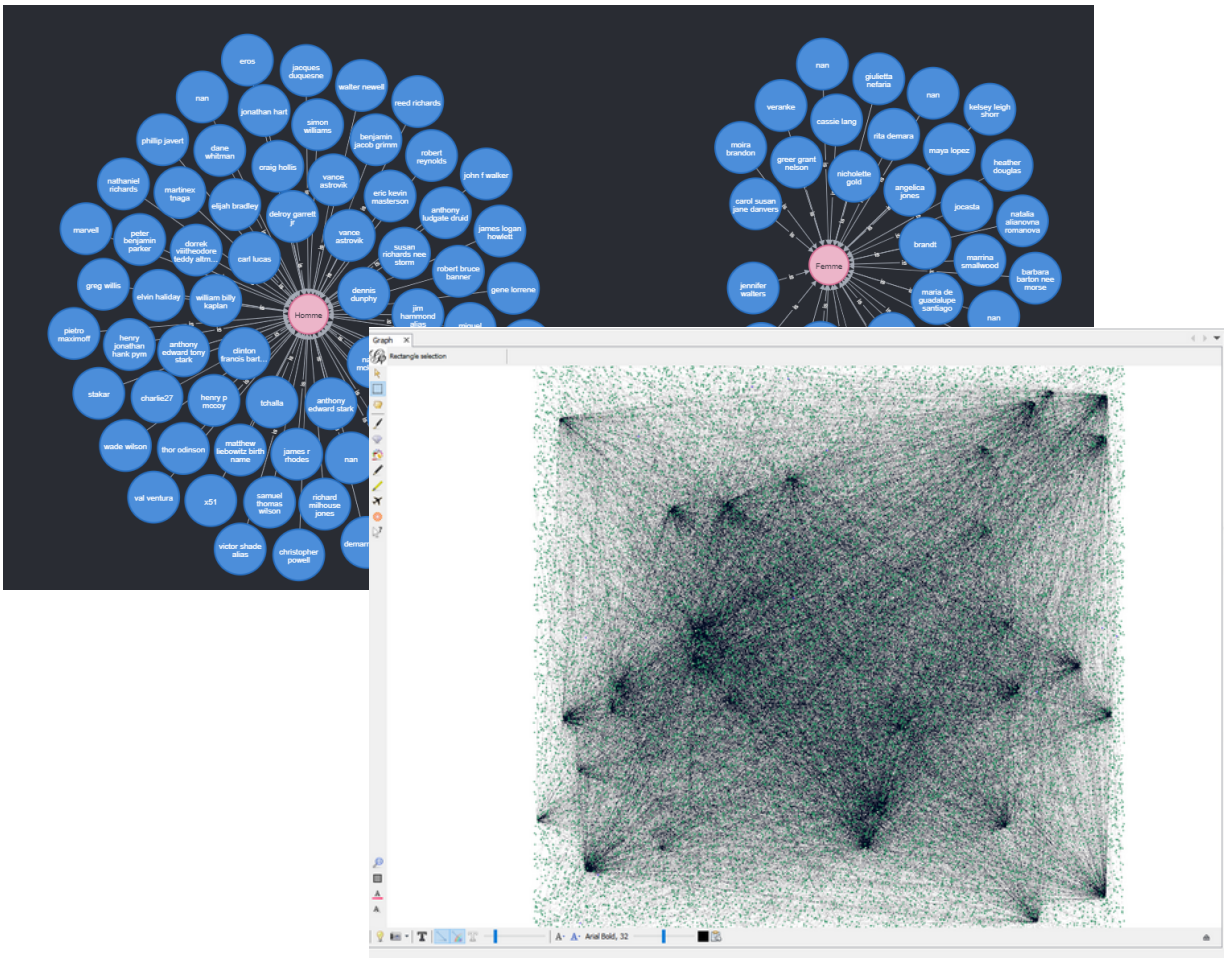
Polytech Annecy-Chambéry – Université Savoie Mont-Blanc

IDU 4 – INFO834 | PERROLLAZ Maverick & CAULLIREAU Dorian | TP2 (11/02/2022)



Partie 1 : Découverte de Neo4j

---

1. Récupération de l'ensemble des Node d'un graphe

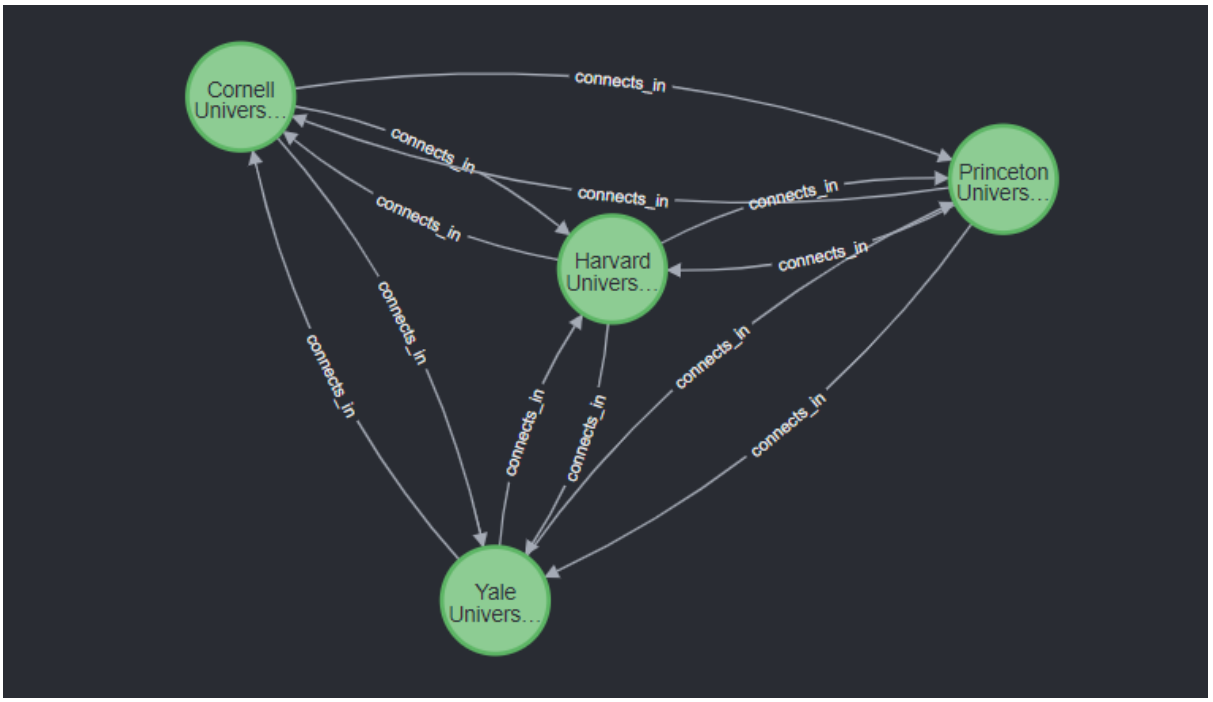
L'objectif est de récupérer l'ensemble des nœuds du graphe. La query permet de récupérer l'ensemble des "university" présent dans le graphe.

