Universitat Autònoma de Barcelona

Facultat de Ciències

Projecte final Neo4j

Bases de dades no relacionals

Francesc Albareda Civit - 1603751
Alba Fernández Coronado - 1600123
Marina Palomar González - 1605547
Guillem Paz García - 1598850

GitHub: `https://github.com/AlbaFernandezCor/Projecte-Neo4j`

Juny 2023

Contents

1 Introducció

Aquest treball tracta sobre el disseny, la implementació i la consulta a una base de dades en Neo4j implementada manualment a través de l'aplicació *Neo4j Browser*. L'objectiu és acabar d'assolir els conceptes teòrics de l'assignatura mitjançant la realització del projecte en grup.

2 Repartició de la feina

El nostre grup de treball està format per quatre integrants: el Francesc Albareda, l'Alba Fernández, la Marina Palomar i el Guillem Paz.

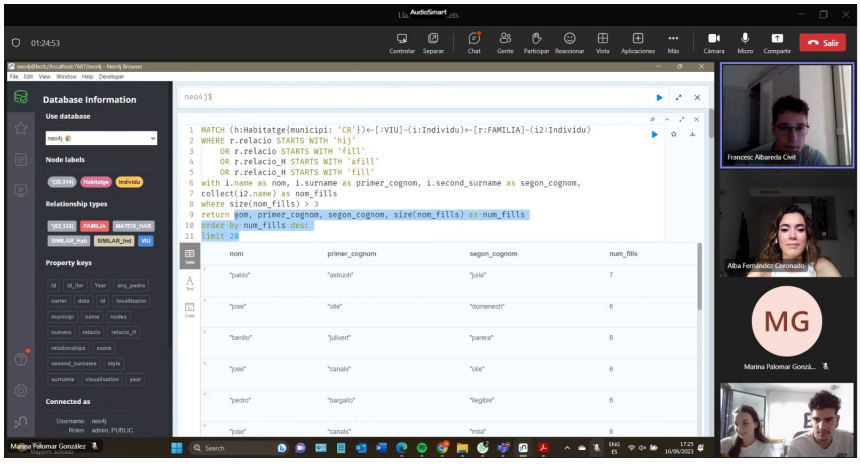
Al llarg de la realització d'aquest treball, la majoria de temps hem estat connectats conjuntament a través de la plataforma Teams, com es pot veure en la captura de pantalla adjuntada al final de la secció. En alguns moments puntuals ens hem repartit algunes tasques a causa de la dificultat de poder coincidir tots quatre, intentant que tothom participés i estigués al cas en cadascuna de les diferents tasques.

En l'exercici 1 es va treballar de manera conjunta en tot moment, ja que es va considerar molt important que tothom conegués a la perfecció la distribució de les dades i la presència de *constraints*. En treballar sempre junts, no s'ha utilitzat el GitHub en gairebé cap moment i s'ha optat per a posar-ho directament a l'informe. L'encarregada de compartir pantalla i liderar aquesta part va ser l'Alba.

Pel que fa a l'exercici 2, en primera instància es va treballar de manera col·lectiva per a fer les comandes número 1, 2, 7, 12 i 13, amb el Guillem al capdavant. De nou, en estar tot l'equip al complet, es van adjuntar directament a l'informe. Pel que fa a la resta de comandes, aquestes es van repartir intentant que cada membre del grup fes diferents tipus de comandes. Finalment, han quedat distribuïdes de la següent manera: la Marina ha fet les consultes 3 i 11, que va pujar al GitHub. Tot seguit, el Guillem va fer la 5 i la 6, l'Alba la número 8 i la número 9 i, finalment, el Francesc va fer la 4 i la 10. Un cop l'script va estar complet amb les comandes que faltaven, aquestes van ser comprovades conjuntament i manualment mitjançant Excel, amb la Marina al capdavant.

El Francesc va liderar l'últim exercici, en què tot el grup va decidir com es realitzaria l'estudi dels diferents apartats. Un cop decidida l'estructura i els passos a seguir, l'Alba i el Francesc van buscar les comandes necessàries per a fer l'apartat A), mentre que el Guillem i la Marina van fer-ho amb l'apartat B). Tot això es va penjar al GitHub i es va fer la posterior comprovació conjuntament.

Per acabar, sempre que s'ha treballat en l'informe usant *Overleaf*, hem estat presents tots els membres del grup. Tot i això, fer menció especial a la Marina, ja que ha estat la que ha passat al davant en aquest apartat.



