**SQL VS NoSQL Databases**

**- Ashwin Harish P**

**What is SQL Databases:**

SQL databases, also known as relational databases, are systems that store collections of tables and organize structured sets of data in a tabular columns-and-rows format, similar to that of a spreadsheet. The databases are built using structured query language (SQL), the query language that not only makes up all relational databases and relational database management systems (RDBMS), but also enables them to “talk to each other”.  It has

1. **ACID Compliance**

SQL databases are typically **ACID-compliant**, which means they guarantee:

* **Atomicity** – All operations in a transaction succeed or none do.
* **Consistency** – Data remains valid before and after transactions.
* **Isolation** – Transactions don’t interfere with each other.
* **Durability** – Once a transaction is committed, it is permanently stored.

1. **Data Integrity and Constraints**

SQL databases support **constraints** (e.g., PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, CHECK) to enforce **data integrity**, ensuring the database remains accurate and reliable.

1. **Relational Schema Design**

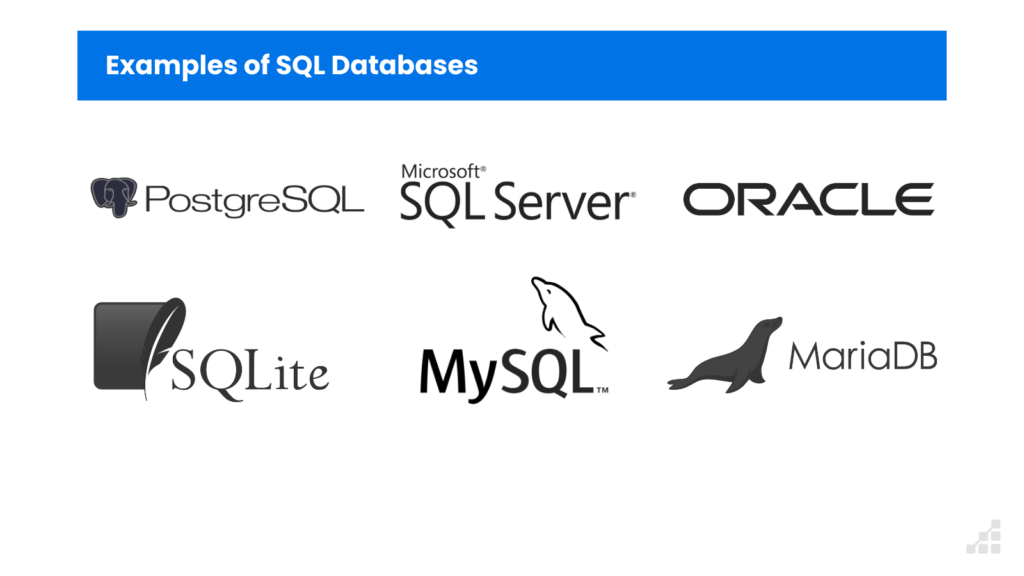
A well-designed SQL database follows normalization principles to reduce redundancy and ensure efficient organization of data through well-defined relationships between tables.

**Benefits of using SQL databases**

SQL databases offer countless benefits that make data management easy, including:

* **Optimized performance.** Due to high processing speeds and minimal storage usage, relational databases can retrieve enormous batches of data transactions in near-real time, then insert, delete, or modify that data almost instantly.
* **Seamless collaboration**. With an SQL database, you have the ability modify the schema in real time. This means any user can add, remove, and modify the data inside each column and row, or even remove the columns and rows themselves, all without disrupting the workflow of other users.
* **Clean, reliable data**. SQL databases not only maintain consistency across all server instances, but they also reduce redundancies in the data related to insertions and deletions—which is essential for maintaining accuracy and speed when processing a large batch of transactions.
* **Easy to access and learn.** SQL is written in plain English and not in complex code, which makes it easier for the average user to select, insert, update, and delete data, so long as they learn the functions and the syntax.
* **Community support.** Because SQL is an open-source language, it is supported by a worldwide community of developers who provide continuous updates, documentation, and troubleshooting help when the need arises.

