Comparación del desempeño de una base de datos relacional y una de grafos

Marcos Sander
Instituto de Computación
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay
mrsantar@gmail.com

***Resumen*—El objetivo de este trabajo es hacer una comparación entre dos bases de datos, una clásica relacional y otra moderna no relacional de grafos. De esa forma se pretende observar y contrastar el comportamiento de ambas bases sobre estructuras de datos representables con grafos. En particular se realizó la comparación utilizando MySQL como base relacional y Neo4j como base de datos de grafos. Este trabajo se basa en el trabajo realizado por Vicknair y colaboradores. Como resultado se observa que ambas bases se desempeñan correctamente y permiten trabajar con este tipo de datos. Sin embargo, se obtienen mejores resultados con la base de datos de grafos en consultas donde se toma en cuenta la estructura tipo grafo.**

I. Introducción

A fines del año 2009 se acuñó el termino NoSQL para referirse a las bases de datos no relacionales que estaban surgiendo en esa época. Este nuevo paradigma que forma parte de la tercera revolución de bases de datos sucedió a mediados de los años 2000 motivado por los requerimientos de las nuevas aplicaciones web de uso masivo que las bases de datos relacionales ya no podían satisfacer [Harrison(2015)]. Un tipo de bases de datos NoSQL es la base de datos de grafos, donde se representa a la información como nodos de un grafo y las relaciones como aristas del mismo. En este trabajo se compara el desempeño de consultas realizadas sobre una base de datos relacional MySQL con el desempeño en una base de datos de grafos Neo4j, donde ambas bases tienen datos representables con grafos. El resto del documento se organiza de la siguiente forma. En la sección II se presentan los trabajos relacionados. Luego, en la sección III se presenta el trabajo realizado, análisis y diseño para generar la base de datos, los datos y las decisiones tomadas de implementación, así como las consultas que se utilizaron para realizar la comparación de desempeño. Le sigue los experimentos realizados y resultados en la sección IV. Por último, en la sección V se presentan las conclusiones obtenidas y posibles trabajos a futuro.

II. Trabajos relacionados

En el trabajo de Vicknair y colaboradores [Vicknair et al.(2010)Vicknair, Macias, Zhao, Nan, Chen, and Wilkins], se compara la performance de una base de datos relacional MySQL con una de grafos Neo4j sobre datos donde la relación entre ellos genera una estructura de grafo dirigido acíclico (DAG). En particular es del interés de los autores determinar si una tecnología de datos tradicional como las bases de datos relacionales o una de grafos es más efectiva para el desarrollo de un sistema de procedencia de datos. Tales sistemas guardan la información acerca de la procedencia de un dato, describiendo de donde procede, qué es y hacia donde va y dicha información se puede representar con DAGs. A su vez, los autores generan las bases de datos tanto relacionales como de grafos que representan la misma información y con una estructura similar a los datos que quieren analizar. Por último ejecutan consultas tanto sobre la estructura del grafo (consultas que recorren dicha estructura), así como otras consultas sobre los datos que contiene cada nodo del grafo, para poder comparar la performance de ambas bases.

Por otro lado en el trabajo de Almabdy [Almabdy(2018)], se realiza una comparación de la perfomance nuevamente utilizando Neo4j y MySQL pero sobre datos de redes sociales, en particular Twitter. Los resultados que se obtienen indican una mejor performance si se utiliza la base de datos de grafos.

Por último en el trabajo de Macack y colaboradores [Macak et al.(2020)Macak, Stovcik, and Buhnova], se compara también la perfomance de bases de datos de grafos y bases de datos relacionales para análisis de Big Data, usando Neo4j como base de datos de grafos y PostgreSQL como base de datos relacional. A diferencia de los anteriores trabajos mencionados ellos realizan tests en un cluster de tres máquinas para poder comparar comportamientos de las bases de datos en un escenario del tipo Big Data.

III. Diseño de experimentos

En esta sección se describe la solución implementada para poder realizar la medición de performance.

III-A. Bases de datos utilizadas

En esta subsección se describe las bases de datos utilizadas: MySQL y Neo4j.

III-A1. MySQL: Es una base de datos relacional, con muchos años en el mercado (creada en el 1994 por MySQL AB y adquirida por Sun Microsystems en el año 2008). Es utilizada ampliamente en la industria, en web populares como: Youtube, Wikipedia, Facebook, Google, Flickr y Twitter. Por ende, es una tecnología con una comunidad madura. Implementa transacciones con características ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad [MySQL()]) propias de muchas de las bases de datos relacionales.

III-A2. Neo4j: Es la base de datos de grafos seleccionada para hacer las pruebas. Posee la característica que es transaccional. También es software libre y tiene versiones libres como la Neo4j Community Edition aunque también versiones comerciales libres y pagas (ver [Neo4j()]). Es una de las implementaciones de bases de grafos pioneras y a la vez de las más usadas.

