Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 5, Mes Mayo, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-195
http://publicaciones.uci.cu

Tipo de artículo: Artículo original
Temática: Soluciones informáticas
Recibido: 01/03/2020 | Aceptado: 25/04/2020 | Publicado: 01/05/2020

Comparación entre Bases de Datos Orientadas a Grafo y Relacional basado en el rendimiento

Comparison between Graph and Relational Databases based on performance

Royland Rodríguez Reyes[1]

[1] Universidad de las Ciencias Informáticas. Carretera de San Antonio de los Baños km 2 ½, Torrens, La Lisa, La Habana, Cuba. rrdguez@estudiantes.uci.cu

[*] Autor para correspondencia: rrdguez@estudiantes.uci.cu

Resumen

Actualmente en cada segundo y minuto en el mundo son generados grandes cantidades de datos que necesitan ser almacenados y procesados, esta información es muy variada en cuanto a la forma de almacenamiento y fuente de procedencia así como son muchos y variados los problemas que surgen a la hora de procesar esta información. Uno de estos problemas a solucionar son los de conexión o ruta a través de herramientas graficas como Neo4j, donde, utilizando el método comparativo en el presente trabajo, se realizara una investigación con el objetivo de demostrar las diferencias entre velocidad de ejecución de consultas y eficiencia entre las base de datos orientada a grafo (Neo4j) y las base de datos relacionales (PostgreSQL).

Palabras clave: grafo; Neo4j; consulta; SQL; Cypher

Abstract

Currently, in every second and minute in the world large amounts of data are generated that need to be stored and processed, this information is very varied in terms of the form of storage and source of origin as well as many and varied problems that arise at the time to process this information. One of these problems to be solved are those of connection or route through graphic tools such as Neo4j, where, using the comparative method in this work, an investigation will be carried out with the aim of demonstrating the differences between the speed of query execution and efficiency between the graph database (Neo4j) and relational databases (PostgreSQL).

Keywords: *graph; Neo4j; query; SQL; Cypher*

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 5, Mes Mayo, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-195
http://publicaciones.uci.cu

Introducción

El mundo de la tecnología avanza cada día de una manera indetenible, y en todos los aspectos de la vida el uso de las tecnologías de la información y las comunicaciones (TIC) es fundamental. Su desarrollo influye en gran cantidad de actividades diarias facilitado el acceso a todo tipo de información necesaria, provocando un aumento considerable de la interactividad entre personas de distintas continentes y países del mundo, brindando la posibilidad de desarrollar sus capacidades y habilidades.

La creciente conectividad que se genera entre un número cada vez mayor de dispositivos trae consigo un flujo de gran cantidad de datos que tienen que guardarse, procesarse y analizarse, lo que requiere ampliar y mejorar la forma de manejar y aprovechar estas oleadas de información que no dejarán de aumentar.

Para el procesamiento o análisis de estos conjuntos de datos de gran tamaño, complejidad y velocidad de crecimiento es necesario el uso de sistemas de gestión de base de datos (SGBD) que se ocupan de la gestión y administración de todo acceso a la base de datos, con el objetivo de servir de interfaz entre ésta, el usuario y las aplicaciones utilizadas.

La tipología de los SGBD es muy variada**,** se puede agrupar atendiendo al modelo de datos, número de usuarios o de sitios. Aunque esto suele ser lo más habitual, la tipología puede obedecer a otras muchas pautas, según convenga desde un determinado enfoque práctico.

Si atendemos al modelo de datos pueden ser:

- Relacionales
- EnRed
- Jerárquicos
- Orientados a objetos

En el presente trabajo se centra en las base de datos orientadas a grafo demostrando que es una herramienta poderosa de modelado de datos según su correcta utilización. Para ello se realizara un estudio comparándolas con las base de datos relacionales utilizando como SGBD a PostgreSQL y Neo4j.

Materiales y métodos

Método comparativo

El método comparativo de investigación es un procedimiento sistemático de contrastación de uno o más fenómenos, a través del cual se buscan establecer similitudes y diferencias entre ellos. El resultado debe ser conseguir datos que conduzcan a la definición de un problema o al mejoramiento de los conocimientos sobre este.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 5, Mes Mayo, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-195
http://publicaciones.uci.cu

Características del método comparativo:

- Es un método versátil, puede utilizarse como complemento de otros métodos.
- Puede formar la estructura completa de un proyecto de investigación.
- Al proceder el análisis, permite agregar aspectos nuevos e incluso retirar los aspectos vanos.
- No amerita registrar los aspectos que similares de los casos, ya que el estudio es una comparación de los mismos.
- En un estudio comparativo, se observan dos o más casos, objetos o eventos, en base del objeto de estudio, se deciden los aspectos, características o tributos interesantes a observar y registrar para cada caso.
- Se da por observación.
- El objetivo de una investigación comparativa, es revelar la estructura sistemática y la invariante para el grupo de donde provienen los casos estudiados. En otras palabras, el objetivo fundamental de dicho método es la generalización empírica y verificación de hipótesis, a fin de comprender eventos desconocidos a partir de otros conocidos.
- Permite y es muy efectivo en el estudio de muestras pequeñas.
- Algunos especialistas le consideran un tipo de estudio muy limitado, debido a que trabaja con factores de tiempo y espacio reducidos, implicado por el tamaño de la muestra.

