SQL & NoSQL

Given that approximately 2.5 quintillion bytes of data are generated, globally, each day (BDAN.com, 2024), this market increase makes perfect sense — and is one of the driving factors in exponential database systems development and usage.

Two of the most commonly used database types are **SQL** databases (e.g., relational databases) and **NoSQL** databases.

**What is SQL(Structured Query Language )?**

SQL, which stands for Structured Query Language, is a domain-specific programming language (e.g., a language targeted to a specific task or problem) that is commonly used for tasks such as inserting, updating, querying, and deleting data within a database. SQL is also used to create and modify database schemas (e.g., data formatting rules, table/index structure ) as well as define database access and administration parameters.

For this Structured data is used which is the data that is organized in a consistent, predefined format and often consists of alphanumeric characters.

Examples include financial transactions, inventory records, or customer lists which are often stored in SQL databases (e.g., relational databases).

A computer with a computer and a group of people

AI-generated content may be incorrect.

Now what is SQL Database then?

When the term "SQL database" is used, it refers to a type of database where SQL is the primary programming language used to create and manage that database. SQL application programming interfaces (APIs) contain groups of functions that enable developers to execute and manage database operations without having to create individual SQL commands over and over.

Regardless of whether a SQL database is used to store transactions for a retailer or financial information for a corporation, SQL databases fall under a type of database referred to as relational databases.

*Relational databases*

Relational databases, or relational database management systems (RDBMSs), store data within rows and columns which are used to form tables. A relationship between the two tables (or more) can be created using a foreign key. These foreign keys (e.g., unique identifiers) maintain predefined relationships that exist between the tables.

Example: an e-commerce relational database housing customer, product, and order information

**A diagram of a company

AI-generated content may be incorrect.**

Relational databases are not able to process unstructured data (e.g., information that is inconsistent in format and isn't aligned to a preset data model) but are excellent at supporting transactional or financial information that includes structured data or semi-structured types of data (e.g., data that has a consistent format and aligns to a preset data model).

**Examples of SQL Databases**

* **Oracle**: A widely used enterprise-grade relational database developed by Oracle Corporation, known for scalability and robustness.
* **MySQL**: An open-source RDBMS popular for web applications. Owned by Oracle, it's known for ease of use and strong community support.
* **MariaDB**: A drop-in replacement for MySQL, created after Oracle’s acquisition of MySQL. It’s open-source and community-driven.
* **PostgreSQL**: An advanced open-source object-relational database known for its rich features and ACID compliance. Ideal for complex data workloads.
* **MSSQL (Microsoft SQL Server)**: A powerful RDBMS from Microsoft, commonly used in enterprise environments for transaction processing and analytics.
* **SQLite**: A lightweight, serverless database embedded within applications. Great for mobile and small-scale projects.

**What is NoSQL(Not only Structured Query Language )?**

NoSQL, which stands for Not only SQL, is a database management system approach used to ingest, store, and retrieve unstructured data and semi-structured data within a database. This means that data that cannot be analyzed or counted through traditional relational databases (e.g., SQL) can remain in its native format and be ingested into a NoSQL database. The reason it is called NoSQL is to emphasize that these databases can handle non-tabular, non-relational data models as well as support SQL-like query languages.

A computer screen with words and a person on it

AI-generated content may be incorrect.

Unlike SQL, NoSQL uses unstructured data which is the data that doesn't have a predefined data model or consistent organization. In addition, unstructured data, such as social media posts, can update and change rapidly while structured data, such as bank transactions, have a much lower rate of change.

Examples of unstructured data include pictures, audio files, videos, and maps.

Now what is a NoSQL Database then?

NoSQL databases are databases that utilize a flexible schema that accommodates unstructured data and semi-structured data while also utilizing a non-tabular data storage method.

The use of a flexible schema enables NoSQL databases to ingest unstructured data in its native format (e.g., .txt, .JPG, MP3), which is not possible with SQL databases due to the requirement that all data align to a predefined format. Further, when NoSQL databases store data, flexible data models are employed so that unstructured data files can have different data structures and still be stored within the same collection.

