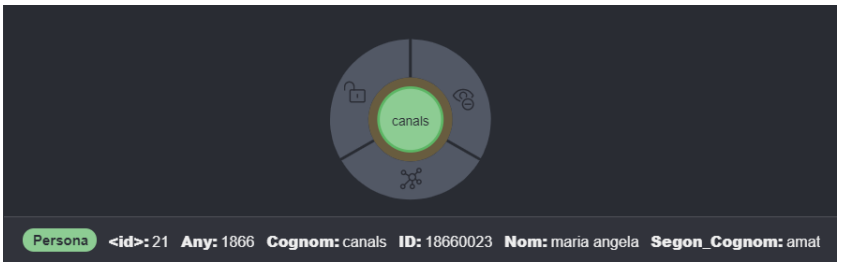
Roberto Manresa Martínez - NIU: 1566754 Javier Ortega Bautista - NIU: 1568416
Daniel Paulí Schmolling - NIU: 1568073 Jaime Rueda Diví - NIU: 1566320
Github: https://github.com/alosdatosmeremito/proyecto2

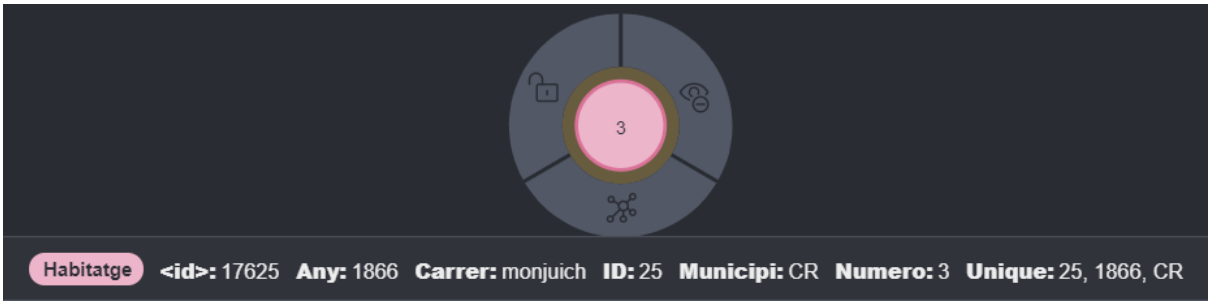
Proyecto Neo4j

Creación de la base de datos

Para la creación de la base de datos hemos seguido el diagrama que se encontraba en el enunciado del proyecto. Hemos creado dos tipos de nodos, uno llamado *Persona* y otro tipo de nodo llamado *Habitatge*.
El nodo de tipo persona almacena 5 datos, el nombre de la persona en el campo *Nom* (string), el apellido en el campo *Cognom* (string), el segundo apellido en el campo *Segon_Cognom* (string), el año del padrón en el campo *Any* (int) y su identificador en el campo *ID* (int).



Los nodos de tipo *Habitatge* tienen 6 campos, *Any* (int) que guarda el año del padrón, *Carrer* (string) que contiene la calle, *Municipi* (string) para el municipio, *Numero* (int) que guarda el número de la casa, *ID* (int) para el identificador y por último un campo extra llamado *Unique* (string) que junta el *ID*, el *Any* y el *Municipi* esto se usa para comprobar que no se introduzcan nodos repetidos como explicaremos más adelante.



Tenemos 3 tipos de ejes, *VIU* que relaciona nodos de tipo *Persona* con nodos de tipo *Habitatge*, *SAME_AS* que relaciona nodos de tipo *Persona* con otros nodos de tipo *Persona* que representan a la misma persona y un eje de tipo *FAMILIA* que relaciona nodos de tipo *Persona* que tienen alguna relación, este último tiene dos campos, uno llamado *Relacio* y otro llamado *Relacio_Harmonitzada* que nos da información del tipo de relación.

