Artículo Científico / Scientific Paper

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

Rendimiento de bases de datos columnares

Performance of Columnar Database

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

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

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 of large organizations for them to get a competitive advantage. Hence, when massive amounts of date are to be stored, new questionings arise for consideration, because traditional relational database are not capable to lodge them. Such questions include aspects that go 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 the research is to evaluate the performance of the columnar databases in data analytics. Make a comparison with relational databases, to determine their efficiency, making measurements in different test scenarios. The present study aims to provide (scientific evidence) an instrument that provides professionals interested in data analytics with a base 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 that best adjusts to the study of database efficiency characteristics. In the measurement the averages method is used for n number of shots and it is supported in the Aqua Data Studio tool that guarantees a high reliability as it is a specialized software for the administration of databases. Finally, it has been determined that the columnar bases have a better performance in data analysis environments.

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

[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,

Recibido: 11-04-2019, aprobado tras revisión: 03-06-2019

Forma sugerida de citación: Durán-Cazar, J.; Tandazo-Gaona, E.; Morales-Morales, M. R. y Morales Cardoso, S. (2019). «Rendimiento de bases de datos columnares». Ingenius. N.° 22, (julio-diciembre). pp. 47-58. doi: https://doi.org/10.17163/ings.n22.2019.05.

1. Introducción

De los muchos modelos de datos diferentes, el modelo relacional ha estado dominando desde los años 80, con implementaciones como bases de datos Oracle, MySQL y Servidores Microsoft SQL, también conocido como Sistema de Gestión de Base de Datos Relacional (RDBMS) [1].

Debido al gran crecimiento de Internet en los últimos años y la llegada del fenómeno de *big data* (macrodatos), surgen nuevas cuestiones a ser consideradas 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 [2].

La capacidad de procesar una gran cantidad de datos de una amplia gama de fuentes en cualquier lugar y momento de manera eficiente es decisiva para el éxito de las empresas. Para lograr una ventaja competitiva el análisis de datos se convierte en una estrategia clave para la mayoría de organizaciones. Por lo tanto, durante los últimos diez años, el enfoque mundial del negocio de gestión ha cambiado profundamente [3].

En un escenario donde los datos tienden a ser más diferentes que similares entre sí, la estructura rígida de los sistemas relacionales dificulta enormemente su modelamiento. El rendimiento se encuentra limitado por su escalamiento vertical, lo que no permite distribuir la carga del sistema en múltiples máquinas sumado a la gran cantidad de peticiones de lectura y escritura de forma concurrente, la propia complejidad de la lógica detrás del funcionamiento de las bases de datos relacionales tiende a perder eficiencia en relación con el crecimiento de los datos.

Esto hace difícil responder con baja latencia en el caso de aplicaciones que atienden un gran número de pedidos a la misma vez. Por tanto, se hacen necesarios sistemas redundantes y fáciles de escalar que brinden un servicio de alta escalabilidad y disponibilidad para la gestión de elevados volúmenes de datos y así garantizar su disponibilidad [4].

Antes de abordar cómo va a realizarse la investigación es necesario revisar ciertos conceptos claves que han sido utilizados a lo largo del presente trabajo.

Base de datos SQL.- El concepto de sistema de base de datos no es algo nuevo en la sociedad, sus predecesores fueron los sistemas de ficheros. Con el pasar del tiempo, las bases de datos se desarrollaron debido a la necesidad de almacenar gran cantidad de información.

En 1970 se definió el modelo relacional del cual nacieron las primeras bases de datos relacionales organizados como tablas (compuesta por filas y columnas) y su propio lenguaje de consulta [5]. Estos sistemas ofrecen cualidades indispensables en un entorno transaccional siguiendo el modelo ACID. El principal éxito comercial de las bases de datos relacionales fue el lenguaje SQL (Structured Query Language) diseñado e instalado en IBM Research, ya que se convirtió en su lenguaje estándar [6].

Big data.- El mundo digital está creciendo muy rápido y se vuelve más complejo en volumen (terabyte a petabyte), variedad (estructurado, no estructurado e híbrido), velocidad (alta velocidad en crecimiento) y naturaleza. Esto se conoce como el fenómeno global llamado *big data.*

Normalmente, esto se considera como una recopilación de datos que ha crecido tanto que no se puede gestionar o explotar de manera efectiva utilizando herramientas de gestión de datos convencionales: por ejemplo, sistemas de gestión de bases de datos relacionales (RDBMS) o motores de búsqueda convencionales. Para manejar este problema, los RDBMS tradicionales se complementan con un conjunto de sistemas de gestión de bases de datos (DBMS) alternativo especialmente diseñado; tales como NoSQL [1].

1.1. Plataforma tecnológica

La analítica empresarial y los conceptos relacionados que describen el análisis de los datos comerciales para la toma de decisiones han recibido amplia atención tanto en la comunidad académica como en la empresarial. La aparición de sistemas de bases de datos en memoria se ha promovido aún más mediante procedimientos de gestión de datos mejorados y arquitecturas de *hardware* multinúcleo que se han vuelto disponibles recientemente [7].