Lorsque nous exécutons la query, nous pouvons alors boucler sur l'ensemble des informations.

```
# Question 1 : Récupération de l'ensemble des noeuds du graphe

print("Question 1 : Récupérer l'ensemble des Node d'un graphe")

# Nous souhaitons récupérer l'ensemble des université
# donc sans l'ensemble des éléments sans contrainte

query = "MATCH (x:university) RETURN x"
nodes = graphDB_Session.run(query)
for node in nodes:
    print(node)
print("\n")
```

Nous avons ainsi l'ensemble des enregistrements de *"university" :*

Sortie Neo4J :



Sortie Python :

```
Question 1 : Récupérer l'ensemble des Node d'un graphe
<Record x=<Node id=4 labels=frozenset({'university'}) properties={'name': 'Cornell University'}>>
<Record x=<Node id=5 labels=frozenset({'university'}) properties={'name': 'Yale University'}>>
<Record x=<Node id=6 labels=frozenset({'university'}) properties={'name': 'Princeton University'}>>
<Record x=<Node id=7 labels=frozenset({'university'}) properties={'name': 'Harvard University'}>>
```

2. Récupération d'un Node en particulier

L'objectif est d'appliquer une contrainte sur une requête, pour récupérer une université en particulier.

```
# Question 2 : Récupération d'un node en particulier

print("Question 2: Récupérer un Node particulier")

# Nous souhaitons récupérer une ligne particulière d'une table (contrainte WHERE)
# Dans cette exemple nous réalisons la requête avec l'université "Yale University"

query = "MATCH (x:university {name:'Yale University'}) RETURN x"
nodes = graphDB_Session.run(query)
for node in nodes:
    print(node)
print("\n")4
```

Nous pouvons donc récupérer le/les université qui ont pour nom **'Yale University'.**

Sortie Neo4J :

Sortie Python :

```
2: Récupérer un Node particulier
<Record x=<Node id=9 labels=frozenset({'university'}) properties={'name': 'Yale University'}>>
```

3. Récupération l'ensemble des relationship d'un graphe

Nous allons essayer maintenant de récupérer l'ensemble des chemins d'un graphe :

```
# Question 3 : Récupération de l'ensemble des liens du graphe

print("Question 3: Récupérer l'ensemble des Relationship (ou Path) d'un graphe")
query = "MATCH (x:university)-[r]->(y:university) RETURN x.name,y.name,r.miles"
nodes = graphDB_Session.run(query)
for node in nodes:
    print(node)
print("\n")
```

Sortie Neo4J :

"x.name"	"y.name"	"r.miles"
"Cornell University"	"Harvard University"	327
"Cornell University"	"Princeton University"	210
"Cornell University"	"Yale University"	259
"Yale University"	"Harvard University"	133
"Yale University"	"Princeton University"	133
"Yale University"	"Cornell University"	259
"Princeton University"	"Yale University"	133
"Princeton University"	"Cornell University"	210

Sortie Python :

```
3: Récupérer l'ensemble des Relationship (ou Path) d'un graphe
<Record x.name='Cornell University' y.name='Harvard University' r.miles=327>
<Record x.name='Cornell University' y.name='Princeton University' r.miles=210>
<Record x.name='Cornell University' y.name='Yale University' r.miles=259>
<Record x.name='Yale University' y.name='Harvard University' r.miles=133>
<Record x.name='Yale University' y.name='Princeton University' r.miles=133>
<Record x.name='Yale University' y.name='Cornell University' r.miles=259>
```

4. Récupération d'un lien spécifique

Si notre ancienne requête avait pour but de récupérer l'ensemble des liens, nous allons maintenant récupérer un lien spécifique (avec une contrainte WHERE).

