Week 13 Class Participation

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.

Instruction: There is nothing to submit, but doing this exercise will help you solve the problems for the last homework.

1) Prepare the database for Homework 5 by running LoadTweets.cypher.
2) Like MySQL, Neo4j allows data import from a specific folder for each database. In this activity, you will import data from three csv files: parts.csv, suppliers.csv, and supplies.csv. To know which folder to put these files in, follow the video recording using the URL provided with the Canvas assignment page for Homework 5. Put the parts.csv, suppliers.csv, and supplies.csv in the import folder. These files were exported from a relational database that was created based on the ER diagram in Fig. 1.

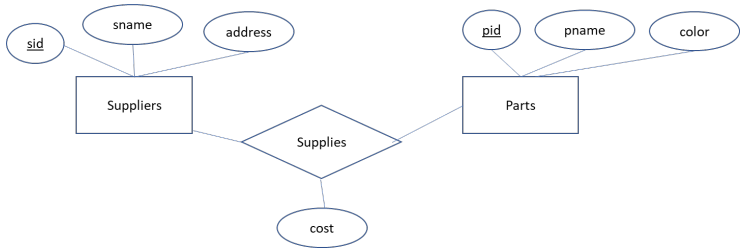

Fig. 1 ER diagram of a database of suppliers, parts, and relationships between suppliers and parts.

In this graph database design, we create a node label for each entity set. We also create an edge label for each two-way relationship set. For properties of an entity set, we make them properties of nodes with the label for that entity set. Similarly, for properties of a relationship set, we make them properties of edges having the edge label created for the relationship set.

Save LoadSupplierDBData.cypher in Favorites local script. Run the script. The script won't work if the older version of Neo4j's DBMS is used or the csv files are put in a wrong folder.

Run the following statement in Neo4j browser to get the pseudo schema of this database.

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

Write Cypher statements to answer the following queries.

a) Return the number of suppliers (number of nodes with the label "Suppliers").
```
match (s:Suppliers) return count(s) as numSuppliers;
```

b) Return the number of edges of the label "SUPPLIES".
```
match (:Suppliers)-[e:SUPPLIES]->(p:Parts) return count(e) as numSupplies;
```

c) Add one new supplier with your name as the value of sname.
```
create (n:Suppliers{sid:18,pname:'Pak'});
```

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 that you just added.
match (s:Suppliers) where not (s)-[:SUPPLIES]->(:Parts) return s.sname order by s.sname asc;

d) Find unique *sids* and *snames* of suppliers who **supply a black part and a blue part**. Show the result in ascending order of *sids*.

match(:Parts{color:'blue'})<-[:SUPPLIES]-(s:Suppliers)-[:SUPPLIES]->(:Parts{color:'black'})
return distinct s.sid, s.sname order by s.sid;

e) Find suppliers who supply exactly 40 parts. List sid and sname values.

match (p:Parts)<-[e:SUPPLIES]-(s:Suppliers)
with s, count(e) as cnt where cnt>=40 return s.sid, s.sname order by s.sid;

f) Find the snames of the suppliers who supply **every green part**. Show the result in ascending order of *snames*.

match (g:Parts{color:'green'}) with count(g) as cntallgreen
match (s:Suppliers)-[r:SUPPLIES]->(g:Parts{color:'green'})
with cntallgreen, s, count(r) as cntg
where cntallgreen=cntg return s.sname order by s.sname asc;