**What is SQL?**

Structured Query Language (SQL) is a programming language that’s been around since the early 1970s. Back then, data storage was expensive, so a key focus of SQL was to cut down on the duplication of data.

SQL is widely used for data management in relational database management systems. The language is used to query relational databases, which are databases that recognize relationships between stored data items.

**Examples of SQL relational databases**

* MySQL
* Oracle
* Microsoft SQL Server
* PostgreSQL
* Db2
* YugabyteDB
* CockroachDB
* Azure SQL Database
* Microsoft Access
* Ingres
* Sybase
* SQLite

**What is NoSQL?**

NoSQL databases are non-relational (as opposed to SQL, which is relational). They first emerged in the late 2000s when developer productivity became more important than storage costs.

Crucially, NoSQL does not mean that these databases never use SQL. (There are NoSQL databases that can and do support SQL.) Instead, it’s best to think of NoSQL as “not only SQL.”

**Examples of NoSQL databases**

* Google Cloud BigTable
* Apache HBase
* Redis
* FaunaDB
* CouchDB
* MongoDB
* Cassandra
* Elasticsearch
* Neo4j
* HBase
* InfinityDB
* DynamoDB
* MariaDB
* Scylla
* ArangoDB
* InfiniteGraph

**What are the differences between SQL and NoSQL?**

Let’s examine the differences between SQL and NoSQL databases:

**What storage models do SQL and NoSQL use?**

SQL databases use tables with fixed columns (attributes) and rows (records). These databases follow specific rules regarding integrity and consistency.

NoSQL databases, however, don’t stick to this rigid format. Instead, they tend to belong to one of four categories:

* **Document stores:** store and encode data in documents in formats such as JSON, XML, YAML, and BSON
* **Graph databases:** structure data as nodes and relationships to show the connections between different data elements
* **Column-oriented:** store data in cells bundled in an unlimited number of columns rather than rows
* **Key-value stores:** store data as key-value pairs, where each key is a unique identifier that maps to an associated value

**What database schemas do SQL and NoSQL use?**

SQL requires a rigid, predefined, static, or fixed schema. It organizes data in a tabular, relational way. Therefore, you need to structure and organize data before building a SQL database.

NoSQL databases have flexible, dynamic schemas for data that is unstructured. Therefore, there isn’t much need to structure or organize data before placing it in a NoSQL database.

**How scalable are SQL and NoSQL databases?**

Both SQL and NoSQL are scalable, although the nature of their scalability is different.

You can scale SQL databases “vertically” if you exceed the current server capacity, meaning you can increase the current hardware’s processing power by migrating to a larger server. This involves adding RAM, SSD, or CPU capacity.

On the other hand, you can easily scale NoSQL databases “horizontally” by adding more servers to handle higher traffic as needed. Think of this as adding more room to a home by building an extension on the same floor.

In general, scaling horizontally gives you greater capacity and more power, which means that NoSQL databases are the best choice if you have large data sets that change regularly.

Although you can scale SQL databases horizontally, this isn’t well supported.

**Is SQL faster than NoSQL?**

Neither SQL nor NoSQL is faster than the other overall. Rather, their speed depends on the context in which they are being used.

SQL databases were designed back when data storage was expensive and data duplication had the potential to waste a lot of money. Therefore, SQL databases avoid duplication and data redundancy.

This means SQL databases are faster for queries, joins, updates, etc.

In contrast, NoSQL databases were designed for unstructured data. As explained previously, these can be column-oriented databases, graph databases, document stores, or key-value stores.

In NoSQL databases, data is stored together (not separately, as with SQL). This means that it’s faster to perform read or write operations on one data entity compared with SQL databases.

Essentially, SQL is great for protecting data validity, whereas NoSQL is ideal for when you need fast availability of big data.

**What are the pros and cons of NoSQL and SQL?**

Next, let’s look at the advantages and disadvantages of NoSQL and SQL databases.

**SQL pros**

**1. Large community of users**

The SQL language has been around since the 1970s. Despite being more than 50 years old, it’s still widely used today. Throughout the years, it’s attracted a large community of SQL experts who are enthusiastic about sharing their knowledge.