1.1.1. Arquitectura

Algunos de los desarrollos más importantes de los últimos años en tecnología informática son los CPU multinúcleo y un aumento en la capacidad de memoria basada en una arquitectura de 64 bits, que admite fácilmente terabytes de espacio directamente direccionable. La arquitectura multinúcleo permite el procesamiento masivo en paralelo de las operaciones de la base de datos, y dado que todos los datos relevantes se guardan permanentemente en la memoria, el procesamiento se realiza a la mayor velocidad posible.

Las operaciones de lectura son completamente independientes de cualquier acceso a dispositivos de almacenamiento de disco más lentos. Las operaciones de escritura también tienen lugar en la memoria, pero también deben registrarse en un almacenamiento no volátil para garantizar la persistencia de los datos [8].

1.1.2. Tecnología in-memory

Ha sido propiciada por la necesidad de procesamiento de grandes volúmenes de datos de manera muy rápida

y fundamentalmente por el desarrollo de los procesadores y el incremento en la capacidad de memoria basada en la arquitectura de 64-bits. Esto ha hecho posible el procesamiento paralelo masivo de las operaciones de base de datos, albergando todos los datos relevantes en memoria [9].

La necesidad de rendimiento en el dominio de las Tecnologías de Información (TI) combinado con las ventajas de la computación *in-memory* son factores importantes que han influenciado la aparición de bases de datos in-memory (IMDB) [10].

1.2. *In-memory* database

Las IMDB constituyen un sistema de gestión de base de datos diseñadas para un alto rendimiento con la condición de que exista suficiente memoria para mantener la data necesaria en ella. Poseen una técnica de almacenamiento columnar lo que posibilita el acceso a la data a grandes velocidades y capacidades de analítica en tiempo-real. En comparación con *Cloud Computing*, el beneficio para el usuario es inmediatamente entendible, porque viene de un rápido análisis de grandes volúmenes de datos [3].

1.3. Bases de datos NoSQL

La comunidad de desarrollo desea una base de datos flexible que se adapte fácilmente a los nuevos tipos de datos y no se vea interrumpida por los cambios en la estructura de contenido. Desafortunadamente, el enfoque rígidamente definido y basado en el esquema utilizado por las bases de datos relacionales hace que sea imposible incorporar rápidamente nuevos tipos de datos. NoSQL proporciona un modelo de datos que se adapta mejor a estas necesidades, ya que no requiere ningún tipo de esquemas de tablas fijas, a diferencia del modelo tradicional.

NoSQL generalmente se escala horizontalmente y evita las principales operaciones de unión en los datos. La base de datos NoSQL cubre un enjambre de múltiples bases de datos, cada una con un tipo diferente de modelo de almacenamiento de datos [11]. Su popularidad se ha incrementado debido a la necesidad de procesamiento rápido en grandes volúmenes de datos aprovechado las ventajas de su arquitectura altamente escalable, estructura de datos flexible (libre de esquemas), menor latencia y alto rendimiento [12]. Pueden ser divididas en 4 categorías de acuerdo con diferentes optimizaciones:

1.3.1. Base de datos clave-valor

Un almacén clave-valor consiste de un conjunto de pares donde una parte representa la clave, y la otra a los valores que pueden ser cadenas de texto o listas y conjuntos más complejos. Las búsquedas de datos se realizan contra claves, no valores. [13]

1.3.2. Bases de datos documentales o basadas en documentos

Diseñadas para almacenar data almacenada en documentos que usan diferentes formatos como JSON, entre las cuales se pueden mencionar MongoDB y CouchDB [14].

1.3.3. Bases de datos gráficas o basadas en grafos

En estas bases de datos se almacenan su información en forma de nodos de un grafo y relaciones como aristas de este. Muy utilizadas en sistemas de recomendación, manejo de contenido, entre otras, siendo las más utilizadas Neo4J, GraphBase, Infinite Graph [14].

1.4. Bases de datos orientadas a columnas

En el formato columnar, todos los valores de un atributo de la tabla son almacenados como un vector usando múltiples bloques de memoria y todos los vectores de atributos de una tabla son almacenados secuencialmente. Al organizar los valores en la forma de un vector de atributos permite una fácil compresión de datos y también permite una alta velocidad de escaneo y filtraje. Esto resulta en mucho procesamiento secuencial donde el formato columnar tiene una enorme ventaja comparada con la tradicional base de datos de disco orientado a filas. Junto con la opción de procesamiento paralelo, se puede alcanzar una muy alta velocidad para filtraje o cualquier tipo de agregación (que constituyen algunas de las principales cargas en procesamiento analítico). La velocidad es en efecto tan alta que se puede dejar de lado la idea de preagregación de la data transaccional, la base de los sistemas de información en las décadas pasadas. Además, ya no se requieren índices adicionales para acceso más rápido a la data [8]. Un esquema de operaciones de filas y columnas se observa en la Figura 1.

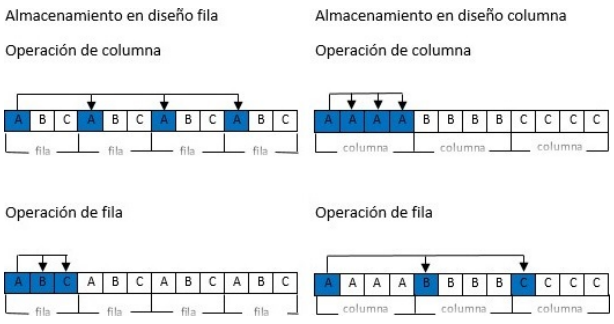

Figura 1. Operaciones de filas y columnas sobre un diseño de datos de filas y columnas [8].