(Conócelo todo sobre el método comparativo, 2019)

Etapas:

1. Detección del problema: identificar cual es el problema o interrogante en cuestión que sería como determinar que las bases de datos orientadas a grafo son más rápidas y eficientes que las base de datos relacionales.
2. Elaboración del armazón teórico: consiste en la búsqueda y revisión de trabajos y estudios anteriores hechos sobre el objeto de esta investigación que permite formular la hipótesis de que las base de datos orientada a grafo son más sofisticadas y rápidas que las base de datos relacionales.
3. Elección del objeto de estudio: comparación entre bases de datos a través de las consultas.
4. Definición de criterios para escoger la muestra: el caso de estudio que se emplea es un proyecto ejemplo que se encuentra en Neo4j llamado Movie Graph el cual contiene actores y directores relacionados a través de las películas en las que colaboran.
5. Análisis de las muestras: se procede a realizar la comparación entre las variables escogidas, las muestras son examinadas, clasificadas y evaluadas y se verificará si la hipótesis planteada es pertinente y demostrable.

6. Explicación de los fenómenos estudiados: constituye la etapa final donde por medio de la explicación es posible vincular los resultados obtenidos con otros hechos conocidos.

El método comparativo permite observar semejanzas y diferencias entre varios objetos de estudios por lo que es muy utilizado en investigaciones científicas y por expertos en la realización de experimentos y posteriormente realizar predicciones desarrollando muchas teorías. En la vida cotidiana usamos la comparación en muchas situaciones, sin embargo, este método es aplicado a niveles científicos e investigativos al ser un procedimiento ordenado que permite la observación y la evaluación de resultados.

Herramienta Neo4j

Desarrollado por Neo Technology es un software libre de base de datos orientada a grafos, implementado en Java es la base de datos de gráficos líder en el mundo. Es un almacén de gráficos de alto rendimiento con todas las características que se esperan de una base de datos sólida y madura, como un lenguaje de consulta amigable y transacciones ACID. El programador trabaja con una estructura de red flexible de nodos y relaciones en lugar de tablas estáticas, pero disfruta de todos los beneficios de una base de datos de calidad empresarial. Para muchas aplicaciones, Neo4j ofrece muchos beneficios de rendimiento en comparación con las bases de datos relacionales.

Características:

- Licencia dual: código abierto y comercial.
- Un modelo intuitivo orientado a gráficos para la representación de datos. En lugar de tablas, filas y columnas estáticas y rígidas, trabaja con una red de gráficos flexible que consta de nodos, relaciones y propiedades.
- Administrador de almacenamiento nativo basado en disco completamente optimizado para almacenar estructuras gráficas para un máximo rendimiento y escalabilidad. Listo para SSD.
- Potente marco transversal para recorridos de alta velocidad en el espacio del nodo.
- La implementación actual está diseñada para manejar gráficos grandes que no caben en la memoria con durabilidad. No es un caché, es un almacén transaccional completamente persistente.
- Separa los datos y la lógica con una representación más "natural" que las tablas.
- Neo4j atraviesa profundidades de 1000 niveles y más a una velocidad de milisegundos. Eso son muchos órdenes de magnitud más rápido que los sistemas relacionales.

Las bases de datos orientados a grafo poseen una estructura de datos conectadas (red social, recomendaciones, GPS, atención médica, entre otros) las cuales más fáciles de almacenar y consultar en Neo4j que en una base de datos relacional al no poder lidiar con problemas de gráficos. El rendimiento es un factor influyente en las base de datos relacionales, con pocos datos en un principio puede ser eficaz pero a medida que aumente el volumen de datos las

uniones entre estos son más débiles y el tiempo de consulta es cada vez más largo al no poder resolver operaciones de ruta. Neo4j no tiene tal limitación ya que los tiempos de consulta aumentan conjuntamente con la cantidad de datos a explorar como parte de su consulta y no con el tamaño general del conjunto de datos, que puede tener grandes volúmenes. En problemas de unión Neo4j es una solución superior a una base de datos relacional con modelo de datos complejos, el modelo de datos de Neo4j es mucho más expresivo y enfatiza la conectividad como concepto principal ya que es la respuesta a problemas de este tipo de análisis.

Resultados y discusión

Para el desarrollo de la propuesta de solución se utilizó como sistema gestor de base de datos a Neo4j 4.1.3 y PostgreSQL 9.4.1, como navegador Mozilla Firefox 81.0, XAMMP 3.2.2 como servidor de la herramienta generatedata 3.4.1 usada para generar datos aleatorios en SQL y para el diseño de la base de datos a Visual Paradigm Suite Windows 5.0 una herramienta de modelado muy potente que permite la generación de cogido entre sus muchas utilidades.

