2 DE JUNIO DE 2021

PROJECTE NEO4J

BASE DE DADES NO RELACIONALS

MIQUEL MONGE DALMAU, NIU: 1565230
MARIA BARNILS SAGUÉS, NIU: 1570110
ORIOL PRAT FONT, NIU: 1565096
ADRIANA MURILLO GONZALO, NIU: 1564419

Contingut

1. Descomposició de tasques

Per a desenvolupar el projecte hem fet ús del repositori de *GitHub*, on s'han mantingut i actualitzat tots els scripts, tenint sempre l'última versió d'aquell moment penjada, per a que tots els integrants del grup puguem treballar sobre les mateixes dades en les diferents tasques.

Per a les tasques, les hem descompost en dos grans blocs, importació de la base de dades i resolució d'exercicis.

BLOCS	**TASQUES**

Importació dades	- **Generar script *Cypher***
	  - Carregar dades amb format *.csv*
	  - Generar nodes i relacions (amb les característiques corresponents).
	  - Evitar duplicats a les dades (múltiples execucions).

Resolució exercici	- **Resoldre l'exercici amb una consulta Cypher**
	  - Explicar la consulta i mostrar els resultats
	  - Validar resultat amb
	  - Joc De Proves
	  - Les dades de la nostra BDnR

Tots els membres del grup hem realitzat conjuntament els diferents blocs que conformen el nostre projecte. Per tal de fer-ho hem realitzat reunions a través de plataformes de videoconferència, on hem realitzat les diferents tasques que se'ns demanava a l'enunciat i, a la vegada, hem anat discutint el que havia realitzat cadascun de nosaltres per tal de revisar-ho, comparar-ho i quedar-nos amb la millor versió.

Destacar que al tenir totes les tasques actualitzades al repositori, qualsevol membre del grup podia col·laborar en qualsevol tasca si ho creia necessari.

Enllaç al repositori.

2. Importació de la base de dades

2.1. Exploració de les dades

Les dades que importarem estan en format *csv (comma-separated-values).* En aquest format, s'utilitzen comes per separar els valors, en cada línia de l'arxiu es troba un registre de dades.

En total tenim cinc arxius que haurem d'importar, alguns d'aquest contenen informació per als nodes (INDIVIDUAL, HABITATGES) i alguns altres contenen informació de les relacions d'aquestes, les arestes (FAMILIA, SAME_AS, VIU).

Cal destacar que per a fer la importació de dades de forma local, és a dir, sense necessitat d'una connexió a internet, fa falta que les dades estiguin dins de la carpeta **/import** on està emmagatzemada la nostra base de dades.

2.2. Importació de les dades

Farem la importació de les dades mitjançant un script Chyper que llegirà les dades, crearà les relacions i constraints per evitar duplicar les dades si el script s'executa més d'un cop.
Observació: Per evitar un ús abusiu de captures de pantalles i omplir el document de soroll, en comptes d'enganxar el script en aquest informe, es pot visualitzar a través del repositori de GitHub que hem comentat en l'inici i que hem fet servir al llarg del projecte. Enllaç al script d'importació de les dades.

Les quatre primeres sentències estan pensades perquè el script funcioni correctament en cas d'execucions repetides. En aquestes línies el que fem és esborrar el graf, ja que el tornarem a afegir en aquesta execució. A més, esborrem les constraint (en cas que existeixin) perquè no hi hagi errors d'execució.

Quant a les constraints, aquestes ens permeten controlar que n s'afegeixin nodes repetits. En total creem dos: UniqueIdLLar per als nodes Habitatge i UniqueNameYear per als nodes Individual. En referència a la primera, cada node de tipus Habitatge s'identifica únicament dels altres amb els camps: id_llar, any_padro i municipi. En cas que s'afegeixi un nou Habitatge on aquests tres camps estiguin presents ja a la nostra base de dades, aquest node no s'afegirà.
Fem servir la mateixa lògica per a la constraint dels nodes de tipus Individual, en aquest cas només hi ha una clau única (camp que identifica de manera única a cada Individu), aquest camp és el id.
Amb relació a la importació de les dades dins dels fitxers .csv, destacar que tots aquests contenen el títol del contingut en la primera línia, per tant, utilitzem l'opció WITH HEADERS en la improtació. Cada arxiu té les seves restriccions per a crear les arestes, com a exemple explicarem la importació de l'arxiu same_as.csv i Individual.csv.

Per al primer, és un arxiu que emmagatzema informació sobre connexions/arestes del graf. El primer que fem és carregar les dades en memòria, i fem una consulta per crear les relacions en el graf. Aquesta consulta agafa dos nodes tipus Individual del graf, si es compleix que aquests dos nodes tenen el mateix id vol dir que són la mateixa persona, en aquest cas s'afegirà l'aresta SAME_AS entre ells. Destacar que neo4j llegeix les dades en tipus string, i nosaltres emmagatzemar el id com a un número enter (int), per tant, hem de transformar les dades que llegim a enter amb la funció toInteger().

