TP 02
Bases des données orientées graphes Neo4j

Important :

Ce TP est noté. Vous devez soumettre un compte rendu (questions, requêtes et résultats) sur votre dépôt GitLab dans un dossier intitulé “TP02” au plus tard à minuit, avant le 10 septembre

Pour ce premier TP, chaque étudiant doit installer Neo4j sur sa machine, si ce n'est pas déjà fait, selon le système d'exploitation de son ordinateur.

1. Rendez-vous sur le site officiel de Neo4j, Suivez la procédure suivant votre système d'exploitation :
   https://neo4j.com/docs/getting-started/get-started-with-neo4j/

Exercice 1 :

Une fois Neo4j installé, il faut démarrer le serveur et ouvrir un navigateur à l’adresse http://localhost:7474. Si tout s’est bien déroulé, vous devriez voir s’afficher l’interface Web de manipulation de la base de données Neo4j (Figure 1).

Prenons quelques instants pour découvrir l’interface.

- La partie gauche regroupe les éléments de menu (les paramètres, les liens vers la documentation, quelques exemples de bases de données, ...);
- La partie du haut permet de saisir des commandes pour interagir avec la base de données. C’est ici que vous saisirez vos requêtes;
- Enfin, la partie principale et centrale vous permettra de visualiser le résultat des requêtes (ou les messages d’erreur). La visualisation des résultats pourra se faire sous forme graphique ou tabulaire comme nous le verrons par la suite.

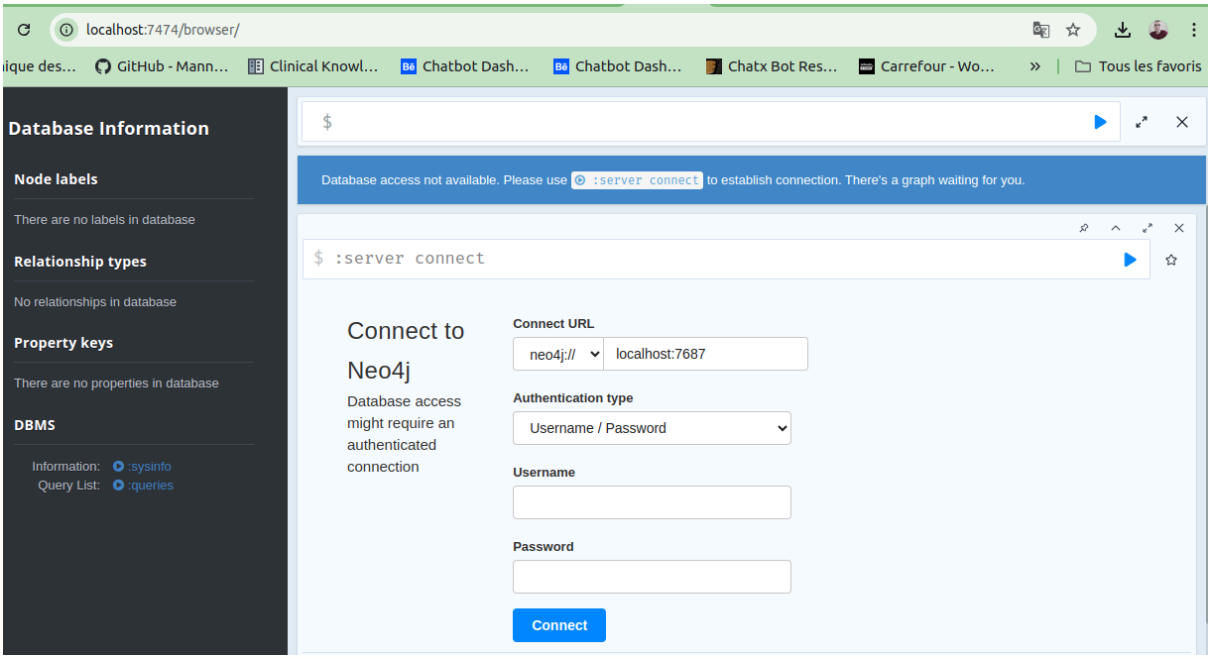


Figure 1: Interface Neo4j juste après son installation

Le démarrage du serveur dépend de votre système d'exploitation (Linux : systemctl {start|stop|restart} neo4j)

