

PROJECTE NEO4J: PADRONS

Matemàtica Computacional i Analítica de Dades
2021-2022

https://github.com/PumaresBenaiges/BDnR_MATCAD2022_Grup8

Àlex Correa Orri 1564967
Júlia Pumares Benaiges 1566252
Cristina Tost Abadias 1569280
Esteban Montoya Aranda 1526966

1. INTRODUCCIÓ

En aquesta pràctica treballarem sobre una base de dades que conté informació sobre habitatges, individus i les seves relacions (relació familiar i en quin habitatge viu, principalment). Aquesta BD consta de 21.288 nodes i 34.647 relacions.

Haurem de carregar les dades a la BD de Neo4J, realitzar algunes consultes per entendre-les millor i extreure informació útil, analitzarem l'estructura a nivell de grafs (components connexes principalment) i la semblança entre els nodes (habitatge i individu).

2. EXERCICI 1

Importa les dades en la BD de Neo4j del projecte. Genera un script en cypher que carregui totes les dades, generi tots els nodes, relacions i afegeix les característiques allà on toqui. A més. Heu d'assegurar-vos que en executar el script dues vegades no es dupliquin les dades.

Per aquest exercici hem generat el fitxer ex1.txt que es troba al repositori GitHub.

Els passos que hem seguit són:

- Crear dues constraints, per individus i habitatges
- Crear els dos tipus de nodes: individus i habitatges
- Crear les relacions: família, same_as i viu
- Després de cada pas hem fet una comanda per veure alguns exemples i comprovar que s'ha fet bé
- Al final hem afegit una comanda per esborrar tots els nodes i relacions

3. EXERCICI 2

- a. **Dels padró de 1866 de Castellví de Rosanes (CR), retorna el número d'habitants i la llista de noms. Elimina duplicats i nan.**

```
MATCH (i:Individu)-[v:VIU]->(h:Habitatge {anyPadro: 1866, municipi: 'CR'})
WHERE i.nom<>'nan'
RETURN count(i), collect(DISTINCT i.nom)
```

count(i)	collect(DISTINCT i.nom)
333	["emilia", "elisa", "manuel", "rosa", "antonio", "miguel", "jose", "juan", "teresa", "magdalena", "francisco", "salvador", "pedro",

```
MATCH (i:Individu)-[v:VIU]->(h:Habitatge {anyPadro: 1866, municipi: 'CR'})
WHERE i.cognom<>'nan'
RETURN count(i), collect(DISTINCT i.cognom)
```

count(i)	collect(DISTINCT i.cognom)
336	["olle", "galceran", "rusell", "suñol", "julibert", "bargallo", "anglada", "vila", "ros", "julia", "julivert", "gaset", "rumeu", "parera", "canals",

L'enunciat ens demanava la llista de noms però a les solucions sortia la llista de cognoms, així que hem fet les dues coses. Surten resultats diferents ja que en la primera treiem els que tinguin valors nan al nom i en la segona treiem els que tinguin valors nan en el cognom.

- b. Dels padrons de Sant Feliu de Llobregat (SFLL) d'abans de l'any 1840 (no inclòs), 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 (i:Individu)-[v:VIU]->(h:Habitatge {municipi: 'SFLL'})
WHERE h.anyPadro < 1840
RETURN count(i) AS poblacio, h.anyPadro, collect(DISTINCT h.id_hab) AS llista_llars
ORDER BY h.anyPadro
```

poblacio	h.anyPadro	llista_llars
1433	1833	[95, 99, 101, 103, 105, 107, 109, 111, 94, 96, 97, 98, 108, 100, 104, 10
287	1838	[321, 324, 326, 328, 320, 322, 323, 325, 327, 329, 330, 331, 332, 333,
1946	1839	[721, 722, 723, 724, 725, 731, 733, 734, 726, 735, 727, 729, 730, 728,

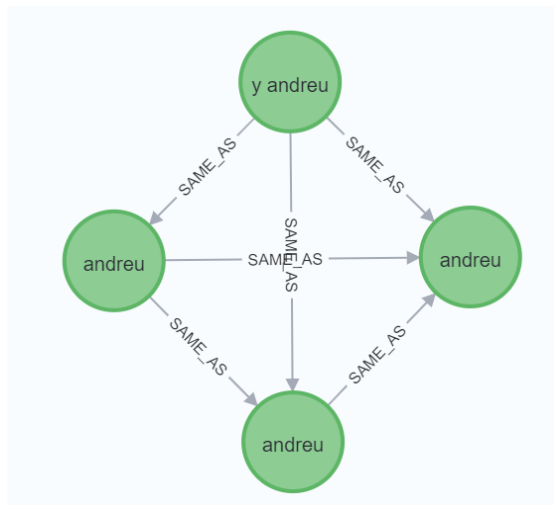
- c. Retorna el nom de les persones que vivien al mateix habitatge que "rafel marti" (no té segon cognom) segons el padró de 1838 de Sant Feliu de Llobregat (SFLL). Retorna la informació en mode graf i mode llista.

