SQL

**SQL** is a **standard programming language** specifically designed for **managing and interacting with relational databases**. It allows users to **create, read, update, and delete** data — often shortened as **CRUD operations**.

SQL is used to:

* Define **how data is stored** (tables, columns, data types)
* **Query** the database to retrieve specific information
* **Insert** new records
* **Update** existing data
* **Delete** unwanted data
* Control access to data using **permissions**
* Ensure **data integrity** through constraints and rules

NoSQL

**NoSQL** stands for "**Not Only SQL**."

NoSQL is a type of database that provides a flexible way to store and retrieve data that doesn’t require fixed table schemas, using various data models such as key-value, document, column-family, or graph formats.

It refers to a broad class of **non-relational databases** that are designed to handle **large volumes of data**, especially when that data is **unstructured**, **semi-structured**, or **rapidly changing**.

When To Use What

| **Requirement** | **Use SQL** | **Use NoSQL** |
| --- | --- | --- |
| Fixed schema | ✅ | ❌ |
| Complex joins/relationships | ✅ | ❌ |
| ACID transactions | ✅ | ⚠️ Some NoSQL support |
| Flexible or dynamic data structure | ❌ | ✅ |
| Massive horizontal scalability | ⚠️ Difficult | ✅ |
| Fast development with changing data | ❌ | ✅ |
| Large volumes of unstructured data | ❌ | ✅ |

Advantages And Disadvantages of SQL

SQL

**Examples**: MySQL, PostgreSQL, Oracle, Microsoft SQL Server

### **Advantages:**

1. **Structured & Reliable**: Follows a well-defined schema and ACID (Atomicity, Consistency, Isolation, Durability) properties – great for data integrity.
2. **Powerful Query Language**: SQL is a mature, standard language for complex queries and joins.
3. **Data Relationships**: Ideal for relational data – supports joins and normalization.
4. **Mature Ecosystem**: Well-established tools, documentation, and community support.

**Disadvantages:**

1. **Scalability Limits**: Harder to scale horizontally (across many servers); vertical scaling can be expensive.
2. **Rigid Schema**: Changing the structure (schema) of the database can be complex and time-consuming.
3. **Performance with Big Data**: Slower with large volumes of unstructured or semi-structured data.

NoSQL

**Types**: Document (MongoDB), Key-Value (Redis), Column-family (Cassandra), Graph (Neo4j)

### **Advantages:**

1. **Flexible Schema**: Handles unstructured or semi-structured data; easy to change data structure.
2. **Scalability**: Designed for horizontal scaling (across many machines), making them better for big data and distributed systems.
3. **Performance**: High read/write throughput, especially for large-scale or real-time applications.
4. **Variety of Models**: Supports different data models (documents, graphs, key-value, etc.), allowing for use-case-specific design.

### **Disadvantages:**

1. **Limited Querying**: Lacks the power and flexibility of SQL joins and complex queries.
2. **Eventual Consistency**: Many NoSQL databases prioritize availability and partition tolerance over strong consistency.
3. **Less Standardization**: No unified query language; each NoSQL database has its own syntax and tools.
4. **Data Integrity**: No built-in support for ACID transactions (though some modern NoSQL databases are improving this).