Voici une réalisation avec l'exemple du lien entre Yale University et Cornell University

```python
# Question 4 : Récupération d'un lien spécifique

print("Question 4: Récupérer un Relationship (ou Path) spécifique")

# Pour l'exemple, nous souhaitons récupérer le lien entre 'Yale University' et 'Cornell University'

query = "MATCH (x:university {name:'Yale University'})-[r]->(y:university {name:'Cornell University'}) RETURN x.name,y.name,r.miles"
nodes = graphDB_Session.run(query)
for node in nodes:
    print(node)
print("\n")
```

Nous avons donc bien le lien entre les deux universités :

Sortie Neo4J :

"x.name"	"y.name"	"r.miles"
"Yale University"	"Cornell University"	259

Sortie Python :

```
4: Récupérer un Relationship (ou Path) spécifique
<Record x.name='Yale University' y.name='Cornell University' r.miles=259>
```

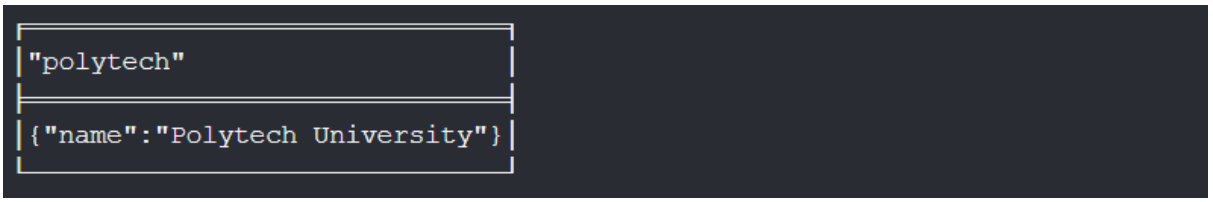
5. Création d'un Node

Pour la création du nouveau nœud, nous considérons que le graphe existe déjà.

Nous ajoutons, et récupérons donc un nouveau noeud (une nouvelle université : “Polytech University”

```
# Question 5 : Création d'un nouveau noeud

print("Question 5: Créer un Node (nous considérons que le graphe existe déjà sur Neo4j Desktop)")
query = "CREATE (polytech:university { name: 'Polytech University'}) RETURN polytech"
nodes = graphDB_Session.run(query)
for node in nodes:
    print(node)
print("\n")
```

Sortie Neo4J :



Sortie Python :

```
5: Créer un Node (nous considérons que le graphe existe déjà sur Neo4j Desktop)
<Record x=<Node id=26 labels=frozenset({'university'}) properties={'name': 'Polytech University'}>>
```

6. Création d'une relation

L'objectif est d'ajouter une nouvelle relation entre deux nœuds.

Dans notre exemple, nous allons ajouter un lien entre "Polytech University" précédemment créé, et un nouveau nœud "Savoie University" que nous allons créer maintenant.

```
print("Question 6: Créer un Relationship (en créant les Node correspondant ou pas)")
query = """
MATCH (x:university {name:'Polytech University'})
CREATE (y:university { name: 'Savoie University'})
CREATE (x)-[r:connects_in {miles: 42}]->(y)
RETURN x.name,y.name,r.miles"""
nodes = graphDB_Session.run(query)
for node in nodes:
    print(node)
print("\n")
```

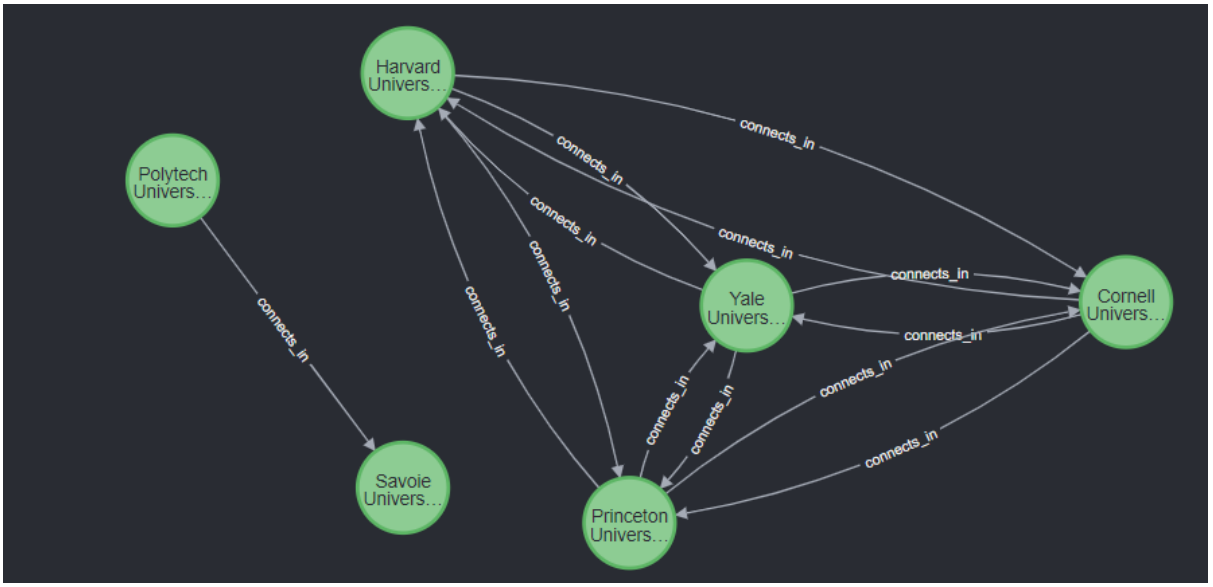
Sortie Neo4J :

"x.name"	"y.name"	"r.miles"
"Polytech University"	"Savoie University"	42

Sortie Python :

```
Question 6: Créer un Relationship (en créant les Node correspondant ou pas)
<Record x.name='Polytech University' y.name='Savoie University' r.miles=42>
```

Nous pouvons donc voir après ajout des deux nouveaux noeuds et du lien entre les deux :



Partie 2 : Même exercice avec un dataset différent

Nous allons maintenant faire une analyse suivant la même idée.

