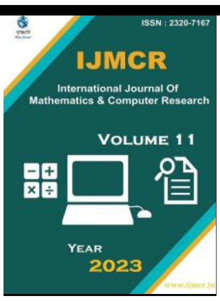
International Journal of Mathematics and Computer Research

ISSN: 2320-7167

Volume 11 Issue 04 April 2023, Page no. – 3364-3370

Index Copernicus ICV: 57.55, Impact Factor: 7.362

DOI: 10.47191/ijmcr/v11i4.09



Evolution of Storage Systems: Strategies of Choice and Experimental Evaluation of the Performance of SQL, NOSQL and NEWSQL Solutions

KABAMBA LUBANGI Nico[1], NYEMBO MPAMPI Augustin[2], NGOYI ILUNGA Elisée[3], KIBUNDULU DJOKO Bienvenue[4]

[1, 2, 3, 4]Faculty of Computer Science / Our Lady of Lomami University

ARTICLE INFO

Published Online: 27 April 2023

Corresponding Author: **KABAMBA LUBANGI Nico**

ABSTRACT

Databases in general and the relational model in particular have existed for several decades. This very powerful model represented the perfect solution for the various actors in the field of data management. Nevertheless, these architectures have reached their limits for certain big data services, such as Google, Yahoo, Facebook. To meet these new needs, several reflections relating to the design of new architectures and new solutions for data management have been proposed, mainly NoSQL and NewSQL.

Currently, many users of relational database management systems want to switch to these new solutions to anticipate the explosion of their data in the future and the support of data unstructured.

To justify and motivate such a changeover, we are developing a comparative study on the performance of SQL, NoSQL and Newsql solutions.

The purpose is to provide a set of criteria and indicators to interested actors, for possible decision-making on the appropriate solutions for their companies, by developing an experimental analysis on these solutions. The criteria for adopting each technology are also presented in order to guide the strategy for choosing one or more of them.

KEYWORDS: NoSQL, NewSQL, SQL, Strategies of choice.

1. INTRODUCTION

Database technology has known four successive eras: the navigational era, the relational era, the multidimensional era and the non-relational era.

In the navigational era (1965-1970), we saw the birth of two data models: the hierarchical model and the network model. The hierarchical model involves storing data in records with a set of fields each having a data type. The records are organized in the form of a tree such that each node (record) has only one parent. Each tree has a root and branches that allow access to different levels of data.

This model suffered from major limitations[1]: the redundancy of information that could lead to inconsistencies between the data and the existence of the data of a record depends on that of its parent record.

The network model stores data in a collection of records organized, not in a tree, but in a network (or graph) whose nodes are the records. This model allowed the elimination of data redundancies and the creation of multiple access paths to the same data. However, this model does not allow the real consideration of more complex phenomena and does not solve all the problems of independence of data structures and processing.

To overcome these limitations, in the early 1970, researcher Edgar Frank "Ted" Codd invented the relational model which revolutionized the academic and industrial world. This model

[1] Piekos, J., *SQL vs NoSQL vs NewSQL : Trouver la bonne solution*. http://dataconomy.com/2015/08/sql-vs-nosql-vs-newsql-finding-the-right-solution/ (consulté en janvier 2018).

is based on set theory and offers independence of the logical and physical levels.

Admittedly, other paradigms have emerged such as object-oriented database systems, but they have not been able to dethrone the relational model, given their complexity and their processing performance.

Nevertheless, the success of the relational model does not escape the rule of the decline of civilizations, according to which civilizations would follow a life cycle: gestation, birth, growth, apogee, and decline. This decline was observed towards the end of the 2000s, after 25 years of reign of the relational model and its products (DBMS, methods, tools, industry, etc.) following the appearance of the NoSQL (Not only SQL) paradigm. The latter was motivated by the explosion of varied data on the Web, the digitization of companies, the digital age, social networks and technological advances in terms of hardware (for example, Cloud, parallel machines, etc.), storage and processing units (e.g. GPU, FPGA, etc.) and new business requirements in terms of scalability, availability, elasticity, performance, etc.

