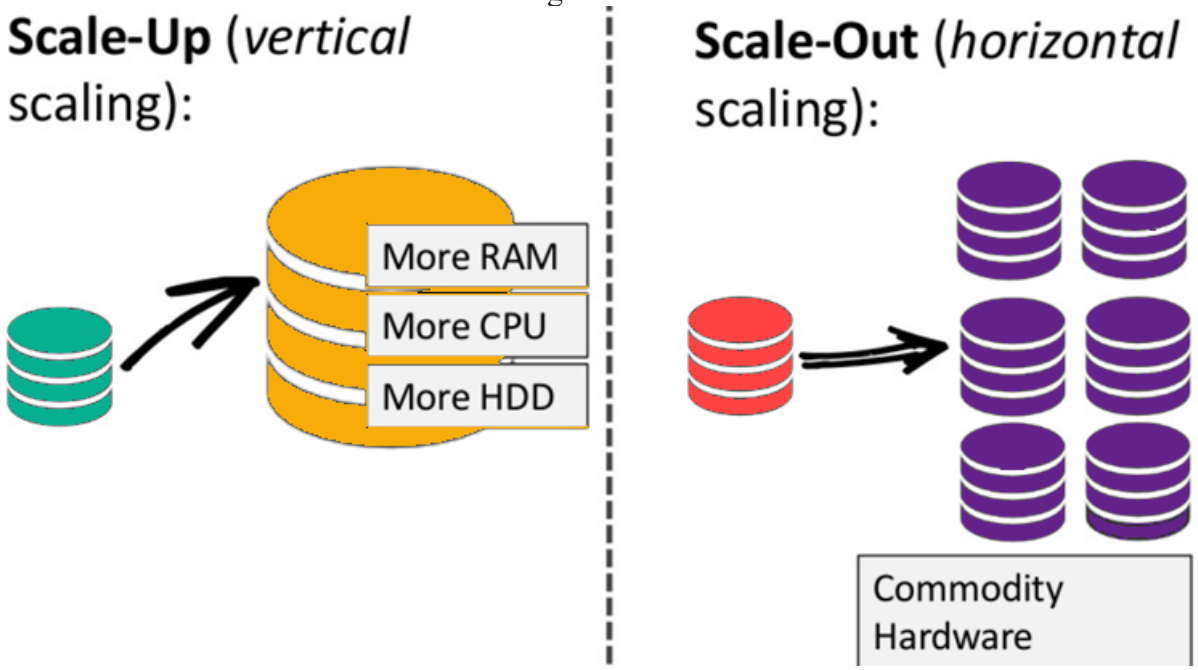
Why NoSQL?

https://youtu.be/0buKQHokLK8

The concept of NoSQL databases became popular with Internet giants like Google, Facebook, Amazon, etc. who deal with huge volumes of data. The system response time becomes slow when you use RDBMS for massive volumes of data.

To resolve this problem, we could "scale up" our systems by upgrading our existing hardware. This process is expensive.

The alternative for this issue is to distribute database load on multiple hosts whenever the load increases. This method is known as "scaling out."



NoSQL database is non-relational, so it scales out better than relational databases as they are designed with web applications in mind.