Para asegurarnos que no se duplican nodos hemos creado dos constraints. Para los nodos de tipo *Persona* hemos añadido la siguiente línea al principio del script:

```
CREATE CONSTRAINT PrimaryKey_Persona IF NOT EXISTS on (p:Persona)
ASSERT p.ID IS UNIQUE;
```

Esta línea asegura que no pueda haber más de un nodo de tipo *Persona* con el mismo *ID*, para que no cree problemas si se ejecuta más de una vez solo la crea si no existe.
Para el caso de los nodos de tipo Habitatge, hemos creado otro constraint que también hemos añadido al principio del script y que es la siguiente:

```
CREATE CONSTRAINT PrimaryKey_Habitatge IF NOT EXISTS on
(h:Habitatge) ASSERT h.Unique IS UNIQUE;
```

Aquí hacemos uso del campo *Unique* que hemos mencionado anteriormente, lo usamos para asegurarnos que no existen duplicados de este tipo de nodo ya que solo puede haber una casa con el mismo *ID, Any y Municipi.*
También hemos añadido un Index para los 3 campos que componen la clave primaria: *ID*, *Any* y *Municipi*; para acelerar la carga de datos y porque pensamos que a lo mejor podríamos necesitar buscar por esos campos en algún momento y así mejorar el rendimiento mediante la siguiente línea:

```
CREATE INDEX IndexHabitatge IF NOT EXISTS FOR (h:Habitatge) ON
(h.ID, h.Any, h.Municipi);
```

Para evitar duplicados en los ejes, simplemente hemos optado por crear estos mediante el uso de *Merge*, evitando que se creen ejes si ya existían con anterioridad.

Consultas

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

```
match (p:Persona)-[:VIU]->(h:Habitatge) where p.Nom <> 'nan' and
h.Any=1866 and h.Municipi = 'CR'
return count(*) as Num_Habitants, collect(distinct p.Nom) as
Llistat
```

"Num_Habitants"	"Llistat"
333	["rosa","elisa","jose","antonio","emilia","miguel","manuel","juan","magdalena","teresa","pedro","francisco","francisca","salvador","luis","madrona","jacoba","jaime","isidro","paula","josefa","dolores","benito","martin","joaquina","mercedes","pascuala","carmen","angela","lorenzo","ramon","isabel","antonia","maria","vicente","pablo","consepcion","eulalia","maria angela","antinio","estevan","tomas","filomena","amelino","vicenta","catalina","esteban","balbina","joaquin","ilegible","concepcion","rosendo","camila","mariangela","ines","fran","margarita","bartolome","merced","magin","lucia","rosalia","cristobal","clemente","celestino","jacinto","agustin","jacinta","maria rosa","florencia","serafin","sebastia","ignes"]

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.

```
match (h:Habitatge)
where h.Municipi='SFLL' and h.Any < 1840
return h.Any as Any, collect(h.ID) as Llista_Llars
ORDER BY h.Any
```

"Any"	"Llista_Llars"
1833	[95,99,101,103,105,107,109,111,94,96,97,98,108,100,104,102,106,110,113,115,118,119,120,12 ,34,35,36,38,39,40,44,46,47,50,51,52,56,57,58,59,12,14,17,19,21,23,25,27,28,30,31,33,37,4 ,149,151,152,154,155,157,159,161,162,164,165,167,150,153,158,160,163,166,168,169,170,171, 2,203,206,211,212,213,214,215,220,222,216,217,218,221,230,231,219,223,224,225,226,227,228 67,268,269,270,271,272,273,274,275,276,278,284,277,279,280,281,282,283,285,286,287,288,28
1838	[321,324,326,328,320,322,323,325,327,329,330,331,332,333,334,335,336,337,338,339,340,341,
1839	[721,722,723,724,725,731,733,734,726,735,727,729,730,728,732,629,631,633,634,637,638,640, 7,498,500,503,504,506,507,460,554,461,553,462,555,463,556,464,557,465,558,466,559,467,560 20,521,522,550,551,552,568,570,602,603,604,610,608,616,618,620,624,605,606,607,609,611,61 670,672,667,669,572,573,574,575,576,577,578,579,580,581,582,583,585,586,587,588,589,369,3 ,407,408,409,410,411,412,413,414,415,416,417,418,419,420,421,422,423,424,425,426,427,428, 8,529,530,531,532,533,534,535,536,537,538,539,540,541,542,543,544,545,546,547,548,549,584 96,697,698,699,700,701,702,703,704,705,706,707,708,709,710,711,712,713,714,715,716,717,71

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.