III-B. Configuración del ambiente para pruebas

Se crearon 12 bases de datos, 6 tanto en MySQL como en Neo4j. Esas 6 bases tienen diferentes tamaños (1000, 5000 y 10000 nodos con aproximadamente 2000, 10000 y 20000 aristas respectivamente). Cada base MySQL tiene su correspondiente Neo4j y viceversa, donde ambas representan el mismo grafo. La información que se guarda en cada nodo puede ser un valor entero o un campo texto. Esto se hizo siguiendo la guía del artículo de referencia [Vicknair et al.(2010)Vicknair, Macias, Zhao, Nan, Chen, and Wilkins] ya que simula el tipo de información que guardan las bases de datos de procedencia.

III-B1. Generación de datos: Para generar los datos de cada base se siguió como guía el trabajo tomado como referencia [Vicknair et al.(2010)Vicknair, Macias, Zhao, Nan, Chen, and Wilkins]. En MySQL, se genera una representación del grafo con un esquema relacional sencillo. Una tabla nodo(id, info) se utiliza para representar a los nodos del grafo, que tienen su id y la información relacionada (que puede ser tanto un valor entero como un valor de texto de hasta 4KB depediendo de la base). A su vez se crea una tabla arista(id_nodo_a, id_nodo_b) para representar la arista que parte del nodo con id id_nodo_a hacia un nodo con id id_nodo_b. Tanto el campo id_nodo_a como id_nodo_b son cláves foráneas al campo id de la tabla nodo.

Para crear la base de forma aleatoria, se realiza el proceso generando el grafo en capas. Primeramente se generan los nodos de la primer capa, luego los de la segunda capa, y a partir de ahí las aristas que unen a los nodos de la capa anterior con la actual. Tanto la cantidad de nodos de cada capa, como la cantidad de aristas entrantes que tiene cada nodo son sorteadas aleatoriamente en un rango [1, maxNodosCapa] ó [0, maxAristasEntrantes] preestablecido. Se continua el proceso hasta llegar a la cantidad de nodos que debe tener esa base.

El proceso de creación de datos se hizo en forma paralela tanto en la base MySQL como en la base Neo4j. En cada paso del algoritmo se genera aleatoriamente la estructura correspondiente, esto es nodos de la capa y aristas conectando a esos nodos con la capa anterior. Por cada nodo o arista obtenido, se inserta dicha información tanto en la base relacional (insertando los registros de nodo y registros de aristas), como en la de grafos (creando el nodo con sus propiedades y la relación arista correspondiente).

La información de cada nodo puede ser un entero o un texto dependiendo si es una base de enteros o texto. Para el caso de enteros, para cada nodo se graba un entero elegido aleatoriamente. Por otro lado para el caso de campo texto, inicialmente se genera un pseudodiccionario a partir de un texto en español de longitud suficiente, tomando sus palabras como parte de ese diccionario. Luego se sortea aleatoriamente la cantidad de palabras que tendrá dicho texto, y cada palabra es obtenida de forma aleatoria del diccionario. La información, aunque no tiene sentido, permite realizar búsquedas de nodos que contengan cierta información como puede ser un texto o dos palabras seguidas, entre otras consultas.

En la tabla I se puede observar cada base generada, así como el tamaño de cada una en nodos y aristas, y el tamaño en mega bytes (MB) que ocupa en disco.

III-B2. Consultas: También usando como guía el paper de referencia [Vicknair et al.(2010)Vicknair, Macias, Zhao, Nan, Chen, and Wilkins] se crearon 2 tipos de consultas: estructurales y de datos. Ambas tienen sentido en sistemas de procedencia, pero también pueden tener sentido en diferentes dominios que utilicen representaciones de grafos dirigidos acíclicos.

Se hicieron tres consultas estructurales, que recorren el grafo buscando alguna característica en las relaciones de los nodos:

- E1- Encontrar nodos aislados. Se busca en el grafo todos los nodos que no tienen conexión con ningún otro.
- E2- Buscar hijos. A partir de un nodo inicial, se buscan todos sus nodos hijos, hasta una profundidad de 5 y se devuelve la cantidad de nodos encontrados.
- E3- Buscar hijos en profundidad grande. A partir de un nodo inicial se buscan todos sus hijos hasta una profundidad de 40 y se devuelve la cantidad de nodos encontrados.

También se hicieron cuatro consultas sobre los datos de nodos. Las consultas N1 y N2 se realizan sobre las bases con datos numéricos y las consultas T1 y T2 sobre las bases con datos de texto.

