Projecte Padrons

Assignatura:BdnR

Pau Clavera i Jordi Brota

Exercici 1:

Aquest exercici l'ha fer l'alumne Pau Clavera amb NIU 1568315.

Primer creem una constraint per tal de no tenir nodes Habitatge duplicats, on la combinació de Id_Llar, Municipi i Any de Padró sigui unica.

CREATE CONSTRAINT UnicaLlar ON (h:Habitatge) ASSERT (h.Id_Llar,h.Municipi,h.Any_Padro) IS NODE KEY;

Després creem la constraint per a tal de no tenir nodes Persona duplicats a la Base de Dades on el Id serà l'identificador unic de cada node.

CREATE CONSTRAINT UnicaPersona ON (p:Persona) ASSERT p.Id IS UNIQUE;

Seguidament comencem a processar l'arxiu 'Habitatges.csv' per tal de guardar en la base de dades tots els nodes habitatge que estan registrats en aquest arxiu. Primer fem un load csv per carregar les dades al programa, després fem un with per separar tota la informació dels camps que ens interessa, a continuació eliminem la informació que inclogui valors nulll amb la comanda where, llavors creem els nodes Habitatge afegint-li els atributs amb la comanda merge i la comanda set.

LOAD CSV FROM 'file:///Habitatges.csv' AS Habitatge

WITH toInteger(Habitatge[1]) AS Id_Llar, Habitatge[0] AS Municipi, toInteger(Habitatge[2]) AS Any_Padro, Habitatge[3] AS Carrer, toInteger(Habitatge[4]) AS Numero

WHERE Id_Llar IS NOT null AND Municipi is not null AND Any_Padro is not NULL AND Carrer IS NOT NULL AND Numero IS NOT NULL

MERGE (h:Habitatge {Id_Llar: Id_Llar, Carrer:Carrer, Any_Padro:Any_Padro, Municipi:Municipi, Numero:Numero})

SET h.Id_Llar = Id_Llar, h.Carrer = Carrer, h.Any_Padro = Any_Padro, h.Municipi = Municipi, h.Numero = Numero;

A continuació carreguem el fitxer "Individual.csv" que conté la informació de cada node Persona. Igual que en el fitxer habitatge, utilitzem el with per separar per columnes i el where per comprovar els valors null que no desitgem. Després utilitzem el merge i el Set per crear els nodes.

LOAD CSV FROM 'file:///Individual.csv' AS Persona

WITH toInteger(Persona[0]) AS Id, toInteger (Persona[1]) AS Year, Persona[2] AS Name, Persona[3] AS Surname, Persona[4] AS Second_Surname

WHERE Id IS NOT null

MERGE
(p:Persona{Id:Id,Year:Year,Name:Name,Surname:Surname,Second_Surname:Second_Surname})

SET p.Id = Id, p.Year = Year, p.Name = Name, p.Surname = Surname, p.Second_Surname = Second_Surname;

Després processem el fitxer 'FAMILIA.csv' que inclou la informació sobre les relacions entre nodes Persona. Primer carreguem les dades al programa, seguidament amb la comanda with ens quedem amb les que ens interessen, en aquest cas ID de la persona, relació harmonitzada i ID de l'altre persona. Fem un Merge per associar les id del arxiu amb els nodes ja existents i fiinalment creem relació Familia entre els nodes amb la comanda create.

LOAD CSV FROM 'file:///FAMILIA.csv' AS Familia

WITH toInteger(Familia[0]) AS ID_1, Familia[1] AS Relacio, Familia[2] AS Relacio_Harmonitzada, toInteger(Familia[3]) AS ID_2

WHERE ID_1 IS NOT NULL AND ID_2 IS NOT NULL

MERGE (f1:Persona{Id:ID_1})

MERGE (f2:Persona{Id:ID_2})

CREATE (f1)-[:Familia {familiar:Relacio_Harmonitzada}]->(f2);

A continuació carreguem el arxiu VIU.csv que ens dona la informació de relació entre nodes Persona i nodes Habitatge. Primer carreguem les dades, després en quedem amb el ID de la persona, el municipi, l'any d'empadronament i l'ID de la casa. Utilitzem el merge una altre vegada per associar els nodes amb les dades que hem carregat i finalment creem relacions entre els nodes que correspongui amb la comanda create.

