International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

The RDBMS vs. NoSQL- A Comparative Study

Arghyarupa Patra

Under the supervision of Miss. Akanksha Kulkarni] Ajeenkya D Y Patil University, Pune, India

Abstract-

Increasing reliance on social media applications, sensors for data collection, and smartphone usage has resulted in a rapid increase in the amount of data transmitted over networks. Typically, such data is unstructured and comes from a variety of sources in various formats. As a result, data abstraction for rendering is difficult, necessitating the creation of a computing system capable of storing data in structured manner and supporting distributed parallel computing. There are approaches to dealing with big data using RDBMS and NoSQL. This paper compares RDBMS, and NoSQL approaches. The mechanics of these approaches can be clearly distinguished from each other by reviewing each approach. Essentially, such systems rely on a number of factors, including the query language. This paper examines each of these database systems and attempts to identify the best solution for Big Data needs.

Keywords- Database, Software program, SQL, NoSQL, RDBMS, Big Data, Data Management

I. INTRODUCTION

A database is a group of data that can be used independently or in conjunction with other data. A database management system is software that enables a computer to perform database functions such as data storage, retrieval, addition, deletion, and modification. The RDBMS is still in use today to support the management of all data types, regardless of format. With the new Big Data concept, it is deemed necessary to develop a new data management method to support applications such as real-time log file analysis, e- commerce transactions, and data uploaded to social media. Despite the effective data retrieval demonstrated by RDBMS, such relational method is ineffective in dealing with large scale and diverse data configurations. As a result, proper data management has been provided to support Big Data management by using the term NoSQL [1], [2], [3]. This system demonstrates the ability to handle large-scale data insertion and retrieval operations.

In general, NoSQL stands for Not Only SQL (NoSQL). It is known as NoSQL because the system lacks the traditional characteristics of RDBMS and SQL (Structured Query Language). The basic features of NoSQL tools are data division and duplication. Because the processes are typically distributed across multiple computers, the load on each machine can be significantly reduced. As a result, applications designed for RDBMS may be incompatible with the system requirements of NoSQL [4]. RDBMS systems typically begin by building conceptual and rational data models, whereas NoSQL databases begin by categorizing the application queries and configuration that the data model can effectively support. Furthermore, changes to the queries in the data model are required to ensure rapid data retrieval. To that end, the NoSQL method can encourage data storage towers, each of which guarantees scalability for various applications [5].

A different retrieval mechanism is used for NoSQL than for RDBMS, as was already mentioned. To support the complexity of a large uncorrelated database, such an approach necessitates a large amount of processing power. The architecture, data model, query language, consumer API, ease of use, and scalability of NoSQL can all be used to quantify its performance. The following section outlines the comparative study between RDBMS and NoSQL.

II. RDBMS vs. NoSQL

A. Features

A comparison between RDBMS and No SQL properties are presented in the Table 1. The Data Manipulation Language (DML) operations in RDBMS provide the required ease with data input, storage, access, process, and analysis, among other things. There are encryption and tokenization solutions available to ensure the security of information throughout its life cycle. An index is a data structure that speeds up data retrieval operations (most notably the SELECT DML Statement) at the expense of more writes and storage space.

Traditional RDBMS storage and processing capabilities can be easily scaled up by increasing the database server's horsepower. RDBMS supports transactional properties such as Atomicity, Consistency, Isolation, and Durability (ACID). SQL is not relational. Data bases with a dynamic schema that avoids joins and is easy to scale. No SQL database is used for distributed data stores with massive data storage requirements. Big data and real-time web apps do not use SQL. Companies like Twitter, Facebook, and Google, for example, collect terabytes of user data every day. No SQL database stands for "Not Only SQL" or "Not SQL." Carl Strozz pioneered the No SQL concept in 1998. Google, Facebook, and Amazon, among others, have made no SQL databases popular. When using RDBMS for large amounts of data, the system response time becomes slow.

The various data model capabilities of No SQL databases make them extremely flexible when it comes to handling data. Developers and architects choose a No SQL database to more easily handle different application development requirements. No SQL data models include graph, document, wide-column, and key value.

Table 1: A Comparison between RDBMS and No SQL

Criteria	**RDBMS**	**No SQL**
Multi- model	It requires data to be put into tables	Data models include graph, document, wide-column, key value.
Scalable	It can't scale, because they are built with master-slave architecture.	No SQL database with master less, peer-to-peer allows easy scaling.
Flexible	Less flexible	Extremely flexible
Distributed	It uses a centralized application that is location dependent	It is designed to distribute data at global scale
Schema	In relational database schema is required.	In No SQL database a schema isn't required.
Volume of Data	Store limited data.	Store large volumes of data
Auto- Sharding	NO	YES
Transactions properties	ACID	CAP
Examples	Oracle, My SQL, DB2, etc.	MongoDB, Cassandra, HBase, etc.

No SQL database with a master less architecture with all nodes being the same. This allows easy scaling to adapt to the data volume and complexity of cloud applications. This scalability also improves performance, allowing for continuous availability and very high read/write speeds.

The multi-model capabilities of No SQL databases make them extremely flexible when it comes to handling data. They can easily process structured, semi-structured, and unstructured data, while relational databases, are designed to handle primarily structured data.

