



Universidad Veracruzana



Universidad Veracruzana
Facultad de Estadística e Informática

Bases de datos orientadas a grafos

PROYECTO FINAL

Saúl Barragán Torres

Experiencia Educativa:

Bases de datos no convencionales

Docente:

Lorena Alonso Ramírez

Fecha de entrega:

27 de mayo de 2025

Contenido

Introducción	3
Bases de datos orientadas a grafos.....	4
Características	4
Funciones de Cypher	6
Bibliografía	12
Ilustración 1 Relación entre nodos y aristas en una base de datos	4
Ilustración 2 Base de datos completa	5
Ilustración 3 Creación de un nodo tipo Película.....	6
Ilustración 4 Script completo para crear los nodos de una película	6
Ilustración 5 Crear los nodos y la relación	6
Ilustración 6 Se obtienen los nodos y luego se crea la relación.....	7
Ilustración 7 Modificación de los datos de un nodo	7
Ilustración 8 Uso de REMOVE	7
Ilustración 9 Uso de DELETE y DETACH	7
Ilustración 10 Resultado de una consulta simple.....	8
Ilustración 11 Ejemplo e WHERE donde se buscan películas cuyo año de estreno sea superior a 2000.....	9
Ilustración 12 Ejemplo donde se utiliza STARTS WITH para buscar actores con cierto nombre	10
Ilustración 13 Uso del COUNT y ORDER BY	11

Introducción

Cada vez más, las bases de datos en los sistemas actuales deben almacenar más información y con mayor complejidad. Para los Sistemas Manejadores de Bases de Datos (SMBD) Relacionales se vuelve más complicado el tener que realizar consultas o actualizaciones en datos con muchas relaciones, donde la búsqueda de un valor conlleva en la navegación a través de varias tablas afectando al rendimiento del sistema. Además de eso, en un entorno donde los datos cambian constantemente la rigidez de las bases de datos relacionales dificultan la agregación y actualización de datos. Cuando tenemos un contexto donde la estructura de los datos no cambiará entonces podemos encontrar útiles estos SMBD, por ejemplo en el manejo de Usuarios, donde los campos de una clase ya están definidos y no pueden cambiar.

Para contextos más dinámicos existen las bases de datos no relacionales donde el SMBD gestiona valores que no tienen una estructura rígida. Si en las bases de datos relacionales hablamos de tablas, para las bases de datos no relacionales encontramos colecciones, documentos o grafos. Las relaciones entre pares de datos se vuelven más rápida y un SMBD puede hallar un valor sin la necesidad de tener que pasar forzosamente por un conjunto de tablas. Además, se habla de que en estos contextos los datos pueden ser dinámicos sin seguir una estructura rígida, esto se refiere a que una base de datos puede almacenar un conjunto de documentos que no contengan el mismo número de campos y cuyos datos puedan ser de diferente tipo, he aquí la ventaja de estas bases de datos.

Este documento se centra en explicar en qué consisten las bases de datos orientadas a grafos.

Bases de datos orientadas a grafos

Una Base de Datos Orientada a Grados (BDOG) se compone de nodos y aristas tal cual como un grafo, donde cada nodo es una entidad y las aristas son las relaciones de esta entidad. Véase la entidad como la tabla, cada entidad puede tener atributos que la definan. Así mismo, las relaciones son marcadas con etiquetas que le dan sentido a la relación. Explicaré esto a continuación:

Características

Grafo

Un nodo es un tipo de entidad que puede tener uno o más etiquetas (por ejemplo: Persona y Cliente), a su vez, cada nodo puede tener una serie de propiedades que lo definan (nombre, fecha_nacimiento, correo, teléfono, etc.).

Una arista es la relación que conecta a dos nodos, puede ser bidireccional o unidireccional. Una relación sólo puede tener un tipo de etiqueta (por ejemplo, un nodo Cliente se relaciona con Producto a través de la arista Compra).

El grafo se compone de las relaciones entre diferentes nodos, un nodo Película puede estar relacionado con el nodo Director a través de DIRIGE, con uno o más nodos Actor a través de ACTUA, con un nodo Productora a través de PRODUCE y con un nodo Año a través de SE_ESTRENA como se ve en la Imagen 1, con una base de datos más grande podemos ver una red de nodos como se ve en la Imagen 2.



Ilustración 1 Relación entre nodos y aristas en una base de datos



Ilustración 2 Base de datos completa

Lenguaje Cypher

Es un lenguaje declarativo inspirado en SQL orientado a grafos desarrollado por Neo4j cuyo propósito es expresar consultas de forma intuitiva, similar a cómo SQL funciona en bases de datos relacionales. Más adelante se mostrarán funciones en este lenguaje.

