TP Neo4j

1 Installation de Neo4j

La première étape consiste à installer Neo4j. Pour cela, rendez-vous sur le site de Neo4j . Nous ne détaillons pas les étapes d'installation de Neo4j car elles sont parfaitement décrites dans la documentation et dépendent de votre configuration matérielle. L'adresse suivante vous permettra de télécharger la dernière version de Neo4j et contient également des instructions détaillées pour l'installation : `https://neo4j.com/download-thanks/?edition=community&flavour=osx&release=3.0.3`. Une fois Neo4j installé, il faut démarrer le serveur comme indiqué lors de la procédure d'installation et ouvrez 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.

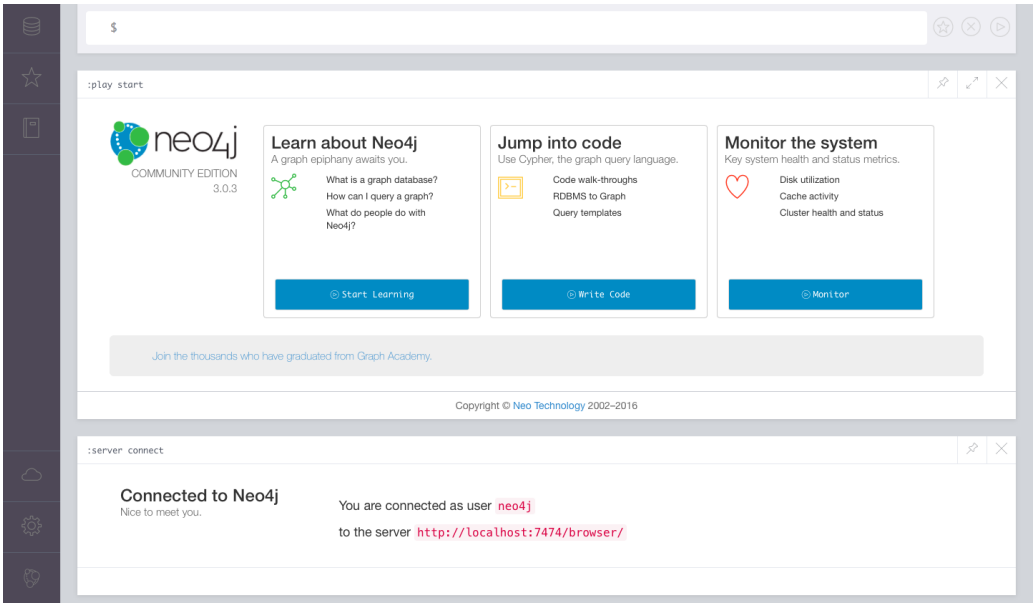

Figure 1: Interface Neo4j juste après son installation

2 Premières manipulations

2.1 Insertion manuelle de données

Dans un premier temps, nous allons effectuer quelques manipulations déconnectées du cas d'étude Twitter pour se familiariser avec Cypher, le langage de manipulation et d'interrogation des bases de données Neo4j. Donner les instructions Cypher qui permettent de réaliser les opérations suivantes :

1. Création de nœuds ayant les caractéristiques spécifiées dans le tableau 1.
2. Création de relations ayant les caractéristiques spécifiées dans le tableau 2

Id	Label	Attribut 1	Attribut 2
1	Personne	Nom : Bob	Age : 33
2	Personne	Nom : Alice	Age : 18
3	Personne	Nom : Ben	Age : 25
4	Véhicule	Type : Voiture	Modèle : Clio
5	Véhicule	Type : Moto	Modèle : R1
6	Véhicule	Type : Voiture	Modèle : A4

Table 1: Nœuds à insérer

De	Label	Vers	Attribut
1	Posséder	4	Depuis : 2014
2	Posséder	5	Depuis : 2007
3	Posséder	6	Depuis : 2010
1	Conduire	4	
2	Conduire	5	
3	Conduire	6	
2	Conduire	4	

Table 2: Relations à insérer

2.2 Quelques mises à jour et premières requêtes

Maintenant que quelques données sont présentes dans la base, nous allons pouvoir effectuer quelques manipulations et interrogations.