However, the benefits of NoSQL come at the cost of relaxing ACID principles: instead of keeping all four parameters consistent throughout each transaction, NoSQL uses the principle of eventual consistency.

NewSQL is a relatively new approach that seeks to unite the best of these latter two worlds using modern programming languages and technology that was not available before. Its goal is to combine the ACID guarantees of SQL with the scalability and high performance of NoSQL.

Unfortunately, today most NewSQL databases are proprietary software or apply only to specific scenarios, which severely limits the spread and adoption of the new technology.

Nowadays, many users of classic DBMSs called "SQL" want to switch to the new NoSQL or NewSQL technologies. On the other hand, the appearance of different models that provide new functionalities, on the one hand, and the absence of tangible proof that justifies this changeover, on the other hand, impose a very relevant question: which NoSQL or NewSQL solution to adopt?

To provide the necessary answers, we are going to develop, within the framework of this article, a critical and comparative analysis between the old relational SQL generation, the NoSQL generation and the new NewSQL generation, to direct users towards the most appropriate solutions according to their needs.

In other words, it is a question of presenting an experimental evaluation of three types of SQL, NoSQL and NewSQL databases, in order to provide a set of criteria and indicators to interested actors, for possible decision-making on the solutions adopted. for their businesses.

2. EVOLUTION OF THE DIFFERENT GENERATIONS OF DBMSS

2.1. Relational DBMS (SQL)

SQL is a relational database management system (RDBMS) and, as the name suggests, it is built around relational algebra and relational tuple calculus. It has been the leading database solution since the 70s and has only recently made room for newcomers[2]. Whatever some people may say, it mostly means that it was, and still is, good enough for a wide range of tasks.

Its main advantages are:

- It uses a single uniform language (DDL) for different roles (developer, user, DBA).
- It uses a single standardized language for different RDBMSs.
- It uses an advanced and non-structural language.
- It respects the principles of ACID (atomicity, consistency, isolation, durability), thus guaranteeing the stability, security and predictability of the entire database and of any transaction in particular.

Many of the benefits of SQL come from its consistency, convenience, and ease of use[3]. Even with very limited knowledge of SQL (or entirely without it, if need be), you can use it reliably with the help of special tools like an online SQL Query Builder.

The downsides, however, make it very unsuitable for certain types of projects. The main problem with SQL is that it is very difficult to use for large-scale projects because its performance degrades quickly as a database grows. Attenuation is also quite problematic.

One of the reasons NoSQL and NewSQL emerged is that older relational databases weren't designed to handle the number of transactions that modern databases must process every second.

2.2. NoSQL (Not Only SQL)

NoSQL is growing in popularity year after year, with its most significant implementations being products such as Apache Cassandra, MongoDB, and others. It was primarily developed to address the characteristic scalability problem of SQL. As a result, it is schema-free and built on distributed systems, making it easy to scale and chip[4].

However, its benefits come at the cost of relaxing ACID principles: instead of keeping all four parameters consistent throughout each transaction, NoSQL uses the principle of eventual consistency. This means that if there are no new updates for a particular piece of data for a certain period of

[2] Cattell, R., Base de données SQL et NoSQL évolutifs. SIGMOD Record, volume 39, Issue 4, Indiana, 2010, p. 12-27

[3] Andreas Meier, Introduction pratique aux bases de données relationnelles, Ed 2, Springer-Verlag, France, 2006, p. 68.

[4] Bruchez R., *Les Bases de données NoSQL Comprendre et mettre en œuvre*, Paris, Eyrolles, 2013, p. 127.

time, all of its hits will eventually revert to the last updated value. This is why such systems are usually described as providing BASE (Basically Available, Soft State, Eventual Consistency) guarantees - as opposed to ACID.