A No SQL database that is designed to distribute data at global scale, meaning it can use many locations involving multiple data centers and/or cloud regions for write and read operations. The advantage of using a distributed database with a master less architecture is that you can maintain continuous availability because data is distributed with multiple copies where it needs to be. The master less architecture, which allows for multiple copies of data to be maintained across different nodes. If a node goes down, no problem: another node has a copy of the data for easy, fast access. As previously stated, in the NoSQL architecture there are four types of models used for data management. It includes document based databases, key value based database, column based databases, and graph based databases.

ACID properties are not supported by No SQL. On the contrary, they follow Brewer's CAP theorem. The CAP theorem, which was introduced by Eric Brewer in 2000, states that there are three fundamental requirements for designing distributed architecture (CAP Theorem). Data consistency: After operations, data continues to be consistent. There is never any downtime. System is constantly on. Adaptability of network Partition: Despite unreliable communication between the servers, the system keeps running.

The CAP theorem's [6] fundamental tenet is that a distributed system can only satisfy two needs at once, not all of them. It can, for instance, at one point satisfy CA, CP, or AP. Since CA stands for Consistent and Available at a Time, all nodes must be connected at all times. The system locks up in the event of partition. CP: Although some data may not be available, the ones that are reliable and consistent. The system is still accessible in the event of partitioning, but some data that is returned might not be accurate.

A reliable performance in an RDBMS makes sure that all transactions are properly changed in the database. For instance, other database consumers can see the updated value when a transaction modifies a value in the database. The ACID properties are also used by the RDBMS when processing data. The operations in NoSQL, however, use the Basic, Availability, Soft State, and Eventual consistency (BASE) properties. The term "basic availability" refers to the data's apparent availability. The data is partially unavailable when a single node fails, but other data layers continue to function. The Soft state event suggests that data can be altered over time even in the absence of input because such a capability might guarantee consistency.

For instance, each client associated with a particular database system will be given a value when there hasn't been a change in the system for a while. It is not possible to conduct all business operations using only SQL or NoSQL. Depending on what a business expects from its database, each business has a unique need for how they intend to use a database. Each business must weigh the advantages and disadvantages of the SQL and NoSQL models to determine which is best for their organization or whether to use a hybrid approach that combines both [7]. When contrasting SQL and NoSQL, businesses should focus on two key factors: price and performance. The lack of SQL performance for many of today's high performance workloads can cause a

company to suffer a catastrophic revenue loss. It is simple to understand why some businesses might not want to install new systems in the current challenging economic climate and why emerging businesses might choose new NoSQL setups from an economical standpoint. Companies can use both SQL and NoSQL, ensuring that their atomicity, consistency, isolation, and durability are maintained while performing complex queries or engaging in multi-table transactions. This eliminates the need for businesses to choose between SQL and NoSQL interfaces without having to worry about many of the security concerns associated with using only NoSQL. This leads to evolvement of new technology called NewSQL that allows for horizontal scaling while preserving ACID properties. This not only enables working with big data by enabling concurrent operations, but it also upholds ACID properties. The ideal balance of consistency, scalability, speed, and availability has been achieved by NewSQL. NewSQL checks all the right boxes to qualify as the ideal database for Big Data OLTP applications, despite still being in its infancy.

III. CONCLUSION

In this paper, we compared RDBMS and no SQL databases. The research demonstrates the advantages of both relational databases and No SQL databases. Both databases have benefits and drawbacks. Most organizations continue to use relational databases. After the evolution of social media, relational databases are no longer sufficient to accommodate large amounts of data. As a result, most organizations employ both relational and No SQL databases. However, there are a few features of traditional RDBMS that are severely lacking. Although there is no standard SQL interface for No SQL, No SQL databases such as MongoDB and Cassandra have their own rich query language. The NewSQL is enhancement of SQL providing horizontal scaling which is ideal technology for Big Data applications, but still in initial stage of development.

REFERENCES

[1]. R. Cattell, "Scalable sql and nosql data stores," vol. 39, no. 4, pp. 12-27, May 2011.

[2]. Benzaken, G. Castagna, K. Nguyen, and J. Simeon, "Static and dynamic semantics of NoSQL languages," SIGPLAN Not. vol. 48, no. I, pp. 101-114, Jan. 2013.

[3]. Cruz, F., Maia, F., Matos, M., Oliveira, R., Paulo, J., Pereira, J., & Vilaça, R. (2013, April). Met: workload aware elasticity for nosql. In Proceedings of the 8th ACM European Conference on Computer Systems (pp. 183-196).

[4]. D. McCreary and A. Kelly, Making Sense of NoSQL: A guide for managers and the rest of us. Manning Publications, 2013.

[5]. C. Mohan, "History repeats itself: Sensible and NonsenSQL aspects of the NoSQL hoopla," in Proceedings of the 16th International Conference on Extending Database Technology, ser. EDBT '13. New York, NY, USA: ACM, 2013, pp.II-16.

[6]. *http://www.w3resource.com/mongodb/nosql.php* CAP Theorem

[7]. Bartholomew, D. (2010) SQL vs. NoSQL, Linux Journal, 195, 54-59. [