Com a últim exemple explicarem la importació de les dades que emmagatzemen els nodes Individual. En aquest cas només importem aquells registres on el id dels individus no són NULL. Observem que aquesta és una de l'avantatge de les bases de dades no relacionals, en ser lliures d'esquema, tenim llibertat d'importar les dades d'aquesta manera. En cas que existeixi id, creem el node amb els seus camps: id, any, name, surname i second_surname.

3. Resolució de consultes

Cal destacar que la sortida de la majoria de les consultes obtenim el mateix resultat que en el document de joc de proves. En tot cas, per a les consultes número 7, 9 i 10, el resultat obtingut no correspon amb les dades esperades. Això és degut al fet que la importació de les dades feta per l'equip docent probablement no és igual a la nostra importació. És a dir, que les peculiaritats de la importació (constraints, tipus de les dades, duplicats, etc) s'han tractat de forma diferent.
En tot cas, per aquestes consultes hem validat que el resultat correspon a les dades que estan emmagatzemades en la nostra base de dades. És a dir, encara que diferent, podem considerar el resultat com a correcte.

3.1. Exercici 2

- **Consulta 1: Dels padró de 1866 de Castellví de Rosanes (CR), retorna el número d'habitants i la llista de noms. Elimina duplicats i nan.**

En aquesta consulta realitzem un MATCH i relacionem els nodes Individual i Habitatge, a través de la relació VIU.

Una vegada associada una nova variable a cadascun dels nodes, realitzem un WHERE per filtrar la nostra Base de Dades, el que volem tractar són aquelles dades on el municipi d'on viu la persona sigui 'CR', i que el cognom de la mateixa persona sigui igual a 'nan'.

Una vegada realitzat tot això, ens interessa mostrar el número d'habitants, el qual ho realitzem a partir de contar tots els cognoms de les persones de la nostra base (a través d'un count), i mostrar la llista dels noms a través d'un collect (el qual ens agrupa) i, a la vegada, també realitzem un distinct() el qual ens transforma una llista de duplicats en un únic conjunt.

Query:

```
MATCH (ind:Individual)-[:VIU]→(p:Habitatge)
WHERE p.municipi = 'CR' and ind.surname<>'nan'
RETURN count(ind.surname) as `Número Habitants`, collect(DISTINCT ind.surname)
as `Llista noms`
```

Resultats:

	Número Habitants	Llista noms
1	336	["olle", "galceran", "suñol", "rusell", "julibert", "bargallo", "anglada", "ros", "julia", "vila", "julivert", "gaset", "rumeu", "parera", "canals", "canald", "dalmases", "rigol", "calaf", "llimona", "astruch", "rafuls", "astrade", "esparchy", "farre", "volta", "llopart", "gallofre", "masana", "farres", "fqrre", "amat", "casanovas", "anducas", "capellades", "cadafaly", "plana", "ilegible", "farreny", "jarrey", "ventura", "domenech", "aregay", "gibert", "vives", "marcade", "cardus", "pañella", "tovella", "mitjans", "rios", "calafi", "galofre", "nicolau", "pujol", "anmatller", "alegre", "rabantos", "bley", "armengol", "bogoña", "gol", "salto", "roca", "pascual", "pujadas", "valls", "esteva", "juntanet", "cartro", "garriga", "rafols", "claramunt", "escayol"]

- **Consulta 2: 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.**

En aquesta segona consulta també hem de tractar la relació VIU i, com aquesta relaciona els nodes Individual i Habitatge, tornem a realitzar un MATCH de la mateixa manera que hem fet a la consulta1 (també assignant-li variables a cadascun dels nodes tractats).

En concret se'ns demana filtrar la nostra Base de Dades per tal de que el municipi en el qual viu la persona sigui igual a 'SFLL', i que l'any padró de l'habitatge sigui més petit que 1840. En aquest cas, per realitzar aquest filtratge, tornem a realitzar un WHERE amb aquestes dues condicions definides anteriorment, i unides a partir d'un AND.

A continuació realitzem un RETURN, on se'ns demana que retornem l'any de padró, el qual per retornar-ho fem un distinct per tal d'evitar duplicats, i també hem de mostrar el id de la llar, on concatenem un collect i un distinct, degut a que volem agrupar a partir del id de la llar el qual es troba al node Habitatge, i a la vegada, volem eliminar duplicats d'aquests.