Although this approach greatly increases access time and scalability, it can lead to data loss – the severity of the problem depends on the support of the database server and the quality of the application code. In some cases, this issue could be very serious.

Another problem presented by NoSQL is the fact that there are many types of NoSQL systems, and there is little uniformity between them[5]. Characteristics such as flexibility, performance, complexity, scalability, etc., vary greatly between different systems, which makes it difficult to choose among them even for experienced specialists. Nevertheless, when chosen in accordance with the specifics of the project, NoSQL can provide a much more efficient solution than a SQL system without significant loss of stability[6].

2.3. NewSQL

NewSQL is a relatively new approach that seeks to unite the best of both possible worlds by using modern programming languages and the technology that was not available before. Its goal is to combine the ACID guarantees of SQL with the scalability and high performance of NoSQL.

Obviously, NewSQL looks very promising because of the combination of advantages that in the past only existed separately; and, perhaps, at some point in the future, it will become the standard used by the majority. Unfortunately, today most NewSQL databases are proprietary software or apply only to specific scenarios, which severely limits the spread and adoption of the new technology.

On top of that, NewSQL is anything but homogeneous, and each solution has its own advantages and disadvantages.

These evolutions can be seen as DBMS families. We then distinguish three families: Relational DBMS, NoSQL DBMS and NewSQL DBMS[7]. Table 1 presents a comparison between these three families, using the following criteria: SQL language support, data model used, storage mode used, support for complex queries, joins and data volume.

This comparison shows that the big difference between these three families generally concerns the unsupported horizontal scalability and the complex queries poorly handled by relational DBMSs. Thus, the join is partially supported by NoSQL and the data is weakly secured by NoSQL and NewSQL.

Table 1. Comparison between Relational DBMS, NoSQL and NewSQL

	Relational	**NOSQL**	**NewSQL**
Application	**Transaction**	**Research**	**Analysis**
Plan	table	Key-value, document, column,graph	both
Horizontal evolution	not supported	not supported	supported
SQL language	supported	supported	supported
Complexquery	weak	strong	very strong
join	Supported	partially supported	supported
OLTP	Supported	Unsupported	supported
Transaction	sour	base	sour
Performance	weak	strong	base

3. Classification based on the CAP theorem and the two properties ACID and BASE

This classification is directly related to the constraints of the CAP theorem and to the ACID and BASE properties, the latter being derived from the CAP theorem.[8]

- CAP theorem: Consistency: all nodes in the system see exactly the same data at the same time, availability (Availability): it must be guaranteed that all read or write requests receive a response and partitioning tolerance (Partition-Tolerance): no failure less important than a total network outage must prevent the system from responding correctly.
- At the end of this theorem, the ACID properties appeared in order to guarantee coherence.
- ACID properties (Atomic, Consistent, Isolated, Durable): These are properties that guarantee the reliability of transaction execution. The ACID properties

[5] Adriano Girolamo Piazza, *NoSQL Etat de l'art et benchmark* ; Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES ; Genève, 9 octobre 2013 Haute École de Gestion de Genève (HEG-GE).

[6] Martins G., Bezerra P., Gomes R., Albuquerque F., Costa A., *Évaluer la dégradation du rendement dans les bases de données NoSQL générées par la virtualisation*. Dans Proceedings IEEE of 8th Latin American Network Operations and Management Symposium(LANOMS),1-3 octobre 2015,JoãoPessoa,Brésil, p. 84-91.

[7] Kumar R., Gupta N., Maharwal H., Charu S., *et al.*, *Analyse critique de la gestion des bases de données à l'aide de NewSQL*, International Journal of Computer Science and Mobile Computing (IJCSMC), volume 3, numéro 5, 2014, p. 434-438.

