

Projecte de Neo4j: Padrons

Exercici 1. Importació de les dades

L'script que hem utilitzat per a la importació de les dades és el següent:

Neteja de la base de dades

```
MATCH (n)
DETACH DELETE n;
```

En primer lloc, per a realitzar una importació correcta de les dades i assegurar-nos que no obtenim duplicats a l'executar l'script dues vegades, el que fem es netejar la base de dades. Per fer-ho, seleccionem tots els nodes de la base de dades (MATCH (n)) i els eliminem juntament amb totes les relacions existents associades als nodes (DETACH DELETE n)

Creació de les restriccions

```
CREATE CONSTRAINT UNIQUE_ID IF NOT EXISTS FOR (i:INDIVIDUAL) REQUIRE i.Id is
UNIQUE;
CREATE CONSTRAINT ID_INDIVIDUAL IF NOT EXISTS FOR (i:INDIVIDUAL) REQUIRE
i.Id is NOT NULL;
CREATE CONSTRAINT ID_HABITATGE IF NOT EXISTS FOR (h:HABITATGE) REQUIRE
h.Id_Habitatge is NOT NULL;
```

Pel que fa a les restriccions (CREATE CONSTRAINT), n'hem creat tres: la primera restricció per garantir que l'atribut ID dels nodes de tipus INDIVIDUAL sigui únic. Les altres dues restriccions creades fan referència al identificador (Id) dels nodes INDIVIDU (Id) i HABITATGE (Id_Habitatge), aquestes restriccions ens asseguruen que l'atribut ID no sigui nul tant per als nodes INDIVIDUAL com pels nodes HABITATGE.

Creació dels índexs

```
CREATE INDEX AnyPadroIndex IF NOT EXISTS FOR ()-[r:VIU]-() ON (r.Year);
CREATE INDEX RelacioFamilia IF NOT EXISTS FOR ()-[r:FAMILIA]-() ON
(r.Relacio_Harmonitzada);
```

Per a poder realitzar consultes de forma més eficient creem índexs mitjançant CREATE INDEX per a l'atribut Any (YEAR) en les relacions del tipus VIU, i l'atribut Relacio_Harmonitzada en les relacions del tipus FAMILIA.

Creació dels nodes de tipus HABITATGE

```
LOAD CSV WITH HEADERS FROM 'file:///HABITATGES.csv' AS row
WITH row WHERE row.Municipi <> 'null' AND row.Id_Llar <> 'null'
MERGE (h:HABITATGE {Municipi: row.Municipi, Id_Habitatge: row.Id_Llar, Any_Padro:
toInteger(row.Any_Padro), Carrer: row.Carrer,
Numero: CASE WHEN row.Numero <> "null" THEN toInteger(row.Numero) ELSE
row.Numero END});
```

Utilitzem la comanda LOAD CSV WITH HEADERS per carregar dades des de l'arxiu csv HABITATGES.csv i filtrem les files on el camp Municipi o el camp Id_Llar sigui null. Utilitzem

la clàusula MERGE per a la creació dels nodes de tipus HABITATGE amb els atributs Municipi, Id_Habitatge, Any_Padro, Carrer i Numero (int, sempre que hi hagi número del carrer)

Creació dels nodes de tipus INDIVIDUAL

```
LOAD CSV WITH HEADERS FROM 'file:///INDIVIDUAL.csv' AS row
WITH row WHERE row.Id <> "null"
MERGE (i:INDIVIDUAL {Id: row.Id})
SET i.Year = toInteger(row.Year), i.name = row.name, i.surname = row.surname,
i.second_surname = row.second_surname;
```

Utilitzem la comanda LOAD CSV WITH HEADERS per carregar dades des de l'arxiu csv INDIVIDUAL.csv i filtrem les files on el camp Id sigui nul. Utilitzem la clàusula MERGE per a la creació dels nodes de tipus INDIVIDUAL amb l'atribut Id. Usem la clàusula SET per definir els atributs Year (tipus enter), name, surname i second_surname per al nom i cognoms de cada individu i l'any de naixement.