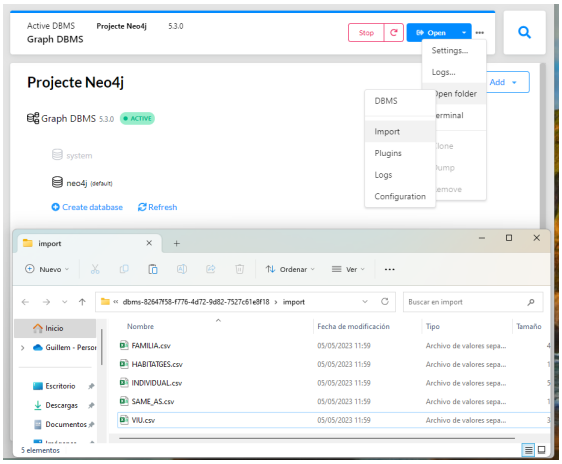
3 Inicialització del projecte

L'exercici 1 consisteix a inserir totes les dades dels diferents fitxers *.csv* a la base de dades, tenint en compte un conjunt de restriccions indicades a l'enunciat.

Primerament cal crear un nou projecte, personalitzar-li el nom (*Projecte_Neo4j* en aquest cas) i crear una base de dades des del botó *(+)Add*, afegint una nova base de dades amb nom i contrasenya. Tot seguit, s'han d'instal·lar els pluggins d'APOC i *Graph Data Science Library* fent clic a la base de dades creada i a pluggins al menú de la dreta.

Un cop es tenen totes les eines a punt, només fa falta carregar els fitxers de les dades, que es fa des de ... → *OpenFolder* → *import*, i moure en aquella carpeta els fitxers que s'utilitzaran.

Mentre un membre del grup compartia pantalla, la resta seguia les mateixes passes per a crear paral·lelament la base de dades i intentar aportar i provar comandes que semblessin poder funcionar correctament segons els nostres interessos.



4 Resolució d'exercicis

Un cop explicat tot el procés de repartició de la feina i el *timing* del projecte, s'han dut a terme els diferents exercicis proposats en l'enunciat d'aquest.

4.1 Exercici 1: Importació de les dades

En aquest primer exercici s'ha generat un script de *cypher* que carrega totes les dades. S'han generat els nodes i relacions i s'han afegit les característiques als llocs corresponents. A més a més, s'han hagut de tenir en compte certes consideracions a l'hora de dur a terme aquesta tasca, que són les següents:

- Ús de *constraints* i *indexos*
- Evitar la duplicació de les dades en executar el script més d'un cop
- No carregar files *null* dels fitxers CSV. Més concretament, tenir en compte únicament aquelles files que tenen valors dels identificadors de municipi, llar o individu diferent de *null*.

4.1.1 Creació dels nodes: Individu i Habitatge

Primerament, per tal de restringir la unicitat dels identificadors dels individus, s'ha realitzat la següent *constraint*:

```
CREATE CONSTRAINT UniqueIndividu FOR (i:Individu) REQUIRE i.Id IS UNIQUE
```

D'aquesta manera, es mostrarà un missatge d'error si s'intenta afegir un individu amb un identificador ja existent.

Un cop fet això, s'ha procedit amb la creació dels nodes de tipus *Individu* de la següent manera:

```
LOAD CSV WITH HEADERS FROM 'file:///INDIVIDUAL.csv' AS row
WITH toInteger(row.Id) AS Id, toInteger(row.Year) AS Year, row.name AS name,
row.surname AS surname1, row.second_surname AS surname2
WHERE row.Id IS NOT NULL
MERGE (i:Individu {Id: Id})
SET i.Year = Year, i.name = name, i.surname = surname1, i.second_surname = surname2
RETURN count(i);
```

Cal comentar que s'han realitzat les següents conversions de tipus, les quals s'han trobat necessàries:

- **Id**: s'ha passat de tipus *string* a tipus *int*.
- **Year**: s'ha passat de tipus *string* a tipus *int*.

Per últim, s'ha imposat la condició que l'ID de l'individu sigui diferent de *null*, ja que és una de les consideracions esmentades anteriorment.

Figure 1: Nodes Individu

Després de realitzar la comanda anterior, s'han creat 17606 nodes de tipus Individu.

A continuació, s'ha prosseguit a realitzar la creació dels nodes de tipus Habitatge. De nou, primer s'ha creat una restricció prèviament necessària a la creació dels nodes. En aquest cas, s'ha construït un *Node Key*, ja que els habitatges s'identifiquen únicament a partir de la tripleta de propietats *Id_llar*, *municipi* i *any_padro*.

```
CREATE CONSTRAINT UniqueHabitatge3 FOR (h:Habitatge)
REQUIRE (h.Id_llar, h.municipi, h.any_padro) IS NODE KEY
```

Un cop això, s'han dut a terme les creacions dels nodes de tipus Habitatge:

```
LOAD CSV WITH HEADERS FROM 'file:///HABITATGES.csv' AS row
WITH toInteger(row.Id_Llar) AS Id_llar, toInteger(row.Any_Padro) AS any_padro,
row.Municipi AS municipi, row.Carrer AS carrer, toInteger(row.Numero) AS numero
WHERE row.Id_Llar IS NOT NULL AND row.Municipi <> 'null' AND row.Any_Padro IS NOT NULL
MERGE (h:Habitatge {Id_llar: Id_llar, municipi: municipi, any_padro: any_padro})
SET h.carrer = carrer,
h.numero = numero
RETURN count(h);
```

Altre cop, s'han realitzat diverses conversions de tipus:

- **Id_llar**: s'ha passat de tipus *string* a tipus *int*.
- **Any_padro**: s'ha passat de tipus *string* a tipus *int*.
- **numero**: s'ha passat de tipus *string* a tipus *int*.

