Applied Databases

Neo4j II Exercise Sheet - Solutions

Lab 7 Neo4j Relationships

Part 1

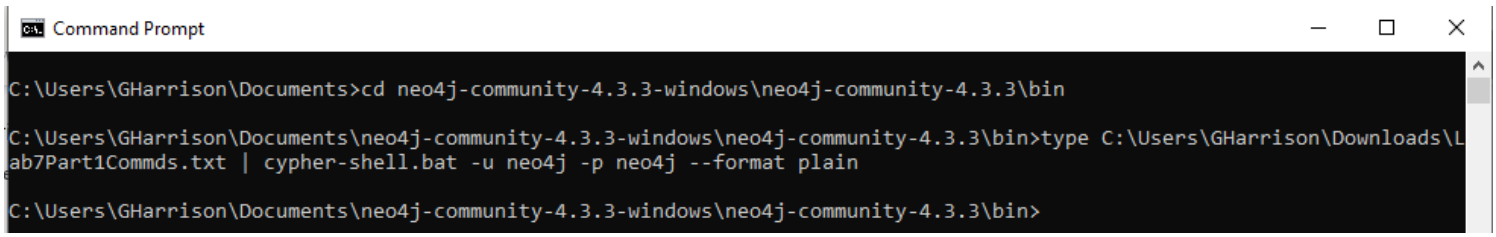
- Get *Lab7Part1Commds.txt* from and run the following command:

```
type Path_to_Lab7Part1Commands.txt | cypher-shell.bat -u neo4j -p neo4j --format plain
```

from the `bin` folder of your Neo4j installation.
E.g. Assuming:

- *Lab7Part1Commds.txt* was downloaded to `C:\Users\GHarrison\Downloads`
- Neo4j installation is at `C:\Users\GHarrison\Documents\neo4j-community-4.3.3-windows\neo4j-community-4.3.3\`
- Neo4j username is `neo4j`
- Neo4j password is `neo4j`

The following should be run, and if no errors are reported the database will be set up.
NOTE: This will delete everything from your current database (as specified in `neo4j.conf`).



- There should be 5 `COUNTY`, 11 `TOWN`, and 15 `PERSON` nodes in the database.

- Create the following relationships (some of which have a property called `countyTown`) between the nodes specified below:

TOWN	RELATIONSHIP	COUNTY
Galway	PART_OF {countyTown:true}	Galway
Tuam	PART_OF	Galway
Clifden	PART_OF	Galway
Carrick-on-Shannon	PART_OF {countyTown:true}	Leitrim
Manorhamilton	PART_OF	Leitrim
Castlebar	PART_OF {countyTown:true}	Mayo
Ballina	PART_OF	Mayo
Roscommon	PART_OF {countyTown:true}	Roscommon
Castlerea	PART_OF	Rocommon
Sligo	PART_OF {countyTown:true}	Sligo
Collooney	PART_OF	Sligo

```
match(t:TOWN{name:"Galway"})
match(c:COUNTY{name:"Galway"})
CREATE(t)-[:PART_OF{countyTown:true}]->(c);

match(t:TOWN{name:"Tuam"})
match(c:COUNTY{name:"Galway"})
CREATE(t)-[:PART_OF]->(c);

match(t:TOWN{name:"Clifden"})
match(c:COUNTY{name:"Galway"})
CREATE(t)-[:PART_OF]->(c);

match(t:TOWN{name:"Carrick-on-Shannon"})
match(c:COUNTY{name:"Leitrim"})
CREATE(t)-[:PART_OF{countyTown:true}]->(c);

match(t:TOWN{name:"Manorhamilton"})
match(c:COUNTY{name:"Leitrim"})
CREATE(t)-[:PART_OF]->(c);

match(t:TOWN{name:"Castlebar"})
match(c:COUNTY{name:"Mayo"})
CREATE(t)-[:PART_OF{countyTown:true}]->(c);

match(t:TOWN{name:"Ballina"})
match(c:COUNTY{name:"Mayo"})
CREATE(t)-[:PART_OF]->(c);
```

```
match(t:TOWN{name:"Roscommon"})
match(c:COUNTY{name:"Roscommon"})
CREATE(t)-[:PART_OF{countyTown:true}]->(c);

match(t:TOWN{name:"Castlerea"})
match(c:COUNTY{name:"Roscommon"})
CREATE(t)-[:PART_OF]->(c);

match(t:TOWN{name:"Sligo"})
match(c:COUNTY{name:"Sligo"})
CREATE(t)-[:PART_OF{countyTown:true}]->(c);

match(t:TOWN{name:"Collooney"})
match(c:COUNTY{name:"Sligo"})
CREATE(t)-[:PART_OF]->(c);
```