```
MATCH (i:Individu {nom: 'rafel', cognom: 'marti'})-[:VIU]->(h:Habitatge
{anyPadro: 1838, municipi: 'SFLL'})<-[:VIU]-(e:Individu)
RETURN collect(e.nom) AS convivents
```

convivents
["jpha", "felipe", "maria", "jpha", "franco", "jph", "miquel", "salvadora"]

- d. Retorna totes les aparicions de "Miguel ballester". 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 subgraf.

```
MATCH (i:Individu)-[:SAME_AS]->(e:Individu {nom: 'miguel', cognom: 'ballester'})
RETURN i, e
```



- e. Mostra totes les persones relacionades amb "antonio farran". Mostra la informació en forma de taula: el nom, cognom1, cognom2, i tipus de relació.

```
MATCH (i:Individu {nom: 'antonio', cognom: 'farran'})-[:rel]->(e:Individu)
RETURN e.nom AS Nom, e.cognom AS Cognom1, e.cognom2 AS Cognom2,
type(rel) AS Tipus
```

	Nom	Cognom	Cognom2	Tipus
1	"antonio"	"farran"	"sole"	"SAME_AS"
2	"antonio"	"farran"	"sele"	"SAME_AS"
3	"esperanza"	"farran"	"colet"	"FAMILIA"
4	"esperanza"	"colet"	"gavarro"	"FAMILIA"
5	"catalina"	"farran"	"colet"	"FAMILIA"
6	"francisco"	"farran"	"colet"	"FAMILIA"

f. Llisteu totes les relacions familiars que hi ha.

```
MATCH (i:Individu)-[rel:FAMILIA]->(e:Individu) WITH collect(DISTINCT
rel.relacio) AS Relacio RETURN Relacio
```

```
1 ["cabeza", "null", "hijo", "hija", "yerno", "nieto", "hermano", "nieta", "nuera", "madre", "hermana", "jefe", "esposa", "filla", "fill", "altres", "net", "parents", ']
```

g. Identifiqueu els nodes que representen el mateix habitatge (carrer i numero) al llarg dels anys de Sant Feliu del Llobregat (SFLl). Mostreu el resultat dels habitatges que tingueu totes dues informacions (carrer i numero), el nombre total d'habitatges, el llistat d'anys dels padrons i el llistat de les Ids de les llars. Ordeneu de més a menys segons el total d'habitatges i mostreu-ne els 10 primers.

```
MATCH (h:Habitatge {municipi:'SFLl'}), (h1: Habitatge {municipi: 'SFLl'})
WHERE h.carrer = h1.carrer AND h.numero = h1.numero RETURN h.carrer as
Carrer, h.numero as Numero, size(collect(DISTINCT(h.anyPadro))) as
Numero_total_habitatges, collect(DISTINCT(h.anyPadro)) as llistat_anys,
collect(DISTINCT(h.id_hab)) as llistat_ids ORDER BY Numero_total_habitatges
DESC, h.carrer LIMIT 10
```

	Carrer	Numero	Numero_total_habitatges	llistat_anys	llistat_ids
1	"creus"	9	5	[1833, 1839, 1878, 1881, 1889]	[150, 610, 467, 367, 347]
2	"creus"	16	5	[1833, 1839, 1878, 1881, 1889]	[158, 616, 473, 372, 388]
3	"creus"	23	5	[1833, 1839, 1878, 1881, 1889]	[166, 624, 480, 380, 353]
4	"creus"	18	5	[1833, 1839, 1878, 1881, 1889]	[160, 618, 475, 374, 389]
5	"iglesia"	11	5	[1833, 1839, 1878, 1881, 1889]	[106, 732, 523, 520, 488]
6	"iglesia"	9	5	[1833, 1839, 1878, 1881, 1889]	[104, 729, 730, 521, 517, 487]
7	"iglesia"	2	5	[1833, 1839, 1878, 1881, 1889]	[96, 722, 515, 510, 498]
8	"iglesia"	1	5	[1833, 1839, 1878, 1881, 1889]	[94, 721, 514, 509, 483]
9	"iglesia"	4	5	[1833, 1839, 1878, 1881, 1889]	[98, 725, 517, 512, 499]
10	"iglesia"	3	5	[1833, 1839, 1878, 1881, 1889]	[97, 723, 724, 516, 511, 484]

- h. 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]-(a:Individu)<-
[rel:FAMILIA]-(e:Individu)
WHERE rel.relacio_h=~"f.*"
WITH e, size(collect(a.id_ind)) AS num
WHERE num > 3
RETURN e.nom AS nom, e.cognom AS cognom, e.cognom2 AS cognom2, num
ORDER BY num DESC LIMIT 20;
```

