PROYECTO INTEGRADOR: NEO4J

Base de Datos No Convencionales

DESCRIPCIÓN BREVE

En el presente documento presento lo que realice y la documentación del uso de Neo4j.

MONTERO MUÑIZ MONSERRAT

Contenido

Introducción

Neo4j es una base de datos NoSQL orientada a grafos la cual almacena la información en grafos formados por nodos y aristas.

Hace uso de los grafos para representar datos y las relaciones entre ellos. Un grafo se define como cualquier representación gráfica formada por vértices (se ilustran mediante círculos) y aristas (se muestran mediante líneas de intersección). Dentro de estas representaciones gráficas, tenemos varios tipos de grafos:

- Grafos no dirigidos: los nodos y las relaciones son intercambiables, su relación se puede interpretar en cualquier sentido. Las relaciones de amistad en la red social Facebook, por ejemplo, son de este tipo.

- Grafos dirigidos: los nodos y la relaciones no son bidireccionales por defecto. Las relaciones en Twitter son de este tipo. Un usuario puede seguir a determinados perfiles en esta red social sin que ellos le sigan a él.

- Grafos con peso: en este tipo de gráficas las relaciones entre nodos tienen algún tipo de valoración numérica. Eso permite luego hacer operaciones.

- Grafos con etiquetas: estos grafos llevan incorporadas etiquetas que pueden definir los distintos vértices y también las relaciones entre ellos. En Facebook podríamos tener nodos definidos por términos como 'amigo' o 'compañero de trabajo' y la relaciones como 'amigo de' o 'socio de'.

- Grafos de propiedad: es un grafo con peso, con etiquetas y donde podemos asignar propiedades tanto a nodos como relaciones (por ejemplo, cuestiones como nombre, edad, país de residencia, nacimiento). Es el más complejo.

Neo4j utiliza grafos de propiedad para extraer valor añadido de los datos de cualquier empresa con gran rendimiento y de una forma ágil, flexible y escalable.

Ventajas	Desventajas
• Modelo muy natural para representar datos conectados. Directamente leíble y fácil de interrogar. • Implementa consultas referidas a la estructura en grafo (recorridos, adyacencia, etc.), gracias al uso de algoritmos basados en grafos (A*, Dijsktra, etc.). • No O/R mismatch – mapeo simple del grafo al lenguajes orientado a objetos como Ruby, Java, C#, phyton. • Es altamente disponible (HA) y tolerante a la partición (AP), lo que puede hacer que devuelva datos fuera de sync.	• El modelo de datos no está estandarizado -> dificultad de cambio de gestor. • Cypher no es un lenguaje estandarizado. • Falta de herramientas para ETL, modelado. • Replicación de grafos completos, no de subgrafos (sharding).

CQL (Cypher Query Language)

Cypher es el lenguaje de consulta de gráficos abiertos de Neo4j. La sintaxis de Cypher proporciona una forma familiar de unir patrones de nodos y relaciones en el gráfico.

Creación de nodos, etiquetas, propiedades

La creación de un nodo simple es: CREATE (n)

La creación de un nodo con etiquetas es: CREATE (n:Product)

La creacion de nodos con datos y una estiqueta es: CREATE (p:Product {name:'Ipad Air',price:450}) RETURN p

Las datos para la creacion de la base de datos fueron tomados de
http://www.diegocalvo.es/insertar-elementos-en-un-grafo-en-neo4j-ejemplo/

Creacion de los nodos

```
CREATE (Paco:Person {name:'Paco', born:1964})
CREATE (Juan:Person {name:'Juan', born:1967})
CREATE (Andres:Person {name:'Andres', born:1961})
CREATE (Hugo:Person {name:'Hugo', born:1960})
CREATE (Natalia:Person {name:'Natalia', born:1967})
CREATE (Miriam:Person {name:'Miriam', born:1965})
CREATE (Rosa:Person {name:'Rosa', born:1952})

CREATE (Telefonica:Company {name:'Telefonica', central_office:'Madrid',
sector:'telecomunicaciones'}),
  (Repsol:Company {name:'Repsol', central_office:'Madrid', sector:'energia'})
```

Creaciones de relaciones (relaciones entre nodos, relaciones con etiquetas, relaciones con propiedades)

Crear relación simple

Creamos los nodos y después usamos las variables para relacionarlos.

```
1CREATE (ipadAir:Product {name:'Ipad Air',price:450})
2CREATE (user1:User {name:'user 1'})
3CREATE (user2:User {name:'user 2'})
4
5CREATE
6  (user1)-[:BUY]->(ipadAir),
7  (user2)-[:BUY]->(ipadAir)
8
9RETURN ipadAir,user1,user2
```

Es una relación con la etiqueta BUY, es decir, de tipo BUY y tiene una dirección.
Las relaciones también pueden tener propiedades

```
1CREATE (ipadAir:Product {name:'Ipad Air',price:450})
2CREATE (user1:User {name:'user 1'})
3CREATE (user2:User {name:'user 2'})
4CREATE
5  (user1)-[:BUY{ date:'2014/04/1'}]->(ipadAir),
6  (user2)-[:BUY{ date: '2014/03/1'}]->(ipadAir)
7
8RETURN ipadAir,user1,user2
```

Creación de Relaciones en mi base de datos

- Relación Amigos de:

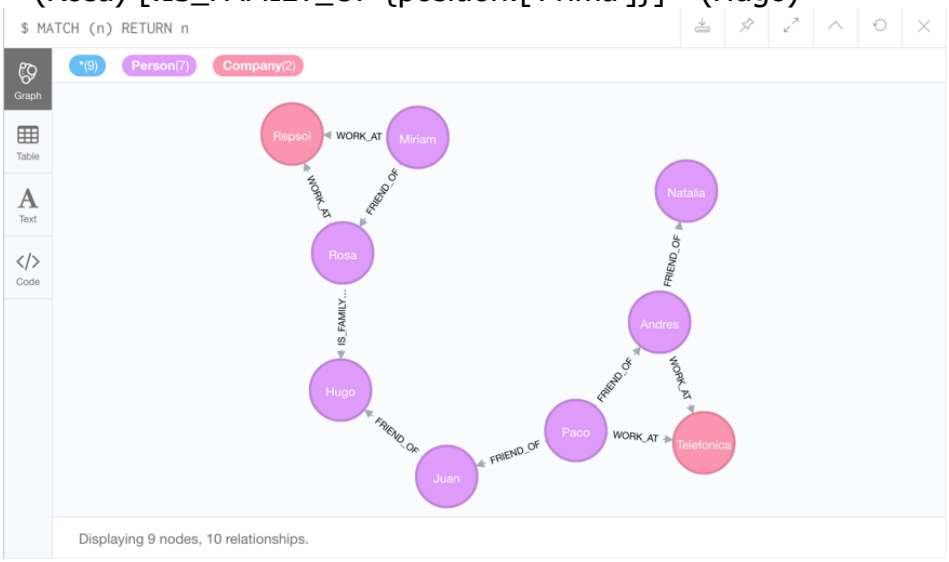
```
CREATE
  (Paco)-[:FRIEND_OF {role:['Amigo de Trabajo']}]->(Juan),
  (Paco)-[:FRIEND_OF {role:['Amigo de Trabajo']}]->(Andres),
  (Juan)-[:FRIEND_OF {role:['Amigo de la infancia']}]->(Hugo),
  (Andres)-[:FRIEND_OF {role:['Amigo de la infancia']}]->(Natalia),
  (Miriam)-[:FRIEND_OF {role:['Amigo de Trabajo']}]->(Rosa)
```

- Relacion trabaja en:

```
CREATE
  (Paco)-[:WORK_AT {position:['Director de Marketing']}]->(Telefonica),
  (Andres)-[:WORK_AT {position:['Director de Marketing']}]->(Telefonica),
  (Miriam)-[:WORK_AT {position:['Director de Marketing']}]->(Repsol),
  (Rosa)-[:WORK_AT {position:['Director de Marketing']}]->(Repsol)
```

- Relacion es familia de:

```
CREATE
  (Rosa)-[:IS_FAMILY_OF {position:['Prima']}]->(Hugo)
```



Modificar

Modificar propiedades a un nodo

```
MERGE (p:Person {name: 'Paco'}) SET p.age = 34, p.coat = 'Yellow' RETURN p
```

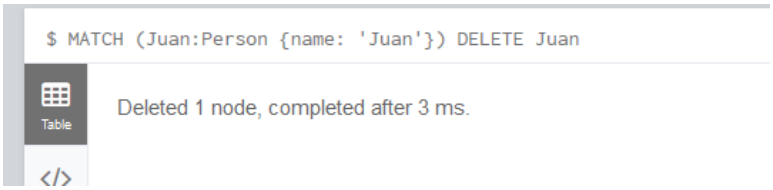
Modificar propiedades a una relación

MERGE (Paco)-[r:FRIEND_OF]->(Juan) SET r.ages = 34 RETURN r

Borrar nodos

Nota: Para poder borrar los nodos se tienen que borrar las relaciones entre ellos

MATCH (Juan:Person {name: 'Juan'}) DELETE Juan



Borrar relaciones entre nodos

MATCH (Miriam)-[:FRIEND_OF]->(Rosa) DELETE r
Borrar todo el grafo

MATCH (n) OPTIONAL MATCH (n)-[r]-() DELETE n,r

Consultas (funciones de agregación)

Contar elementos derivados de dos relaciones

Contar para cada persona el número de amigos que tiene trabajando en los diferentes sectores

MATCH (p1:Person)-[:FRIEND_OF]->(p2:Person)-[:WORK_AT]->(c:Company)
RETURN p1.name AS Persona , COUNT(c.sector) AS Sectores

```
$ MATCH (p1:Person)-[:FRIEND_OF]->(p2:Person)-[:WORK_AT]->(c:Company) RETUR…
```

Table | Text | Code

Persona	Sectores
"Andres"	1
"Paco"	2
"Natalia"	1
"Juan"	1
"Miriam"	1
"Rosa"	1

Agrupaciones

Para cada persona agrupar por sectores en que trabajan sus amigos

MATCH (p1:Person)-[:FRIEND_OF]->(p2:Person)-[:WORK_AT]->(c:Company)
RETURN p1.name, COLLECT(c.sector)

```
$ MATCH (p1:Person)-[:FRIEND_OF]->(p2:Person)-[:WORK_AT]->(c:Company) RETUR…
```

Table | Text | Code

p1.name	p2.name	c.sector
"Paco"	"Andres"	"telecomunicaciones"
"Miriam"	"Rosa"	"energia"
"Andres"	"Natalia"	"alimentacion"
"Natalia"	"Juan"	"alimentacion"
"Paco"	"Juan"	"alimentacion"
"Rosa"	"Hugo"	"alimentacion"
"Juan"	"Hugo"	"alimentacion"

Referencias

https://bbvaopen4u.com/es/actualidad/neo4j-que-es-y-para-que-sirve-una-base-de-datos-orientada-grafos Fecha de consulta: 02/06/2018

https://neo4j.com/

http://xurxodeveloper.blogspot.com/2014/04/cypher-como-crear-datos.html

https://neo4j.com/blog/neo4j-video-tutorials/

http://www.diegocalvo.es/tutorial-neo4j-espanol/