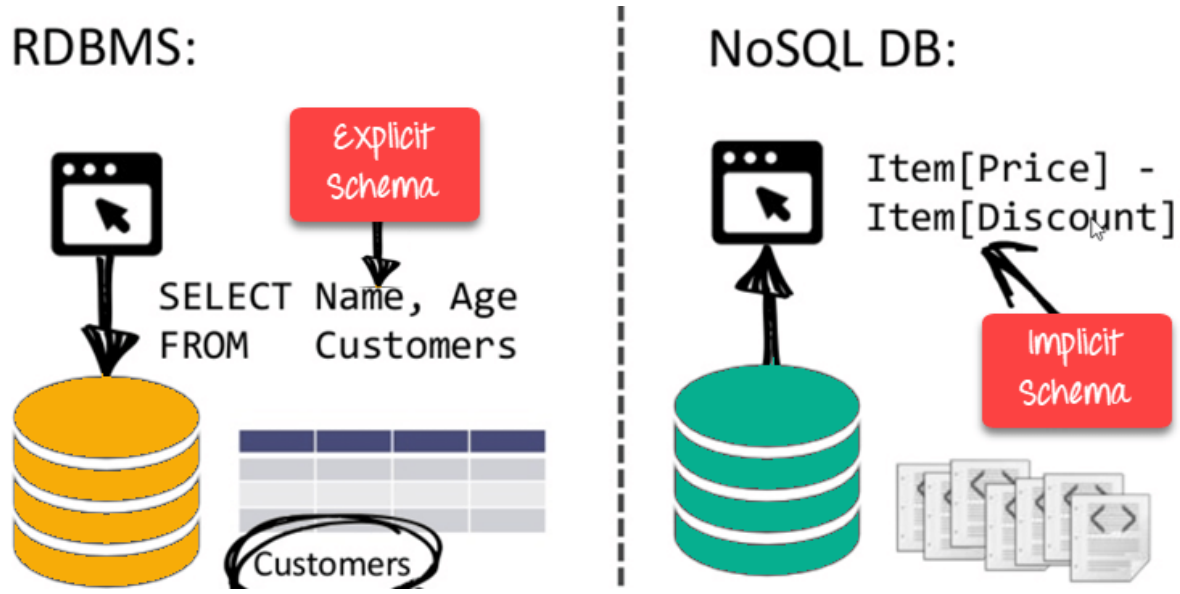
Features of NoSQL

Non-relational

- NoSQL databases never follow the relational model
- Never provide tables with flat fixed-column records
- Doesn't require object-relational mapping and data normalization
- No complex features like query languages, query planners, referential integrity joins, ACID

Schema-free

- NoSQL databases are either schema-free or have relaxed schemas
- Do not require any sort of definition of the schema of the data
- Offers heterogeneous structures of data in the same domain



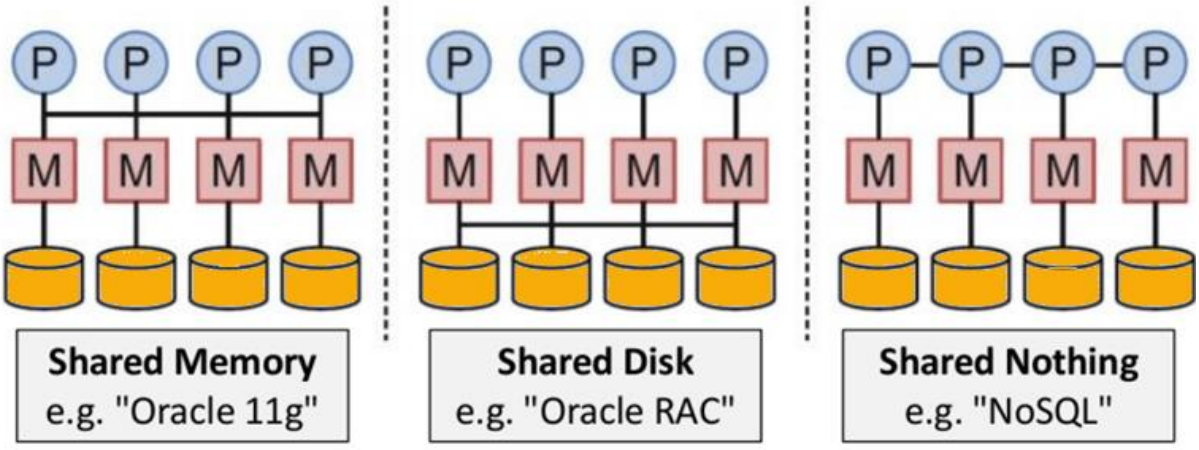
NoSQL is Schema-Free

Simple API

- Offers easy to user interfaces for storage and querying data provided
- APIs allow low-level data manipulation & selection methods
- Text-based protocols mostly used with HTTP REST with JSON
- Mostly used no standard based NoSQL query language
- Web-enabled databases running as internet-facing services