Creació de la relació VIU

```
LOAD CSV WITH HEADERS FROM 'file:///VIU.csv' AS row
WITH row WHERE row.IND <> 'null' AND row.HOUSE_ID <> 'null'
// WITH row.IND AS IND, toInteger(row.HOUSE_ID) AS HOUSE_ID, row.Location AS
Location, toInteger(row.Year) AS Year
MATCH (i:INDIVIDUAL), (h:HABITATGE)
WHERE row.IND = i.Id AND toInteger(row.HOUSE_ID) = toInteger(h.Id_Habitatge) AND
toInteger(row.Year) = toInteger(h.Any_Padro)
MERGE (i)-[:VIU {Location: row.Location, Year: row.Year}]->(h);
```

Utilitzem la comanda LOAD CSV WITH HEADERS per carregar dades des de l'arxiu csv VIU.csv i filtrem les files on els camps IND i HOUSE_ID siguin nuls. Utilitzem la clàusula MATCH per trobar tots els nodes de tipus INDIVIDUAL i HABITATGE que compleixin amb els criteris especificats per a l'Id de l'individu, l'Id de l'habitatge i l'any de padró. Finalment utilitzem MERGE per a crear la relació VIU entre els nodes INDIVIDUAL i HABITATGE amb els atributs Location i Year.

Creació de la relació SAME_AS

```
LOAD CSV WITH HEADERS FROM 'file:///SAME_AS.csv' AS row
WITH row WHERE row.Id_A <> "null" AND row.Id_B <> "null"
MATCH (p:INDIVIDUAL {Id: row.Id_A}), (s:INDIVIDUAL {Id: row.Id_B})
MERGE (p)-[:SAME_AS]->(s);
```

Fem servir la comanda LOAD CSV WITH HEADERS per carregar dades des de l'arxiu csv SAME_AS.csv i filtrem les files on els camps Id_A i Id_B siguin nuls. Utilitzem la clàusula MATCH per trobar tots els nodes de tipus INDIVIDUAL i HABITATGE que compleixin amb els criteris especificats per a l'Id de cada l'individu (Id_A i Id_B). Amb la clàusula MERGE creem la relació SAME_AS entre els nodes INDIVIDUAL i HABITATGE.

Creació de la relació FAMILIA

```
LOAD CSV WITH HEADERS FROM 'file:///FAMILIA.csv' AS row
WITH row WHERE row.ID_1 <> "null" AND row.ID_2 <> "null"
MATCH (p:INDIVIDUAL), (s:INDIVIDUAL)
WHERE row.ID_1 = p.Id AND row.ID_2 = s.Id
CREATE (p)-[:FAMILIA {Relacio: row.Relacio, Relacio_Harmonitzada:
row.Relacio_Harmonitzada}]->(s);
```

Useu la comanda LOAD CSV WITH HEADERS per carregar dades des de l'arxiu csv FAMILIA.csv i filtreu les files on els camps Id_1 i Id_2 siguin nuls. Utilitzem la clàusula MATCH per trobar tots els nodes de tipus INDIVIDUAL que compleixin amb els criteris especificats per a l'Id de cada individu (Id_1 i Id_2). Amb la clàusula CREATE creem la relació FAMILIA entre nodes de tipus INDIVIDUAL amb els atributs Relacio i Relacio_Harmonitzada.

Exercici 2. Consultes Cypher

a) Per a cada padró (any) de Sant Feliu de Llobregat (SFLL), retorna l'any de padró, el número d'habitants, i la llista de cognoms. Elimina duplicats i "nan".

Utilitzem el WHERE per treure els habitatges nuls i els que no siguin de Sant Feliu de Llobregat, després declarem primer l'any del padró per a que sigui la clau del group, després afegim un count dels habitants relacionats a un habitatge d'aquell any i finalment la llista de cognoms diferents amb un DISTINCT + COLLECT.

```
MATCH (h:HABITATGE)-[:VIU]-(i:INDIVIDUAL)
WHERE h.Municipi = 'SFLL' AND NOT i.surname = 'nan' AND NOT i.surname IS NULL
RETURN h.Any_Padro as any_padro, count(i) as habitants, collect(DISTINCT i.surname) as
diferent_cognoms
```