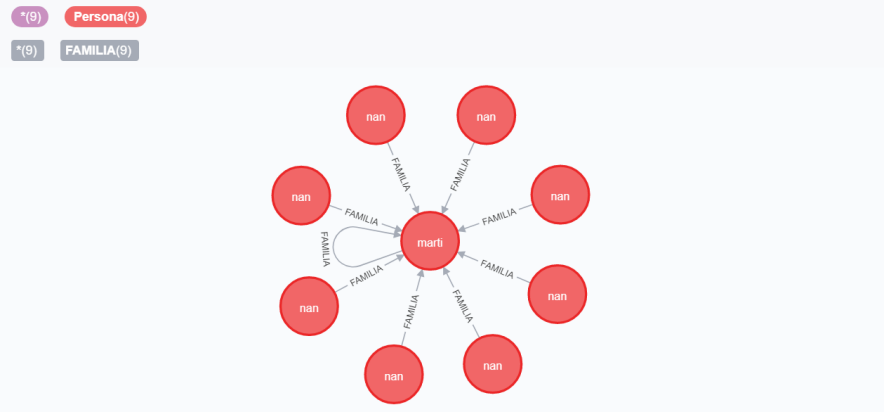
//llista

```
match (p:Persona)-[:VIU]-(h:Habitatge)-[v:VIU]-(p2:Persona)
where toLower(p.Nom)='rafel' and toLower(p.Cognom)='marti' and
h.Any=1838
return p.Nom as Nom, collect(p2.Nom) as Convivents
```

"Nom"	"Convivents"
"rafel"	["felipe","salvadora","franco","maria","jpha","miquel","jpha","jph"]

//graf

```
match (p:Persona)-[:VIU]-(h:Habitatge)-[v:VIU]-(p2:Persona)
where toLower(p.Nom)='rafel' and toLower(p.Cognom)='marti' and
h.Any=1838
return p, collect(p2)
```



4. Retorna totes les aparicions de "Miguel ballester". 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 path=(p:Persona)<-[r:SAME_AS]->(n) where
toLower(p.Nom)='miguel' and toLower(p.Cognom)='ballester'
return path
```

"path"
[{"Cognom":"ballester","Segon_Cognom":"andreu","ID":18812684,"Nom":"miguel","Any":1881},{} ,{"Cognom":"balleste","Segon_Cognom":"y andreu","ID":18570153,"Nom":"miquel","Any":1857}]
[{"Cognom":"ballester","Segon_Cognom":"andreu","ID":18812684,"Nom":"miguel","Any":1881},{} ,{"Cognom":"balleste","Segon_Cognom":"andreu","ID":18780202,"Nom":"miguel","Any":1878}]
[{"Cognom":"ballester","Segon_Cognom":"andreu","ID":18812684,"Nom":"miguel","Any":1881},{} ,{"Cognom":"ballester","Segon_Cognom":"andreu","ID":18892687,"Nom":"miguel","Any":1889}]
[{"Cognom":"ballester","Segon_Cognom":"andreu","ID":18892687,"Nom":"miguel","Any":1889},{} ,{"Cognom":"balleste","Segon_Cognom":"andreu","ID":18780202,"Nom":"miguel","Any":1878}]
[{"Cognom":"ballester","Segon_Cognom":"andreu","ID":18892687,"Nom":"miguel","Any":1889},{} ,{"Cognom":"balleste","Segon_Cognom":"y andreu","ID":18570153,"Nom":"miquel","Any":1857}]
[{"Cognom":"ballester","Segon_Cognom":"andreu","ID":18892687,"Nom":"miguel","Any":1889},{} ,{"Cognom":"ballester","Segon_Cognom":"andreu","ID":18812684,"Nom":"miguel","Any":1881}]

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

