Universitat Autònoma de Barcelona
Facultat de Ciències

Pràctica NEO4J

Autors:
Manuel Arnau & Sofia Di Capua & Gerard Lahuerta & Ona Sánchez
1597487 — 1603685 — 1601350 — 1601181

18 de Juny de 2023

Contents

0 Introducció

En aquest projecte es treballa el disseny, la implementació i la consulta a una base de dades en NEO4J. A partir dels requisits i les dades que s'han subministrat, s'ha implementat un script en Cypher que processa i insereix les dades en una base de dades de NEO4J. Seguidament, s'han implementat les consultes demanades per corroborar l'ús correcte de la base de dades.

L'objectiu d'aquest projecte és afermar els conceptes ensenyats a classe mitjançant una aplicació diversa i realista d'una base de dades.

Podeu veure tots els codis a la següent pàgina de github:

https://github.com/Gerard-Lahuerta/Projecte_NEO4J.git

1 Exercici 1

Per tal de poder treballar amb NEO4J s'ha hagut d'importar les dades que s'hi disposen en els fitxers (*FAMILIA*, *HABITATGES*, *INDIVIDUAL*, *SAME_AS* i *VIU*)[1] en format *CSV* al format Cypher.

Per tal de fer la correcta importació de les dades amb els tractaments de no duplicitat de dades, no importació de dades amb codis identificatius *null* i tipus de variables correctes, s'ha creat un document en Cypher encarregat de fer tots aquests processos: .

Mostrem a continuació el contingut del fitxer i els processos que fa.

1.1 Importació i creació del node Habitatge

```
LOAD CSV WITH HEADERS FROM
 "https://docs.google.com/spreadsheets/d/e/2PACX-
1vT0ZhR6BSO_M72JEmxXKs6GLuOwxm_Oy0UruLJeX8_R04KA
cICuvrwn2OENQhtuvddU5RSJSclHRJf/puboutput=csv" as row
with toInteger(row.Id_Llar) as LlarID, row.Municipi as
Municipi, toInteger(row.Any_Padro) as AnyPadro, row.Carrer as
 Carrer, toInteger(row.Numero) as Numero

WHERE LlarID is not NULL and Municipi<>'null'

CREATE (h:Habitatge {LlarID: LlarID, Municipi: Municipi,
 AnyPadro: AnyPadro})

SET h.Carrer = Carrer,  h.Num = Numero

CREATE CONSTRAINT FOR (h:Habitatge) REQUIRE (h.LlarID,
h.Municipi, h.AnyPadro) IS NODE KEY
```

Es crea una constraint del conjunt d'atributs que representen de manera inequívoca cada habitatge, i es converteix en la clau del node (node key) per evitar així crear duplicats. S'ha utilitzat create en comptes de merges, ja que si s'utilitzava merge no es creaven tots els nodes del CSV.

[1] Aquests fitxer per a un desenvolupament adient i escalable del treball en el nostre grup s'han decidit subministrar-los en el fitxer mitjançant un URL a un directori Drive on són descarregat quan es generen les dades al NEO4J.

1.2 Importació i creació del node Individu

```
LOAD CSV WITH HEADERS FROM
"https://docs.google.com/spreadsheets/d/e/
2PACX-1vTfU6oJBZhmhzzkV_0-avABPzHTdXy8851ySDbn2gq32WwaNmYxfiBtCGJGOZ
sMgCWjzlEGX4Zh1wqe/pub?output=csv" as row
WITH row
WHERE row.Id is not NULL
CREATE (i:Individu {IndividuID: row.Id})
SET i.Year = toInteger(row.Year), i.Name = row.name,
    i.Surname = row.surname, i.SecondSurname = row.second_surname

CREATE CONSTRAINT UniqueIndividuID FOR (i:Individu)
REQUIRE i.IndividuID IS UNIQUE;
```

Es crea una constraint sobre la ID del individu, i es requereix que sigui única, per evitar duplicats. Com amb habitatge, s'utilitza create en comptes de merge perquè amb merge no s'aconseguia crear tots els nodes del CSV.

