Rajat Kumar Pradhan
19111045
6th semester, Biomed

NoSQL databases for big data

January 26, 2022

1 Abstract

Many challenges encountered while dealing with particular specific applications, such as the storing of very large datasets, have led to the development of NoSQL solutions. Traditional RDMS, on the other hand, ensure data integrity and transaction consistency. However, this comes at the expense of a tight storage architecture and difficult management. Data consistency and integrity are necessary in many circumstances, such as financial applications, however they are not always required. The purpose of this paper is to provide a detailed picture of NoSQL's evolution and mechanics, as well as the benefits and drawbacks of the most popular NoSQL data models and frameworks. To that end, a detailed comparison of SQL and NoSQL databases is offered first. Scalability, performance, consistency, security, analytical capabilities, and fault-tolerance techniques are all explored. Second, the four major types of NoSQL databases: key-value stores, document databases, column-oriented databases, and graph databases are defined and compared. Third, we compare the key technical solutions for each NoSQL data model.

2 Introduction

There are a number of phenomena that have aided the proliferation of NoSQL databases and bolstered their appeal. For example, the advent of new web technologies and low-cost storage, as well as tendencies toward Web 2.0, Web 3.0, and big data, have all contributed to the internet's rapid growth. Data storage, processing, analysis, and display face increased demands and obstacles as a result of these trends.

Indeed, massive data sets contain a variety of organized, semi-structured, and unstructured data that relational database management systems RDBMS and SQL struggle to handle. In fact, standard data storage and querying languages necessitate data organization in a specific and specified way, which is insufficient in the context of big data (where rapid and huge volumes of data are generated in heterogeneous formats).

To meet this problem, NoSQL databases (a phrase that encompasses all non-relational databases) were created to address big data challenges and issues such as large-scale storage of a variety of data types, the requirement for flexible schemas, and the need for scalable, quick, and distributed databases.

In fact, they offer a variety of categories to meet the needs of various use cases, including key-value pairs, documents, graphs, columnar databases, and geospatial databases. As a result, NoSQL is a database management system that is well-suited to distributed systems and non-relational data storage systems such as HDFS. It's important to note that NoSQL systems supplement rather than replace RDBMSs with their common query language SQL.

It's also worth noting that NoSQL solutions were not designed with the same goals in mind as SQL-based solutions. While relational databases are designed to store structured data and perform transactions, NoSQL databases were built to address the storage issues associated with large unstructured datasets. Both systems, in reality, have advantages and downsides.

NoSQL was created a long time ago before it was widely adopted. According to some sources, the word NoSQL was initially used in 1998 for relational databases that did not use SQL, and then again in 2009 for a non-relational database conference in San Francisco. Indeed, the adoption of NoSQL databases was aided by the proliferation of clouds, particularly Databases as a Service, and the pressing need for fast, scalable, and less expensive databases to handle massive data.

There were other factors that aided the spread of NoSQL. That is, to support object-oriented principles while avoiding costly object-relational mapping, data must be stored in a simpler structure. As a result, NoSQL was developed to meet the needs of simple, non-complex applications. Another trend is the rise of online technologies and cloud computing, both of which require low maintenance and scalability. There was also a trend in programming languages and frameworks to mask the complexities of SQL and relational databases in order to provide more flexible and convenient solutions (e.g., Ruby, Java Persistent API, ADO.net, etc.).

The purpose of this paper is to provide a clear picture of NoSQL's evolution and mechanics, as well as the benefits and drawbacks of the most popular NoSQL data models and frameworks. To that end, a detailed comparison of SQL and NoSQL databases is offered first. The four major types of NoSQL databases are defined and compared in the second section. Finally, we present and compare the various technical solutions for each NoSQL data model.

3 Comparing SQL and NoSQL databases features

This section gives a quick overview of the most essential characteristics of several NoSQL systems, allowing you to compare their performance, benefits, and drawbacks to traditional relational database systems.

For many years, RDMS and SQL databases were the standard. However, they can only support structured datasets and can only manage one type of preconfigured schema. They ensure atomicity, consistency, isolation, and durability (ACID) qualities, which are critical in a variety of applications. In fact, ACID properties are one of the most important features for ensuring the security of online transactions.

NoSQL, on the other hand, is better suited to non-relational, unstructured datasets, such as big data. They support a wide range of data schemas. They are, however, slow to handle massive queries, whereas SQL databases are faster and better suited to complicated and intense queries. Using commodity servers and clusters, NoSQL provides a less expensive solution to manage huge data.

When dealing with huge data and some web applications, RDMS runs into three major challenges. This includes the following:

1. Scale out data
2. Performance of single servers
3. Rigid schema design

Scalability, performance, flexibility, cloud usage, data model, queries and analytics, security, data replication, standardization, and maturity are all factors to consider when comparing relational vs NoSQL databases.

3.1 Scalability

In relational databases, big data storage, retrieval, and processing are complicated concerns. SQL databases are, in fact, vertically scalable. Users must enhance the capacity and performance of a single server to manage increased load. This is accomplished by boosting the dedicated database server's CPU, RAM, or SSD capacity, for example. Sharding numerous tables across huge clusters or grids, on the other hand, is expensive and difficult in such an architecture.

NoSQL databases, on the other hand, are horizontally scalable. Users only need to add servers to the NoSQL database infrastructure to accommodate high data volumes. As a result, the workload is divided among a large number of servers. As a result, achieving system scalability using NoSQL is easier and less expensive.

3.2 Performance

Relational databases, on the other hand, necessitate a specified data model and organized data. They provide advanced SQL functionality for managing, updating, and querying data. They have a number of advantages, including maintaining the integrity, consistency, and reliability of data and transactions. In comparison to NoSQL databases, relational databases are unquestionably more reliable. By adhering to ACID principles, SQL databases ensure the consistency and integrity of data and transactions.

In the event of large, expanding datasets, however, ensuring ACID characteristics is difficult. As a result, NoSQL databases rely on BASE concepts (essentially available, softstate, eventually consistent). As a result, they provide a versatile architecture that can handle organized, unstructured, and semi-structured data. It is simple for users to do frequent code pushes and iterations.