nom	cognom	cognom2	num
"pablo"	"astudch"	"juli"	7
"jose"	"ole"	"domenech"	6
"pedro"	"juvent"	"parral"	6
"jose"	"canals"	"ole"	6
"pedro"	"bargal"	"legible"	6
"jose"	"canals"	"mala"	6

- i. 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.

```
MATCH
(a:Individu)-[rel:FAMILIA]->(e:Individu)-[:VIU]->(h:Habitatge{municipi:
"SFL",anyPadro:1881})<-[:VIU]-(a)
WHERE rel.relacio_h=~"f.*"
RETURN count(a) as total_fills, count(distinct h) as num_llars,
round(count(a)/toFloat(count(distinct h)), 2) as mitjana;
```

total_fills	num_llars	mitjana
1292	465	2.78

- j. Per cada any que hi ha a la base de dades, quin és el carrer amb menys habitants de Sant Feliu de Llobregat?

```
MATCH (a:Individu)-[:VIU]->(h:Habitatge{municipi: "SFL"})
WITH h.anyPadro as any, h.carrer as carrer, count(a) as total
ORDER BY total
WITH any, collect(carrer)[0] as min_carrer
RETURN any, min_carrer
ORDER BY any;
```

any	map_carrer
1833	"carrera de la part de molins de rey"
1838	"carrera de bama"
1839	"casas del bonany"
1878	"carrera"
1881	"Carrera"
1889	"y n anterior"

4. EXERCICI 3

En aquest exercici analitzarem les dades del graf per entendre millor l'estructura de les dades. Els següents apartats tenen com objectiu orientar-vos respecte a com utilitzar algunes de les eines que ofereix Neo4J.

En primer lloc creem el graf amb la comanda project (degut a la versió que tenim del gds):

```
CALL gds.graph.project('fams',['Habitatge','Individu'],['FAMILIA','SAME_AS','VIU'])
```

```
YIELD graphName AS graph, nodeProjection, nodeCount AS nodes, relationshipCount AS rels
```

	graph	nodeProjection	nodes	rels
1	"fams"	{ "Habitatge": { "label": "Habitatge", "properties": { } }, }, "Individu": { "label": "Individu", "properties": { } } } }	21288	34647

a) Estudi de les components connexes (cc) i de l'estructura de les component en funció de la seva mida. A continuació uns indiquem algunes consultes que podeu fer per explorar les dades:

- Taula agrupant els resultats segons la mida de la cc.
- Distribució de tipus de nodes (Individu o Habitatge) segons la mida de la cc.
- Per cada municipi i any el nombre de parelles del tipus: (Individu)—(Habitatge)
- quantes components connexes no estan connectades a cap node de tipus 'Habitatge'.

