TP1 Analyse de graphes

Rajouter vos requêtes à la suite de chaque question est déposer le fichier sur moodle

Nom: KIARED

Prénom: Siham

Nom: VINCENT

Prénom: Yann

Nom: TANG

Prénom: Kévin

### Partie 1:

1. Combien de Movies avez-vous dans la base ?

MATCH (m:Movie)

RETURN count(m);

Résultat : 400

2. Donner L’âge de l’utilisateur dont l’id est 5

MATCH (u:User) WHERE u.id = 5 RETURN u.age;

Résultat : 33

3. Lister les utilisateurs de sexe féminin qui sont soit écrivaines (writer) soit artistes (artist)

MATCH (u:User)

WHERE (u.occupation = 'artist' OR u.occupation = 'writer')

AND u.sex = 'F'

RETURN u;

4. Quel est l’âge moyen des étudiants ?

MATCH (u:User) WHERE u.occupation='student' return avg(u.age);

Résultat : 23.193548387096776

5. Quel est l’âge moyen par occupation ?

MATCH (u:User) RETURN u.occupation, avg(u.age);

6. Combien d’« occupations » ont un age moyen plus grand que 30 ?

MATCH (u:User)

WITH u.occupation AS occupation, AVG(u.age) AS averageAge

WHERE averageAge > 30

RETURN COUNT(DISTINCT occupation);

7. Quelles sont les 3 « occupations » les plus populaires ?

https://neo4j.com/docs/developer-manual/current/cypher/clauses/order-by/

MATCH (u:User)

WITH u.occupation AS occupation, COUNT(u) AS userCount

RETURN occupation, userCount

ORDER BY userCount DESC

LIMIT 3;

8. Combien de valeurs différentes existent-ils pour l’attribut occupation ?

MATCH (u:User)

RETURN COUNT(DISTINCT u.occupation);

9. Quels sont les films produits en 1995 ? (l'année de production est contenue dans le titre du film)

MATCH (m:Movie)

WHERE m.title ENDS WITH '(1995)'

RETURN m.title;

### Partie 2:

10. Combien de notations y-a-t-il (i.e., combien d’occurrences de l’association RATED)

match ()-[r:RATED]->() return count(r);

Résultat : 11310

11. Afficher les notations faites par l’utilisateur dont l’id est 1 et afficher le graphe

avec les liens entre l’utilisateur et les films qu’il a notés.

match (u:User)-[r:RATED]->(m:Movie)

where u.id=1

return u,r,m;

12. Afficher les films qui ont été notés par l’utilisateur dont l’id est 1 ainsi

que leur genres (limiter l'affichage à 10)

MATCH (u:User {id: 1})-[:RATED]->(m:Movie)-[:CATEGORIZED\_AS]->(g:Genre)

RETURN m.title, g.name

LIMIT 10;

13. Combien d’utilisateurs ont noté le film Braveheart (1995)?

MATCH (u:User)-[r:RATED]->(m:Movie)

WHERE m.title = 'Braveheart (1995)'

RETURN COUNT(DISTINCT u);

14. Quels sont les 5 films les plus notés (ont reçu le plus grand nombre de notations)

match (u:User)-[r:RATED]->(m:Movie)

return m, count(r) as nbRatings

order by nbRatings DESC

limit 5;

15. Combien de films ont reçu au moins une fois la note 1

MATCH (u:User)-[r:RATED]->(m:Movie)

WHERE r.score = 1

RETURN COUNT(DISTINCT m);

16. Donner la liste des films qui ont reçu au moins une fois la note 1

en donnant le nombre de fois où ils ont reçu cette note et

qui a donné la note.

MATCH (u:User)-[r:RATED]->(m:Movie)

WHERE r.score = 1

RETURN m.title, COUNT(r), COLLECT(u.id)

ORDER BY COUNT(r) DESC;

17. Quels sont les films qui ont une note moyenne >4

MATCH (u:User)-[r:RATED]->(m:Movie)

WITH m, AVG(r.score) AS avgScore

WHERE avgScore > 4

RETURN m.title, avgScore

18. Quels sont les films non notés par le user 13 ?

(les comparaisons et négations sont similaires au SQL, utiliser NOT)

MATCH (m:Movie)

WHERE NOT (:User {id: 13})-[:RATED]->(m)

RETURN m.title;

## Partie 3:

**19. Lister les 5 films qui ont été évalués avec la note maximale (note = 5) le plus de fois ?**

La requete :

MATCH (m:Movie)<-[r:RATED {score: 5}]-(u:User)

RETURN m.title AS Film, COUNT(r) AS NombreDeNotesMax

ORDER BY NombreDeNotesMax DESC

LIMIT 5;

Les Films retournées :

Star wars (1977) 68 notes max

Fargo (1996) 59 notes max

Raiders of the lost ark (1981) 51 notes max

Pulp Fiction ( 1994 ) 47 notes max

Godfather 46 notes max

**20. Lister les 5 films qui ont été évalués avec la note maximale (note = 5) le plus de fois**

**en affichant leurs notes moyennes?**

La requête :

MATCH (m:Movie)<-[r:RATED]-(u:User)

WITH m, AVG(r.score) AS NoteMoyenne, COUNT(CASE WHEN r.score = 5 THEN 1 END) AS NombreDeCinq

ORDER BY NombreDeCinq DESC, NoteMoyenne DESC

LIMIT 5

RETURN m.title AS Film, NoteMoyenne, NombreDeCinq

Les films :