[8] Heinrich L., Architecture *NoSQL et réponse au Théorème CAP*. Bachelor HES (Haute École de Gestion de Genève, 2012.

are: atomicity: this property ensures that a transaction completes completely or not at all, consistency: this property ensures that each transaction brings the system back to a valid state, isolation: this property ensures that any transaction executes as if it were the only one on the system and durability: this property ensures that when a transaction is confirmed, it remains recorded even following a failure.

- The loss of consistency by distributed computer systems has led to the appearance of BASE properties, an alternative to ACID, in order to maintain the reliability of these systems[9].
- BASE properties (Basically Available, Soft State, Eventual Consistency): BASE properties are considered the opposite of ACID properties[10]. They are fundamental availability: all data is distributed and available according to the CAP theorem (high availability). Even in case of failure, the system continues to function and the response will not necessarily be 100% correct, soft state: no guarantee of consistency, the state of the system could change over time, even during idle periods. Due to eventual consistency, the system may have ongoing changes and its state is always soft and eventually consistent: the system will become constant over time and end in a valid state, even when the data is not consistent at some point. Moment t.

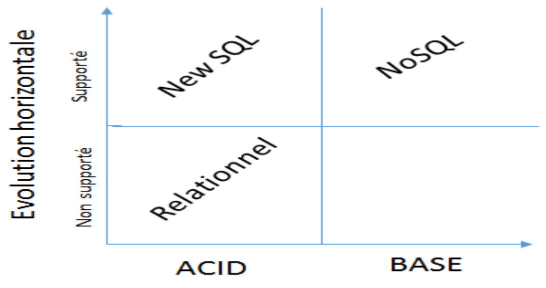


Figure 1-4. Classification of DBMS according to ACID, CAP and BASE properties.

4. EXPERIMENTAL RESULTS AND COMPARATIVE STUDY

This section aims to develop a comparative study between three database management systems of different families, namely MySQL which is part of the SQL family, MongoDB which is part of the NoSQL family and VoltDB which is part of the new NewSQL family. The purpose of this study is to provide information and indicators to guide users towards possible decision-making for the choice of solutions appropriate to their needs and the context of use of the data.

The various tests carried out were varied according to the following three axes:

1. The first concerns the nature of the operations performed in a workload.
2. The second concerns the number of records loaded (600,000 records).
3. The third relates to the number of operations executed (1,000 operations).

The performance of the databases has been defined on the basis of the speed of execution of the operations. Each workload will be launched at least three times on three different days, to keep only the average of the execution times obtained. For a DBMS (NoSQL or relational), each thread performs a sequential series of operations calling on the database interface layer, both to load the database (the load phase) and to execute the workload (phase transaction).

4.1. Loading data

Before executing the different workloads, you must load the data by indicating the number of records to load. The loading times for each DBMS are retrieved from the text files generated (loadjdbc600000.txt, loadmongo600000.txt) during the execution of load-process.

The following table summarizes the load times generated by YCSB for the three systems, once a load of 600,000 records has been loaded:

Table 2. Loading times of the three DBMSs

DBMS	**MySQL**	**MongoDB**	**VoltDB**
time	30576.282 sec	146.68 sec	110.039 sec

In this first test, we can see that the best times were recorded by VoltDB which was much faster compared to MongoDB and MySQL in this first phase of loading.

4.2. Execution of workloads

The workloads will contain 1,000 operations performed on the 600,000 records, with the following results:

Workload A (50% read / 50% update):

DBMS	**MySQL**	**MongoDB**	**VoltDB**
Time	12.23 sec	5.1 sec	3.33 sec

Workload B (95 % Read, 5 % Update) :

DBMS	**MySQL**	**MongoDB**	**VoltDB**
Time	32.16 sec	8.9 sec	1.81 sec

[9] Abramova V. et Bernardino J., *Bases de données NoSQL : MongoDB vs Cassandra*. Dans Proceedings of the International C* Conference on Computer Science and Software Engineering (C3S2E, 13). 10-12 juillet 2013, Porto, Portugal, p. 14-22.