1.3 Importació i creació de la relació Familia

```
LOAD CSV WITH HEADERS FROM
"https://docs.google.com/spreadsheets/d/e/2PACX-1vRVOoMAMoxHiGboT
jCIHo2yT30CCWgVHgocGnVJxiCTgyurtmqCfAFahHajobVzwXFLwhqajz1fqA8d/
pub?output=csv" AS row
WITH row.ID_1 AS Id_1, row.ID_2 AS Id_2, row.Relacio AS relacio,
    row.Relacio_Harmonitzada AS relacio_harmonitzada
MATCH (p:Individu {IndividuID:Id_1})
MATCH (o:Individu {IndividuID:Id_2})
MERGE (o)-[rel:Relacio_Familiar {relacio: relacio,
    relacio_harmonitzada:relacio_harmonitzada}]->(p)
RETURN count(rel);
```

1.4 Importació i creació de la relació SAME_AS

```
LOAD CSV WITH HEADERS FROM
"https://docs.google.com/spreadsheets/d/e/2PACX-1vTgC8TBmdXhjUOPK
JxyiZSpetPYjaRC34gmxHj6H2AWvXTGbg7MLKVdJnwuh5bIeer7WLUi0OigI6wc/
pub?output=csv" AS row
WITH row.Id_A AS Id_A, row.Id_B AS Id_B
MATCH (p:Individu {IndividuID:Id_A})
MATCH (o:Individu {IndividuID:Id_B})
MERGE (o)-[rel: SAME_AS]-(p)
RETURN count(rel);
```

1.5 Importació i creació de la relació VIU

```
LOAD CSV WITH HEADERS FROM 'https://docs.google.com/spreadsheets/d/e/
2PACX-1vRM4DPeqFmv7w6kLH5msNk6_Hdh1wuExRirgysZKO_Q70L21MKBkDISIyjvd
m8shVixl5Tcw_5zCfdg/pub?output=csv' AS vivencia
WITH vivencia
MATCH (p:Individu {IndividuID:vivencia.IND})
MATCH(h:Habitatge{LlarID:toInteger(vivencia.HOUSE_ID),
                  Municipi:vivencia.Location,
                  AnyPadro:toInteger(vivencia.Year)})
MERGE (p)-[rel:VIU_A]-(h)
return count(rel)
```

2 Exercici 2

En aquest exercici desenvolupem certes queries per fer un primer contacte amb les dades generades, i així, comprovar que tot s'ha carregat correctament.

2.1 Apartat 1

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

```
MATCH (h:Habitatge)<-[v:VIU_A]-(i:Individu)
WHERE h.Municipi='CR'
RETURN count(i.IndividuID) AS N_habitants,
       collect(i.Surname) AS Cognoms
```

2.2 Apartat 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 (h:Habitatge)<-[v:VIU_A]-(i:Individu)
where h.Municipi='SFLL' and i.Surname<>'nan'
    and i.Surname<>'?' is not null
return distinct h.YearPadro as Any_Padro,
    count(i.IndividuID) as N_habitants,
    collect(DISTINCT i.Surname) as Cognoms
```

2.3 Apartat 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 h.AnyPadro < 1845 and h.AnyPadro > 1800
return h.Municipi as `Població`,
       h.AnyPadro as `Any_Padro`,
       collect(h.LlarID) as `Identificador_LLar`
```

2.4 Apartat 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 (h:Habitatge)<-[v:VIU_A]-(i:Individu)
WHERE h.Municipi='SFLL' AND
      i.Surname<>'nan' AND
      i.Surname<>'?' IS NOT NULL
RETURN DISTINCT h.YearPadro AS Any_Padro,
       count(i.IndividuID) AS N_habitants,
       collect(DISTINCT i.Surname) AS Cognoms
```

2.5 Apartat 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 {Name:"miguel",
                   Surname:"estape",
                   SecondSurname:"bofill"})
      -[r:SAME_AS]->(l:Individu)
return i,l
```