2.2.1 Manipulation de données

Effectuez les mises à jour suivantes :

1. Modifiez le nom de *Bob* par *Jack* et augmenter son âge de 10 ans;
2. Ajouter un attribut *PermisMoto* valant *OUI* aux personnes possédant une moto;
3. Ajouter un nouvel attribut *NbRoues* aux nœuds de type *Véhicule* et remplissez selon le type de véhicule.

2.2.2 Interrogation

Ecrivez et testez les requêtes suivantes :

1. Qui possède une voiture depuis au moins 2012 ? Vous ne renverrez que l'attribut *Nom* de la ou des personnes concernées.

2. Quels sont les propriétaires de moto ? Vous ne renverrez que l'attribut *Nom* de la ou des personnes concernées.

3. Qui conduit une voiture conduite également par *Alice* ?

3 Passons maintenant à Twitter...

La section précédente vous a permis de prendre en main le langage Cypher et d'exécuter quelques requêtes simples. Nous allons maintenant revenir au cas d'étude évoqué lors des séances de cours : Twitter.

3.1 Importation des données

Pour cela, nous allons d'abord commencer par charger les données dans la base. Pour ce faire, appliquez le protocole suivant :

1. Ecrire la requête Cypher qui permet de supprimer tous les nœuds et relations de la base de données

2. Télécharger l'archive contenant les données ainsi qu'un script python pour les charger sur Moodle

3. A l'endroit de votre choix, décompresser l'archive

4. Modifier la ligne 12 du script Python avec le mot de passe que vous avez saisi lors de l'installation de Neo4j

5. Si nécessaire, installer les librairies `pandas` et `py2neo` grâce à la commande
   `pip install --user py2neo pandas`

6. Exécuter le script python grâce à la commande `python loadCsv.py`

7. Valider la réussite de l'opération en vérifiant que les données ont bien été insérées à travers l'interface graphique.

3.2 Interrogation des données

Maintenant que les données sont insérées, vous devez mettre au point les requêtes Cypher afin de répondre aux besoins suivants :

1. Donner le nombre des utilisateurs;

2. Donner le nombre de tweets;

3. Donner le nombre d'hashtags;

4. Donner le nombre de tweets contenant le hashtag *actualite*;

5. Donner le nombre d'utilisateurs différents qui ont tweeté un tweet contenant le hashtag *valls*;

6. Donner les tweets qui sont des réponses à un autre tweet;

7. Donner le nombre de followers de *Spinomade* (sur Twitter en général);

8. Donner le nombre d'utilisateurs suivis par *Spinomade* (sur Twitter en général);

9. Donner le nom des followers de *Spinomade* (dans ce jeu de données);

10. Donner le nom des utilisateurs suivis par *Spinomade* (dans ce jeu de données);

11. Donner le nom des utilisateurs qui sont à la fois followers et followees de *Spinomade*;
12. Donner les utilisateurs ayant plus de 10 followers (dans ce jeu de données);
13. Donner les utilisateurs qui follows plus de 5 utilisateurs (dans ce jeu de données);
14. Donner les 10 tweets les plus populaires[1];
15. Donner les 10 hashtags les plus populaires[2];
16. Donner les tweets qui initient une discussion[3];
17. Quelle est la plus longue discussion ?
18. Pour chaque conversation, donnez-en le début et la fin.

4 Application : moteur de recommandation de hashtags

La dernière partie de ce TP a pour objectif la construction d'un moteur de recommandation de hashtags. Le principe est le suivant : soit h un hashtag que vous souhaitez inclure dans votre tweet, le système retournera une liste ordonnée $\langle (h_1, s_1), \ldots, (h_k, s_k) \rangle$ de hashtags, h_i (avec $1 \leq i \leq k$), associés à leur score de pertinence, s_i (avec $1 \leq i \leq k$). Le score de pertinence d'un hashtag est fonction de trois facteurs, enoncés et formalisés ci-dessous :