Nous avons choisi un dataset avec l'ensemble des super-héros des Avengers. Nous avons un total de 174 lignes :

```
url,name_alias,appearances,current,gender,probationary introl,full_reserve avengers intro,year,years since joining,honorary,death1,return1,death2,return2,death3,return
http://marvel.wikia.com/henry_pym_(earth-616),"henry jonathan ""hank"" pym",1269,yes,male,,sep-63,1963,52,full,yes,no,,,,,,,,merged with ultron in rage of ultron vol.
http://marvel.wikia.com/janet_van_dyne_(earth-616),janet van dyne,1165,yes,female,,sep-63,1963,52,full,yes,yes,,,,,,,,dies in secret invasion v1:i8. actually was sent
http://marvel.wikia.com/anthony_stark_(earth-616),"anthony edward ""tony"" stark",3068,yes,male,,sep-63,1963,52,full,yes,yes,,,,,,,,"death: ""later while under the in
http://marvel.wikia.com/robert_bruce_banner_(earth-616),robert bruce banner,2089,yes,male,,sep-63,1963,52,full,yes,yes,,,,,,,,"dies in ghosts of the future arc. howev
http://marvel.wikia.com/thor_odinson_(earth-616),thor odinson,2402,yes,male,,sep-63,1963,52,full,yes,yes,yes,no,,,,,,dies in fear itself brought back because that's k
http://marvel.wikia.com/richard_jones_(earth-616),richard milhouse jones,612,yes,male,,sep-63,1963,52,honorary,no,,,,,,,,,na
http://marvel.wikia.com/steven_rogers_(earth-616),steven rogers,3458,yes,male,,mar-64,1964,51,full,yes,yes,,,,,,,,dies at the end of civil war. later comes back.
http://marvel.wikia.com/clint_barton_(earth-616),clinton francis barton,1456,yes,male,,may-65,1965,50,full,yes,yes,yes,yes,,,,,,,dies in exploding kree ship in avernge
http://marvel.wikia.com/pietro_maximoff_(earth-616),pietro maximoff,769,yes,male,,may-65,1965,50,full,yes,yes,,,,,,,,dies in house of m vol 1 issue 7. later comes bac
http://marvel.wikia.com/wanda_maximoff_(earth-616),wanda maximoff,1214,yes,female,,may-65,1965,50,full,yes,yes,,,,,,,,dies in uncanny_avengers_vol_1_14. later comes b
http://marvel.wikia.com/jacques_duquesne_(earth-616),jacques duquesne,115,no,male,,sep-65,1965,50,full,yes,yes,,,,,,,,dies in avengers_vol_1_130. brought back by the
http://marvel.wikia.com/hercules_(earth-616),heracles,741,yes,male,,oct-67,1967,48,full,no,,,,,,,,,na
http://marvel.wikia.com/t%27challa_(earth-616),t'challa,780,no,male,,may-68,1968,47,full,no,,,,,,,,,na
http://marvel.wikia.com/vision_(earth-616),victor shade (alias),1036,yes,male,,nov-68,1968,47,full,yes,yes,,,,,,,,dies in avengers_vol_1_500. is eventually rebuilt.
http://marvel.wikia.com/dane_whitman,dane whitman,482,no,male,,dec-69,1969,46,full,no,,,,,,,,,na
http://marvel.wikia.com/natalia_romanova_(earth-616)#,natalia alianovna romanova,1112,yes,female,,may-73,1973,42,full,yes,yes,,,,,,,,killed by the hand. later revived
http://marvel.wikia.com/mantis_(earth-616)#,brandt,160,no,female,,aug-73,1973,42,full,yes,yes,,,,,,,,,"dies in silver_surfer_vol_3_3. actually ""fragments of her essen
```

Pour lire et utiliser les données, nous avions deux possibilités :

* Utiliser la fonction d'importation de Neo4J et insérer l'ensemble des données
* Mettre en place une fonction "home made" qui importe l'ensemble des données par lecture du csv

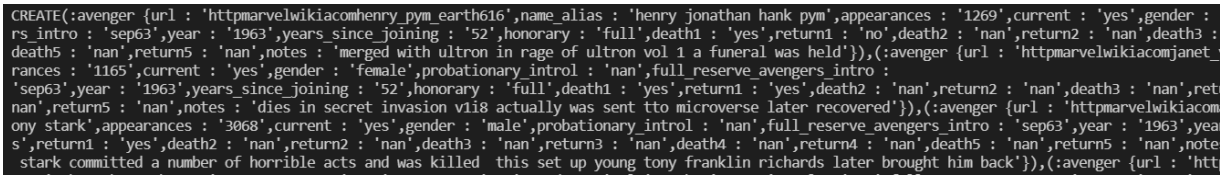
Nous avons dû utiliser une méthode maison, au vu de la complexité de lecture de certains champs (exemple : "name/alias" qui ne peut pas être lu), ou encore le manque de certaines données.

Voici la fonction qui permet de générer la requête :