2.6 Apartat 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 {Name:"miguel",
                   Surname:"estape",
                   SecondSurname:"bofill"})
      -[r:SAME_AS]->(l:Individu)
return l.Name,
       collect(distinct l.Surname),
       collect(distinct l.SecondSurname)
```

2.7 Apartat 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 (p:Individu {Name:"benito", Surname: "julivert"})
      -[rel]- (r:Individu)
where p.IndividuID <> r.IndividuID
return r.Name+"␣"+r.Surname+"␣"+r.SecondSurname as `Individu`,
       type(rel) as `relació`
```

2.8 Apartat 8

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

```
match (p:Individu {Name:"benito", Surname: "julivert"})
      -[rel]- (r:Individu)
where rel.relacio_harmonitzada = "fill" or
      rel.relacio_harmonitzada = "filla"
return r.Name+"␣"+r.Surname+"␣"+r.SecondSurname as `Individu`,
       type(rel) as `relació`
order by r.Name
```

2.9 Apartat 9

Llisteu totes les relacions familiars que hi ha.

```
match (i:Individu)-[f:Relacio_Familiar]-(i2:Individu)
where f.relacio<>'null'
return distinct f.relacio
```

2.10 Apartat 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 (h:Habitatge)<-[v:VIU_A]-(i:Individu)
where h.Municipi='SFLL' and h.Carrer IS NOT NULL
    and h.Num is not null
return h.Carrer as Carrer, h.Num as Numero,
    count(h.YearPadro) as TotalPadrons,
    collect(distinct h.YearPadro) as AnysPadrons,
    collect(distinct h.LlarID) as Id_Llars
order by TotalPadrons DESC limit 15
```

2.11 Apartat 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_A]-(cap:Individu)<-[rel:Relacio_Familiar]-
      (fill:Individu)
WHERE rel.relacio = "hijo" or
      rel.relacio = "hija" and
      cap.IndividuID <> fill.IndividuID
WITH  collect(distinct fill.IndividuID) as Fills,
      cap.IndividuID as Pare,
      collect(fill.Name) as NomsFills,
      cap.Name as NomPare
WHERE size(Fills) >= 3
RETURN NomsFills, NomPare, size(NomsFills)
ORDER BY size(NomsFills) DESC LIMIT 20
```

2.12 Apartat 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", AnyPadro:1881})
    RETURN count(h.LlarID) as NombreLlars
}

MATCH (h:Habitatge{Municipi:"SFLL", AnyPadro:1881})
      -[:VIU_A]-(cap:Individu)<-[rel:Relacio_Familiar]-
      (fill:Individu)
WHERE rel.relacio_harmonitzada = "fill" or
      rel.relacio_harmonitzada = "filla" and
      cap.IndividuID <> fill.IndividuID
WITH  distinct fill.IndividuID as Fills, NombreLlars
RETURN count(Fills), NombreLlars, count(Fills)/NombreLlars as Mitja
```

2.13 Apartat 13

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 (i:Individu)-[v:VIU_A]->(h:Habitatge)
    where h.Municipi='SFLL'
    return count(i) as habitants,
           h.Carrer as carrer,
           h.AnyPadro as any
}
with habitants, carrer, any
order by habitants ASC
return collect(carrer)[0] as carrer,
       min(habitants) as Num_Habitants,
       any
order by any
```

3 Exercici 3

3.1 Apartat a

L'objectiu d'aquest exercici és fer un estudi de les components connexes (cc) i de l'estructura de les components en funció de la seva mida.