[10] Hecht R., Jablonski S., *Évaluation NoSQL : Enquête axée sur les cas d'utilisation*, Dans Proceedings of the International Conference on Cloud and Service Computing, IEEE. 22-24 novembre 2012, Huashan Road, Shanghaï, p. 336-341.

Workload C (100 % Read) :

DBMS	**MySQL**	**MongoDB**	**VoltDB**
Time	8.47 sec	8.6 sec	1.08 sec

Workload D (50 % Read, 50 % Read-Modify-Write) :

DBMS	**MySQL**	**MongoDB**	**VoltDB**
Time	28.02 sec	9.1 sec	0.95 sec

The experimental results of the various tests carried out in the form of workloads made it possible to evaluate and compare the three types of DBMS SQL (MySQL), NoSQL (MongoDB) and NewSQL (VoltDB) based on the execution time of the different workloads, consisting of 1,000 operations each.

On the first data loading test, VoltDB performed very well, registering large deviations from MySQL with 110 seconds for VoltDB against 9 hours for MySQL. This poor performance of MySQL is explained by the fact that this system is not designed to import or insert large volumes of data, unlike VoltDB which relies on data management.

specific memory.

The different results obtained after running the workloads showed that VoltDB performs very well compared to MondoDB and MySQL in all workloads, especially those that contain heavy write operations in the database, namely: A (50% update), D (50% read-modify-write).

We can justify this performance of VoltDB as well as the slowness of MySQL by the strict requirements relative to the ACID properties imposed by the relational DBMS (lock of the table in the event of writing by preventing the operations of reading), contrary to the NewSQL systems which propose in-memory storage distributed across multiple machines by honoring ACID properties in a reasonable manner with eventual consistency.

In contrast, MySQL produced its best results running the C workload with only read operations.

After reading the results obtained by running the four workloads, we can conclude that the SQL MySQL solution may be adequate for pure read operations in small-scale environments. But in large-scale environments or for read/write operations, it is very interesting to adopt the NoSQL Mongo solution and even better the NewSQL VoltDB.

5. STRATEGIES OF CHOICE

We have seen in the previous sections what these technologies are, what needs they meet and what their selection criteria are. In this section we will summarize these.

The main elements for opposing and choosing these technologies are strong data consistency and horizontal scalability[11]. The graph below summarizes these observations well.

In view of what is shown above, we can conclude that we will choose the relational model if we only need highly consistent data and if our system does not require a strong horizontal load increase. In addition, we will opt for a denormalization of it for better performance with read requests, to the detriment of strong data consistency and horizontal scalability. NoSQL will be adopted for strong horizontal scalability and if the eventual data consistency is sufficient for the system to support.

The latter explains the positioning more to the right on the graph compared to the denormalized relational model, which does not guarantee it.

Given its extreme youth and the lack of sufficient feedback from the companies that implemented it, NewSQL is not included in this graph. However, if we refer to his "promises" we could place it at the very top right.

For more details, we have summarized all the adoption choice criteria for these technologies in the comparative table below.

[11] Hadrien F., - Bachelor HES (Haute École de Gestion de Genève), 2015.

	Relational model	Denormalized relational model	NoSQL
ACID transactions	×	-	-
Possible data consistency	×	×	×
Strong data consistency	×	-	-
Data redundancy	-	×	×
Disk space optimization	×	-	-
Single language for manipulating and defining access rights to data	×	×	-
Easy-to-access database query language for a non-IT specialist	×	×	-
Performance of read requests on a large volume of data	-	×	×
Big data analysis	-	×	×
Reduced query complexity	-	×	×
Horizontal scalability on a large volume of data	-	×	×
Distribution and replication of data without human intervention (or almost) on a large volume of data	-	-	×
Automatic parallelization of request processing without human intervention (or almost)	-	-	×
Have a flexible data schema	-	-	×
Storage, processing and reading of mostly unstructured and semi-structured data	-	-	×
For a large volume of data, their high availability is a priority	-	×	×
Predictive analytics on continuous data streams	-	-	×
Opportunity to reduce costs by investing in mid-range hardware	-	-	×
Representation of data in the form oftwo-dimensional tables	×	×	-
Most widely used data model standard	×	-	-