Distributed

- Multiple NoSQL databases can be executed in a distributed fashion
- Offers auto-scaling and fail-over capabilities
- Often ACID concept can be sacrificed for scalability and throughput
- Mostly no synchronous replication between distributed nodes Asynchronous Multi-Master Replication, peer-to-peer, HDFS Replication
- Only providing eventual consistency
- Shared Nothing Architecture. This enables less coordination and higher distribution.

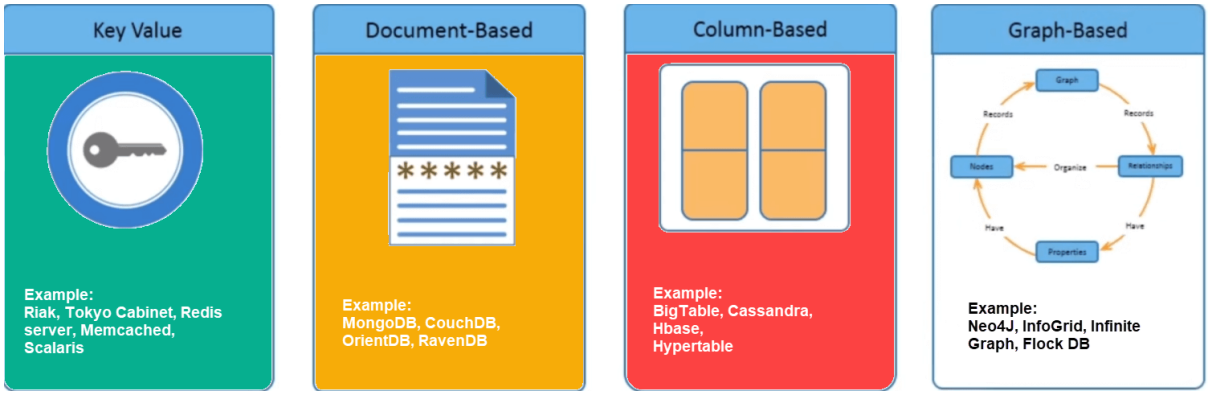


NoSQL is Shared Nothing.

Types of NoSQL Databases

NoSQL Databases are mainly categorized into four types: Key-value pair, Column-oriented, Graph-based and Document-oriented. Every category has its unique attributes and limitations. None of the above-specified database is better to solve all the problems. Users should select the database based on their product needs.
Types of NoSQL Databases:

- Key-value Pair Based
- Column-oriented Graph
- Graphs based
- Document-oriented

