UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR
INGENIUS
Revista de Ciencia y Tecnología
https://doi.org/10.17163/ings.n22.2019.05
pISSN: 1390-650X / eISSN: 1390-860X

Performance of Columnar Database

Rendimiento de bases de datos columnares

Jhonatan W. Durán-Cazar[1], Eduardo J. Tandazo-Gaona[1],
Mario R. Morales-Morales[1,*], Santiago Morales Cardoso[1]

Abstract

Companies' capacity to efficiently process a great amount of data from a great variety of sources anywhere and anytime is essential for them to succeed. Data analysis becomes a key strategy for most large organizations to get a competitive advantage. Hence, new issues should be considered when massive amounts of date are to be stored, because traditional relational database are not capable to lodge them. Such questions include aspects that range from the capacity to distribute and escalate the physical storage, to the possibility of using schemes or non-usual types of data. The main objective of this research is to evaluate the performance of the columnar databases in data analysis., comparing them with relational databases, to determine their efficiency using measurements in different test scenarios. The present study seeks to provide (scientific evidence) professionals interested in data analysis with a basic instrument for their knowledge, to include comparative tables with quantitative data that can support the conclusions of this research. A methodology of applied type and quantitative-comparative descriptive design is used, as it is the one of the most appropriate to study database efficiency characteristics. In the measurement, the method of averages is used for a number n of records, and it is supported in the Aqua Data Studio tool that guarantees a high reliability, as a specialized software for the administration of databases. Finally, it has been determined that the columnar databases have a better performance in data analysis environments.

Keywords: data analytics, columnar database, in memory, NoSQL, performance.

Resumen

En la actualidad para el éxito de las empresas es decisiva la capacidad de procesar de manera eficiente una considerable cantidad de datos de una amplia gama de fuentes en cualquier lugar y momento. El análisis de datos se convierte en una estrategia clave para la mayoría de las grandes organizaciones para lograr una ventaja competitiva. Por tanto, surgen nuevas cuestiones a ser tomadas en cuenta a la hora de almacenar y consultar cantidades masivas de datos que, en general, las bases de datos relacionales tradicionales no pueden abarcar. Estas cuestiones incluyen desde la capacidad de distribuir y escalar el procesamiento o el almacenamiento físico, hasta la posibilidad de utilizar esquemas o tipos de datos no usuales. El objetivo principal de la investigación es evaluar el rendimiento de las bases de datos columnares en analítica de datos. Efectuar una comparación con bases de datos de tipo relacional, para determinar su eficiencia, realizando mediciones en distintos escenarios de pruebas. El presente estudio pretende proporcionar (evidencia científica) un instrumento que facilite a los profesionales interesados en la analítica de datos una base para sus conocimientos, al incluir cuadros y tablas comparativos con datos cuantitativos con los que se pueda sustentar las conclusiones de esta investigación. Se usa una metodología aplicada y de diseño descriptivo cuantitativo-comparativo al ser el que mejor se ajusta al estudio de características de eficiencia de bases de datos. En la medición se usa el método de promedios para *n* número de tomas y se soporta en la herramienta Aqua Data Studio que garantiza una alta confiabilidad al ser un programa especializado para la administración de bases de datos. Finalmente, se ha logrado determinar que las bases columnares tienen un mejor rendimiento en ambientes de análisis de datos.

Palabras clave: análisis de datos, base de datos columnar, en memoria, NoSQL, rendimiento

[1,*]Facultad de Ciencias Físicas y Matemáticas, Universidad Central del Ecuador, Ecuador. Autor para correspondencia ✉: mmoralesm@uce.edu.ec. http://orcid.org/0000-0002-8574-1435, http://orcid.org/0000-0002-2209-3952, http://orcid.org/0000-0002-7493-8072, http://orcid.org/0000-0002-3833-9654,

Received: 11-04-2019, accepted after review: 03-06-2019

Suggested citation: Durán-Cazar, J.; Tandazo-Gaona, E.; Morales-Morales, M. R. and Morales Cardoso, S. (2019). «Performance of Columnar Database». Ingenius. N.°22, (july-december). pp. 47-58. doi: https://doi.org/10.17163/ings.n22.2019.05.

1. Introduction

Among the different data models, the relational one has been dominating from the 80s, with database implementations such as Oracle, MySQL and SQL Microsoft Servers, also known as Relational Database Management Systems (RDBMS) [1].

As a consequence of the significant growth of Internet in the last years and the appearance of the big data phenomenon, new issues should be considered when storing and accessing massive amounts of data that, in general, traditional relational databases are not able to cover. These issues include from the capacity to distribute and scale the processing or the physical storage, up to the possibility of employing non usual schemes or types of data [2].

The capacity of efficiently processing a great amount of data from a wide variety of sources, at any place and moment, is decisive for the success of a company. Data analysis becomes a key strategy for most organizations, to obtain a competitive advantage. Therefore, during the last decade, the worldwide focus on the management business has changed profoundly [3].

In a scenario where the data tend to be more different, the rigid structure of relational systems makes significantly difficult to model them. The performance is limited by the vertical scaling, which does not allow the distribution of the system load among multiple machines, together with the great number of concurrent read and write requests and the own complexity of the logic behind the operation of relational databases; all these factors may lead to a efficiency loss regarding the growth of the data.

As a consequence, it is difficult to respond with low latency in the case of applications that simultaneously serve a large number of requests. Therefore, redundant and easy to scale systems are necessary to provide a service with high scalability and availability, to manage large volumes of data and guarantee their availability [4].

Prior to defining how the research will be carried out, it is necessary to review some key concepts that will be used in the present work.

SQL Database.- The concept of database systems is not new in the society, their predecessors were the file systems. As time has gone by, the database was developed due to the requirement of storing a large amount of information.

The relational model was defined in 1970, from which the first relational databases were originated organized as tables (constituted by rows and columns) and with their own query language [5]. These systems provide necessary characteristics in a transactional environment, following the ACID model. The main commercial success of the relational databases was the SQL (Structured Query Language) language, designed and installed at IBM Research, because it became its standard language [6].