Neo4j

Neo4j es el SMDb para BDOG más utilizado. Desarrollado por Neo Technologies es un sistema de gestión de bases de datos orientado a grafos, diseñado específicamente para almacenar, consultar y analizar datos que están altamente interconectados.

Funciones de Cypher

Utilizaremos el ejemplo de la base de datos de películas para ilustrar los comandos a continuación:

- **Crear nodos**

El comando que utilizamos para crear cualquier cosa, ya sea nodo o arista, es CREATE. Para crear un nodo se tiene que poner dentro de un paréntesis, primero indicando la etiqueta, la cual va siempre después de dos puntos ':'. Después si queremos agregar algún parámetros se deben de colocar dentro de unas llaves seguido del formato *clave:valor*.

```
CREATE (:Película {nombre: 'Interestelar'})
```

Ilustración 3 Creación de un nodo tipo Película

A continuación, se ve un ejemplo completo para crear todos los nodos necesarios para la película Interestelar (cabe aclarar que no es necesario crear todos los nodos pues no existe una estructura fija, esa es la gracia de las bases de datos no convencionales):

```
1 //Creamos el nodo de la película
2 CREATE (:Película {nombre: 'Interestelar'})
3 //Nodo del director
4 CREATE (:Director {nombre: 'Christopher Nolan'})
5 //Nodo del protagonista
6 CREATE (:Actor {nombre: 'Matthew McConaughey'})
7 //Nodo de la casa productora
8 CREATE (:Productora {nombre: 'Paramount Pictures'})
9 //Nodo del año en que salió
10 CREATE (:Año {valor: 2014})
```

Ilustración 4 Script completo para crear los nodos de una película

- **Crear relaciones**

Para crear relaciones se utiliza la misma estructura, sin embargo, para indicar que vamos a crear una relación necesitamos el nodo emisor, luego la relación indicada entre un par de llaves cuadradas y seguida del nodo receptor.

Hay dos formas de crear una relación, creando en el proceso todos los nodos y la relación o simplemente obteniéndolos y guardándolos.

```
neo4j$ CREATE (:Director {nombre: 'Guillermo del Toro'})-[:DIRIGE]->(:Película {nombre: 'La cumbre escarlata'})
```

Ilustración 5 Crear los nodos y la relación

```

1 MATCH (d:Director {nombre: 'Christopher Nolan'})
2 MATCH (p:Pelicula {nombre: 'Interestelar'})
3 CREATE (d)-[:DIRIGE]->(p)

```

Ilustración 6 Se obtienen los nodos y luego se crea la relación

En este último ejemplo se hace uso del método MATCH el cuál es comparable a SELECT en SQL, hablaremos de él más adelante. Nótese cómo se guardan las entidades en variables que se colocan antes de escribir las etiquetas, esto también funciona para relaciones.

- **Modificar información**

Para modificar la información, ya sea de una propiedad o para agregar otra etiqueta se utiliza el comando SET.

```

1 MATCH (guillermo:Director {nombre: 'Guillermo del Toro'})
2 //Le cambiamos el nombre
3 SET guillermo.nombre = 'Guillermo del Toro Gómez'
4 //Le agregamos un atributo
5 SET guillermo.nacionalidad = "Mexicana"
6 //Le agregamos otra etiqueta
7 SET guillermo:Persona

```

Ilustración 7 Modificación de los datos de un nodo

Esto también se podría realizar a una relación, pero esta sólo puede tener una etiqueta.

- **Borrar información**

Para borrar información tenemos dos métodos: REMOVE y DELETE.

REMOVE sirve para borrar propiedades y etiquetas:

```

1 MATCH (guillermo:Director {nombre: 'Guillermo del Toro Gómez'})
2 REMOVE guillermo.nacionalidad
3 REMOVE guillermo:Persona