El caso de estudio empleado como antes se describe es una base de datos proporcionada por neo4j llamada Movie Graph la cual contiene la relación entre actores y directores a través de las películas en las que participaron; de esta solo se utilizara el diseño de la base de datos ya que para realizar la investigación es necesario tener grandes volúmenes de datos para que los resultados sean factibles.

La herramienta generatedata permite generar de forma rápida y sencilla grandes volúmenes de datos de forma personalizada de acuerdo con el diseño de la base de datos en cualquier formato, para poder usarla es necesario montar el servidor en XAMMP. En la página de generatedata se genera 8000 tuplas de la tabla person, 2000 tuplas de la tabla movie y 20000 tuplas de la tabla person_movie que contiene las relaciones entre las tablas o aristas del grafo, quedando en total 10000 nodos y 20000 aristas del grafo, las propiedades y nombres se personalizaran de acuerdo al diseño de la base de datos. Cada uno de los datos se exportará en formato CSV.

El diseño de la base de datos en SQL se realizara en la herramienta Visual Paradigm que permite generar el código a partir del diseño realizado, siendo de gran utilidad y comodidad al poder tener una perspectiva más general del modelado que se está realizando entre otras posibilidades, claro está que también se puede realizar directamente en PostgreSQL a través de código.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 5, Mes Mayo, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-195
http://publicaciones.uci.cu

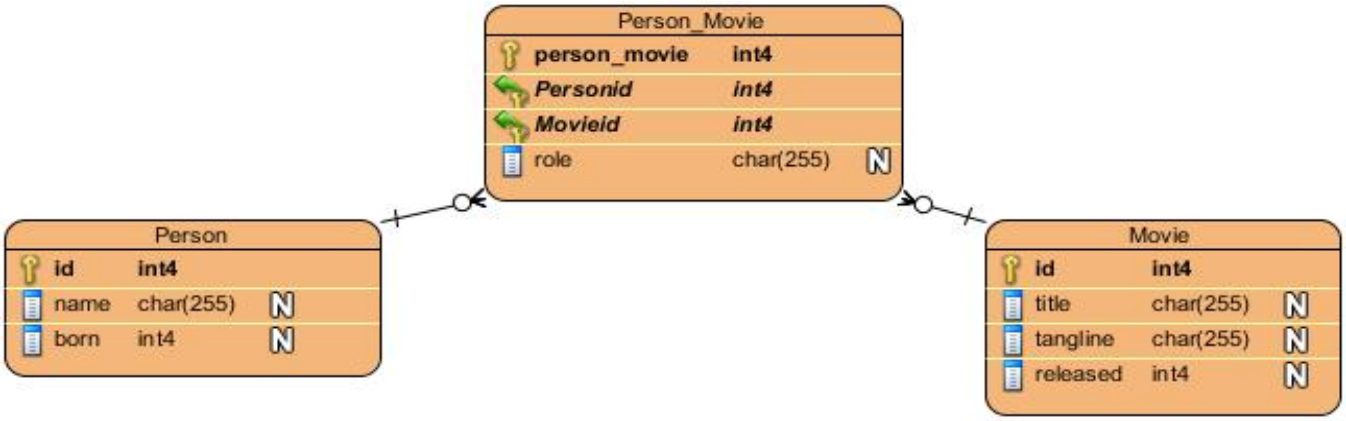

Figura 1: Diseño de la base de datos en Visual Paradigm.

Código de la base de datos generado por Visual Paradigm a partir del diseño anterior:

CREATE TABLE Person_Movie (person_movie int4 NOT NULL, Personid int4 NOT NULL, Movieid int4 NOT NULL, role char(255), PRIMARY KEY (person_movie, Personid, Movieid));

CREATE TABLE Movie (id int4 NOT NULL, title char(255), tangline char(255), released int4, PRIMARY KEY (id));

CREATE TABLE Person (id int4 NOT NULL, name char(255), born int4, PRIMARY KEY (id));

ALTER TABLE Person_Movie ADD CONSTRAINT FKPerson_Mov518391 FOREIGN KEY (Movieid) REFERENCES Movie (id) ON UPDATE Cascade;

ALTER TABLE Person_Movie ADD CONSTRAINT FKPerson_Mov473029 FOREIGN KEY (Personid) REFERENCES Person (id) ON UPDATE Cascade;

El código anterior se ejecuta en PostgreSQL para modelar la base de datos y luego poder realizar la importación empleando el comando COPY en cada uno de las tablas de la siguiente forma:

COPY person (id, name, born)

FROM 'E:\Documentos\Publicaciones\person_db.csv'

DELIMITER ','

CSV HEADER;

COPY movie (id, title, tangline, released)

FROM 'E:\Documentos\Publicaciones\movie_db.csv'

```
DELIMITER ','
CSV HEADER;

COPY person_movie (person_movie, personid, movieid, role)
FROM 'E:\Documentos\Publicaciones\person_db.csv'
DELIMITER ','
CSV HEADER;
```

