Understanding SQL vs NoSQL Databases

In 2024, big data market volume is expected to reach $84 billion for the first time ever. 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. Read on to learn more about SQL databases and NoSQL databases, how they work, common use cases, and the pros and cons of each type of database.

Table of contents

* [Structured Query Language (SQL)](https://www.mongodb.com/resources/basics/databases/nosql-explained/nosql-vs-sql#structured-query-language-sql)
* [Not only Structured Query Language (NoSQL)](https://www.mongodb.com/resources/basics/databases/nosql-explained/nosql-vs-sql#not-only-structured-query-language-nosql)
* [Key differences between SQL vs NoSQL databases](https://www.mongodb.com/resources/basics/databases/nosql-explained/nosql-vs-sql#key-differences-between-sql-vs-nosql-databases)
* [Use cases for SQL and NoSQL databases](https://www.mongodb.com/resources/basics/databases/nosql-explained/nosql-vs-sql#use-cases-of-sql-vs-nosql-database-technology)
* [FAQs](https://www.mongodb.com/resources/basics/databases/nosql-explained/nosql-vs-sql#faqs)

Structured Query Language (SQL)

What is SQL?

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.

What is structured data?

Structured data is 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).

What is a SQL database?

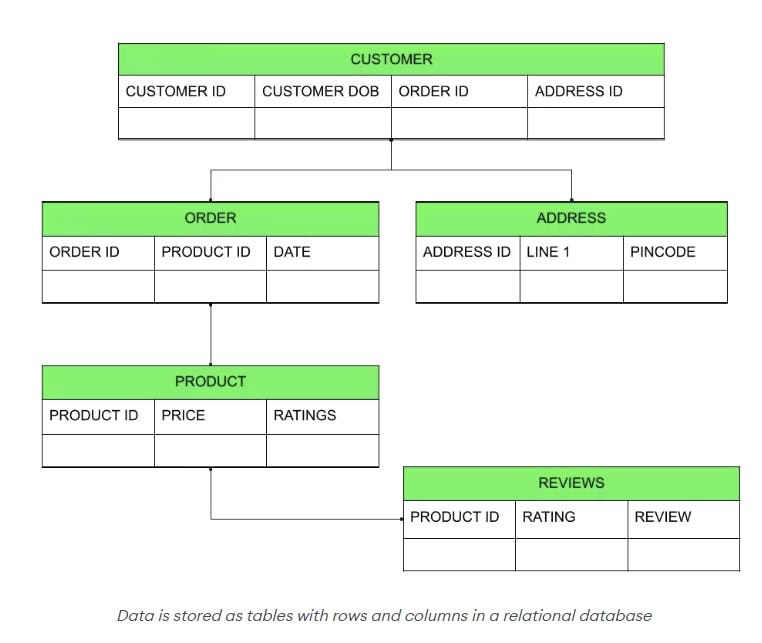
When the term "SQL database" is used, it refers to a type of database where SQL is the primary programming language used to create and manage that database. SQL application programming interfaces (APIs) contain groups of functions that enable developers to execute and manage database operations without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



It's important to note that relational databases are created and managed using a fixed schema. A fixed schema means that all data ingested into the database must be precisely aligned to predefined formatting standards which limits the types of data structures that relational databases can store. For example, 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

There are a variety of different SQL database examples, including:

* Oracle: Oracle Database is a relational database management system (RDBMS) developed and marketed by Oracle Corporation and is one of the most commonly used enterprise database systems in the world.
* MySQL: MySQL is a commonly used, open-source, relational database management system used for creating and administering databases. Developed and distributed by Oracle Corporation, MySQL is known for ease of use, extensive community support, and reliability.
  + Note: An open-source alternative to MySQL is called MariaDB which was designed as a replacement for MySQL after the acquisition of MySQL by Oracle Corporation.
* PostgreSQL: PostgreSQL is an open-source object-relational database management system known for its advanced, feature-rich capabilities which extend the abilities of SQL. Developed as part of the POSTGRES project at the University of California at Berkeley, PostgreSQL offers [ACID](https://www.mongodb.com/basics/acid-transactions)-compliant features that safely store and scale complicated data workloads (PostgreSQL.org 2024).
* MSSQL: MSSQL, which stands for Microsoft SQL Server, is a relational database management system developed by Microsoft. This database platform is commonly used in large enterprise environments to support high-volume transaction processing, business intelligence, and analytics applications.
* SQLite: Unlike other examples in this list, SQLite is actually a software library that provides an RDBMS. Unlike the other RDBMSs in this list, SQLite is serverless and self-contained with zero configuration. This is because it is embedded within the application using SQLite and, as a result, doesn't need a separate server.

