DOI https://doi.org/10.30525/978-9934-26-597-6-21

# EXPLORING LATEST NOSQL, NEWSQL AND SQL DATABASE TECHNOLOGIES

**Joseph Rufus Parthiban maria Dasan, Amit Joshi**
*ISMA University, Valērijas Seiles iela 1-korpuss 6, Rīga, LV-1019*
**Corresponding author's e-mail: mdrufus02@gmail.com*
*www.isma.lv*

**Abstract**

This article is about the research of the evolving landscape of technologies, which are the latest NoSQL and NewSQL databases. This article explores and looks into the inherent limitations of the relational database especially their struggle with scalability, flexibility and handling enormous amounts of data or unstructured data. The need for alternative databases that can handle the increasing amount and complexity of modern data is emphasized in this article by conducting a comprehensive literature review and performance test between NoSQL databases like Redis and MongoDB against relational databases like MySQL. This article also provides insights into the NoSQL and NewSQL database's, capabilities and suitability.

*Keywords:* NoSQL database, NewSQL database, SQL, Data modelling, data performance

## 1 Introduction

Databases in the current modern world, across industries, the rapid expansion of digital information has brought a fundamental necessity of how an organization manages, derives and accesses the vast amount of data. Traditional relational database struggles to keep up with scalability, flexibility and diverse data types which are common in this current interconnected modern world, that's why we need to explore new database technologies like NoSQL and NewSQL which handle the vast amount of data and diverse types of data. The topicality of researching NoSQL and NewSQL database technology is important because the traditional SQL database has some problems that are harder to fix. The problems include organizing data in an overly rigorous manner, struggling to handle disorganized or chaotic data and finding it difficult to expand when there is a large amount of data. Therefore, we require new database types that can handle these kinds of problems. Writing an article on this NoSQL database provides me a opportunity to learn

about how data is managed in new ways. NoSQL varies from other databases as they don't have strict rules in organizing data and also can expand easily. Learning about NoSQL databases helps me out with practical problems in the real world. NewSQL databases combine the best parts of NoSQL and Traditional databases, which makes it really exciting to learn and write about them. To sum up, by selecting this topic I will learn to understand how these new databases can solve modern data problems and also how it's been used in the real world practically**.**

**2 Comparing NewSQL with NoSQL and SQL**

NoSQL databases don't have a fixed schema for storing data which makes it flexible in data structures and types. Whilst, NewSQL databases can be both schema-fixed or schema-free which is why it is a mixture of traditional databases and NoSQL databases. Both NoSQL and NewSQL databases are horizontally scalable which means they can manage increasing loads by adding more servers to the system. NoSQL databases don't follow ACID principles instead they prioritise CAP theorem whereas, NewSQL databases follow ACID rules in order to ensure that the data is reliable and accurate. NoSQL databases don't support complex online transactional processing scenarios, while NewSQL completely supports online transactional processing. Both NoSQL and NewSQL databases are distributed databases which improves their performance and reliability. NoSQL databases have fewer security concerns as they are built in a simpler model. In contrast, NewSQL databases have moderate security concerns due to their mixture of the complexity of SQL systems and the scalability of NoSQL. NoSQL databases are suitable for applications where data schema can change and evolve such as social networks, IoT and big data analytics. Use cases like e-commerce, telecom, and gaming where high performance and transactional integrity are essential are better suited for NewSQL.

When NewSQL databases are compared with SQL databases, the main difference to be addressed first is that SQL databases scale vertically whilst maintaining the ACID properties and NewSQL databases scale horizontally whilst maintaining the ACID properties from SQL. NewSQL databases are distributed databases but SQL databases are not distributed databases as they work in a single database which may affect their performance and reliability. SQL databases can handle simple queries but it would struggle with complex queries. NewSQL databases are designed to be highly efficient with simple and complex queries. To summarize the differences addressed, SQL databases store data in an organized way and follow strict rules with ACID principles, whereas NoSQL databases are flexible in storing data and don't follow strict rules. NewSQL databases try to take the best parts from both NoSQL and

NewSQL, which is to store data in an organized manner and follow strict rules while being flexible and able to handle lots of work efficiently.

**3 Data modelling in NewSQL, NoSQL and SQL**

Schema-less approach – A schema-less approach is often used in NoSQL databases such as MongoDB etc. The schema-less approach will rule out the need for the database to store data according to a predefined or determined template. Due to this flexibility, any document or record can have a distinct collection of fields. The schema-less approach will be perfect for managing a wide range of data types and structures, especially because it would be helpful for applications that would evolve without updating the database schema frequently. For the schema-less approach, the use cases are content management systems, real-time approaches and applications where the data would change over time.

Schema-on-read approach – Rather than applying a schema at the moment of data storage, the schema-on-read approach applies the schema dynamically at the time of data querying. The schema-on-read approaches are often implied by several NoSQL systems, especially in big data and analytics contexts such as Hadoop's HBase. This means these systems will allow the data to be stored without a strict schema and give scalability. However, the schema will be applied or inferred during the data reading or querying process. These approaches would work well in data lakes and analytical applications where the data has various structures and also originates from several resources.