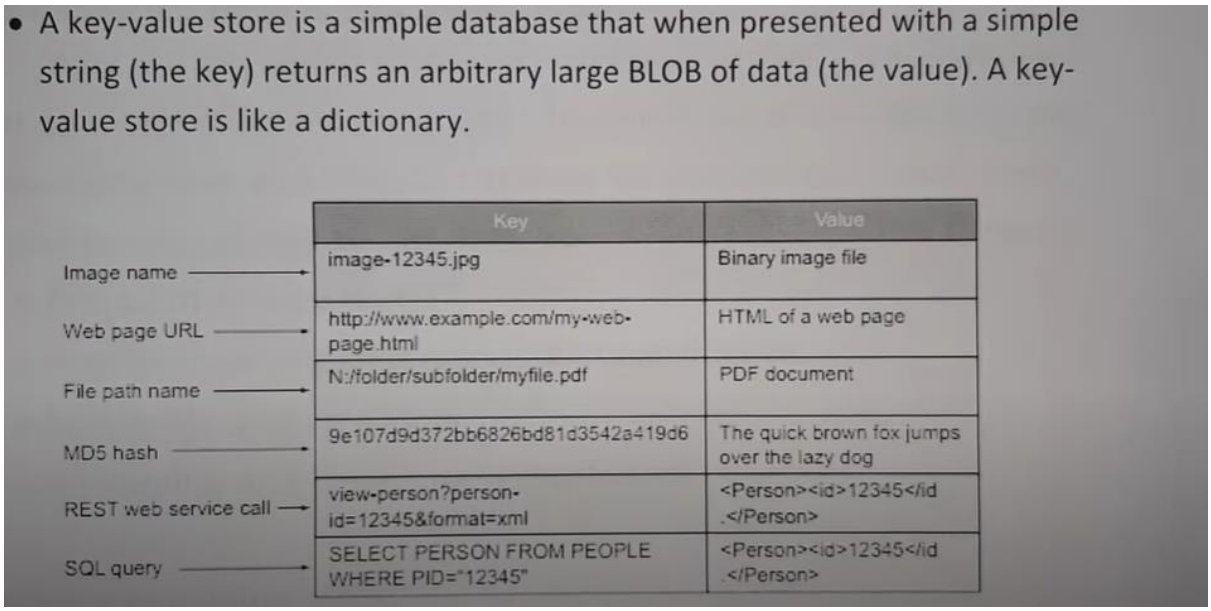


Key Value Pair Based

Data is stored in key/value pairs. It is designed in such a way to handle lots of data and heavy load.

Key-value pair storage databases store data as a hash table where each key is unique, and the value can be a JSON, BLOB(Binary Large Objects), string, etc.

Key	Value
Name	Joe Bloggs
Age	42
Occupation	Stunt Double
Height	175cm
Weight	77kg

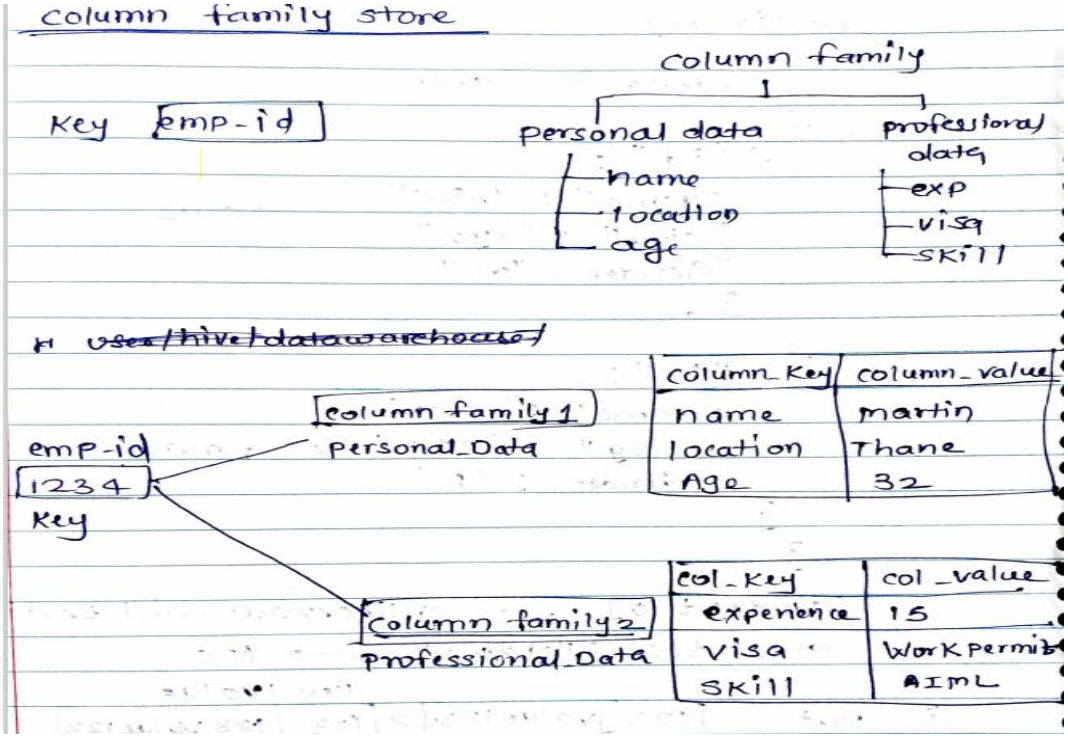


It is one of the most basic NoSQL database example. This kind of NoSQL database is used as a collection, dictionaries, associative arrays, etc. Key value stores help the developer to store schema-less data. They work best for shopping cart contents.