Per últim, realitzem un ORDER BY de l'any padró del node Habitatge, el qual ens ordenarà aquests de forma ascendent, és a dir, de més petit a més gran. Quan en un ORDER BY no s'especifica com volem que sigui aquesta ordenació, per defecte es realitzarà de forma ascendent, si volguéssim realitzar-la de forma descendent, al final d'aquest ORDER BY hauríem d'especificar-li que volem que sigui DESC.

Query:

```
MATCH (h:Habitatge)<-[:VIU]-(p:Individual)
WHERE h.any_padro < 1840 AND h.municipi = 'SFLL'
RETURN DISTINCT h.any_padro as `Any del padró`, collect(distinct(h.id_llar)) as
`id de la llar`
ORDER BY h.any_padro
```

Alguns resultats:

	Any del padró	id de la llar
1	1833	[95, 99, 101, 103, 105, 107, 109, 111, 94, 96, 97, 98, 108, 100, 104, 102, 106, 110, 113, 115, 118, 119, 120, 122, 124, 126, 128, 129, 130, 132, 134, 135, 136, 137, 140, 142, 144, 112, 114, 116, 117, 121, 123, 125, 127, 131, 133, 138, 139, 141, 143, 145, 10, 11, 15, 16, 18, 20, 22, 24, 26, 29, 32, 34, 35, 36, 38, 39, 40, 44, 46, 47, 50, 51, 52, 56, 57, 58, 59, 12, 14, 17, 19, 21, 23, 25, 27, 28, 30, 31, 33, 37, 41, 42, 43, 45, 48, 49, 53, 54, 55, 13, 60, 61, 64, 65, 66, 67, 68, 72, 73, 74, 75, 76, 77, 78, 80, 81, 82, 83, 86, 88, 89, 91, 93, 62, 63, 69, 70, 71, 79, 84, 85, 87, 90, 92, 146, 147, 148, 156, 149, 151, 152, 154, 155, 157, 159, 161, 162, 164, 165, 167, 150, 153, 158, 160, 163, 166, 168, 169, 170, 171, 173, 174, 176, 178, 180, 172, 175, 179, 177, 181, 182, 184, 186, 188, 190, 192, 193, 194, 197, 198, 200, 204, 205, 207, 208, 209, 210, 183, 185, 187, 189, 191, 195, 196, 199, 201, 202, 203, 206, 211, 212, 213, 214, 215, 220, 222, 216, 217, 218, 221, 230, 231, 219, 223, 224, 225, 226, 227, 228, 233, 229, 232, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 253, 258, 260, 261, 250, 251, 252, 254, 255, 256, 257, 259, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 278, 284, 277, 279, 280, 281, 282, 283, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 302, 309, 310, 311, 312, 313, 297, 298, 299, 300, 301, 303, 304, 305, 306, 307, 308]
2	1838	[321, 324, 326, 328, 320, 322, 323, 325, 327, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 314, 315, 316, 317, 318, 319]

- **Consulta 3: 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.**

Aquesta consulta es divideix en dos subapartats, en els quals se'ns demana pràcticament la mateixa consulta, l'únic que canvia és el RETURN. A continuació, ho explicarem.
Se'ns demana tractar la relació VIU, la qual relaciona els nodes Individual i Habitatge. En aquest cas, també li associem una variable a cadascun d'ells, per tal de poder realitzar el filtratge posteriorment. Això ho fem, tal com portem fent fins ara, a partir d'un MATCH on, en el node Individual, li especifiquem que volem que el nom d'aquest sigui 'rafael' i el cognom sigui 'marti'. No obstant això, per la relació VIU li establim dos nodes Individual, els quals seran diferents. I, a més, volem realitzar una altra relació, que serà l'anomenada FAMILIA, que relaciona els dos nodes Individual creats anteriorment (en la relació VIU).
Se'ns especifica que filtrem la nostra consulta de tal forma que l'any padró de l'habitatge que estem tractant sigui igual a 1838, i que el municipi d'aquest mateix sigui igual a 'SFLL'. Aquesta especificació la realitzem, de la mateixa manera que portem fent fins ara, a partir d'un WHERE.
Per últim, hem de mostrar el que se'ns demana a partir d'un RETURN. No obstant això, depenent en quin apartat ens trobem ho he de fer d'una manera diferent:

1) Mostrarem el resultat en forma de graf, per tant, ael nostre RETURN únicament hauriem de retornar els nodes Individual. Això es faria retornant la variable que prèviament (en el MATCH) li hem especificat a cadascun d'ells.
2) En aquest altre apartat, se'ns demana que mostrem els resultats en forma de llista. Per tant, en aquest cas no ens és suficient mostrar els nodes, sinó que hem d'especificar quins són els camps que volem mostrar de cadascun d'ells. En aquest cas en concret, el que mostrem és el nom del primer individu establert al MATCH, i els convivents, els quals surten de realitzar un collect (per tal d'agrupar) dels noms del segon individu.