any_padro	habitants	diferent_cognoms
1881	2999	["esplugas", "guiu", "bigas", "serralabos", "subirats", "angles",
1889	3109	["amat", "dordal", "rius", "navines", "majo", "aldrofen", "planas
1878	2739	["iglesias", "duran", "llanas", "ribas", "estape", "majo", "bausili
1833	345	["roig", "aloma", "guiu", "casas", "coca", "majo", "parellada",
1839	492	["ribas", "illegible", "roig", "parera", "camprubi", "aloma", "majo
1838	71	["julia", "vidal", "valles", "rovira", "campderros", "carcereny",

b) Retorna totes les aparicions de "miguel estape bofill". Fes servir la relació SAME_AS per poder retornar totes les instàncies, independentment de si hi ha variacions lèxiques (ex. diferents formes d'escriure el seu nom/cognoms). Mostra la informació en forma de taula: el nom, la llista de cognoms i la llista de segon cognom (elimina duplicats).

Després de declarar que estem mirant la relació SAME_AS, escollim l'origen de la relació per obtenir totes les variacions, després només hem de fer un DISTINCT + COLLECT.

```
MATCH (a:INDIVIDUAL)-[s:SAME_AS]-(i:INDIVIDUAL)
WHERE (i.name = 'miguel' AND i.surname = 'estape' AND i.second_surname = 'bofill') OR
(a.name = 'miguel' AND a.surname = 'estape' AND a.second_surname = 'bofill')
RETURN i.name as nom, collect(DISTINCT i.surname) as primers_cognoms,
collect(DISTINCT i.second_surname) as segons_cognoms
```

nom	primers_cognoms	segons_cognoms
"miguel"	["estape"]	["bofill", "bofill"]

c) Mostra els fills o filles(només) de "benito julivert". Mostra la informació en forma de taula: el nom, cognom1, cognom2, i tipus de relació. Ordena els resultats alfabèticament per nom.

Busquem els fills en aquest cas, fem la relació FAMILIA i busquem a benito com l'origen, després només hem de mostrar els destinataris i fer l'ORDER BY.

```
MATCH (i:INDIVIDUAL)-[f:FAMILIA]->(a:INDIVIDUAL)
WHERE i.name = 'benito' AND i.surname = 'julivert' AND f.Relacio_Harmonitzada = 'fill'
RETURN a.name as nom, a.surname as cognom1, a.second_surname as cognom2,
f.Relacio_Harmonitzada as relació
ORDER BY nom
```

nom	cognom1	cognom2	relació
"jose"	"julibert"	"julia"	"fill"
"juan"	"julibert"	"julia"	"fill"
"martin"	"julibert"	"julia"	"fill"

d) Mostreu les famílies de Castellví de Rosanes amb més de 3 fills. Mostreu el nom i cognoms del cap de família i el nombre de fills. Ordeneu-les pel nombre de fills fins a un límit de 20, de més a menys.

Ens quedem primer amb tots els caps de família i els relacionem amb els habitatges per quedar-nos només amb els de Castellví de Rosanes (CR). Després només necessitem un count dels fills per cada cap i al final tornar el cap amb el nombre, mostrant només 20 amb el LIMIT i ordenant per la quantitat de més a menys.

```
MATCH (h:HABITATGE)<-[v:VIU]-(i:INDIVIDUAL)-[f:FAMILIA]->(a:INDIVIDUAL)
WHERE h.Municipi = 'CR'
WITH h, i, count(f.Relacio_Harmonitzada = 'fill') as fills
WHERE fills>3
RETURN [i.name, i.surname, i.second_surname] as cap, fills as nombre_fills
ORDER BY nombre_fills DESC
LIMIT 20
```

cap	nombre_fills
["miguel", "canals", "sagarra"]	12
["mariangela", "aregay", "ilegible"]	10
["jose", "canals", "mila"]	9
["jacinto", "pujadas", "mestres"]	9
["magdalena", "parera", "sabat"]	9
["jose", "canals", "olle"]	9
["pablo", "astruch", "julia"]	9
["jose", "olle", "domenech"]	8
["benito", "julivert", "parera"]	8
["antonio", "julia", "rafuls"]	8