Posteriormente se procede a importar los archivos CSV a la base de datos de Neo4j para ello usamos el comando LOAD CSV para obtener los datos en la consulta y escribirlos en la base de datos usando las cláusulas de consulta siguientes:

```
LOAD CSV WITH HEADERS FROM 'file:///person_db.csv' AS line
CREATE (:Person{id: toInteger(line.id), name:line.name,
born: toInteger(line.born)})

LOAD CSV WITH HEADERS FROM 'file:///movie_db.csv' AS line
CREATE (:Movie{id: toInteger(line.id), title:line.title,
tangline:line.tagline, released: toInteger(line.released)})
```

En el caso de movie_graph como contiene las aristas del grafo para crear las relaciones con los nodos correspondientes el archivo se dividió según el tipo de relación.

```
LOAD CSV WITH HEADERS FROM 'file:///actor_db.csv' AS line
CREATE (:Actor{personid: toInteger(line.personid),
movieid: toInteger(line.movieid)})

LOAD CSV WITH HEADERS FROM 'file:///actor_db.csv' AS line
CREATE (:Directed{personid: toInteger(line.personid),
movieid: toInteger(line.movieid)})
```

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 5, Mes Mayo, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-195
http://publicaciones.uci.cu

```
LOAD CSV WITH HEADERS FROM 'file:///produced_db.csv' AS line
CREATE (:Produced{personid: toInteger(line.personid),
movieid: toInteger(line.movieid)})

LOAD CSV WITH HEADERS FROM 'file:///wrote_db.csv' AS line
CREATE (:Wrote{personid: toInteger(line.personid),
movieid: toInteger(line.movieid)})
```

Neo4j utiliza Cypher como lenguaje de consulta que es similar en muchos sentidos al SQL, excepto que SQL se refiere a elementos almacenados en una tabla, mientras que Cypher se refiere a elementos almacenados en un grafo.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 5, Mes Mayo, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-195
http://publicaciones.uci.cu

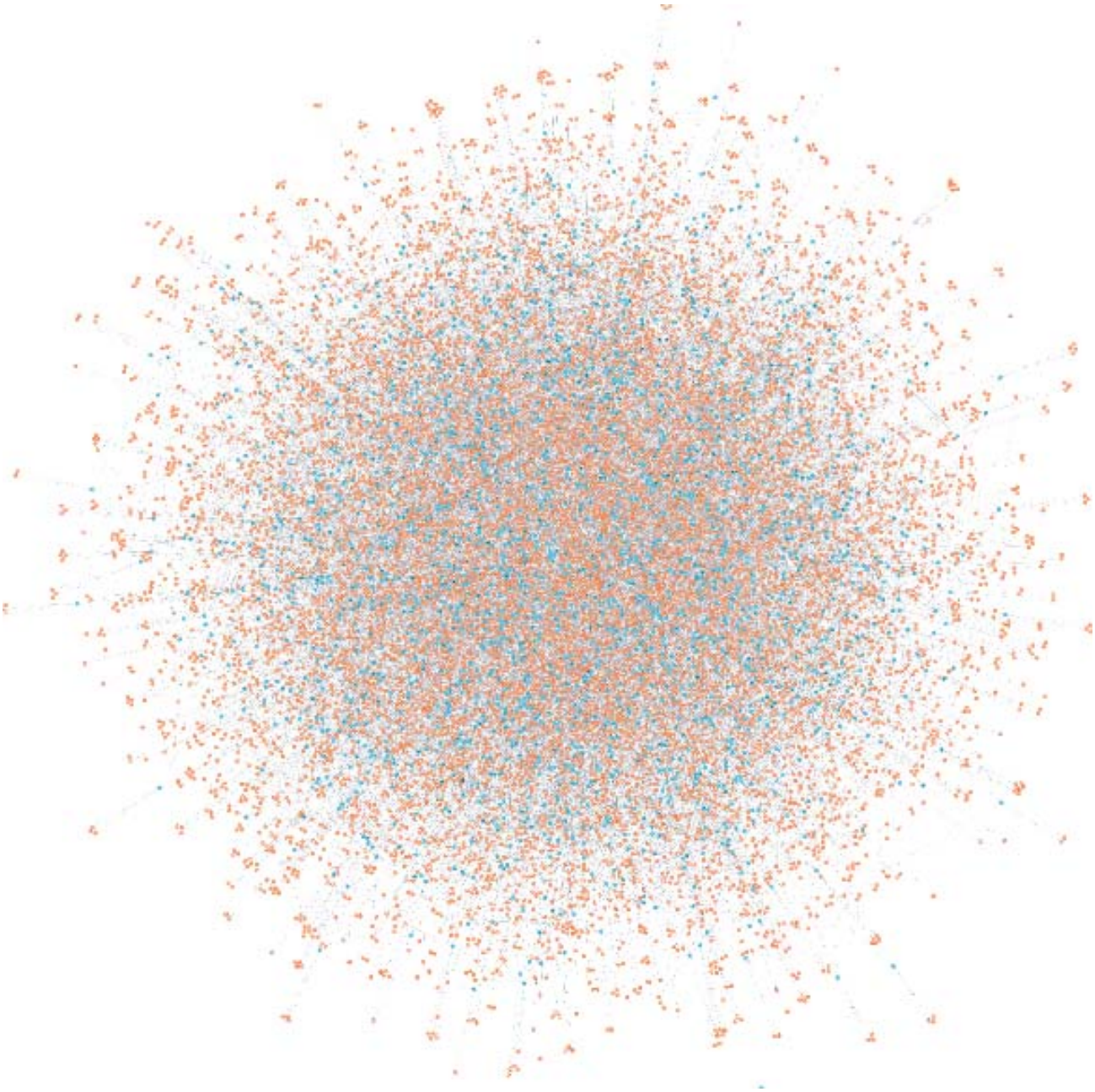

Figura 2: Base de Datos en Neo4j, esta es una captura de la foto original ya que tiene un tamaño de 105mb.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 5, Mes Mayo, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-195
http://publicaciones.uci.cu