Redis, Dynamo, Riak are some NoSQL examples of key-value store DataBases. They are all based on Amazon's Dynamo paper.

Column-based

Column-oriented databases work on columns and are based on BigTable paper by Google. Every column is treated separately. Values of single column databases are stored contiguously.



Column based NoSQL database

They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN etc. as the data is readily available in a column.

Column-based NoSQL databases are widely used to manage data warehouses, business intelligence, CRM, Library card catalogs,

HBase, Cassandra, HBase, Hypertable are NoSQL query examples of column based database.

Document-Oriented:

Document-Oriented NoSQL DB stores and retrieves data as a key value pair but the value part is stored as a document. The document is stored in JSON or XML formats. The value is understood by the DB and can be queried.

Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data	Data	Data

Document 1

```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

Document 2

```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

Document 3

```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

Relational Vs. Document

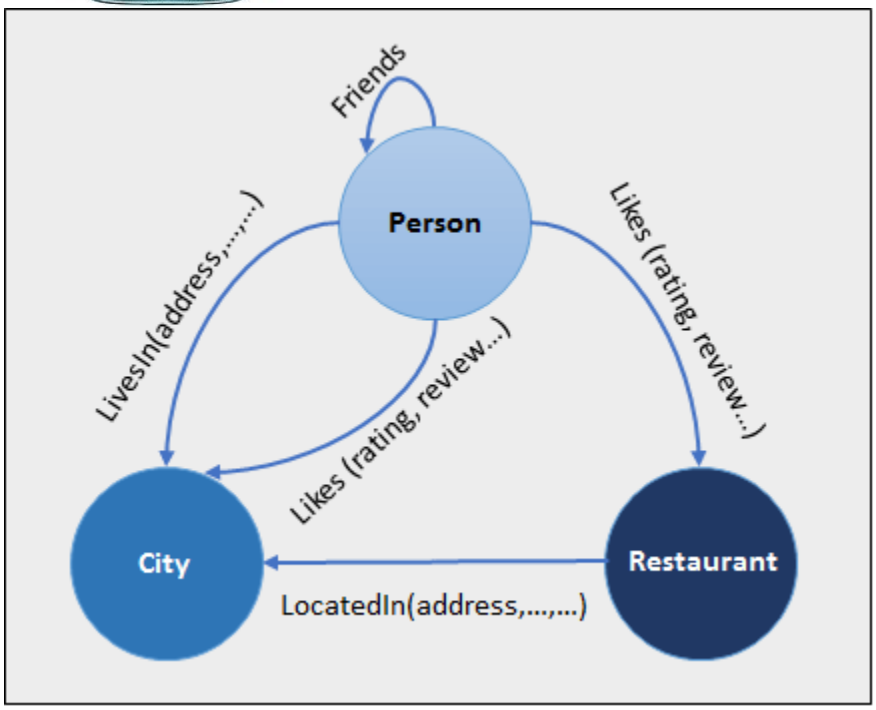
In the above diagram on your left you can see we have rows and columns, and in the right, we have a document database which has a similar structure to JSON. Now for the relational database, you have to know what columns you have and so on. However, for a document database, you have data store like JSON object. You do not require to define which make it flexible.

The document type is mostly used for CMS systems, blogging platforms, real-time analytics & e-commerce applications. It should not use for complex transactions which require multiple operations or queries against varying aggregate structures.

Amazon SimpleDB, CouchDB, MongoDB, Riak, Lotus Notes, MongoDB, are popular Document originated DBMS systems.

Graph-Based

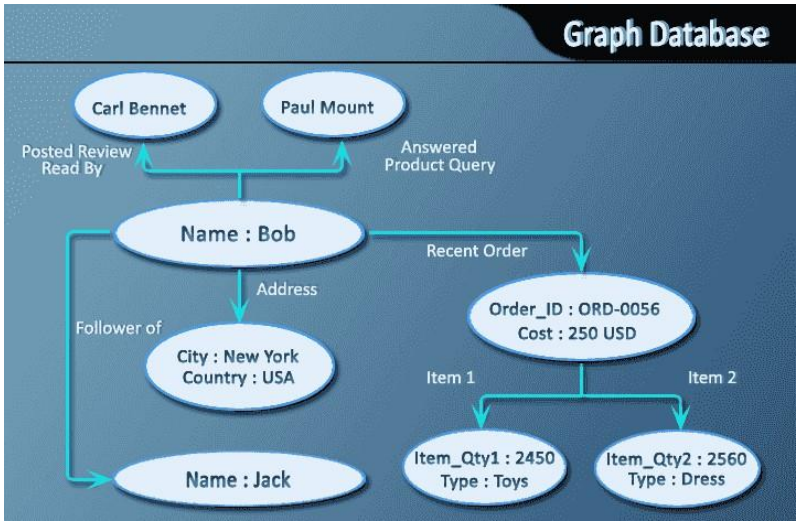