Query:

a) En forma de graf:

```
MATCH (ind1:Individual {name:"rafel", surname:"marti"})-[:VIU]→(h:Habitatge)←
[:VIU]-(ind2:Individual), (ind1)-[:FAMILIA]→(ind2)
WHERE h.any_padro = 1838 AND h.municipi = "SFLL"
RETURN ind1, ind2
```

b) En forma de llista:

```
MATCH (ind1:Individual {name:"rafel", surname:"marti"})-[:VIU]→(h:Habitatge)←
[:VIU]-(ind2:Individual), (ind1)-[:FAMILIA]→(ind2)
WHERE h.any_padro = 1838 AND h.municipi="SFLL"
RETURN ind1.name as `nom`, collect(ind2.name) as `convivents`
```

Resultats:

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

- **Consulta 4: 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.**

En aquesta consulta se'ns demana retornar tots aquells camins on apareix l'individu anomenat "Miguel Ballester", a més, hem de fer servir la relació SAME_AS.

Per tal de fer-ho, creem una variable que anomenarem "path", la qual representarà el camí entre dos nodes Individual, on un d'ells serà el node que estem tractat, i aquests dos nodes estaran relacionats a través de la relació SAME_AS.

Per establir que un node Individual sigui el que estem tractant, li establim els atributs name and surname adequats.

Per últim, realitzem un RETURN d'aquesta variable "path" creada al MATCH.

Query:

```
MATCH path= ((i:Individual{name:'miguel', surname:'ballester'})-[:SAME_AS]-
(i2:Individual))
RETURN path
```

Alguns resultats:


```
╒═══════════════════════════════════════════════════════════════════════════╕
"path"                                                                      
╞═══════════════════════════════════════════════════════════════════════════╡
[{"name":"miguel","id":18812684,"year":1881,"surname":"ballester","sec
ond_surname":"andreu"},{},{"name":"miguel","id":18780202,"year":1878,"
surname":"balleste","second_surname":"andreu"}]                             
├───────────────────────────────────────────────────────────────────────────┤
[{"name":"miguel","id":18812684,"year":1881,"surname":"ballester","sec
ond_surname":"andreu"},{},{"name":"miquel","id":18570153,"year":1857,"
surname":"balleste","second_surname":"y andreu"}]                           
├───────────────────────────────────────────────────────────────────────────┤
[{"name":"miguel","id":18812684,"year":1881,"surname":"ballester","sec
ond_surname":"andreu"},{},{"name":"miguel","id":18892687,"year":1889,"
surname":"ballester","second_surname":"andreu"}]                            
├───────────────────────────────────────────────────────────────────────────┤
[{"name":"miguel","id":18812684,"year":1881,"surname":"ballester","sec
ond_surname":"andreu"},{},{"name":"miguel","id":18780202,"year":1878,"
surname":"balleste","second_surname":"andreu"}]                             
└───────────────────────────────────────────────────────────────────────────┘
```


- **Consulta 5: Mostra totes les persones relacionades amb "antonio farran". Mostra la informació en forma de taula: el nom, cognom1, cognom2, i tipus de relació.**

En aquesta consulta utilitzem un MATCH per relacionar dos nodes Individual que estiguin relacionats de qualsevol manera.

A continuació realitzem un WHERE per filtrar la cerca. El que farem és buscar només a les persones amb el nom 'antonio' i el primer cognom 'farran'.

El que volem retornar és el nom de les persones relacionades amb 'antonio farran', juntament amb els seus primer i segons cognoms i el tipus de relació entre totes dues persones. Aquests resultats els retornarem ordenant segons el nom de manera ascendent.

Query:

```
MATCH (ind1:Individual)-[rel]-(ind2:Individual)
WHERE toLower(ind1.name) = 'antonio' AND toLower(ind1.surname) = 'farran'
RETURN distinct ind2.name as `Nom`, ind2.surname as `Cognom1`,
ind2.second_surname as `Cognom2`, type(rel) as `Tipus`
ORDER BY ind2.name
```

Alguns resultats:

	Nom	Cognom1	Cognom2	Tipus
1	"antonio"	"ferran"	"sole"	"SAME_AS"
2	"antonio"	"ferran"	"sele"	"SAME_AS"
3	"antonio"	"farran"	"sole"	"FAMILIA"
4	"catalina"	"farran"	"colet"	"FAMILIA"
5	"esperanza"	"farran"	"colet"	"FAMILIA"

- **Consulta 6: Llisteu totes les relacions familiars que hi ha.**

En aquesta consulta fem un MATCH de dos individuals amb la relació FAMILIA entre ells.

A continuació fem un RETURN de les diferents relacions per així obtenir totes les relacions familiar del graf.

