Université Mohammed V de Rabat
Ecole Nationale Supérieure d'Arts et Métiers
Cycle Ingénieur Filière INDIA (3ème année) A.U. : 2023/2024 Pr. A. El Qadi

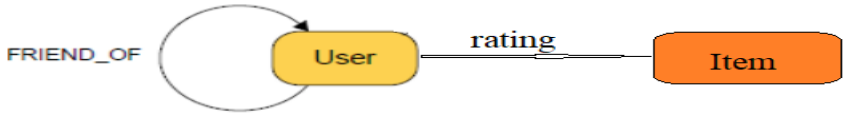
TP n4 en Big Data

Exercice1.

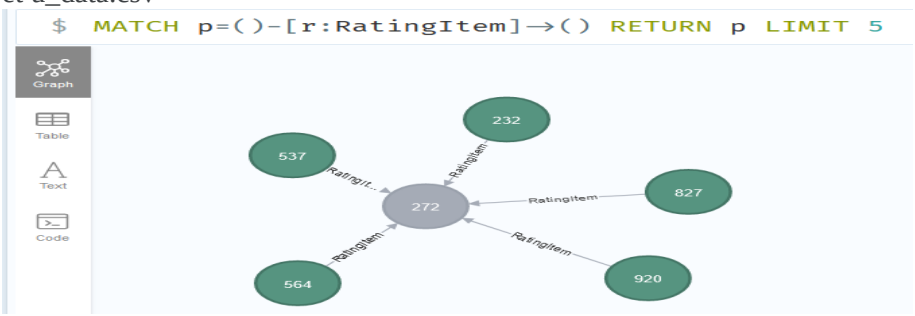
Au cours des dernières décennies, avec l'essor de Youtube, Amazon, Netflix et de nombreux autres services Web de ce type, les systèmes de recommandation ont pris de plus en plus de place dans nos vies. De l'e-commerce (proposer aux acheteurs des articles susceptibles de les intéresser) à la publicité en ligne (proposer aux utilisateurs les bons contenus, correspondant à leurs préférences), les systèmes de recommandation sont aujourd'hui incontournables dans nos parcours quotidiens en ligne.

D'une manière très générale, les systèmes de recommandation sont des algorithmes visant à suggérer des articles pertinents aux utilisateurs (des éléments tels que des films à regarder, du texte à lire, des produits à acheter ou tout autre chose selon les industries).

Dans cet exercice, vous allez importer un petit ensemble de données contenant des nœuds et des relations. L'ensemble de données est divisé en trois fichiers CSV, où chaque fichier comporte une ligne d'en-tête décrivant les données.
Il s'agit de données extraites du Web à partir de critiques de films. L'ensemble de données contient des informations sur les films, les utilisateurs et les notes. Les données des films (movies) et des utilisateurs sont stockées sous forme de nœuds et les notes (ratings) sont stockés sous forme de relations.



Les fichiers à partir desquels vous souhaitez importer des données sont les suivants : u_user.csv, u_item.csv, et u_data.csv



1. Copier les fichiers dans le répertoire « import » du neo4j community
2. Charger les donnees des trosi fichiers

```
load csv with headers from 'file:///u_user.csv' as row
CREATE (u:User{user_id:toInteger(row.user_id),age:toInteger(row.age),sex:row.sex,occupation:row.occupation});

load csv with headers from 'file:///u_item.csv' as row
CREATE (i:Item{movie_id:toInteger(row.movie_id),movie_title:row.movie_title,release_date:row.release_date});

CREATE INDEX ON :User(user_id);
CREATE INDEX ON :Item(movie_id);

load csv with headers from 'file:///u_data.csv' as row
MERGE(u:User{user_id:toInteger(row.user_id)})
MERGE(i:Item{movie_id:toInteger(row.movie_id)})
CREATE (u)-[:RatingItem {rating_item:toInteger(row.rating)}]->(i);
```

3. Requêtes
   a. Lister les 20 premières données de la base (nœuds et leur relations)
   b. Donner L'âge de l'utilisateur dont l'id est 2

c. Quelles sont les 3 « occupations » les plus populaires
d. Combien de valeurs différentes existent-ils pour l'attribut occupation ?
e. Quels sont les items produits en 1995 ?
f. Afficher les items qui ont été notés par l'utilisateur dont l'id est 2
g. Afficher les notes des items realises en 1995
h. Quels sont les 5 items les plus notés 1
i. Combien des items ont reçu au moins une fois la note 1
j. Quels sont les items qui ont une note moyenne >4
k. Lister les amis et les amis des amis du user 1 (Attention user 1 ne doit pas être ami de lui-même)
l. Quel est l'utilisateur qui a le plus d'amis en communs avec l'utilisateur 1.

