Projet Neo4j

A rendre: Une archive .zip comprenant:

- Un compte-rendu de TP au format **pdf** avec les réponses aux différentes questions (avec le code Cypher utilisé, et les réponses aux requêtes, éventuellement sous forme de copie d'écran).
- Le code Python **commenté** produit pour la partie 2.

1 Cas d'étude

Le cas d'étude proposé concerne un réseau social d'amateur de musiques. Le méta modèle du graphe que vous allez manipuler est présenté dans la Figure 1. Des utilisateurs se suivent entre eux et indiquent également les morceaux aimés. Pour chaque morceau, nous disposons des informations suivantes :

- le(s) album(s) dont il fait partie
- son interprète (et le(s) genre(s) de l'interprète)
- l'ensemble des noms utilisés dans les parolee du morceau

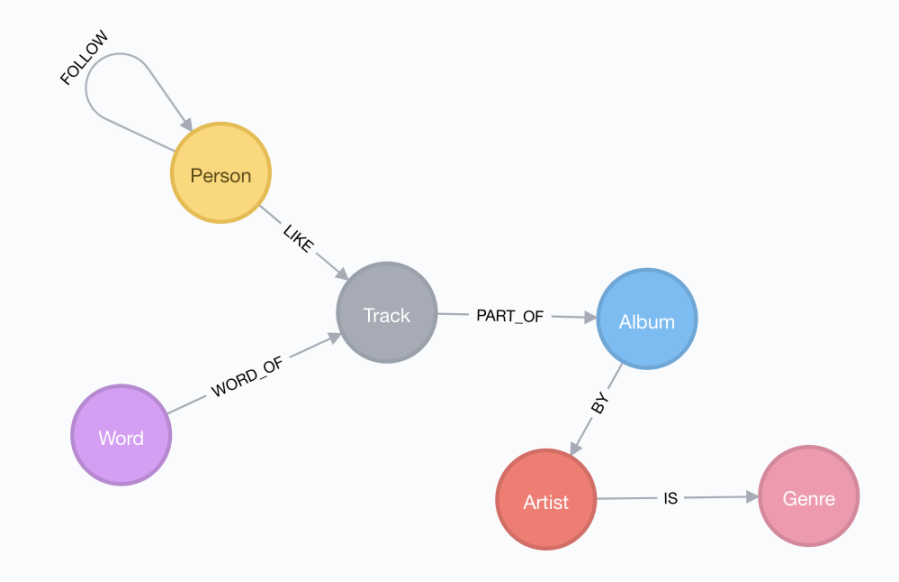

Figure 1: Méta modèle du graphe manipulé dans le cas d'étude

1.1 Importation des données

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

1. Retournez sur la fenêtre de manipulation des projets et des bases de données et créez une nouvelle base de données locale (nom et mot de passe à votre convenance). Ne la démarrez pas !

2. Télécharger le dump de la base de données en suivant le lien indiqué sous Moodle
3. Positionnez-la à l'endroit qui vous arrange
4. Dans la zone correspondant à votre nouvelle base de données, cliquez sur *Manage*, puis sur l'onglet *Terminal*
5. Entrez la commande suivante
   `bin/neo4j-admin load --from=<chemin absolu de disco.dump> --database=graph.db`
6. Si aucun message d'erreur n'apparaît, vous avez réussi !
7. Il ne vous reste plus qu'à démarrer la base et ré-ouvrir le navigateur pour commencer à manipuler le graphe [1]

1.2 Découverte des données

A l'aide de l'interface, listez tous les labels de nœuds et les types de relations ainsi que les attributs associés aux nœuds et relations. Les fonctions `labels`, `type` et `keys` vous seront utiles.

1.3 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 de nœuds par label;
2. Donner le nombre de relations par type;
3. Donner le nom du ou des genres de l'artiste *prince*;
4. Donner les morceaux (et les albums) qui apparaissent dans plusieurs albums (Remarque: la fonction `id` pourrait vous être utile;
5. Donner le nom des artistes qui ont sorti un album éponyme;
6. Donner le nom des albums (et de l'artiste associé) qui contiennent un titre ayant le même titre que celui de l'album (attention à la différence de casse);
7. Quels sont les 10 mots les plus utilisés par *eminem* dans ses chansons ?
8. Dans quels morceaux, un même mot est-il répété au moins 50 fois ? Affichez le nom du morceau, le mot en question et le nombre d'occurrences. Les résultats seront triés par ordre décroissant sur le nombre d'occurrences.
9. Qui sont les followers (un follower est une personne qui suit) de *Bogota Worcestershire* ?
10. Quels sont les chemins de taille au plus 3 qui existent entre *Bogota Worcestershire* et *Hebrides Afrikaans* selon la relation *FOLLOW* ?
11. Quel est la longueur du plus court chemin entre *Bogota Worcestershire* et *Hebrides Afrikaans* ?
12. Existe-t-il des influenceurs dans ce jeu de données ? Ici, nous supposerons qu'un influenceur possède au moins 3 fois plus de followers que de followees. Donnez le nom des influenceurs, ainsi que leur nombre de followers et de followees.

