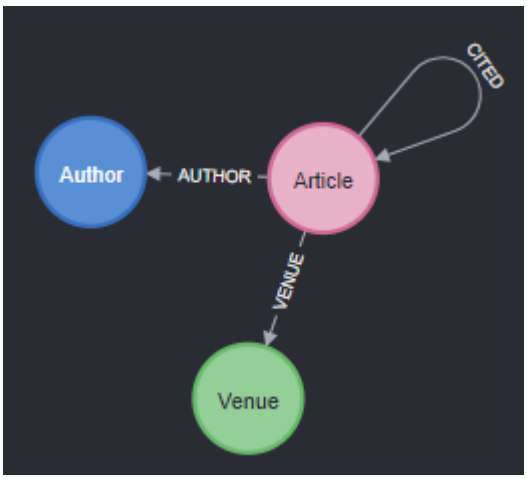
TIW2 - TP Noté 2 : Analyse d'un graphe bibliographique en Neo4j

Binôme

- Jérémy Thomas p1702137
- Julien Giraud p1704709

1. Inférer le schéma

```
CALL db.schema();
```



2. Quel est le label le plus fréquent qui apparaît dans le jeu de données ?

```
MATCH (n)
WITH labels(n) as Label, count(*) as Count
ORDER BY Count DESC
RETURN Label, Count
LIMIT 1;
```

Le label le plus fréquent dans le jeu de données est `Author` avec 80299 apparitions :

"Label"	"Count"
["Author"]	80299

3. Quel est le nombre moyen d'articles publiés par un auteur ?

```
MATCH (author:Author)<-[:AUTHOR]->(article:Article)
WITH author, count(*) as Count
RETURN AVG(Count) as nombre_moyen_article_par_auteur;
```

Le nombre moyen d'articles publiés par un auteur est d'environ `1,75` :

"nombre_moyen_article_par_auteur"
1.7506444663071998

4. Quel est l'auteur qui a publié le plus?

```
MATCH (author:Author)<-[:AUTHOR]->(article:Article)
RETURN author.name as Auteur, count(*) as nombre_articles_publies
ORDER BY nombre_articles_publies DESC
LIMIT 1;
```

L'auteur qui a publié le plus est `Peter G. Neuman` avec 89 articles :

"Auteur"	"nombre_articles_publies"
"Peter G. Neumann"	89

5. Quel est l'année dans laquelle le plus de publications sont apparues ?

```
MATCH (article:Article)
RETURN article.year as Annee, count(*) as nombre_articles_publies
ORDER BY nombre_articles_publies DESC
LIMIT 1;
```

L'année pour laquelle le plus de publications sont apparues est `2006` avec 7536 articles :

"Annee"	"nombre_articles_publies"
2006	7536

6. Les 20 futurs collaborateurs les plus probables de Brian Fitzgerald.

```
MATCH (authorX:Author{name:'Brian Fitzgerald'})<-[:AUTHOR]->(article:Article)<-
[:AUTHOR]->(authorZ:Author)<-[:AUTHOR]->(article2:Article)<-[:AUTHOR]->
(authorY:Author)
WHERE authorX <> authorY
WITH authorX, authorZ, authorY, count(authorY) as nombre_collaborations_z_y
MATCH (authorY)
WHERE NOT (authorX)<-[:AUTHOR]->(:Article)<-[:AUTHOR]->(authorY)
RETURN authorX.name as auteur_x, authorZ.name as co_auteur_z, authorY.name as
collaborateur_probable_y, nombre_collaborations_z_y
ORDER BY nombre_collaborations_z_y DESC LIMIT 20;
```