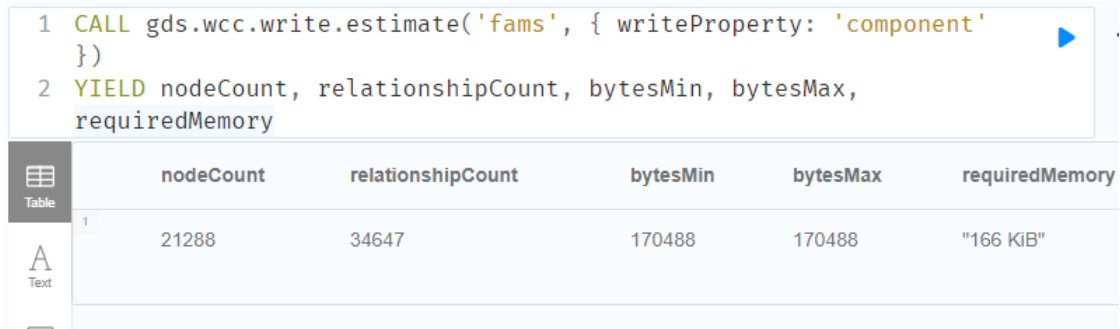
Aquestes consultes són només una orientació del tipus d'exploració de dades que podeu fer. Tant si feu aquestes com d'altres, heu d'acompanyar-les d'una motivació (que preteneu saber amb la consulta) i d'una explicació dels resultats. Acompanyeu aquesta explicació

amb el codi de la consulta i el resultat obtingut. (Indicació: utilitzeu la funció wcc en mode 'stream')

Abans d'estudiar les cc i l'estructura mirem quins són els costos de memòria:

```
CALL gds.wcc.write.estimate('fams', { writeProperty: 'component' })
```

```
YIELD nodeCount, relationshipCount, bytesMin, bytesMax, requiredMemory
```



The screenshot shows a query editor with two lines of code. Below the editor, a table displays the results. The table has five columns: nodeCount, relationshipCount, bytesMin, bytesMax, and requiredMemory. The first row of data shows values for a single component.

	nodeCount	relationshipCount	bytesMin	bytesMax	requiredMemory
1	21288	34647	170488	170488	"166 KiB"

Ara sí, veiem que la memòria necessària és realment baixa i comencem a estudiar les components connexes (cc):

""The WCC algorithm finds sets of connected nodes in an undirected graph, where all nodes in the same set form a connected component""

```
CALL gds.wcc.stream('fams')
```

```
YIELD componentId, nodeId
```

```
RETURN componentId, size(collect(nodeId)) AS mida, collect(nodeId) AS nodes
```

```
ORDER BY mida DESC
```



```

1 CALL gds.wcc.stream('fams')
2 YIELD componentId, nodeId
3 RETURN componentId, size(collect(nodeId)) AS mida, collect(nodeId) AS nodes
4 ORDER BY mida DESC

```

	componentId	mida	nodes
1	12	7613	[12, 64, 65, 66, 69, 70, 71, 75, 78, 81, 84, 88, 89, 91, 92, 106, 107, 109, 110, 113, 121, ...]
2	79	138	[79, 80, 155, 370, 394, 564, 566, 594, 2405, 2980, 3037, 3139, 3140, 3141, 3145, 3199, ...]
3	315	134	[315, 316, 417, 469, 602, 606, 667, 670, 694, 717, 2310, 3125, 3126, 3197, 3201, 3204, ...]
4	353	134	[353, 380, 426, 427, 428, 444, 560, 570, 584, 1963, 1981, 1987, 1988, 2979, 3110, 3180, ...]
5	382	130	[382, 383, 433, 553, 629, 640, 790, 876, 1684, 1685, 1714, 2395, 2396, 3176, 3186, 3187, ...]
6	108	108	[108, 153, 170, 171, 206, 327, 349, 401, 531, 577, 615, 621, 2224, 2965, 2977, 2999, 3000, ...]

Aquesta comanda ens retorna els conjunts de nodes relacionats de mida més gran a mida més petita (sense tenir en compte quin tipus de node són).