A continuación se procede a realizar varias consultas en ambas base de datos para el posterior análisis de los resultados, primeramente se describe la consulta a realizar seguido por su sintaxis en ambos lenguajes y el tiempo de ejecución. Para mostrar el tiempo de ejecución de las consultas en SQL se utiliza el comando EXPLAIN ANALYSE.

Consulta 1: El nombre de todos los actores y las películas en las que participan.

Cypher:

Match (p:Person)-[:ACTED_IN]->(movie) return p.name

Started streaming 4997 records in less than 1 ms and completed after 4 ms.

SQL:

SELECT person.name FROM person

JOIN person_movie ON person_movie.personid = person.

JOIN movie ON person_movie.movieid = movie.id

WHERE person_movie.role='Actor'

"Planning time: 0.395 ms"

"Execution time: 21.427 ms"

Consulta 2: El nombre de todos los directores y el titulo las películas en las que participan.

Cypher:

Match (p:Person)-[:DIRECTED]->(movie) return p.name, m.title

Started streaming 4957 records in less than 1 ms and completed after 5 ms.

SQL:

SELECT person.name, movie.title FROM person

JOIN person_movie ON person_movie.personid = person.id

JOIN movie ON person_movie.movieid = movie.id

WHERE person_movie.role='Directed'

"Planning time: 0.342 ms"

"Execution time: 24.187 ms"

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 5, Mes Mayo, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-195
http://publicaciones.uci.cu

Consulta 3: El nombre de todos los productores y las películas en las que participan.

Cypher:

Match (person)-[:PRODUCED]->() return person.name

Started streaming 5034 records in less than 1 ms and completed after 1 ms.

SQL:

SELECT person.name FROM person

JOIN person_movie ON person_movie.personid = person.id

JOIN movie ON person_movie.movieid = movie.id

WHERE person_movie.role='Produced'

"Planning time: 0.301 ms"

"Execution time: 22.551 ms"

Consulta 4: El año de nacimiento y el nombre de todos los escritores y el año de estreno de las películas en las que participan.

Cypher:

Match (p:Person)-[:WROTE]->(m:Movie) return p.name,p.born,m.released

Started streaming 5012 records in less than 1 ms and completed after 1 ms.

SQL:

SELECT person.name, person.born, movie.released FROM person

JOIN person_movie ON person_movie.personid = person.id

JOIN movie ON person_movie.movieid = movie.id

WHERE person_movie.role='Wrote'

"Planning time: 0.313 ms"

"Execution time: 21.415 ms"

Se puede observar que las consultas en Cypher se expresan con gran facilidad y permite concentrarse en su dominio en lugar de perderse en el acceso a la base de datos como es el caso de SQL, sin embargo sus estructuras son similares al utilizar clausulas encadenadas entre sí.

Consulta 5: El nombre de todas las personas de la base de datos.

Cypher:

Match (p:Person) return p.name

Started streaming 10000 records in less than 1 ms and completed after 2 ms.

SQL:

SELECT person.name FROM person

"Planning time: 0.052 ms"

"Execution time: 1.737 ms"

Consulta 6: El título de todas las películas de la base de datos.

Cypher:

Match (m:Movie) return m.title

Started streaming 2000 records in less than 1 ms and completed after 3 ms

SQL:

SELECT movie.title FROM movie

"Planning time: 0.046 ms"

"Execution time: 0.395 ms"

Consulta 7: El título de todas las películas que se estrenaron después del 2000.

Cypher:

Match (m:Movie) where m.released>2000 return m.title

Started streaming 1281 records in less than 1 ms and completed after 2 ms.

SQL:

SELECT movie.title FROM movie where released>2000

"Planning time: 0.057 ms"

"Execution time: 0.505 ms"

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 5, Mes Mayo, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-195
http://publicaciones.uci.cu

Se observa que en la realización de consultas simples sin relación entre sus elementos neo4j muestra una velocidad de consulta inferior a PostgreSQL ya que el mejor uso de este gestor es para la solución de problemas de conectividad y puede apreciarse que las base de dato relacional sobresale en esta consulta al tener los elementos organizados en una tabla, pero esto es solo en casos de consultas gran simplicidad y sin relación entre sus datos lo que no es muy utilizado ya que por lo general las consultas realizadas a las bases de datos son de gran complejidad, sin embargo en estas consulta a pesar del resultado, la diferencia del tiempo de ejecución de ambas base de datos no es muy distante.
Además de la velocidad de consulta neo4j ofrece herramientas de visualización gráfica que muestra el resultado de una consulta a través de un grafo completo permitiendo analizar el resultado desde diferentes perspectivas a través de las conexiones fácilmente visibles en un formato visual gráfico como se muestran en las siguientes consultas:

Consulta 8: Todas las películas en las que actuó Ori Cash.