["pedro", "bargallo", "ilegible"]	8
["jose", "rafuls", "mila"]	7
["jaime", "jarrey", "ilegible"]	7
["juan", "olle", "ilegible"]	7
["ramon", "canals", "amat"]	7
["pablo", "bargallo", "armangol"]	7
["francisco", "olle", "mas"]	7
["pablo", "canals", "llimona"]	7
["jose", "llopart", "domenech"]	7
["francisco", "aregay", "rigol"]	7

e) Per cada padró/any de Sant Feliu de Llobregat, mostra el carrer amb menys habitants i el nombre d'habitants en aquell carrer. Fes servir la funció min() i CALL per obtenir el nombre mínim d'habitants. Ordena els resultats per any de forma ascendent.

Primer hem de fer la subconsulta que ens retorn la quantitat mínima de habitants i l'any del padró d'aquest habitatge. Després utilitzem aquestes dades per filtrar els carrers per quedar-nos amb els carrers que tenen una quantitat d'habitants igual al mínim.

Per mostrar un sol carrer, només fa falta agafar el primer de la llista.

Versió per ensenyar només 1 carrer:

```
CALL {MATCH (h2:HABITATGE)<-[v:VIU]-(i2:INDIVIDUAL)
WHERE h2.Municipi = 'SFLL'
WITH h2,count(i2) as numH2
RETURN h2.Any_Padro as any_p, min(numH2) as n}
MATCH (h:HABITATGE)<-[v:VIU]-(i:INDIVIDUAL)
WITH h,any_p, n,count(i) as numH
WHERE h.Any_Padro = any_p AND numH = n AND h.Municipi = 'SFLL'
RETURN h.Any_Padro as any_padro, COLLECT(DISTINCT h.Carrer)[0] as carrer, numH as num
```

any_padro	carrer	num
1881	"carretera"	1
1889	"carretera"	1
1878	"carretera"	1
1833	"rectoria"	1
1839	"rectoria"	1
1838	"carretera"	1

Versió per ensenyar tots els carrers amb el nombre mínim:

```
CALL {MATCH (h2:HABITATGE)-[v:VIU]-(i2:INDIVIDUAL)
WHERE h2.Municipi = 'SFLL'
WITH h2,count(i2) as numH2
RETURN h2.Any_Padro as any_p, min(numH2) as n}
MATCH (h:HABITATGE)-[v:VIU]-(i:INDIVIDUAL)
WITH h,any_p, n,count(i) as numH
WHERE h.Any_Padro = any_p AND numH = n AND h.Municipi = 'SFLL'
RETURN h.Any_Padro as any_padro, COLLECT(DISTINCT h.Carrer) as carrers, numH as num
```

any_padro	carrers	num
1881	["carretera", "iglesia", "falguera", "masovernou", "serra", "abajo"]	1
1889	["carretera", "plaza constitucion", "creus", "falguera", "masovernou", "masover nou", "sta maria", "calle de la carretera", "s n antonio"]	1
1878	["carretera", "iglesia", "creus", "falguera", "carretà", "masovernou", "masover nou", "serra", "cruces", "afueras", "sta maria"]	1
1833	["rectoria", "carretera de la part de molins de rey", "masovenou", "creus", "baix", "dal", "falguera", "del ribas"]	1
1839	["rectoria", "carretera de la part de molins de rey", "creus", "dal", "carretera de barna", "plaza", "masove nou"]	1
1838	["carretera"]	1

Exercici 3. Anàlisi de Gracs

Primer apartat. a)

- **Pas previ. Creació del graf en memòria.**

Per tal d'anàlitzar el graf utilitzant la llibreria de Graph Data Science Library (GDS) necessitem crear un graf en memòria. És necessari ja que els algorismes de GDS funcionen amb gracs que estan carregats a la memòria del sistema. Amb això s'aconsegueix millorar el rendiment i permet fer operacions complexes en el graf.

