SQL vs. NoSQL

In the world of databases, there are two main types of solutions: SQL and NoSQL - or rela databases and non-relational databases. Both of them differ in the way they were built, the information they store, and how they store it.

Relational databases are structured and have predefined schemas, like phone books that st numbers and addresses. Non-relational databases are unstructured, distributed and have a schema, like file folders that hold everything from a person's address and phone number t Facebook 'likes' and online shopping preferences.

SQL

Relational databases store data in rows and columns. Each row contains all the informatio

one entity, and columns are all the separate data points. Some of the most popular relation databases are MySQL, Oracle, MS SQL Server, SQLite, Postgres and MariaDB.

NoSQL

Following are most common types of NoSQL:

Key-Value Stores: Data is stored in an array of key-value pairs. The 'key' is an attribute which is linked to a 'value'. Well-known key value stores include Redis, Voldemort and D

Document Databases: In these databases data is stored in documents, instead of rows an in a table, and these documents are grouped together in collections. Each document can h entirely different structure. Document databases include the CouchDB and MongoDB.

Wide-Column Databases: Instead of 'tables,' in columnar databases we have column fa which are containers for rows. Unlike relational databases, we don't need to know all the up front, and each row doesn't have to have the same number of columns. Columnar data best suited for analyzing large datasets - big names include Cassandra and HBase.

Graph Databases: These databases are used to store data whose relations are best repres graph. Data is saved in graph structures with nodes (entities), properties (information abo entities) and lines (connections between the entities). Examples of graph database include 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 tab different columns could be 'Color', 'Make', 'Model', and so on.

NoSQL databases have different data storage models. The main ones are key-value, docu graph and columnar. We will discuss differences between these databases below.

Schema: In SQL, each record conforms to a fixed schema, meaning the columns must be and chosen before data entry and each row must have data for each column. The schema altered later, but it involves modifying the whole database and going offline.

Whereas in NoSQL, schemas are dynamic. Columns can be added on the fly, and each 'ro equivalent) doesn't have to contain data for each 'column.'

Querying: SQL databases uses SQL (structured query language) for defining and manipu data, which is very powerful. In NoSQL database, queries are focused on a collection of documents. Sometimes it is also called UnQL (Unstructured Query Language). Different have different syntax for using UnQL.

Scalability: In most common situations, SQL databases are vertically scalable, i.e., by inc the horsepower (higher Memory, CPU, etc.) of the hardware, which can get very expensiv possible to scale a relational database across multiple servers, but this is a challenging an consuming process.

On the other hand, NoSQL databases are horizontally scalable, meaning we can add more easily in our NoSQL database infrastructure to handle large traffic. Any cheap commodity hardware or cloud instances can host NoSQL databases, thus making it a lot more cost-ef than vertical scaling. A lot of NoSQL technologies also distribute data across servers auto

Reliability or ACID Compliancy (Atomicity, Consistency, Isolation, Durability): The majority of relational databases are ACID compliant. So, when it comes to data reliability guarantee of performing transactions, SQL databases are still 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 m businesses rely on both relational and non-relational databases for different needs. Even a databases are gaining popularity for their speed and scalability, there are still situations w highly structured SQL database may perform better; choosing the right technology hinges use case.

Reasons to use SQL database

Here are a few reasons to choose a SQL database:

1. We need to ensure ACID compliance. ACID compliance reduces anomalies and pr integrity of your database by prescribing exactly how transactions interact with the Generally, NoSQL databases sacrifice ACID compliance for scalability and process speed, but for many e-commerce and financial applications, an ACID-compliant da remains the preferred option.
2. Your data is structured and unchanging. If your business is not experiencing massiv that would require more servers and if you're only working with data that's consist there may be no reason to use a system designed to support a variety of data types a traffic volume.

Reasons to use NoSQL database

When all the other components of our application are fast and seamless, NoSQL database data from being the bottleneck. Big data is contributing to a large success for NoSQL data mainly because it handles data differently than the traditional relational databases. A few examples of NoSQL databases are MongoDB, CouchDB, Cassandra, and HBase.

1. Storing large volumes of data that often have little to no structure. A NoSQL databa limits on the types of data we can store together and allows us to add different new the need changes. With document-based databases, you can store data in one place 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 excell saving solution but requires data to be easily spread across multiple servers to scale Using commodity (affordable, smaller) hardware on-site or in the cloud saves you t of additional software, and NoSQL databases like Cassandra are designed to be sca multiple data centers out of the box without a lot of headaches.
3. Rapid development. NoSQL is extremely useful for rapid development as it doesn't be prepped ahead of time. If you're working on quick iterations of your system whi require making frequent updates to the data structure without a lot of downtime bet versions, a relational database will slow you down.

✔Great! You've completed this section.

Not Yet

← Previous
Redundancy and Replication

Next
CAP Theor

Send Feedback or Ask a Question

24 Reco