- N1- Contar la cantidad de nodos cuyo valor numérico es mayor a cierto valor dado.
- N2- Contar la cantidad de nodos cuyo valor numérico es igual a cierto valor dado.
- T1- Contar la cantidad de nodos cuyo valor texto contiene una palabra dada.
- T2 - Contar la cantidad de nodos cuyo valor texto contiene una secuencia de 3 palabras.

Cuadro I: Bases de datos generadas

Base	**#nodos**	**#aristas**	**Dato**	**MySQL**	**Neo4j**
1000int	1000	1810	Int	0,2MB	0,2MB
5000int	5000	9650	Int	0,7MB	1,1MB
10000int	10000	18874	Int	2,2MB	2,2MB
1000char	1000	1912	Char	2,7MB	4,1M
5000char	5000	9608	Char	13MB	20,1MB
10000char	10000	19549	Char	28MB	41MB

se observa cada base generada, tamaño en nodos y aristas y espacio en disco utilizado, así como el tipo de dato que se almacena

III-C. Comparación subjetiva

Se hizo también una comparación subjetiva respecto a algunos aspectos de ambas bases: Madurez, Lenguajes de consulta y Herramientas de acceso a la base de forma de poder comparar otros aspectos importantes de ambas bases.

IV. Experimentación

Los experimentos se realizaron en una notebook con las siguientes características:

- Procesador Intel Core i7-4702MQ, 2.20GHz
- Memoria RAM 16.0 GB
- Sistema Operativo Windows 10, 64-bits
- Disco Duro Mecánico modelo HGST HTSS541010A9E680, 5400 rpm, SATA, 6 Gbps y 8 MB de caché. Las bases se guardan en una partición con un sistema de archivos NTFS y 170 GB libres.

IV-1. Resultados ejecución de consultas: En la tablas II, III y IV se observan los resultados obtenidos al realizar las consultas estructurales sobre todas las bases, las consultas sobre bases con nodos con datos numéricos y las consultas sobre bases con nodos con datos de texto respectivamente.

Respecto a las consultas estructurales se observa que Neo4j tiene mejores resultados respecto a MySQL. En la consulta de recorrida de hijos para cierto nodo, en las bases de datos de texto se vió un mejor desempeño con Neo4j. Este resultado no se explica por el hecho de tener datos tipo texto, ya que la estructura de esta base debería ser similar a la estructura de la base de datos de datos numéricos. Una prueba de interés sería ver el desempeño en la consulta de hijos de un nodo con mayor profundidad ya que teóricamente en ese caso Neo4j debería mostrar mejores resultados. Esta consulta (E3), no se pudo realizar en este trabajo ya que devolvía errores y no se pudo determinar la causa.

En las consultas sobre nodos con datos numéricos se observa un mejor desempeño de MySQL. Un resultado similar obtuvieron Vicknair y colaboradores [Vicknair et al.(2010)Vicknair, Macias, Zhao, Nan, Chen, and Wilkins] y según mencionan en su artículo, Neo4j utiliza Lucene por defecto y trata a los datos numéricos como texto lo que hace más lenta la comparación por igualdad o desigualdad.

Por último en las consultas sobre bases con nodos con datos de texto se observa un mejor desempeño de Neo4j, y se ve una mayor diferencia de desempeño a medida que el tamaño de la base crece. Neo4j es aproximadamente 10 veces más rápido en las consultas de comparación de texto sobre la base de 10 mil nodos.

Cuadro II: Resultados de consultas estructurales en milisegundos

Base	E1-R	E1-G	E2-R	E2-G
1000int	3	11	3	8
5000int	12	12	8	8
10000int	21	15	7	8
1000char	2	8	8	6
5000char	291	10	16	7
10000char	1291	14	14	7

Cuadro III: Resultados de consultas numéricas en milisegundos

Base	N1-R	N1-G	N2-R	N2-G
1000int	0	6	0	5
5000int	1	8	1	7
10000int	2	9	2	9

IV-A. Comparación subjetiva

IV-A1. Madurez: Si bien Neo4j es una herramienta que ya tiene más de 10 años desde su creación, es una comunidad menos madura que MySQL si la medimos a la madurez como cantidad de desarrolladores que la utilizan y tiempo de trabajo con la herramienta. De todas formas durante el trabajo ante dudas de desarrollo y consultas que realice pude encontrar las respuestas a preguntas típicas en foros de desarrollo o propios de Neo4j.

