**SQL vs NoSQL**

**What is SQL?**

SQL databases, also known as Relational Database Management Systems (RDBMS), use structured tables to store data. They rely on a predefined schema that determines the organization of data within tables, making them suitable for applications that require a fixed, consistent structure.

* **Structured Data**: Data is organized in tables with rows and columns, making it easy to relate different types of information.
* **ACID Compliance**: SQL databases follow the **ACID** properties (Atomicity, Consistency, Isolation, Durability) to ensure reliable transactions and data integrity.
* **Examples**: Popular SQL databases include **MySQL**, **PostgreSQL**, **Oracle**, and **MS SQL Server**.

**What is NoSQL?**

NoSQL, which stands for Not Only SQL, is a database management system approach used to ingest, store, and retrieve unstructured and semi-structured data. Unlike traditional relational databases (SQL), NoSQL databases can handle non-tabular, non-relational data models and often support SQL-like query languages.

This flexibility allows NoSQL databases to store data in its native format (e.g., .txt, .JPG, .MP3) without needing to convert it into a structured schema.

* **Unstructured data** is data that lacks a predefined format or consistent organization. It can change or update frequently and does not fit neatly into tables.

**Examples**: Social media posts, Images, Audio files, videos, maps

* **CAP Theorem**: NoSQL databases are designed based on the **CAP theorem** (Consistency, Availability, Partition Tolerance), which prioritizes **availability** and **partition tolerance** over strict consistency.
* **Examples**: Well-known NoSQL databases include **MongoDB**, **Cassandra**, **CouchDB**, and **HBase**.

**Types of NoSQL Databases**

**1. Document Databases**

* Store data as **JSON-like documents**.
* Schema-less: Each document can have a different structure.
* Data can be nested.
* Do not require an ORM (Object Relational Mapper).
* **Example**: MongoDB

**2. Key-Value Databases**

* Store data as key-value pairs.
* Best for fast lookups where values are retrieved via unique keys.
* Highly performant for caching and session storage.
* Examples: AWS DynamoDB, ScyllaDB

**3. Column-Family Stores**

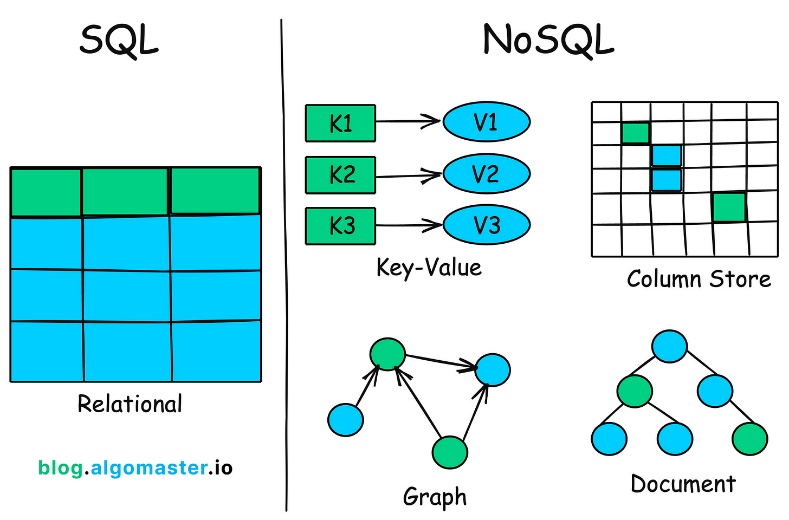
* Organize data into columns instead of rows.
* Suitable for wide datasets with large volumes and sparse fields.
* Allow horizontal scaling for handling huge workloads.
* **Examples**: Apache Cassandra, HBase

**4. Graph Databases**

* Store data in nodes and edges to represent entities and relationships.
* Excellent for relationship-heavy data like social networks, fraud detection, or recommendation systems.
* **Examples**: Neo4j, AWS Neptune, Kibana

**SQL vs NoSQL: Which is Faster?**

* **SQL Databases:**SQL databases perform well for complex queries, structured data, and systems requiring data consistency and integrity. However, as the volume of data grows, they may struggle with scalability and could require significant infrastructure upgrades.
* **NoSQL Databases:**  
  NoSQL databases excel in scenarios that demand high performance and scalability. Thanks to their horizontal scalability (adding more servers instead of upgrading a single one), they handle large datasets and high-velocity workloads better.  
  For example, MongoDB or Cassandra is a common choice when dealing with big data or high-traffic applications.



