TUTORIAL NEO4J

Sistemas de Bases de Datos

Prof. María Constanza Pabón

En este tutorial se aplican las operaciones básicas de Neo4J, con sus parámetros básicos. Para mayor detalle por favor refiérase al manual.

Para desarrollar este tutorial usaremos datos de películas (datos adaptados de: Open Movie Database, http://www.omdbapi.com/)

1. Servidor

   Neo4j solamente guarda un grafo. Por lo tanto, se sugiere que cada uno instale Neo4j en su computador para evitar conflictos.

   Si usa el servidor instalado en labsistemas, anteponga sus iniciales a los labels de los nodos y arcos (XXPerson, XXMovie, …).

   Conexión al servidor en labsistemas

   a. Conectarse a labsistemas.javerianacali.edu.co

      ssh bd0X@labsistemas.javerianacali.edu.co -p 9020

   b. Conectarse al servidor Neo4j

      ssh bd0X@serverneo4j

   c. Ejecutar el Shell y conectarse con su usuario

      cypher-shell
      (usuario bd0X, password $bd0X!)

      Salir del shell: :exit

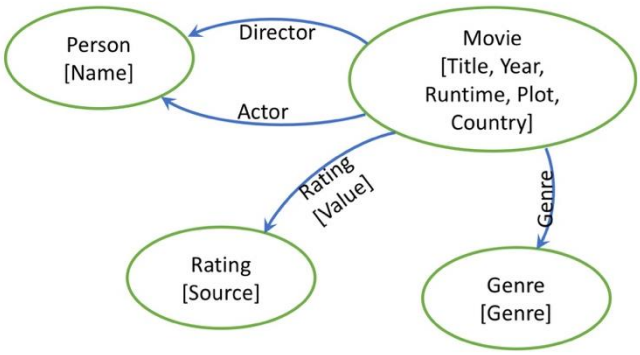
   d. Si es necesario, ejecutar neo4j

      sudo neo4j start
      sudo neo4j stop

2. Modelo de datos

La información consta de nodos tipo: Person (con atributo Name), Movie (con atributos Title, Year, Runtime, Plot, Country), Rating (con atributo Source), y Genre (con atributo Genre).

Las relaciones son: Director (entre Movie y Person), Actor (entre Movie y Person), Rating (entre Movie y Rating, con atributo Value), y Genre (entre Movie y Genre)



3. Crear nodos

```
CREATE (variable:Etiqueta {attributeName1: value1, attributeName2:
value2, … });
```

Ejemplos:

```
CREATE (p:Person { Name: 'Tim Burton' } );
CREATE (p:Person { Name: 'Mark Wahlberg' } );
CREATE (r:RatingSource { Source: 'Internet Movie Database' } );
CREATE (m:Movie { Title:  'Planet of the Apes', Year:2001, Runtime: '119
min', Plot: 'An Air Force astronaut crash lands on a mysterious planet
where evolved, talking apes dominate a race of primitive humans. ',
Country: 'USA'  });
CREATE (g:Genre {Genre : 'Action' });
```

Ejercicio:

- Ejecutar los ejemplos
- Consultar los nodos creados

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

4. Crear arcos

```
MATCH (n1: Etiqueta), (n2:Etiqueta) …
WHERE n1.atributo1 = valor1 AND n2.atributo2 = valor2 …
CREATE (n1)-[r1:Relacion1]->(n2), (n1)-[r2:Relacion2]->(n3) …;
```

```
MATCH (n1: Etiqueta {atributo1:valor1} ), (n2:Etiqueta {atributo2:valor} ) ...
CREATE (n1)-[r1:Relacion1]->(n2), (n1)-[r2:Relacion2]->(n3) ...;
```

Ejemplos:

```
MATCH (m:Movie), (p:Person)
WHERE m.Title = 'Planet of the Apes' AND p.Name = 'Tim Burton'
CREATE (m)-[r:Director ]->(p);

MATCH (m:Movie {Title:'Planet of the Apes'} ), (p:Person {Name:'Mark
Wahlberg'} )
CREATE (m)-[r:Actor ]->(p);

MATCH (m:Movie), (r:RatingSource)
WHERE m.Title = 'Planet of the Apes' AND r.Source='Internet Movie
Database'
CREATE (m)-[rel:Rating {value: 5.7}]->(r);

MATCH (m:Movie), (g:Genre)
WHERE m.Title = 'Planet of the Apes' AND g.Genre='Action'
CREATE (m)-[rel:Genre]->(g);
```

Ejercicio:

- ejecutar los ejemplos
- consultar los datos:

```
MATCH (m:Movie)-[rel]->(n) RETURN m.Title, rel, n;
```

- Para cargar los datos, ejecutar los comandos del archivo moviesNeo4j.txt

5. Reglas de integridad del esquema

Ejemplo:

```
CREATE CONSTRAINT ON (p:Person) ASSERT p.Name IS UNIQUE;

DROP CONSTRAINT ON (p:Person) ASSERT p.Name IS UNIQUE;
```