1. La co-occurrence. La co-occurence désigne le nombre de fois où 2 hashtags apparaissent dans le même tweet normalisé par la fréquence minimale des deux hashtags. On note $|h|$ le nombre de tweets où le hashtag h apparaît. Sur le même principe, on note $|h, h'|$ le nombre de tweets où les hashtags h et h' co-apparaissent. Le facteur de co-occurrence, $s^1(h, h_i)$, s'obtient donc par :

$$s^1(h, h_i) = \frac{|h, h_i|}{min(|h|, |h_i|)}$$

2. L'appartenance au champ lexical d'un utilisateur. Ce facteur désigne à quel point les utilsateurs qui mentionnent un hashtag ont pour habitude d'en utiliser un autre. Formellement, on note $|h, h'|_u$ le nombre d'utilisateurs qui ont utilisé au moins une fois les hashtags h et h' (pas nécessairement dans le même tweet) et $|u|$ le nombre d'utilisateurs. Le second facteur, de cohérence utilisateur, noté $s^2(h, h_i)$, s'obtient par :

$$s^2(h, h_i) = \frac{|h, h_i|_u}{|u|}$$

3. La cohérence dans la discussion. Ce facteur désigne à quel point deux hashtags apparaissent dans une discussion (selon la définition de discussion précédemment donnée). Formellement, on note $|h, h'|_d$ le nombre de discussions qui ont utilisé au moins une fois les hashtags h et h' (pas nécessairement dans le même tweet) et $|d|$ le nombre de discussions. Le troisième facteur, de cohérence dans la discussion, noté $s^3(h, h_i)$, s'obtient par :

$$s^3(h, h_i) = \frac{|h, h_i|_d}{|d|}$$

Finalement, le score d'un hashtag est défini comme une somme pondérée des 3 facteurs définis ci-dessus. Soit h le hashtag pour lequel on souhaite recommander d'autres hashtags et h_i un hashtag ($h \neq h_i$). Le score associé à h_i, noté s_i, est défini par :

$$s_i = \alpha s_i^1 + \beta s_i^2 + \gamma s_i^3$$

[1] Ici, la popularité se mesure au nombre de fois où le tweet est favori.
[2] Ici, la popularité se mesure au nombre de tweets dans lesquels ils apparraissent.
[3] Ici, une discussion est un enchaâ®nement de tweets deux à deux reliés par un lien de type *REPLY_ TO*;

Dans ce TP nous utiliserons les valeurs suivantes pour les constantes : $\alpha = 0,5$, $\beta = 0,25$ et $\gamma = 0,25$. Nous vous demandons de produire une application, reposant sur la base de données Neo4j, prenant en entrée un hashatg présent dans la base (vous renverrez un message d'erreur dans le cas contraire) et retournant les 20 hashtags les plus pertinents (avec les scores associés). Vous utiliserez le langage de votre choix (de préférence Java ou Python) pour réaliser l'application demandée.

Le code ci-dessous vous montre comment interfacer votre programme Java avec une base Neo4j.

```
import org.neo4j.driver.v1.*;

Driver driver = GraphDatabase.driver( "bolt://localhost", AuthTokens.basic( "neo4j",
    "neo4j" ) );
Session session = driver.session();

session.run( "CREATE (a:Person {name:'Arthur', title:'King'})" );

StatementResult result = session.run( "MATCH (a:Person) WHERE a.name = 'Arthur' RETURN
    a.name AS name, a.title AS title" );
while ( result.hasNext() )
{
    Record record = result.next();
    System.out.println( record.get( "title" ).asString() + " " +
        record.get("name").asString() );
}

session.close();
driver.close();
```

Le code ci-dessous vous montre comment interfacer votre programme Python avec une base Neo4j.

```
from py2neo import Graph, Path
graph = Graph()

tx = graph.cypher.begin()
for name in ["Alice", "Bob", "Carol"]:
    tx.append("CREATE (person:Person {name:{name}}) RETURN person", name=name)
alice, bob, carol = [result.one for result in tx.commit()]

friends = Path(alice, "KNOWS", bob, "KNOWS", carol)
graph.create(friends)
```