**Difference between SQL and NoSQL**

Choosing between **SQL**(Structured Query Language) and **NoSQL**(Not Only SQL) databases is a critical decision for **developers**, **data engineers**, and organizations looking to handle large datasets effectively. Both **database types** have their **strengths**and **weaknesses**, and understanding the key differences can help us make an informed decision based on our project’s needs.

In this article, we will explain the key differences between **SQL**and **NoSQL**databases, including their **structure**, **scalability**, and **use cases**. We will also explore which database is more suitable for various types of applications and provide insights into when to choose SQL over NoSQL and vice versa.

**Differences Between SQL and NoSQL**

| **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 |

**1. Type**

SQL databases are primarily called Relational Databases [(RDBMS)](https://www.geeksforgeeks.org/rdbms-full-form/); whereas NoSQL databases are primarily called non-relational or distributed databases.

**2. Language**

SQL databases define and manipulate data-based [structured query language (SQL)](https://www.geeksforgeeks.org/structured-query-language/). Seeing from a side this language is extremely powerful. SQL is one of the most **versatile**and **widely-used options** available which makes it a **safe choice**, especially for great **complex queries**. But from another side, it can be restrictive.

SQL requires you to use predefined [schemas](https://www.geeksforgeeks.org/create-schema-in-sql-server/) to determine the structure of your data before you work with it. Also, all of our data must follow the same structure. This can require significant **up-front preparation** which means that a change in the structure would be both difficult and disruptive to your whole system.

**3. Scalability**

In almost all situations SQL databases are vertically scalable. This means that you can increase the load on a single server by increasing things like [RAM](https://www.geeksforgeeks.org/random-access-memory-ram/), [CPU,](https://www.geeksforgeeks.org/central-processing-unit-cpu/) or [SSD](https://www.geeksforgeeks.org/introduction-to-solid-state-drive-ssd/). But on the other hand, NoSQL databases are **horizontally scalable**. This means that you handle more traffic by **sharing**, or adding more servers in your **NoSQL database**.

It is similar to adding more floors to the same building versus **adding more buildings** to the neighborhood. Thus NoSQL can ultimately become larger and more powerful, making these databases the preferred choice for large or ever-changing data sets.

**4. Structure**

**SQL databases** are table-based on the other hand **NoSQL databases** are either **key-value pairs**, **document-based**, **graph databases**, or **wide-column stores**. This makes relational SQL databases a better option for applications that require **multi-row transactions** such as an accounting system or for legacy systems that were built for a relational structure.

Here is a simple example of how a structured data with rows and columns vs a non-structured data without definition might look like. A product table in SQL **db**might accept data looking like this:

{

"id": "101",

"category":"food"

"name":"Apples",

"qty":"150"

}

Whereas a unstructured NOSQL DB might save the products in many variations without constraints to change the underlying table structure

Products=[

{

"id":"101:

"category":"food",,

"name":"California Apples",

"qty":"150"

},

{

"id":"102,

"category":"electronics"

"name":"Apple MacBook Air",

"qty":"10",

"specifications":{

"storage":"256GB SSD",

"cpu":"8 Core",

"camera": "1080p FaceTime HD camera"

}

}

]

**5. Property followed**

SQL databases follow [ACID properties](https://www.geeksforgeeks.org/acid-properties-in-dbms/) (Atomicity, Consistency, Isolation, and Durability) whereas the NoSQL database follows the Brewers [CAP theorem](https://www.geeksforgeeks.org/the-cap-theorem-in-dbms/) (Consistency, Availability, and Partition tolerance).

**6. Support**

Great support is available for all **SQL databases** from their vendors. Also, a lot of independent consultants are there who can help you with SQL databases for very large-scale deployments but for some **NoSQL databases** you still have to rely on community support and only limited outside experts are available for setting up and deploying your**large-scale NoSQL deploy**.

**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](https://www.geeksforgeeks.org/acid-transactions-in-mongodb/)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 databases, on the other hand, are designed to handle **unstructured or semi-structured data**. Unlike SQL databases, NoSQL offers **dynamic schemas** that allow for more flexible data storage, making them ideal for handling massive volumes of data from various sources.

* **Flexible Schema**: NoSQL [databases](https://www.geeksforgeeks.org/what-is-database/)allow the storage of data without a predefined structure, making them more adaptable to changing data requirements.
* **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**.

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

* **SQL Databases**: Generally, 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 may require significant infrastructure upgrades.
* **NoSQL Databases**: NoSQL databases excel in scenarios that demand **high performance** and **scalability**. Because of their **horizontal scalability** (accommodating more servers), they handle large amounts of data and high-velocity workloads better. For instance, MongoDB or Cassandra is a common choice when dealing with [big data](https://www.geeksforgeeks.org/what-is-big-data/) or applications with high traffic.

**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**.

**Conclusion**

Both [SQL](https://www.geeksforgeeks.org/what-is-sql/)and NoSQL databases offer unique **advantages**, depending on the **application’s requirements**. SQL databases are great for structured, relational data where consistency and complex queries are a priority. On the other hand, [NoSQL databases](https://www.geeksforgeeks.org/types-of-nosql-databases/) are better suited for **flexible**, **large-scale**, unstructured data handling and fast, scalable performance. **SQL** is ideal for data with **well-defined relationships** and **consistency requirements**.