Query:

```
MATCH(p:Individual)-[r:FAMILIA]→ (:Individual)
RETURN DISTINCT(r.relacio) as `Relacio Familiars`
```

Resultats:

"Relacio Familiars"			
"esposa"	"germana"	"nebot"	"afillada"
"fill"	"cunyada"	"neboda"	"religios"
"filla"	"cunyat"	"ne"	"tiet"
"jefe"	"sogre"	"marit"	"jefe_3"
"null"	"familiar"	"pare"	"esposa_3"
"gendre"	"altres"	"mae"	"fill_3"
"net"	"servents"	"besnet"	"jefe_1"
"jove"	"parents"	"consogre"	"esposa_1"
"neta"	"afillat"	"besneta"	"fill_1"
"germa"	"ala"	"jefe_2"	"filla_2"
"mare"	"al"	"esposa_2"	"esposa_4"
	"fll"	"fill_2"	"fill_5"
			"fill_8"

- **Consulta 7: 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.**

Per aquesta consulta realitzem un match on es creen dos habitatges diferents, però no establim cap relació entre ells. El següent que fem és fer una comprovació de que els dos habitatges seleccionats comparteixin el mateix carrer i el mateix número.

Do tots els habitatges que compleixen aquestes condicions, retornem el nom del carrer i el número de l'habitatge, mostrem el nombre d'habitatges i els agrupem per anys. Per últim ordenem les respostes en funció del nombre d'habitatges i mostrem només els 10 primers resultats.

Query:

```
MATCH (h1:Habitatge {municipi:"SFLL"}), (h2:Habitatge {municipi:"SFLL"})
WHERE h1.carrer = h2.carrer AND h1.numero = h2.numero
RETURN h1.carrer, h1.numero, count(distinct(h1)) as total_habs, collect(distinct(h1.any_padro)),
collect(distinct(h1.id_llar))
ORDER BY total_habs DESC
LIMIT 10
```

Resultats:

"h1.carrer"	"h1.numero"	"total_habs"	"collect(distinct(h1.any_padro))"	"collect(distinct(h1.id_llar))"
"falguera"	5	9	[1833,1878,1881,1889]	[247,359,360,361,416,398,414,415,417]
"falguera"	3	8	[1833,1878,1881,1889]	[245,357,358,668,411,397,412,413]
"san antonio"	1	7	[1889]	[562,563,564,565,566,567,568]
"falguera"	22	7	[1833,1878,1881,1889]	[269,376,377,378,429,430]
"carretera"	28	6	[1838,1878,1881,1889]	[346,347,34,35,26,128]
"iglesia"	3	6	[1833,1839,1878,1881,1889]	[97,723,724,516,511,484]
"carretera"	194	6	[1878,1881,1889]	[229,230,231,232,211,205]
"carretera"	36	6	[1838,1878,1881,1889]	[354,355,41,42,34,133]
"carretera"	141	6	[1878,1881,1889]	[172,173,174,673,152,73]
"carretera"	98	6	[1878,1881,1889]	[112,664,97,162,98,99]

- **Consulta 8: 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.**

En aquesta consulta realitzem un MATCH i relacionem dos nodes de tipus individual a través de la relació FAMILIA. També fem una comprovació que els nodes individus seleccionats visquin a Castellví de Rosanes. Guardem tots aquests nodes en diferents variables per poder fer un filtratge.
El següent que fem és un where on comprovem que la relació entre els dos individus sigui una relació pare/fill-filla o mare/fill-filla.
En el return mostrem el que ens demana en l'enunciat, és a dir el nom, el cognom i el segon cognom dels caps de les famílies. També fem un recompte del nombre de fills. Aquest return el mostrem en funció del nombre de fills, és a dir, les famílies amb més fills es mostren primer.
Per últim fem un límit de 20, és a dir, que només mostrem els primers 20 resultats que apareixen, aquesta és la raó per la qual no cal comprovar que les famílies tinguin més de 3 fills en el where, ja que els vint primers resultats ja són superiors a 3.

Query:

```
MATCH (i1:Individual)-[rel:FAMILIA]→(i2:Individual), (i1)-[:VIU]→(:Habitatge{municipi:"CR"})
WHERE rel.relacio in ["fill","filla"]
RETURN i1.name as `nom`, i1.surname as `1r cognom`, i1.second_surname as `2n cognom`, count(i2) as `total`
ORDER BY total DESC
LIMIT 20
```

Alguns resultats:

"nom"	"1r cognom"	"2n cognom"	"total"
"pablo"	"astruch"	"julia"	7
"pedro"	"bargallo"	"ilegible"	6
"jose"	"olle"	"domenech"	6
"jose"	"canals"	"olle"	6
"jose"	"canals"	"mila"	6
"benito"	"julivert"	"parera"	6
"francisco"	"aregay"	"rigol"	5
"pablo"	"bargallo"	"armangol"	5
"jaime"	"jarrey"	"ilegible"	5
"jose"	"rafuls"	"mila"	5
"pedro"	"farres"	"rigol"	4

