Graph Database Topics.

Assignments – Neo4j

You will be querying three Neo4j databases, provided to you together with
the software. These databases are: (1) A graph representation of the
Northwind operational database, denoted northwindhg.db; (2) A
database containing information about movies, denoted movies.db; (3) A
database containing historical information about the world cups, from the
start through the 2014 world cup, denoted worldcup.db.

Once you login to your account (in Linux), you need to go to the /tmp
folder. Once there, open a terminal and type the following commands:

cd /tmp
tar xJf /serveur/neo4j.tar.xz
cd neo4j

At this point, you are about to start the Neo4j server. First, you need to
choose the database you will work with. For this, you go to the conf
folder, and edit the neo4j.conf file. You will find something like this:

#dbms.active_database=graph.db
#dbms.active_database=trajectories-NYC-4sq.db
#dbms.active_database=trajectories.db
#dbms.active_database=worldcup.db
#dbms.active_database=web.db
#dbms.active_database=telco.db
dbms.active_database=northwindhg.db
#dbms.active_database=movies.db
#dbms.active_database=calls.db

Since dbms.active_database=northwindhg.db is unmarked, when you start
the server, you will also load the northwindhg.db database. To change it
to the movies.db, you mark #dbms.active_database=northwindhg.db,
and unmark dbms.active_database=movies.db. Save the changes, and
quit the file. Then you run: . /bin/neo4j console

And the server starts. Then, open a browser, and type the following url:
localhost:7474.
Now you can start writing Cypher queries.
Assignment 1.

Consider the Northwind database, whose schema is:

This database has been exported to Neo4j, and you can find it at: /........./data/databases/northwindhg.db. The graph schema is:
Write in Cypher the following queries over the northwindhg.db database:

1 - List products and their unit price.

MATCH (p:Product)
RETURN p.productName, p.unitPrice
ORDER BY p.unitPrice DESC

2 - List information about products 'Chocolade' & 'Pavlova'.

MATCH (p:Product)
WHERE p.productName IN ['Chocolade','Pavlova']
RETURN p

3 - List information about products with names starting with a "C", whose unit price is greater than 50.

MATCH (p:Product)
WHERE p.productName STARTS WITH "C" AND p.unitPrice > 100
RETURN p.productName, p.unitPrice;

4 - Same as 3, but considering the sales price, not the product's price.

MATCH (p:Product) <- [c:Contains] - (o:Order)
WHERE p.productName STARTS WITH "C" AND ToInt(c.unitPrice) > 15
RETURN distinct p.productName, p.unitPrice,c.unitPrice;

5 - Total amount purchased by customer and product.

MATCH (c:Customer)
OPTIONAL MATCH (p:Product)<-[pu:Contains]-(:Order)<-[::Purchased]-(:c)
RETURN c.companyName,p.productName, toInt(sum(toint(pu.unitPrice) *toInt (pu.quantity))) AS volume
ORDER BY volume DESC;

6 - Top 10 employees, considering the number of orders sold.

MATCH (:Order)<-[::Sold]-(:e:Employee)
RETURN e.firstName,e.lastName, count(*) AS Ordenes
ORDER BY Ordenes DESC LIMIT 10

7 - For each employee, list the assigned territories.
MATCH (t:Territory)<-[[:AssignedTo]]-(e:Employee)
RETURN t.name, collect(e.lastName);

8 - For each city, list the companies settled in that city.

MATCH (c:City)<-[[:LivesIn]]-(c1:Customer)
RETURN c.cityName, COLLECT(c1.companyName);

9 - How many persons an employee reports to, either directly or transitively?

MATCH (e:Employee)
OPTIONAL MATCH (e)<-[rel:ReportsTo*]-(report)
RETURN report.lastName AS employee, COUNT(rel) AS reports

10 - To whom do persons called “Robert” report to?

MATCH (e:Employee)<-[[:ReportsTo*]]-(sub:Employee)
WHERE sub.firstName = 'Robert'
RETURN e.firstName,e.lastName,sub.lastName

11 - Who does not report to anybody?

MATCH (e:Employee)
WHERE
NOT (e)-[:ReportsTo]->()
RETURN e.firstName as TopBoss

12 - Suppliers, number of categories they supply, and a list of such categories

MATCH (s:Supplier)-->(:Product)-->(c:Category)
WITH s.companyName as Company, collect(distinct c.categoryName) as Categories
WITH Company, Categories, length(Categories) AS Cantidad ORDER BY Cantidad DESC
RETURN Company, Cantidad, Categories;

13 - Suppliers who supply beverages

MATCH (c:Category {categoryName:"Beverages"})<--(:Product)<--(s:Supplier)
RETURN DISTINCT s.companyName as ProduceSuppliers;