**Key Differences: SQL vs NoSQL**

| **Feature** | **SQL Databases (Relational)** | **NoSQL Databases (Non-Relational)** |
| --- | --- | --- |
| **Data Model** | Tables (Rows & Columns) | Documents, Key-Value, Columns, Graphs |
| **Data Type** | Structured | Unstructured & Semi-Structured |
| **Schema** | Rigid, Predefined | Flexible, Dynamic |
| **Storage Format** | Fixed tabular format | Native format (JSON, images, files, etc.) |
| **Scalability** | Vertical (scale-up: more powerful servers) | Horizontal (scale-out: more servers) |
| **Use Case** | Banking, ERPs, structured data apps | Big Data, Real-time apps, IoT, social media platforms |
| **Examples** | MySQL, PostgreSQL, Oracle, SQL Server | MongoDB, Cassandra, Neo4j, DynamoDB |

**MongoDB:**

**MongoDB** is a **NoSQL document database** that stores data in a **flexible, JSON-like format called BSON** (Binary JSON). Unlike traditional relational databases that store data in tables with rows and columns, MongoDB stores data in **documents** inside **collections**. This makes MongoDB more flexible and scalable, especially for modern web and mobile applications.

The following is an example of a **MongoDB document**:

{

title: "Post Title 1",

body: "Body of post.",

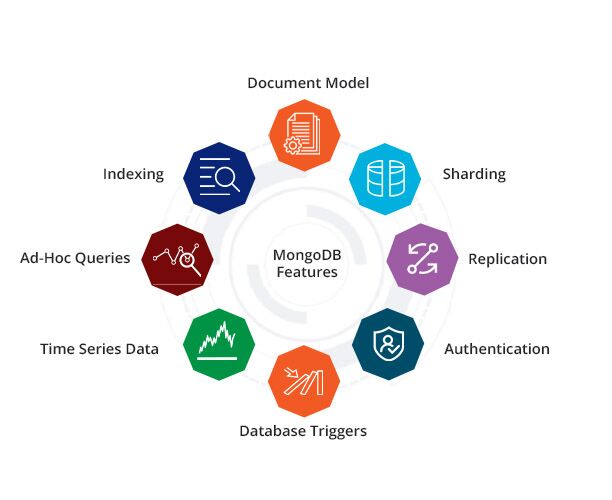
category: "News",

likes: 1,

tags: ["news", "events"],

date: Date()

}



**Key Features of MongoDB:**

**1. Document-Oriented Model**

* Stores data in flexible, JSON-like BSON documents.
* Documents are grouped into collections, allowing schema flexibility.
* Ideal for modeling real-world objects and working with nested data.

**2. Horizontal Scalability via Sharding**

* Supports sharding to distribute large datasets across multiple servers.
* Enhances performance and availability for high-traffic applications.
* Uses mongos to route queries to the correct shard based on shard key.

**3. High Availability via Replication**

* Uses replica sets to ensure data redundancy and fault tolerance.
* Automatic failover if the primary node goes down.
* Read scaling possible using read preferences on secondary nodes.

**4. Fully Managed Cloud Service - MongoDB Atlas**

* Deploy MongoDB on AWS, Azure, or Google Cloud.
* Built-in features like global clusters, auto-scaling, and backups.
* Offers database triggers, real-time analytics, and serverless options.

**5. Ad-Hoc Querying & Rich Query Language**

* Supports field-level, range, regex, geo, and text search.
* Real-time aggregation framework for advanced analytics.
* Dynamic query support even without predefined schemas.

**6. Powerful Indexing**

* Indexes on any field (including arrays and nested fields).
* Supports compound, TTL, text, geospatial, and wildcard indexes.
* Significantly improves query performance.

**7. Load Balancing**

* Built-in load distribution for read and write operations.
* Works in conjunction with replication and sharding.
* No need for an external load balancer.

**8. Authentication & Security**

* Supports SCRAM, LDAP, x.509, and Kerberos authentication.
* Role-Based Access Control (RBAC) and TLS/SSL encryption.
* Fine-grained auditing and IP whitelisting in MongoDB Atlas.