Big data.- The digital world is growing very fast, and becomes more complex in terms of volume (terabyte to petabyte), variety (structured, non-structured and hybrid), speed (growing high speed) and nature. This is known as the big data global phenomenon.

This is normally considered as a data collection that has grown up to a point that it cannot be managed nor exploited in an effective manner using traditional data managing tools: for instance, relational database managing systems (RDBMS) or traditional search engines. To handle this problem, traditional RDBMS are complemented by a collection of specially designed alternative database managing systems (DBMS), such as NoSQL [1].

1.1. Technological Platform

The corporate analytics, and related concepts, that describe the analysis of commercial data for decision making, have received wide attention both by the academic and corporate community. The appearance of database systems in memory, has been promoted even more by means of improved data managing procedures and multicore hardware architectures that recently have become available [7].

1.1.1. Architecture

In recent years, some of the most important developments in computing technology are the multicore CPU and the increase in memory capacity based on a 64-bit architecture, which easily supports directly addressable space in terabytes. The multicore architecture enables the parallel massive processing of the database operations, and since all relevant data are permanently stored in the memory, the processing is carried out at the greatest possible speed.

The read operations are completely independent of any access to devices of slower disk storage. On the other hand, the write operations also take place in the memory, but should also be registered in a non-volatile storage to guarantee persistence of the data [8].

1.1.2. In-memory technology

Has been promoted by the need of processing large volumes of data in a very fast manner, and fundamentally by the progress in the processors and the increment in memory capacity based on the 64-bit architecture. This has enabled the parallel massive processing of the database operations, lodging all relevant data in memory [9].

The performance requirement in the Information Technology (IT) domain combined with the advantages of in-memory computing, are important factors that have influenced the appearence of in-memory databases (IMDB) [10].

1.2. In-memory database

The IMDB constitute a database management system designed for a high performance, with the condition that the existent memory is enough to lodge the necessary data. They possess a technique of columnar storage, which enables the access to the data at high speeds and with real-time analytical capabilities. In comparison with Cloud Computing, the advantage for the user is immediately understandable, since it comes from a rapid analysis of big data volumes [3].

1.3. NoSQL databases

The development community desires a flexible database that easily adapts to the new types of data, and is not interrupted by changes in the structure of the content. Unfortunately, the rigid approach defined and based on the scheme utilized by relational databases, makes impossible to rapidly incorporate new types of data. NoSQL provides a data model that better adapts to these needs, since it does not require any type of scheme with fixed tables, as opposed to the traditional model.

In general NoSQL scales horizontally, and prevents the main joining operations in the data. The NoSQL database covers a swarm of multiple databases, each with a different model of data storage [11]. Its popularity has increased due to the need of fast processing in large volumes of data, taking advantage of its highly scalable architecture, flexible data structure (free of schemes), reduced latency and high performance [12]. They can be divided in four categories according to different optimizations:

1.3.1. Key-value database

A key-value storage consists of a set of pairs where one part represents the key, and the other the values, such as text chains or lists, and more complex sets. The data queries are made using keys, not values [13]

1.3.2. Documentary or document-based databases

They are designed to store data from documents that use different formats such as JSON; MongoDB and CouchDB can be mentioned among these databases [14].

1.3.3. Graphic or graph-based databases

These databases store the information as nodes of a graph, and the relations as the edges. They are extensively utilized in recommendation systems and content management, among others. Among these, Neo4J, GraphBase and Infinite Graph are employed most frequently [14].

1.4. Column oriented databases

In the columnar format, all the values of an attribute of the table are stored as a vector using multiple memory blocks, and all the vectors of attributes of a table are stored sequentially. Organizing the values as a vector of attributes enables an easier understanding of the data, and also a high scanning and filtering speed. This results in significant sequential processing, where the columnar format has an enormous advantage compared with the traditional row-oriented disk database. In conjunction with the option of parallel processing, a very high speed can be reached for filtering or any type of aggregation (which constitutes some of main loads in analytical processing). In fact, the speed is so high, that the idea of pre-aggregation of the transactional data, which was the foundation of information systems in previous decades, can be set aside. Besides, additional indices for faster access to the data are not required [8]. A scheme of row and column operations can be observed in Figure 1. Some of the most remarkable functional characteristics include: high compression, implementation, direct operation on compressed data, iteration per block and efficiency of Join operators, among others.

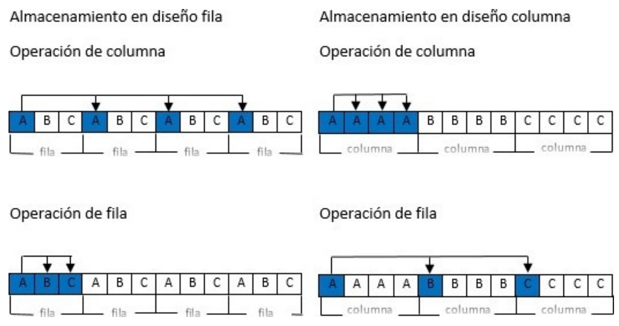

Figura 1. Row and column operations on a data design with rows and columns [8].

1.5. Brewer's Theorem

Since the size of the data grew significantly, it was necessary to find more scalable solutions tan the ACID (Atomicity, Consistency, Isolation and Durability) databases existent so far. As a result, new principles were developed, summarized in the BASE (Basic Availability, Soft-state, Eventual Consistency) paradigm [15].

The ACID properties are centered in the consistency, and are a traditional approach of the databases. Brewer and his team created BASE at the end of the 1990s, to capture the emergent design approaches for high availability. Modern systems, including the Cloud, use a combination of both approaches, traditional and emergent [16].

The objective of Brewer's theorem was to justify the need to explore a broader design space; hence its formulation. The designers and researchers have utilized Brewer's theorem, as a reason to explore a broad variety on novel distributed systems. It has also been applied by the NoSQL movement, as an argument against traditional databases. In a sense, in the NoSQL movement it is about creating options that first focus in availability and then in consistency; the databases that adhere to the ACID properties do the opposite [16].