Entre las características funcionales más resaltantes están: alta compresión, materialización, operación directa en datos comprimidos, iteración por bloque, eficiencia en operadores Join, entre otros.

1.5. Teorema de Brewer

Debido a que el tamaño de los datos creció inmensamente, fue necesario encontrar más soluciones escalables que las bases de datos ACID (Atomicity, Consistency, Isolation and Durability) existentes hasta ahora. Como resultado, se desarrollaron nuevos principios, resumidos bajo el paradigma BASE (Basic Availability, Soft-state, Eventual Consistency) [15].

Las propiedades de ACID se centran en la consistencia y son el enfoque tradicional de las bases de datos. Brewer y su equipo crean BASE a finales de la década de 1990 para capturar los enfoques de diseño emergentes para la alta disponibilidad. Los sistemas modernos, incluida la nube, usan una combinación de ambos enfoques, tradicional y emergente [16].

El objetivo del teorema Brewer era justificar la necesidad de explorar un espacio de diseño más amplio, de ahí su formulación. Los diseñadores e investigadores han utilizado el teorema de Brewer como una razón para explorar una amplia variedad de novedosos sistemas distribuidos. El movimiento NoSQL también lo ha aplicado como argumento contra las bases de datos tradicionales. En cierto sentido, en el movimiento NoSQL se trata de crear opciones que se enfoquen primero en disponibilidad y luego en consistencia; las bases de datos que se adhieren a las propiedades de ACID hacen lo contrario [16].

De acuerdo con este teorema cuando se trabaja en sistemas distribuidos es imposible garantizar las tres características simultáneamente. Solo dos de las tres cualidades son posibles, es necesario renunciar o bien sacrificar parcialmente una característica para obtener las otras [17].

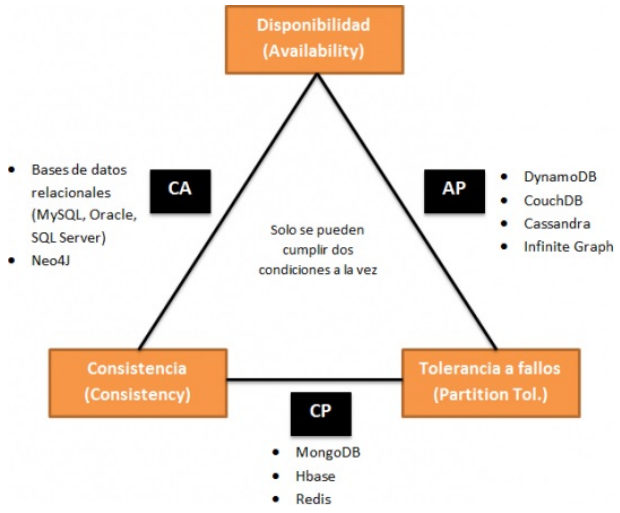

Figura 2. Teorema de Brewe. Adaptado de [18]

- Consistencia (C) equivalente a tener una única copia actualizada de los datos.
- Alta disponibilidad (A) de esos datos (para actualizaciones).
- Tolerancia a las particiones de red (P).

Una forma popular de caracterizar NoSQL ha sido examinar su enfoque para satisfacer el teorema de coherencia, disponibilidad y tolerancia de partición (CAP) de Brewer. La mayoría de los sistemas NoSQL han sido diseñados para sacrificar la consistencia a cambio de una alta disponibilidad en un entorno particionado [19]. La Figura 2 presenta un enfoque del teorema en relación con algunas bases de datos como ejemplo.

La opción de renunciar a la tolerancia de partición no es factible en entornos realistas, ya que siempre se tiene particiones de red. Por lo tanto, se deduce que se debe decidir entre disponibilidad y consistencia, que puede representarse mediante ACID (consistencia) y BASE (disponibilidad). Sin embargo, Brewer reconoció que la decisión no es binaria. Todo el espectro intermedio es útil; mezclar diferentes niveles de disponibilidad y consistencia generalmente produce un mejor resultado [15]]. El objetivo actual del teorema debe ser maximizar las combinaciones de consistencia y disponibilidad que tienen sentido para una aplicación específica [16].

2. Materiales y métodos

Este trabajo desarrolla una investigación aplicada con el objetivo de que los resultados obtenidos al final sean utilizados en la solución de problemas empresariales. El diseño es descriptivo cuantitativo-comparativo ya que pretende precisar qué tipos de bases de datos tienen un mejor rendimiento a partir de la medición y estudio de las características de las mismas. El estudio usa como instrumento pruebas estandarizadas comparando dos grupos de bases de datos: columnares y relacionales.

El procedimiento a seguir se constituye de los siguientes pasos: i) determinar la muestra, en la que se selecciona los motores de bases de datos a estudiar a través de un muestreo no probabilístico por criterio, ii) selección / creación del conjunto de datos, iii) diseño del escenario de pruebas, donde se establece cómo se realizarán las pruebas, qué consultas se ejecutarán, el número de mediciones a efectuar, entre otros; además, se especifica la infraestructura de *hardware* y *software* a usarse, iv) carga de datos, donde se procede a la carga en todas las bases determinadas en la muestra, v) medición, en donde se las realizan bajo el método de promedios y con una herramienta especializada; asimismo se registran los resultados en todos los escenarios definidos, vi) análisis de resultados, en donde se interpretan los resultados mediante gráficos y tablas.