21. **Lister les amis et les amis des amis du user 1 (Attention user 1 ne doit pas être ami de lui-même) (vous pouvez utiliser un pattern de chemin de longueur 2 : voir cours ou** [**http://neo4j.com/docs/developer-manual/current/cypher/syntax/patterns/**](http://neo4j.com/docs/developer-manual/current/cypher/syntax/patterns/)**)**

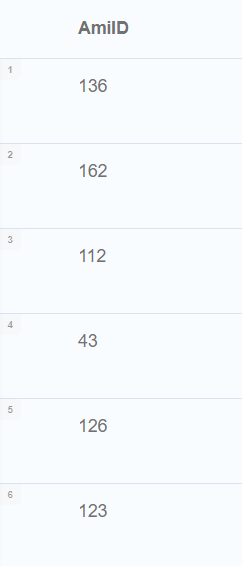
la Requete :

MATCH (user1:User {id: 1})-[:FRIEND\_OF\*1..2]->(friend:User)

WHERE NOT user1 = friend

RETURN DISTINCT friend.id AS AmiID

Resultat : ( exemple )



22. **Lister les amis et les amis des amis du user 1 en donnant leur distance du user 1 (cette distance est 1 lorsque c’est un ami direct et est 2 lorsque c’est un ami d’un ami).**

La requete :

MATCH path = (user1:User {id: 1})-[:FRIEND\_OF\*1..2]->(friend:User)

WHERE NOT user1 = friend AND NONE(rel in relationships(path) WHERE startNode(rel) = endNode(rel))

RETURN friend.id AS AmiID, LENGTH(path) AS Distance

ORDER BY Distance ASC, AmiID ASC

23. **Quel est l’utilisateur qui a le plus d’amis en commun avec l’utilisateur 1 ?**

MATCH (u1:User {id: 1})-[:FRIEND\_OF]->(commonFriend)-[:FRIEND\_OF]->(u2:User)

WHERE u1 <> u2

WITH u2, COUNT(commonFriend) AS nbCommonFriends

RETURN u2.id, nbCommonFriends

ORDER BY nbCommonFriends DESC

LIMIT 1;

24. **Quel est l’utilisateur qui a le plus d’amis en commun avec l’utilisateur 1 et**

**qui n’est pas son ami**

MATCH (u1:User {id: 1})-[:FRIEND\_OF]->(commonFriend)-[:FRIEND\_OF]->(u2:User)

WHERE u1 <> u2 AND NOT (u1)-[:FRIEND\_OF]->(u2)

WITH u2, COUNT(commonFriend) AS nbCommonFriends

RETURN u2.id, nbCommonFriends

ORDER BY nbCommonFriends DESC

LIMIT 1;

25. **Quel utilisateur a noté le plus grand nombre de films que l’utilisateur 1 a également notés et quel est ce nombre ?**

MATCH (u1:User {id: 1})-[:RATED]->(m:Movie)<-[:RATED]-(u2:User)

WHERE u1 <> u2

WITH u2, COUNT(DISTINCT m) AS nbCommonMovies

RETURN u2.id, nbCommonMovies

ORDER BY nbCommonMovies DESC

LIMIT 1;

### Partie 4:

Neo4j permet de tirer parti des connexions qui existent entre les données pour recueillir et recommander des informations pertinentes aux utilisateurs. Par exemple, pour la base de ce TP, nous sommes intéressés par recommander des films à un utilisateur.

Si l’on suit le principe de la recommandation collaborative (collaborative filtering utilisé en particulier par Amazon) et qui consiste à recommander à un utilisateur l’article apprécié par les utilisateurs qui ont un profile similaire, nous recommanderons à notre utilisateur les films, qu’il n’a pas vus (on suppose qu’un film noté est un film vu), qui ont été bien notés par des utilisateurs qui ont apprécié les mêmes films que notre utilisateur (ceux qui ont regardé le plus grand nombre de films en commun avec notre utilisateur et les ont bien notés (score = 4 ou 5)=profile similaire).

- Ecrire cette requête pour l’utilisateur 21.

MATCH (u1:User {id: 21})-[:RATED]->(m:Movie)<-[:RATED]-(similarUser:User),

(similarUser)-[r:RATED]->(recommendedMovie:Movie)

WHERE u1 <> similarUser AND r.score >= 4

AND NOT (u1)-[:RATED]->(recommendedMovie)

WITH recommendedMovie, COUNT(DISTINCT similarUser) AS recommendationCount

ORDER BY recommendationCount DESC

RETURN recommendedMovie.title AS RecommendedMovie, recommendationCount

- Proposer un modèle de recommandation de rencontres entre utilisateurs de sexe différents sur la base à la fois de l’amitié ( ils ont le plus grand nombre d’amis en commun) et d’appréciation des mêmes films. Ecrire une requête de cette recommandation.

MATCH (male:User {sex: 'M'})-[:FRIEND\_OF]-(commonFriend)-[:FRIEND\_OF]-(female:User {sex: 'F'}),

(male)-[r1:RATED]->(commonMovie)<-[r2:RATED]-(female)

WHERE r1.score >= 4 AND r2.score >= 4 AND male <> female

WITH male, female, COUNT(DISTINCT commonFriend) AS CommonFriends, COUNT(DISTINCT commonMovie) AS CommonMovies

ORDER BY CommonFriends DESC, CommonMovies DESC

RETURN male.id AS MaleUserID, female.id AS FemaleUserID, CommonFriends, CommonMovies