A graph type database stores entities as well the relations amongst those entities. The entity is stored as a node with the relationship as edges. An edge gives a relationship between nodes. Every node and edge has a unique identifier.



Compared to a relational database where tables are loosely connected, a Graph database is a multi-relational in nature. Traversing relationship is fast as they are already captured into the DB, and there is no need to calculate them.

Graph base database mostly used for social networks, logistics, spatial data.

Neo4J, Infinite Graph, OrientDB, FlockDB are some popular graph-based databases.



Q. What is the CAP Theorem?

CAP theorem is also called brewer's theorem. It states that is impossible for a distributed data store to offer more than two out of three guarantees

1. Consistency
2. Availability
3. Partition Tolerance

Consistency:

The data should remain consistent even after the execution of an operation. This means once data is written, any future read request should contain that data. For example, after updating the order status, all the clients should be able to see the same data.

Availability:

The database should always be available and responsive. It should not have any downtime.

Partition Tolerance:

Partition Tolerance means that the system should continue to function even if the communication among the servers is not stable. For example, the servers can be partitioned into multiple groups which may not communicate with each other. Here, if part of the database is unavailable, other parts are always unaffected.

Q. What is BASE theorem?

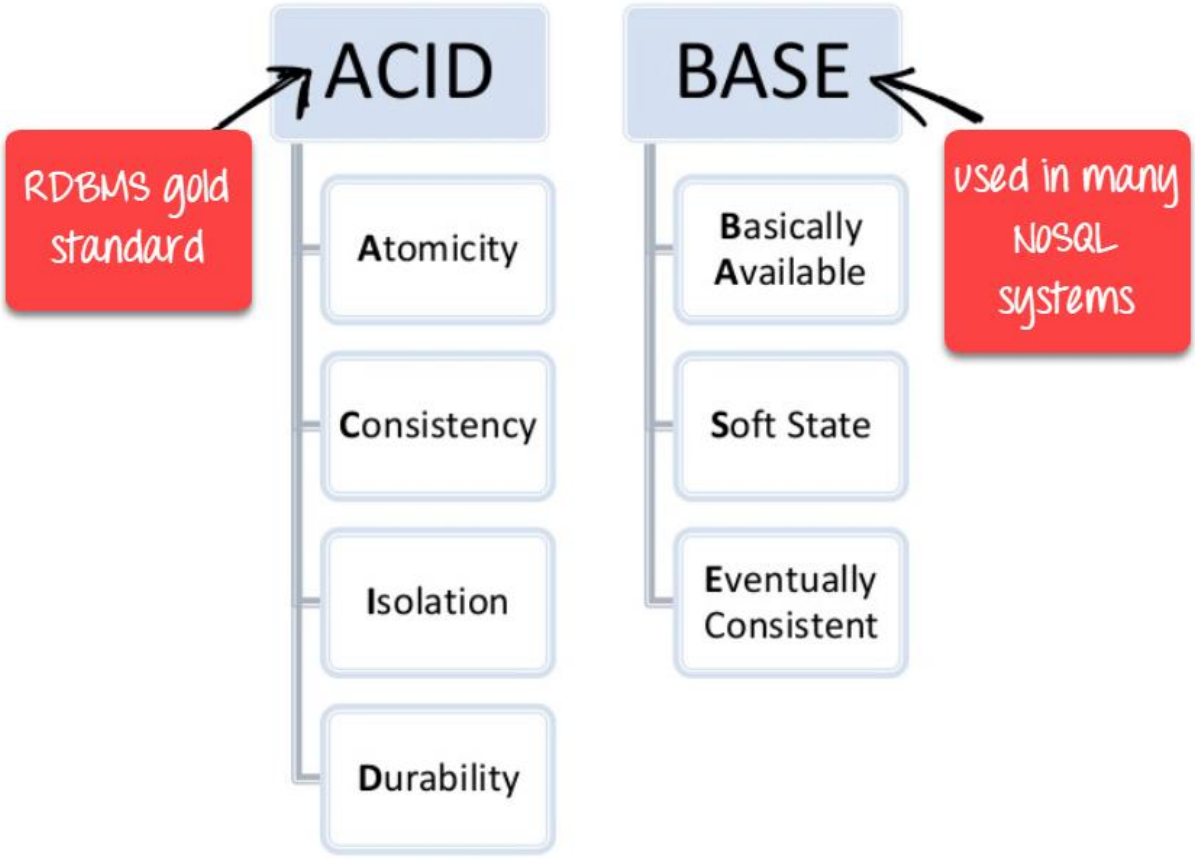
Eventual Consistency

The term "eventual consistency" means to have copies of data on multiple machines to get high availability and scalability. Thus, changes made to any data item on one machine has to be propagated to other replicas.

Data replication may not be instantaneous as some copies will be updated immediately while others in due course of time. These copies may be mutually, but in due course of time, they become consistent. Hence, the name eventual consistency.

BASE: **B**asically **A**vailable, **S**oft state, **E**ventual consistency