2.1. Determinación de la muestra

Antes de escoger la muestra, se estableció que la población son todas las bases de datos columnares y base de datos relacionales existentes hasta el desarrollo de la presente investigación. Para la elección se usó un muestreo no probabilístico por criterio, este es el mejor tipo de muestreo no probabilístico. Los criterios de inclusión y exclusión para la delimitación poblacional son:

- Base de datos de código abierto (sin licencia).
- Experiencia de los investigadores.

Las bases de datos SQL evaluadas en este artículo son PostgreSQL y MySQL. Por su posicionamiento con respecto a bases de datos similares, como se observa en el cuadrante de mejores bases relacionales código abierto [20].

Bajo los mismos criterios, las bases de datos NoSQL evaluadas serán: MongoDB, Cassandra, MonetDB. Estas alternativas se eligieron por ser de código abierto y de gran utilización, sus características como escalabilidad, tolerancia a fallos, almacenamiento columnar junto con su almacenamiento en memoria las hacen pioneras entre sus pares como se observa en el Ranking de bases de datos columnares [21].

Otro factor que se tomó en cuenta es que pueden interpretar sintaxis SQL, lo que reduce el impacto de cambiar a un ambiente NoSQL. MongoDB, si bien no es una base de datos columnar, es un tipo de base de datos NoSQL, específicamente documental; se la seleccionó para comparar las bases columnares, no solo con bases de datos SQL, sino también con otro tipo de base de datos NoSQL, en este caso documental. Adicional MongoDB también usa tecnología *in-memory*.

Por tanto, la muestra final tendrá las bases de datos expuestas en la Tabla 1, donde se detalla a qué familia de base de datos pertenece y la versión con la que se trabajará durante la investigación.

Tabla 1. Detalle bases de datos

Nombre base de datos	Tipo base de datos	Versión
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. Selección / creación del conjunto de datos

Para evaluar y comparar el rendimiento de las bases de datos, se ha seleccionado un conjunto ya existente que se obtuvo de una fuente pública [22]. Este corresponde a las ventas de una gran corporación comercial con registros de facturas realizadas entre los años 2015-2016. La cantidad de registros que contiene este archivo es 125 000 000 (125 millones). Los datos están grabados sobre archivos CSV para su fácil y uniforme acceso. La descripción de los campos contenidos en el archivo se muestra en la Tabla 2.

Tabla 2. Descripción de los campos

Campo	Tipo	Descripción
Id	INT	Identificador único
Date	DATE	Fecha de registro del producto
Store_nbr	INT	Identificador de las tiendas
Item_nbr	INT	Identificador de los productos
Unit_sales	DECIMAL	Número de unidades vendidas, siendo un número entero, si son valores negativos representa retornos
Onpromotion	BOOLEAN	Indica si el ítem estaba en promoción para un específico *date* y *store*

2.3. Diseño del escenario de pruebas

Primero se ejecutarán las pruebas con cargas incrementales de datos, es decir, el archivo principal de datos que contiene 125 millones de registros se lo dividirá de la siguiente manera: un millón, diez millones, veinticinco millones y cincuenta millones de registros. Ahora se tienen cuatro archivos los cuales conformarán cuatro escenarios distintos; estos cuatro archivos contienen el mismo número de columnas y tipo de datos. En estos cuatro escenarios se ejecutarán las consultas en todas las bases de datos. Con esto se pone a prueba el rendimiento de las bases de datos relacionales frente a las columnares en iguales escenarios. Las especificaciones de los escenarios de prueba se detallan en la Tabla 3.

2.4. Diseño de consultas

Se elaboran tres tipos de consultas que se ejecutarán en los cuatro escenarios de pruebas ya definidos.

i. Primera consulta (clave-valor): este tipo de consulta devuelve un solo registro de todo el conjunto de datos, el cual será buscado en la base por una clave (id). Ejemplo:

```
SELECT id, item_nbr, store_nbr, date

FROM train

WHERE id = 500023352;
```

ii. Segunda consulta (cláusula where - Set de datos): para su diseño se tomó en consideración lo siguiente: el set de datos resultante debe retornar por lo menos un tercio o más del total de datos en cada escenario, por lo tanto, se tuvo cuatro consultas, una por cada escenario. Como se observa en la Tabla 4 en cada consulta cambia la fecha para devolver aproximadamente el 30 % de datos del total.