Cypher:

Match p=(:Person{name:'Ori Cash'})-[:ACTED_IN]->() return p

Started streaming 3 records in less than 1 ms and completed in less than 1 ms.

SQL:

SELECT person.*,movie.*, person_movie.* FROM person
JOIN person_movie ON person_movie.personid=person.id
JOIN movie ON person_movie.movieid=movie.id
WHERE person.name= 'Ori Cash' and person_movie.role='Actor'
"Planning time: 0.345 ms"
"Execution time: 4.410 ms"

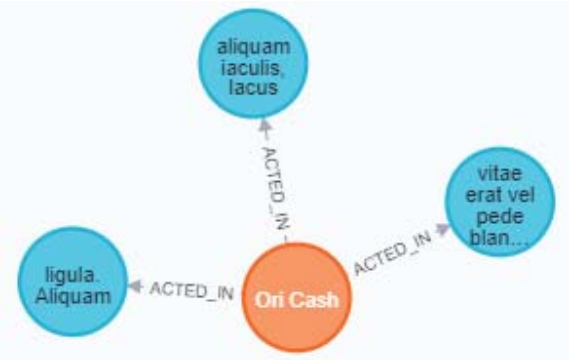

Figura 3: Grafo resultante de la ejecución de la Consulta 8, cabe recordar que

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 5, Mes Mayo, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-195
http://publicaciones.uci.cu

el nombre de las película fue generado de forma aleatoria por el generatedata.

	id integer	name character(255)	born integer	id integer	title character(255)	tangline character(255)	released integer	person intege	personid integer	movieid integer	role charac
1	34	Ori Cash	1952	995	vitae erat vel p	nec	1995	10389	34	995	Actor
2	34	Ori Cash	1952	1089	aliquam iaculis,	et netus et	2012	11726	34	1089	Actor
3	34	Ori Cash	1952	1110	ligula. Aliquam	tellus. Susper	2014	16687	34	1110	Actor

Figura 4: Tabla resultante de la ejecución de la Consula 8.

Consulta 9: Las películas que tiene en común Chanda West y Brianna Peters así como el rol desempeñado.

Cypher:

Match p=(:Person{name:'Chanda West'})-[]->()<-[]-(:Person{name:'Brianna Peters'}) return p

Started streaming 1 records in less than 1 ms and completed in less than 1 ms.

SQL:

```
CREATE OR REPLACE FUNCTION query9 (name1 varchar, name2 varchar)
RETURNS SETOF record AS
$body$
DECLARE
id1 integer;
id2 integer;
BEGIN
for id1 in (SELECT movie.id from movie
        JOIN person_movie ON person_movie.movieid=movie.id
        JOIN person ON person_movie.personid=person.id
        where person.name = $1)
loop
   for id2 in (SELECT movie.id from movie
        JOIN person_movie ON person_movie.movieid=movie.id
        JOIN person ON person_movie.personid=person.id
        where person.name = $2)
   loop
      if id1 = id2 then
```

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 5, Mes Mayo, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-195
http://publicaciones.uci.cu

```
        return query
        SELECT movie.id, movie.title, movie.released, movie.tangline from movie
        where movie.id=id1
        UNION
        SELECT person.id, person.name, person.born, person_movie.role from person
        JOIN person_movie ON person_movie.personid=person.id
        where person.name=$1 and person_movie.movieid=id1
        UNION
        SELECT person.id, person.name, person.born, person_movie.role from person
        JOIN person_movie ON person_movie.personid=person.id
        where person.name=$2 and person_movie.movieid=id1;
      end if;
   end loop;
end loop;
return;
END
$body$
LANGUAGE 'plpgsql'
```

Select * from query9 ('Chanda West','Brianna Peters') as (movieid int, title char(255), released int, tangline char(255))

"Planning time: 0.021 ms"

"Execution time: 3.347 ms"

Figura 5: Grafo resultante de la ejecución de la Consulta 9.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 5, Mes Mayo, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-195
http://publicaciones.uci.cu

	movieid integer	title character(255)	released integer	tangline character(255)
1	833	sed	2001	eu
2	2462	Chanda West	1937	Actor
3	5466	Brianna Peters	1980	Directed

Figura 6: Tabla resultante de la ejecución de la Consulta 9.

Seguido del tiempo de ejecución de las consultas se muestran imágenes del resultado de ambas bases de datos para observar las diferencias en cuanto a la presentación de los resultados, donde las relacionales muestran una tabla de elementos organizados sin una clara relación sin embargo las orientadas a grafo muestran un diseño claro y más visible de las relaciones existentes entre los nodos involucrados en la consulta facilitando su comprensión.

