IIC3432 — Tópicos Avanzados en Bases de Datos

Neo4j

1. Para empezar

En este pdf encontrarás una guía para ayudarte a navegar por el sistema Neo4j, para que tengas un sabor de cómo funcionan las bases de datos de grafos.

Puedes bajar el desktop edition en `https://neo4j.com/download/`.

2. Cypher, basic and complex graph patterns

Una vez bajado Neo4j, puedes hacer click en el explorador, que ya tiene cargada una base de datos de películas.

Para ver el grafo, siempre puedes hacer

```
MATCH (n) RETURN n
```

2.1. Cypher

Cypher es el lenguaje de consulta de la base de datos Neo4j. Hay una tarjeta de referencia en `http://neo4j.com/docs/cypher-refcard/current/`, y el manual completo está en `https://neo4j.com/docs/cypher-manual/current/`. Seguro los vas a necesitar para terminar esta guía, y como siempre los buscadores web también son tus amigos.

La estructura basica de consultas en Cypher es bastante similar a la de SQL. Quizá algo que llama inmediatamente la atención es lo fácil que resulta expresar patrones básicos.

Patterns. Los patrones en Cypher se expresan así:

- `(p) --> (q)` y `(p) <-- (q)`, que representa una arista dirigida desde `p` a `q` o desde `q` a `p`, respectivamente, y `(p) --> ()`, que representa aristas de `p` a *cualquier* nodo. Aquí `p` y `q` representan identificadores de nodo (node IDs).
- `(p) -[n:LABEL]-> (q)`, representa una arista dirigida con ID `n` desde `p` a `q`, y etiquetada (con label) `LABEL`.

- Podemos combinar patrones, e.g. `(p) <-- (q) --> (r)` o si queremos usar labels, `(p) <-[n:LABEL]- (q) <-- (r)`

Consultas. Para hacer algo que corra de verdad tenemos que hacer `MATCH` de un patrón en el grafo, y hacer `RETURN` a algunos identificadores de nodos o aristas. Esto se parece a la parte SELECT FROM WHERE en SQL. La forma básica de consultas es como sigue (`p` es un patrón como los de arriba, y n_i son IDs en este patrón):

```
MATCH p
RETURN n1, n2, ...
```

Como siempre, para retornar todo uno usa `*`.

Armar algunas consultas básicas. Trata de escribir algunas consultas. Trata de obtener todas las películas. Trata de obtener todos los actores/actrices. Trata de encontrar todas las películas donde actuó alguien en específico, o dirigidas por alguien en específico. Ahora obtén todas las películas donde el director también actuó en ese película.

Baconfest. Escribe las siguientes consultas. Esto tiene que ver con la discusión sobre semántica y sobre simular todo en una base de datos relacional:

- Encuentra todos los pares de actores/actrices que hayan actuado en "Unforgiven"
- Mira los resultados de esa consulta. Se repiten los actores? Por qué? Cuál es la semántica en Cypher?
- En el paper se habla de una 'homomorphism-based semantics". Podemos simularla en Cypher? Trata de hacerlo encadenando dos `MATCH`. E.g. `MATCH p1 MATCH p2 RETURN *`. Qué pasa con la película de arriba cuando hacemos esto? Será verdad que siempre podemos simular esa semántica?
- Encuentra todas las películas donde Clint Eastwood actuó, dirigió, o ambas. Usa `UNION ALL`.
- Encuentra todas las películas donde Clint Eastwood actuó pero no dirigió. Qué operador usarías?
- Si no sabes lo que es, busca sobre el "Bacon number", o número de Bacon.
- Encuentra todas las películas donde actuó Kevin Bacon.
- Encuentra todas las actrices/actores que actuaron en una película junto a Kevin Bacon.
- Encuentra todas las personas con número de Bacon igual o menor a dos. Puedes hacerlo sin `UNION ALL`?

- Encuentra todos los actores/actrices que actuaron en al menos dos películas distintas junto a Kevin Bacon.
- Podemos encadenar también dos `MATCH` para definir patrones que no se pueden escribir en una sola línea. Trata de definir una consulta así. Crees que podrías caracterizar todas las consultas que se pueden escribir usando solo una cláusula de `MATCH`?

Asi nomás con Neo. Corre la consulta `MATCH c= (p) -[n]-> (q) RETURN * LIMIT 10` Qué es el `c` que estás retornando?

3. Consultas navegacionales (caminos)

Ahora pasamos a consultas de caminos. La idea es ver qué soporta Cypher, y cuales son algunas de sus limitaciones.

La principal forma de navegar en Cypher son las 'regular path queries". Por ejemplo, podemos usar `*` para un camino arbitrario, o usar `knows*` para mostrar una consulta que atraviesa solo por aristas etiquetadas con `knows`. También podemos especificar un mínimo y un máximo de veces por repetir (e.g. `[knows*2..7]`).

Trata primero de correr una consulta que compute a todas las actrices/actores con número Bacon positivo. Te funciona esa consulta?

Ahora construye un ejemplo simple de una red social (como el que vimos en clases), ponlo como una empty database (carpeta sin nada) para que Neo4j le haga load y luego pobla ese grafo con algunos nodos/aristas.

Escribe las siguientes consultas

- La relación de friend-of-a-friend, es decir, la clausura transitiva de la relación friend
- Todos los pares de nodos conectados mediante algún camino. Se repiten los nodos? por qué?
- Escribe alguna consulta que involucre la relación friend pero también la relación tag.
- Asegúrate de que haya un loop en Julia dada por alguna relación de amigos. Escribe una consulta que retorne todos los amigos de Julia, y luego una que retorne los caminos también (como la consulta de la sección anterior). Cuantos caminos obtienes? se corresponde eso con la cantidad total de caminos?
- Ahora hace lo mismo pero retorna los caminos más cortos. Compara las respuestas.
- Crees que se puedan expresar todas las regular path queries? si no, qué cosas no se pueden expresar?

Ahora vuelve a las películas, y escribe consultas para:

- Todos los nodos conectados entre si.
- todas las actrices/actores con número Bacon positivo.
- Verifica que hay un camino entre Kevin Bacon y un actor de tu elección
- Todas las personas con número Bacon entre 2 y 4.

Te corrieron bien esas consultas? Dónde hay problemas?

Iterando sobre caminos. Cypher también te deja iterar sobre los nodos que pertenecen a algún camino que retornas. Puedes ver cómo se hace? Qué tipo de cosas puedes hacer?

4. Agregación

Nuevamente sobre las películas, escribe consultas para encontrar:

1. Número de nodos en el grafo
2. Número de películas en las que actúa Kevin Bacon.
3. El promedio de actrices/actores por película
4. El actor/actriz que actúa en más películas
5. La actriz/Actor con mas caminos a Kevin Bacon.

Se puede expresar todo esto? Cuáles corren bien? cuáles crees tú que son más demandantes?