14 - Customer who purchases the largest amount of beverages

MATCH (cust:Customer)-[:Purchased]->(:Order)-[:Contains]-(p:Product),
(p)-[:HasCategory]-(c:Category {categoryName:"Beverages"})
RETURN DISTINCT cust.companyName as CustomerName, SUM(toInteger(o.quantity)) AS TotalProductsPurchased ORDER BY TotalProductsPurchased DESC LIMIT 1;
15 - List the 5 most popular products (considering number of orders)

MATCH (c:Customer)-[:Purchased]->(o:Order)-[:Contains]->(p:Product)
return c.companyName, p.productName, count(o) as orders
order by orders desc
LIMIT 5

16 - Products ordered by customers from the same country than their suppliers

MATCH (c:Customer)-[:LivesIn]->(cy:City)<-[:Contains]<(o:Order)<-[:Purchased]-(c)
WITH co,c MATCH (s:Supplier) WHERE co.countryName=s.country
WITH s,co,c MATCH (s)-[:Supplies]->(p:Product)<-[:Contains]->(o:Order)<-[:Purchased]-(c)
RETURN c,s,co,p
Assignent 2.

**Switch to the movies.db database.** For this, stop the server, using the ctrl-c command in the terminal. Then, edit the neo4j.conf as explained, and unmark the movies.db line. Then, start the server again. When you open the browser and type the url localhost:7474, you’ll have the neo4j database available. The schema is (you can get this writing call db.schema at the prompt):

![Diagram of the movies.db database schema](image)

**Write in Cypher the following queries over the movies.db database:**

1 - Actors who played in two movies directed by the same director. Return the actor, the films, and the director.

```
MATCH p={v1:Actor}-[r1:ACTS_IN]->(m1:Movie)<-[dm1:DIRECTED]-(d1:Director)-
[dm2:DIRECTED]->(m2:Movie)<-[r2:ACTS_IN]-(v2:Actor)
WHERE m1.title <> m2.title AND v1.name=v2.name
RETURN v1.name, m1.title, m2.title,d1.name
```

2 - Actors who played in the same film with Kevin Bacon.

```
MATCH (v1:Actor)-[r1:ACTS_IN]->(m1:Movie)<-[r2:ACTS_IN]-(v2:Actor)
WHERE v1.name = 'Kevin Bacon' AND v1.name<>v2.name
RETURN v1.name,v2.name,m1.title
```
3 - Actors who played in a movie directed by Robert De Niro.

MATCH (v1:Actor)-[r1:ACTS_IN]->(m1:Movie)<-[r2:DIRECTED]-(v2:Director)
WHERE v2.name = 'Robert De Niro' AND v1.name<>v2.name
RETURN v1.name,v2.name,m1.title

4 - For each actor, list the number of actors she played with in a movie.

MATCH (v1:Actor)-[r1:ACTS_IN]->(m1:Movie)<-[r2:ACTS_IN]-(v2:Actor)
WHERE v1.name<>v2.name
RETURN v1.name, count(distinct v2.name) as friends order by friends desc

5 - Actors who played in a movie with Samuel L. Jackson.

MATCH (v1:Actor)-[r1:ACTS_IN]->(m1:Movie)<-[r2:ACTS_IN]-(v2:Actor)
WHERE v1.name= 'Samuel L. Jackson' and v1.name<>v2.name
RETURN v1.name, v2.name, m1.title

6 - Shortest path between Robert De Niro and Kevin Bacon.

MATCH (v1:Actor{name:'Robert De Niro'}),(v2:Actor{name:'Kevin Bacon'})
WITH v1,v2
MATCH p= ShortestPath((v1)-[*]-(v2))
RETURN p, length(p) as l

7 - Shortest path between Kevin Bacon and Stephen Lang.

MATCH (v1:Actor{name:Stephen Lang'}),(v2:Actor{name:'Kevin Bacon'})
WITH v1,v2
MATCH p= ShortestPath((v1)-[*]-(v2))
RETURN p, length(p) as l

8 - Shortest path between Kevin Bacon and any other actor.

MATCH (v1:Actor), v2:Actor{name:'Kevin Bacon'})
WITH v1,v2
MATCH p= ShortestPath((v1)-[*]-(v2))
RETURN p, length(p) as l
Assignment 3.

Switch to the worldcup.db database, doing the same steps as in Assignment 2. Now, the database is worldcup.db. The schema is:

Write in Cypher the following queries over the worldcup.db database:

1 - Who hosted the world cup?

MATCH (wc:WorldCup)-[:HOSTED_BY]->(country)
RETURN wc.name, wc.year, country.name
ORDER BY wc.year

2 - Who hosted the World Cup more than once, and when.

MATCH (host:Country)<-[::HOSTED_BY]-(wc)
WITH wc, host ORDER BY wc.year
WITH host, count(*) AS times, collect(wc.year) AS years
WHERE times > 1
return host.name, times, years

3 - Hosts that won the World Cup, and the result of the final match.