- Create the following relationships (some of which have a property called `since`) between the nodes specified below:

PERSON	RELATIONSHIP	TOWN
Tom	LIVES_IN	Galway
Sean	LIVES_IN since:2010	Galway
Bob	LIVES_IN	Galway
Mary	LIVES_IN since:2018	Clifden
Alice	LIVES_IN since:2010	Clifden
Pat	LIVES_IN since:1959	Carrick-on-Shannon
Alan	LIVES_IN	Carrick-on-Shannon
Bill	LIVES_IN	Manorhamilton
Yvonne	LIVES_IN	Castlebar
Walter	LIVES_IN	Ballina
Colin	LIVES_IN	Roscommon
Brendan	LIVES_IN since:2013	Castlerea
Susan	LIVES_IN	Castlerea
Lucy	LIVES_IN	Sligo
Michael	LIVES_IN	Sligo

```
match(p:PERSON{name:"Tom"})
match(t:TOWN{name:"Galway"})
create(p)-[:LIVES_IN]->(t);

match(p:PERSON{name:"Sean"})
match(t:TOWN{name:"Galway"})
create(p)-[:LIVES_IN{since:2010}]->(t);

match(p:PERSON{name:"Bob"})
match(t:TOWN{name:"Galway"})
create(p)-[:LIVES_IN]->(t);

match(p:PERSON{name:"Mary"})
match(t:TOWN{name:"Clifden"})
create(p)-[:LIVES_IN{since:2018}]->(t);

match(p:PERSON{name:"Alice"})
match(t:TOWN{name:"Clifden"})
create(p)-[:LIVES_IN{since:2010}]->(t);

match(p:PERSON{name:"Pat"})
match(t:TOWN{name:"Carrick-on-Shannon"})
create(p)-[:LIVES_IN{since:1959}]->(t);

match(p:PERSON{name:"Alan"})
match(t:TOWN{name:"Carrick-on-Shannon"})
create(p)-[:LIVES_IN]->(t);
```

```
match(p:PERSON{name:"Bill"})
match(t:TOWN{name:"Manorhamilton"})
create(p)-[:LIVES_IN]->(t);

match(p:PERSON{name:"Yvonne"})
match(t:TOWN{name:"Castlebar"})
create(p)-[:LIVES_IN]->(t);

match(p:PERSON{name:"Walter"})
match(t:TOWN{name:"Ballina"})
create(p)-[:LIVES_IN]->(t);

match(p:PERSON{name:"Colin"})
match(t:TOWN{name:"Roscommon"})
create(p)-[:LIVES_IN]->(t);

match(p:PERSON{name:"Brendan"})
match(t:TOWN{name:"Castlerea"})
create(p)-[:LIVES_IN{since:2013}]->(t);

match(p:PERSON{name:"Susan"})
match(t:TOWN{name:"Castlerea"})
create(p)-[:LIVES_IN]->(t);

match(p:PERSON{name:"Lucy"})
match(t:TOWN{name:"Sligo"})
create(p)-[:LIVES_IN]->(t);

match(p:PERSON{name:"Michael"})
match(t:TOWN{name:"Sligo"})
create(p)-[:LIVES_IN]->(t);
```