LOAD CSV FROM 'file:///VIU.csv' AS VIU

WITH toInteger(VIU[0]) AS IND, VIU[2] AS Location, toInteger(VIU[3]) AS Year, toInteger(VIU[4]) AS HOUSE_ID

WHERE IND IS NOT NULL AND Location IS NOT NULL AND Year IS NOT NULL AND HOUSE_ID IS NOT NULL

MERGE (hab:Habitatge{Municipi:Location,Id_Llar:HOUSE_ID,Any_Padro:Year})

MERGE (per:Persona{Id:IND})

CREATE (per)-[:VIU]->(hab);

I per ultim carreguem l'arxiu 'same_as.csv' que ens dona la informació dels nodes amb ID diferent que són la mateixa persona. Primer carreguem el fitxer, a continuació ens quedem amb els ID de cada una de les persones i finalment creem un edge same_as entre aquests dos nodes i fem que es retorni tota la base de dades.

```
LOAD CSV FROM 'file:///same_as.csv' AS SAME_AS

WITH toInteger(SAME_AS[0]) AS ID1, toInteger(SAME_AS[2]) AS ID2

WHERE ID1 IS NOT NULL AND ID2 IS NOT NULL

MERGE (p1:Persona{Id:ID1})

MERGE (p2:Persona{Id:ID2})

CREATE (p1)-[:SAME_AS]->(p2)

RETURN *;
```

Exercici 2:

Aquest exercici l'ha fet l'alumne Jordi Brota amb NIU 1563359.

2.1- Primer seleccionem totes les persones que viuen a Castellví de Rosanes el qual l'any de l'empadronament és el 1866, seguidament eliminem aquelles persones les quals el seu nom és 'nan' o NULL. Després, utilitzar el return per retornar la llista de noms diferents amb collect i distinct i el número de persones del municipi.

```
MATCH (p:Persona)-[:VIU]->(h:Habitatge{Municipi:'CR',Any_Padro:1866})
WHERE NOT p.Name ='nan' AND p.Name IS NOT NULL
RETURN COUNT(p) AS Numero_De_Persones,collect(DISTINCT p.Name) as Noms
```

2.2- Seleccionem els habitatges que tinguin com a Municipi SFLL (Sant Feliu de Llobregat) i els quals l'any de Padró sigui menor a 1840. Seguidament retornem el municipi, l'any de padró, i la llista de llars i ordenem per any_padro.

```
MATCH (h:Habitatge{Municipi:'SFLL'})
WHERE h.Any_Padro < 1840
RETURN h.Municipi,h.Any_Padro,collect(h.Id_Llar)
ORDER BY h.Any_Padro
```

2.3- Primer seleccionem les persones que viuen a Sant Feliu de Llobregat i amb any de padró 1838, i amb la mateixa consulta de match agafem només aquells que visquin a la mateixa llar que Rafael Marti. Després simplement retornem la llista amb els noms.

```
MATCH (p:Persona)-[:VIU]->(:Habitatge{Any_Padro:1838,Municipi:'SFLL'})<-[:VIU]-(:Persona{Name:'rafel',Surname:'marti'})
RETURN collect(p.Name)
```

```
MATCH (p:Persona)-[:VIU]->(:Habitatge{Any_Padro:1838,Municipi:'SFLL'})<-[:VIU]-(:Persona{Name:'rafel',Surname:'marti'})
RETURN p
```