**Examples of SQL Databases:**



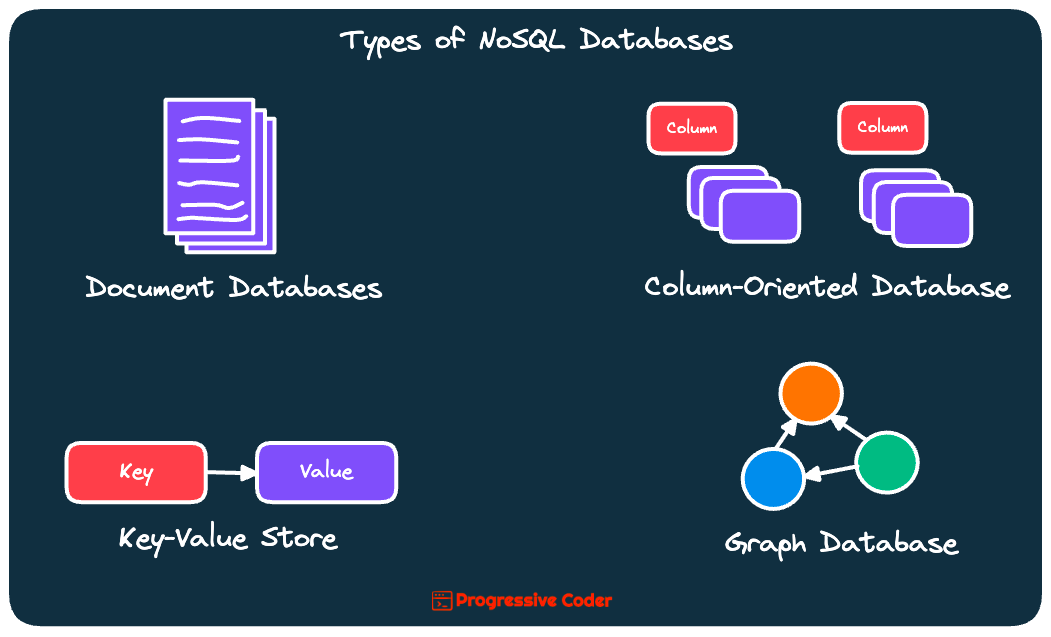
**What is NoSQL Databases:**

Unlike relational databases, which uses Structured Query Language, NoSQL databases don't have a universal query language. Instead, each type of NoSQL database typically has its unique query language. Traditional relational databases follow ACID (Atomicity, Consistency, Isolation, Durability) principles, ensuring strong consistency and structured relationships between data.

However, as applications evolved to handle big data, real-time analytics, and distributed environments, NoSQL emerged as a solution with:

* Scalability – Can scale horizontally by adding more nodes instead of upgrading a single machine.
* Flexibility – Supports unstructured or semi-structured data without a rigid schema.
* High Performance – Optimized for fast read/write operations with large datasets.
* Distributed Architecture – Designed for high availability and partition tolerance in distributed systems.

**Types of NoSQL Databases**



**Benefits of using NoSQL databases**

* **High scalability:**NoSQL databases use sharding for horizontal scaling. Partitioning of data and placing it on multiple machines in such a way that the order of the data is preserved is sharding. Vertical scaling means adding more resources to the existing machine whereas horizontal scaling means adding more machines to handle the data. Vertical scaling is not that easy to implement but horizontal scaling is easy to implement. Examples of horizontal scaling databases are MongoDB, Cassandra, etc. NoSQL can handle a huge amount of data because of scalability, as the data grows NoSQL scalesThe auto itself to handle that data in an efficient manner.
* **Flexibility:** NoSQL databases are designed to handle unstructured or semi-structured data, which means that they can accommodate dynamic changes to the data model. This makes NoSQL databases a good fit for applications that need to handle changing data requirements.
* **High availability:** The auto, replication feature in NoSQL databases makes it highly available because in case of any failure data replicates itself to the previous consistent state.
* **Scalability:**NoSQL databases are highly scalable, which means that they can handle large amounts of data and traffic with ease. This makes them a good fit for applications that need to handle large amounts of data or traffic
* **Performance:** NoSQL databases are designed to handle large amounts of data and traffic, which means that they can offer improved performance compared to traditional relational databases.