- Show the PERSONs who live in Galway TOWN.

```
match(t:TOWN{name:"Galway"})<-[l:LIVES_IN]-(p:PERSON) return p
```

or

```
match(p:PERSON)-[l:LIVES_IN]->(t:TOWN{name:"Galway"}) return p
```

or

```
match(:TOWN{name:"Galway"})<--(p:PERSON) return p
```

- Show the age of the oldest PERSONs who lives in Carrick-on-Shannon.

```
match(t:TOWN{name:"Carrick-on-Shannon"})<-[:LIVES_IN]-(p:PERSON)
return max(p.age)
```

or

```
match(p:PERSON)-[l:LIVES_IN]->(t:TOWN{name:"Carrick-on-Shannon"})
return max(p.age)
```

or

```
match(t:TOWN{name:"Carrick-on-Shannon"})<--(p:PERSON)
RETURN max(p.age)
```

- Show the average age of males who live in Roscommon COUNTY.

```
match(c:COUNTY{name:"Roscommon"})<-[:PART_OF]-(:TOWN)<-[:LIVES_IN]-
(p:PERSON)
where p.sex="M"
return avg(p.age)
```

or

```
match(p:PERSON)-[l:LIVES_IN]->(t:TOWN)-[po:PART_OF]-
>(c:COUNTY{name:"Roscommon"})
where p.sex="M"
return avg(p.age)
```

or

```
match(c:COUNTY)<-[:PART_OF]-(:TOWN)<-[:LIVES_IN]-(p:PERSON)
where c.name="Roscommon" and p.sex="M"
return avg(p.age)
```

- Show the number of males who live in Galway COUNTY.

```
match(c:COUNTY{name:"Galway"})<-[:PART_OF]-(t:TOWN)<-[:LIVES_IN]-
(p:PERSON{sex:"M"}) return count(p)
```

or

```
match(p:PERSON)-[l:LIVES_IN]->(t:TOWN)-[po:PART_OF]-
>(c:COUNTY{name:"Galway"})
where p.sex="M"
return count(p)
```

or

```
MATCH(c:COUNTY{name:"Galway"})<-[*2]-(p:PERSON) RETURN p
```

- Show the name and population of the COUNTY where Lucy lives.

```
match(p:PERSON{name:"Lucy"})-[:LIVES_IN]->(t:TOWN)-[:PART_OF]-
>(c:COUNTY)
return c.name,c.pop
```

or

```
match(p:PERSON{name:"Lucy"})-[:LIVES_IN]->(t:TOWN)-[:PART_OF]-
>(c:COUNTY)
return c.name, c.pop
```

or

```
MATCH (p:PERSON{name:"Lucy"})-[*2]->(c:COUNTY)
RETURN c.name, c.pop
```

- Show the COUNTY name , TOWN name and PERSON name where the person has lived in the town since the year 2010.

```
match(c:COUNTY)<-[:PART_OF]-(t:TOWN)<-[:LIVES_IN{since:2010}]-(p:PERSON)
return c.name, t.name, p.name
```

or

```
match(c:COUNTY)<-[]-(t:TOWN)<-[l:LIVES_IN]-(p:PERSON)
where l.since=2010
return c.name, t.name, p.name
```

- Show the COUNTY name and the TOWN name of all towns with a population of less than 5000.

```
match(c:COUNTY)<-[:PART_OF]-(t:TOWN)
where t.pop < 5000
return c.name, t.name
```

- For people living in towns since 2011 or later, show the person's name (as *Name*), how long they've been living in the town (as *Since*), and the name of the town (as *Town*), in chronological order.

```
MATCH(p:PERSON)-[l:LIVES_IN]->(t:TOWN)
WHERE l.since >= 2011
RETURN p.name, l.since, t.name
ORDER by l.since
```

- Show the total population of the towns in county Galway (as *County_Galway_Pop*),

```
MATCH(t:TOWN)-[:PART_OF]->(c:COUNTY{name:"Galway"})
WITH t as townsInGalway
RETURN sum(townsInGalway.pop) AS County_Galway_Pop
```

- Show the county name (as *County*), the towns in the county (as *Towns*) and the number of towns in the county (as *Num_Towns*).
  E.g.

