Class Participation for Week 12

Learning objectives:

1. Gain more understanding about graph DBMS and graph databases.
2. Practice Cypher queries.
3. Practice Extract-Transform-Load of data into a Neo4j database.

Submission:

Submit the zip file of the screenshot of the pseudo schema for Q3, Q3.cypher, and Q0.txt that has the number of answers you provided in your Plus/Delta survey submission. The rest of the questions are for your practice to prepare you for Homework 5.

Instruction:

1) Recall the Tuesday class participation on Week 10, which asks you to create a Neo4j graph database storing real-world entities and relationships using Person nodes, Location nodes, and Interest nodes and edges among them. Write Neo4j queries that answer the following questions.

   a) Find all friends of friends of Liam who posted the message “Happy Birthday”. Only show the name of the friends.

      Match (a:Person)-[:Friendship]→()→[:Friendship]→(b:person)-[p:Posted]→() where a.name=”Liam” and p.message=”Happy Birthday” return b.name

   b) List the names of all James’ friends who played "Soccer" in ascending order of their name.

      Match (a:Person)-[:Friendship]→(b:Person)-(:Plays)→(i:Interest) where a.name=”James” and I.game=”Soccer” return b.name order by b.name asc

   c) For each Person node, list the name of the Person node, and the number of friends of the person node in descending order of the number of friends.

      Match (a:Person)-[r:Friendship]→(:Person) with a.name as newname, count(r) as numFriends return newname, numFriends order by numFriends desc

   d) List the names of Person nodes without any friends (i.e., no one thinking of him/her as a friend). In other words, find the nodes without any incoming edges and list the values of the name attribute.

      Match (a:Person) where not (a)←[:Friendship]-(:Person) return a.name

2) Recall the Thursday class participation on Week 10, which asks you to create a Neo4j graph database and store the data exported from relations created per the ER diagram in Fig. 1.

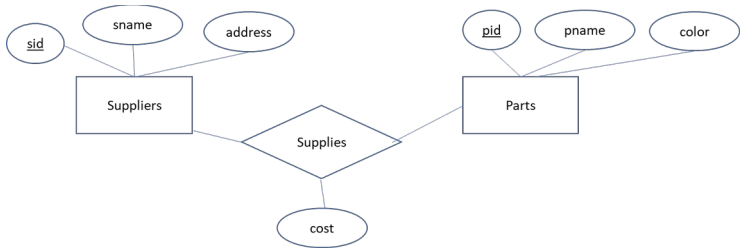


Fig. 1 ER diagram of a database of suppliers, parts, and suppliers supply parts.

Write Cypher statements to answer the following queries.

a) Add one new supplier with your name as the value of the sname attribute. Find the *snames* of suppliers who do not supply any part. Show the result in ascending order of *snames*. This query should return only the sname value that you just added.
match (:Suppliers)-[e:SUPPLIES]->(:Parts) return sname order by sname asc
b) Find unique *sids* and *snames* of suppliers who **supply a black part and a blue part**. Show the result in ascending order of *sids* values.
Match (:Parts{color:'blue'})<-[:SUPPLIES]-(:Suppliers)-[e:SUPPLIES]->(:Parts{color:'black'}) return distinct s.sid, s.sname order by s.sid
c) Find suppliers who supply exactly 40 parts. List sid and sname values.
match (s:Suppliers)-[e:SUPPLIES]→(:Parts) where count(*) =40 return s.sname, s.sid
d) Find the snames of the suppliers who supply **every green part**. Show the result in ascending order of *snames* values.
match (s:Suppliers)-[e:SUPPLIES]→(:Parts{color:'green'}) return s.sname

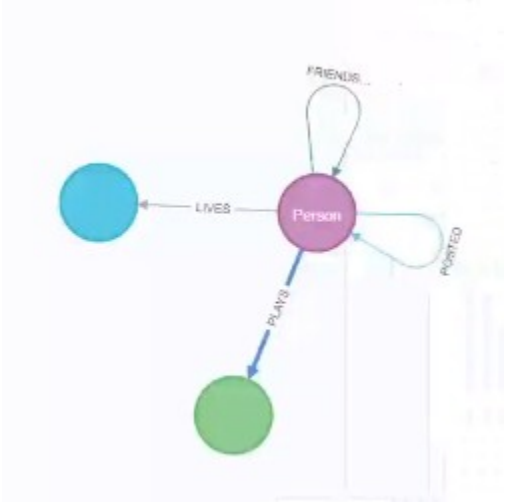
3) Prepare the database for Homework 5. See the instruction on how to create the tweetsdb database. After you execute LoadTweets.cypher per the instruction, execute the following statement.

```
call db.schema.visualization()
```

Configure your pseudo schema to show the node labels. Take the screenshot of the pseudo schema. Submit the screenshot for this class participation and for Homework 5 Question 1.

Run Neo4j statements to make sure that the values of the location attribute for the User nodes who lived in the same state is same. For instance, all users who lived in the state of Florida should have the same value for the location attribute as "Florida" instead of "Florida" or "FL".

Also submit Q3.cypher that contains these Neo4j statements.


FRIENDS
LIVES
Person
POSTED
PLAYS