Análisis Comparativo de Rendimiento entre MongoDB y Neo4j: Una Evaluación de Operaciones CRUD

Anthony Montero Roman Email: anthonymr2010@estudiantec.cr
Kun Kin Zheng Liang Email: kzheng@estudiantec.cr
Escuela de Administración de Tecnologías de Información
Instituto Tecnológico de Costa Rica

***Abstract*—This report presents a comparative performance analysis between two popular NoSQL databases: MongoDB and Neo4j. A set of tests was implemented focusing on CRUD operations (Create, Read, Update, and Delete) with realistic data and controlled environment conditions. The execution time of each operation was evaluated to identify strengths and weaknesses of each system in specific scenarios. Results show that MongoDB offers excellent performance in bulk data insertion and simple queries, while Neo4j excels in updating and deleting records, especially when data relationships are complex. These conclusions provide clear guidelines for selecting the most suitable database according to requirements and workload type in modern applications.**

***Palabras clave*—NoSQL, MongoDB, Neo4j, Rendimiento, CRUD, Análisis Comparativo**

I. Introducción

En la actualidad, gestionar grandes volúmenes de información se ha convertido en un reto que abarca desde pequeñas startups hasta gigantes corporativos. Para afrontar esta necesidad, han surgido las bases de datos NoSQL (Not Only SQL) [1], [2], que ofrecen mayor flexibilidad de esquemas, escalabilidad horizontal y la capacidad de manejar datos no estructurados a gran escala. Dentro de este ecosistema, existen cuatro categorías principales: documentales, grafos, clave-valor y columnares, cada una pensada para satisfacer distintos casos de uso y requerimientos [3], [4].

Entre ellas, MongoDB [5] se posiciona como un referente en la categoría documental gracias a su habilidad para almacenar datos en documentos de estilo JSON con esquemas dinámicos. Esto permite una representación más intuitiva de datos jerárquicos y facilita tanto las actualizaciones frecuentes como las consultas complejas. No es de extrañar que su popularidad vaya en aumento en sectores como comercio electrónico y aplicaciones de contenido, donde su robusta escalabilidad horizontal y su eficiencia con datos semi-estructurados son altamente valoradas [2].

Por otro lado, Neo4j [6] se basa en un enfoque de grafos, resaltando la importancia de las relaciones entre datos tanto como la de los datos mismos. Este modelo ofrece una ventaja significativa al manejar datos que están fuertemente interconectados, permitiendo analizar relaciones complejas de manera más rápida y eficiente que los esquemas tradicionales. Gracias a ello, Neo4j se utiliza con éxito en ámbitos como redes sociales, sistemas de recomendación y análisis de fraude, donde la conexión entre diferentes entidades es clave para obtener información valiosa.

Dado que MongoDB y Neo4j representan dos paradigmas diferentes dentro de NoSQL, compararlos adquiere especial relevancia. Mientras que MongoDB sobresale en el manejo de datos jerárquicos y consultas sobre documentos, Neo4j destaca cuando se requiere navegar y analizar relaciones profundas. Reconocer estas diferencias es esencial para entender cómo rinden en operaciones CRUD (Create, Read, Update, Delete), las cuales resultan imprescindibles en la mayoría de aplicaciones.

En lo que sigue en este documento, se describen inicialmente el diseño del experimento y la metodología de evaluación utilizadas. Posteriormente, se presentan los resultados obtenidos a través de las pruebas comparativas de rendimiento y se discuten los hallazgos más relevantes. Finalmente, se exponen las conclusiones y recomendaciones, destacando los escenarios en los que cada base de datos ofrece las mejores ventajas.

II. Diseño del Experimento y Metodología de Evaluación

Para llevar a cabo este estudio comparativo entre MongoDB y Neo4j, se diseñó un experimento centrado en las operaciones CRUD (Crear, Leer, Actualizar y Eliminar) que permitiera medir su desempeño de forma clara y sistemática. El objetivo principal era obtener resultados cuantitativos que facilitaran el análisis posterior de la eficiencia y rendimiento de cada base de datos.