Tabla 3. Especificaciones de los escenarios

Especificaciones	Escenario 1	Escenario 2	Escenario 3	Escenario 4
Tamaño de datos Variable	1 000 000	10 000 000	25 000 000	50 000 000
	Tiempo de ejecución (ms)			
Descripción	Se ejecutará tres tipos de consultas: · Tipo clave – valor · Set de datos · Función de agregación			
Políticas de consulta	Consultas a una tabla para las bases de datos relacionales y base de dato columnares			

Tabla 4. Consulta (set de datos) para los cuatro escenarios

Escenarios	Consulta	Datos devueltos
1.er escenario (1 millón)	SELECT id, item_nbr, store_nbr, date FROM train1m WHERE date >= '2015-07-05';	388 964
2.° escenario (10 millones)	SELECT id, item_nbr, store_nbr, date FROM train10m WHERE date >= '2015-09-15';	3 392 156
3.er escenario (25 millones)	SELECT id, item_nbr, store_nbr, date FROM train25m WHERE date >= '2015-12-31';	8 637 780
4.° escenario (40 millones)	SELECT id, item_nbr, store_nbr, date FROM train50m WHERE date >= '2016-06-25';	16 907 734

iii. Tercera consulta (función de agregación): usará la función de agregación SUM() para calcular las ventas totales de una tienda determinada.

```
SELECT SUM (unit_sales)

FROM train

WHERE store_nbr = '12'
```

2.5. Entorno de pruebas

Las pruebas se realizaron en un único equipo con las características descritas en la Tabla 5.

Tabla 5. Consulta (set de datos) para los cuatro escenarios

Software / Hardware	Descripción
Sistema operativo	Ubuntu 14.04 (64-bits)
Memoria RAM	16 GB
Procesador	AMD Radeon R7, 12 compute Cores 4C + 8G – 3,70 GHz
Disco duro	1 Terabyte

2.6. Mediciones

Se procede a ejecutar las respectivas consultas en cada escenario para registrar los tiempos de respuesta de cada base de datos, para este fin se utilizó Aqua Data Studio que es una herramienta gráfica para tareas de administración, diseño y consulta en diferentes bases de datos, con el objetivo de que las mediciones sean más confiables y evitar el error humano.

Medición en el primer escenario – 1 millón de registros (Tablas 6, 7, 8).

Tabla 6. Consulta 1 (clave-valor)

	Tiempo en (ms) Tipo de base de datos	M1	M2	M3	M4	M5	Promedio
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. Consulta 2 (set de datos)

	Tiempo en (ms) Tipo de base de datos	M1	M2	M3	M4	M5	Promedio
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. Consulta 3 (función de agregación)

	Tiempo en (ms) Tipo de base de datos	M1	M2	M3	M4	M5	Promedio
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

Medición en el segundo escenario – 10 millones de registros (Tablas 9, 10, 11).

Tabla 9. Consulta 1 (clave-valor)

	Tiempo en (ms) Tipo de base de datos	M1	M2	M3	M4	M5	Promedio
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. Consulta 2 (set de datos)

	Tiempo en (ms) Tipo de base de datos	M1	M2	M3	M4	M5	Promedio
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. Consulta 3 (función de agregación)

	Tiempo en (ms) Tipo de base de datos	M1	M2	M3	M4	M5	Promedio
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. Consulta 1 (clave-valor)

	Tiempo en (ms) Tipo de base de datos	M1	M2	M3	M4	M5	Promedio
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. Consulta 2 (set de datos)

	Tiempo en (ms) Tipo de base de datos	M1	M2	M3	M4	M5	Promedio
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. Consulta 3 (función de agregación)

	Tiempo en (ms) Tipo de base de datos	M1	M2	M3	M4	M5	Promedio
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. Consulta 1 (clave-valor)

	Tiempo en (ms) Tipo de base de datos	M1	M2	M3	M4	M5	Promedio
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. Consulta 2 (set de datos)

	Tiempo en (ms) Tipo de base de datos	M1	M2	M3	M4	M5	Promedio
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. Consulta 3 (función de agregación)

	Tiempo en (ms) Tipo de base de datos	M1	M2	M3	M4	M5	Promedio
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. Resultados y discusión

Analizamos los resultados de las mediciones en dos secciones, resultados por escenario y resultados por consulta.

3.1. Resultados por escenario

Aquí se presentan y analizan los tiempos promedio en milisegundos resultantes de la ejecución de las tres consultas en los cuatro escenarios con 1, 10, 25 y 50 millones de registros.

3.1.1. Primer escenario – 1 millón de registros

La Tabla 18 muestra los resultados en milisegundos, obtenidos durante la ejecución de las tres consultas con un total de un millón de registros.

Tabla 18. Resultados 1 millón de registros

Primer escenario Tiempo en (ms)			
	C1 clave-valor	**C2 set de datos**	**C3 agregación**
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

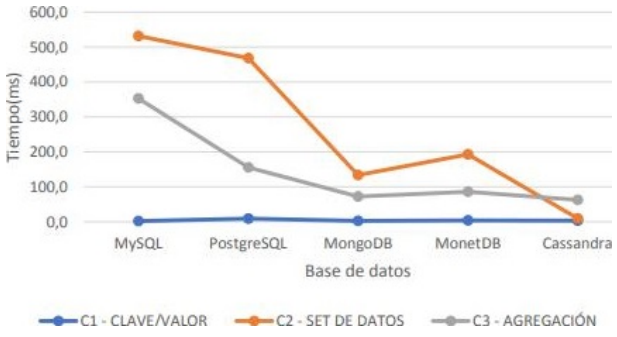

Figura 3. Resultados 1 millón de registros

La Figura 3 muestra que para la primera consulta de tipo clave-valor no se observan cambios en los tiempos de ejecución entre las bases de datos comparadas. En la segunda consulta donde se usa una cláusula where que retorna un set de datos, ya se observa variaciones entre el rendimiento de las bases de datos, en donde MySQL muestra el peor tiempo de respuesta con 531,2 milisegundos seguida de PostgreSQL, a comparación de la base de datos columnar Cassandra que presenta un tiempo de respuesta de 9,4 milisegundos, esto es un tiempo 56,51 veces más eficiente que el mostrado por MySQL. En la tercera consulta al emplear la función de agregación SUM se observa que los mejores tiempos de respuesta se obtienen con Cassandra, MonetDB y MongoDB, siendo Cassandra la de mejor eficiencia con un tiempo de 62,6 milisegundos que es 5,64 más eficiente a comparación de MySQL que tiene un tiempo de 353 milisegundos.