According to this theorem, it is impossible to simultaneously guarantee the three characteristics when working with distributed systems. Only two of the three features are possible, it is necessary to resign or even partially sacrifice one feature to obtain the others [17].

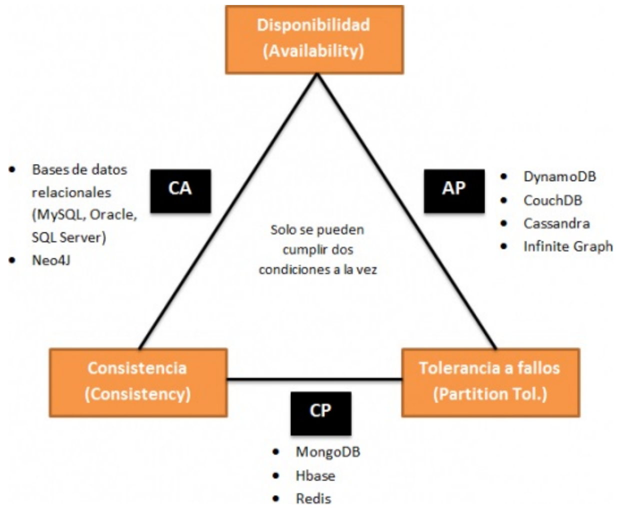
Disponibilidad (Availability)
Bases de datos relacionales (MySQL, Oracle, SQL Server)
Neo4J
CA
AP
DynamoDB
CouchDB
Cassandra
Infinite Graph
Solo se pueden cumplir dos condiciones a la vez
Consistencia (Consistency)
Tolerancia a fallos (Partition Tol.)
CP
MongoDB
Hbase
Redis

Figura 2. Brewer's theorem [18].

- Consistency (C) is equivalent to have a unique updated copy of the data.
- High availability (A) of these data (for updated).
- Tolerance to partitions of the network (P).

A popular way to characterize NoSQL has been to examine its approach to meet Brewer's theorem of coherence, availability and tolerance to partitions (CAP). Most of the NoSQL systems has been designed to sacrifice consistency in exchange of a high availability in a partitioned environment [19]. Figure 2 presents a view of the theorem, in relation to some example databases.

The option of resigning to the tolerance to partition is not feasible in real environments, since there are always partitions in the network. Therefore, in can be deducted that the decision is between availability and consistency, which can be represented as ACID (consistency) and BASE (availability). Nevertheless, Brewer acknowledged that the decision is not binary. All the intermediate spectrum is useful; in general, mixing different levels of availability and consistency yields a better result [15]. The current objective of the theorem should be maximizing combinations of consistency and availability that make sense for a specific application [16].

2. Materials and methods

This work conducts an applied research, with the objective that the final results are utilized in solving corporate problems. The design is descriptive quantitative-comparative, since it aims at specifying what types of databases have a better performance, by measuring and studying their characteristics. The instruments used in the study include standardized tests to compare two groups of databases: columnar and relational.

The procedure that will be utilized comprises the following steps: i) determine the sample, in which the database engines under study are chosen, through a non-probabilistic sampling by criterion, ii) selection/creation of the data set, iii) design of the test scenario, to establish how tests are carried out, which queries will be executed, the number of measurements that will be conducted, among others; the hardware and software infrastructure that will be used is also specified, iv) data loading, where all the databases determined in the sample are loaded, v) measurement, which are carried out using the method of averages and with a specialized tool; similarly, results are registered in all defined scenarios, vi) analysis of results, where the results are interpreted by means of graphs and tables.

2.1. Determining the sample

Before choosing the sample, it was established that the population is constituted by all columnar and relational databases existing up to the present research work. A non-probabilistic sampling by criterion was used for the selection, which is the best type of non-probabilistic sampling. The inclusion and exclusion criteria for delimiting the population are:

- Open source databases (without license).
- Experience of the researchers.

The SQL databases evaluated in this paper are PostgreSQL and MySQL. In comparison with similar databases, they are included in the quadrant among the best open source relational databases [20].

Under the same criteria, the NoSQL databases evaluated are: MongoDB, Cassandra, MonetDB. These alternatives were chosen for being open source and of massive utilization; as can be observed in the Ranking of columnar databases [21], they are pioneers among their peers due to characteristics such as scalability, fault tolerance and columnar storage in conjunction with memory storage.

Another factor that was taken into account is that they can interpret the SQL syntax, which reduces the impact of switching to a NoSQL environment. Although it is not a columnar database, MongoDB is a type of NoSQL database specifically documentary. It was chosen to compare columnar databases, not only with SQL databases, but also with other types of NoSQL databases, documentary in this case. Additionally, MongoDB also employs in-memory technology.

Therefore, the final sample will contain the databases in Table 1, which also shows the family of databases that it belongs to, and the version that will be used in this research.

Tabla 1. Details of databases

Name de datos	Type	Version
MySQL	Relacional – SQL	8.1.0
PostgreSQL	Relacional – SQL	9.6.2
Cassandra	Columnar – NoSQL	3.1.0
MonetDB	Columnar – NoSQL	11.29.3
MongoDB	Documental – NoSQL	3.6.5

2.2. Selection / creation of the data set

An existing set of databases obtained from a public source [22], was chosen to evaluate and compare the performance of the databases. This corresponds to the sales of a large commercial corporation, considering the invoices in the period 2015-2016. The file has a total number of 125,000,000 (125 million) records. The data is stored in CSV files for an easy and uniform access. Table 2 includes a description of the fields contained in the file.

Tabla 2. Description of the fields

Field	Type	Description
Id	INT	Unique identifier
Date	DATE	Product registration date
Store_nbr	INT	Store identifier
Item_nbr	INT	Product identifier
Unit_sales	DECIMAL	Number of units sold, it is an integer number, a negative value represents a return
Onpromotion	BOOLEAN	Indicates if the item was on a promotion for a specific date and store

2.3. Design of the test scenario

The tests with incremental loading of the data will be executed first, i.e., the main data file that contains 125 million records will be divided in the following way: one million, ten million, twenty-five million and fifty million records. The resulting four files will constitute for different scenarios; these four files contain the same number of columns and types of data. The queries to all databases will be executed in these four scenarios. In this way, the performance of the relational databases is tested against the columnar databases in similar scenarios. The specification of the test scenarios are detailed in Table 3.