En primer lugar, la configuración del entorno de pruebas se estableció en una misma máquina con sistema operativo Windows 10, equipada con un procesador Intel(R) Core(TM) i7-8650U, arquitectura de 64 bits y 500 GB de almacenamiento. En esta máquina se instalaron las dos bases de datos objeto de estudio: MongoDB (versión 8.0.4) en el puerto 27017 y Neo4j (versión 5.25.0) en el puerto 7687. Para garantizar condiciones equivalentes, tanto la asignación de recursos como la ausencia de otras cargas de trabajo se mantuvieron bajo control durante la ejecución de las pruebas.

El conjunto de datos empleado consiste en un archivo CSV que contiene aproximadamente 10,200 registros de información relacionada con contenidos de streaming (series y películas). En cada registro se incluyen campos como el identificador

del show, el tipo de contenido (series o películas), el título, el director, el elenco, el país de origen, la fecha de inclusión en la plataforma, el año de estreno, la clasificación por edades, la duración, el género (o categorías) y la descripción. Este archivo posee un tamaño aproximado de 3.61 MB, y se cargó de forma idéntica en ambas bases de datos para asegurar la consistencia de las pruebas.

La metodología de evaluación se diseñó a través de la clase denominada ‘DatabaseTester‘, la cual implementa las operaciones CRUD de manera controlada. Para la operación de creación, se realizó una inserción masiva de los 10,200 registros y se midió el tiempo total que tomó completar esta carga inicial. En cuanto a la operación de lectura, se seleccionaron 100 identificadores aleatorios y se midió el tiempo individual de cada consulta, obteniendo así el tiempo promedio, así como el mínimo y el máximo de las 100 lecturas realizadas. La operación de actualización se centró en modificar de manera masiva un campo específico en todos los registros, registrando el tiempo total dedicado a esta tarea. Finalmente, la operación de eliminación implicó el borrado masivo de los registros y el cálculo del tiempo necesario para completar dicha acción.

Con la intención de observar el desempeño de cada operación de forma clara, se recopilaron métricas de tiempos de ejecución (promedio, mínimo y máximo) con una granularidad de milisegundos. Además, para presentar los resultados de manera intuitiva, se generaron gráficos de barras comparativos, mapas de calor (heatmaps) de rendimiento y tablas que muestran los datos de forma detallada. Para la medición del tiempo se emplearon las funciones del módulo ‘time‘ de Python, y en el entorno de desarrollo se utilizaron librerías como ‘pymongo‘ y ‘neo4j‘ para gestionar las conexiones a las bases de datos, así como ‘pandas‘ y ‘seaborn‘ para la manipulación y visualización de la información obtenida.

Por último, se cuidaron varios aspectos para garantizar la validez del experimento. Se empleó la misma máquina para las pruebas de ambas bases de datos, de modo que no existieran diferencias en términos de hardware. Asimismo, se utilizó el mismo conjunto de datos, sin alteraciones, asegurando una comparación justa. La conexión a las bases de datos fue local y directa, sin cargas concurrentes o procesos adicionales que pudieran distorsionar los tiempos de respuesta. Con estas precauciones, la comparación refleja de manera objetiva el rendimiento de MongoDB y Neo4j en operaciones CRUD con una carga de datos realista y condiciones de prueba equitativas.

III. Análisis de Resultados

En la Figura 1 y 2 se presentan los resultados obtenidos del experimento comparativo entre MongoDB y Neo4j, los cuales revelaron patrones significativos en el rendimiento de operaciones CRUD bajo condiciones controladas. El análisis de los tiempos de ejecución muestra diferencias sustanciales que merecen una evaluación detallada.

A. CREATE

La operación de creación de registros fue la que reflejó la mayor disparidad de rendimiento entre los dos sistemas.

