UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
School of Professional & Executive Development

Big Data Management and Analytics
Session: Graph Databases - Neo4j

Lecturer: Rizkallah Touma and Jovan Varga

November 30th, 2016

1 Required Tools

- SSH and SCP client
- `eclipse` IDE with JDK 7 and Maven plugin installed

2 Tasks To Do Before The Session

Before coming to this Graph Databases session, you should have checked the session slides and write down all doubts you might have about them.

3 Part A: Examples & Questions (30min)

The first 30 minutes of the session will be spent on answering all the possible questions you might have come up with during the week by preparing the tasks before the session (see above).

4 Part B: In-class Practice (2h 30min)

4.1 Exercise 1 (15min): Setup of the environment

Setting up Neo4j in this case is very easy as it is not a distributed database like we saw in the Hadoop ecosystem practical sessions. That means today we are only going to work in the master node. To start with, you need to get Neo4j from the tarballs folder you have.

```
cp tarballs/neo4j-community-2.3.0-M03-unix.tar.gz .
tar xf neo4j-community-2.3.0-M03-unix.tar.gz
```

Now open and edit the file the configuration file *neo4j-community-2.3.0-M03/conf/neo4j-server.properties* and update/uncomment the following properties already defined and which you will find along this file:

```
org.neo4j.server.database.location=/home/bdma**/data/neo4j/graph.db
org.neo4j.server.webserver.address=0.0.0.0
dbms.security.auth_enabled=false
```

At some point during the following exercises, you will be asked to compare Cypher to SQL queries so some statements need to be run on MySQL before continuing. Prior to this, you need to know the credentials to log in MySQL are ***user:root/password:MyNewPass***. The command to log in is:

```
mysql -u root -p
```

And what you need to do here is simply to create the *labo8* database:

```
CREATE DATABASE labo8;
```

And whenever you want to query any of this database tables, you will first need to:

```
USE labo8;
```

Finally, let's populate both databases. The population of both Neo4j and MySQL is completely given by the Java program you are given but you will need to compile (remember to update the database path in *Launch.java*) and upload it to the master node. Once this is done, you will need to run (to populate with 50 nodes):

```
jre1.7.0_75/bin/java -jar labo8.jar 50
```

And then start the Neo4j server up (the first time you run it after booting up the machines you will need to define the *JAVA_HOME* variable):

```
export JAVA_HOME="/home/bdma**/jre1.7.0_75"
neo4j-community-2.3.0-M03/bin/neo4j start
```

To stop Neo4j (you will need to do is every single time you want to populate the database), the command is the following:

```
neo4j-community-2.3.0-M03/bin/neo4j stop
```

4.2 Exercise 2 (15min): Understanding the model

If you successfully started up the server, you should be able to open a web UI at port 7474 of your master (check your cluster mail). The advantage of using this UI is that it really helps at graphically visualizing the result of our queries, but its main drawback is that it can be very heavy for the computer to keep it open if the results are too large. Nevertheless, we are going to use it during the first part of the session.

What you are asked to do in this exercise is to spend some minutes understanding the graph model you are given. To get a full picture of it you can run:

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

Then choose one node as starting point for the following queries. To do so run:

```
MATCH (n: Customer)
RETURN n.number
LIMIT 10;
```

And pick whatever node you like the most from the list. We will call this random customer C from now on.

4.3 Exercise 3 (1h): Querying the Graph Database with Cypher

In this exercise you are asked to obtain the following data from Neo4j. Firstly, queries on relationships:

1. Query all the relationships that customer C has with any other customer. If your result does not show many relationships, please choose a different C for the next exercises.

2. Query all the customers that called C or are called by C. Let's call this query **Q1**.

3. Query all the websites that customers called or texted by C visit.

4. Query all the websites that customers called or texted by C visit and which C also visits. Let's call this query **Q2**.

Secondly, queries on aggregations:

1. Query, by customer number, the total price he/she paid for the calls he/she made. Order (descending) the result by this total price. Let's call this query **Q3**.

2. Query, by website URL, the total number of visits it received on Saturdays and Sundays. Order (descending) the result by this total number of visits.

3. Query, by each pair of customers, the total number of interactions (calls or texts) that happened between them. Order (descending) the result by this total number of interactions.

Thirdly, subqueries:

1. Query the pair of customers with the longest call. Let's call this query **Q4**.

4.4 Exercise 3 (1h): Comparing to SQL

In this exercise you are asked to compare the flexibility and the performance of Cypher query language to SQL. For such reason, you were previously given the details of MySQL. In order to compare the performance, we will need some more data to be loaded in the database. The more the better, but just to prevent it from taking too long my advise is to repopulate with 100 nodes.

Once this is done you have to provide the following information:

1. Translate queries **Q1, Q2, Q3** and **Q4** in SQL and run them in MySQL. To check the correctness of your solution, you can also run them back in Neo4j through its shell instead of through its web UI. This shell can be opened by running:

   ```
   neo4j-community-2.3.0-M03/bin/neo4j-shell
   ```

   You can return only certain attributes from the nodes and from the relationships to help at easily check the correctness of your new queries and you can also use DISTINCT keyword in both Neo4j and SQL to simplify getting rid of some duplicated rows that might pop up in the result.

2. What type of queries do you think are easier to write in Cypher, and which in SQL?

3. Write down the execution times between your Cypher and SQL queries and discuss what queries run faster in Neo4j and what queries run faster in MySQL. In what kind of problems you would go for a graph database rather than a relational one? Justify your answers.

Deliverable: Deliver in hand or upload the answer sheet you have been given at the beginning of the class. If you upload it, name such file as "name_surname.pdf"

5 Part C: Optional Practice (3h)

In this exercise you are asked to work on tuning Neo4j. To do so, you should firstly add more nodes to the database (around 500 might be good; it will take five minutes to insert). Your goal is to optimize the same queries as before **Q1, Q2, Q3** and **Q4**. Some examples of tuning techniques in graphs you can use for this purpose are:

- Rethink the graph schema we are using and change it for convenience.
- Use Neo4j indexes.

- Use artificial nodes to group some other nodes of interest for the query.
- ...

Deliverable: Write and upload a small document (1-2 pages) explaining the tuning you have done and comparing the new execution time with the old one. Name such file as "name_surname.pdf"