THE DIFFERENCE BETWEEN RELATIONAL AND NON-RELATIONAL DATABASES IN PROGRAMMING

Crinu DIMITRIU

Department of Computer Science and Systems Engineering, group MI-221, Faculty of Computers, Informatics and Microelectronics, Technical University of Moldova, Chișinău, Republic of Moldova

Corresponding author: Crinu Dimitriu, email: crinu.dimitriu@iis.utm.md

Coordinator: Ala PUȘCAȘU, university assistant, Department of Foreign Languages, UTM

Summary. *Databases are integral for programming, particularly in the development of web applications. SQL (relational) and NoSQL (nonrelational) databases are two popular and powerful types of databases used for data storage. However, people often do not understand the difference between SQL and NoSQL types. This work will focus on analyzing and comparing the benefits and downsides of each of them, in order to clarify what exact type fits perfectly specific scenarios of work.*

Keywords*: database, data, storage, DBMS, SQL, MongoDB*

Introduction

In recent times, because of the increasing amount of data that is being transferred throughout the Internet, it has become essential to develop technologies that allow developers to store and interact with large quantities of data. Either it is a social media app like Facebook or a complex structured AI that processes enormous volumes of information, they both use a database as the only system that collects and stores data: Oracle, MySQL, PostgreSQL, MongoDB or others. Nowadays, two main types of data repositories are being used: relational and non-relational, usually called by developers SQL and NoSQL types.

Overall, both relational and non-relational databases are huge constructions that keep records and allow various data manipulation operations. SQL and NoSQL databases are essential components of programming, and each offers its own advantages and disadvantages depending on the project. However, it is fundamental to understand the difference between these two types in order for the developers to choose the most suitable type of database for their projects.

The relational database model

A relational database is a type of collection of data [1] that works on the concept of storing and organizing information in tables that have rows and columns. Basically, there are multiple tables in a database, each being connected to the other via keys. Keys, or identifiers, define the relations between the tables inside the repository. SQL (Structured Query Language) databases are the more traditional and widely used form of a database. It uses a structured query language to store and retrieve data. SQL databases are great for storing structured data and offer powerful search capabilities and are often used in business intelligence applications. By having a relatively simple structure portrayed by tables, this type of database quickly became the most used in projects.

The advantages and disadvantages of using SQL databases

As it was previously mentioned, the most popular type of relational database is the Structured Query Language (SQL) [2] database, which is the most widely used language for managing relational databases. It provides the developers with the necessary tools to access, analyze and modify data. SQL is highly reliable, robust, and easy to learn, and it is used in pretty much any Relational Database Management System (RDBMS): Oracle, PostgreSQL, MySQL and Microsoft SQL Server [1].

The benefits of using a SQL database are:

1) Ease of use and advanced querying capabilities: SQL is very easy to understand and use. Queries written in this language are like a sentence in English, in which the user "asks" the database to interact with the data in a certain wanted way, allowing users to perform CRUD operations (create, read, update, delete) and analyze data efficiently. Moreover, users can "select" data directly to access it like it is in Image 1:

```
SELECT model, speed, hd FROM PC WHERE price < 500

The result of your query

model   speed   hd
1232    500     10.0
1232    450     8.0
1232    450     10.0
1260    500     10.0
```

Image 1. Example of a query in SQL

2) Structure and consistency: SQL databases maintain strict data consistency. This means that data is organized into tables and rows, and all data is related to one another. It ensures that data is accurate and reliable, making SQL databases an ideal choice for mission-critical applications. Image 2 represents a table of superheroes in SQL [3]:

id	name	align	eye	hair	gender	appearances	year	universe
1	Spider-Man (Peter Parker)	Good Characters	Hazel Eyes	Brown Hair	Male Characters	4043	1962	marvel
2	Captain America (Steven Rogers)	Good Characters	Blue Eyes	White Hair	Male Characters	3360	1941	marvel
4854	Batman (Bruce Wayne)	Good Characters	Blue Eyes	Black Hair	Male Characters	3093	1939	dc
3	Wolverine (James \"Logan\" Howlett)	Neutral Characters	Blue Eyes	Black Hair	Male Characters	3061	1974	marvel
4	Iron Man (Anthony \"Tony\" Stark)	Good Characters	Blue Eyes	Black Hair	Male Characters	2961	1963	marvel
4855	Superman (Clark Kent)	Good Characters	Blue Eyes	Black Hair	Male Characters	2496	1986	dc
5	Thor (Thor Odinson)	Good Characters	Blue Eyes	Blond Hair	Male Characters	2258	1950	marvel
6	Benjamin Grimm (Earth-616)	Good Characters	Blue Eyes	No Hair	Male Characters	2255	1961	marvel