En aquest cas, les conidcions imposades han estat que aquestes mateixes propietats siguin diferents de *null*, ja que volem que existeixin les 3 sempre. Cal comentar que s'ha hagut d'implementar l'operador "<> *'null'* " per fer aquest filtratge perquè la comanda "*IS NOT NULL*" no detectava els valors nuls de les propietats.

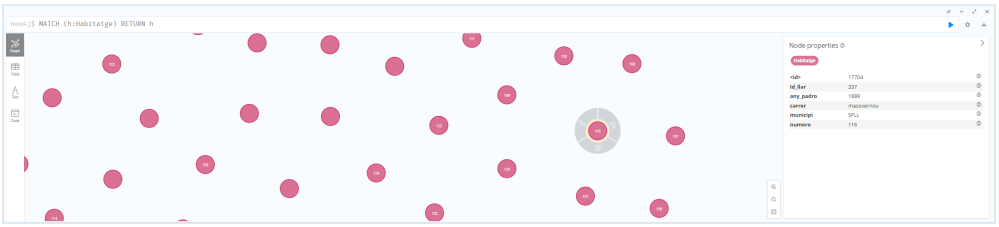


Figure 2: Nodes Habitatge

Després de realitzar la comanda anterior, s'han creat 2708 nodes de tipus Habitatge.

4.1.2 Creació de les arestes: *FAMILIA*, *SAME_AS* i *VIU*

Per crear les arestes de tipus FAMILIA, s'han fet diverses conversions de tipus necessàries:

- **ID_1**: s'anomenarà *Id_cap*, i s'ha passat de tipus *string* a tipus *int*.
- **ID_2**: s'anomenarà *Id_individu*, i s'ha passat de tipus *string* a tipus *int*.

```
LOAD CSV WITH HEADERS FROM 'file:///FAMILIA.csv' AS row
WITH toInteger(row.ID_1) AS Id_cap, toInteger(row.ID_2) AS Id_individu,
row.Relacio AS relacio, row.Relacio_Harmonitzada AS relacio_H
WHERE row.Relacio<>"null" OR row.Relacio_Harmonitzada<>"null"
MATCH (i1:Individu {Id: Id_cap})
MATCH (i2:Individu {Id: Id_individu})
MERGE (i2)-[rel:FAMILIA {relacio: relacio, relacio_H: relacio_H}]->(i1)
RETURN count(rel);
```

A més a més, s'ha considerat important tenir en compte només aquelles files que ens informessin de la relació entre els individus, ja fos a partir de la propietat *Relacio* o de la propietat *Relacio_Harmonitzada*. D'altra manera, no les podríem saber quina relació hi hauria entre els individus.

El graf amb aquesta relació es veuria de la següent manera:

Figure 3: Relació FAMILIA

El nombre de relacions de tipus FAMILIA creades ha estat de 12022.

Després d'acabar aquesta primera tasca, s'ha dut a terme la creació d'una segona relació: *SAME_AS*. Aquesta ens determina els nodes que representen el mateix individu al llarg del temps, i s'han construït amb la següent comanda:

```
LOAD CSV WITH HEADERS FROM 'file:///SAME_AS.csv' AS row
WITH toInteger(row.Id_A) AS Id_A, toInteger(row.Id_B) AS Id_B
MATCH (i1:Individu {Id: Id_A})
MATCH (i2:Individu {Id:Id_B})
MERGE (i1)-[rel:SAME_AS]->(i2)
RETURN count(rel);
```

De nou, els identificadors (*Id_A* i *Id_B*) s'han passat de tipus *string* a tipus *int*. Els resultats obtinguts es poden observar en el següent graf: El nombre de relacions creades d'aquest tipus ha estat de 7199.

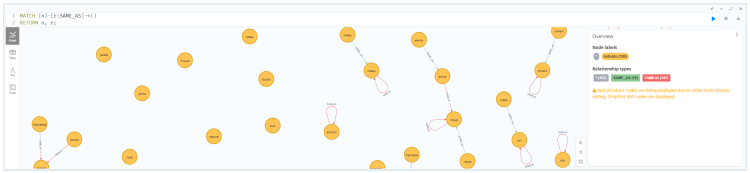


Figure 4: Relació SAME_AS

Per últim, s'ha dut a terme el procés de creació de la relació *VIU*, la qual representa el lloc on viu cada individu. En aquest cas, les conversions de tipus implementades han estat les següents:

- **IND**: s'anomenarà *Id_individu*, i s'ha passat de tipus *string* a tipus *int*.
- **HOUSE_ID**: s'anomenarà *Id_house*, i s'ha passat de tipus *string* a tipus *int*.
- **Year**: s'anomenarà *year*, i s'ha passat de tipus *string* a tipus *int*.

```
LOAD CSV WITH HEADERS FROM 'file:///VIU.csv' AS row
WITH toInteger(row.IND) AS Id_individu, toInteger(row.HOUSE_ID) AS Id_casa,
    toInteger(row.Year) AS year, row.Location AS localitzacio
MATCH (i:Individu {Id: Id_individu})
MATCH (h:Habitatge {Id_llar: Id_casa, municipi: localitzacio, any_padro:year})
MERGE (i)-[rel:VIU {year: year, localitzacio: localitzacio}]->(h)
RETURN count(rel);
```

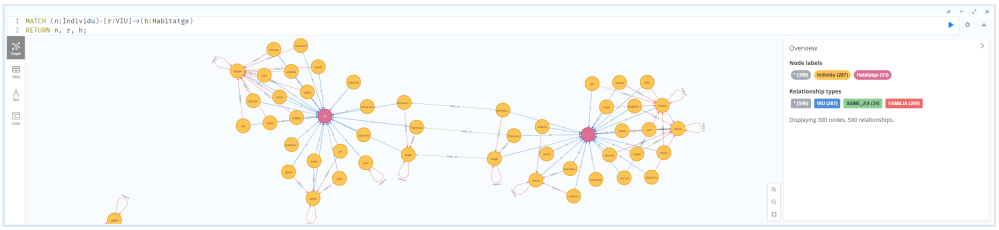


Figure 5: Relació VIU

Finalment, s'han creat 12865 relacions d'aquest tipus.

5 Queries

Query 1: Del padró de 1866 de Castellví de Rosanes (CR), retorna el número d'habitants i la llista de cognoms, sense eliminar duplicats.

```
MATCH(i:Individu)-[:VIU]->(h:Habitatge)
WHERE h.any_padro=1866 AND h.municipi='CR'
RETURN count(i) as NumHabitants, collect(i.surname) as Cognoms
```

Query 2: Per a cada padró de Sant Feliu de Llobregat (SFLL), retorna l'any de padró, el número d'habitants, i la llista de cognoms. Elimina duplicats i "nan".

```
MATCH(i:Individu)-[:VIU]->(h:Habitatge)
WHERE h.municipi='SFLL' AND i.surname<>'nan' AND i.second_surname<>'nan'
RETURN h.any_padro as `Any Padró` ,count(i) as NumHabitants, COLLECT(DISTINCT(i.surname))
    as Cognom1, COLLECT(DISTINCT(i.second_surname)) as Cognom2
```

Query 3: Dels padrons de Sant Feliu de Llobregat (SFLL) d'entre 1800 i 1845 (no inclosos), retorna la població, l'any del padró i la llista d'identificadors dels habitatges de cada padró. Ordena els resultats per l'any de padró.

```
match (h:Habitatge{municipi: 'SFLL'})
where 1800 < h.any_padro < 1845
return h.municipi as Població, h.any_padro as Any_Padró, collect(h.Id_llar) as Id_Hab
order by Any_Padró
```

Query 4: Retorna el nom de les persones que vivien al mateix habitatge que "rafel marti" (no té segon cognom) segons el padró de 1838 de Sant Feliu de Llobregat (SFLL). Retorna la informació en mode graf i mode llista.

```
MATCH (I:Individu)-[r:VIU]->(h:Habitatge)<-[r2:VIU]-(I2:Individu)
WHERE r.year=1838 AND r2.year=1838 AND I.name='rafel' AND I.surname='marti'
    AND h.municipi='SFLL'
RETURN I2 //Per retornar graph
//RETURN collect(I2.name) //Per retornar llista
```

Query 5: Retorna totes les aparicions de "miguel estape bofill". Fes servir la relació SAME_AS per poder retornar totes les instancies, independentment de si hi ha variacions lèxiques (ex. diferents formes d'escriure el seu nom/cognoms). Mostra la informació en forma de subgraf.

```
MATCH (i:Individu)-[:SAME_AS]-(i2:Individu)
WHERE (i.name='miguel' AND i.surname='estape' AND i.second_surname='bofill')
RETURN i, i2
```

Query 6: De la consulta anterior, retorna la informació en forma de taula: el nom, la llista de cognoms i la llista de segon cognom (elimina duplicats).

```
MATCH (i:Individu)-[:SAME_AS]-(i2:Individu)
WHERE i.name = 'miguel' AND i.surname = 'estape' AND i.second_surname = 'bofill'
RETURN i.name AS nom, COLLECT(DISTINCT i2.surname) AS cognoms,
    COLLECT(DISTINCT i2.second_surname) AS segonCognoms
```

Query 7: Mostra totes les persones relacionades amb "benito julivert". Mostra la informació en forma de taula: el nom, cognom1, cognom2, i tipus de relació.

```
MATCH(i1:Individu)-[r:FAMILIA]->(i2:Individu)
WHERE i2.name='benito' AND i2.surname='julivert'
RETURN i1.name, i1.surname, i1.second_surname, r.relacio, r.relacio_H
```