It’s important to note that other types of databases can also establish relationships between pieces of data. In the case of normalized tabular databases (e.g., SQL or relational databases), these relationships are expressed using foreign keys or intersection tables. In the case of database management systems (DBMSs) such as MongoDB (e.g., a NoSQL database), these relationships are established by embedding or referencing data.

Not only Structured Query Language (NoSQL)

What is NoSQL?

[NoSQL](https://www.mongodb.com/nosql-explained), 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.

What is unstructured data?

Unstructured data is 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.

What is a NoSQL database?

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](https://www.mongodb.com/docs/manual/data-modeling/#document-structure).

*To learn more, follow our tutorial on*[*NoSQL databases*](https://www.mongodb.com/nosql-explained)*.*

Types of NoSQL databases

There are different types of NoSQL databases, including:

* Document databases: Document databases, sometimes referred to as object-oriented databases, store data in documents similar to JSON (JavaScript Object Notation) objects, although they're not JSON stores. They use the drivers returned from native objects to the programming language used by the developer without needing an object relational mapper (ORM). Each document itself is treated as a record and can contain values including numbers, arrays, objects, strings, or even Boolean characters. In addition, key-value pairs, nested documents, or other structured data can be included. A popular provider of these NoSQL databases is [MongoDB](https://www.mongodb.com/databases/what-is-an-object-oriented-database).
* Key-value databases: Key-value databases collect, retrieve, and store data as groupings of key-value pairs. This means that each data record is represented by a unique key and an associated value. The key is used to retrieve the corresponding value from the database. For example, in an interior design key-value database, a key might be "color" and the value might be "purple." Popular providers of these NoSQL database systems include AWS and ScyllaDB.
* Column-family stores: Column-family databases organize data into columns rather than rows, which is helpful when working with wide datasets that are sparse in depth. In fact, column-family stores are sometimes referred to as "wide-column stores." In column-family stores, each row has a different set of columns, with columns then gathered into "families." These data models are helpful when working with large-scale datasets that benefit from horizontal scaling to optimize performance. Popular providers of these NoSQL databases include Apache Cassandra and HBase.
* Graph databases: Graph databases store data in nodes and edges. Nodes typically store information about people, places, and things, while edges store information about the relationships between the nodes. Graph databases are excellent tools for querying graph structures (e.g., social networks, hierarchies). Popular providers of these NoSQL databases include Neo4j, AWS, and Kibana.

Key differences between SQL vs NoSQL databases

While both SQL and NoSQL databases offer valuable functionality, it's important to understand the key differences between them.

Database storage model

The difference between SQL and NoSQL database systems relating to data storage is a stark one. Specifically, SQL databases store data in tables containing rows and columns whereas NoSQL systems store data using various methods depending on the type of unstructured data being ingested (e.g., JSON documents, key-value pairing, family grouping, graph nodes/edges).

Data type

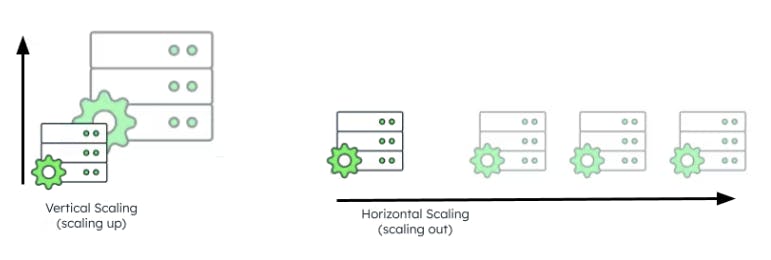
While NoSQL databases, sometimes referred to as non-relational databases, are able to ingest, store, and retrieve unstructured data, SQL databases (e.g., traditional relational databases) are not. SQL databases are only able to ingest, store, and retrieve structured data. This is due to the difference between SQL vs NoSQL schemas utilized.

Schemas

SQL databases rely on a strict, predefined data schema with which data to be ingested must align. However, NoSQL databases use flexible schemas which enable them to ingest data in its various native formats.

Scalability

It's important for database administrators to plan for the growth and expansion of their database systems — this is another clear point of differentiation between SQL vs NoSQL databases.