"auteur_x"	"co_auteur_z"	"collaborateur_probable_y"	"nombre_collaborations_z_y"
"Brian Fitzgerald"	"Matthias Tichy"	"Holger Giese"	5
"Brian Fitzgerald"	"Scott A. Hissam"	"Kurt C. Wallnau"	4
"Brian Fitzgerald"	"Scott A. Hissam"	"Robert C. Seacord"	4
"Brian Fitzgerald"	"Scott A. Hissam"	"Gabriel A. Moreno"	4
"Brian Fitzgerald"	"Walt Scacchi"	"Chris Jensen"	4
"Brian Fitzgerald"	"Scott A. Hissam"	"Grace A. Lewis"	4
"Brian Fitzgerald"	"Scott A. Hissam"	"Judith A. Stafford"	4
"Brian Fitzgerald"	"André van der Hoek"	"Nicolás López"	3
"Brian Fitzgerald"	"André van der Hoek"	"Thomas D. LaToza"	3
"Brian Fitzgerald"	"André van der Hoek"	"Gerald Bortis"	3
"Brian Fitzgerald"	"Andrea Capiluppi"	"Neil Smith"	2
"Brian Fitzgerald"	"Andrea Capiluppi"	"Juan Fernandez-Ramil"	2
"Brian Fitzgerald"	"Paul Ralph"	"Ben Shreeve"	2
"Brian Fitzgerald"	"Walt Scacchi"	"Thuan Q. Pham"	2
"Brian Fitzgerald"	"Matthias Tichy"	"Sven Burmester"	2
"Brian Fitzgerald"	"Matthias Tichy"	"Stefan Henkler"	2
"Brian Fitzgerald"	"Walt Scacchi"	"Salah Bendifallah"	2
"Brian Fitzgerald"	"Walt Scacchi"	"Peiwei Mi"	2
"Brian Fitzgerald"	"Walt Scacchi"	"Alf Inge Wang"	2
"Brian Fitzgerald"	"Walt Scacchi"	"Pankaj K. Garg"	2

7. Écrire une requête pour déterminer combien de ces collaborateurs les plus probables ont publié avec des collaborateurs de Brian plus de 3 fois ?

```
MATCH (authorX:Author{name:'Brian Fitzgerald'})<-[:AUTHOR]->(article:Article)<-
[:AUTHOR]->(authorZ:Author)<-[:AUTHOR]->(article2:Article)<-[:AUTHOR]->
(authorY:Author)
WHERE authorX <> authorY
WITH authorX, authorZ, authorY, count(authorY) as nombre_collaborations_z_y
MATCH (authorY)
WHERE NOT (authorX)<-[:AUTHOR]->(:Article)<-[:AUTHOR]->(authorY) AND
nombre_collaborations_z_y > 3
WITH authorY, nombre_collaborations_z_y
ORDER BY nombre_collaborations_z_y DESC
RETURN collect(authorY.name + ' (' + nombre_collaborations_z_y + '
collaborations)') as collaborateurs_probables_nbZYSup3, count(*) as
nombre_collaborateurs_probables;
```

"collaborateurs_probables_nbZYSup3"	"nombre_collaborateurs_probables"
["Holger Giese (5 collaborations)","Chris Jensen (4 collaborations)","Grace A. Lewis (4 collaborations)","Robert C. Seacord (4 collaborations)","Judith A. Stafford (4 collaborations)","Gabriel A. Moreno (4 collaborations)","Kurt C. Wallnau (4 collaborations)"]	7

8. Écrire une requête pour ajouter, à chaque node Article, le PageRank correspondant à son influence par rapport aux relations CITED.

```
CALL gds.graph.create('cited_rank', 'Article', {CITED:{}});

CALL gds.pageRank.stream('cited_rank')
YIELD nodeId, score
WITH gds.util.asNode(nodeId) as node, score
SET node.page_rank = score
RETURN node;
```

9. Écrire une requête pour calculer les articles les plus influents par rapport à cette mesure PageRank. Renvoyer aussi le score associé à chaque article.

```
MATCH (article:Article)
RETURN article.title as Article, article.page_rank as PageRank
ORDER BY article.page_rank DESC
LIMIT 5;
```

"Article"	"PageRank"
"A method for obtaining digital signatures and public-key cryptosystems"	93.94312708644435
"Secure communications over insecure channels"	79.86924247565909
"Rough sets"	25.609109045405006
"An axiomatic basis for computer programming"	23.029374313916765
"Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems"	21.469559568758996