MATCH (match:Match {round: "Final"})<-[:played_IN]-[host:Country],
(host)<-[::HOSTED_BY]-(worldCup),
(worldCup)-[:contains_MATCH]->(match),
(match)<-[:oppositionPlayed:played_IN]-[opposition]
WHERE (hostPlayed.score > oppositionPlayed.score) OR (hostPlayed.penalties > oppositionPlayed.score)
RETURN host.name, worldCup.year, hostPlayed.score + "-" + oppositionPlayed.score AS score, opposition.name
ORDER BY worldCup.year

4 - Top scorers per world cup.

MATCH (player)-->(stats)-->(goal),
  (stats)-->(match)<-(wc:WorldCup)
WHERE goal.type IN ["goal", "penalty"]
WITH player.name AS player, count(*) AS goals,
    collect(DISTINCT wc.year) AS competitions
UNWIND competitions AS competition
WITH player, goals, competition ORDER BY player, goals, competition
RETURN player, goals, collect(competition) AS competitions
ORDER BY goals DESC
LIMIT 5

5 - Top scorer playing in the 2018 World Cup.

MATCH (player:Player)-->(stats)-->(goal),
  (stats)-->(match)<-(wc:WorldCup)
WHERE goal.type IN ["goal", "penalty"]
WITH player, count(*) AS goals
ORDER BY goals DESC
MATCH (player)-->(squad:Squad {year: 2018}),
  (squad)<-(country)
RETURN player.name, country.name, goals

6 - Which hosts won the World Cup that they hosted?

MATCH (match:Match {round: "Final"})<-[hostPlayed:PLAYED_IN]-[host:Country],
  (host)<-[HOSTED_BY]-(worldCup),
  (worldCup)<-[CONTAINS_MATCH]-(match),
  (match)<-[oppositionPlayed:PLAYED_IN]-(opposition)
WHERE (hostPlayed.score > oppositionPlayed.score) OR (hostPlayed.penalties > oppositionPlayed.score)
RETURN host.name
ORDER BY worldCup.year

7 - Which countries have never won a match at a World Cup?

WHERE (homePlayed.score > awayPlayed.score)
WITH collect(distinct home) as home
MATCH (losers:Country)
WHERE NOT losers IN home
RETURN (losers)

8 - What's the highest number of goals scored in a World Cup match?

MATCH(match:Match)
RETURN max(match.h_score + match.a_score) as max_goals

9 - Which stadium has hosted the most World Cup matches?

MATCH (stad:Stadium)<-[[:PLAYED_IN_STADIUM]]-(match:Match)
RETURN stad.name, count(match) as cant
ORDER BY cant DESC
LIMIT 1

10 - Which country has scored the most goals across all World Cups?

MATCH(c1:Country)-[r:PLAYED_IN]->(match:Match)
RETURN c1.name, sum(r.score) as sum_goals
ORDER BY sum_goals DESC
LIMIT 1

11 - Which country has participated in the most World Cups?

MATCH(c1:Country)-[r:NAMED_SQUAD]->(squad:Squad)-[:FOR_WORLD_CUP]->(wc:WorldCup)
RETURN c1.name, count(c1) as cant
ORDER BY cant DESC
LIMIT 1

12 - Which hosts won the World Cup that they hosted?

MATCH (match:Match {round: "Final"})<-[hostPlayed:PLAYED_IN]-(host:Country),
(host)<-[[:HOSTED_BY]]-(worldCup),
(worldCup)-[:CONTAINS_MATCH]->(match),
(match)<-[oppositionPlayed:PLAYED_IN]-(opposition)
WHERE (hostPlayed.score > oppositionPlayed.score) OR (hostPlayed.penalties > oppositionPlayed.score)
RETURN host.name
ORDER BY worldCup.year

13 - Which countries have never won a match at a World Cup?

MATCH (away:Country)-[awayPlayed:PLAYED_IN]->(match:Match)<-
[homePlayed:PLAYED_IN]-(home:Country)
WHERE (homePlayed.score > awayPlayed.score)
WITH collect(distinct home) as home
MATCH (losers:Country)
WHERE NOT losers IN home
RETURN (losers)

14 - What's the highest number of goals scored in a World Cup match?

MATCH(match:Match)
RETURN max(match.h_score + match.a_score) as max_goals

15 - Which stadium has hosted the most World Cup matches?

MATCH (stad:Stadium)<-[:PLAYED_IN_STADIUM]-(match:Match)
RETURN stad.name, count(match) as cant
ORDER BY cant DESC
LIMIT 1

16 - Which country has scored the most goals across all World Cups?

MATCH(c1:Country)-[r:PLAYED_IN]->(match:Match)
RETURN c1.name, sum(r.score) as sum_goals
ORDER BY sum_goals DESC
LIMIT 1

17 - Which country has participated in the most World Cups?

MATCH(c1:Country)-[r:NAMED_SQUAD]->(squad:Squad)-[:FOR_WORLD_CUP]->(wc:WorldCup)
RETURN c1.name, count(c1) as cant
ORDER BY