Developers can easily find answers to their questions and discover opportunities for collaboration. There are also numerous SQL vendors and consultants to call on should you need extra support.

**2. Rigid, standardized schema**

On the surface, this might seem like a con, but having an inflexible schema is helpful when your goals are maintaining security, the integrity of data, consistency, and compliance. This leads to…

**3. ACID database compliance**

ACID (Atomicity, Consistency, Isolation, and Durability) is a set of database properties that guarantee the validity of data, even when there are power failures, errors, or other mishaps.

Because of the rigid structure of SQL’s relational databases, their ACID compliance is high.

SQL is the best choice when running applications if data integrity is paramount and there is no room for error.

**4. Simple, keyword-based language**

Few programming languages are as user-friendly as SQL. No coding is required – you can query and manage databases with keywords. SQL is often one of the first languages back-end developers learn at college because of its simplicity.

**SQL cons**

**1. Ongoing expense**

As explained previously, you scale SQL databases “vertically” if you exceed the current server capacity. You can increase processing power by adding RAM, SSD, or CPU capacity.

However, this requires ongoing investment since you’ll need to spend more as your data grows.

**2. Slows down over time**

SQL’s advantage of removing data duplication and redundancy is also its disadvantage. As the database gets bigger, the joins between the tables can slow it down.

**3. Upfront planning**

SQL requires a rigid, predefined, static, or fixed schema. And once it is in place, it’s difficult to make changes.

This means you must allocate substantial time to planning before you can produce the database.

**NoSQL pros**

**1. Fast queries**

With NoSQL, it’s faster to run queries on large data sets. SQL can also be fast, but its speed becomes hampered as the database grows. This does not happen with NoSQL databases since they don’t require joins.

**2. Resilient and stable**

Because NoSQL data is spread over many regions and servers, there is no single point of failure. This means they provide zero downtime and continuous availability.

**3. Cheaper in comparison**

Since you can scale NoSQL databases horizontally by adding servers, you can expand them relatively cheaply. You should be able to find affordable options regardless of your organization’s needs.

**4. Agile**

Rows and columns don’t stifle NoSQL databases, so they can handle all kinds of data, such as unstructured, structured, and polymorphic data.

This flexibility enables you to add new functionality and features as needed, and you can make modifications easily.

**NoSQL cons**

**1. Learning curve**

There is no standard language for NoSQL queries. Whereas SQL has one easy-to-learn programming language, you can expect a steep learning curve for NoSQL.

For example, if a developer is experienced with wide-column databases, they may find it takes a lot of time to learn how to build graph databases.

**2. Smaller community of users**

NoSQL databases have been around for more than a decade, so NoSQL’s community is growing. However, it’s still dwarfed by the SQL community. This means developers may find it difficult to get some of their questions answered.

Additionally, there are far fewer experts and consultants to call on compared with SQL.

**3. Complex querying problems**

NoSQL’s flexibility is a double-edged sword. Querying with NoSQL is less efficient than SQL because of the variety of data structures.

It could also mean you’ll need to spend more money on technical staff to run queries.

**Does Google use SQL or NoSQL?**

Google is a great example of a company that understands its goals and can therefore make the best choice for its needs between a SQL and a NoSQL database.

Because it deals with massive data sets, it’s chosen to work with a NoSQL database. The company uses Bigtable, which is a NoSQL database built in-house.

Bigtable is scalable to huge sizes, making it ideal for projects with big data sets, like YouTube, Google Search, Google Analytics, Google Finance, and Google Earth.

**When to use SQL vs. NoSQL in your organization**

SQL databases are ideal when:

* You need a high level of data security and integrity
* You have highly structured data that doesn’t change regularly
* You need to perform ad hoc requests or other complex queries
* You don’t need to scale horizontally
* You support transactional systems, such as financial or accounting apps

It’s better to use NoSQL databases when:

* You don’t require a high level of data security and integrity
* You have a lot of unstructured or semi-structured data
* You have data that changes frequently and need the flexibility of a dynamic schema
* You’re looking to streamline development and save money by using a structured approach
* You need to scale horizontally