Comparison between SQL and NoSQL Databases

Posted on December 22, 2024

Introduction

Developers, at some point in their coding journey, need to run into using a database because they have to persist data on any kind of application that they build. For that, they have to choose the right database technology for their needs and budget. From engineers and IT decision-makers to analysts, everyone needs to generate interest in the depth and sheer variety of database management systems to manage the data on the applications. To do so, they interact with Structured Query Language (SQL) and Relational Database Management Systems (RDBMS), which are decades-old paradigms. However, today, rising volumes of all kinds of data, availability of storage, evolving requirements of analytics, and availability of processing power require different kinds of technologies to work with data. Thus, in today's development World, there are two main categories of databases in use that are commonly referred to as SQL and NoSQL and according to the situations each is best suited for in order to make informed decisions about which query system should be used. This paper compares and contrasts SQL and NoSQL database types based on their similarities and differences, as well as the advantages and drawbacks both database types offer to the development world.

SQL vs NoSQL

Structured Query Language (SQL) accessed through relational databases was developed in the 1970s with the focus on manipulating data in predefined categories that would allow for reducing data duplication. On the other hand, Not-only SQL (NoSQL) databases were developed with a great focus on fast queries, scaling, and frequent application changes that made programming simpler and easier for developers in the late 2000s. SQL programming databases are an established programming language that tends to have tabular schemas, vertical scaling with a large server, ACID compliant, rigid structure, have complex data to object mapping, and are difficult to modify, but the benefit is that, data will be clean and consistent. Although SQL databases are quite restrictive for some developers due to the standardized and rigid schema that every object added must conform to the recognized schema of the rows and columns of the linked tables in the relational database, it is essential for the integrity, security, compliance, and consistency of the data (Khan et al., 2023).

Moreover, SQL databases are ACID compliant, that help supports the high-level integrity of the data by ensuring durability, atomicity, isolation and consistency of the data and all transactional changes. ACID property of atomicity executes all data and transactional changes in a single operation that

keeps the data valid and consistent at the beginning and completion of each transaction. Furthermore, every transaction runs synchronously and acts in isolation without any competition, as “I” in ACID signifies “Isolation”. Once a transaction is complete, the connected data in the transaction cannot be altered, which makes the transaction durable and permanent once it is completed (Grolinger et al., 2013). For instance, in an inventory management system, it is essential to remove items as soon as they are purchased to avoid understocking or overstocking problems in the inventory. Once an order is placed, the related tables of the inventory, customer information, payment information, and a new shipment data object should be created and updated to complete the transaction.

On the other hand, NoSQL databases are commonly referred to as non-relational databases that can be set up very quickly and with very minimal planning without any structured tabular schema. NoSQL databases persist large amounts of data with simple lookups and predictable query patterns along with flexible schemas to analyze and traverse relationships between connected data driven by DevOps and agile practices that allow for rapid application change. Most of the NoSQL databases do not support multi-record ACID transactions and also do not require Object Relational Mapping (ORM) but some popular programming databases such as MongoDB support ACID transactions as well as map documents and data directly to data structures in popular programming languages. Examples of NoSQL databases include FlockDB, CouchBase, SimpleDB, Neo4j, DynamoDB, HBase, and MongoDB, whereas examples of SQL databases include Microsoft SQL Server, Oracle Database, MYSQL, IBM DB2, Microsoft Access, and SQLite.

Main Differences between SQL and NoSQL

In comparison to SQL databases, NoSQL databases are easy for developers to work with because of their flexible data models, horizontal scale, and incredibly fast query structure. The main differences between SQL databases that use a programming language with predefined categories and NoSQL databases that use a programming language which does not store and query data in tables or other means than SQL are curated as follows:

- ## Flexibility

In NoSQL databases, data does not need a predefined schema and therefore tends to be more flexible than in SQL databases. NoSQL databases easily adapt and evolve with the agile changes and iterative updates as applications in NoSQL databases do not need to spend more time defining the relationships of objects and the structure of data; rather, applications can be built and scaled even if the data is partially structured. SQL databases, on the other hand, have a well-established presence as their structure is rigid, where a developer needs to update all the respective fields in the tables to ensure thorough documentation. With SQL databases, reformatting or changing the data can be

tedious because it requires a back-endless schema to reformat data to ensure atomicity, consistency, isolation, and durability.

• Scalability

Both databases, SQL and NoSQL, can handle lots of data based on the situation, but they have their own different ways of scalability. NoSQL databases are scaled horizontally, whereas SQL databases are designed to be scaled vertically. Therefore, in a NoSQL database, a developer can add cheaper commodity servers, whereas SQL databases require expensive processing units and hardware to handle the increasing load of the data while building an application. For this reason, big and popular companies opt for NoSQL databases to scale the data because of their extensive scalability, as handling the myriad of tables in an SQL environment can get unwieldy for big companies that deal with huge amounts of data.