Query 8: De la consulta anterior, mostra ara només els fills o filles de "benito julivert". Ordena els resultats alfabèticament per nom.

```
MATCH (p1:Individu)-[r:FAMILIA]->(p2:Individu)
WHERE (p2.name = "benito" AND p2.surname = "julivert") AND (r.relacio = 'hijo'
    OR r.relacio = 'hija')
RETURN p1.name, p1.surname, p1.second_surname
ORDER BY p1.name ASC
```

Query 9: Llisteu totes les relacions familiars que hi ha.

```
MATCH ()-[r:FAMILIA]->()
WHERE r.relacio <> "null" AND r.relacio_H <> "null"
RETURN DISTINCT r.relacio, r.relacio_H
```

Query 10: Identifiqueu els nodes que representen el mateix habitatge (carrer i numero) al llarg dels padrons de Sant Feliu del Llobregat (SFLL). Seleccioneu només els habitatges que tinguin totes dues informacions (carrer i numero). Per a cada habitatge, retorneu el carrer i número, el nombre total de padrons on apareix, el llistat d'anys dels padrons i el llistat de les Ids de les llars (eviteu duplicats). Ordeneu de més a menys segons el total de padrons i mostreu-ne els 15 primers.

```
MATCH (:Individu)-[r:VIU]->(h:Habitatge)
WHERE h.municipi='SFLL' AND h.carrer IS NOT NULL AND h.numero IS NOT NULL
WITH h.carrer as carrer, h.numero as numero, collect(DISTINCT h.any_padro) AS any_padro,
    COLLECT(DISTINCT h.Id_llar) AS Id_llar, COUNT(*) AS totalPadrons
WHERE totalPadrons > 1
RETURN carrer, numero, totalPadrons, any_padro, Id_llar
ORDER BY totalPadrons DESC
LIMIT 15
```

Query 11: 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.

```
MATCH (h:Habitatge{municipi: 'CR'})<-[:VIU]-(i:Individu)<-[r:FAMILIA]-(i2:Individu)
WHERE r.relacio STARTS WITH 'hij'
    OR r.relacio STARTS WITH 'fill'
    OR r.relacio_H STARTS WITH 'afill'
    OR r.relacio_H STARTS WITH 'fill'
WITH i.name AS nom, i.surname AS primer_cognom, i.second_surname AS segon_cognom,
COLLECT(i2.name) AS nom_fills
WHERE SIZE(nom_fills) > 3
RETURN nom, primer_cognom, segon_cognom, size(nom_fills) AS num_fills
ORDER BY num_fills desc
LIMIT 20
```

Query 12: Mitja de fills a Sant Feliu del Llobregat l'any 1881 per família. Mostreu el total de fills, el nombre d'habitatges i la mitja de fills per habitatge. Fes servir CALL per obtenir el nombre de llars.

```
CALL {
    MATCH (h:Habitatge{municipi: 'SFLL'})<-[:VIU]-(i:Individu)<-[r:FAMILIA]-(i2:Individu)
    WHERE h.any_padro = 1881
    RETURN h, i, i2
}
MATCH (h)<-[:VIU]-(i)<-[r:FAMILIA]-(i2)
WHERE r.relacio STARTS WITH 'hij'
    OR r.relacio STARTS WITH 'fill'
    OR r.relacio_H STARTS WITH 'afill'
    OR r.relacio_H STARTS WITH 'fill'
WITH COUNT(i2.name) as nom_fills, COUNT(DISTINCT(h.Id_llar)) as num_habitatges
RETURN nom_fills AS num_fills, num_habitatges, toFloat(nom_fills)/toFloat(num_habitatges)
    AS mitja_fills
```

Query 13: 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.

```
CALL {
  MATCH (h:Habitatge{municipi: 'SFLL'})<-[:VIU]-(i:Individu)
  WITH h.any_padro AS any_padro, h.carrer AS carrer, count(i) AS num_persones
  ORDER BY num_persones ASC
  RETURN any_padro, COLLECT({carrer: carrer, num_persones: num_persones})[0] AS ind_carrers
}
RETURN any_padro, ind_carrers.carrer AS carrer, ind_carrers.num_persones AS min_persones
ORDER BY any_padro
```

6 Estudi del Graf

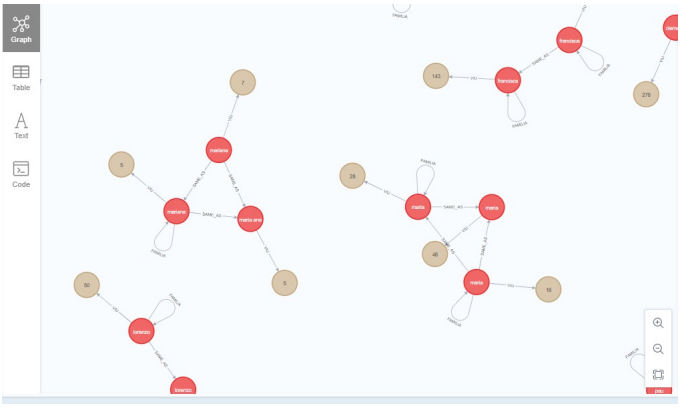
6.1 Estudi de les components connexes

Per tal d'estudiar les components connexes d'un graf format per nodes de tipus *Individu* i *Habitatge* i la relació *VIU*, primerament es crea el graf desitjat amb la comanda següent:

```
CALL gds.graph.project('graf_3a_viu', ['Individu', 'Habitatge'], ['VIU'])
```