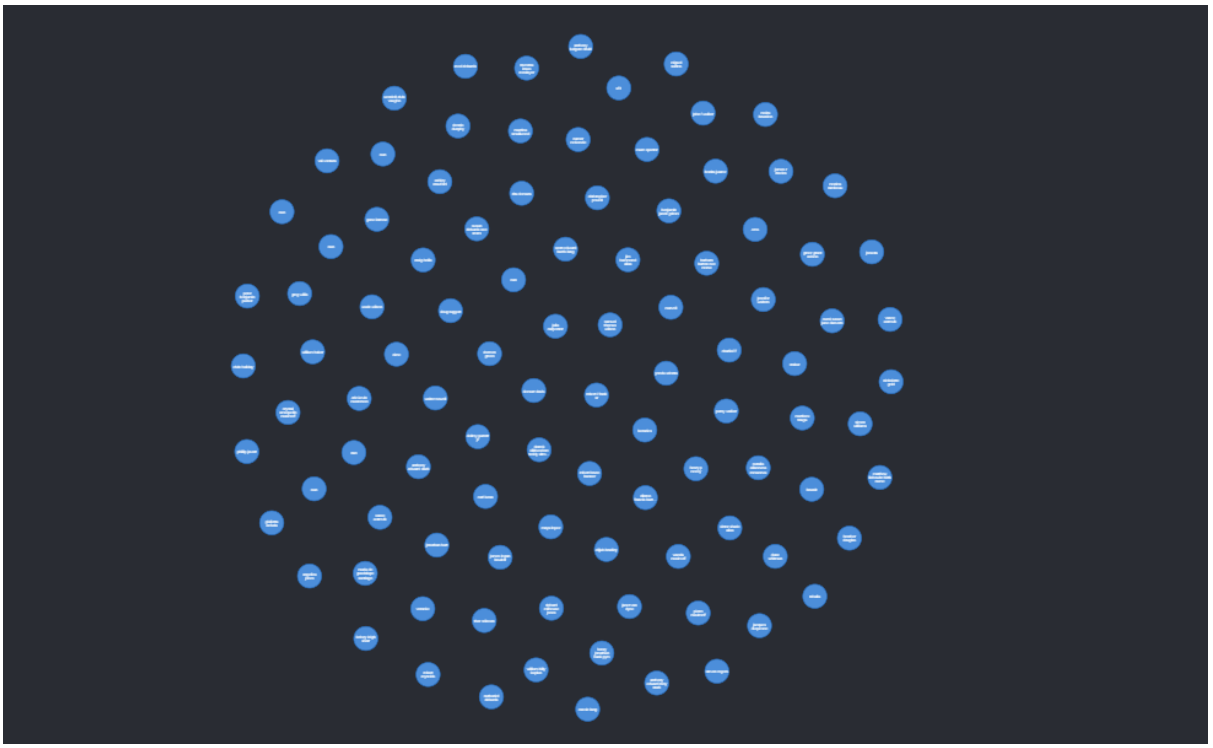
```
cqlCreate = "CREATE"
for index in df.head(100).iterrows():
```

```
    cqlCreate += "(:avenger {"
    for (key, value) in index[1].items():
                cqlCreate  +=  str(key).replace("  ",  "_")  +  "  :  '"  +
re.sub(r'[^\w\s]', '', str(value)) + "',"
    cqlCreate = cqlCreate[:-1]
    cqlCreate += "}),"
cqlCreate = cqlCreate[:-1]
```

Nous avons donc, par exemple pour notre dataset la génération de la requête suivante :

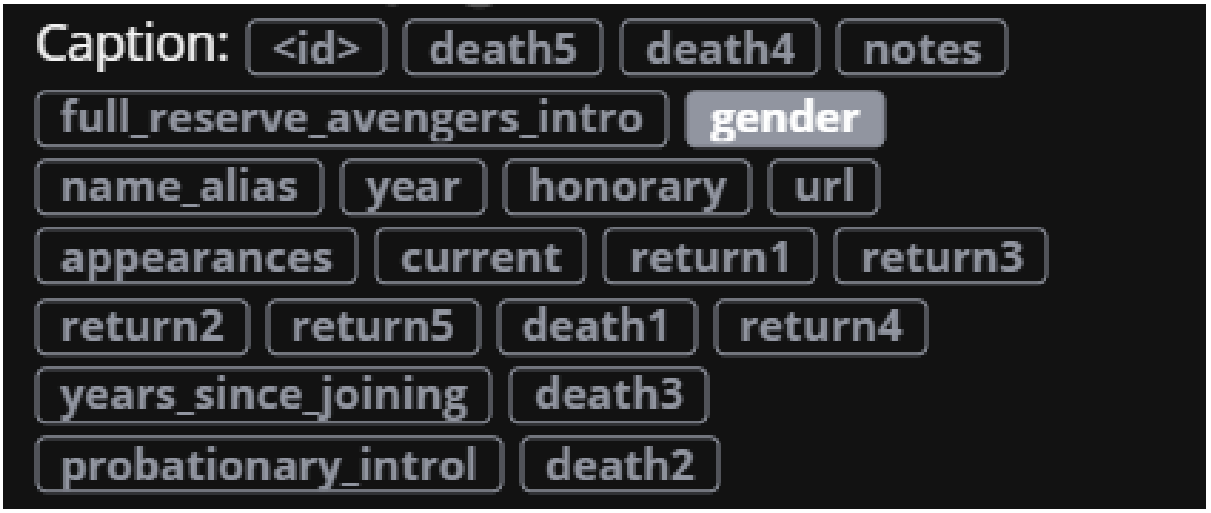


Nous pouvons donc voir que nous avons bien l'ensemble des avengers enregistrées :



Nous allons maintenant essayer de faire des liens entre les personnages.

Il est rapidement compliqué dans certains cas de trouver des liens logiques entre deux personnages.



Nous avons donc décider de faire des liens en fonction du sexe (Homme ou Femme).

Pour celà, nous commençons par définir deux nouveaux points, représentant les hommes ainsi que les femmes.