"County"	"Towns"	"Num_Towns"
"Galway"	["Clifden","Tuam","Galway"]	3

```
MATCH(c:COUNTY)<-[:PART_OF]-(t:TOWN)
RETURN c.name, collect(t.name), count(t.name)
```

or

```
MATCH(c:COUNTY)<-[:PART_OF]-(t:TOWN)
RETURN c.name AS County,
       collect(t.name) AS Towns,
       count(t) AS Num_Towns
```

Part 2

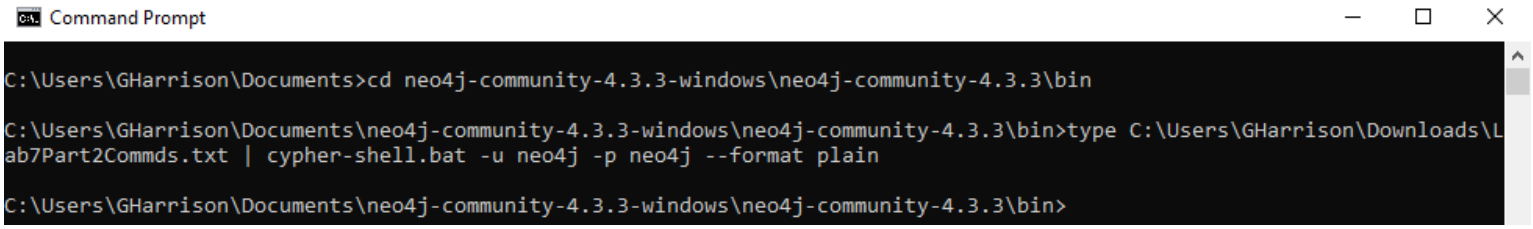
- Get *Lab7Part2Commds.txt* from and run the following command:

```
type Path_to_Lab7Part2Commands.txt | cypher-shell.bat -u neo4j -p neo4j --format plain
```

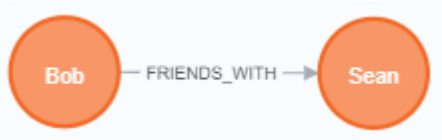
from the `bin` folder of your Neo4j installation.
E.g. Assuming:

- *Lab7Part2Commds.txt* was downloaded to `C:\Users\GHarrison\Downloads`
- Neo4j installation is at `C:\Users\GHarrison\Documents\neo4j-community-4.3.3-windows\neo4j-community-4.3.3\`
- Neo4j username is `neo4j`
- Neo4j password is `neo4j`

The following should be run, and if no errors are reported the database will be set up.
NOTE: This will delete everything from your current database (as specified in `neo4j.conf`).



NOTE: In this database, the FRIENDS_WITH relationship can be read in either direction. In this example, "Bob" is FRIENDS_WITH "Sean", however it can be taken that "Sean" is also FRIENDS_WITH "Bob".



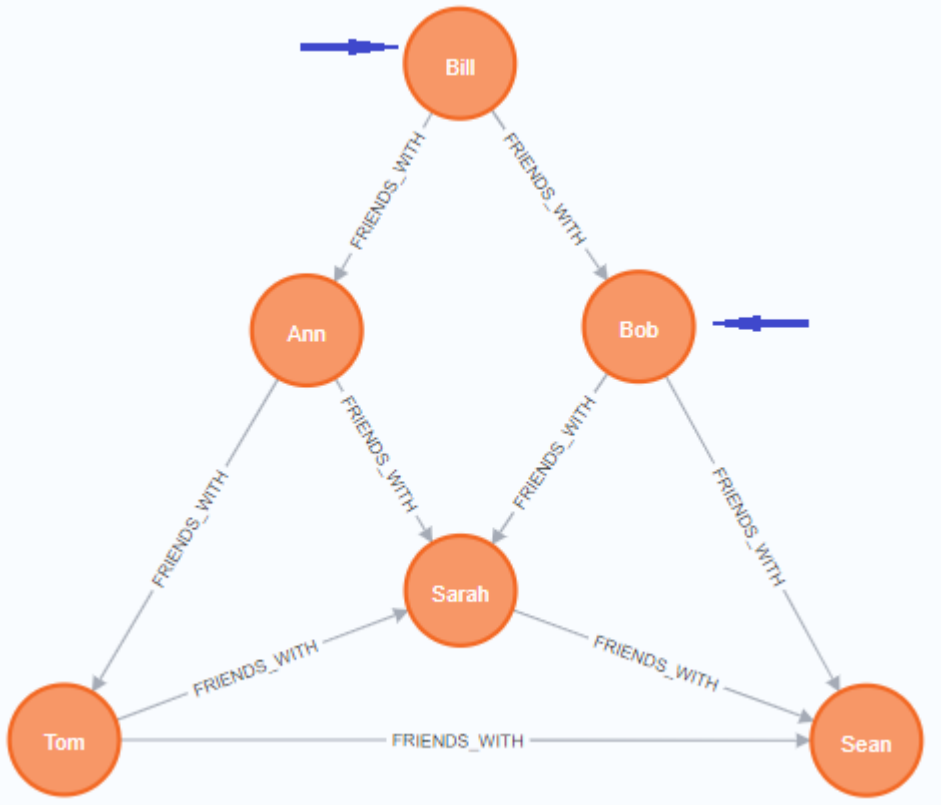
- Show the names of Bill's hobbies.

```
MATCH(p:Person{name:"Bill"})-[:LIKES]->(h:Hobby) RETURN h.name
```