- Basically, available means DB is available all the time as per CAP theorem
- Soft state means even without an input; the system state may change.
- Eventual consistency means that the system will become consistent over time.



This property refers to the fact that the database system should always be available to respond to user requests, even if it cannot guarantee immediate access to all data. The database may experience brief periods of unavailability, but it should be designed to minimize downtime and provide quick recovery from failures.

Soft State

This property refers to the fact that the state of the database can change over time, even without any explicit user intervention. This can happen due to the effects of background processes, updates to data, and other factors. The database should be designed to handle this change gracefully, and ensure that it does not lead to data corruption or loss.

Eventual Consistency

This property refers to the eventual consistency of data in the database, despite changes over time. In other words, the database should eventually converge to a consistent state, even if it takes some time for all updates to propagate and be reflected in the data. This is in contrast to the immediate consistency required by traditional ACID-compliant databases.

Uses of BASE Databases

BASE databases are used in modern, highly-available, and scalable systems that handle large amounts of data. Examples of such systems include online shopping websites, social media platforms, and cloud-based services.

Advantages of NoSQL

- Don’t require a schema
- Data are replicated
- Data can be partitioned
- Cheap, easy to implement
- Easy to distribute
- Can scale up and down
- Quickly process large amounts of data
- Can handle web-scale data (relational DBs can’t)

Disadvantages of NoSQL

- No standardized schema
- No standard format for queries
- Data is generally duplicated, potential for inconsistency
- No standard language
- No guarantee of support
- Difficult to impose complicated structures
- Depend on the application layer to enforce data integrity