Schema-on-write approach – Traditional relational databases and few NewSQL databases use this schema-on-write approach. In this approach, the data schema must be established before the data insertion into the respective database. The schema-on-write approach enforces datatypes, relationships and constraints upfront, which ensures data consistency and integrity. This approach is suitable for applications where data accuracy and transaction integrity are crucial. For this approach, The use cases are applications that need high data integrity and complicated transactions, such as financial systems and inventory management.

The schema-less and schema-on-read approaches, which offer flexibility and scalability are particularly well-suited for applications handling massive volumes of diverse data and are synonymous with NoSQL database structure. The schema-on-write approach is particularly well-suited for NewSQL databases that seek to combine the scalability of NoSQL systems with the ACID properties and schema requirements of traditional SQL databases. So, from the above analysis, we can conclude that The application's particular

requirements, including the need for performance, scalability, flexibility, and data integrity, will determine which of these approaches is the best.

**4 CAP theorem in NoSQL**

**Having** (Baramand kumar, 2020) When creating systems that share data over a network, system designers are made aware of the use of the CAP theorem. Numerous distributed data system designs have been affected by the cap theorem which is also known as Brewer's theorem. The cap stands for consistency, availability and partition tolerance. With the CAP theorem, at most, we can only have two out of three guarantees for a database in a distributed system (like many NoSQL databases), which is a network that stores data on multiple machines (nodes) at the same time. Consistency means when clients connect to a distributed system they must see the same data, no matter which node they enter into. To ultimately achieve this, whenever data is written into a node it must immediately sent into or replicated into the rest of the nodes before the write is considered successful. Availability means each and every working node should respond to requests like read and write even if some of the nodes are down. Partition tolerance means the system should continue working even if some of the computers are unreachable or if there is a network problem. Once a partition heals, distributed systems that ensure partition tolerance can smoothly recover from partitions. According to the CAP theorem, in a distributed system like many of the NoSQL databases, it is not possible to attain all three aspects mentioned above at their maximum level simultaneously. So we have to trade off based on our prior system and they have been categorized into three types by the CAP theorem.

Consistent and partition tolerant database – A CP (Consistent and partition tolerant) database ensures consistency and partition tolerance at the cost of the availability. When there is a communication break between nodes, the non-consistent node becomes unavailable until the partition is fixed.

Available and partition tolerant database – An AP (Available and partition tolerant) database ensures the availability and the partition tolerance at the cost of consistency. Even during a partition, all nodes will remain available but it might cause some inconsistencies in the data. The AP database will later fix all the inconsistency by resyncing all the nodes, once the partition is resolved.

Consistent and available database – A CA (Consistent and available) database ensures availability and consistency at the cost of partition tolerance. A single-node database often falls into this CA database category because they don't deal with partition tolerance. Partition tolerance is a must when it comes to network-shared data or distributed systems like most of the NoSQL databases. Based on the system's requirement we have to choose between consistency and availability which applies to any trade-off willing to make

**5 Figures**

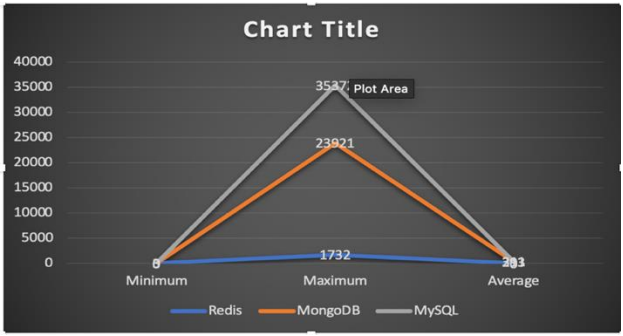


**Figure 1. The graph plotted for average, minimum and maximum response time of three databases.(using JMeter tool)**

**6 Acid properties in DBMS**

A series of database operations like adding, updating, changing and deleting data which when grouped is called as a transaction. Certain properties are followed both before and after the transaction in order to preserve consistency in the database, Which are called as ACID characteristics. The acid properties contain a set of rules that make sure a database works correctly and reliably. These ACID properties will ensure that the database will work properly in a valid state even in the case of unforeseen errors when we perform a transaction. So to conclude, ACID properties are the important rules that help databases to run properly and maintain order. The acronym ACID stands for atomicity, consistency, isolation and durability. A short explanation of each property is as follows.

Atomicity – The atomicity is what guarantees, that a transaction is handled as a solo and unbreakable unit of work. Either every transaction operation is successfully completed or none of them are completed. Data consistency and integrity are guaranteed as the entire transaction will go back to its initial state if one among them fails.

Consistency – Consistency is what ensures that the database is in a consistent and organized state. When a transaction is carried out, the database should be consistent before and after the transaction. To guarantee the data consistency constraints like foreign keys and unique keys should be

maintained. for example, in the context of a unique key, we shouldn't repeat the same value when it comes to unique key columns.

Isolation – Isolation guarantees that when multiple transactions are carried out simultaneously, they will not correlate or interfere with each other. Each transaction is isolated from one another until it is completed. Phantom reads unclean reads and non-repeatable reads are all avoided by this isolation.