SQL databases

SQL databases are traditionally scaled vertically. This means that resources (e.g., CPUs, storage, or memory) are added to a single server. And, while this may cause limitations to the amount of growth possible as there is only one server with finite capacity being scaled, there are several reasons for this scaling choice:

* Atomicity, Consistency, Isolation, and Durability (ACID) compliance: ACID compliance refers to a set of properties that guarantee the reliability, consistency, and data integrity of database transactions. This is very important as many SQL databases contain banking and financial information which must be compliant with government and industry standards. However, it is more difficult to maintain ACID compliance across a distributed system (e.g., many computers linked by a network) where resources are increased via horizontal scaling vs one computer and one server that is vertically scaled.
  + Note: There are some distributed system NoSQL databases that can maintain ACID compliance, such as [MongoDB Atlas](https://www.mongodb.com/databases/acid-compliance).
* Transaction management: Transaction management mechanisms are employed by SQL databases to maintain data integrity and database consistency. Managing simultaneous transactions across multiple nodes in a distributed database environment would likely create additional complexity and resource usage which could impact both data integrity and possibly overall database performance. If horizontal scaling were to be utilized, these issues would be a possibility.
* Schema rigidity: SQL databases employ rigid, predefined schemas with which they ingest data. While this is simple to maintain in a one computer/server environment, complexity would be added if a distributed system with horizontal scaling were employed. Specifically, each node could potentially have a different schema version which would increase administration overhead and potentially cause additional data consistency issues.

NoSQL databases

NoSQL database systems are often configured in what's called a distributed system. This means a number of independent computers (e.g., nodes) are linked via a network and work together to yield common objectives. Being part of a distributed system also means that [horizontal scaling vs vertical scaling](https://www.mongodb.com/basics/horizontal-vs-vertical-scaling#:~:text=The%20vertical%20scaling%20approach&text=One%20of%20the%20advantages%20of,there%20is%20less%20complexity%20involved) can be utilized.

Horizontal scaling involves increasing the available resources and capacity of a distributed system by adding more nodes (e.g., computers, servers) to that system. In doing so, more nodes are available to support the system's workload. In addition, there is virtually no limit to how large the database can grow from a capacity perspective as additional nodes can continue to be added.

*Learn more about*[*database scaling*](https://www.mongodb.com/basics/scaling)*.*

Use cases of SQL vs NoSQL database technology

While SQL databases excel at managing structured, relational data while maintaining transactional integrity and executing complex queries, NoSQL databases are unsurpassed in querying and storing unstructured or semi-structured data while providing unlimited scalability and flexibility.

With that said, use cases are a driving force in the selection between SQL and NoSQL databases (e.g., relational vs non-relational databases). Here is a summary of the strengths and associated use cases of SQL databases and NoSQL databases.

SQL database use cases

The key characteristics of SQL databases lend themselves to the following use cases:

Regulatory compliance

Because SQL database structure lends itself to ACID compliance, they are often used for the purposes of storing data that must meet certain governmental or industry standards.

The use cases below all involve some level of ACID compliance.

Transactional databases

[Transactional databases](https://www.mongodb.com/databases/types/transactional-databases) store data that results from an interaction between two or more parties.

Examples:

* Retailer point-of-sale (PoS) databases
* Healthcare prescription and order databases
* Commercial banking
* Accounting and financial recordkeeping databases

Enterprise resource planning (ERP) systems

ERP systems are used to help businesses manage processes that are key to operations, employee management, production, and more.

Examples:

* Human resources databases
* Supply chain management systems
* Risk management systems

NoSQL database use cases

While it's a common notion that NoSQL (e.g., non-relational) databases aren't ACID-compliant, some actually are. MongoDB is a leading example of a NoSQL, [ACID-compliant database](https://www.mongodb.com/basics/acid-transactions).

Bearing this in mind, here are some examples of commonly found NoSQL database use cases:

Transactional databases

Transactional databases can also be supported by NoSQL databases in that they are used to store unstructured data that results from an interaction between two or more parties.

Examples:

* Healthcare patient files requiring non-relational database capabilities (e.g., patient records, x-ray/scan photos and videos)
* Insurance case files (e.g., auto accident photos, injury documentation)
* Legal document databases (e.g., depositions, pleadings, case files)

Document databases and digital asset management (DAM)

Document databases and digital asset management store and manage documents, images, multimedia content, videos, and more.

Examples:

* Online libraries (e.g., legal libraries, online Library of Congress)
* Digital publishing platforms such as Kindle or Nook
* Media streaming services such as Netflix and Hulu
* Online photo-sharing platforms such as Instagram or Meta

Graph and network analysis

Graph and network databases are excellent at managing such data structures as recommendation engines, social networks, and associated network analysis given their ability to identify and analyze nonintuitive relationships within interconnected data elements.

Examples:

* Social network analysis (e.g., post and user metrics)
* Fraud detection which isolates unusual transactions or other anomalies
* Knowledge graphs (e.g., Gartner product/service quadrants)

Internet of Things (IoT) platforms

IoT platforms are often employed to store and analyze sensor data and device metadata in real time.

Examples:

* Smart home systems (e.g., Google Nest, Amazon Alexa)
* Smart city systems (e.g., traffic light operation)
* Meteorological information gathering