CONCLUSION

In this work we have made a comparative evaluation of the performance of SQL, NoSQL and NewSQL solutions. The latter currently still meet antagonistic needs. Each of them cannot guarantee both strong data consistency, high availability, horizontal scalability and performance of read requests on very large volumes of data[12]. Thus, we have extracted a set of criteria for choosing these technologies. These will allow those in charge of a computerization project to answer questions such as: Is the architecture I am using the most suitable? Do I have to turn to other architectures? How to choose the architecture according to the project? To achieve this, they will have to be used in concert.

REFERENCES

1. Abramova V., et Bernardino J., *Bases de données NoSQL : MongoDB vs Cassandra*. Dans Proceedings of the International C* Conference on Computer Science and Software Engineering (C3S2E, 13). 10-12 juillet 2013, Porto, Portugal, p. 14-22.
2. Abramova V., Bernardino J., et Furtado P., *SQL ou NoSQL ? Évaluation du rendement et de l'évolutivité*, Int. J. Intégration et gestion des processus opérationnels (IJBPIM), volume 7, numéro 4, 2015, p. 314-321.
3. Adriano Girolamo Piazza, *NoSQL Etat de l'art et benchmark* ; Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES ; Genève, 9 octobre 2013 Haute École de Gestion de Genève (HEG-GE).
4. Andreas Meier, *Introduction pratique aux bases de données relationnelles*, Ed. 2, Springer-Verlag, 2006, p. 68.
5. Bruchez R., *Les bases de données NoSQL Comprendre et mettre en œuvre*, Paris, Eyrolles, 2013, p. 127.
6. Cattell R., *Base de données SQL et NoSQL évolutifs*. SIGMOD Record, volume 39, Issue 4, p. 12-27, Indiana, 2010.

[12] Abramova V., Bernardino J. et Furtado P., *SQL ou NoSQL ? Évaluation du rendement et de l'évolutivité*, Int. J. Intégration et gestion des processus opérationnels (IJBPIM), volume 7, numéro 4, 2015, p. 314-321.

7. Hadrien F., *SQL, NoSQL, NewSQL stratégie de choix*, Bachelor HES (Haute École de Gestion de Genève), 2015.
8. Hecht R., Jablonski S., *Évaluation NoSQL : Enquête axée sur les cas d'utilisation*, Dans Proceedings of the International Conference on Cloud and Service Computing, IEEE. 22-24 novembre 2012, Huashan Road, Shanghai, p. 336-341.
9. Heinrich L., Architecture *NoSQL et réponse au Théorème CAP*. Bachelor HES (Haute École de Gestion de Genève), 2012.
10. Kumar R., Gupta N., Maharwal H., Charu S., *et al*., *Analyse critique de la gestion des bases de données à l'aide de NewSQL*, International Journal of Computer Science and Mobile Computing (IJCSMC), volume 3, numéro 5, 2014, p. 434-438.
11. Martins G., Bezerra P., Gomes R., Albuquerque F., Costa A., *Évaluer la dégradation du rendement dans les bases de données NoSQL générées par la virtualisation*. Dans Proceedings IEEE of 8th Latin American Network Operations and Management Symposium (LANOMS), 1-3 octobre 2015, João Pessoa, Brésil, p. 84-91.
12. Piekos J., *SQL vs NoSQL vs NewSQL : Trouver la bonne solution*. http://dataconomy.com/2015/08/sql-vs-nosql-vs-newsql-finding-the-right-solution/ (consulté en janvier 2018).