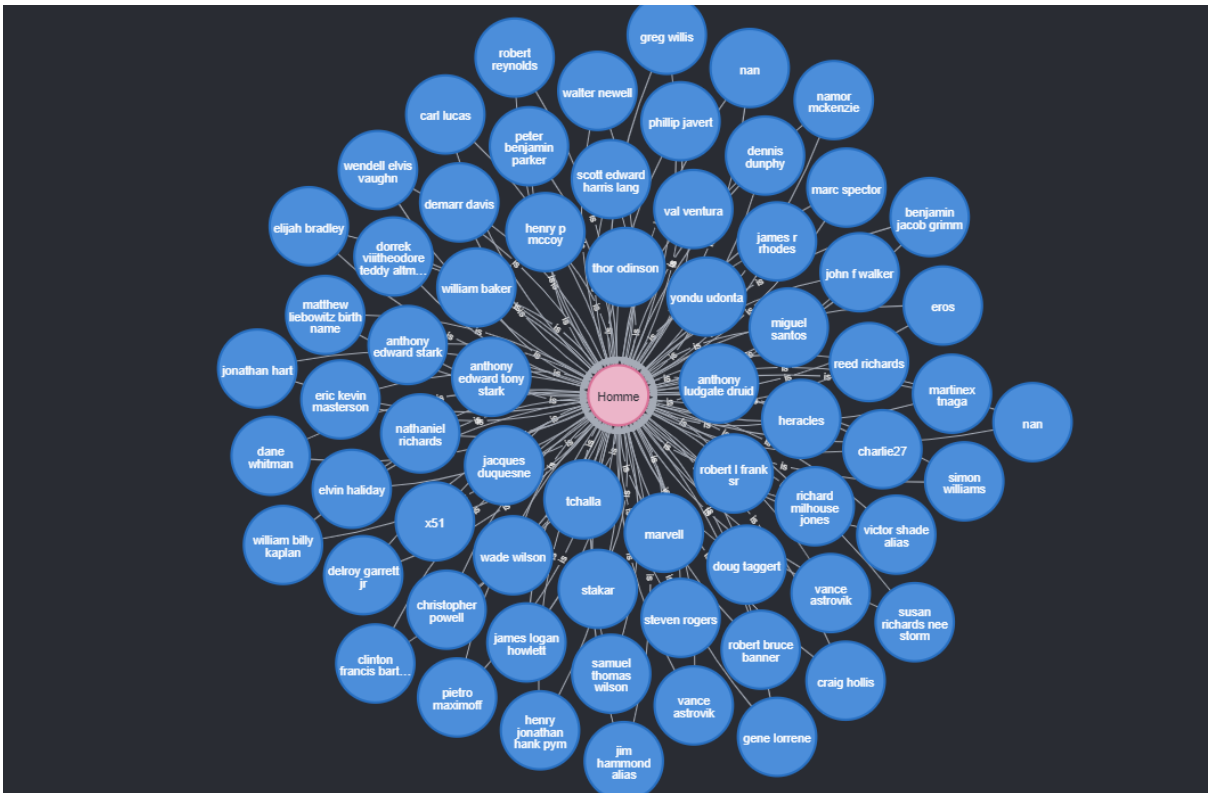
```
cqlCreateGender = """CREATE (male:gender { name: "Homme"}),
(female:gender { name: "Femme"})"""
```

Nous ajoutons ensuite le lien entre les personnes :

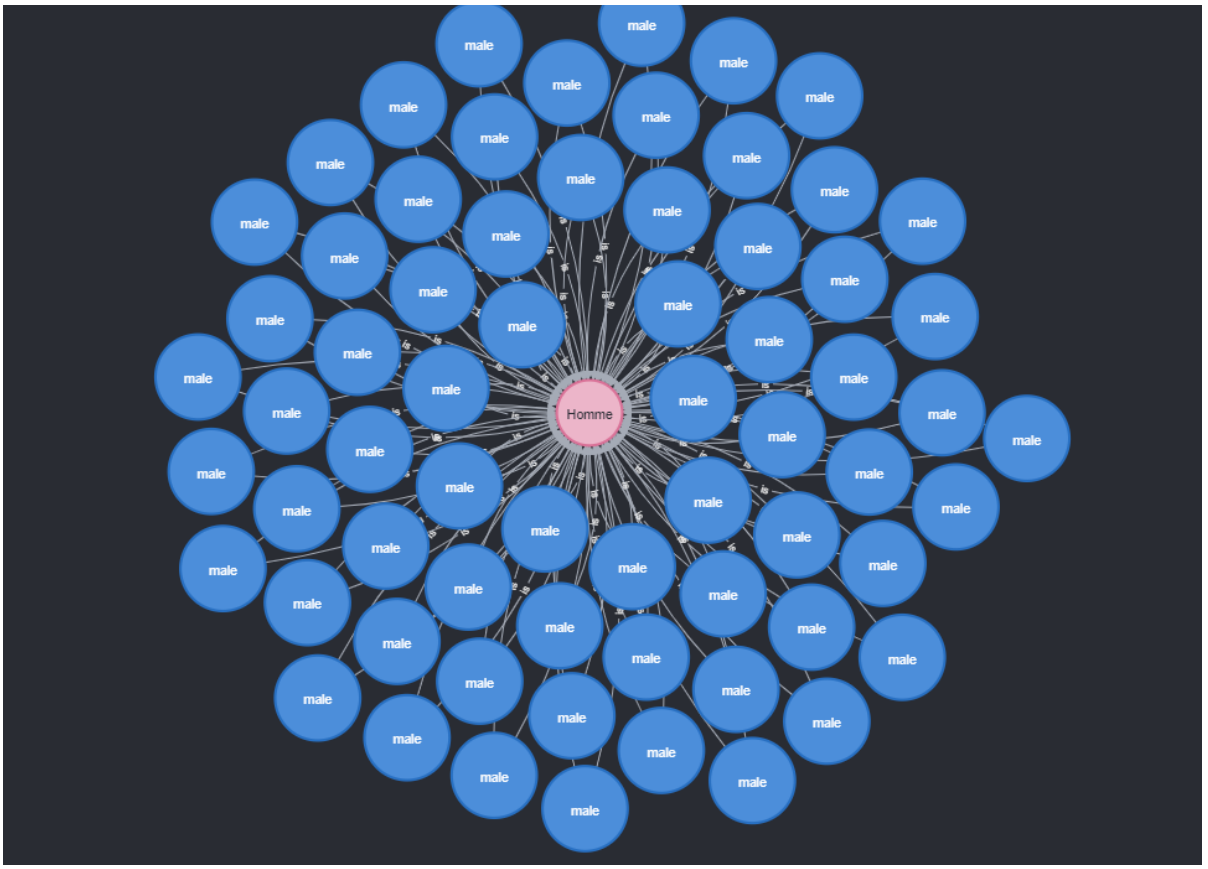
On commence par le lien entre les hommes :

```
cqlLinkGenderMale = """match (h:avenger {gender:"male"}), (gh:gender
{name:"Homme"}) CREATE (h)-[:is]->(gh)"""
```