2.4- Aquí busquem amb la relacio de same_as tots els nodes que tinguin aquesta relació entre ells, i després amb el where associem un dels noms amb Miguel Ballester. Simplement després retornem totes les persones que compleixin el match.

```
MATCH (p:Persona)-[:SAME_AS]-(p2:Persona)
WHERE toLower(p2.Name) = 'miguel' and toLower(p2.Surname) = 'ballester'
RETURN p,p2
```

2.5- Primer fem match amb totes aquelles persones amb relació família amb antonio farran, ja que no volem la mateixa persona, només la resta relacionada. Després posem la condició de que les persones siguin diferents a Antonio Farran per eliminar nodes que puguin coincidir i finalment retornem els noms i cognoms i el tipus de relació familiar en forma de taula.

```
MATCH (p1:Persona)-[r:Familia]-(p2:Persona{Name:'antonio',Surname:'farran'})
WHERE p1 <> p2
RETURN p1.Name,p1.Surname,p1.Second_Surname,r.familiar
```

2.6- Fem match de les relacions familiars amb la variable f i després retornem en forma de llista els diferents valors de f que hem obtingut.

```
MATCH (:Persona)-[f:Familia]-(:Persona)
RETURN COLLECT(DISTINCT f.familiar)
```

2.7- Primer fem match amb els habitatges de Sant Feliu de Llobregat i els quals el carrer y el numero no siguin null, seguidament retornem el carrer, el numero, el count de habitatges, la llista d'anys de padró i la llista amb els ID de les llars. Gràcies a la funció count s'agrupa tot automàticament amb el resultat desitjat.

```
MATCH (h1:Habitatge{Municipi:'SFLL'})
WHERE h1.Carrer IS NOT NULL AND h1.Numero IS NOT NULL
RETURN (h1.Carrer)as Carrer,h1.Numero as Numero, COUNT(h1) AS NumeroSimilars, collect(h1.Any_Padro) as Llista_anys,collect(h1.Id_Llar) as Llista_ID
ORDER BY NumeroSimilars DESC
LIMIT 10
```

2.8- Primer busquem aquells nodes que tinguin relacions de família i que visquin a Castellví de Rosanes, on la relació familiar sigui de tipus fill o de tipus filla. Després utilitzem with per agrupar el número de fills i retornem amb p2 i seguidament posem la condició de que el número de fills sigui major a 3. Utilitzem el with per poder fer la comparació, ja que el count només es pot fer servir quan el precedeix una comanda tipus return. Finalment retornem el número de fills, el nom del cap de família i els cognoms i ordenem per el número de fills de major a menor.

```
MATCH (p2:Persona)-[f:Familia]->(p1:Persona)-[:VIU]->(h:Habitatge{Municipi:'CR'})
WHERE f.familiar = 'fill' OR f.familiar = 'filla'
WITH COUNT(f) as num_fills,p2
WHERE num_fills >3
RETURN num_fills,p2.Name,p2.Surname,p2.Second_Surname
ORDER BY (num_fills) DESC
LIMIT 20
```

2.9- Comencem fent match amb aquelles persones que tinguin relació família com a l'exercici anterior pero que visquin a Sant Feliu de Llobregat i que l'any de Padró sigui el 1881. Després com a la consulta anterior, seleccionem aquells nodes on la relació familiar sigui de fills o filles i apliquem el with amb la mateixa raó. Finalment tornem a buscar els nodes que ens interessen i retornem la suma de fills, el número d'habitatges i la mitja de fills amb la funció avg().

```
MATCH (p2:Persona)-[f:Familia]->(p1:Persona)-[:VIU]->(h:Habitatge{Municipi:'SFLL',Any_Padro:1881})
WHERE f.familiar = 'fill' or f.familiar = 'filla'
WITH COUNT(f) as num_fills,p2
MATCH (h:Habitatge{Municipi:'SFLL',Any_Padro:1881})
RETURN sum(num_fills) as NumFills,count(h) as NumHabitatges, avg(num_fills) as MitjaFills
```