**Examples of NoSQL Databases:**



**SQL VS NoSQL Databases**

| **Aspect** | **SQL (Relational)** | **NoSQL (Non-relational)** |
| --- | --- | --- |
| **Data Structure** | Tables with rows and columns | Document-based, key-value, column-family, or graph-based |
| **Schema** | Fixed schema (predefined structure) | Flexible schema (dynamic and adaptable) |
| **Scalability** | Vertically scalable (upgrading hardware) | Horizontally scalable (adding more servers) |
| **Data Integrity** | ACID-compliant (strong consistency) | BASE-compliant (more available, less consistent) |
| **Query Language** | SQL (Structured Query Language) | Varies (e.g., MongoDB uses its own query language) |
| **Performance** | Efficient for complex queries and transactions | Better for large-scale data and fast read/write operations |
| **Use Case** | Best for transactional systems (banking, ERP, etc.) | Ideal for big data, real-time web apps, and data lakes |
| **Examples** | MySQL, PostgreSQL, Oracle, MS SQL Server | MongoDB, Cassandra, CouchDB, Neo4j |

**When to Choose SQL?**

SQL databases are well-suited for use cases where:

* Data consistency and transactional integrity are critical (e.g., banking systems, customer relationship management).
* The application needs a well-defined schema and structured data.
* Complex queries and relational data are involved.
* Applications requiring multi-row transactions (such as inventory management) benefit from SQL’s robust features.

**When to Choose NoSQL?**

NoSQL databases are a better choice when:

* You need to handle large, unstructured data sets, like social media data or logs.
* The application requires horizontal scalability to accommodate high traffic and big data.
* There is a need for real-time data processing and flexible data models (e.g., a content management system).
* You are dealing with applications requiring frequent changes in data structures.

**Features of MongoDB**

MongoDB offers a wide range of features that make it a preferred choice for modern applications.

**1. Schema-less Database**

Unlike traditional relational databases, MongoDB collections:

* Allow different structures within the same collection.
* Do not require fixed column definitions.
* Enable easy updates and modifications.

**2. Document Oriented**

In MongoDB, all the data stored in the documents instead of tables like in RDBMS. In these documents, the data is stored in fields(key-value pair) instead of rows and columns which make the data much more flexible in comparison to RDBMS. And each document contains its unique object id.

**3. Indexing**

In MongoDB database, every field in the documents is indexed with primary and secondary indices this makes easier and takes less time to get or search data from the pool of the data. If the data is not indexed, then database search each document with the specified query which takes lots of time and not so efficient.

**4. Scalability**

MongoDB provides horizontal scalability with the help of sharding. Sharding means to distribute data on multiple servers, here a large amount of data is partitioned into data chunks using the shard key, and these data chunks are evenly distributed across shards that reside across many physical servers. It will also add new machines to a running database.

**5. Replication**

MongoDB provides high availability and redundancy with the help of replication, it creates multiple copies of the data and sends these copies to a different server so that if one server fails, then the data is retrieved from another server.

**Use Cases of MongoDB**

**1. Handling Unstructured or Semi-Structured Data**

MongoDB stores data in BSON (Binary JSON) format, making it ideal for:

* JSON-like documents
* Varying data fields across records
* Dynamic schemas

**2. Real-Time Analytics**

MongoDB supports high-speed read/write operations, making it suitable for:

* Real-time dashboards
* Event logging systems
* Monitoring applications

**3. Content Management Systems (CMS)**

Great for storing:

* Articles, blogs, comments
* Tags, categories, metadata
* Media files & descriptions

Its flexibility allows different types of content in the same collection.

**Advantages of MongoDB**

* It is a schema-less NoSQL database. We need not to design the schema of the database when we are working with MongoDB.
* It does not support join operation.
* It provides great flexibility to the fields in the documents.
* It contains heterogeneous data.
* It provides high performance, availability, scalability.

**Disadvantages of MongoDB**

* High Memory Usage - Requires additional storage
* No Complex Joins - Relies on embedding or referencing instead
* Limited Document Size - Maximum 16MB per document
* Nesting Limits - Supports up to 100 levels of nested documents