Ara busquem quin és el municipi amb més grups connexos (més famílies):

```

CALL gds.wcc.stream('fams')
YIELD nodeId, componentId
WHERE gds.util.asNode(nodeId).municipi is not null
RETURN distinct gds.util.asNode(nodeId).municipi AS municipi, size(collect(distinct componentId)) AS midacomp
ORDER BY municipi, midacomp

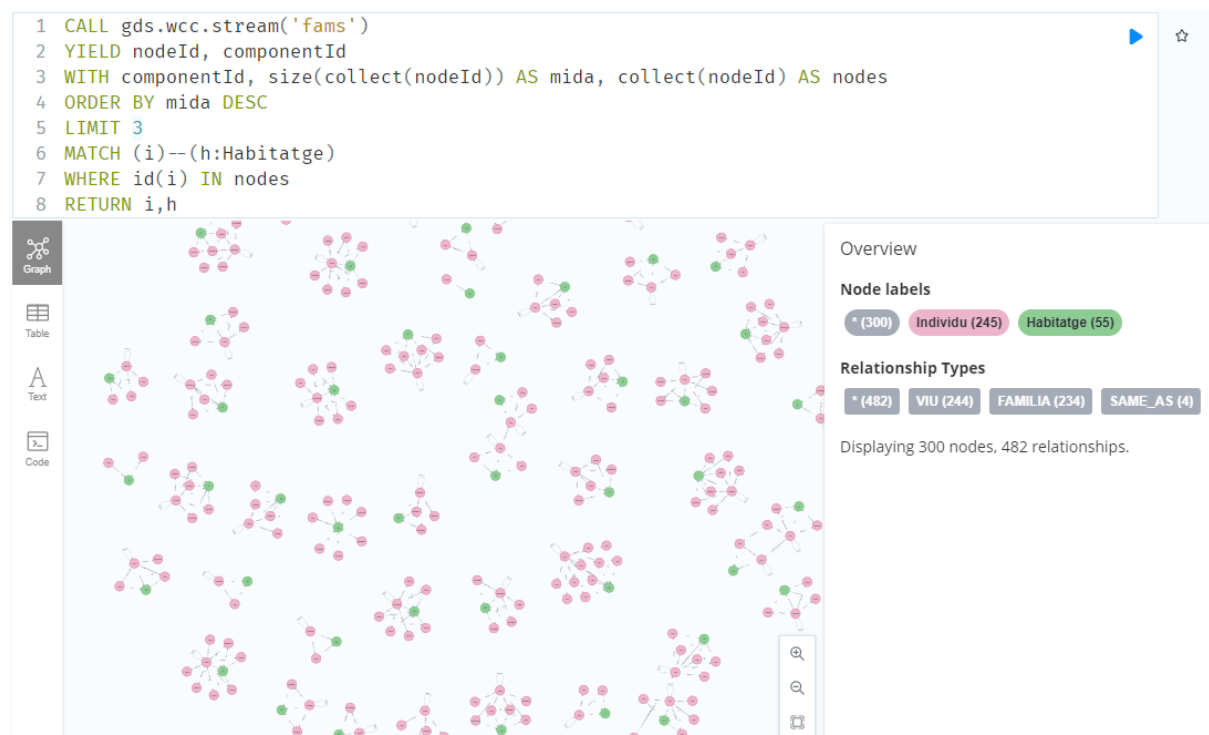
```

	municipi	midacomp
1	"CR"	60
2	"SFLI"	1056
3	"null"	974

Veiem que el Municipi de Sant Feliu de Llobregat és el municipi amb més famílies a la base de dades.

Ara estudiem la distribució de famílies per cada habitatge. Amb la següent comanda veiem els nodes individu distribuïts sobre cada habitatge:

```
CALL gds.wcc.stream('fams')
YIELD nodeId, componentId
WITH componentId, size(collect(nodeId)) AS mida, collect(nodeId) AS nodes
ORDER BY mida DESC
LIMIT 3
MATCH (i)--(h:Habitatge)
WHERE id(i) IN nodes
RETURN i,h
```



Amb aquesta comanda el que veiem són “subgrafs” en que tots els individus apunten a l’habitatge en que resideixen. La majoria d’individus comparteixen habitatge, però si busquem observem que n’hi ha que resideixen sols.