Durability – Durability guarantees that when a transaction is executed and when the changes are made in the database it will be permanent. It will also withstand any further system failures. The modifications made during the transaction are permanently stored in the database. In the event of a system crash, the changes are still there and recoverable.

In conclusion, using ACID properties will slow down a system because it needs extra work to make sure that the data is always correct and consistent. ACID properties will become a problem in big distributed systems where lots of transactions take place in parallel. Implementing ACID properties will make a system more complicated and will require lots of knowledge and resources. When there is huge data to manage it is hard to follow the ACID rules, it is like trying to manage a huge event with strict rules. Even though ACID properties follow strict rules, they make sure that data is more reliable and accurate which is more crucial in many situations. Whilst, CAP theorem in NOSQL databases can make trade-offs based on their system's requirement as we have seen earlier. However, we cannot use the full potential of the CAP theorem, at most we can only have two out of three guarantees from the CAP theorem for a database in a distributed system like NoSQL databases.

**7 Conclusions**

NoSQL databases don't have a fixed schema for storing data which makes it flexible in data structures and types. Whilst, NewSQL databases can be both schema-fixed or schema-free which is why it is a mixture of traditional databases and NoSQL databases. Both NoSQL and NewSQL databases are horizontally scalable which means they can manage increasing loads by adding more servers to the system. NoSQL databases don't follow ACID principles instead they prioritise CAP theorem whereas, NewSQL databases follow ACID rules in order to ensure that the data is reliable and accurate. NoSQL databases don't support complex online transactional processing scenarios, while NewSQL completely supports online transactional processing. Both NoSQL and NewSQL databases are distributed databases which improves their performance and reliability. NoSQL databases have fewer security concerns as they are built in a simpler model. In contrast, NewSQL databases have moderate security concerns due to their mixture of the complexity of SQL systems and the scalability of NoSQL. NoSQL

databases are suitable for applications where data schema can change and evolve such as social networks, IoT and big data analytics. Use cases like e-commerce, telecom, and gaming where high performance and transactional integrity are essential are better suited for NewSQL.

When NewSQL databases are compared with SQL databases, the main difference to be addressed first is that SQL databases scale vertically whilst maintaining the ACID properties and NewSQL databases scale horizontally whilst maintaining the ACID properties from SQL. NewSQL databases are distributed databases but SQL databases are not distributed databases as they work in a single database which may affect their performance and reliability. SQL databases can handle simple queries but it would struggle with complex queries. NewSQL databases are designed to be highly efficient with simple and complex queries. To summarize the differences addressed, SQL databases store data in an organized way and follow strict rules with ACID principles, whereas NoSQL databases are flexible in storing data and don't follow strict rules. NewSQL databases try to take the best parts from both NoSQL and NewSQL, which is to store data in an organized manner and follow strict rules while being flexible and able to handle lots of work efficiently.

**Acknowledgments**

I would like to express my heartfelt thanks to the faculty of the IT Department for their constant support and guidance throughout this article. I'm especially grateful to my supervisor, Amit Joshi, for their valuable advice and encouragement. I also appreciate the efforts of the supervisor who provided assistance whenever needed. I 'd like to thank ISMA University for the opportunity.

## References

[1] Kumar, Barmanand. "CAP Theorem and NoSQL Databases." *Medium*, 20 Sept. 2020,https://medium.com/@kumar.barmanand/cap-theorem-and-nosql-databases-589e26e15905

[2] I Nevliudov, S Novoselov, O Sychova, V Gopejenko, N Kosenko Decentralized information systems in intelligent manufacturing management tasks. Advanced Information Systems 8 (3), 100-110 https://doi.org/10.20998/2522-9052.2024.3.12

[3] Keita, Zoumana 2022. "NoSQL Databases – Types of NoSQL Databases and How to use them." *Www.datacamp.com*, June 2022, www.datacamp.com/blog/nosql-databases-what-every-data-scientist-needs-to-know.

[4] Bezkorovainyi, V., Kolesnyk, L., Gopejenko, V., Kosenko V (2024) The Method of Ranking Effective Project Solutions in Conditions of Incomplete Certainty. Advanced Information Systems, 2024, 8(2), pp. 27–38. DOI: 10.20998/2522-9052.2024.2.04

[5] R Mukhamediev, Y Kuchin, N Yunicheva, Z Kalpeyeva, E Muhamedijeva, Viktors Gopejenko (2024) Classification of Logging Data Using Machine Learning Algorithms, Applied Sciences 14 (17), 7779 https://doi.org/10.3390/app14177779

**Authors**

**Joseph Rufus Parthiban Maria Dasan, 25th July 2002, INDIA**
**Current position, grades:** Student
**University studies:** ISMA University
**Scientific interest:** Computer Systems
**Publications (number or main):** main
**Experience:** 4 years

**Amit Joshi, 18th July 1987, INDIA**
**Current position:** Lecturer at ISMA University
**University studies:** BA School of business and Finance
**Scientific interest:** Artificial intelligence and machine learning, iOT
**Publications (number or main):** 6th
**Experience:** 14 + years