• Organization

Both databases, SQL and NoSQL, distinctively organize and structure data in different ways. A NoSQL database is more flexible than a relational SQL database as the data in a NoSQL environment is not beholden to relationships. A developer can still build relationships irrespective of tables and fields as the structure of a NoSQL database can take a couple of different forms, such as a document-based database in which it is easier to adjust data and relationships as the use-case of iterative changes in the non-relational database (Haseeb & Pattun, 2017). Contrary to NoSQL database organization, SQL database enables the developers to store, organize, and access data in a myriad of fields in distinctive tables with clearly defined relations and schema. In a relational SQL database, each item is connected and directly associated with the other item in a table in some capacity.

- Data Normalization and Retrieval

In the NoSQL database, data is quickly available depending on the queried server, but the only drawback is that it is difficult to ensure that the data is always consistent because queries might not return updated information. However, distributed nature of the NoSQL database type makes it possible for that database to return distinct values consecutively. This is the reason why the NoSQL database is not ACID compliant but BASE compliant, which means that this database type does not ensure that data is always consistent at the beginning and completion of the transaction but ensures that data has an “E” that signifies eventual consistency of the data. Thus, inconsistency is one of the major drawbacks of NoSQL, which makes consistency in data retrieval impossible but places importance on the speed and availability of the data (Chandra, 2015). On the other hand, SQL

negates data duplication while information can be queried, connected, and stored in different tables, which is the main goal behind the development of relational SQL databases. To put it more simply, SQL allows a developer to store different information in different tables using common values. When SQL databases become large, more processing power, lookups, and joins are required to complete the query between several tables without the database system slowing down significantly.

- Language

In NoSQL queries, there is no fixed language, unlike SQL, where there is plain English language in use. Different NoSQL database types have different variations in the syntax that is used for conducting queries. As SQL requires only one language to learn, NoSQL has a higher learning curve because of the distinctive languages used in every type of NoSQL database. SQL, on the other hand, has a large user community because of its well-established structure and only one language in use that allows developers numerous opportunities to build applications, consult, conduct, collaborate, and sharpen their working knowledge and skills. However, in the NoSQL databases' case, it is a very crucial and difficult task to find experienced developers who are also well-versed in different NoSQL languages so that they can implement their knowledge of the NoSQL system in an application which is time as well as cost-consuming at the same time.

- ACID Compliance

SQL databases, as compared to NoSQL databases, are more ACID compliant as their tables are in-sync and precisely structured, which leaves no room for error, ensuring complete and valid transactions. It is a developer-friendly language but not more than languages used in NoSQL databases. SQL language uses only simple keywords in plain English without any coding, making it easy to craft, access, and modify precise data. On the other hand, NoSQL databases do not offer ACID compliance except the popular MongoDB database, as NoSQL databases offer eventual consistency to data that guarantee database transactions are processed reliably. Therefore, in SQL databases, data is reliable, complete, consistent, and accurate due to ACID compliance which stands for "Atomicity, Consistency, Isolation, and Durability" of the data, whereas in NoSQL databases, ACID compliance is compromised in favor of performance and scalability as changes to data in NoSQL may take time to propagate through the system (Al Fatah & Chowdhury, 2016).

Conclusion

SQL and NoSQL databases have their own time, situation, needs, and places, and none is inherently better than another database type. SQL database has its own pros and cons. Similarly, the NoSQL database has its own advantages and drawbacks depending upon the choice of developer that what database type he selects that can achieve all of his use cases in a single database. For instance, graph databases (SQL database type) can analyze relationships in the data excellently but may not provide what a developer needs for range queries or for everyday retrieval of the data to be

reformatted. Both are used in meeting the specific needs of the developers for building certain applications using different programming languages depending on the data environment as well as the goals of an organization. In a nutshell, both database types, SQL and NoSQL, play to their own strengths and can be used with the preferred choice of the organization that best suits its needs and goals. So, organizations can opt for one specific database type or can use both technologies together within their cloud architecture to have a predefined schema in SQL as well as the flexibility to reformat data in NoSQL.

References

Al Fatah, J., & Chowdhury, L. (2016). Consistency Issues on NoSQL Databases: Problems with Current Approaches and Possible Solution (s). *Update*, *4*(4), 3.

Chandra, D. G. (2015). BASE analysis of NoSQL database. *Future Generation Computer Systems*, *52*, 13–21.

Grolinger, K., Higashino, W. A., Tiwari, A., & Capretz, M. A. (2013). Data management in cloud environments: NoSQL and NewSQL data stores. *Journal of Cloud Computing: Advances, Systems and Applications*, *2*, 1–24.

Haseeb, A., & Pattun, G. (2017). A review on NoSQL: Applications and challenges. *International Journal of Advanced Research in Computer Science*, *8*(1).

Khan, W., Kumar, T., Zhang, C., Raj, K., Roy, A. M., & Luo, B. (2023). SQL and NoSQL Database Software Architecture Performance Analysis and Assessments—A Systematic Literature Review. *Big Data and Cognitive Computing*, *7*(2), 97.