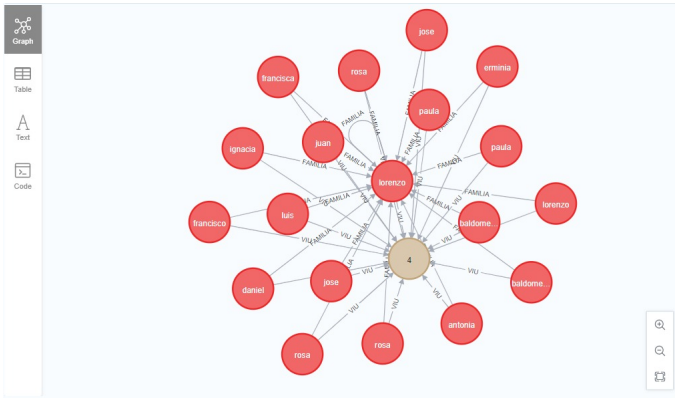
1. Quantes persones han viscut soles durant tots els anys?
Amb aquesta comanda s'ha volgut analitzar el nombre de persones que han viscut soles en un habitatge. Per fer aquesta consulta s'ha utilitzat l'algoritme *weakly connected component* en mode *stream* per cercar les components de mida dos, és a dir, un Individu i un Habitatge. Per relacionar els individus que viuen a un habitatge, s'ha usat la relació *VIU*. Cal remarcar, que hi ha components amb dues persones i dos habitatges, però això és degut a que agafa les mateixes persones en diferents anys. Com a resultat s'ha obtingut un total de 132 persones. Si es fa la comanda de manera que es mostri el graf, s'obté el següent:

```
CALL gds.wcc.stream('graf_3a_viu')
YIELD componentId, nodeId
WITH componentId, collect(nodeId) as nodes, size(collect(nodeId)) as mida
match (i:Individu) - [r:VIU]->(h:Habitatge)
where id(i) in nodes and mida = 2
return i,r,h
```



2. A quina casa han viscut més persones al llarg del temps?
L'objectiu d'aquesta segona cerca és investigar quina ha estat la casa amb més habitants durant els anys de la base de dades. Per realitzar aquesta consulta, i com bé s'ha esmentat abans, s'han relacionat els individus i els habitatges amb la relació *VIU*. Per tal de retornar la casa amb més habitants de la base de dades, s'ha fet un collect() dels habitants per cada casa habitada i s'ha ordenat de més gran a més petit afegint un límit 1. Com a resultat de la consulta s'ha obtingut que a la casa número "4" amb id 604, han viscut durant un total de 19 persones en tots els anys. Si es fa la comanda de manera que es mostri el graf, s'obté el següent:

```
CALL gds.wcc.stream('graf_3a_viu')
YIELD componentId, nodeId
WITH componentId, collect(nodeId) as nodes, size(collect(nodeId)) as mida
ORDER BY mida DESC LIMIT 1
match (n)
where id(n) in nodes
return n;
```



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

L'objectiu d'aquesta tercera cerca és investigar les relacions Individu - Habitatge per cada municipi i any. Per tal de retornar el nombre de parelles, s'ha fet un *count()* d'un *distinct()*, per tal de fer un recompte dels valors únics. Si es fa la comanda de manera que es mostri una taula amb el resultat, s'obté el següent:

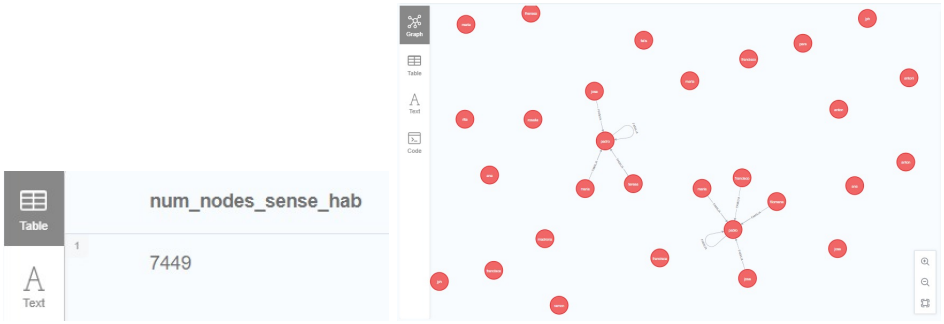
```
MATCH (i:Individu)-[:VIU]->(h:Habitatge)
RETURN h.municipi AS municipi, h.any_padro AS any_padro, COUNT(DISTINCT i) AS num_parelles
ORDER BY any_padro
```

	municipi	any_padro	num_parelles
1	"SFLL"	1833	1433
2	"SFLL"	1838	287
3	"SFLL"	1839	1946
4	"CR"	1866	337
5	"SFLL"	1878	2745
6	"SFLL"	1881	3000
7			

Started streaming 7 records after 1 ms and completed after 32 ms.