2.4. Design of queries

Three types of query will be executed in the four scenarios already defined.

i. First query (key-value): this type of query returns a single register of all data set, which will be searched for in the database by means of a key (id). Example:

```
SELECT id, item_nbr, store_nbr, date

FROM train

WHERE id = 500023352;
```

ii. Second query (clause where – data set): the following is considered for its design: the resulting set of data should return at least one third or more of the total of data in each scenario. As can be seen in Table 4, the date changes in each query to return approximately 30% of the total of data.

iii. Third query (aggregation function): it will use the aggregation function SUM() to calculate the total sales of a particular store.

```
SELECT SUM (unit_sales)

FROM train

WHERE store_nbr = '12'
```

Tabla 3. Specifications of the scenarios

Specification	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Data size	1 000 000	10 000 000	25 000 000	50 000 000
Variable	Execution time (ms)			
Description	Three types of queries will be executed: · Key – value · Data set · Agregation function			
Query policies	Queries to a table for relational and columnar databases databases			

Tabla 4. Query (set of data) for the four scenarios

Scenarios	Consult	Returned data
1.st scenario (1 million)	SELECT id, item_nbr, store_nbr, date FROM train1m WHERE date >= '2015-07-05';	388 964
2.nd scenario (10 millions)	SELECT id, item_nbr, store_nbr, date FROM train10m WHERE date >= '2015-09-15';	3 392 156
3.rd scenario (25 millions)	SELECT id, item_nbr, store_nbr, date FROM train25m WHERE date >= '2015-12-31';	8 637 780
4.th scenario (40 millions)	SELECT id, item_nbr, store_nbr, date FROM train50m WHERE date >= '2016-06-25';	16 907 734

2.5. Test environment

The tests were carried out in a single machine with the characteristics described in Table 5.

Tabla 5. Characteristics of the test environment

Software / Hardware	Description
Operating sistema	Ubuntu 14.04 (64-bits)
RAM Memory	16 GB
Processor	AMD Radeon R7, 12 compute Cores 4C + 8G – 3,70 GHz
Hard disk	1 Terabyte

2.6. Measurement

The respective queries were executed in each scenario, to register the response times of each database. Aqua Data Studio, a graphical tool for tasks of administration, design and query in different databases, was used here with the objective that measurements are more reliable and human errors are avoided.

Measurement in the first scenario – 1 million of records (Tables 6, 7, 8).

Tabla 6. First query (key-value)

	Time in (ms) Type of data base	M1	M2	M3	M4	M5	Average
MySQL	SQL	2	1	2	3	2	2
POSTGRESQL	SQL	10	10	8	9	8	9
MONGODB	NoSQL	2	3	3	2	4	2,8
MONETDB	NoSQL	5	3	3	4	5	4
CASSANDRA	NoSQL	4	3	3	4	3	3,4

Tabla 7. Second query (data set)

	Time in (ms) Type of data base	M1	M2	M3	M4	M5	Average
MySQL	SQL	515	594	547	484	516	531,2
POSTGRESQL	SQL	462	468	460	497	453	468
MONGODB	NoSQL	130	124	114	110	189	133,4
MONETDB	NoSQL	191	184	190	189	211	193
CASSANDRA	NoSQL	3	12	11	8	13	9,4

Tabla 8. Third query (aggregation)

	Time in (ms) Type of data base	M1	M2	M3	M4	M5	Average
MySQL	SQL	359	344	360	343	359	353
POSTGRESQL	SQL	155	156	158	155	153	155,4
MONGODB	NoSQL	72	64	70	88	68	72,4
MONETDB	NoSQL	83	96	81	84	84	85,6
CASSANDRA	NoSQL	69	72	61	49	62	62,6

Measurement in the second scenario – 10 million of records (Tables 9, 10, 11).

Tabla 9. First query (key-value)

	Time in (ms)						
	Type of data base	M1	M2	M3	M4	M5	**Average**
MySQL	SQL	4	1	2	2	3	2,4
POSTGRESQL	SQL	9	10	11	10	11	10,2
MONGODB	NoSQL	4	2	2	3	4	3
MONETDB	NoSQL	5	4	4	5	5	4,6
CASSANDRA	NoSQL	5	5	4	6	5	5

Tabla 10. Second query (data set)

	Time in (ms)						
	Type of data base	M1	M2	M3	M4	M5	**Average**
MySQL	SQL	6375	6141	6469	6672	6453	6422
POSTGRESQL	SQL	4960	5739	4720	4119	5420	4991,6
MONGODB	NoSQL	249	255	246	262	245	251,4
MONETDB	NoSQL	267	287	248	235	305	268,4
CASSANDRA	NoSQL	15	35	16	14	22	20,4

Tabla 11. Third query (aggregation function)

	Time in (ms)						
	Type of data base	M1	M2	M3	M4	M5	**Average**
MySQL	SQL	4984	3437	3547	3469	3485	3784,4
POSTGRESQL	SQL	1447	1411	1551	1489	1527	1485
MONGODB	NoSQL	632	588	590	596	628	606,8
MONETDB	NoSQL	716	724	704	705	694	708,6
CASSANDRA	NoSQL	523	538	525	521	518	525

Medición del tercer escenario – 25 millones de registros (Tablas 12, 13, 14).