Image 2. Example of a table in SQL

3) ACID compliance [2]: SQL databases are ACID (Atomicity, Consistency, Isolation, Durability) compliant, ensuring that all transactions are completed successfully, and data remains consistent. This is crucial for applications that require high levels of reliability, such as financial systems. The definition of ACID is shown in Image 3:

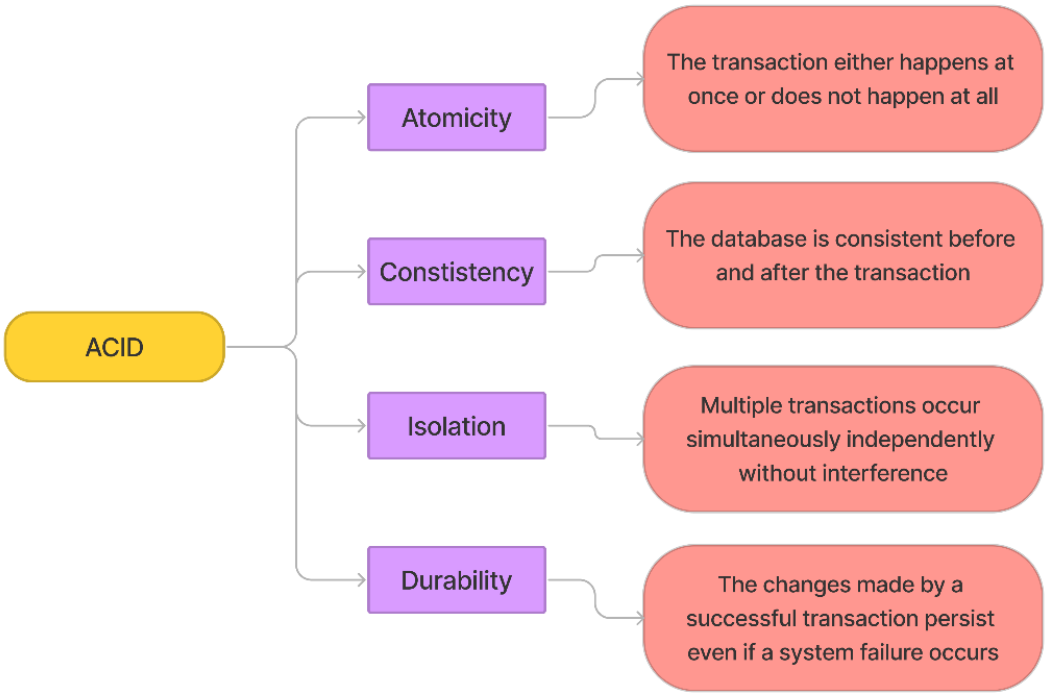

Image 3. ACID model explained

4) Security: SQL databases offer robust security features that ensure data is protected against unauthorized access. SQL databases support role-based access control, allowing users to grant or restrict access to different levels of data.

Still, relational databases are not perfect, they have a significant drawback when it comes to the way data is held within the repository: it is concentrated in machines (servers, computers) that have limited scaling possibilities. Basically, it is impossible to infinitely enlarge the storage capacity in a relational database, as the data is not partitioned or compressed in any way. It is still possible to add more and more machines according to the amount of data, but, even so, there is a physical limit, making SQL databases less suitable for large-scale data processing requirements. SQL databases have a rigid structure and are less flexible than other types of databases. Adding or modifying data structures can be time-consuming and complex, requiring significant effort to maintain data integrity.

Additionally, SQL databases can be expensive to set up and maintain. Licensing fees, hardware requirements, and maintenance costs can add up quickly, making SQL databases less attractive to businesses looking to save money.

The non-relational database model

A non-relational database, also known as NoSQL [4], is more of a modern type of a database that is not built on the traditional model used in relational databases like MySQL and PostgreSQL, because, unlike them, NoSQL databases are not bound to strict schemas. NoSQL databases instead are characterized by their non-tabular, distributed data stores, and their ability to handle large volumes of structured, semi-structured, and unstructured data. They are ideal for applications that require real-time data analysis and can scale relatively easily. There are multiple database management systems that run the non-relational type, the more popular being: MongoDB, InfluxDB, Firebase and Cassandra.