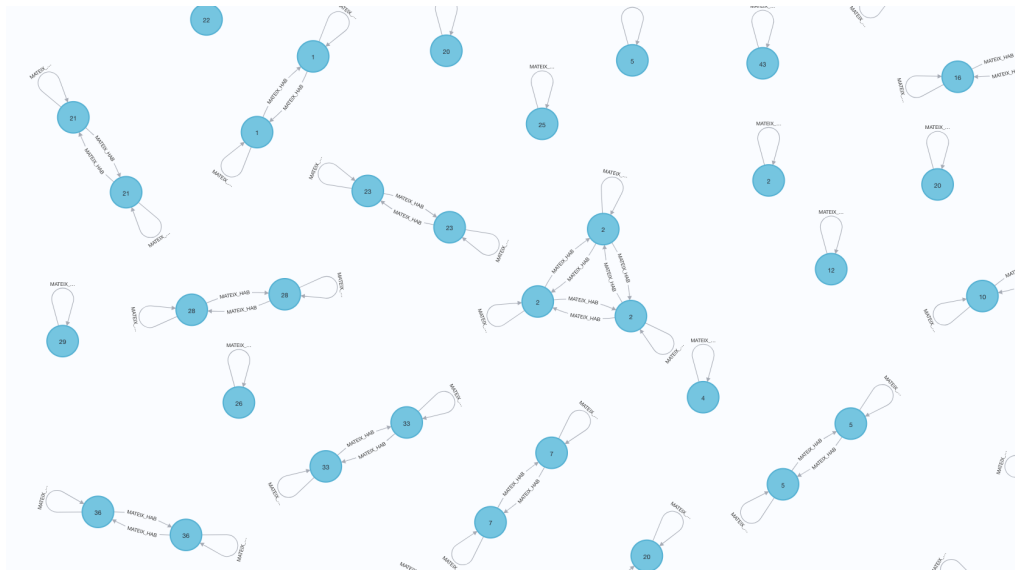
b) Semblança entre els nodes. Ens interessa saber quins nodes són semblants com a pas previ a identificar els individus que són el mateix (i unirem amb una aresta de tipus SAME_AS). Abans de fer aquest anàlisi:

- Determineu els habitatges que són els mateixos al llarg dels anys. Afegiu una aresta amb nom “MATEIX_HAB” entre aquests habitatges. Per evitar arestes duplicades feu que la aresta apunti al habitatge amb any de padró més petit.

```

MATCH (h:Habitatge),(h2:Habitatge)
where h.numero=h2.numero AND h.carrer=h2.carrer
merge (h)-[rel:MATEIX_HAB]->(h2)
RETURN h
order by h.anyPadro asc

```



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

Creem el graf amb la comanda project (degut a la versió que tenim del gds):

```

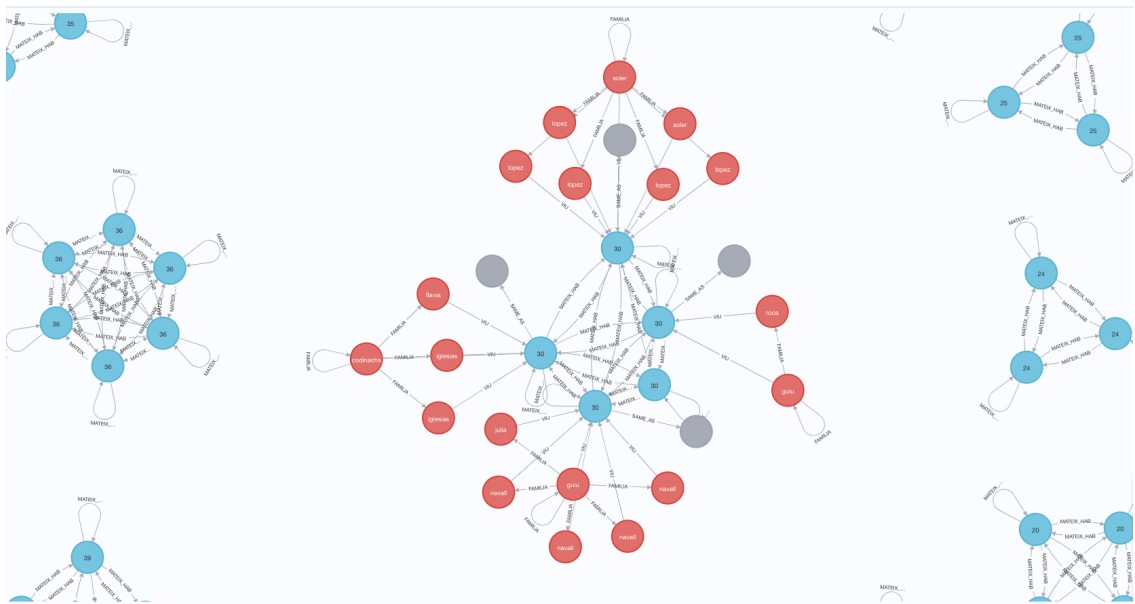
CALL
gds.graph.project('Ind_Hab',['Habitatge','Individu'],['FAMILIA','VIU','MATEIX_HAB'])
YIELD graphName AS graph, nodeProjection, nodeCount AS nodes, relationshipCount
AS rels