"jose"	"llopart"	"pareyada"	4
"ramon"	"canals"	"amat"	4
"juan"	"julibert"	"parera"	4
"pablo"	"canals"	"llimona"	4
"estevan"	"gallofre"	"bertran"	4
"jaime"	"gallofre"	"bartran"	4
"tomas"	"parera"	"roig"	4
"cristobal"	"olle"	"rabantos"	4
"jose"	"suñol"	"henan"	4

- **Consulta 9: 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.**

En aquesta consulta realitzem un MATCH i relacionem dos nodes de tipus individual a través de la relació FAMILIA, de la mateixa manera en la qual ho hem fet en la consulta anterior. També relacionem els dos nodes amb un tercer de tipus Habitatge a través de la relació VIU. A cada un d'aquests nodes els hi associem una variable per tal de poder realitzar un filtratge.
Després de fer el match, fem un where per filtrar les dades i quedar-nos només amb aquells habitatges registrats en l'any 1881 i en el municipi de Sant Feliu del Llobregat. També comprovem que el primer individu seleccionat sigui el pare o mare del segon individu. Aquesta última comprovació la fem mirant el tipus de relació que uneix als individus.
Per últim fem un return on fem un recompte del nombre de fills que hem trobat, el nombre de llars i la mitjana de fills que es van tenir durant aquell any.

Query:

```
MATCH (ind1:Individual)-[r:FAMILIA]→(ind2:Individual)-[:VIU]→(p:Habitatge)←[:VIU]-(ind1)
WHERE p.any_padro = 1881 AND p.municipi = 'SFLL' AND r.relacio IN ['fill', 'filla']
RETURN count(distinct ind1) as `total_fills`, count(distinct p) as `num_llars`, round(count(distinct
ind1)/toFloat(count(distinct p)), 2) as `mitjana`
```

Resultats:

"total_fills"	"num_llars"	"mitjana"
1235	580	2.13

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

En aquesta consulta realitzem un match on relacionem un node de tipus individu amb un altre de tipus habitatge a través de la relació VIU. També especifiquem que aquest habitatge es trobi en el municipi de Sant Feliu del Llobregat. Després fem un with on contem tots els nodes de tipus individu.

Retornem l'any del padró, i el nom de carrer que té el mínim nombre d'habitants. Per últim ordenem aquest resultats de manera ascendent, de forma que quedi el carrer amb menys nombre d'habitants en primera posició.

Query:

```
MATCH (I:Individual)-[:VIU]→(h:Habitatge {municipi:'SFLL'})
WITH h, count(I) as num_habitants
RETURN h.any_padro as year, collect((h.carrer))[..1] as street, min(num_habitants) as habitants
ORDER BY year ASC, habitants ASC, street ASC
```

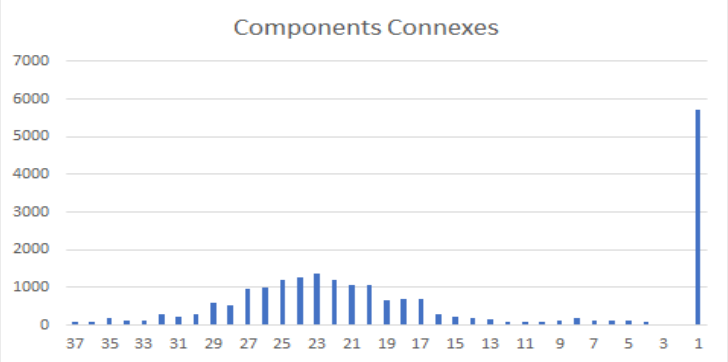
Resultats:

"year"	"street"	"habitants"
1833	["iglesia"]	5
1838	["carretera"]	9
1839	["iglesia"]	1
1878	["carretera"]	2
1881	["s antonio"]	7
1889	["carretera"]	5

3.2. Exercici 3

a) Estudi de les components connexes

Hem decidit crear un gràfic que representi el nombre de components connexes que hi ha en el gràfic agrupats segons la mida d'aquestes.



A partir del resultat obtingut se'ns han generat dubtes que ens han portat a fer un anàlisi més extens de les dades del graf.

- **Motivació**

La motivació principal per realitzar aquesta consulta és conèixer com són les relacions internes del nostre graf. Com es pot veure en la gràfica que hem generat, la component connexa més gran està formada per 37 nodes, però la majoria de components són de mida 1.