10. Écrire une requête pour trouver les auteurs qui ont publié le plus d'articles 'open source'.

```
// Créer un index sur les propriétés 'title' et 'abstract' des noeuds Article en
exécutant la requête
CALL db.index.fulltext.createNodeIndex('articles', ['Article'], ['title',
'abstract']);

// Vérification
CALL db.indexes()
YIELD description, indexName, tokenNames, properties, state, type, progress
WHERE type = "node_fulltext"
RETURN *;

// Requête
CALL db.index.fulltext.queryNodes("articles", "open source")
YIELD node
WITH [(author)<-[:AUTHOR]-(node) | author.name] AS authors
UNWIND authors as auteur
RETURN auteur, count(*) as nombre_articles_open_source
ORDER BY nombre_articles_open_source DESC
LIMIT 20;
```

"auteur"	"nombre_articles_open_source"
"Denys Poshyvanyk"	16
"Rocco Oliveto"	13
"Ahmed E. Hassan"	13
"Gail C. Murphy"	12
"Michele Lanza"	11
"Gabriele Bavota"	11
"Massimiliano Di Penta"	11
"Tao Xie"	11
"Tien N. Nguyen"	11
"Reid Holmes"	10
"Martin P. Robillard"	10
"Andrian Marcus"	9
"Audris Mockus"	9
"John Hatcliff"	9
"Margaret-Anne D. Storey"	8
"Zhendong Su"	8
"Andrea De Lucia"	8
"James D. Herbsleb"	7
"Sonia Haiduc"	7
"Collin McMillan"	7

11. Écrire un index de recherche pour la propriété name des nœuds avec le label Authors.

```
CALL db.index.fulltext.createNodeIndex('authors', ['Authors'], ['name']);
```

"description"	"indexName"	"progress"	"properties"	"state"	"tokenNames"	"type"
"INDEX ON NODE:Article(title, abstract)"	"articles"	100.0	["title","abstract"]	"ONLINE"	["Article"]	"node_fulltext"
"INDEX ON NODE:Authors(name)"	"authors"	100.0	["name"]	"ONLINE"	["Authors"]	"node_fulltext"

12. Créer une relation CO_AUTHOR qui permet de relier chaque deux auteurs qui ont collaboré sur au moins un article. Ajouter une seule arête CO_AUTHOR même si la paire d'auteurs a collaboré sur plusieurs articles. Ajouter les propriétés suivantes a ces arêtes CO_AUTHOR : year : l'année de publication du premier article que les auteurs ont écrit ensemble; collaborations : le nombre d'articles sur lesquelles la paire d'auteurs a collaboré.

```
// Requête
MATCH (author1: Author)<-[:AUTHOR]->(article:Article)<-[:AUTHOR]->(author2:Author)
WITH author1, author2, count(article) as nombre_collaboration, min(article.year)
as annee
CREATE (author1)-[:CO_AUTHOR{year:annee, collaborations:nombre_collaboration}]->
(author2);

// Vérification
MATCH (author1:Author)-[:CO_AUTHOR]-(author2:Author)
RETURN *
LIMIT 20;
```

13. Créer des nouvelles relations CO_AUTHOR_EARLY et CO_AUTHOR_LATE entre des nœuds correspondant à des co-auteurs qui ont été publiés ensemble avant, respectivement, après 2006.

```
// CO_AUTHOR_EARLY
MATCH (author1:Author)-[relation:CO_AUTHOR]-(author2:Author)
WHERE relation.year < 2006
CREATE (author1)-[:CO_AUTHOR_EARLY]->(author2);

// CO_AUTHOR_LATE
MATCH (author1:Author)-[relation:CO_AUTHOR]-(author2:Author)
WHERE relation.year > 2006
CREATE (author1)-[:CO_AUTHOR_LATE]->(author2);
```

14. Créer une projection en mémoire du graphe en exécutant la requête. Utiliser l'algorithme de comptage des triangles (cliques de trois nœuds), gds.alpha.triangleCount.write, sur la relation CO_AUTHOR_EARLY et, respectivement, sur la relation CO_AUTHOR_LATE. Interpréter les résultats.

```
// Projection du graphe
CALL gds.graph.create.cypher(
  'whole-graph',
  'MATCH (n) RETURN id(n) AS id',
  'MATCH (n)-[r]->(m) RETURN id(n) AS source, id(m) AS target, type(r) AS type'
);

CALL gds.alpha.triangleCount.write('whole-graph', {
  writeProperty: 'triangles',
  relationshipTypes: ['CO_AUTHOR_EARLY']
})
YIELD triangleCount as triangle_count_early;

CALL gds.alpha.triangleCount.write('whole-graph', {
  writeProperty: 'triangles',
  relationshipTypes: ['CO_AUTHOR_LATE']
})
YIELD triangleCount as triangle_count_late;
```