2.10- Primer fem match per seleccionar les persones que visquin a Sant Feliu de Llobregat. Després llistem els anys de padró a llista_anys i fem un unwind per seleccionar any a any el número d'habitants i el carrer. Utilitzem el with per quedar se amb el número d'habitants, el carrer, i l'any i ordenem per número d'habitants per agafar els menors. Finalment creem una llista amb la combinació número d'habitants i nom carrer com una llista i l'any i dividim la llista per retornar el carrer i l'any.

```
MATCH (p:Persona)-[:VIU]->(h:Habitatge{Municipi:'SFLL'})
WITH collect(DISTINCT p.Year)as llista_anys
UNWIND llista_anys as Year
MATCH (p:Persona)-[:VIU]->(h:Habitatge{Municipi:'SFLL',Any_Padro:Year})
WITH count(p) as numHab, h.Carrer as NomCarrer,Year
ORDER BY numHab
WITH collect(numHab+','+NomCarrer) as llista,Year
WITH split(llista[0],',') as nom,Year
RETURN nom[1] as Carrer,Year
ORDER BY Year
```

Exercici 3:

Aquest exercici l'hem fet entre els alumnes Pau Clavera i Jordi Brota amb NIUs 1568315 i 1563359

3.a-

Comencem creant un graf per a analitzar de la següent manera:

```
CALL gds.graph.create.cypher(
    'Padrons',
    'MATCH (n) WHERE n:Persona OR n:Habitatge RETURN id(n) as id, labels(n) as labels',
    'MATCH (n)-[e]-(m) RETURN id(n) AS source, e.weight AS weight, id(m) AS target'
)
```

Si anem analitzant el graf utilitzant aquesta consulta:

```
call gds.wcc.stream( 'Padrons')
yield componentId, nodeId
with componentId, collect(nodeId) as nodes,
size(collect(nodeId)) as mida
order by mida DESC
match(n)
where id(n) in nodes AND mida =NUMBER
return n;
```

Quan canviem el numero "NUMBER" del codi, veurem components connexes del graf del tamany que escollim i podem començar a veure l'estructura d'aquests components en funció de la mida d'aquesta. D'aquí podem començar a veure que l'estructura quan la mida és petita (és a dir quan el número de components connectats es petit) conté només un node de Habitatge i la resta son nodes de persones en familia. Conforme va creixent el tamany van apareixent relacions SAME_AS i algun habitatge. Això té sentit ja que per cada habitatge tenim varies persones convivent.

Si introduïm la consulta:

call gds.wcc.stream('Padrons')
yield componentId, nodeId
with componentId, collect(nodeId) as nodes,
size(collect(nodeId)) as mida
order by mida DESC
match(m:Habitatge)
where id(m) in nodes
RETURN count(m) as numHab,mida,componentId;

Seguit de la consulta

call gds.wcc.stream('Padrons')
yield componentId, nodeId
with componentId, collect(nodeId) as nodes,
size(collect(nodeId)) as mida
order by mida DESC
match(m:Persona)
where id(m) in nodes
RETURN count(m) as numPersones,mida,componentId;

Podem comparar la diferencia de proporció entre els nodes persona i nodes habitatge i comprovem que efectivament, la proporció entre nodes habitatge i nodes persona cada vegada tendeix a que hi hagi més nodes habitatge per persona, ja que al principi hi han molts components connexos sense node habitatge pero conforme més gran es la cc, més habitatges veiem per persona

Veiem que la mitja de components connexos es de 5.7 de tal manera que entenem que l'estructura que segueixen aquestes cc és de famílies agrupades en un habitatge principalment

A més a més, amb aquesta consulta veiem que hi han molts components connexes on només hi ha persones:

call gds.wcc.stream('Padrons')
yield componentId, nodeId
with componentId, collect(nodeId) as nodes,
size(collect(nodeId)) as mida
order by mida DESC
match(m:Persona)
where id(m) in nodes AND NOT (m)-[:VIU]->()
RETURN count(componentId) as numcomponents;