- Show the names of hobbies people who live in Galway like (as *Galway_Hobbies*) in alphabetical order.

```
MATCH(c:County{name:"Galway"})-[*2]-(h:Hobby)
RETURN h.name AS Galway_Hobbies ORDER BY Galway_Hobbies
```

- Show all friends-of-friends of Tom.
  A friend-of-a-friend (FOAF) is someone whom your friend is friends with, but not you.
  In the example below, Bill and Bob are FOAFs of Tom.
  Ann is a friend of Tom, and her friend is Sarah. So, Sarah would be a FOAF of Tom, but as Tom is already friends with her, she's not FOAF of Tom. Similarly, for Ann and Sean.



Try 1: Returns same node multiple times.

```
match(n:Person{name:"Tom"})-[:FRIENDS_WITH*2]-(p) RETURN p.name
```

Try 2: Returns friends as well as FOAFs.

```
match(n:Person{name:"Tom"})-[:FRIENDS_WITH*2]-(p)
RETURN DISTINCT p.name
```

Try 3: Correct.

```
match(n:Person{name:"Tom"})-[:FRIENDS_WITH*2]-(p)
WHERE NOT exists((n)-[:FRIENDS_WITH]-(p))
RETURN DISTINCT p.name
```

- Show the unique hobbies that people who live in Westmeath like (as *Westmeath_Hobbies*).

Try 1: Get people from Westmeath.

```
MATCH(c:County{name:"Westmeath"})<-[:LIVES_IN]-(p:Person) RETURN p
```

Try 2: Get people from Westmeath's hobbies.

```
MATCH(c:County{name:"Westmeath"})<-[:LIVES_IN]-(p:Person)
WITH p AS whPerson
MATCH(whPerson)-[:LIKES]->(h:Hobby) RETURN h.name
```

Try 3: Correct.

```
MATCH(c:County{name:"Westmeath"})<-[:LIVES_IN]-(p:Person)
WITH p AS whPerson
MATCH(whPerson)-[:LIKES]->(h:Hobby)
RETURN DISTINCT h.name as Westmeath_Hobbies
```

- Show the number of people who like relaxation hobbies (as *Relaxation*).

Try 1: Returns Sean twice, as he LIKES 2 relaxation hobbies.

```
MATCH(h:Hobby{type:"relaxation"})<-[:LIKES]-(p:Person) RETURN p
```

Try 2: Returns DISTINCT nodes.

```
MATCH(h:Hobby{type:"relaxation"})<-[:LIKES]-(p:Person)
RETURN DISTINCT p
```

Try 3: Correct.

```
MATCH(h:Hobby{type:"relaxation"})<-[:LIKES]-(p:Person)
RETURN count(DISTINCT p) AS Relaxation
```

- Show a heading called *Likes_Basketball* that returns true if Sarah LIKES basketball, or false if Sarah doesn't like basketball.

```
match (p:Person{name:"Sarah"})
match (h:Hobby{name:"Basketball"})
RETURN exists((p)-[:LIKES]->(h)) AS Likes_Basketball
```