Abans de començar hem de fer una projecció del graf en memòria en la qual executarem algorismes de la GDS. Per fer-ho, usem la següent comanda a Neo4j:

```
CALL gds.graph.project('ex3a',['Individu','Habitatge'],
['VIU_A','SAME_AS','Relacio_Familiar'])
```

A continuació, indiquem els motius, les consultes i l'explicació dels resultats que hem fet per explorar les dades:

- **Taula agrupant els resultats segons la mida de la cc.**

```
CALL gds.wcc.stream('ex3a')
YIELD componentId, nodeId
WITH componentId, size(collect(nodeId))
AS mida, collect(nodeId) AS nodes
ORDER BY mida DESC
RETURN componentId, mida
```

Amb aquesta cerca bàsica volíem saber les mides de les components creades per l'algorisme WCC i, a més a més, tenir un codi base per a les altres cerques.

Els resultats obtinguts són:

	componentId	mida
1	2374	7613
2	3758	138
3	3994	134
4	4032	134
5	4061	130
6	3170	108

Aquests resultats permeten observar com hi ha un component que comprèn una gran quantitat dels nodes del graf, i la resta de components són d'una mida molt inferior. Ens dona peu a pensar que, si no hi faltés informació, molt probablement tots els nodes del graf o gaire bé tots formarien una gran component connexa.

• **Per cada municipi i any el nombre de parelles del tipus: Individu—Habitatge.**

Reutilitzem la projecció del graf feta al punt anterior per tal de fer la cerca del nombre de parelles (Individu)—(Habitatge) que existeixen. Així obtindrem un coneixement més ampli de les relacions que hi ha en el nostre graf.

La comanda en *Cypher* utilitzada per aquesta tasca és la següent:

```
CALL gds.wcc.stream('ex3a')
YIELD componentId, nodeId
WITH componentId, size(collect(nodeId)) AS mida,
collect(nodeId) AS nodes
ORDER BY mida DESC
MATCH(m:Individu) -[rel:VIU_A]-> (n:Habitatge)
WHERE id(n) IN nodes
return n.Municipi as municipi, n.AnyPadro as Any,
count(rel) as num_relacions
```

Aquesta comanda retorna el nombre de parelles comentades abans per municipi i per any; de forma que l'output que genera és:

	municipi	Any	num_relacions
1	"SFLL"	1881	3000
2	"SFLL"	1878	2745
3	"SFLL"	1889	3117
4	"SFLL"	1839	1946
5	"SFLL"	1833	1433
6	"SFLL"	1838	287
7	"CR"	1866	337

Els resultats obtinguts a la taula permeten extreure conclusions sobretot del municipi de Sant Feliu de Llobregat (ja que de Castellví de Rosanes només es té el padró d'un any).

La primera informació que es pot extreure és que la població a les darreres dècades ha crescut respecte al padró de la dècada dels 30, perquè hi ha un major nombre de relacions.

A aquest fet molt probablement també contribueix que els padrons dels darrers anys estiguin en millors condicions.

Un fet a destacar és el nombre de relacions de l'any 1838 comparat amb el del 1839 i 1833. Això porta a pensar que el padró d'aquest any hauria de estar en molt males condicions o es devia perdre part d'ell, però amb l'ajuda dels padrons dels anys pròxims segurament es podria reconstruir part de la informació perduda.

- **Distribució de tipus de nodes (Individu o Habitatge) segons la mida de la cc.**

Amb aquesta query volem provar la teoria de què hi ha més persones que habitatges en la base de dades. Ja que en una mateixa casa hi viurà mínim una persona, però no hi ha màxim.
De forma similar, reutilitzarem la projecció del graf *ex3a* i utilitzem la següent comanda per a cercar la informació:

```
CALL gds.wcc.stream('ex3a')
YIELD componentId, nodeId
WITH componentId, size(collect(nodeId)) AS mida,
collect(nodeId) AS nodes
ORDER BY mida DESC
MATCH(n:Individu)
WHERE id(n) IN nodes
RETURN componentId, mida,
       round(count(n)* 1.0/mida,2) as num_ind,
```

D'aquesta forma, obtenim l'output que es mostra a continuació:

	componentId	mida	num_ind	num_hab
1	2374	7613	0.86	0.14
2	3758	138	0.85	0.15
3	3994	134	0.87	0.13
4	4032	134	0.88	0.12
5	4061	130	0.83	0.17
6	3170	108	0.83	0.17
7	4292	99	0.85	0.15
8	3964	97	0.85	0.15
9	4326	91	0.87	0.13

Es pot veure clarament com les components connexes compleixen la teoria que hi hauria d'haver un major percentatge d'individus que no pas d'habitatges.

