INTRODUCTION TO NOSQL

What is NoSQL?

- NoSQL (commonly known as "Not Just SQL") represents a completely different database framework that can achieve high-performance and agile processing of large-scale information.
- In other words, it is a database infrastructure, very suitable for the huge needs of big data.
- The efficiency of NoSQL can be achieved because, unlike highly structured relational databases, NoSQL databases are inherently unstructured, which makes up for the strict consistency requirements for speed and agility.
- NoSQL focuses on the concept of distributed databases, where unstructured data can be stored on multiple processing nodes, and usually on multiple servers.
- This distributed architecture allows NoSQL databases to scale horizontally; as the data continues to grow, just add more hardware to keep up without reducing performance.
- NoSQL Distributed Database Infrastructure has always been a solution for handling some of the largest data warehouses on the planet, such as Google, Amazon, and the Central Intelligence Agency.

Where is NoSQL used?

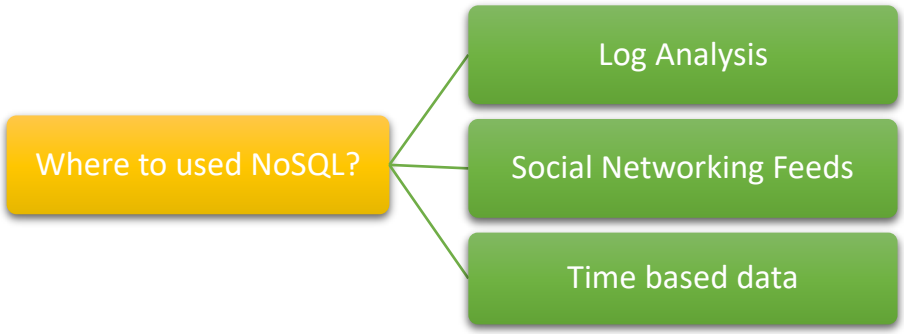
- NoSQL databases are widely used in big data and other real-time web applications.
- NoSQL databases is used to stock log data which can then be pulled for analysis. Likewise, it is used to store social media data and all such data which cannot be stored and analyzed comfortably in RDBMS.

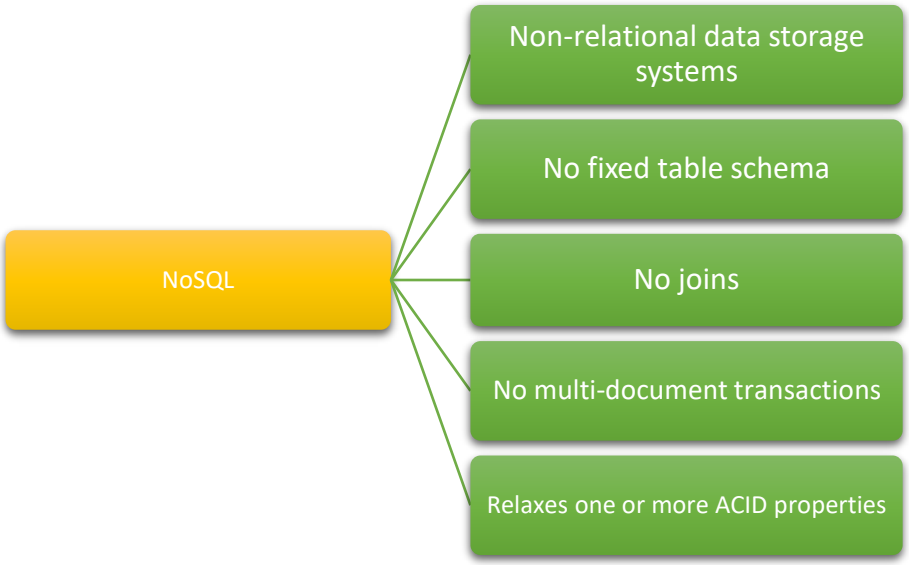
Features and advantages of NoSQL

Few features of NoSQL databases are as follows:

1. They are open source.
2. They are non-relational.
3. They are distributed.
4. They are schema-less.
5. They are cluster friendly.
6. They are born out of 21st century web applications.

- NoSQL databases provide various important advantages over traditional relational databases.
- A few core features of NoSQL are listed here, which apply to most NoSQL databases.





Advantages of NoSQL Database

- Schema Agnostic
  - NoSQL databases are schema agnostic.
  - Easy to designing your schema before you can store data in NoSQL databases.
  - You can start coding, and store and retrieve data without knowing how the database stores and works internally.
  - Schema agnosticism may be the most significant difference between NoSQL and relational databases.

- Scalability
  - NoSQL databases support horizontal scaling methodology that makes it easy to add or reduce capacity quickly without tinkering with commodity hardware.
  - This eliminates the tremendous cost and complexity of manual sharing that is necessary when attempting to scale RDBMS.
- Performance
  - Some databases are designed to operate best (or only) with specialized storage and processing hardware.
  - With a NoSQL database, you can increase performance by simply adding cheaper servers, called commodity servers.
  - This helps organizations to continue to deliver reliably fast user experiences with a predictable return on investment for adding resources again, without the overhead associated with manual sharing.
- High Availability
  - NoSQL databases are generally designed to ensure high availability and avoid the complexity that comes with a typical RDBMS architecture, which relies on primary and secondary nodes.
  - Some 'distributed' NoSQL databases use a masterless architecture that automatically distributes data equally among multiple resources so that the application remains available for both read and write operations, even when one node fails.

Using NoSQL to Manage Big Data

- The main reason behind organization moving towards a NoSQL solution and leaving the RDBMS system behind is the requirement to analyze a large volume of data.
- It is any business problem which could be so large and single processor cannot manage it.
- We need to move single processor environment to distributed computing environment due to big data problem.
- It has own problems and challenges while solving big data problems.

Big Data Use-case

Typical big data use-cases:

1. Bulk Image Processing
2. Public Web Page Data
3. Remote Sensor Data
4. Event Log Data
5. Mobile Phone Data
6. Social Media Data
7. Game Data

Bigdata Use-case Solutions

- Scaling linearly with growing data size by becoming an efficient with input and output.
- Organizations not able to afford to hire many people to run the server, so becoming operationally efficient.
- Not every business can afford a full time java developer to write on demand queries, so its require that reports and analysis be performed by nonprogrammers using simple tools.
- Meeting the challenges of distributed computing, with consideration of latency between systems and eventual node failures.
- Meeting both the need of overnight batch processing economy of scale and time critical event processing.

SQL Vs. NoSQL

SQL	**NoSQL**
Relational database	Non-relational, distributed database
Relational model	Model-less approach
Pre-defned schema	Dynamic schema for unstructured data
Table based databases	Document-based or graph-based or wide column store or key–value pairs databases
Vertically scalable (by increasing system resources)	Horizontally scalable (by creating a cluster of commodity machines)
Uses SQL	Uses UnQL (Unstructured Query Language)

Not preferred for large datasets	Largely preferred for large datasets
Not a best ft for hierarchical data	Best ft for hierarchical storage as it follows the key–value pair of storing data similar to JSON (Java Script Object Notation)
Excellent support from vendors	Relies heavily on community support
Supports complex querying and data keeping needs	Does not have good support for complex querying
Can be confgured for strong consistency	Few support strong consistency (e.g., MongoDB), some others can be confgured for eventual consistency (e.g., Cassandra)
Examples: Oracle, DB2, MySQL, MS SQL, PostgreSQL, etc.	Examples: MongoDB, HBase, Cassandra, Redis, Neo4j, CouchDB, Couchbase, Riak, etc.