Nous pouvons donc voir l'ensemble des hommes avec le lien vers le genre :



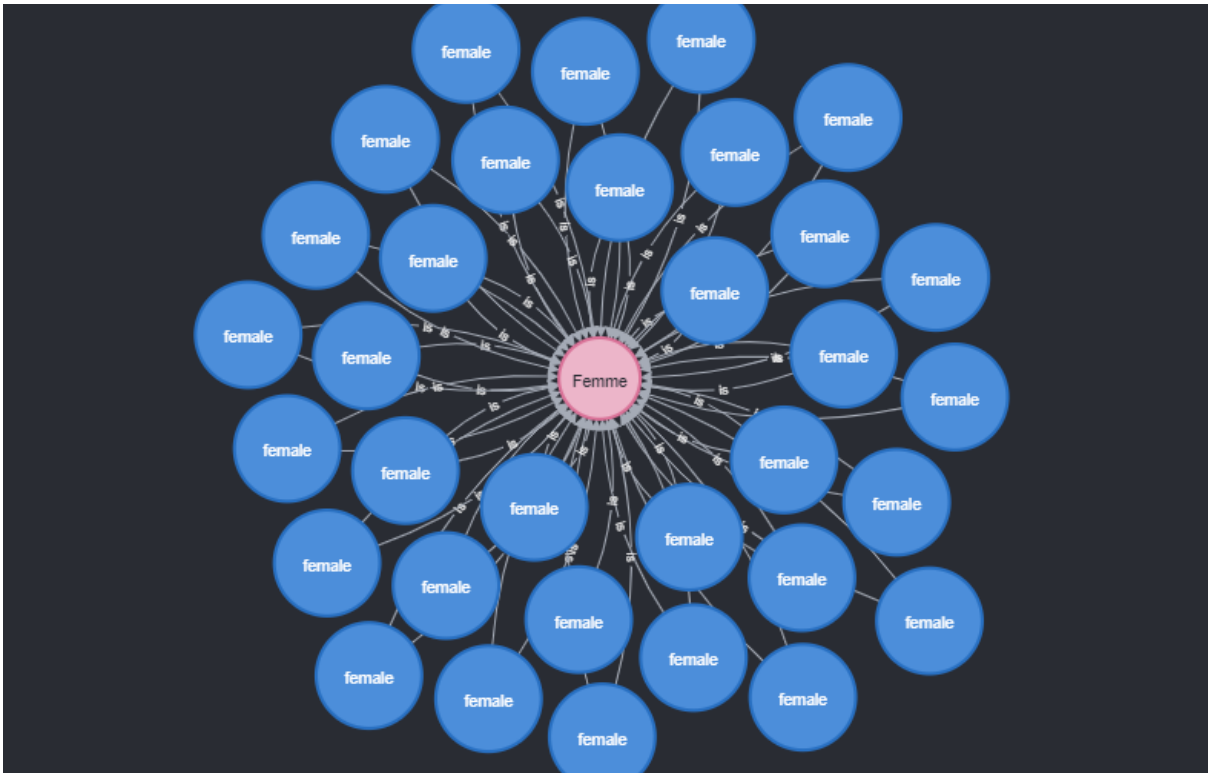
Nous pouvons confirmer que l'ensemble des noeuds reliés au genre homme sont bien des hommes :



Même stratégie mais pour les avenger du genre ‘femmes’ :

```
cqlLinkGenderFemale = """match (f:avenger {gender:"female"}), (gf:gender
{name:"Femme"}) CREATE (f)-[:is]->(gf)"""
```

Nous pouvons également bien vérifier que ce sont toutes des femmes avec le genre de chaque ligne :



Partie 3 : Utilisation de Gephi

Dans cette partie nous allons utiliser Gephi et Neo4j. Pour découvrir l'outil, nous allons utiliser les données des données sur les localités en France (Commune, départements, régions).

Sous format .csv, nous avons les différents champs pour pouvoir faire notre arborescence suivant les différentes localités.

```
code_commune_INSEE,nom_commune_postal,code_postal,libelle_acheminement,l
atitude,longitude,code_commune,article,nom_commune,nom_commune_complet,c
ode_departement,nom_departement,code_region,nom_region
```

La principale difficulté de l'exercice réside surtout dans la lecture et le traitement des données, nous avons un grand nombre de données.