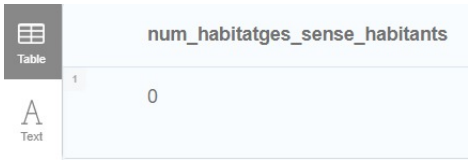
4. Quantes components connexes no estan connectades a cap node de tipus 'Habitatge' L'objectiu d'aquesta quarta cerca és estudiar quantes components connexes no estan connectades a cap Habitatge. És a dir, quantes persones sense sostre hi ha a la base de dades. Com a resultat s'obté que el nombre de persones sense sostre registrades a la base de dades és 7449.

```
MATCH (n)
WHERE NOT (n)-[]->(:Habitatge)
RETURN count(n) AS num_nodes_sense_hab
```



5. Comprovar si algun habitatge no té cap habitant associat Per acabar, i focalitzant aquesta consulta com a antagonista de l'anterior, el que es vol estudiar aquí és si hi ha algun habitatge en el qual no visqui ningú, és a dir, que no tingui cap individu associat. Si es fa la comanda de manera que es mostri una taula amb el resultat, s'obté el següent, on es pot veure que tots els habitatges estan habitats.

```
MATCH (h:Habitatge)
WHERE NOT (h)<-[:VIU]-(:Individu)
RETURN count(h) AS num_habitatges_sense_habitants
```



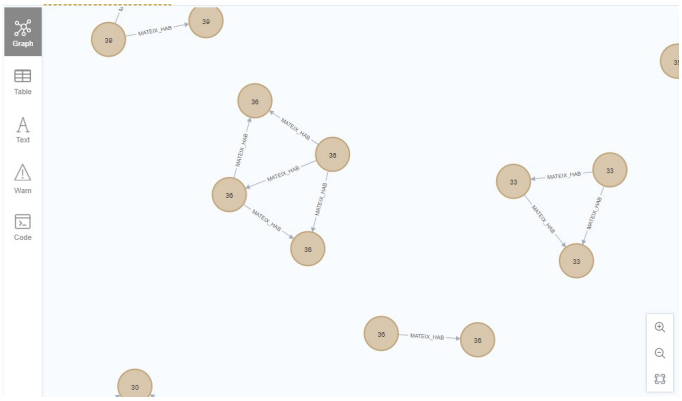
6.2 Semblança entre nodes

En aquest apartat s'ha realitzat l'estudi de la similitud entre els nodes del graf sobre el qual s'ha estat treballant.

Primerament, s'han determinat els habitatges que al llarg dels anys se n'han anat actualitzant a la base de dades mitjançant una aresta de relació dirigida cap al node més antic, creant així una relació similar a la de *SAME_AS* entre individus però amb habitatges.

Per dur a terme hem tingut en compte que els habitatges no canvien de municipi, número ni carrer sinó que només se'ls modifica l'identificador a la base de dades.

```
match (h1:Habitatge),(h2:Habitatge)
where h1.numero = h2.numero and h1.carrer = h2.carrer and h1.municipi = h2.municipi
and h2.any_padro < h1.any_padro
create (h1)-[r:MATEIX_HAB]->(h2)
return h1, r, h2
```



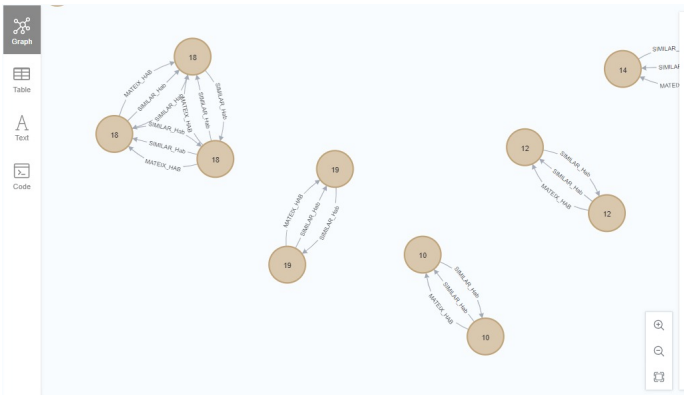
Seguidament, s'ha generat un graf en memòria amb els nodes i les relacions existents a la base de dades exceptuant la relació *SAME_AS*.

```
call gds.graph.project('graph3b',['Individu','Habitatge'],['VIU','FAMILIA','MATEIX_HAB'])
```

Finalment, per tal de visualitzar la similitud entre nodes s'han realitzat tres estudis diferents: similitud entre nodes de tipus *Habitatge*, similitud entre nodes de tipus *Individu* i similitud del graf sencer, usant el graf creat en memòria anteriorment.