```

1	"Ind_Hab"	<pre>{ "Habitatge": { "label": "Habitatge", "properties": { } }, "Individu": { "label": "Individu", "properties": { } } } }</pre>	21288	34046
---	-----------	---	-------	-------

Visualització del graph:

MATCH(Ind_Hab) **RETURN**(Ind_Hab)



- **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.**

Calculem la similaritat entre els nodes del graph creat anteriorment i creem les relacions SIMILAR en aquells casos on la similaritat superi el 0.45.

```
CALL gds.nodeSimilarity.write('Ind_Hab',{
  writeRelationshipType:'SIMILAR',
  writeProperty:'score',
  similarityCutoff:0.45,
  topK:5
})
YIELD nodesCompared, relationshipsWritten
```

1	15245	49312
---	-------	-------

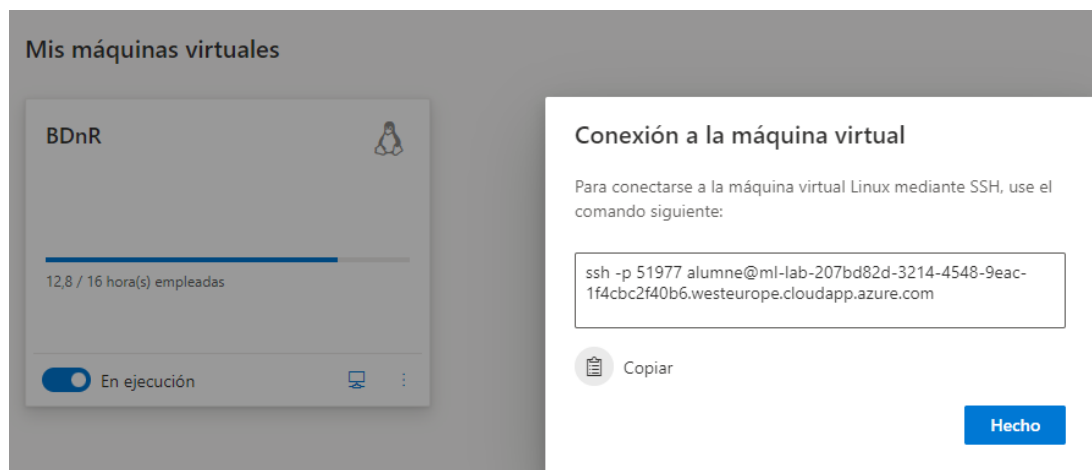
Aquí tendríamos nuestra nodos similares (15245) y sus relaciones de similitud(49312). De media sabemos que cada nodo tiene 3 relaciones (esto se deduce al dividir nuestros nodos por sus relaciones que aproximadamente dan 3 relaciones por nodo).

Per fer aquest exercici us podeu ajudar i inspirar de l'exercici guiat de "node similarity" que trobareu a :play 4.0-intro-graph-algos-exercises)