La query per escriure el graf a memòria és la següent:

```
//Per crear el graf
CALL gds.graph.project(
  'Padrons',
  ['INDIVIDUAL', 'HABITATGE'],
  {
    VIU: {orientation: NATURAL},
    FAMILIA: {orientation: NATURAL},
    SAME_AS: {orientation: NATURAL}
  }
);
```

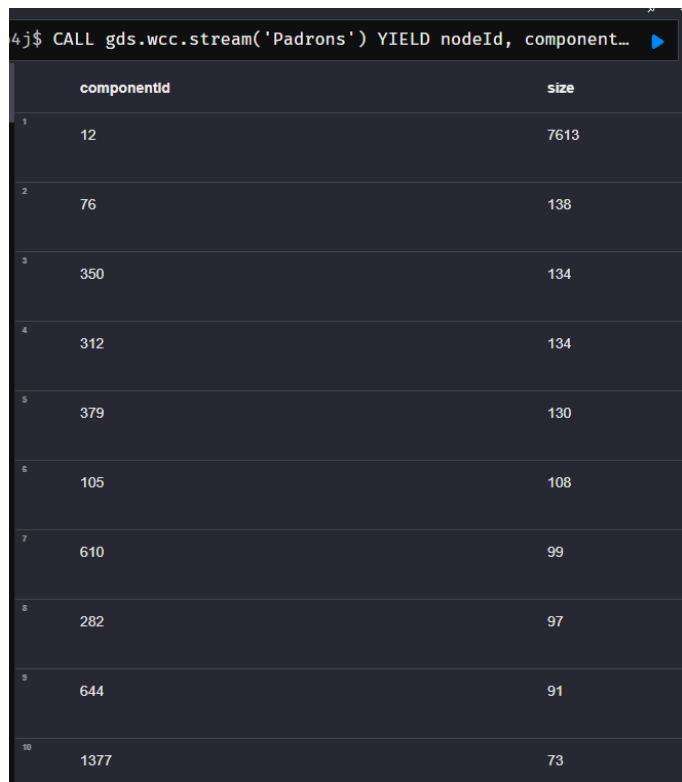
* Posem les relacions a NATURAL per tal de mantenir les direccions de les arestes del graf original.

- **a1) Mostra, en forma de taula, les 10 components connexes més grans (ids i mida).**

```
CALL gds.wcc.stream('Padrons')
YIELD nodeId, componentId
RETURN componentId, count(*) AS size
ORDER BY size DESC
LIMIT 10;
```

Aquesta query executa l'algoritme de components connexes, o wcc (weakly connected components), obté les components, les ordena pel número de nodes de les que disposa i filtra les 10 que més en tenen.

Podem veure en els resultats que la component amb Id = 12 és la que té més nodes en comparació de la resta.



4j\$ CALL gds.wcc.stream('Padrons') YIELD nodeId, component...

	componentid	size
1	12	7613
2	76	138
3	350	134
4	312	134
5	379	130
6	105	108
7	610	99
8	282	97
9	644	91
10	1377	73

- **a2) Per cada municipi i any el nombre de parelles del tipus: (Individu)—(Habitatge).**

```
CALL gds.wcc.stream('Padrons')
YIELD nodeId, componentId
WITH gds.util.asNode(nodeId) AS individu, componentId
MATCH (individu:INDIVIDUAL)-[v:VIU]->(h:HABITATGE)
RETURN h.Municipi AS Municipi, h.Any_Padro AS Any, count(*) AS Parelles
ORDER BY Parelles DESC, Any;
```

Per assolir aquest objectiu també és necessari utilitzar l'algorisme wcc ja que ens permet identificar grups de nodes que estan connectats entre ells. Després d'aconseguir els grups i havent recuperat la id i el component de cada node, necessitem obtenir la referència de cada node de tipus INDIVIDUAL per establir la relació amb els nodes d'HABITATGE. Finalment, utilitzant aquesta informació, i classificant els resultats per municipi i any de padró, obtenim el següent resultat:

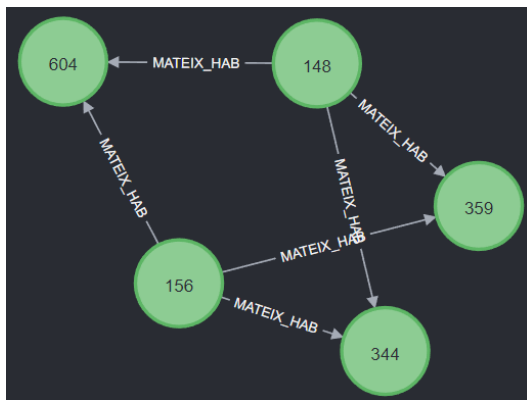
	Municipi	Any	Parelles
1	"SFLL"	1889	3117
2	"SFLL"	1881	3000
3	"SFLL"	1878	2745
4	"SFLL"	1839	1946
5	"SFLL"	1833	1433
6	"CR"	1866	337