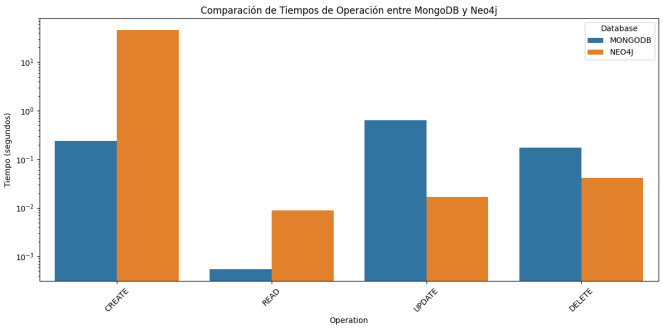

Figura 1. Comparación de Tiempos de Operación ente MongoDB y Neo4j.

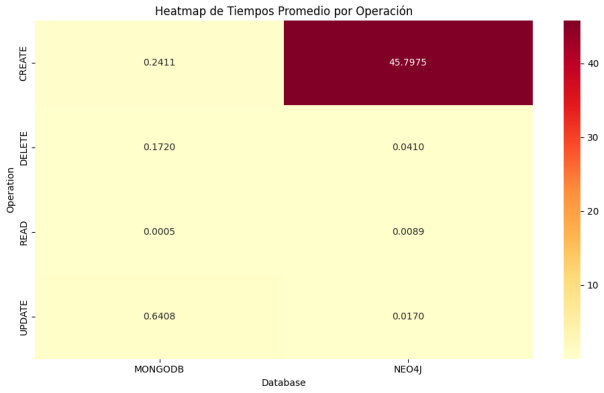

Figura 2. Heatmap de Tiempos Promedio por Operación.

Mientras MongoDB completó la inserción masiva de 10,200 registros en tan solo 0.2411 segundos, Neo4j requirió 45.7975 segundos para la misma tarea.

Esta brecha, cercana a dos órdenes de magnitud, deja ver claramente la eficiencia de la arquitectura orientada a documentos de MongoDB para la ingesta de datos no relacionales. La estructura basada en documentos JSON parece optimizar la carga masiva de información de manera mucho más efectiva que el modelo de grafos de Neo4j, al menos bajo las condiciones y el conjunto de datos evaluados en este experimento.

B. READ

En las pruebas de lectura, ambas bases de datos lograron resultados más similares. MongoDB respondió en 0.0005 segundos frente a los 0.0089 segundos de Neo4j. Aunque se aprecia una diferencia, esta no resulta tan drástica como en la inserción de datos.

Estos tiempos de respuesta sugieren que tanto MongoDB como Neo4j implementan mecanismos de indexación y acceso a datos razonablemente eficientes. Sin embargo, es relevante mencionar que esta evaluación se centró en consultas simples por identificador. En escenarios con consultas más complejas o que involucren relaciones múltiples, los resultados podrían cambiar de forma considerable.

C. UPDATE

El panorama se vuelve interesante en el caso de la actualización masiva de registros. Contrariamente a lo que podría esperarse si se asumiera la superioridad de las bases documentales en esta área, Neo4j se impuso con un tiempo de 0.0170 segundos, mientras que MongoDB necesitó 0.6408 segundos.

Esta notable diferencia sugiere que el modelo de grafos de Neo4j puede manejar cambios en nodos y propiedades con gran agilidad, probablemente al no requerir la reestructuración de documentos completos. El resultado plantea un escenario en el que Neo4j se vuelve muy atractivo para aplicaciones que requieran un gran volumen de modificaciones de datos de manera frecuente.

D. DELETE

Por último, en lo referente a la eliminación de registros, Neo4j también mostró un rendimiento superior (0.0410 segundos) frente a los 0.1720 segundos de MongoDB. Esta ventaja refuerza la idea de que la gestión interna de nodos y relaciones de Neo4j resulta eficiente no solo para actualizar, sino también para eliminar grandes volúmenes de información. En cambio, MongoDB, pese a su buen desempeño en otras operaciones, se vio superado en este ámbito.