Constraints: EXISTS (una propiedad es obligatoria para un tipo de nodo o de relación), NODE KEY

(Mayor información: http://neo4j.com/docs/developer-manual/current/cypher/schema/constraints/#query-constraint-unique-nodes)

6. Consultas

```
MATCH  pattern WITH expression WHERE condition  RETURN variables;
```

Ejemplos:

```
MATCH  (m:Movie {Country: 'USA'}) RETURN m.Title, m.Plot;

MATCH  (m) WHERE m.Year >= 2000 RETURN m;

MATCH  (m:Movie)-[r]->(n) WITH m, n ORDER BY n RETURN n, m.Title;

MATCH  (m:Movie)-[r:Actor]->(n) WITH m, count(n) as NumActors ORDER BY NumActors RETURN m.Title, NumActors;
```

Ejercicio:

- ejecutar los ejemplos
- Realizar las siguientes consultas:
  a. Seleccionar el título y los ratings de las películas dados por "Metacritic"

  b. Seleccionar el nombre del director y el título de las películas realizadas desde 2005 y con al menos un rating mayor que 6.0. Incluir también la fuente que generó el rating. Ordene los resultados por rating en orden descendente.

  c. Seleccionar los títulos de las películas, el nombre de su director, y si tienen ratings registrados, la fuente y valor del rating.

  d. Seleccione el nombre de cada actor y el promedio de los ratings de las películas en que ha participado.

  e. Encontrar los nombres de los actores que han actuado en una película donde también actuó uno de los actores de '2001: A Space Odyssey'

f. Encuentre todos los nodos que están a distancia 3 de la película ‘2001: A Space Odyssey’

7. Actualizar
   Cambiar el RETURN del query por SET (o REMOVE) seguido de la lista de atributos a actualizar.
   Ejemplo:

```
MATCH (m:Movie {Title: 'Planet of the Apes'})-[rd:Director]->(d)
SET d.Name = 'Otro';
```

8. Borrar
   Cambiar el RETURN del query por DELETE seguido de la lista de nodos o relaciones a borrar, referenciados por las variables.
   Ejemplo:

```
MATCH (m:Movie {Title: 'Planet of the Apes'})-[r1]->(n1), (m:Movie)<-[r2]-(n2)
DELETE m, r1, r2;
```

9. Crear índices
   Ejemplo:

```
CREATE INDEX ON  :Movie(Title)
```

10. Importar nodos desde csv

```
LOAD CSV WITH HEADERS FROM 'file:///Clientes.csv' AS line FIELDTERMINATOR
';'
CREATE (p:Person {Name:line.Name, direccion:line.direccion, email:line.correo});
```

El archivo debe estar en /var/lib/neo4j/import/ o modificar la ruta en /etc/neo4j/neo4j.conf

11. Importar arcos desde csv

```
LOAD CSV WITH HEADERS FROM 'file:///telefonosregistrados.csv' AS line
FIELDTERMINATOR ';'
MATCH (l:Lineas {Numero:line.numeroLinea}),
        (p:Person {Name:line.NombreCliente} )
CREATE (p)- [r:dueño]->(l)
```

12. Conexión desde Python

En serverneo4j están instalados el py2neo y el neo4j Python driver para Python 2 (py2neo-3.1.2). El driver oficial de Neo4j es el Neo4j Python Driver.

Neo4j Python driver

```
from neo4j.v1 import GraphDatabase
uri = "bolt://localhost:7687"
driver = GraphDatabase.driver(uri, auth=("bd10", "$bd10!"))

def create_Person(name):
   with driver.session() as session:
       with session.begin_transaction() as tx:
           result = tx.run("CREATE (p:Person { Name: 'Tim Burton' }) RETURN
id(p)")
   return result.single()[0]

def print_Persons ():
    with driver.session() as session:
        with session.begin_transaction() as tx:
            for record in tx.run("MATCH (n) RETURN n"):
                print(record)
```

py2neo

```
from py2neo import Graph, Node, Relationship
graph = Graph(user='bd0X', password='$bd0X!')

tx = graph.begin( )          # inicia una transacción
a = Node("Person", name="Alice")
tx.create(a)
b = Node("Person", name="Bob")
tx.create(b)
ab = Relationship(a, "Knows", b)
tx.create(ab)
```

```
tx.commit( )
graph.exists(ab)

a=graph.run("MATCH (n) RETURN n").data()
for i in range(0,len(a),1):
...      print a[i]["n"]
...      print a[i]["n"].labels()
...     print a[i]["n"].properties["name"]
```

```
        LOAD CSV FROM "file:///Orden.csv" AS line FIELDTERMINATOR ','
CREATE (:f2255 {categoria :line[10], ciudad :line[9], precio
:line[8],nom_producto:line[7], cantidad :line[6],estado :line[5],fechaOrden
:line[4],nomCliente: line[3],idCliente: line[2],producto: line[1], orden: line[0]})

        MATCH (c:cliente),(o:orden)
        WHERE c.idCliente = '294' AND o.nroOrden = '5102'
CREATE (c)-[r:ordeno ]->(o);
```