Aquesta consulta ens ajuda a veure què hi ha molts individus que no estan relacionats amb cap habitatge ni cap altre individu i, per tant, són una component connexa de mida 1. Tot i això, la gran majoria de nodes es troba entre l'interval [20, 27] pel que fa a la seva mida.

3.2.2 Estudi del municipi amb més components connexes de cada any

Hem decidit fer aquesta segona consulta sobre les components connexes per tal de saber quin municipi és el que té més components connexes al llarg dels anys registrats en la base de dades.

Per fer-ho hem realitzat la següent consulta, on comprovem que els municipis no siguin de tipus *null* i només retornem aquells municipis que tinguin la component connexa més gran.

```
CALL gds.wcc.stream('Ex3_h')
YIELD nodeId AS individual, componentId
WITH gds.util.asNode(individual) AS node1, componentId
WITH collect(node1) AS allNodes, componentId, node1
WHERE node1.municipi <> 'null'
RETURN Distinct node1.any_padro as any_padro,  node1.municipi as municipi, max(componentId) as  component_connexa
```

Aquest és el resultat que ens genera aquesta consulta.

Aquesta consulta ens serveix per saber si hi ha un clar municipi que lidera a tots quant a nombre d'habitants i en aquest cas, a partir de les dades podem intuir que el municipi de Sant Feliu del Llobregat és el municipi més poblat, ja que, en la majoria d'anys, és el que té més components connexes.

Continuant amb l'anàlisi sobre les component connexes, un altre estudi que podem fer és trobar aquelles components connexes que no estan connectades a cap node de tipus 'Habitatge'.

"any_padro"	"municipi"	"component_connexa"
1866	"CR"	11109
1881	"SFLL"	6418
1878	"SFLL"	6418
1889	"SFLL"	6418
1838	"SFLL"	6418
1833	"SFLL"	3474
1839	"SFLL"	13435

3.2.3 Estudi dels components connexes de mida 1

Hem realitzar una tercera consulta per saber quins son aquells nodes que no tenen família i que per lo tant, al mirar a quants nodes estan connectats veiem que la seva mida és 1 (ell mateix).

Per fer això creem el següent graf per només quedar-nos amb els nodes amb relacions de Família:

```
CALL gds.graph.create('Ex3_prova3',['Individual'], 'FAMILIA')
```

Un cop creat aquest graf que hem anomenat com a ex3_prova3 fem la següent consulta:

```
CALL gds.wcc.stream('Ex3_prova3')
YIELD componentId, nodeId
WITH componentId, collect(nodeId) as nodes,
size(collect(nodeId)) as mida
WHERE mida = 1
MATCH (n)
WHERE id(n) in nodes
return n;
```

La motivació per fer aquesta consulta és saber quins són aquells nodes que no tenen connexió i que per lo tant no ens servirien si volguéssim transmetre un missatge per exemple, i volguéssim arribar al major nombre de nodes partint d'un sol.

Per últim, podem analitzar aquelles persones que més temps porten dins de la base de dades. Això es interesant ja que dins d'aquesta no hi ha cap valor emmagatzemat que ens doni informació sobre l'antiguitat de les persones en la base de dades. Si sabem que la recopilació es fa cada 5 anys, totes aquelles persones amb almenys una relació de SAME_AS hauran estat 5 anys en la base de dades. Fem la query per saber quines persones tenen aquesta relació i quants cops la tenen.

Query:

```
MATCH (p:Individual)-[:SAME_AS]-(q:Individual)
WITH p {.name, .surname} as Nom, p.id as ID, p
RETURN ID as `Person ID`, size(collect(Nom)) as Num, collect(Nom) as Noms, p.year
ORDER BY Num DESC
```

Resultats:

"Person ID"	"Num"	"Noms"	"p.year"
18280376	6	[{"name":"jph","surname":"campderros"},{"name":"jph","surname":"campderros"},{"name":"jph","surname":"campderros"},{"name":"jph","surname":"campderros"},{"name":"jph","surname":"campderros"},{"name":"jph","surname":"campderros"}]	1828
18330752	6	[{"name":"jph","surname":"campderros"},{"name":"jph","surname":"campderros"},{"name":"jph","surname":"campderros"},{"name":"jph","surname":"campderros"},{"name":"jph","surname":"campderros"},{"name":"jph","surname":"campderros"}]	1833
18391290	6	[{"name":"jph","surname":"campderros"},{"name":"jph","surname":"campderros"},{"name":"jph","surname":"campderros"},{"name":"jph","surname":"campderros"},{"name":"jph","surname":"campderros"},{"name":"jph","surname":"campderros"}]	1839

Aquestes son algunes de les persones, com hem ordenat en ordre decreixent, ens sortiran els valors és grans al principi, per tant, veiem que el màxim de relacions SAME_AS es de 6. Per tant, aquest primers 3 nodes, entre altres, porten mínim 30 anys emmagatzemades en la base de dades (6 relacions * 5 ANYS cada padró).

b) Semblança entre els nodes

Comencem creant una nova relació *aresta* entre els nodes de tipus *Habitatge*. Aquesta relació s'anomena *MATEIX_HAB* i relaciona aquells habitatges que es poden considerar iguals.