IV. Conclusiones

En primer lugar, las evidentes diferencias arquitectónicas marcan el rumbo del rendimiento. MongoDB demostró una gran superioridad en la creación masiva de registros, siendo hasta 190 veces más rápido que Neo4j. Por otra parte, Neo4j destacó en las operaciones de actualización y eliminación, superando a MongoDB por factores de 37 y 4, respectivamente. Este hallazgo confirma que cada base de datos está diseñada para optimizar determinados tipos de operaciones, con MongoDB sobresaliendo en el ingreso de información y Neo4j mostrando mayor eficiencia en la modificación y borrado de datos. En cuanto a la lectura, los resultados fueron comparables, lo que sugiere que ambos sistemas gestionan bien las consultas simples por identificador.

En segundo lugar, pueden distinguirse escenarios de uso óptimos para cada base de datos. MongoDB resulta una gran elección cuando el sistema requiere manejar un gran volumen de inserciones o documentos con estructuras jerárquicas, así como para consultas rápidas y simples. En contraste, Neo4j se vislumbra como la opción ideal para aplicaciones que exigen numerosas actualizaciones, eliminaciones frecuentes o una gestión intensiva de relaciones. La forma en que cada modelo de datos (documental en MongoDB y grafos en Neo4j) organiza y procesa la información se traduce en un rendimiento diferencial en cada tipo de operación.

Por último, se observa que el impacto del modelo de datos va más allá de la simple estructura: la forma de almacenar y vincular la información incide de manera decisiva en los tiempos de respuesta y en la escalabilidad. MongoDB, gracias a su almacenamiento en documentos JSON, favorece la ingesta masiva de datos no relacionales, mientras que el enfoque en grafos de Neo4j brilla al modificar o eliminar nodos y relaciones, lo que parece ser clave en escenarios donde las actualizaciones y los borrados son constantes.

V. Recomendaciones

En aquellos proyectos que enfrentan alta frecuencia de inserciones o sistemas de logging y comercio electrónico, la adopción de MongoDB permitirá gestionar de manera ágil la entrada de información y las consultas por identificador. En cambio, si el desarrollo se orienta hacia redes sociales, motores de recomendación o aplicaciones con actualizaciones y eliminaciones regulares, Neo4j resultará mucho más adecuado.

Para optimizar el rendimiento, en el caso de MongoDB se recomienda profundizar en la creación de índices apropiados, así como evaluar técnicas de sharding que faciliten la distribución de datos en grandes volúmenes. De igual modo, la definición cuidadosa de la estructura de los documentos puede evitar sobrecostos a la hora de actualizar. En Neo4j, conviene diseñar la estructura de grafos de forma que reduzca los cuellos de botella al momento de crear nuevos registros, y aprovechar los índices para propiedades consultadas con frecuencia. Asimismo, las operaciones en lotes (batch operations) resultan beneficiosas para mejorar la velocidad de inserción.

Un punto esencial es la realización de pruebas de carga específicas antes de adoptar una base de datos en producción, corroborando que el rendimiento observado se mantenga en escenarios reales. También se aconseja mantener las versiones de MongoDB y Neo4j actualizadas, ya que las mejoras periódicas en cada lanzamiento suelen impactar de manera directa en la velocidad y la confiabilidad.

Referencias

[1] J. Pokorný, “Nosql databases: A step to database scalability in web environment,” *International Journal of Web Information Systems*, vol. 9, no. 1, pp. 69–82, 2013.

[2] G. Harrison, *Next Generation Databases: NoSQL and Big Data*. New York, USA: Apress, 2015.

[3] R. Hecht and S. Jablonski, “Nosql evaluation: A use case oriented survey,” in *Proceedings of the 2011 International Conference on Cloud and Service Computing (CSC)*, pp. 336–341, IEEE, 2011.

[4] K. Pramod and J. Smith, “Sql vs nosql databases: A comparative analysis,” *Journal of Data Management*, vol. 14, no. 2, pp. 45–53, 2018.

[5] “Mongodb documentation.” https://docs.mongodb.com/. Último acceso: 20 de enero de 2025.

[6] “Neo4j official documentation.” https://neo4j.com/docs/. Último acceso: 20 de enero de 2025.