Segon apartat. b)

- **b1) Determineu els habitatges que són els mateixos al llarg dels anys.**

```
MATCH (h1:HABITATGE), (h2:HABITATGE)
WHERE h1.Municipi = h2.Municipi AND h1.Carrer = h2.Carrer AND h1.Numero =
h2.Numero AND h1.Any_Padro < h2.Any_Padro
MERGE (h1)-[:MATEIX_HAB]->(h2);
```

Primerament es fa un MATCH per trobar dos nodes que siguin habitatges i seguidament es filtren només els que tenen tots els atributs que no són l'Any_Padro iguals (excepte la id de l'habitatge que pot canviar durant els anys). Llavors cal confirmar que l'Any_Padro és més recent en un habitatge que en l'altre i finalment es crea la relació MATEIX_HAB.



Tots aquests habitatges són en realitat el mateix al llarg dels anys.

Com podem veure en les fotos dels atributs, el únic que canvia és l'atribut Any_Padro.

HABITATGE	
<elementId>	4:aed7a117-10f3-49b7-b974-905309e6a4a1:131
<id>	131
Any_Padro	1881
Carrer	creus
Id_Habitatge	359
Municipi	SFLL
Numero	3

HABITATGE	
<elementId>	4:aed7a117-10f3-49b7-b974-905309e6a4a1:132
<id>	132
Any_Padro	1889
Carrer	creus
Id_Habitatge	344
Municipi	SFLL
Numero	3

- **b2) Creeu un graf en memòria que inclogui els nodes Individu i Habitatge i les relacions VIU, FAMILIA, MATEIX_HAB que acabeu de crear.**

```
CALL gds.graph.project(
  'Subgraf',
  ['INDIVIDUAL', 'HABITATGE'],
  {
    VIU: {orientation: 'UNDIRECTED'},
    FAMILIA: {orientation: 'UNDIRECTED'},
    MATEIX_HAB: {orientation: 'UNDIRECTED'}
  }
);
```

La creació del graf és similar a la del apartat a) però ara requerim de que els nodes siguin no dirigits.

- **b3) Calculeu la similaritat entre els nodes del graf que acabeu de crear, escriviu el resultat de nou a la base de dades i interpreteu els resultats obtinguts.**

```
CALL gds.nodeSimilarity.stream('Subgraf')
YIELD node1, node2, similarity
WITH gds.util.asNode(node1) AS node1, gds.util.asNode(node2) AS node2, similarity
WHERE (labels(node1)[0] = labels(node2)[0])
MERGE (node1)-[:SIMILAR_TO {Similaritat: similarity}]->(node2)
```

Calculem la similaritat dels nodes utilitzant la funció nodeSimilarity que ja ens proporciona GDS. Ho fem només sobre els nodes que són del mateix tipus entre ells (INDIVIDUAL o HABITATGE) ja que no tendria sentit calcular-la d'una altra manera.

Treball en equip (distribució de tasques)

Pel que fa a la distribució del nostre projecte, les tasques han sigut repartides de la següent manera:

- **Creació Github:** Joan Collillas
- **Exercici 1 (Importació dels CSV)**

Adrià Díaz (Creació del codi)

Joan Colilles (Revisió i correcció)

Bernat Vidal (Revisió i correcció)

- **Exercici 2 (Queries Cypher)**

Adrià Fraile

- **Exercici 3 (Anàlisi de grafs)**

Joan Colilles (primera part)

Adrià Díaz (segona part)

Redacció de l'informe:

- **Exercici 1:** Bernat Vidal
- **Exercici 2:** Adrià Fraile
- **Exercici 3:** Joan Colillas (primera part) i Adrià Díaz (segona part)
- **Treball en equip:** Adrià Díaz i Joan Colillas

[Link al repositori de Github](#)