurrently, they should not be affected by each other; i.e., the result should be the same as the result obtained if the transactions were running sequentially.

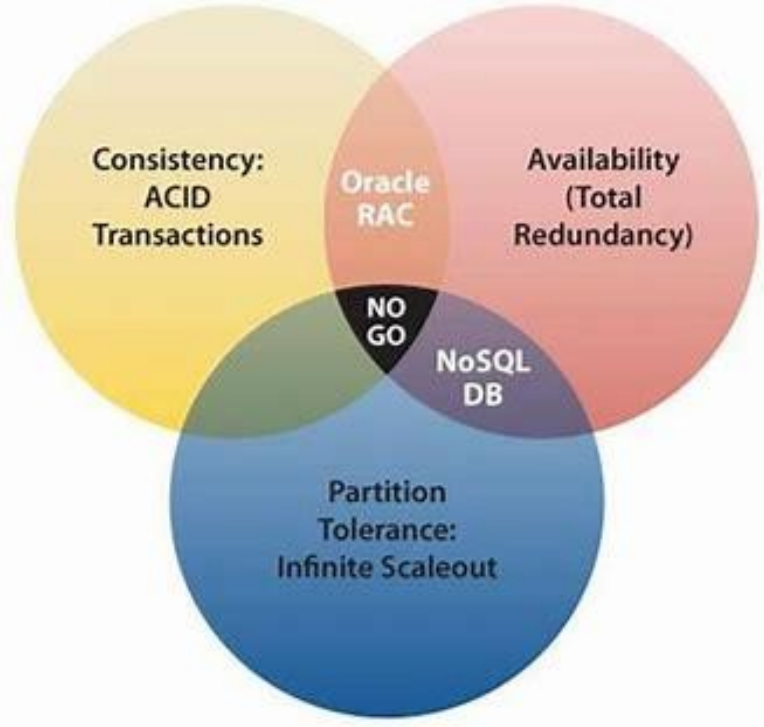
Durability

Changes that have been committed to the database should remain even in the case of software and hardware failure. For instance, if a account contains RS 10000/-, this information should not disappear upon hardware or software failure.

CAP Theorem

The CAP theorem (also called Brewer's theorem) states that a distributed database system can only guarantee two out of these three characteristics: Consistency, Availability, and Partition Tolerance.





The CAP theorem, also named Brewer's theorem after computer scientist Eric Brewer, states that it is impossible for a distributed data store to simultaneously provide more than two out of the following three guarantees :

1. Consistency
2. Availability
3. Partition Tolerance

Consistency

A system is said to be consistent if all nodes see the same data at the same time.
Simply, if we perform a read operation on a consistent system, it should return the value of the most recent write operation. This means that, the read should cause all nodes to return the same data, i.e., the value of the most recent write.

Availability

Availability in a distributed system ensures that the system remains operational 100% of the time. Every request gets a (non-error) response regardless of the individual state of a node.

Note: this does not guarantee that the response contains the most recent write.

Partition Tolerance

This condition states that the system does not fail, regardless of if messages are dropped or delayed between nodes in a system.

Partition tolerance has become more of a necessity than an option in distributed systems. It is made possible by sufficiently replicating records across combinations of nodes and networks.

When a network partition failure happens, one has to decide to

- Cancel the operation and thus decrease the availability but ensure consistency
- Proceed with the operation and thus provide availability but risk inconsistency

The CAP theorem implies that in the presence of a network partition, one has to choose between consistency and availability. Note that consistency as defined in the CAP theorem is quite different from the consistency guaranteed in ACID database transactions.

No distributed system is safe from network failures, thus network partitioning generally has to be tolerated. In the presence of a partition, one is then left with two options: consistency or availability.

- When choosing consistency over availability, the system will return an error or a time out if particular information cannot be guaranteed to be up to date due to network partitioning.
- When choosing availability over consistency, the system will always process the query and try to return the most recent available version of the information, even if it cannot guarantee it is up to date due to network partitioning.

In the absence of network failure – that is, when the distributed system is running normally – both availability and consistency can be satisfied.

	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 analyse 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.

	SQL	NOSQL
Best suited for	An ideal choice for the complex query intensive environment.	It is not good fit complex queries.
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.
Consistency	It should be configured for strong consistency.	It depends on DBMS as some offers strong consistency like MongoDB, while others offer only offers eventual consistency, like Cassandra.
Best Used for	RDBMS database is the right choice 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 is more important to have fast data than correct data

	SQL	NOSQL
Best choice	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
Storage and Network Type	Highly Available Storage and Network	Commodity drives storage and network
Best features	Cross-platform support, Secure and free	Easy to use, High performance, and Flexible tool.
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

Schema

An example of a client who needs a database design for his website. His website has the following requirements:

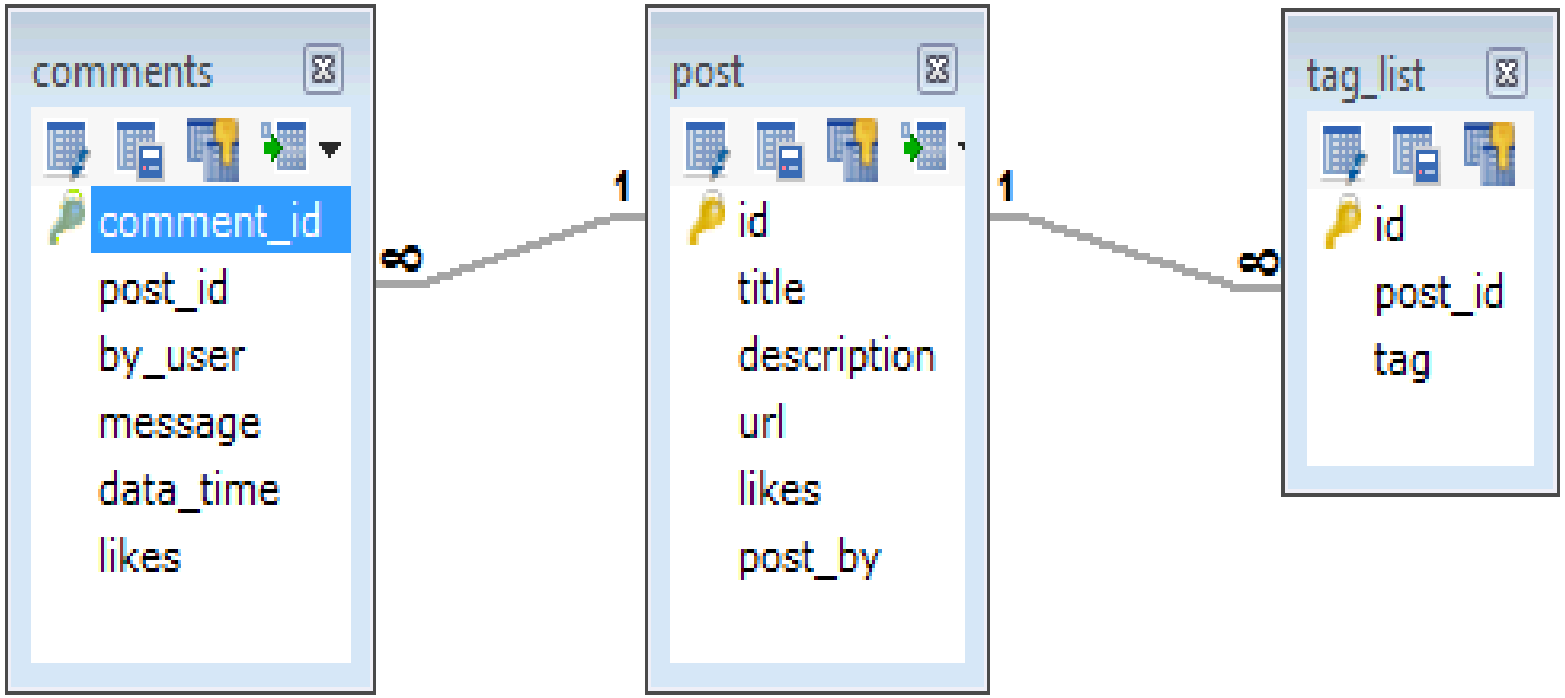
Every post is distinct (contains unique title, description and url).

Every post can have one or more tags.

Every post has the name of its publisher and total number of likes.

Each post can have zero or more comments and the comments must contain user name, message, data-time and likes.

For the above requirement, a minimum of three tables are required in RDBMS.


comments
comment_id
post_id
by_user
message
data_time
likes
1
∞
post
id
title
description
url
likes
post_by
1
∞
tag_list
id
post_id
tag

But in MongoDB, schema design will have one collection post and has the following structure:

```
{ _id: POST_ID
title: TITLE_OF_POST,
description: POST_DESCRIPTION,
by: POST_BY,
url: URL_OF_POST,
tags: [TAG1, TAG2, TAG3],
likes: TOTAL_LIKES,
comments: [ {
user: 'COMMENT_BY',
message: TEXT,
datecreated: DATE_TIME,
like: LIKES },
{ user: 'COMMENT_BY',
message: TEST,
dateCreated: DATE_TIME,
like: LIKES
}]
}
```

Multi-master replication is a method of database replication which allows data to be stored by a group of computers, and updated by any member of the group. All members are responsive to client data queries. The multi-master replication system is responsible for propagating the data modifications made by each member to the rest of the group, and resolving any conflicts that might arise between concurrent changes made by different members.

Master-slave replication is a method of database replication, in which a single member of the group is designated as the "master" for a given piece of data and is the only node allowed to modify that data item. Other members wishing to modify the data item must first contact the master node. Allowing only a single master makes it easier to achieve consistency among the members of the group, but is less flexible than multi-master replication.

REDIS

Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperlogs, geospatial indexes with radius queries and streams. Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.