3.1.2. Segundo escenario – 10 millones de registros

La Tabla 19 muestra los resultados en milisegundos, obtenidos durante la ejecución de las tres consultas con un total de 10 millones de registros.

Tabla 19. Resultados 10 millones de registros

Primer escenario Tiempo en (ms)			
	C1 clave-valor	**C2 set de datos**	**C3 agregación**
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

En la Figura 4 se observa una notable diferencia en los tiempos de respuesta de la segunda y tercera consultas entre las bases de datos columnares en comparación con las bases de datos relacionales; sin embargo, para la primera consulta los tiempos de respuesta en los dos tipos de base de datos sigue manteniéndose regular sin variaciones. Las bases MongoDB, MonetDB y Cassandra tuvieron tiempos similares. En la segunda consulta el rendimiento más bajo presentó MySQL que expuso un tiempo de 6422 milisegundos casi similar a Postgres, que a comparación con Cassandra con un tiempo de 20,4 milisegundos; esta última es 314,8 veces más eficiente que MySQL. La tercera consulta se mantiene MySQL con el tiempo de respuesta más bajo con 3784 milisegundos a comparación de Cassandra con 525 milisegundos siendo 7,21 veces más eficiente la base de datos columnar.

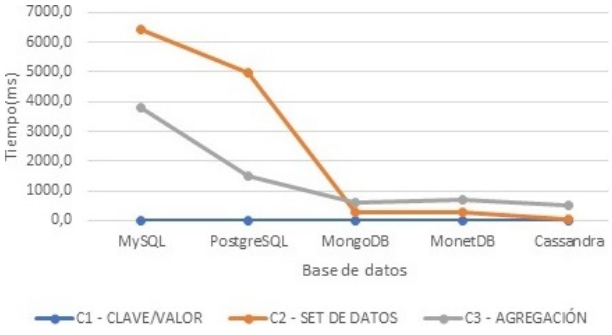

Figura 4. Resultados 10 millones de registros

3.1.3. Tercer escenario – 25 millones de registros

La Tabla 20 muestra los resultados en milisegundos obtenidos durante la ejecución de las tres consultas con un total de 25 millones de registros.

Los resultados del tercer escenario mostrado en la Figura 5 se mantienen semejantes los tiempos de respuesta para la primera consulta en todas las bases de datos. Para las consultas 2 y 3 se logra un buen rendimiento en las bases de datos MongoDB, MonetDB y Cassandra. Entre las bases de datos del tipo orientado a columnas, Cassandra en las consultas 2 y 3 exhibió un tiempo de respuesta similar a MonetDB. En la segunda consulta el peor rendimiento fue

presentado por la base de datos de MySQL con 14 921,8 milisegundos, esto es un tiempo de ejecución 1000 veces mayor en comparación con la base de datos columnar Cassandra con 14,6 milisegundos. Al igual que en la tercera consulta MySQL presenta un tiempo de 10 115,8 milisegundos, lo cual es 5,38 veces más lento que Cassandra con 1879 milisegundos.

Tabla 20. Resultados 25 millones de registros

	Primer escenario Tiempo en (ms)		
	C1 clave-valor	**C2 set de datos**	**C3 agregación**
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

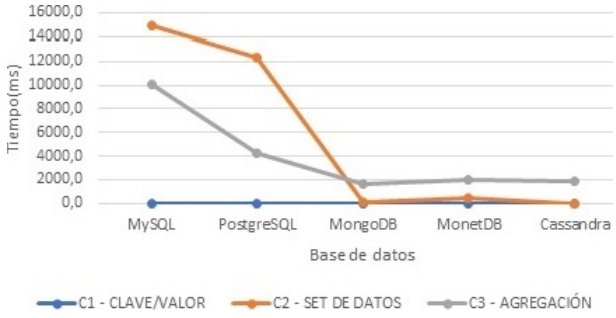

Figura 5. Resultados 25 millones de registros

3.1.4. Cuarto escenario – 50 millones de registros

La Tabla 21 muestra los resultados en milisegundos, obtenidos durante la ejecución de las tres consultas con un total de 50 millones de registros.

Tabla 21. Resultados 50 millones de registros

	Primer escenario Tiempo en (ms)		
	C1 clave-valor	**C2 set de datos**	**C3 agregación**
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

La Figura 6 del cuarto escenario muestra que la primera consulta permanece sin variaciones en el tiempo de respuesta en todas las bases de datos. Para las consultas 2 y 3 se puede observar que las bases de datos con peor rendimiento son MySQL seguida de PostgreSQL. MySQL es 46,1 veces más lenta que MonetDB. PostgreSQL respondió un poco mejor en la tercera consulta siendo solo 2,7 veces más lenta que Cassandra. Esta última lidera en eficiencia siendo 46 veces más rápida que su homóloga MonetDB en la segunda consulta.