```
match(p:Persona)-[:SAME_AS]-(n)
where p.Nom = 'antonio' and p.Cognom = 'farran'
return n.Nom as Nom,n.Cognom as Cognom1,n.Segon_Cognom as
Cognom2,"SAME_AS" as Tipus
UNION ALL
match(p:Persona)-[:FAMILIA]-(n)
where p.Nom = 'antonio' and p.Cognom = 'farran'
return n.Nom as Nom,n.Cognom as Cognom1,n.Segon_Cognom as
Cognom2,"FAMILIA" as Tipus
```

"Nom"	"Cognom1"	"Cognom2"	"Tipus"
"antonio"	"ferran"	"sole"	"SAME_AS"
"antonio"	"ferran"	"sele"	"SAME_AS"
"esperanza"	"farran"	"colet"	"FAMILIA"
"esperanza"	"colet"	"gavarro"	"FAMILIA"
"catalina"	"farran"	"colet"	"FAMILIA"
"francisco"	"farran"	"colet"	"FAMILIA"
"isidro"	"farran"	"colet"	"FAMILIA"
"antonio"	"farran"	"sole"	"FAMILIA"

6. Llisteu totes les relacions familiars que hi ha. Projecte Neo4j. Bases de Dades No Relacionals 2 Alicia Fornés, Oriol Ramos Terrades, 2021

```
match (:Persona)-[x:FAMILIA]-(:Persona)
where x.Relacio_Harmonitzada <> 'null'
return distinct x.Relacio_Harmonitzada as Relacions_Familiars
```

"Relacions_Familiars"
"jefe"
"esposa"
"sogre"
"altres"
"germa"
"mare"
"fill"
"filla"
"familiar"
"net"
"servents"

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.

```
match (h:Habitatge{Municipi:'SFLL'})
where h.Carrer <> 'nan' and h.Numero <> 'nan'
return (h.Carrer) as Carrer, h.Numero as Numero_Carrer,
count(distinct h.Any) as Total, collect(distinct h.Any) as Anys,
collect(h.ID) as Ids
order by Total desc, h.Carrer, h.Numero limit 10
```

"Carrer"	"Numero_Carrer"	"Total"	"Anys"	"Ids"
"creus"	9	5	[1833,1839,1878,1881,1889]	[150,610,467,367,347]
"creus"	16	5	[1833,1839,1878,1881,1889]	[158,616,473,372,388]
"creus"	18	5	[1833,1839,1878,1881,1889]	[160,618,475,374,389]
"creus"	23	5	[1833,1839,1878,1881,1889]	[166,624,480,380,353]
"iglesia"	1	5	[1833,1839,1878,1881,1889]	[94,721,514,509,483]
"iglesia"	2	5	[1833,1839,1878,1881,1889]	[96,722,515,510,498]
"iglesia"	3	5	[1833,1839,1878,1881,1889]	[97,723,724,516,511,484]
"iglesia"	4	5	[1833,1839,1878,1881,1889]	[98,725,517,512,499]
"iglesia"	8	5	[1833,1839,1878,1881,1889]	[102,728,520,516,500]
"iglesia"	9	5	[1833,1839,1878,1881,1889]	[104,729,730,521,517,487]

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.

```
match (h:Habitatge)<-[:VIU]-(p:Persona)-[r:FAMILIA]->(n),
(h:Habitatge)<-[:VIU]-(pe:Persona)-[re:FAMILIA]->(n)
where h.Municipi='CR' and (toLower(r.Relacio)='cap' or
toLower(r.Relacio)='cabeza') and (toLower(re.Relacio)='hijo' or
toLower(re.Relacio)='hija' or toLower(re.Relacio_Harmonitzada) =
'fill' or toLower(re.Relacio_Harmonitzada) = 'filla')
return distinct p.Nom as Nom, p.Cognom as Cognom1, p.Segon_Cognom
as Cognom2, size(collect(pe)) as total
order by total Desc
limit 20
```

"Nom"	"Cognom1"	"Cognom2"	"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

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.