Tabla 12. First query (key-value)

	Time in (ms)						
	Type of data base	M1	M2	M3	M4	M5	**Average**
MySQL	SQL	3	2	3	4	2	2,8
POSTGRESQL	SQL	11	14	12	12	14	12,6
MONGODB	NoSQL	2	3	2	4	3	2,8
MONETDB	NoSQL	6	6	5	4	4	5
CASSANDRA	NoSQL	6	4	6	4	4	4,8

Tabla 13. Second query (data set)

	Time in (ms)						
	Type of data base	M1	M2	M3	M4	M5	**Average**
MySQL	SQL	14500	14750	14593	14641	16125	14921,8
POSTGRESQL	SQL	12604	12870	12121	11930	11883	12281,6
MONGODB	NoSQL	147	130	153	131	124	137
MONETDB	NoSQL	406	404	419	397	413	407,8
CASSANDRA	NoSQL	17	8	13	13	22	14,6

Tabla 14. Third query (aggregation function)

	Time in (ms)						
	Type of data base	M1	M2	M3	M4	M5	**Average**
MySQL	SQL	10781	10937	10579	9204	9078	10115,8
POSTGRESQL	SQL	4660	3778	4846	4109	3709	4220,4
MONGODB	NoSQL	1488	1765	1776	1487	1454	1594
MONETDB	NoSQL	1818	2513	1823	1799	1788	1948,2
CASSANDRA	NoSQL	1855	1715	1936	1962	2017	1897

Medición del cuarto escenario – 50 millones de registros (Tablas 15, 16, 17).

Tabla 15. First query (key-value)

	Time in (ms)						
	Type of data base	M1	M2	M3	M4	M5	**Average**
MySQL	SQL	3	3	5	2	3	3,2
POSTGRESQL	SQL	12	13	15	12	13	13
MONGODB	NoSQL	4	3	2	2	3	2,8
MONETDB	NoSQL	5	4	4	4	4	4,2
CASSANDRA	NoSQL	7	4	11	6	9	7,4

Tabla 16. Second query (data set)

	Time in (ms)						
	Type of data base	M1	M2	M3	M4	M5	**Average**
MySQL	SQL	28625	28829	29891	29828	29953	29425,2
POSTGRESQL	SQL	24369	24709	26570	25182	26190	25404
MONGODB	NoSQL	295	298	292	293	296	294,8
MONETDB	NoSQL	779	718	654	656	767	714,8
CASSANDRA	NoSQL	19	8	11	25	14	15,4

Tabla 17. Third query (aggregation function)

	Time in (ms)						
	Type of data base	M1	M2	M3	M4	M5	**Average**
MySQL	SQL	21172	21266	21125	20641	20562	20953,2
POSTGRESQL	SQL	7930	8110	9876	8504	8179	8519,8
MONGODB	NoSQL	3196	3337	3446	2978	3667	3324,8
MONETDB	NoSQL	3791	4058	3745	3629	4424	3929,4
CASSANDRA	NoSQL	2989	3212	3015	3172	2905	3058,6

3. Results and discussion

The measurement results are analyzed in two sections, by scenario and by query.

3.1. Results by scenario

The average times, in milliseconds, resulting from the execution of the three queries in the four scenarios (with 1, 10, 25 and 50 million records) are presented and analyzed.

3.1.1. First scenario – 1 million records

Table 18 shows the results obtained, in milliseconds, during the execution of the three queries with a total of 1 million records.

Tabla 18. Results 1 million records

	First scenario Time in (ms)		
	C1 key-value	**C2 data set**	**C3 agregation**
MySQL	2	531,2	353
PostgreSQL	9	468	155,4
MongoDB	2,8	133,4	72,4
MonetDB	4	193	85,6
Cassandra	3,4	9,4	62,6

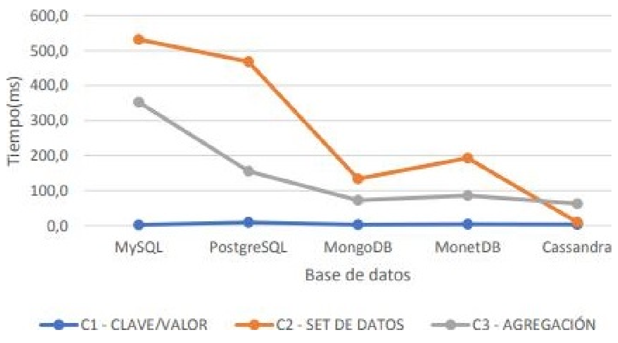

Figura 3. Results 1 million records

Figure 3 shows that for the first query of type key-value, no changes are observed in the execution times among the compared databases. In the second query, in which a where clause that returns a data set is used, variations can be seen between the performance of the databases, with MySQL exhibiting the worst time of response with 531.2 milliseconds, followed by PostgreSQL, compared with the columnar database Cassandra that has a time of response of 9.4 milliseconds, which is 56.51 times more efficient than MySQL. When employing the SUM aggregation function in the third query, it is observed that the best times of responses are obtained with Cassandra, MonetDB and MongoDB; Cassandra is the most efficient with 62.6 milliseconds, which is 5.64 times more efficient than MySQL, that has a time of 353 milliseconds.

3.1.2. Second scenario – 10 million of records

Table 19 shows the results obtained, in milliseconds, during the execution of the three queries with a total of 10 million records.

Tabla 19. Results 10 million records

	Second scenario Time in (ms)		
	C1 key-value	**C2 data set**	**C3 agregation**
MySQL	2,4	6422	3784,4
PostgreSQL	10,2	4991,6	1485
MongoDB	3	251,4	606,8
MonetDB	4,6	268,4	708,6
Cassandra	5	20,4	525

Figure 4 shows a significant difference in the times of response corresponding to the second and third queries, of the columnar databases compared to the relational databases; however, for the first query the times of response in the two types of database keep being regular, without variations. The databases MongoDB, MonetDB and Cassandra had similar times. In the second query, MySQL exhibited the lowest performance with 6422 milliseconds, almost similar to Postgres, compared to Cassandra with a time of 20.4 milliseconds, 314.8 times more efficient than MySQL. In the third query, MySQL again showed the highest time with 3784 milliseconds, compared to the 525 milliseconds obtained with Cassandra; hence, the columnar database was 7.21 times more efficient.

