SQL vs. NoSQL

In the world of databases, there are two main types of solutions: **SQL** and **NoSQL** (or **relational** and **non-relational** databases).

Both of them differ in the way they were built, the kind of information that they store, and the storage method that they use.

Relational Databases are structured, and have predefined schemas like phone books that store phone numbers and addresses.

Non-relational Databases are unstructured, distributed, and have a dynamic schema, like file folders that hold everything from a person's address and phone number to their Facebook likes and shopping preferences.

SQL

Relational databases store data in rows and columns.

Each row contains all the information about one entity, and each column contains all the separate data points.

Some of the most popular relational databases are:

- MySQL
- Oracle
- MS SQL Server
- SQLite
- PostgreSQL
- MariaDB

NoSQL

The following are the most common types of NoSQL:

- **Key-Value Stores** — data is stored in an array of key-value pairs.

They key is an attribute name that is linked to a value.

Well-known key-value stores include Redis, Voldemort, and Dynamo.

- **Document Databases** — in these databases, data is stored in *documents* (as opposed to rows and columns in a table), and these documents are grouped together in collections.

  Each document can have an entirely different structure.

  Document databases include CouchDB and MongoDB.
- **Wide-Column Databases** — instead of “tables”, in columnar databases we have column families, which are containers for rows.

  Unlike relational databases, we don’t need to know all the columns up front, and each row does not have to have the same number of columns.

  Columnar databases are best suited for analysing large datasets — big names include Cassandra and HBase.
- **Graph Databases** — these databases are used to store data whose relations are best represented in a graph.

  Data is saved in graph structures with nodes (entities), properties (information about the entities), and lines (connections between the entities).

  Examples of graph databases include Neo4J and InfiniteGraph.

High-level Differences between SQL and NoSQL

Storage

SQL stores data in tables, where each row represents an entity and each column represents a data point about that entity.

For example, if we are storing a car entity in a table, different columns could be “Colour”, “Make”, “Model” etc.

NoSQL databases have different data storage models.

The main ones, as stated, are key-value, document, graph, and columnar.

Schema

In SQL, each record conforms to a fixed schema, meaning that the columns must be decided and chosen before data entry, and each row must have data for each column.

The schema can be altered later, but it involves modifying the whole database and going offline.

In NoSQL, schemas are dynamic.

Columns can be added on the fly, and each ‘row’ (or equivalent) doesn’t have to contain data for each ‘column’.

Querying

SQL databases use SQL (structured query language) for defining and manipulating the data, which is very powerful.

In a NoSQL database, queries are focused on a collection of documents.

Sometimes it is also called UnQL (unstructured query language).

Different databases have different syntax for using UnQL.

Scalability

In most common situations, SQL databases are vertically scalable, i.e. by increasing the horsepower (more memory, CPU power etc.) of the hardware, which can get very expensive.

It is possible to scale a relational database across multiple servers, but this is a very challenging and time-consuming process.

On the other hand, NoSQL databases are horizontally scalable, meaning that we can add more servers easily in our NoSQL database infrastructure to handle a lot of traffic.

Any cheap commodity hardware or cloud instances can host NoSQL databases, thus making it a lot more cost-effective than vertical scaling.

A lot of NoSQL technologies also distribute data across servers automatically.

Reliability or ACID Compliancy (Atomicity, Consistency, Isolation, Durability)

The vast majority of relational databases are ACID compliant.

So when it comes to data reliability and safe guarantee of performing transactions, SQL databases are the better bet.

Most of the NoSQL solutions sacrifice ACID compliance for performance and scalability.

SQL vs. NoSQL — Which one to use?

When it comes to database technology, there’s no one-size-fits-all solution.

That’s why many businesses rely on both relational and non-relational databases for different needs.

Even as NoSQL databases are gaining popularity for their speed and scalability, there are still situations where a highly structured SQL database may perform better.

Choosing the right technology hinges on the use case.

Reasons to use a SQL Database

Here are a few reasons to choose a SQL database:

1. <u>We need to ensure ACID compliance.</u>

   ACID compliance reduces anomalies and protects the integrity of your database by prescribing exactly how transactions interact with the database.

   Generally, NoSQL databases sacrifice ACID compliance for scalability and processing speed, but for many e-commerce and financial applications, an ACID compliant database remains the preferred option.

2. <u>Our data is structured and unchanging.</u>

   If our business is not experiencing massive growth that would require more servers, and if we’re only working with data that is consistent, then there may be no reason to use a system designed to support a variety of data types and high traffic volume.

Reasons to use a NoSQL Database

When all of the other components of our application are fast and seamless, NoSQL databases prevent data from being the bottleneck.

Big data is contributing to a large success for NoSQL databases, mainly because it handles data differently than the traditional relational databases.

A few popular examples of NoSQL databases are MongoDB, CouchDB, Cassandra, and HBase.

Here are a few reasons to choose a NoSQL database:

1. Storing large volumes of data that often have little to no structure.

   A NoSQL database sets no limits on the types of data that we can store together, and allows us to add new types as the need changes.

   With document-based databases, you can store data in one place without having to define what “types” of data those are in advance.
2. Making the most of cloud computing and storage.

   Cloud-based storage is an excellent cost-saving solution, but requires data to be easily spread across multiple servers to scale up.

   Using commodity (affordable, smaller) hardware onsite, or in the cloud, saves you the hassle of additional software, and NoSQL databases like Cassandra are designed to be scaled across multiple data centres out of the box, without a lot of headaches.
3. Rapid development.

   NoSQL is extremely useful for rapid development as it doesn’t need to be prepped ahead of time.

   If you’re working on quick iterations of your system that require making frequent updates to the data structure without a lot of downtime between versions, a relational database will slow you down.