No-SQL

What is a NoSQL database?

NoSQL, also referred to as "not only SQL", "non-SQL", is an approach to database design that enables the storage and querying of data outside the traditional structures found in relational databases. While it can still store data found within relational database management systems (RDBMS), it just stores it differently compared to an RDBMS. The decision to use a relational database versus a non-relational database is largely contextual, and it varies depending on the use case.

Instead of the typical tabular structure of a relational database, NoSQL databases, house data within one data structure, such as JSON document. Since this non-relational database design does not require a schema, it offers rapid scalability to manage large and typically unstructured data sets.

NoSQL is also type of distributed database, which means that information is copied and stored on various servers, which can be remote or local. This ensures availability and reliability of data. If some of the data goes offline, the rest of the database can continue to run.

- **Examples of No-SQL** : MongoDB, CouchDB, CouchBase, Cassandra, HBase, Redis, Riak, Neo4J are the popular NoSQL databases examples.
- **NoSQL** databases store data in documents rather than relational tables.
- **NoSQL** Database is a non-relational Data Management System, that does not require a fixed schema. It avoids joins, and is easy to scale.
- **NoSQL** is a non-relational DMS, that does not require a fixed schema, avoids joins, and is easy to scale
- The concept of **NoSQL** databases became popular with Internet giants like Google, Facebook, Amazon, etc. who deal with huge volumes of data

- In the year 1998- Carlo Strozzi use the term **NoSQL** for his lightweight, opensource relational database
- **NoSQL** databases never follow the relational model it is either schema-free or has relaxed schemas
- Four types of **NoSQL** Database are
  - Key-value Pair Based
  - Column-oriented Graph
  - Graphs based
  - Document-oriented
- **NOSQL** can handle structured, semi-structured, and unstructured data with equal effect
- **CAP** theorem consists of three words Consistency, Availability, and Partition Tolerance

- **Features of NoSQL**
  - Non-relational
  - Schema-free
  - Simple API
  - Distributed

- **WHY WE NEED NOSQL ?**

- 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."

    - Scale-up(vertical scaling)
    - Scale-out(horizontal scaling)

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

Advantages of NoSQL

Each type of NoSQL database has strengths that make it better for specific use cases. However, they all share the following advantages for developers and create the framework to provide better service customers, including:

- **Cost-effectiveness:** It is expensive to maintain high-end, commercial RDBMS. They require the purchase of licenses, trained database managers, and powerful hardware to scale vertically. NoSQL databases allow you to quickly scale horizontally, better allocating resources to minimize costs.
- **Flexibility:** Horizontal scaling and a flexible data model also mean NoSQL databases can address large volumes of rapidly changing data, making them great for agile development, quick iterations, and frequent code pushes.
- **Replication:** NoSQL replication functionality copies and stores data across multiple servers. This replication provides data reliability, ensuring access during down time and protecting against data loss if servers go offline.
-

- **DISADVANTAGES OF NOSQL**

- No standardization rules
- Limited query capabilities
- RDBMS databases and tools are comparatively mature
- It does not offer any traditional database capabilities, like consistency when multiple transactions are performed simultaneously.
- When the volume of data increases it is difficult to maintain unique values as keys become difficult
- Doesn't work as well with relational data

- The learning curve is stiff for new developers
- Open source options so not so popular for enterprises.