Exercice 2. Supposons un graphique composé d'étudiants, de cours, de projets, de salles et de relations entre eux. Les nœuds et les relations sont spécifiés dans Cypher (le langage de requête graphique déclaratif utilisé par Neo4j)

Noeuds:

```
CREATE (s1: Student {studentID: "1", lastName: "Doe", firstName: "Ana", middleName: "Maria"})
CREATE (s2: Student {studentID: "2", lastName: "Ung", firstName: "Peter", middleName: "John"})
CREATE (s3: Student {studentID: "3", lastName: "Doe", firstName: "John"})
CREATE (s4: Student {studentID: "4", lastName: "Berre", firstName: "Stine"})
CREATE (s5: Student {studentID: "5", lastName: "Travolta", firstName: "John"})

CREATE (c1: Course {courseNr: "1", courseName: "Databases"})
CREATE (c2: Course {courseNr: "2", courseName: "Programming"})
CREATE (c3: Course {courseNr: "3", courseName: "Graphics"})

CREATE (p1: Project {projectNr: "34", projectName: "eCommerce database"})
CREATE (p2: Project {projectNr: "24", projectName: "eCommerce website"})
CREATE (p3: Project {projectNr: "13", projectName: "User interface"})
CREATE (p4: Project {projectNr: "26", projectName: "Reporting"})

CREATE (r1: Room {roomName: "Pascal"})
CREATE (r2: Room {roomName: "Seminar C"})
CREATE (r3: Room {roomName: "Alpha"})
CREATE (r4: Room {roomName: "Beta"})
```

Relationships

```
CREATE (c1) – [:TAKESPLACEIN] –> (r1)
CREATE (c1) – [:TAKESPLACEIN] –> (r3)
CREATE (c1) – [:TAKESPLACEIN] –> (r4)
CREATE (c2) – [:TAKESPLACEIN] –> (r2)

CREATE (s1) – [:ENROLLEDIN] –> (c1)
CREATE (s2) – [:ENROLLEDIN] –> (c1)
CREATE (s3) – [:ENROLLEDIN] –> (c2)
CREATE (s4) – [:ENROLLEDIN] –> (c1)

CREATE (s1) – [:WORKSON {hours: "1"} ] –> (p1)
CREATE (s1) – [:WORKSON {hours: "2"} ] –> (p2)
CREATE (s2) – [:WORKSON {hours: "3"} ] –> (p1)
CREATE (s2) – [:WORKSON {hours: "4"} ] –> (p2)
CREATE (s2) – [:WORKSON {hours: "1"} ] –> (p3)
CREATE (s2) – [:WORKSON {hours: "1"} ] –> (p4)
CREATE (s3) – [:WORKSON {hours: "1"} ] –> (p1)
CREATE (s3) – [:WORKSON {hours: "2"} ] –> (p2)
CREATE (s3) – [:WORKSON {hours: "3"} ] –> (p4)
```

Spécifier les requêtes suivantes dans Cypher et exécutez-les dans Neo4j.

1. Dans quelles salles se déroule le cours portant le numéro « 1 » ? Récupérer le nom du cours et les noms des salles
2. Combien d'heures et sur quels projets l'étudiant portant le numéro « 1 » travaille-t-il ? 3. Quels étudiants et combien d'heures travaillent-ils sur le projet portant le numéro de projet « 24 » ? Récupérer le nom du projet, le nom de famille de l'étudiant et le nombre d'heures travaillées correspondant sur le projet.
4. Quels étudiants travaillent sur quels projets et combien d'heures ? Récupérer le nom des étudiants, le nom des projets sur lesquels ils travaillent et le nombre d'heures correspondant. Classer les résultats par nom de famille des élèves. Limiter les résultats à quatre.
5. Quels étudiants travaillent sur plus de deux projets et sur combien de projets exactement ? Récupérer le nom des étudiants et le nombre de projets correspondant. Classer les résultats par nombre de projets.