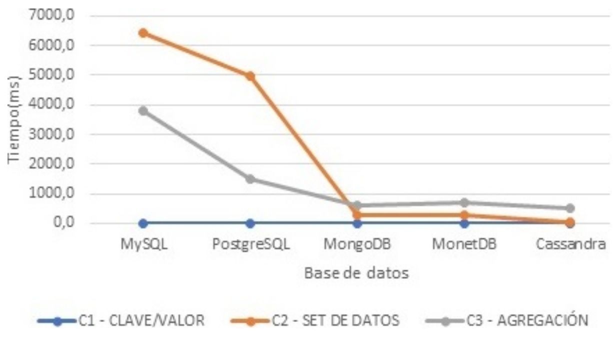

Figura 4. Results 10 million records

3.1.3. Third scenario – 25 million records

Table 20 shows the results obtained, in milliseconds, during the execution of the three queries with a total of 25 million records.

In the results corresponding to the third scenario, which are shown in Figure 5, the times of response for the first query remain similar in all databases. For queries 2 and 3, a good performance is attained for databases MongoDB, MonetDB and Cassandra. Among the column oriented type databases, Cassandra exhibited a time of response similar to MonetDB in queries 2 and 3. In the second query, MySQL exhibited the worst performance with 14921.8 milliseconds, which is 1000 times greater compared to the columnar database Cassandra, which had 14.6 milliseconds. Similarly, in the third query MySQL showed a time of

10115.8 milliseconds, which is 5.38 times slower than the 1879 milliseconds corresponding to Cassandra.

Tabla 20. Results 25 million records

	Third scenario Time in (ms)		
	C1 key-value	**C2 data set**	**C3 agregation**
MySQL	2,8	14921,8	10115,8
PostgreSQL	12,6	12281,6	4220,4
MongoDB	2,8	137	1594
MonetDB	5	407,8	1948,2
Cassandra	4,8	14,6	1897

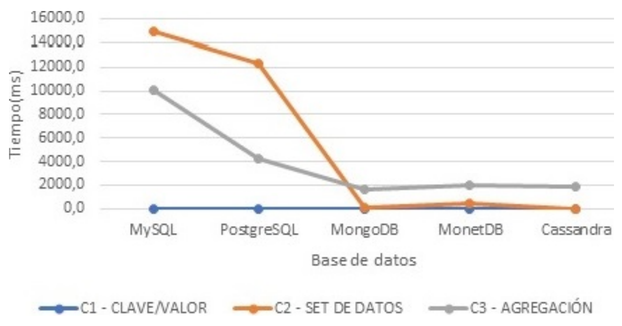

Figura 5. Results 25 million records

3.1.4. Fourth scenario – 50 million records.

Table 21 shows the results obtained, in milliseconds, during the execution of the three queries with a total of 50 million records.

Tabla 21. Results 50 million records

	Fourth scenario Time in (ms)		
	C1 key-value	**C2 data set**	**C3 agregation**
MySQL	3,2	29425,2	20953,2
PostgreSQL	13	25404	8519,8
MongoDB	2,8	294,8	3324,8
MonetDB	4,2	714,8	3929,4
Cassandra	7,4	15,4	3058,6

Figure 6 of the fourth scenario shows that the first query remains without variations, in the time of response in all databases. For queries 2 and 3 it can be observed that the databases with the worst performance are MySQL followed by PostgreSQL. MySQL is 46.1 times slower than MonetDB, while the PostgreSQL responded slightly better in the third query being only 2.7 times slower than Cassandra. The latter is the leader in efficiency, being 46 times faster than its counterpart MonetDB in the second query.

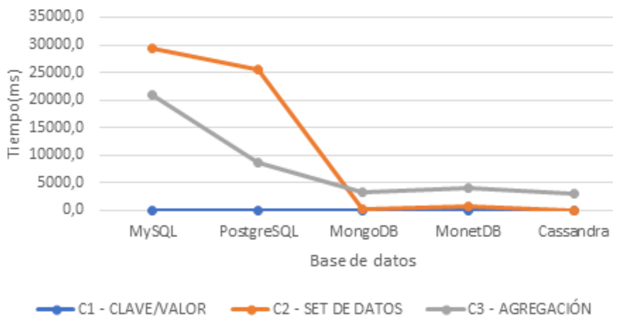

Figura 6. Results 50 million records

3.2. Results by query

The resulting average times, in milliseconds, grouped by query in all scenarios are presented and analyzed.

3.2.1. First query – key-value

Table 22 shows the results obtained, in milliseconds, during the execution of the first query (key-value) in all scenarios.

Tabla 22. Results first query

	First query – key-value # records			
Database	**1 MM**	**10 MM**	**25 MM**	**50 MM**
MySQL	2	2,4	2,8	3,2
PostgreSQL	9	10,2	12,6	13
MongoDB	2,8	3	2,8	2,8
MonetDB	4	4,6	5	4,2
Cassandra	3,4	5	4,8	7,4

It can be observed in Figure 7, that the times of response for the first query are very similar and efficient for all databases. For both MySQL and PostgreSQL, the times of response do not vary significantly as the volume of data grows; the same occurs with the times of response of Cassandra, MongoDB and MonetDB, which remain without notable changes. None of these databases delays more than one second in carrying out this query.

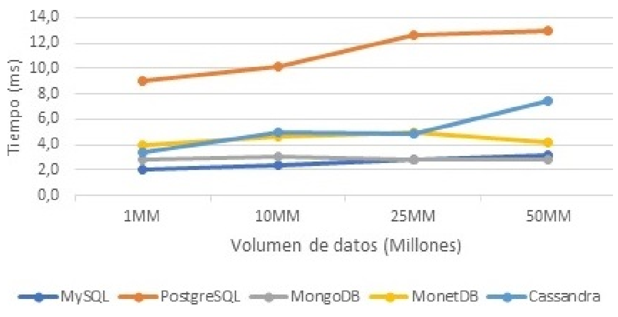

Figura 7. Results first query

3.2.2. Second query – data set

Table 23 shows the results obtained, in milliseconds, during the execution of the second query (data set) in all scenarios.