IV-A2. Herramientas de acceso a la base: Debido a los años que está en el mercado y a su popularidad, MySQL posee varias herramientas de gestión que permiten consultar, crear y eliminar datos, tablas y bases en MySQL. En este proyecto se utilizó como herramienta de apoyo el Toad versión 2.2 [Toad()]. Neo4j tiene menos herramientas en ese sentido, aunque las herramientas que se utilizaron (Neo4j Browser y Neo4j Desktop versión 1.4.7) son cómodas y permiten realizar las consultas de forma correcta. En particular Neo4j Browser permite visualizar las consultas gráficamente, pudiendo observar los nodos, sus relaciones y propiedades de ambos.

IV-A3. Lenguajes de consulta: El lenguaje de consultas SQL es un lenguaje estándar y ampliamente conocido por los desarrolladores de hoy en día y es el lenguaje de acceso a datos en MySQL. Neo4j tiene como lenguaje de acceso Cypher, que aunque es un lenguaje pensado para bases de datos de grafos, tiene una estructura de consulta similar en algunos casos a SQL (como por ejemplo la claúsula WHERE para establecer condiciones en el filtro). Se observó una curva de aprendizaje rápida para alguien que ya ha trabajado con otros lenguajes de consulta a bases de datos, al menos para poder realizar consultas relativamente sencillas en la base de datos.

V. Conclusiones y trabajo a futuro

Luego de realizado el trabajo se llegan a las siguientes conclusiones. Las bases de datos relacionales permiten representar datos en forma de grafo. Puntualmente se observó esto con

Cuadro IV: Resultados de consultas de texto en milisegundos

Base	T1-R	T1-G	T2-R	T2-G
1000char	22	16	26	19
5000char	326	53	364	53
10000char	1168	98	983	100

MySQL y se pudo crear y consultar datos de forma efectiva, pero por otro lado se vió que son menos eficientes en algunas consultas, particularmente las que recorren la estructura de grafos. En ese escenario las bases de datos de grafos, en particular visto en Neo4j, debido a como representan esa estructura logran un mejor rendimiento.

Como trabajo a futuro se propone lo siguiente:

- Probar generar diferentes bases de datos, como por ejemplo generar una base de datos de tamaño mayor. Una idea inicial era probar con bases de datos de 100 mil o más nodos, pero no se logro debido al tiempo que insume crear dicha base y el poco tiempo disponible para la realización de este trabajo. También se puede probar con diferentes estructuras de grafos, modificando el tamaño de cada capa de nodos, la cantidad de conexiones entrantes de cada nodo, la forma de sortear esos valores (ya que solamente se sortearon valores con igual probabilidad entre 0 y un número máximo).
- Otra idea inicial del proyecto era hacer test de performance utilizando bases de datos ya generadas, que no se llegó a hacer debido al tiempo acotado para la realización del proyecto. Una línea de trabajo a futuro puede ser usar esas bases y resolver como importar esos datos tanto hacia la base de datos relacional como la de grafos.
- Incorporar algún framework para evaluación comparativa como por ejemplo Java Microbenchmark Harness [Java Microbenchmark Harness()] o JMeter [JMeter()], que permita por un lado hacer una medida más precisa del tiempo y que a su vez permita otro tipo de pruebas de carga, etc.

REFERENCIAS

[Almabdy(2018)] Soad Almabdy. Comparative analysis of relational and graph databases for social networks. In *2018 1st International Conference on Computer Applications Information Security (ICCAIS)*, pages 1–4, 2018. doi: 10.1109/CAIS.2018.8441982.

[Harrison(2015)] Guy Harrison. Next generation databases: NoSQL, NewSQL, and big data, 2015. URL https://www.amazon.com/Next-Generation-Databases-NoSQLand-Data/dp/1484213300.

[Java Microbenchmark Harness()] Java Microbenchmark Harness. Code Tools: jmh. https://openjdk.java.net/projects/code-tools/jmh/. Online; accessed 20 July 2021.

[JMeter()] JMeter. Apache JMeter. https://jmeter.apache.org/. Online; accessed 20 July 2021.

[Macak et al.(2020)Macak, Stovcik, and Buhnova] Martin Macak, Matus Stovcik, and Barbora Buhnova. The suitability of graph databases for big data analysis: A benchmark. 04 2020.

[MySQL()] MySQL. InnoDB and the ACID Model. https://dev.mysql.com/doc/refman/8.0/en/mysql-acid.html. Online; accessed 20 July 2021.

[Neo4j()] Neo4j. Neo4j Licensing. https://dev.mysqhttps://neo4j.com/licensing/. Online; accessed 20 July 2021.

[Toad()] Toad. Toad for MySQL. https://www.toadworld.com/downloads#mysql. Online; accessed 20 July 2021.

[Vicknair et al.(2010)Vicknair, Macias, Zhao, Nan, Chen, and Wilkins] Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, and Dawn Wilkins. A comparison of a graph database and a relational database: A data provenance perspective. volume 10, page 42, 01 2010. doi: 10.1145/1900008.1900067.