```
match(h1:Habitatge) where h1.Any = 1881 and h1.Municipi = 'SFLL'
with count(distinct(h1)) as num
match(h:Habitatge)<-[:VIU]-(p:Persona)-[r:FAMILIA]->(n)
where h.Any = 1881 and h.Municipi = 'SFLL' and
(toLower(r.Relacio)='hijo' or toLower(r.Relacio)='hija' or
toLower(r.Relacio_Harmonitzada) = 'fill' or
toLower(r.Relacio_Harmonitzada) = 'filla')
return count(distinct(p)) as Total_Fills, num as Num_Llars,
toFloat(count(distinct(p)))/toFloat(num) as Mitjana
```

"Total_Fills"	"Num_Llars"	"Mitjana"
1289	596	2.162751677852349

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

```
match (h:Habitatge)-[:VIU]-(p:Persona)
where h.Municipi='SFLL'
with h.Any as Any, h.Carrer as Carrer, count(*) as num_hab
Order by Any, num_hab
return Any, collect(Carrer)[0] as Carrer, collect(num_hab)[0] as
Num_Habitants
```

"Any"	"Carrer"	"Num_Habitants"
1833	"carrtera de la part de molins de rey"	5
1838	"carretera de barna"	30
1839	"casas del 3onmany"	3
1878	"carrretera"	2
1881	"Carretera"	5
1889	"s n antonio"	1

Ejercicio 3

Apartado A

Con el objetivo de ver cuanta gente hay que no tiene conexión con ninguna casa,y analizar el resultado hemos decidido usar el siguiente subgrafo de nuestra base de datos:

```
CALL gds.graph.create('Padrons',
['Persona','Habitatge'],['VIU', 'FAMILIA'])
```

Vemos cuántas componentes conexas hay sin conexión con el nodos tipo *Habitatge*:

```
CALL gds.wcc.stream("Padrons", {}) YIELD nodeId, componentId AS
community
WITH gds.util.asNode(nodeId) AS node, community
WITH collect(node) AS allNodes, community
where not any(node in allNodes where labels(node)=['Habitatge'])
RETURN count(*)
```

"count(*)"
2778

Como podemos ver, hay 2778 familias sin conexión con algún hogar en total a través de los años.

Debido a que nos parecía extraño que hubiese tantas familias sin casa, hemos extraído el total de componentes conexas con la siguiente consulta:

```
CALL gds.wcc.stream("Padrons", {}) YIELD nodeId, componentId AS
community
WITH gds.util.asNode(nodeId) AS node, community
WITH collect(node) AS allNodes, community
RETURN count(*)
```

"count(*)"
6460

Podemos ver que las familias sin conexión con ninguna casa corresponden al 2778/6460*100=43%, un porcentaje muy elevado, consideramos que es irrealista. Así que nos preguntamos si hay algún error, como por ejemplo: cargar la base de datos, la consulta, nuestras interpretaciones de los datos obtenidos con la consulta, la integridad de los datos registrados o la forma de leerlos de los libros de padrón. Por esa razón, hemos comprobado nuestro script que carga la base de datos y no hemos encontrado ningún error. Para ver si la consulta tiene errores hemos decidido visualizar los nodos que obtenemos con la siguiente consulta:

```
CALL gds.wcc.stream("Padrons", {}) YIELD nodeId, componentId AS
community
WITH gds.util.asNode(nodeId) AS node, community
WITH collect(node) AS allNodes, community
where not any(node in allNodes where labels(node)=['Habitatge'])
RETURN community, size(allNodes) as num
order by num desc
```

509	7281	2
510	11280	2
511	58	1
512	59	1
513	111	1

Started streaming 2778 records after 11 ms and completed after 11 ms, displaying first 1000 rows.

Nos da que solo 510 de los 2778 son comunidades de más de un nodo. Viendo esto se nos ha ocurrido ver que porcentaje de las personas registradas tienen casa ya que es más representativo de la sociedad que las comunidades. Con la siguiente consulta, podemos ver cuanta gente hay sin casa:

```
CALL gds.wcc.stream("Padrons", {}) YIELD nodeId, componentId AS
community
WITH gds.util.asNode(nodeId) AS node, community
WITH collect(node) AS allNodes, community
with community, size(allNodes) as pers
where not any(node in allNodes where labels(node)=['Habitatge'])
RETURN sum(pers)
```

	sum(pers)
1	4741

Como podemos ver que son 4741 personas sin casa, haciendo la misma consulta si el where obtenemos que el total de personas es de 16547, por lo tanto un 4741/16547*100 = 28.65%, que como podemos ver es un porcentaje menor que el que nos daba con las comunidades. Aunque es menos, nos sigue pareciendo un porcentaje astronómico, por lo tanto, creemos que hay algún problema y que no hay tanta gente sin casa. En vez de eso creemos que hay gente que no tiene la casa registrada por algún fallo cuando se recabaron los datos o que por algún motivo ha habido algún error con la manipulación de los mismos a la hora de reducir la base de datos original a una más pequeña para la realización de este proyecto.

Apartado B

En este apartado analizaremos la similitud entre nodos con el algoritmo recomendado. Hemos creado un eje, llamado *MATEIX_HAB*, entre cada *Habitatge* que tiene el mismo *Municipi*, el mismo *Numero* y el mismo *Carrer*, con la siguiente consulta:

```
match (h1:Habitatge), (h2:Habitatge)
where h1.Numero <> 'nan' and h2.Numero <> 'nan' and h1.Municipi <>
'null' and h2.Municipi <> 'null' and h1.Municipi=h2.Municipi and
h1.Carrer=h2.Carrer and h1.Numero=h2.Numero and h1.Any>h2.Any
Merge (h1)-[:MATEIX_HAB]->(h2)
return h1, h2
```

Con el objetivo de analizar los datos hemos decidido usar el siguiente subgrafo:

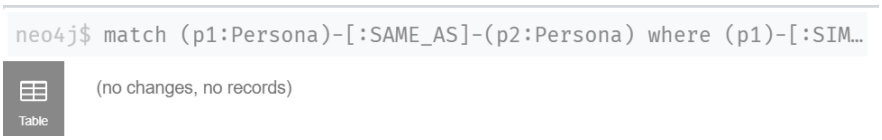
```
CALL gds.graph.create('Similar',
  ['Persona','Habitatge'],['VIU', 'FAMILIA', 'MATEIX_HAB'])
```

Como se expone en el enunciado, vamos a calcular la similitud entre los nodos y crear entre ellos un eje llamado *SIMILAR*, con la siguiente consulta:

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

Para ver si el cálculo de similaridad ha funcionado bien vamos ha hacer una consulta sobre las *Personas*, vamos a ver si existen dos personas que entre ellos exista a la misma ver una relación *SAME_AS* y un *SIMILAR*.

```
match (p1:Persona)-[:SAME_AS]-(p2:Persona)
where (p1)-[:SIMILAR]-(p2)
return p1, p2
```



Como podemos ver en la imagen la consulta no ha devuelto nada, esto significa que la precisión del algoritmo usado es del 0% para este caso.

Ahora vamos a comprobar cómo ha funcionado entre los Habitatges con la siguiente consulta:

```
match (p1:Habitatge)-[:MATEIX_HAB]-(p2:Habitatge)
where (p1)-[:SIMILAR]-(p2)
return count(*) as nCoincidencias
```

"nCoincidencias"
750

Como podemos ver hay 750 parejas de *Habitatges* con los dos tipos de relaciones, *MATEIX_HAB* y *SIMILAR*. Con la misma consulta sin el where podemos ver que hay un total de 3504 de ejes tipo *MATEIX_HAB*, por lo tanto si hacemos 750/3504*100=21.4%. La conclusión que podemos extraer de todo esto es que no funciona correctamente, probablemente debido a que este algoritmo no está preparado para trabajar con el tipo de datos que le hemos introducido.

Trabajo en equipo

La creación del script que crea la base de datos y la realización del ejercicio 3 ha estado a cargo de Javi y Roberto, que han trabajado juntos y a la vez a través de videollamada, para poder así analizar y discutir los resultados.
Las consultas a realizar se han repartido entre los miembros del grupo, se han trabajado individualmente pero nos hemos ayudado entre nosotros si surgía alguna duda o problema.
Este informe ha sido redactado por Javi, Roberto y Dani.