Tabla 23. Results second query

Second query – data set				
	# records			
Database	1 MM	10 MM	25 MM	50 MM
PostgreSQL	468	4991,6	12281,6	25404
MySQL	531,2	6422	14921,8	29425,2
MongoDB	133,4	251,4	137	294,8
MonetDB	193	268,4	407,8	714,8
Cassandra	9,4	20,4	14,6	15,4

When the second query is executed, a clause where is utilized that returns 30% of all the data. A clear difference in the times of response can be observed in Figure 8 as the number of records increase, between the relational and columnar databases. With 1 MM data, the time of response of MySQL was 531 milliseconds, but with 50 MM data its time of response significantly increased to 29425 milliseconds; the case of PostgreSQL was similar. On the other hand, columnar databases maintain an average time which is independent of the volume of data. For instance, with 1 MM records Cassandra had an execution time of 9.4 milliseconds, while for 50 MM records such time was 15.4 milliseconds.

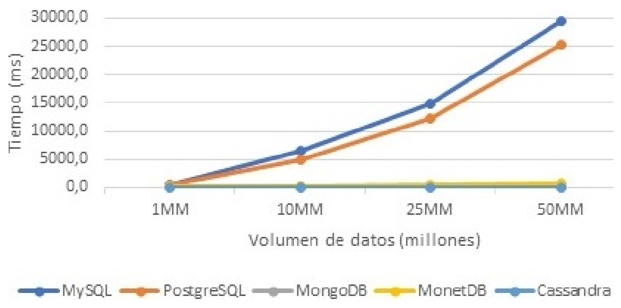

Figura 8. Results second query

3.3. Third query – aggregation function sum ()

Table 24 shows the results obtained, in milliseconds, during the execution of the third query (aggregation function) in all scenarios

Tabla 24. Results third query

Third query – aggregation function (SUM)				
	# records			
Database	1 MM	10 MM	25 MM	50 MM
MySQL	353	3784,4	10115,8	20953,2
PostgreSQL	155,4	1485	4220,4	8519,8
MongoDB	72,4	606,8	1594	3324,8
MonetDB	85,6	708,6	1948,2	3929,4
Cassandra	62,6	525	1897	3058,6

Analyzing Figure 9for 1 and 10 million records, the variations on the times of response in all databases do not exhibit a significant difference, as opposed to the cases when the volume of data increases to 25 and 50 million, for which there is a considerable variation in the time of response between the relational and columnar databases; when the query is executed with 50 million records, the time of response for PostgreSQL is 20953 milliseconds and for MongoDB 3324 milliseconds. As the number of records increases, the difference in performance between MongoDB and PostgreSQL becomes evident. The times of response of Cassandra, MonetDB and MongoDB is slightly affected as the volume of data increases.

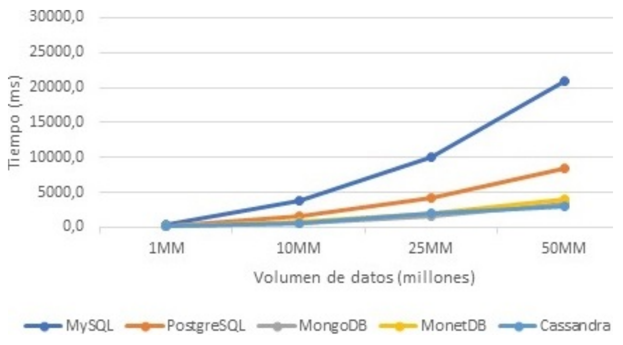

Figura 9. Results third query

Based on the results obtained and the specific characteristics of each database, it was found that for the relational databases MySQL and Postgres there is a directly proportional relationship between volume of data and time, i.e. as the volume of data increases, the time of query increases in a larger proportion. In contrast, for the columnar databases Cassandra and MonetDB, an increase in the volume of data has a smaller impact in the times of response.

The columnar databases exhibit a better performance since they incorporate the in-memory technology (in the RAM memory) for data storage and recovery, which enables a smaller execution time of the queries, as opposed to the relational databases where the performance is affected by the fact that the records should be read from disc, which is much slower compared to the RAM memory.

4. Conclusions

At the end of the present research the stated objectives have been attained, and thus it is concluded that the performance of a columnar database is optimal in data analysis environments.

For the MySQL and Postgres databases, the relationship between volume of data and time is direct and incrementally proportional; on the contrary, in the databases Cassandra and MonetDB that belong to the columnar family, the times of execution do not show notable variations as the volume of data increases.

All the databases compared had the same efficiency in the execution of the first query of type key-value; due to the presence of the primary key, all databases exhibited similar execution times, thus for this query both types of databases have an optimal performance. On the contrary, for the second (data set) and third (aggregation function) queries the difference in the times of execution is rather evident. The superior performance of the columnar databases, which improved the efficiency up to 7.21 and 1900 times in the second and third queries, respectively, is because they highly occupy the volatile memory for data storage and recovery, which enables a smaller execution time of the queries, as opposed to the relational databases for which the performance is not the best, due to the fact that registers should be read from disc, which is much slower than the volatile memory.

The type of columnar databases and, in general, the NoSQL paradigm is adequate for tackling the current big data problem, which refers to the management of large amounts of data. It is therefore recommended to first analyze the business logic, use case and infrastructure, to verify what type of database is the most appropriate for solving problems of interest; regarding this, other existing types of NoSQL databases may be evaluated.

Data analysis requires databases capable of effectively storing and processing large amounts of data, and demands high performance when reading and writing; hence, traditional relational databases are not the most adequate solution. Columnar databases arise as a solution that fulfill performance expectations in this field.

The SQL and NoSQL databases provide different features, and one cannot replace the other. If the system is not flexible in terms of consistency, the relational database administration system is the correct option. If the system can resign to consistency up to a certain point, the NoSQL databases may be the best option to provide more availability, scalability and high performance.

Therefore, depending on the stated objective, a hybrid model, which combines both the SQL and NoSQL technologies, may be the choice in mind; if it is necessary to maintain greater consistency, a relational way of storage may be implemented, while for immediate or recurrent queries, columnar databases would be used.