```

Ilustración 8 Uso de REMOVE

DELETE sirve para borrar nodos o relaciones, pero estos no deben de tener relaciones, sin embargo, se utiliza DETACH junto con DELETE para borrar al nodo junto con todas sus relaciones.

```

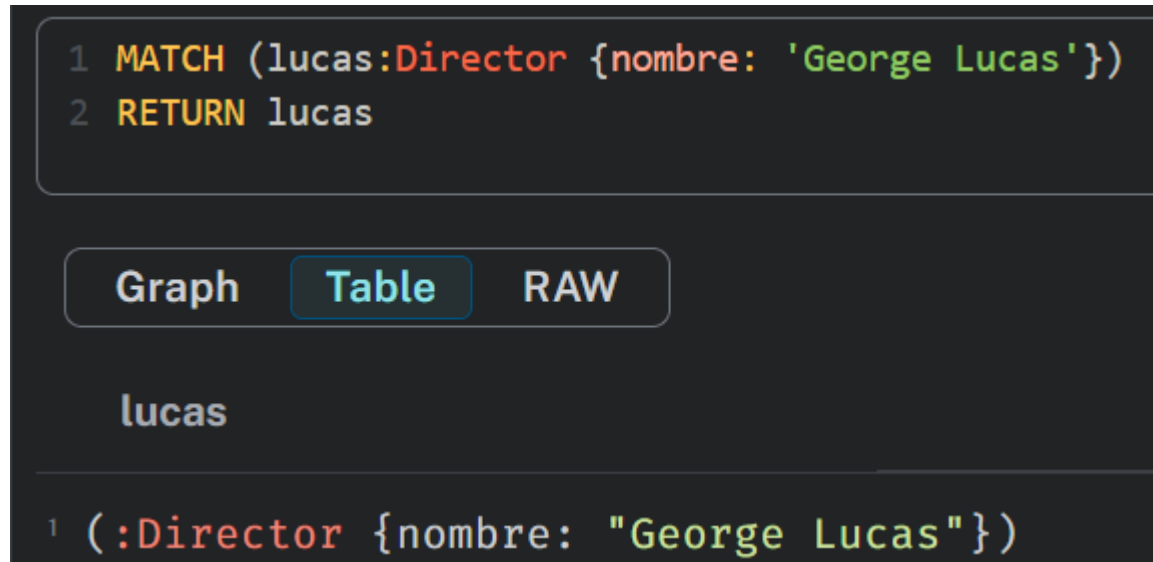
1 //Borramos la relación
2 MATCH (:Director {nombre: 'Guillermo del Toro Gómez'})-[x:DIRIGE]->(y:Pelicula {nombre: 'La cumbre escarlata'})
3 DELETE x
4 //Borramos el nodo Película asegurándonos de borrar sus relaciones también
5 DETACH DELETE y

```

Ilustración 9 Uso de DELETE y DETACH

- **Consultas**

El principal comando para realizar consultas es MATCH, el cual selecciona el o los nodos que coincidan con esa información. Para visualizar los resultados guardamos el nodo o la relación en una variable y usamos RETURN.



```
1 MATCH (lucas:Director {nombre: 'George Lucas'})
2 RETURN lucas
```

Graph Table RAW

lucas

```
1 (:Director {nombre: "George Lucas"})
```

Ilustración 10 Resultado de una consulta simple

Si embargo, si queremos realizar consultas más complejas podemos utilizar el comando WHERE, el cual recibe una condición y busca en toda la base de datos aquellos nodos o relaciones que cumplan con los criterios de MATCH y WHERE.

```
1 MATCH (p:Pelicula)-[:SE_ESTRENA]->(a:Anio)
2 WHERE a.valor > 2000
3 RETURN p.titulo, a.valor
```

	p.titulo	a.valor
1	"John Wick"	2014
2	"El Hobbit: La batalla de los cinco ejércitos"	2014
3	"John Wick: Capítulo 2"	2017
4	"Piratas del Caribe: La venganza de Salazar"	2017
5	"John Wick: Capítulo 3 – Parabellum"	2019

Ilustración 11 Ejemplo de WHERE donde se buscan películas cuyo año de estreno sea superior a 2000

```
1 MATCH (a:Actor)
2 WHERE a.nombre STARTS WITH 'Keanu'
3 RETURN a.nombre
```

Table

RAW

a.nombre

"Keanu Reeves"

Ilustración 12 Ejemplo donde se utiliza STARTS WITH para buscar actores con cierto nombre

Existen otras funciones muy utilizadas para consultar como COUNT() que contabiliza la cantidad de coincidencias de una búsqueda por cada nodo encontrado y las guarda en una variable determinada. Este comando se suele utilizar con ORDER BY donde se indica qué parámetro se va a tomar en cuenta para ordenar la lista y en qué forma, ASC para ascendente y DESC para descendente.

```
1 MATCH (d:Director)-[:DIRIGE]->(p:Pelicula)
2 RETURN d.nombre, COUNT(p) AS cantidadPelículas
3 ORDER BY cantidadPelículas DESC
```

	d.nombre	cantidadPelículas
1	"Peter Jackson"	6
2	"George Lucas"	4
3	"Chad Stahelsk i"	4
4	"David Yates"	4
5	"Gore Verbinsk i"	3
6	"Chris Columbu s"	2
7	"Irvin Kershne r"	1

Ilustración 13 Uso del COUNT y ORDER BY

Bibliografía

Neo4j documentation - Neo4j Documentation. (s. f.). Neo4j Graph Data Platform.

<https://neo4j.com/docs/>

colaboradores de Wikipedia. (2024, 4 septiembre). *Neo4J*. Wikipedia, la Enciclopedia Libre.

<https://es.wikipedia.org/wiki/Neo4j>

¿Qué es una base de datos de grafos?- Explicación de base de datos de grafos - AWS. (s. f.).

Amazon Web Services, Inc. <https://aws.amazon.com/es/nosql/graph/>