Consulta 10: Todas las personas que participaron en las películas arcu. Aliquam ultrices iaculis y neque, así como el rol desempeñado.

Cypher:

```
Match p=(:Movie{title:'arcu. Aliquam ultrices iaculis'})<-[]-(:Person)-[]->(:Movie{title:'neque'}) return p
```

Started streaming 2 records in less than 1 ms and completed in less than 1 ms.

SQL:

```
CREATE OR REPLACE FUNCTION query10 (title1 varchar, title2 varchar)
RETURNS SETOF record AS
$body$
DECLARE
id1 integer;
id2 integer;
BEGIN
for id1 in (SELECT person.id from person
        JOIN person_movie ON person_movie.personid=person.id
        JOIN movie ON person_movie.movieid=movie.id
        where movie.title = $1)
loop
```

```
    for id2 in (SELECT person.id from person
            JOIN person_movie ON person_movie.personid=person.id
            JOIN movie ON person_movie.movieid=movie.id
            where movie.title = $2)
    loop
      if id1 = id2 then
        return query
        SELECT person.id, person.name, person.born from person
        JOIN person_movie ON person_movie.personid=person.id
        where person.id=id1
        UNION
        SELECT movie.id, person_movie.role, movie.released from movie
        JOIN person_movie ON person_movie.movieid=movie.id
        where movie.title=$1 and person_movie.personid=id1
        UNION
        SELECT movie.id, person_movie.role, movie.released from movie
        JOIN person_movie ON person_movie.movieid=movie.id
        where movie.title=$2 and person_movie.personid=id2;
      end if;
    end loop;
end loop;
return;
END
$body$
LANGUAGE 'plpgsql'

Select * from query10 ('arcu. Aliquam ultrices iaculis','neque') as (id int, role char(255), born int)
"Planning time: 0.032 ms"
"Execution time: 37.377 ms"
```

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 5, Mes Mayo, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-195
http://publicaciones.uci.cu

Figura 7: Grafo resultante de la ejecución dela Consulta 10.

	id integer	role character(255)	born integer
1	6207	Shelley Collins	1962
2	329	Produced	2018
3	316	Directed	2019
4	329	Directed	2018
5	8395	Fallon Rose	1953
6	316	Directed	2019

Figura 8: Tabla resultante de la ejecución de la Consulta 10.

Consulta 11: Todas las películas dirigidas por los directores que trabajaron con el actor Alec Leonard.

Cypher:

Match p=(:Person{name:'Alec Leonard'})-[:ACTED_IN]->(:Movie)<-[:DIRECTED]-(a:Person)-[:DIRECTED]->(:Movie) return p

Started streaming 6 records in less than 1 ms and completed in less than 1 ms.

SQL:

```
CREATE OR REPLACE FUNCTION query11 (name varchar)
RETURNS SETOF record AS
$body$
DECLARE
id1 integer;
```

```
id2 integer;
cont integer;
BEGIN
for id1 in (SELECT movie.id from movie
        JOIN person_movie ON person_movie.movieid=movie.id
        JOIN person ON person_movie.personid=person.id
        where person.name = $1 and person_movie.role = 'Actor')
loop
   if((SELECT count(*) FROM person
      JOIN person_movie ON person_movie.personid=person.id
      JOIN movie ON person_movie.movieid=movie.id
      WHERE movie.id = id1 AND person_movie.role = 'Directed') > 1)
      then
      for id2 in (SELECT person.id FROM person
            JOIN person_movie ON person_movie.personid=person.id
            JOIN movie ON person_movie.movieid=movie.id
            WHERE movie.id = id1 AND person_movie.role = 'Directed')
      loop
         return query
         Select movie.id, movie.title, movie.released, person.id,
         person.name, person.born, person_movie.role from movie
         JOIN person_movie ON person_movie.movieid=movie.id
         JOIN person ON person_movie.personid=person.id
         where person.id = id2 and person_movie.role = 'Directed' and movie.id != id1;
      end loop;
   end if;
end loop;
return;
END
$body$
```

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 13, No. 5, Mes Mayo, 2020
ISSN: 2306-2495 | RNPS: 2343
Pág. 174-195
http://publicaciones.uci.cu

LANGUAGE 'plpgsql'

Select * from query11('Alec Leonard') as (movieid int, title char(255), released int,

personid int, name char(255), born int, role char(255))

"Planning time: 0.022 ms"

"Execution time: 5.838 ms"