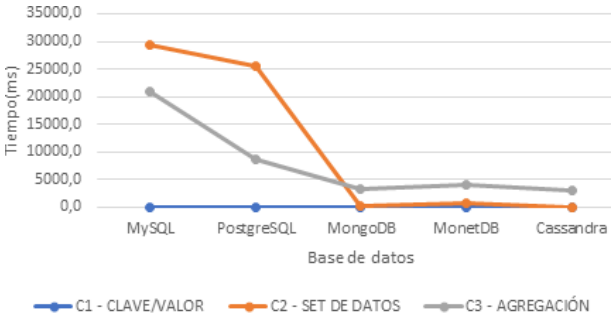

Figura 6. Resultados 50 millones de registros

3.2. Resultados por consulta

Aquí se presentan y analizan los tiempos promedio de respuesta resultantes agrupados por consulta en todos los escenarios.

3.2.1. Primera consulta – clave-valor

La Tabla 22 muestra los resultados en milisegundos, obtenidos durante la ejecución solo de la primera consulta (clave-valor) en todos los escenarios.

Tabla 22. Resultados primera consulta

	Primera consulta – clave-valor # registros			
Base de datos	**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

En la Figura 7 se observa que para la primera consulta los tiempos de respuesta son muy similares y eficientes en todas las bases de datos, tanto en MySQL como en PostgreSQL los tiempos de respuesta no varían considerablemente mientras crece el volumen de datos, lo mismo ocurre con los tiempos de respuesta de Cassandra, MongoDB y MonetDB que permanecen sin cambios notables. Ninguna de estas bases de datos demora más de un segundo en realizar esta consulta.

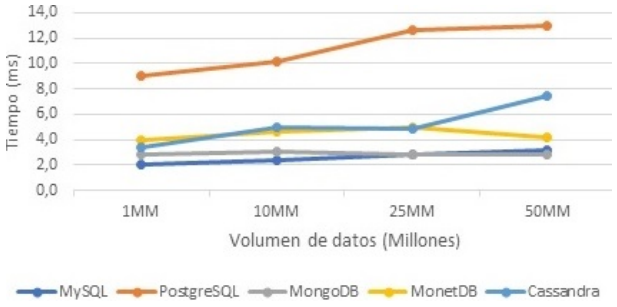

Figura 7. Resultados primera consulta

3.2.2. Segunda consulta – set de datos

La Tabla 23 muestra los resultados en milisegundos, obtenidos durante la ejecución de la segunda consulta (set de datos) en todos los escenarios.

Tabla 23. Resultados primera consulta

Resultados segunda consulta # registros				
Base de datos	**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

En la Figura 8 cuando se ejecuta la segunda consulta que utiliza una cláusula where que retorna el 30 % de todos los datos. Se observa una clara diferencia en los tiempos de respuesta cuando va aumentado la cantidad de registros, entre las bases de datos relacionales y columnares. MySQL con 1M de datos su tiempo de respuesta fue 531 milisegundos, pero con 50M de datos su tiempo de respuesta tiene un fuerte incremento presentando un tiempo de 29425 milisegundos, caso similar presenta PostgreSQL. Mientras que con bases de datos columnares mantienen un tiempo promedio independiente del volumen de datos. Como Cassandra que con 1M de registros su tiempo de respuesta es 9,4 milisegundos y con 50M de registros presentó un tiempo de respuesta de 15,4 milisegundos.

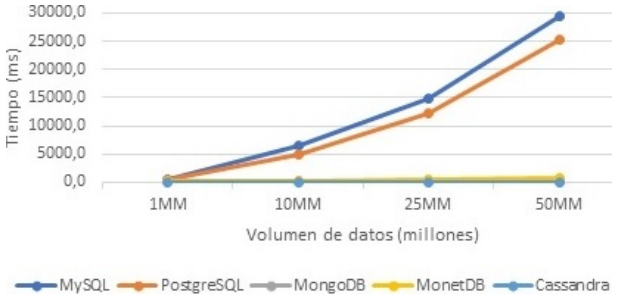

Figura 8. Resultados segunda consulta

3.3. Tercera consulta – función de agregación sum ()

La Tabla 24 muestra los resultados en milisegundos, obtenidos durante la ejecución de la tercera consulta (función de agregación) en todos los escenarios.

Tabla 24. Resultados primera consulta

Resultados tercera consulta # registros				
Base de datos	**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

Analizando la Figura 9 para 1 y 10 millones de registros, las variaciones en tiempos de respuesta en todas las bases de datos no arrojan una diferencia significativa, en cambio, cuanto aumenta el volumen de datos a 25 y 50 millones respectivamente ya se tiene una variación considerable en los tiempos de respuesta entre el tipo de base relacionales y columnares, PostgreSQL cuando se ejecuta la consulta con 50 millones de registros el tiempo de respuesta es 20 953 milisegundos y en MongoDB 3324 milisegundos. A medida que aumenta la cantidad de registros, la diferencia de rendimiento entre MongoDB y PostgreSQL se hace evidente. Cassandra, MonetDB y MongoDB son ligeramente afectadas en el tiempo de respuesta conforme incrementa el volumen de datos.