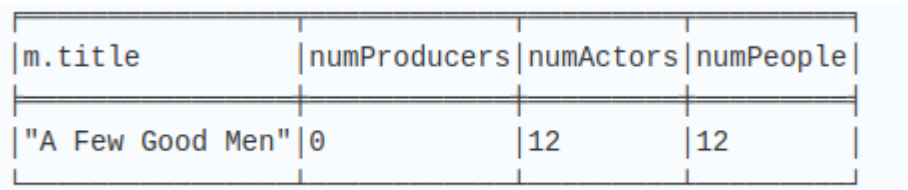
1. Connexion : Entrez neo4j comme nom d'utilisateur et neo4j comme mot de passe.
2. Après la connexion initiale, vous serez invité à définir un nouveau mot de passe.
3. Exécutez et interprétez les requêtes suivantes :
    a. CREATE (:Person:Actor {name: 'Tom Hanks', born: 1956})
    b. MATCH (n) DETACH DELETE n
    c. CREATE (n:Person:Actor {name: 'Tom Hanks', born: 1956}) CREATE (n)-[:KNOWS]->(n)
    d. CREATE (:Person:Actor {name: 'Tom Hanks', born: 1956})
    e. MATCH (n:Person)-[:KNOWS]->(n) RETURN n
    f. CREATE (n:Person {name: 'Alice'}) CREATE (n)-[:KNOWS]->(n)
    g. MATCH (n:Person)-[:KNOWS]->(n) RETURN n
    h. MATCH (n:Person {name: 'Alice'})-[r:KNOWS]->(n) SET r.since = 2020
    i. MATCH (n:Person {name: 'Alice'})-[r:KNOWS]->(n) DELETE r

j. MATCH (n) DETACH DELETE n CREATE (:Person:Actor {name: 'Tom Hanks', born: 1956})-[:ACTED_IN {roles: ['Forrest']}]->(:Movie {title: 'Forrest Gump', released: 1994})<-[:DIRECTED]-(:Person {name: 'Robert Zemeckis', born: 1951})
k. MATCH (n) DETACH DELETE n
l. CREATE (:Person:Actor {name: 'Tom Hanks', born: 1956})-[:ACTED_IN {roles: ['Forrest']}]->(:Movie {title: 'Forrest Gump', released: 1994})<-[:DIRECTED]-(:Person {name: 'Robert Zemeckis', born: 1951})
m. MATCH (ee:Person) WHERE ee.name = 'Emil' RETURN ee;
n. MATCH (ee:Person) WHERE ee.name = 'Emil'
CREATE (js:Person { name: 'Johan', from: 'Sweden', learn: 'surfing' }),
(ir:Person { name: 'Ian', from: 'England', title: 'author' }),
(rvb:Person { name: 'Rik', from: 'Belgium', pet: 'Orval' }),
(ally:Person { name: 'Allison', from: 'California', hobby: 'surfing' }),
(ee)-[:KNOWS {since: 2001}]->(js),(ee)-[:KNOWS {rating: 5}]->(ir),
(js)-[:KNOWS]->(ir),(js)-[:KNOWS]->(rvb),
(ir)-[:KNOWS]->(js),(ir)-[:KNOWS]->(ally),
(rvb)-[:KNOWS]->(ally)
o. MATCH (ee:Person)-[:KNOWS]-(friends)
WHERE ee.name = 'Emil' RETURN ee

Exercice 2 :

1. Réinitialisez la base de données en supprimant toutes les données.
2. Soit la requête Cypher suivante permet de créer une contrainte unique :
CREATE CONSTRAINT FOR (n:Movie) REQUIRE (n.title) IS UNIQUE
    a. Quels sont les effets de cette contrainte sur les données des nœuds de type Movie dans la base de données ?
    b. Quels types de données sont affectés par cette contrainte ?
    c. Que se passera-t-il si vous essayez d'ajouter deux nœuds de type Movie avec le même title ?
    d. Pourquoi est-il important d'utiliser des contraintes uniques dans une base de données ?

3. Donnez la requête qui permet de créer une contrainte équivalente pour le nœud Person sur la propriété name
4. Exécutez le bloc de script fourni dans movies.txt
5. Donnez le titre de tous les films
6. Donnez les titres des films produits par Lilly Wachowskt
7. Donnez les noms de toutes les personnes dans la base de données
8. Donnez le titre du film avec le plus grand nombre de personnes associées en tant que producteurs.
9. Donnez le titre du film avec le plus grand nombre de personnes associées en tant qu'acteurs
10. Donnez le titre du film avec le plus grand nombre de personnes associées, soit en tant que producteurs, soit en tant qu'acteurs.
11. Donnez la requête qui permet d'obtenir ce résultat :



m.title	numProducers	numActors	numPeople
"A Few Good Men"	0	12	12

12. Interprétez la requête suivante :

```
MATCH (p:Person)-[r1:PRODUCED]->(m:Movie)<-[:PRODUCED]-(p2:Person)
WHERE p <> p2
RETURN p.name, p2.name, m.title;
```

13. Donnez les personnes qui ont produit le plus grand nombre de films

Bon travail