Par exemple, la lecture et l'insertion des données nous à pris un temps important :

```
1000 cities load and inserted in 0.58 seconds
2000 cities load and inserted in 0.95 seconds
3000 cities load and inserted in 1.31 seconds
4000 cities load and inserted in 1.46 seconds
5000 cities load and inserted in 1.95 seconds
6000 cities load and inserted in 2.17 seconds
7000 cities load and inserted in 2.68 seconds
8000 cities load and inserted in 2.93 seconds
9000 cities load and inserted in 3.30 seconds
10000 cities load and inserted in 3.51 seconds
11000 cities load and inserted in 22.27 seconds
12000 cities load and inserted in 41.18 seconds
13000 cities load and inserted in 60.35 seconds
14000 cities load and inserted in 83.19 seconds
15000 cities load and inserted in 103.05 seconds
16000 cities load and inserted in 122.10 seconds
17000 cities load and inserted in 141.38 seconds
18000 cities load and inserted in 160.09 seconds
```

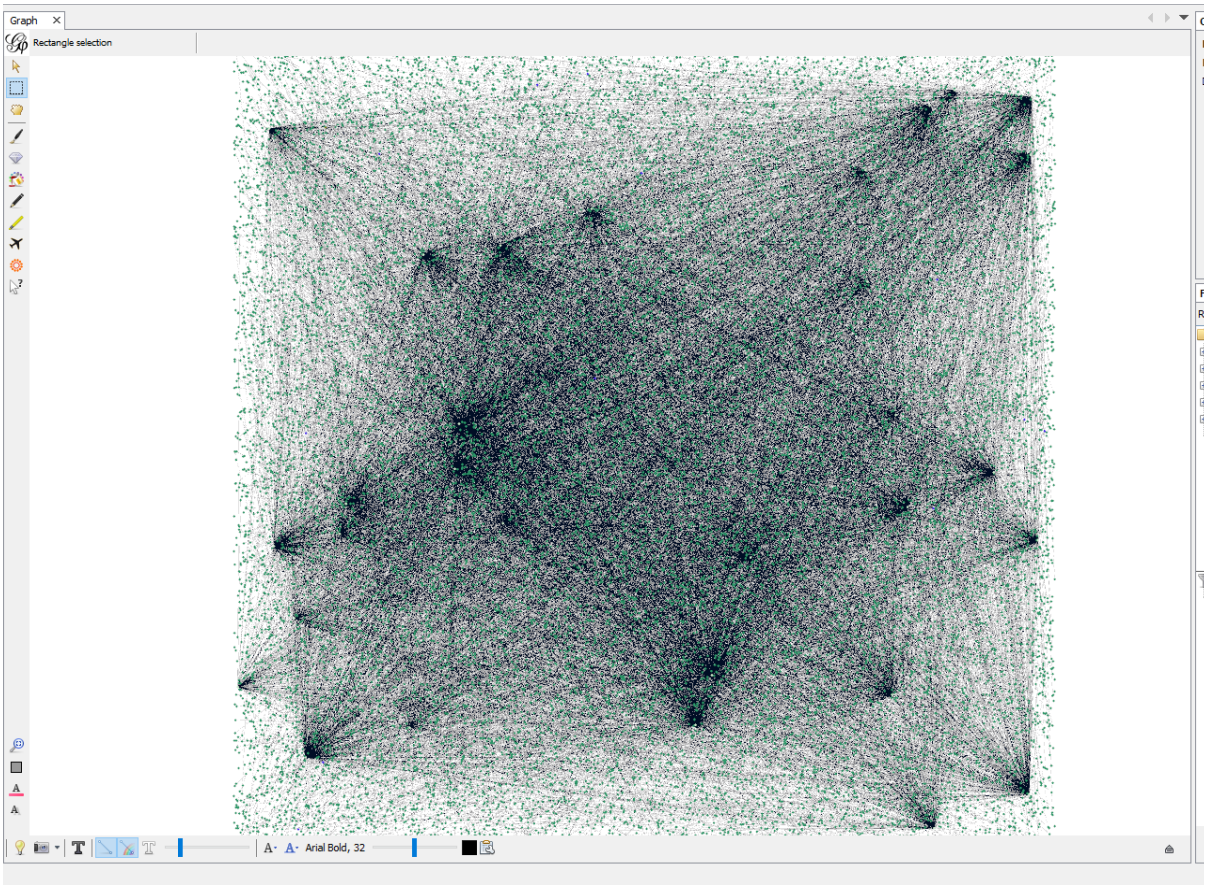
Dans un premier temps nous ajoutons l'ensemble des départements, régions et localités dans Neo4j pour ensuite faire le traitement directement dans Gephi :

```
cqlCreateDepartment = """MATCH (c:city) MERGE (d:departement { name:
apoc.text.clean(c.nom_departement), nom_region:
apoc.text.clean(c.nom_region) }) CREATE (c)-[:in]->(d) """
cqlCreateRegion = """MATCH (d:departement) MERGE (r:region { name:
apoc.text.clean(d.nom_region)}) CREATE (d)-[:in]->(r)"""
cqlCreateFrance= """MATCH (r:region) MERGE (f:france { name: 'France'})
CREATE (r)-[:in]->(f)"""
```

Pour ensuite l'affichage de l'arborescence nous utilisons l'API de Gephi :

```
cqlGephi = """
MATCH path = (:city)-[in]-(:departement)-[:in]-(:region)-[:in]-(:france)
WITH path LIMIT 39000
WITH collect(path) as paths
CALL apoc.gephi.add('http://localhost:8080','workspace1', paths) YIELD
nodes, relationships, time
RETURN nodes, relationships, time"""
```

Voici le rendu sur Gephi :



Analyse du rendu Gephi :

Nous pouvons voir que le rendu est difficilement compréhensible, un très grand nombre d'informations sont présentes et nous pouvons avoir des difficultés à voir les informations principales.
Par exemple, une coloration de certains éléments aurait pu être possible pour permettre d'avoir une meilleure lisibilité, ou encore la séparation de certains éléments pour avoir moins d'informations et plus de visibilité.