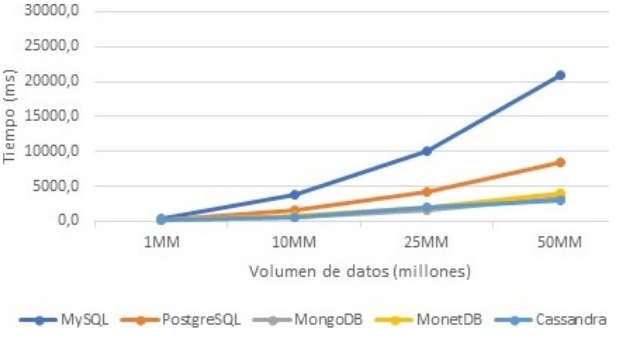

Figura 9. Resultados tercera consulta

Considerando los resultados obtenidos y las características propias de cada base de datos, se encontró que, en las bases de datos relacionales MySQL y Postgres, hay una relación directamente proporcional entre el volumen de datos y el tiempo, es decir, mientras se incrementa el volumen de datos, el tiempo de consulta también se incrementa en mayor proporción. A diferencia de las bases de datos columnares Cassandra y MonetDB, en las cuales, al incrementar el volumen de datos, tiene un impacto menor en los tiempos de respuesta.

Las bases de datos columnares tienen un rendimiento superior debido a que ocupan tecnología in-memory (en memoria RAM) para el almacenamiento y la recuperación de datos, lo que permite un menor tiempo de ejecución de las consultas, a diferencia del tipo de base de datos relacionales donde el rendimiento se ve afectado debido al hecho de que los registros deben leerse desde el disco, que es mucho más lento en comparación con la memoria RAM.

4. Conclusiones

Como resultado de la presente investigación se ha podido cumplir con los objetivos planteados, por lo cual, se concluye que el rendimiento de una base de datos columnar es óptimo en ambientes de análisis de datos.

En las bases de datos MySQL y Postgres la relación entre el volumen de datos y el tiempo es directa e incrementalmente proporcional, al contrario, en las bases de datos de la familia columnar Cassandra y MonetDB, los tiempos de ejecución no sufren variaciones notables mientras aumenta el volumen de los datos.

Todas las bases de datos comparadas tuvieron la misma eficiencia en la ejecución de la primera consulta tipo clave-valor debido a la presencia de la clave primaria, todas las bases presentaron tiempos de ejecución similares, por tanto, para este tipo de consultas, ambos tipos de bases de datos tienen un óptimo rendimiento. A diferencia de los resultados en la segunda consulta (set de datos) y la tercera consulta (función de agregación) donde la diferencia de tiempos de ejecución es bastante notoria.

El rendimiento superior de las bases de datos columnares que presentó mejoras de hasta 7,21 y 1900 veces más eficiencia en la tercera y segunda consulta respectivamente, se debe a que ocupan altamente la memoria volátil para el almacenamiento y la recuperación de datos, lo que permite un menor tiempo de ejecución de las consultas, a diferencia del tipo de base de datos relacionales donde el rendimiento no fue el mejor debido al hecho de que los registros deben leerse desde el disco, que es mucho más lento en comparación con la memoria volátil.

El tipo de base de datos columnar y el movimiento NoSQL en general, es adecuado para resolver el problema actual del big data, que es el manejo de grandes cantidades de datos. Por lo cual se recomienda analizar primero la lógica de negocio, caso de uso e infraestructura para verificar qué tipo de base de datos es la más apta para la solución de las problemáticas que se posean, referente a esto se puede evaluar otros tipos de base de datos NoSQL que existen en el mercado.

El análisis de datos necesita bases de datos capaces de almacenar y procesar grandes cantidades de datos con eficacia, la demanda de alto rendimiento al leer y escribir, así que la base de datos relacional tradicional no resulta ser la solución más adecuada. Bases de datos columnares surgen como una solución que cumple con las expectativas de rendimiento en este campo.

Las bases de datos SQL y NoSQL proporcionan diferentes características y una no puede reemplazar a otra. Si el sistema no es flexible en términos de consistencia, entonces el sistema de administración de base de datos relacional es la opción correcta. Si el sistema puede renunciar a cierta consistencia, las bases de datos NoSQL pueden ser la mejor opción para proporcionar más disponibilidad, escalabilidad y alto rendimiento.

Por lo cual dependiendo del objetivo que se persiga, se podría pensar en un modelo híbrido que combine las dos tecnologías SQL y NoSQL, donde sí se necesita mantener mayor consistencia se puede almacenar de una manera relacional mientras que para consultas inmediatas o recurrentes, se utilizarían bases de datos columnares.

Como trabajo futuro, se podría realizar el mismo estudio, pero en un entorno distribuido y paralelo para contrastar y verificar los resultados obtenidos en esta investigación, también se deja la posibilidad de continuar este estudio más a profundidad en temas de configuración y elaboración de las consultas para obtener un mejor provecho de estas herramientas. Otra línea futura de investigación se enfocaría en un análisis detallado de escritura en bases de datos columnares con respecto a base de datos relacionales.

El presente trabajo resume los elementos y consideraciones más importantes que fueron desarrollados en su totalidad en el trabajo de tesis de [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