A diagram of a database

AI-generated content may be incorrect.

**Types of NoSQL Databases**

* **Document Databases**: Store data as documents (similar to JSON). Each document is a record with flexible fields. Great for hierarchical data.  
  **Example**: MongoDB
* **Key-Value Stores**: Store data as key-value pairs. Simple and fast for lookups using unique keys.  
  **Examples**: AWS DynamoDB, ScyllaDB
* **Column-Family Stores**: Organize data in columns instead of rows, ideal for large, sparse datasets and horizontal scaling.  
  **Examples**: Apache Cassandra, HBase
* **Graph Databases**: Use nodes and edges to represent entities and relationships. Ideal for social networks and recommendation engines.  
  **Examples**: Neo4j, AWS Neptune

A screenshot of a computer

AI-generated content may be incorrect.Comparison between **SQL** and **NoSQL** databases

| **Feature** | **SQL (Relational)** | **NoSQL (Non-relational)** |
| --- | --- | --- |
| **Data Model** | Structured tables with rows and columns | Flexible models: documents, key-value, graphs, etc. |
| **Schema** | Fixed schema; predefined structure | Dynamic schema; flexible structure |
| **Scalability** | Vertical scaling (more CPU/RAM) | Horizontal scaling (more servers) |
| **Query Language** | SQL (Structured Query Language) | Varies by type (e.g., MongoDB uses JSON-like queries) |
| **Transactions** | Strong ACID compliance | Often eventual consistency; some support ACID |
| **Best For** | Complex queries, structured data | Unstructured or semi-structured data, fast development |
| **Examples** | MySQL, PostgreSQL, Oracle, MSSQL | MongoDB, Cassandra, Redis, Neo4j |

**What are the scenarios where we need to choose one over another?**

We should choose SQL when:  
 **1.** **Our data is structured and consistent**• We have a fixed schema and well-defined relationships.  
• Example: Customers → Orders → Products, with foreign keys.  
  
**2. ACID compliance is critical**• We need strong guarantees on data accuracy and consistency — for example, in banking, financial transactions, or inventory management.  
  
**3. We require complex joins and queries**• If we’re running reports, aggregations, or need complex relationships across multiple tables, SQL excels.  
  
**4. We want maturity and standardization**• SQL databases like PostgreSQL, MySQL, Oracle, and SQL Server are mature, reliable, and widely supported with strong tooling.  
  
**5. We care deeply about data integrity**• SQL supports strong constraints like NOT NULL, UNIQUE, CHECK, and foreign keys, which help enforce business rules.  
  
⸻

We should choose NoSQL when:  
  
**1. Our data is unstructured or semi-structured**• Ideal when we’re dealing with JSON documents, key-value data, or dynamic fields that don’t fit neatly into rows and columns.  
  
**2. We need to scale horizontally**• If we expect high traffic, large datasets, or global reach, NoSQL databases are designed for horizontal scaling across distributed systems.  
  
**3. We want high performance for large-scale reads/writes**• Great for handling big data, real-time analytics, logs, and high-speed data ingestion.  
  
**4. Our schema evolves frequently**• NoSQL gives us schema flexibility — helpful in agile development or projects where requirements change rapidly.

**SQL Databases**

**Advantages:**

* Structured and organized data storage.
* Strong data integrity through ACID compliance.
* Standardized query language (SQL).
* Mature tools and widespread community support.

**Disadvantages:**

* Rigid schema; changes require migrations.
* Less suited for unstructured or rapidly evolving data.
* Vertical scaling can be costly and complex.
* Not ideal for high-speed, high-volume data ingestion.

**NoSQL Databases**

**Advantages:**

* Flexible schema; handles unstructured and semi-structured data.
* Easily scalable horizontally across multiple servers.
* High performance for large-scale and real-time applications.
* Supports various data models (document, key-value, graph, column).

**Disadvantages:**

* Lack of standard query language across systems.
* May sacrifice consistency for scalability and speed.
* Limited support for complex joins and relational logic.
* Varying levels of maturity and tooling depending on the database.