On veiem que hi ha 4741. Si a més modifiquem una mica la consulta per veure també la quantitat de nodes així en funció de la mida de les components connexes (afegint

mida al return i fent que la mida <> 7613) veiem que hi han molts nodes sols sense ningun veí i també observem que per a cc petits, hi ha moltíssims més cc sense habitatge que no amb cc grans. Per exemple hi ha 84 cc de diferencia entre la mida de cc de 134 nodes i la de 4 nodes. La conclusió és que l'estructura amb mides petites es de un o cap node habitatge i tota la resta amb node Persona, i conforme el tamany de la cc creix, cada vegada veiem més habitatges com és normal, ja que a més quantitat de persones necessites més habitatges però tenint en compte que sempre hi haurà moltes més Persones que Habitatges.

3.b- Comencem determinant els nodes habitatges que representen al mateix habitatge i creem una relació entre ells anomenada MATEIX_HAB amb la següent consulta:

```
MATCH (h1:Habitatge),(h2:Habitatge)
WHERE h1.Carrer = h2.Carrer AND h1.Numero = h2.Numero AND h1.Municipi = h2.Municipi
AND h1.Any_Padro > h2.Any_Padro
CREATE (h1)-[:MATEIX_HAB]->(h2)
RETURN h1,h2
```

A partir d'aquí creem un subgraf de la base de dades que contingui tota la informació excepte les relacions de tipus SAME_AS amb la següent comanda:

```
CALL gds.graph.create.cypher(
    'padrons3b',
    'MATCH (n) WHERE n:Persona OR n:Habitatge RETURN id(n) as id, labels(n) as labels',
    'MATCH (n)-[e]-(m) WHERE NOT e:SAME_AS RETURN id(n) AS source, e.weight AS
weight, id(m) AS target'
)
```

I finalment busquem la similaritat entre els nodes del nostre graf creat separant els node labels de persona i de habitatge:

```
CALL gds.nodeSimilarity.stream('padrons3b',{nodeLabels:['Habitatge']})

YIELD
  node1,
  node2,
  similarity

CALL gds.nodeSimilarity.stream('padrons3b',{nodeLabels:['Persona']})

YIELD
  node1,
  node2,
  similarity
```

Al calcular el node similarity amb els habitatges veiem molst habitatges amb una similitut semblant i veiem com els resultats son diversos. Als nodes, en canvi, ens surt que tots son el mateix node (similitut de 1.0) i això pot ser degut a la gran interconnexio de tots els nodes per la relacio familia. Creant per tant que tots els nodes connexes siguin molt similars entre ells. Com els habitatges son menys, i menys interconnectats, no surten els resultats tant exagerats.

Treball en equip:

El treball l'hem fet només Jordi Brota Tricaz i Pau Clavera Comas. Això es deu a que quan vam començar el projecte i la repartició de tasques, les altres persones del grup no van contestar a ninguna dels nostres missatges ni ho han fet encara a dia d'avui (sabent bé que han rebut els missatges i els han llegit). No sabem la situació de cara a la carrera d'aquestes persones així que ens vam posar a treballar sols. El treball encara que vam repartir tasques ha estat força conjunt, ja que una part positiva de ser només dos es que era més fàcil coordinar-nos. En cada exercici hem especificat qui ha sigut la persona encarregada principalment de l'exercici tot i que el feedback i comunicació ha estat constant durant tot el projecte. A nivell de les persones que hem fet el treball, la càrrega ha estat més intensa del que ens hagués agradat pero per sort nosaltres dos hem sapigut treballar en equip. Això si, no sabiem utilitzar massa Github i no sabem exactament si el procediment a l'hora de penjar i compartir els arxius ha sigut 100% correcte.

El repositori de Github és: https://github.com/PauClaveraComas/Neo4jPadrons