Per tal de que dos nodes *Habitatge* es considerin iguals, les seves propietats número, municipi i carrer han de ser iguales. No obstant això, per tal d'evitar que es creïn relacions d'un habitatge amb ell mateix (en el cas que aparegui en diferents anys) hem d'afegir una condició que restringeixi aquells habitatges amb el mateix id (id_llar).

Query:

```
MATCH (h:Habitatge),(h2:Habitatge)
WHERE h.id_llar <> h2.id_llar
    AND h.numero=h2.numero
    AND h.municipi=h2.municipi
    AND h.carrer=h2.carrer
MERGE (h)-[:MATEIX_HAB]→(h2)
```

Una vegada creada la nova relació "MATEIX_HAB", tenim les dades per crear i carregar un graf a memòria. Aquest ha de tractar els nodes Individu i Habitatge, i les relacions VIU, FAMILIA i MATEIX_HAB i el farem servir per executar l'algorisme *nodeSimilarity.*

Query:

```
CALL gds.graph.create('Ex3_b',['Individual','Habitatge'],['VIU','FAMILIA','MATEIX_HAB'])
```

Un cop tenim el graf en memòria podem calcular la similitud entre els nodes del graf. A més, escrivim el resultat com una propietat dels nodes en la nostra base de dades. A aquest algoritme li apliquem el privilegi write, el qual ens executa qualsevol comanda d'escriptura en un gràfic. En aquest cas, escrivim el resultat de l'algorisme en la propietat score dins la nostra base de dades no relacional.

Query:

```
CALL gds.nodeSimilarity.write('Ex3_b',{
  writeRelationshipType:'MATEIX_HAB',
  writeProperty:'score',
  similarityCutoff:0.45,
  topK:5
  writeProperty:'score'
  })
YIELD nodesCompared, relationshipsWritten
```

Observació: Executem l'algorisme amb dos paràmetres addicionals *similarity cutoff* i *topK* ja que hem obtingut millors resultats en quant a relacions escrites.

Resultat:

nodesCompared	relationshipsWritten
15243	58455

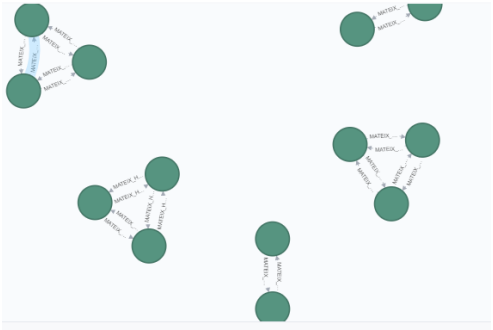
Veiem que en total s'han escrit 58.455 relacions entre 15.243 nodes de tipus Habitatge. En mitjana, cada casa es considera similar a 3.8 altres cases.

Podem escriure una query per veure algunes d'aquestes relacions MATEIX_HAB que hi ha en el graf

Query:

```
MATCH p=(:Habitatge)-[:MATEIX_HAB]-(:Habitatge)
RETURN p LIMIT 10
```

Resultat:



Hem escrit una query que ens permet emmagatzemar el resultat de l'algorisme en una propietat, ara executem una per poder analitzar la sortida.

Query:

```
CALL gds.nodeSimilarity.stats('Ex3_b')
YIELD nodesCompared, similarityDistribution
```

Resultat:

nodesCompared	similarityDistribution
15243	{ "p1": 0.14285707473754883, "max": 1.000007152557373, "p5": 0.20000028610229492, "p90": 1.000007152557373, "p50": 1.000007152557373, "p95": 1.000007152557373, "p10": 0.2500014305114746, "p75": 1.000007152557373, "p99": 1.000007152557373, "p25": 0.5714297294616699, "p100": 1.000007152557373, "min": 0.08333301544189453, "mean": 0.8172385461029535, "stdDev": 0.29223086353859007

La puntuació de similitud de Jaccard oscil·la entre 0 i 1, on 1 significa que la parella de nodes de tipus Habitatge tenen les mateixes característiques que hem definit abans.

A jutjar per la puntuació de p90, el 10% de les parelles d'Habitatges comparteixen. També podem examinar la puntuació p10, el 90% dels habitatges comparteixen més del 16% dels atributs que els defineix com a semblants.

Cal destacar que podem jugar amb el paràmetre ***similarityCutoff,*** per aconseguir una xarxa de similitud més escassa o més abundant.