"graphName"	"nodeProjection"	"relationshipProjection"	"nodeCount"	"relationshipCount"	"createMillis"
"whole-graph"	{"*":{"properties":{},"label":"*"}}	{"CO_AUTHOR_EARLY":{"orientation":"NATURAL","aggregation":"DEFAULT","type":"CO_AUTHOR_EARLY","properties":{}},"CO_AUTHOR_LATE":{"orientation":"NATURAL","aggregation":"DEFAULT","type":"CO_AUTHOR_LATE","properties":{}},"CO_AUTHOR":{"orientation":"NATURAL","aggregation":"DEFAULT","type":"CO_AUTHOR","properties":{}},"VENUE":{"orientation":"NATURAL","aggregation":"DEFAULT","type":"VENUE","properties":{}},"AUTHOR":{"orientation":"NATURAL","aggregation":"DEFAULT","type":"AUTHOR","properties":{}},"CITED":{"orientation":"NATURAL","aggregation":"DEFAULT","type":"CITED","properties":{}}}	132259	1056409	954

"triangle_count_early"
244251

"triangle_count_late"
179563

244 251 triangles pour la relation CO_AUTHOR_EARLY et 179 563 triangles pour la relation CO_AUTHOR_LATE. On en déduit qu'il y a eu plus de collaborations entre les auteurs avant 2006.

15. Utiliser l'algorithme de calcul des composantes connexes, gds.alpha.scc.write, sur la relation CITED pour calculer les différents clusters du graphe.

```
CALL gds.alpha.scc.write({
  nodeProjection:'Article',
  relationshipProjection: 'CITED',
  writeProperty: 'componentId',
  relationshipProjection: {
    CITED: {
      orientation: "UNDIRECTED"
    }
  }
})
YIELD setCount, maxSetSize, minSetSize;
```

"setCount"	"maxSetSize"	"minSetSize"
34027	14169	1

16. Écrire une requête pour calculer quels sont les nœuds qui font partie de chaque composante.

```
MATCH (article:Article)
RETURN article.componentId as Composante, collect(article.title) as Articles
ORDER BY Composante ASC;
```

17. Quelles sont les 3 composantes avec le plus de nœuds ? Écrire une requête permettant de les visualiser.

```
MATCH (article:Article)
RETURN article.componentId AS Composante, count(*) AS Count
ORDER BY Count DESC
LIMIT 3;
```

"Composante"	"Count"
0	14169
1749	61
1928	53

18. Écrire une requête pour calculer, pour chaque composante, sa densité de publication (le nombre total d'articles publiés par les auteurs qui font partie de cette composante / le nombre total d'auteurs dans la composante).

```
MATCH (article:Article)-[:AUTHOR]-(author:Author)
WITH article.componentId as Composante, count(article) as Count, author.name as
Name
RETURN Composante, avg(Count) as densite
ORDER BY densite DESC;
```

"Composante"	"densite"
16989	5.0
13222	4.0
20738	4.0
4243	3.5
9425	3.0
12963	3.0
13002	3.0
3804	3.0
14553	3.0
15157	3.0
16589	3.0

19. Quelles sont les 3 composantes avec la plus forte densité ?

```
MATCH (a:Article)-[:AUTHOR]-(author:Author)
WITH a.componentId as ComponentID, author.name as Name, count(a) as Title
RETURN ComponentID, avg(Title) as densite
ORDER BY densite DESC
LIMIT 3;
```

"ComponentID"	"densite"
16989	5.0
20738	4.0
13222	4.0

20. Ajouter une propriété weight = 1 sur les arêtes CO_AUTHOR et utiliser l'algorithme Louvain de détection de communautés pour identifier les clusters d'auteurs qui publient ensemble. Enregistrer la vitesse d'exécution de l'algorithme de clustering et la comparer avec celle correspondant au calcul des composantes connexes. Interpreter les resultats.

```
MATCH ()-[relation:CO_AUTHOR]->()
SET relation.weight = 1;

CALL gds.louvain.write(
    'whole-graph', {
        writeProperty: 'community'
    }
)
```