Prenent només el node *Habitatge*, s'ha fet ús la llibreria de Neo4j *GDS* escrivint a la base de dades la relació *SIMILAR_Hab* entre aquells nodes que comparteixen veïns. El resultat obtingut són 1036 nodes comparats i 1238 relacions escrites. Analitzant gràficament aquesta sortida concloem que s'adequa a la sortida que teníem prevista veure, ja que la majoria d'ells es relacionaven per ser el mateix habitatge registrat a la base de dades al llarg del temps.

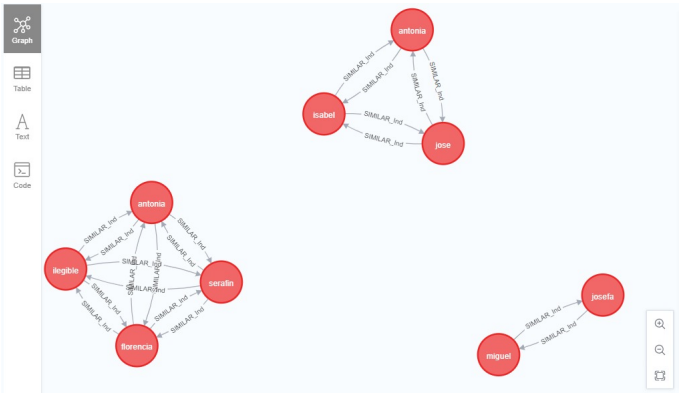
```
CALL gds.nodeSimilarity.write('graph3b',{
writeRelationshipType: 'SIMILAR_Hab',
writeProperty: 'score',
nodeLabels: ['Habitatge']
})
YIELD nodesCompared, relationshipsWritten
```



Fent el mateix estudi, però usant el node *Individu* i afegint un paràmetre restrictiu del 0.9 per assegurar més similitud entre nodes, ens trobem amb 8973 nodes comparats i 34604 relacions escrites.

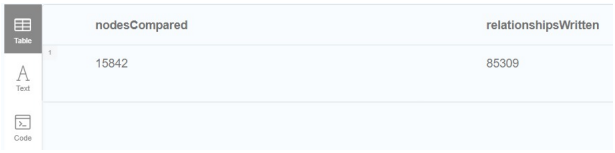
Observant la gràfica situada més avall, aquest resultat ens ha sorprès tot i tenir sentit, ja que ens esperàvem veure únicament nodes que guardessin el registre del mateix individu al llarg del temps (com un *SAME_AS*). En canvi, ens hem trobat que es relacionaven entre els individus que havien viscut en el mateix habitatge o fossin família. No obstant, com s'ha dit anteriorment, aquest fet té sentit, pel fet que en realitzar *nodeSimilarity* es busca per quantitat de nodes veïns comuns i no per atributs comuns entre nodes.

```
CALL gds.nodeSimilarity.write('graph3b',{
writeRelationshipType: 'SIMILAR_Ind',
writeProperty: 'score',
similarityCutoff: 0.9,
nodeLabels: ['Individu']
})
YIELD nodesCompared, relationshipsWritten
```



Finalment, s'ha realitzat l'estudi de la similitud entre nodes de tot el graf creat anteriorment amb les relacions *FAMILIA, VIU i MATEIX_HAB*. Aquest cop els resultats són els següents:

```
CALL gds.nodeSimilarity.write('graph3b', {
writeRelationshipType: 'SIMILAR',
writeProperty: 'score'
}) YIELD nodesCompared, relationshipsWritten
```



7 Conclusions

En aquest projecte s'han treballat diferents àrees de Neo4j, més concretament la inserció de dades, les consultes i l'anàlisi del graf.

En la base de dades apareixen algunes discrepàncies de carrers que a simple vista poden ser els mateixos, com per exemple *parra* i *calle de la parra*. Aquests inconvenients s'han intentat solucionar de la millor manera possible, tot i que alguns cops no ha estat possible. El mateix ha succeït amb les relacions de tipus $fill/a$, de les quals existeixen gran varietat de noms, com ara $fill\_1$, $fill\_2$, etc. Aquest últim fet ha estat solucionat mitjançant la comanda *STARTS WITH*.

És cert que no s'ha utilitzat el GitHub de la manera que se'ns demanava, ja que a l'haver estat treballant de manera conjunta la major part del temps, tots els membres del grup disposaven de tot el material realitzat durant la sessió. Quan no s'ha fet feina conjuntament, s'ha repartit la càrrega d'aquesta de manera correcta i equilibrada i s'han penjat els resultats al GitHub per tal que tots els membres del grup poguessin tenir accés al material creat. Tots els canvis i modificacions fetes durant el projecte s'han fet i notificat des de l'Overleaf, compilador en línia de *LaTeX* al qual tenim un document compartit on tots tenim accés.

Finalment, després de tot l'esforç invertit en aquesta feina, estem molt satisfets amb el treball fet per part de tot el grup i amb els continguts assolits.