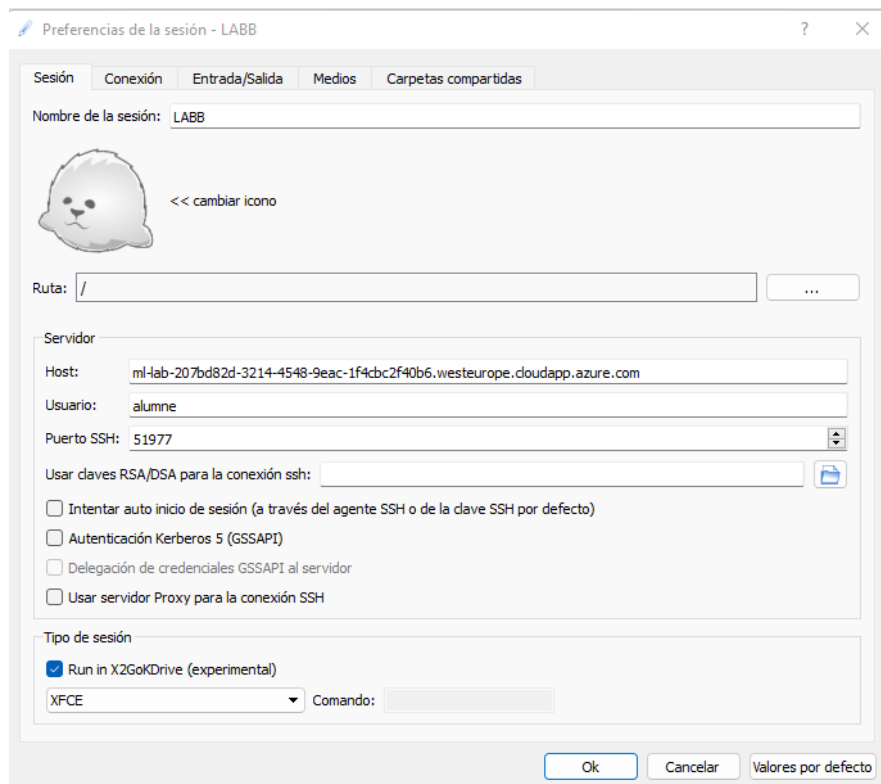
5. Connexió al núvol

Hem tingut problemes per fer la connexió al núvol. A continuació posarem els passos que hem seguit:

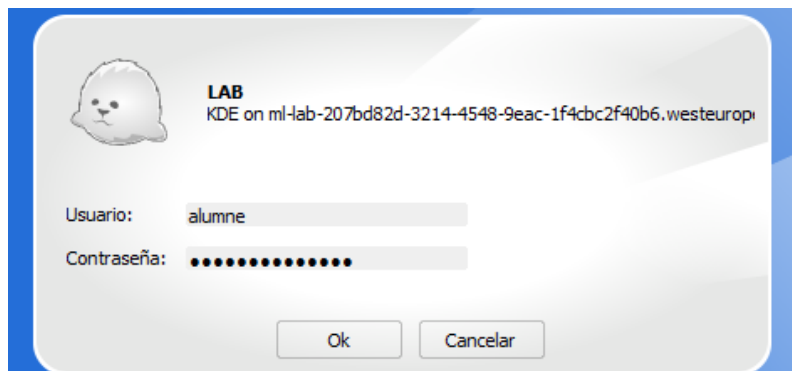
- Entrem a microsoft azure i engegum la màquina virtual:



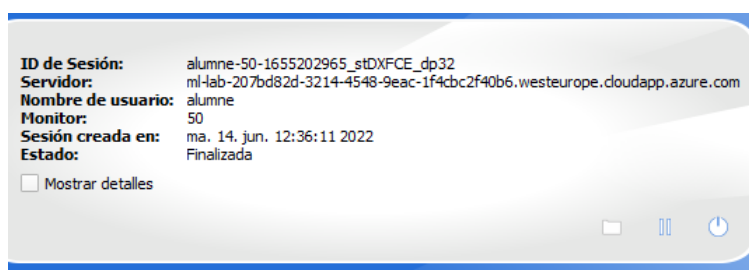
- Obrim el X2GoClient i creem una nova sessió amb les dades de SSH:



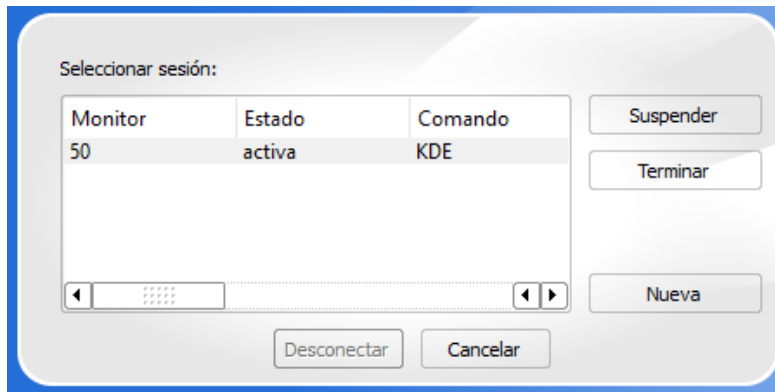
- Posem la contrasenya:



- I al cap d'una estona de posar la contrasenya surt això i es tanca la sessió:



- Alguna altra vegada que hem provat ens ha sortit això, però no ens ha deixat fer res més:



6. Treball en equip

Ens hem repartit el treball de la següent manera:

- Àlex i Júlia: exercici 1, exercici 2: 1-4 queries, exercici 3a
 - Júlia: 1, 1-2 queries
 - Àlex: 3-4 queries, 3a
- Cris i Esteban: exercici 2: 5-10 queries, exercici 3b
 - Cris: 5-7 queries, 3b
 - Esteban 8-10 queries, 3b