SQL and NoSQL comparison

Guo An Liew s3895776

RMIT University

19/10/2023

# Introduction

The choice of SQL or NoSQL is context dependent. Both designs provide advantages that the other cannot replicate. In summary, the advantage of NoSQL is apparent in its performance with smaller databases on non-relational data. On the other hand, SQL remains consistent for larger databases on relational data. We will discuss this topic with a comparison of key database features.

# Comparison

I will use the following metrics to compare SQL and NoSQL.

• Query performance, such as CRUD operations and joins.

• Resource requirements

• Security

• Concurrency and data integrity

• Scalability.

• Differences in data types, such as media and date storage.

## Query performance.

**Read and write.**

Research from the University of Rochester (R. Wang, Z. Yang, 2017) indicates NoSQL is significantly faster than SQL when performing reads or writes up to 1 million records. However, SQL has more consistent read and write speeds as records increases, while NoSQL speeds slows down as records grow.

**Joins**

SQL supports joining between several entities, by using foreign keys within each entity that refer to other entities. This enables queries requiring several entities to be generated quickly. NoSQL entities are isolated and independent, which do not support joins, unless a field implicitly refers to another entity. SQL is therefore more reliable than NoSQL when performing joins, and therefore more reliable with complex queries.

## Resource requirements

SQL is more demanding in resources due to its structure and consistency standards. NoSQL is built without a demanding structure, meaning it is more efficient to store in.

Consider that SQL demands normalisation in its database. Normalisation reduces data duplication, which reduces the size of the database. NoSQL can de-normalise its databases, meaning there can be the same data across two or more documents, or even within separate collections. This means that NoSQL would require more resources replicate the same data but is more efficient than SQL otherwise.

## Security

SQL is an older model and therefore its security services are developed. NoSQL is a newer model; therefore, security services are also new. Furthermore, consistency standards for SQL are higher than NoSQL, meaning SQL is built to secure data integrity to a certain extent in its databases.

Overall, there is an expectation for SQL to be more secure than NoSQL, but that does not mean NoSQL databases are insecure.

## Concurrency / data integrity

Sourced from IBM ([B. Anderson](https://www.ibm.com/blog/author/benjamin-anderson/), [B. Nicholson](https://www.ibm.com/blog/author/brad-nicholson/), June 12, 2022)

SQL supports ACID properties, meaning that the database is reliable when performing CRUD operations.

NoSQL follows the CAP theorem, and some NoSQL databases support ACID properties. The CAP theorem states that up to two of these concurrent properties can be met:

* Consistency: each request receives the most recent result or an error.
* Availability: each request returns a non-error result
* Partition tolerance: delays or losses between servers/nodes do not affect the entire operation.

For distributed systems such as NoSQL databases, the CAP theorem asserts that a metric of concurrency cannot be met at the same time as two other conditions. Therefore, NoSQL operations in NoSQL are less concurrent than SQL operations.

SQL is cited for its data validity, meaning the data will be reliable in many situations where concurrent operations occur. NoSQL is favoured for its performance in handling large amounts of data fast, while sacrificing validation to an extent. Therefore, SQL will have greater integrity of its data.

## Scalability

Sourced from IBM ([B. Anderson](https://www.ibm.com/blog/author/benjamin-anderson/), [B. Nicholson](https://www.ibm.com/blog/author/brad-nicholson/), June 12, 2022).

SQL supports vertical scaling, which means increasing the processing power, RAM, and other storage types, of one CPU.

NoSQL scales horizontally, allowing additional CPUs or servers to increase the load of the database.

## Data types

NoSQL is efficient with “large volumes of rapidly changing unstructured data” (NoSQL Database - What is NoSQL? Microsoft). Unstructured data is varying and omnipresent in the world. The following lists a few examples of unstructured data:

* Documents
  + Emails
  + JSON
  + BSON
  + Receipts
  + Cheques
* Media
  + MP3, MP4
  + JPEG, PNG

These types of media may be frequently edited for various purposes. For example, several people may want to change their profile picture on Instagram. NoSQL would provide these changes smoothly.

While NoSQL handles unstructured data better, SQL provides efficiency with structured, otherwise known as relational data. Relational data indicates that data from various sources are linked to each other typically by a real-world representation. For example, transactional data refers to a set of data representing a financial transaction, involving different entities such as a store and a buyer. This information is stored in a relational database, meaning these entities together represent a financial transaction. Financial transactions are often repeated, such as a repeat customer buying from the same store. For each transaction that occurs the same data is stored; hence it is structured.

# Recommendation

The following table from Microsoft summarizes the use case scenarios of SQL and NoSQL (No SQL Database – What is NoSQL? Microsoft).

|  |  |  |
| --- | --- | --- |
|  | NoSQL | SQL |
| Best for: | When data is non-relational, large, and change often.  Applications with constant availability across several locations.  Performance over consistency. | When data is rigid and structured.  Consistent schema of entities and relationships.  Legacy systems built for relational structure.  Complex queries |

If security is a concern, it is still possible to seek out secure database engines in NoSQL. For resource requirements, while it is likely that NoSQL will burden the CPU less, if data duplication is a big concern for the database size, consider SQL. For concurrency or data integrity, other factors allow consistent and smooth operation from the application level or processing level. When database consistency is not always necessary, hence either NoSQL or SQL will work when concurrent operations occur.

We have mainly discussed query complexity, performance, and data types as deciding factors for using either NoSQL or SQL. However, some of these concerns may not concern stakeholders who prioritize certain functionality in their service. It is important to communicate potential problems with stakeholders to discuss priorities of certain qualities, which will inform a better decision for the database engine.

# References

R. Wang, Z. Yang, 2017. *SQL vs NoSQL: A Performance Comparison.* University of Rochester. Accessed 19/10/2023. <https://www.cs.rochester.edu/courses/261/fall2017/termpaper/submissions/06/Paper.pdf>

*NoSQL Database - What is NoSQL?* Microsoft. Accessed 20/10/2023.

<https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-nosql-database>

[B. Anderson](https://www.ibm.com/blog/author/benjamin-anderson/), [B. Nicholson](https://www.ibm.com/blog/author/brad-nicholson/), June 12, 2022. *SQL vs. NoSQL Databases: What’s the Difference?* IBM. Accessed 20/10/2023. <https://www.ibm.com/blog/sql-vs-nosql/>