Els percentatges d'individus i d'habitants entre les diferents components són molt semblants independentment de la mida, i permet veure que per a cada habitatge s'espera que visquin al voltant de 5, 6 individus.

La query s'ha implementat de manera que el resultat es retorni en percentatge, i per tal de fer això calia transformar el nombre d'individus a float, d'aquí la multiplicació per 1.0.

• **Quantes components connexes no estan connectades a cap node de tipus 'Habitatge'.**

De forma similar, reutilitzarem la projecció del graf *ex3a* i utilitcem la següent comanda per a cerca la informació:

```
CALL gds.wcc.stream('ex3a')
YIELD componentId, nodeId
WITH componentId as totalCC,
     componentId,
     size(collect(nodeId)) AS mida,
     collect(nodeId) AS nodes
MATCH(h:Habitatge)
WHERE id(h) in nodes
WITH componentId, nodes, totalCC
return count(totalCC)-count(distinct componentId)
       as CCNotConnected
```

Amb aquesta query s'obté que hi ha un total de **1512** components connexes que no tenen cap node de tipus habitatge que formi part de la component.
Dit d'una altra manera, hi ha un total de 1512 unitats familiars (que poden estar formades per un sol individu o més) de les quals no s'ha pogut recuperar informació sobre el seu habitatge en els padrons.
Un següent pas que es podria fer seria comptar quants individus representen aquestes components.

Per tal d'aconseguir el resultat s'ha optat per calcular el nombre total de components connexes i després el nombre de components connexes que si tenen una connexió amb almenys un node de tipus Habitatge, de manera que la resta proporciona els que no tenen cap connexió.

3.2 Apartat b

L'objectiu d'aquest exercici és fer un estudi de les similituds entre els nodes. Ens interessa saber quins nodes són semblants per a identificar els individus que són el mateix.
Per a fer-ho, seguim els següents passos:

1. 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 l'aresta apunti a l'habitatge amb any de padró més petit.

```
MATCH (h1:Habitatge), (h2:Habitatge)
WHERE h1 <> h2 AND h1.LlarID = h2.LlarID
    AND h1.Municipi = h2.Municipi
    AND h1.AnyPadro < h2.AnyPadro
MERGE (h1)<-[:MATEIX_HAB]-(h2)
```

Es busca unir aquelles llars que tenen la mateixa ID i són del mateix municipi, ja que representen el mateix habitatge al llarg dels anys. S'afegeix la condició que l'any de l'habitatge h1 sigui menor al de l'habitatge h2 per tal d'unir amb l'any de padró més petit.

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

```
CALL gds.graph.project('ex3b',['Individu','Habitatge'],
    ['VIU_A', 'MATEIX_HAB', 'Relacio_Familiar'])
```

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

```
CALL gds.nodeSimilarity.stats('ex3b')
YIELD nodesCompared, similarityDistribution
```

```
CALL gds.nodeSimilarity.write('ex3b',{
writeRelationshipType:'SIMILAR',
writeProperty:'score',
similarityCutoff:1.0,
degreeCutoff: 2})
YIELD nodesCompared, relationshipsWritten
```