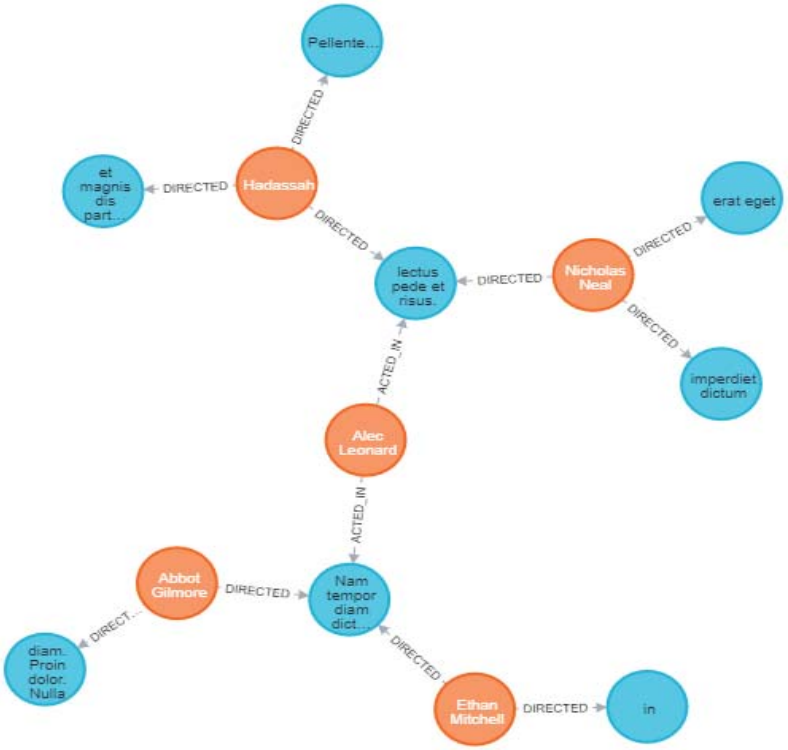

Figura 9: Grafo resultante de la ejecución de la Consulta 11.

	movieid integer	title character(255)	released integer	personid integer	name character(255)	born integer	role character(255)
1	1430	in	2020	6108	Ethan Mitchell	1967	Directed
2	212	diam. Proin dolor. Nulla	2005	7735	Abbot Gilmore	1983	Directed
3	119	imperdiet dictum magna.	2004	8023	Nicholas Neal	1988	Directed
4	630	erat eget	2008	8023	Nicholas Neal	1988	Directed
5	1778	Pellentesque habitant morbi tristique senectus	2013	732	Hadassah Hahn	1954	Directed
6	1267	et magnis dis parturient montes,	2006	732	Hadassah Hahn	1954	Directed

Figura 10: Tabla resultante de la ejecución de la Consulta 11.

Como se observa Cypher permite expresar consultas complejas a la base de datos con gran facilidad obteniendo un rendimiento excelente que son visualizadas en un grafo dando una perspectiva única para la comprensión del resultado; sin embrago, expresar estas consultas en SQL no resulta sencillo ya que no es adecuado para dar solución a problemas de conexión o ruta. El rendimiento de la base de dato relacional es otro factor también se ve afectado en este tipo de consultas, con pequeños datos puede ser eficaz pero a medida que aumente el volumen de datos las uniones entre tablas disminuyen provocando que los tiempos de consulta se hagan cada vez mas largos. Neo4j no tiene este problema en los problemas de ruta ya que solo explora los datos que forman parte de su consulta independientemente del volumen de datos que puede ser enorme. Es notable que en la solución de problemas de conexión Neo4j es una opción superior a las bases de datos relacionales, tanto por el tiempo como por la forma de expresar las consultas, y así se refleja a continuación en la siguiente tabla:

Consulta	SQL	Cypher
1	21.427 ms	after 4 ms
2	24.187 ms	after 5 ms
3	22.551 ms	after 1 ms
4	21.415 ms	after 1 ms
8	4.410 ms	after 1 ms
9	3.347 ms	less than 1 ms
10	37.377 ms	less than 1 ms
11	5.838 ms	less than 1 ms

Tabla 1: Resultados de las consultas en ambas base de datos.

Conclusiones

Al concluir con la investigación según los resultados obtenidos podemos confirmar que las bases de datos orientados a grafo son una mejor solución que las base de datos relacionales a problemas de conexión o ruta demostrando entonces la veracidad de la hipótesis planteada. Son muchas las aplicaciones que tiene esta herramienta que propone gran comodidad con un potente modelado, muy expresivo, que enfatiza la conectividad propiciando una excelente velocidad de consultas al recorrer los caminos más cortos, según la expresión de su consulta, independientemente del volumen de datos existentes, a diferencia con las base de datos relacionales que sufren mucho al dar solución a este tipo de problemas debido a las restricciones de su modelo.

Referencias

Group, T. P. (2018). *PostgreSQL 9.4.18 Documentation.* University of California.

Ian Robinson, J. W. (2015). *Graph Databases Second Edition.* Sebastopol, CA: O'Reilly Media, Inc.

Momjian, B. (2000). *PostgreSQL Introduction and Concepts.* Upper Saddle River, NJ: Addison–Wesley.

Neo4j, I. (2020). *The Neo4j Cypher Manual v4.2.* from https://neo4j.com/docs/

Neo4j, I. (2020). *The Neo4j Operations Manual v4.2.* from https://neo4j.com/docs/

Pacheco, J. (Oct 15, 2019). Método Comparativo (definición, usos, características). *webyempresas*.

Penchikala, S. (2014). Data Modeling in Graph Databases: Interview with Jim Webber and Ian Robinson. *infoq*.

PostgreSQL, E. e. (2000). *Guia del Programador de PostgreSQL*. Universidad de California, Berkeley: Thomas Lockhart.