The key differences over the traditional databases

NoSQL databases offer a different approach to data storage, with several advantages and disadvantages. These key features are what made this new approach of data storing an alternative to relational schemas, that some companies may consider less viable.

Advantages of NoSQL databases:

1) Scalability: NoSQL databases are designed to handle large amounts of unstructured that can be contained in a dynamic way or semi-structured data that cannot be easily organized into tables or rows like in a relational database. This makes them ideal for handling large-scale data processing requirements, such as those needed in big data or IoT applications.
2) Flexibility: NoSQL databases offer more flexibility than relational ones. They can easily accommodate changes to data structures and schema without requiring the migration of existing data. This makes them more adaptable to changing business requirements.
3) Performance: NoSQL databases are known for their high-performance capabilities. They can handle large amounts of data at high speeds, making them ideal for real-time applications that require low latency and high throughput.
4) Cost: NoSQL databases are typically less expensive than traditional databases. They require less hardware, and licensing fees are often lower, making them an attractive option for businesses looking to save money on their data storage.

Although this type of data storing seems extremely attractive, especially compared to the other type, it is still not perfect, as it also has drawbacks:

1) Limited functionality: NoSQL databases lack the advanced querying and reporting capabilities of traditional relational databases. They may also lack support for ACID (Atomicity, Consistency, Isolation, Durability) transactions, which can make them less suitable for mission-critical applications.
2) Data consistency: NoSQL databases often sacrifice consistency for scalability and flexibility. This can lead to data inconsistencies, which can be difficult to detect and correct.
3) Difficult to start working with: NoSQL databases require a different set of skills than traditional relational databases. Developers and database administrators need to learn new data modelling techniques and query languages, which can be time-consuming and challenging. Furthermore, different database management systems (MongoDB, Cassandra, InfluxDB) may have different approaches on how to access and manipulate data. For example, InfluxDB uses InfluxQL – a very similar to SQL language. On the other hand, to use MongoDB, it is necessary to learn a completely different to SQL language, as shown in Image 4 below, where the user extracts from the database the books that contain "MongoDB" in their titles:

```
5   db.books.find({ title: /MongoDB/ })
6
7   result:
8
9   [
10    {
11      _id: ObjectId("617fd2d2f44c134e94f15c0d"),
12      title: "MongoDB in Action",
13      author: "Kyle Banker",
14      publication_date: ISODate("2011-12-16T00:00:00Z"),
15      pages: 450,
16      publisher: "Manning Publications"
17    },
18    {
19      _id: ObjectId("617fd2d2f44c134e94f15c0e"),
20      title: "MongoDB Applied Design Patterns",
21      author: "Rick Copeland",
22      publication_date: ISODate("2013-12-16T00:00:00Z"),
23      pages: 318,
24      publisher: "O'Reilly Media"
25    }
26  ]
```

Image 4. Example of a query in MongoDB

4) Integration: NoSQL databases may not integrate well with existing systems and applications. This can make it difficult to migrate existing data from a relational database to a NoSQL database or to use data from a non-relational database in other systems.

Conclusion

It is essential to evaluate the specific needs of the business and choose the right type of database accordingly. NoSQL databases are great for applications where there are enormous amounts of unstructured data that need to be processed quickly as they offer high performance. Other benefic aspects of NoSQL are the scalability and the low cost of maintaining the repositories. Nevertheless, non-relational databases are not ideal: they have limited functionality compared to SQL-based systems, they are harder to use and difficult to integrate into projects.

Overall, in the majority of cases, the best choice is to stick with the traditional SQL relational-based data repositories, even though they are less scalable and may underperform compared to non-relational ones, because they offer a more structured and safer environment to work with data. SQL is being used everywhere: from small projects to big tech companies such as Google, Facebook, Uber, Amazon, Netflix and others [5].

References

1. What is a relational database? [online]. [accessed 26.02.2023]. Available: https://cloud.google.com/learn/what-is-a-relational-database
2. What is a relational database? [online]. [accessed 26.02.2023]. Available: https://www.ibm.com/topics/relational-databases
3. Супергеройское введение в SQL [online]. [accessed 26.02.2023]. Available: https://www.cat-in-web.ru/superheroes-sql/
4. What is NoSQL? [online]. [accessed 26.02.2023]. Available: https://www.mongodb.com/nosql-explained
5. SQL vs. NoSQL Databases: What's the Difference? [online]. [accessed 26.02.2023]. Available: https://www.ibm.com/cloud/blog/sql-vs-nosql