A future work may consider carrying out the same study, but in a distributed and parallel environment, to contrast and verify the results obtained in this research. There is also the possibility of continuing this study in more depth regarding issues such as configuration and carrying out queries, to take better advantage of these tools. Another future research line would focus in a detailed analysis of writing in columnar databases, with respect to relational databases.

This work summarizes the most important elements and considerations that were totally developed in thesis [23].

Referencias

[1] A. B. M. Moniruzzaman and S. A. Hossain, "Nosql database: New era of databases for big data analytics - classification, characteristics and comparison," *International Journal of Database Theory and Application*, vol. 6, no. 4, pp. 1–5, 2013. [Online]. Available: http://bit.ly/2XaKoPK

[2] M. F. Pollo Cattaneo, M. López Nocera, and G. Daián Rottoli, "Rendimiento de tecnologías nosql sobre cantidades masivas de datos," *Cuaderno Activa*, no. 6, pp. 11–17, 2014. [Online]. Available: http://bit.ly/2Rb8zrO

[3] I. Mihaela-Laura, "Characteristics of in-memory business intelligence," *Informatica Economică*, vol. 18, no. 3, pp. 17–25, 2014. [Online]. Available: http://doi.org/10.12948/issn14531305/18.3.2014.02

[4] D. Robles, M. Sánchez, R. Serrano, B. Adárraga, and D. Heredia, "?'qué características tienen los esquemas nosql?" *Investigación y desarrollo en TIC*, vol. 6, no. 1, pp. 40–44, 2015. [Online]. Available: http://bit.ly/2MJ1wZa

[5] M. Marqués, *Bases de datos.* Universitat Jaume, 2011. [Online]. Available: http://bit.ly/2RcPtS9

[6] E. Ramez and S. B. N., *Fundamentals of Database Systems.* Pearson Education., 2015. [Online]. Available: http://bit.ly/2IG3pAk

[7] G. Hahn and J. Packowski, "A perspective on applications of in-memory analytics in supply chain management," *Decision Support Systems*, vol. 76, pp. 45–52, 2015. [Online]. Available: https://doi.org/10.1016/j.dss.2015.01.003

[8] H. Plattner and B. Leukert, *The In-Memory Revolution. Springer.* Springer, 2015. [Online]. Available: http://bit.ly/2F3ezhO

[9] M. R. Morales Morales and S. L. Morales Cardoso, "Inteligencia de negocios basada en bases de datos in-memory," *Revista Publicando*, vol. 11, no. 2, pp. 201–217, 2017. [Online]. Available: http://bit.ly/2WB3vmC

[10] R. Babeanu and M. Ciobanu, "In-memory databases and innovations in Business Intelligence," *Database Systems Journal*, vol. 6, no. 1, pp. 59–67, July 2015. [Online]. Available: http://bit.ly/2wZLFL7

[11] V. D. Shetty and S. J. Chidimar, "Comparative study of sql and nosql databases to evaluate their suitability for big data application," *International Journal of Computer Science and Information Technology Research*, vol. 4, no. 2, pp. 314–318, 2016. [Online]. Available: http://bit.ly/2KlNZor

[12] A. T. Kabakus and R. Kara, "A performance evaluation of in-memory databases," *Journal of King Saud University - Computer and Information Sciences*, vol. 29, no. 4, pp. 520–525, 2017. [Online]. Available: https://doi.org/10.1016/j.jksuci.2016.06.007

[13] M. T. González-Aparicio, M. Younas, J. Tuya, and R. Casado, "Testing of transactional services in nosql key-value databases," *Future Generation Computer Systems*, vol. 80, pp. 384–399, 2018. [Online]. Available: https://doi.org/10.1016/j.future.2017.07.004

[14] A. Nayak, A. Poriya, and D. Poojary, "Type of nosql databases and its comparison with relational databases," *International Journal of Applied Information Systems (IJAIS)*, vol. 5, no. 4, pp. 16–19, 2013. [Online]. Available: http://bit.ly/2X2fIQQ

[15] S. Simon, "Report to brewer's original presentation of his cap theorem at the symposium on principles of distributed computing (podc) 2000," University of Basel, HS2012, Tech. Rep., 2018. [Online]. Available: http://bit.ly/2XFBo2l

[16] E. Brewer, "Cap twelve years later: How the "rules" have changed," *Computer*, vol. 45, no. 2, pp. 23–29, Feb 2012. [Online]. Available: https://doi.org/10.1109/MC.2012.37

[17] M. Indrawan-Santiago, "Database research: Are we at a crossroad? reflection on nosql," in *2012 15th International Conference on Network-Based Information Systems*, Sep. 2012, pp. 45–51. [Online]. Available: https://doi.org/10.1109/NBiS.2012.95

[18] GENBETA. (2019) Nosql: clasificación de las bases de datos según el teorema cap. [Online]. Available: http://bit.ly/2WHVvR4

[19] R. D. L. Engle, B. T. Langhals, M. R. Grimaila, and D. D. Hodson, "Evaluation criteria for selecting nosql databases in a single-box environment," *International Journal of Database Management Systems (IJDMS)*, vol. 10, no. 4, pp. 1–12, 2018. [Online]. Available: http://bit.ly/2ZgXEQc

[20] Crowd. Inc. (2019) Best relational databases software. [Online]. Available: http://bit.ly/2RbQPge

[21] DB-Engines. (2019) Db-engines ranking of wide column stores. [Online]. Available: http://bit.ly/2KOBYHs

[22] Kaggle. (2019) Corporación favorita grocery sales forecasting. [Online]. Available: http://bit.ly/2F7QYMS

[23] J. W. Durán Cazar, E. J. Tandazo Gaona, and M. R. Morales Morales, *Estudio del rendimiento de una base de datos columnar en el análisis de datos*. Tesis de Grado. Universidad Central del Ecuador, 2018. [Online]. Available: http://bit.ly/2KhB0nl