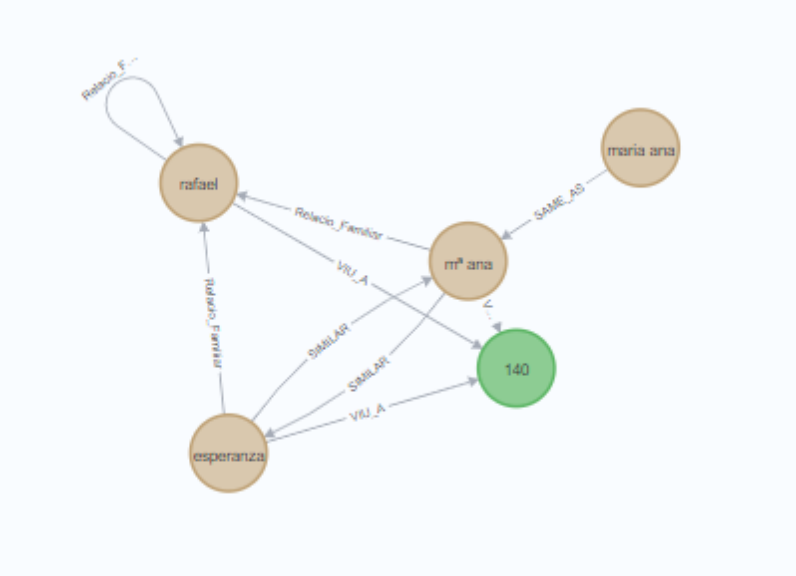
```
match (i:Individu)-[r:SIMILAR]->(p:Individu)
return r.score, i.IndividuID, p.IndividuID
```

Cal comentar la utilització dels diversos fragments: el primer per a cercar els *stats*, el segon per a crear la relació de similitud entre nodes; i el tercer i quart per a mirar aquesta similitud.

Anàlisi dels resultats

Es pot observar que la relació creada a partir de la similitud entre nodes utilitzant l'algorisme nodeSimilarity no ens retorna els individus que són el mateix en tots els casos, sinó que aquesta relació creada ens indica individus que han viscut al mateix habitatge. S'ha afegit el paràmetre degreeCutoff per comprovar que hagin viscut al mateix habitatge al llarg dels anys, de manera que si un individu ha viscut en dos habitatges diferents en dos anys diferents respecte a un altre individu no siguin tildats de similars. S'ha optat pel valor 2 perquè per valors més grans no es creava cap relació.
Cal destacar que només s'uneixen els nodes que tenen una relació de similitud d'1 segons l'algorisme, és a dir que tinguin en comú els mateixos nodes.

Visualment la imatge següent ens mostra un cas de la relació Similar, on es considera que Maria Ana i Esperanza són la mateixa persona (que només pel nom es pot saber que, en principi, no és el cas), però ambdues persones viuen al mateix habitatge i tenen les mateixes relacions familiars.



4 Treball en equip

Per a assegurar una eficiència òptima i l'aprofitament de les habilitats individuals en el nostre projecte de Neo4j, l'equip va decidir dividir estratègicament les diverses tasques. Aquesta metodologia de treball va permetre a cada membre de l'equip centrar-se en àrees específiques, la qual cosa va resultar en una major qualitat i atenció als detalls.

Les responsabilitats es van repartir de la següent manera:

	Exercici 1	Exercici 2
Ona Sánchez	Node Habitatge	2.4, 2.11, 2.12
Manuel Arnau	Relació Same_As i Família	2.1, 2.5, 2.6
Sofia Di Capua	Relació Viu_A	2.2, 2.9, 2.10, 2.13
Gerard Lahuerta	Node Individu	2.3, 2.7, 2.8

No sols cada integrant del grup es va encarregar de la seva part designada, sinó que també va contribuir activament a la redacció de les conclusions i a la descripció de les seves tasques en aquest informe.
A més, tots van participar en la redacció de l'anàlisi dels exercicis d'anàlisis de grafs (exercici 3), la qual cosa va permetre una comprensió més profunda del projecte en el seu conjunt.

Cal destacar que el procés de desenvolupament d'aquest projecte ha estat documentat de manera exhaustiva. L'equip va utilitzar *GitHub* com a plataforma principal per a la col·laboració i el seguiment del progrés.
En aquest enllaç es pot accedir al nostre repositori del projecte, que proporciona una visió detallada del desenvolupament cronològic del nostre treball.