[1] Si la base refuse de démarrer et que dans les logs vous avez une erreur du genre " `ERROR Neo4j cannot be started because the database files require upgrading and upgrades are disabled in the configuration. Please set 'dbms.allow_upgrade' to 'true' in your configuration file and try again`":

- modifiez le fichier de `settings` (onglet *Settings*) en mettant la propriété `dbms.allow_upgrade` à `true`;
- dans l'onglet *Upgrade*, sélectionnez la version 3.4.7.

13. Combien de mots sont utilisés à la fois dans les chansons de l'album *abbey road* des *Beatles* et dans les chansons de l'album *relapse 2009* d'*Eminem* ?

14. Combien de paires de personnes ont au moins 10 tracks aimés en commun ?

2 Application : moteur de recommandation de hashtags

La dernière partie de ce TP a pour objectif la construction d'un moteur de recommandation de morceaux. Le principe est le suivant : soit u un utilisateur à qui l'on souhaite recommander des morceaux. On souhaite que le système retourne une liste ordonnée $\langle (t_1, s_1), \ldots, (t_k, s_k) \rangle$ de morceaux, t_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 morceau est fonction de trois facteurs, énoncés et formalisés ci-dessous :

1. L'adéquation au genre musical. On note Γ_u l'ensemble des genres aimés par l'utilisateur u. Un genre g sera dans Γ_u si u aime une chanson d'un artiste dont le genre est g. Cette adéquation sera exprimée par un facteur de *boost* si le genre d'un track t, noté g_t, appartient à Γ_u. Concrètement, le facteur de boost, noté $fb(u,t)$ se calcule ainsi :

$$\begin{cases} fb(u,t) & = & 2 & si\ g_t \in \Gamma_u \\ fb(u,t) & = & 1 & sinon \end{cases}$$

2. Le facteur social. Ce facteur retranscrit l'intuition que plus un morceau t est aimé par des personnes proches de u, plus l'utilisateur u sera susceptible de l'aimer. Pour cela, nous combinons deux critères : la taille du plus court chemin entre u et un utilisateur u' qui aime t, et le nombre de ces plus courts chemins. Formellement, nous notons $PCC(u,t)$ un plus court chemin entre u et t. Notons que ce chemin ne sera composé que de relations de type `FOLLOW` à l'exception de la dernière relation qui sera de type `LIKE`. Formellement, le facteur social, noté $fs(u,t)$, est calculé ainsi :
$$fs(u,t) = |PCC(u,t)| \times taille(PCC(u,t))$$

3. La cohérence thématique. Ce facteur retranscrit l'intuition qu'un utilisateur u aura tendance à aimer un morceau t dont la thématique des paroles est proche d'au moins un morceau déjà aimé par l'utilisateur. Nous notons V_t le vocabulaire du morceau t (considéré comme un sac de mots) et Θ_u l'ensemble des morceaux aimés par u. La similarité entre deux vocabulaires est calculée en utilisant le coefficient de Jaccard défini comme $Jaccard(t,t') = \frac{|V_t \cap V_{t'}|}{|V_t \cup V_{t'}|}$. Le facteur thématique, noté $ft(u,t)$, est calculé ainsi :

$$ft(u,t) = \max_{t' \in \Theta_u} Jaccard(t,t')$$

Finalement, le score d'un morceau t est défini comme une somme pondérée des 2 facteurs fs et ft définis ci-dessus, éventuellement boosté par fb. Soit u l'utilisateur pour qui on souhaite recommander des morceaux et T_u l'ensemble des morceaux recommandables ($\forall t \in T_u,\ t_i \notin \Theta_u$). Le score associé à t, noté $score(u,t)$, est défini par :

$$score(u,t) = fb(u,t) \times (\alpha \times fs(u,t) + \beta \times ft(u,t))$$

Dans ce TP nous utiliserons les valeurs suivantes pour les constantes : $\alpha = 0,25$, $\beta = 0,75$. Nous vous demandons de produire une application, reposant sur la base de données Neo4j, prenant en entrée un utilisateur présent dans la base (vous renverrez un message d'erreur dans le cas contraire) et retournant les 20 morceaux les plus pertinents (avec les scores associés).

Vous utiliserez le langage Python. Le code ci-dessous vous montre comment interfacer vos programmes Python avec une base Neo4j. Notez bien que pour l'authentification, le `username` est `neo4j`, et le mot de passe est celui de la base, que vous avez défini au moment de sa création (voir TP de prise en main).

Pour Python:

```
from py2neo import Graph

graph = Graph("bolt://localhost:7687", auth=("neo4j", "votremotdepassepourlabase"))

q = "match (t:Track) return t.name as track"
res = graph.run(q).to_table()

for record in res :
   print(record[0])
```

Un peu d'aide: `https://neo4j.com/developer/python/`. Il vous faut bien sûr récupérer la librairie `Py2neo`.