R5.A.10: Nouveaux paradigmes de base de données.

Feuille de TP n°2

Modélisation

Exercice 1 *Proposez la modélisation de la base de données graphe qui doit répondre à la spécification ci-dessous. Faites la modélisation sur papier, ensuite faites son implémentation sur Neo4J en proposant une instance que vous inventerez, avec au moins 10 produits et 5 clients.*

Nous souhaitons modéliser un système de recommandation par courrier électronique. Ces systèmes sont utilisés pour proposer aux clients les produits ou services les plus pertinents. Les spécialistes du marketing et les analystes doivent donc surveiller le comportement des clients pour lancer des campagnes de marketing par courrier électronique efficaces.
La base de données associée à notre système de recommandation par courrier électronique devra stocker les produits, pour lesquels nous connaissons titre, description, prix, disponibilité, possibilité d'expédition. Les produits sont organisés en catégories et chaque catégorie a un titre. La base stocke nom, email et date d'enregistrement pour chaque client. La promotion d'un produit peut être faite via des offres promotionnelles ayant un type. Un client peut voir un produit (et nous souhaitons stocker son nombre de clicks lors de cette visualisation), l'ajouter dans sa *WishList* ou l'acheter.

Exercice 2 *Supposons que nous travaillons dans un studio de cinéma et le travail est d'analyser des données concernant les films. Pour réaliser cette analyse, il faut établir une base de données graph dans Neo4j. Mais jusqu'à présent, nous avons seulement un fichier CSV : Movie.csv. Il nous faut d'abord importer le fichier dans Neo4j (Pas besoin de valeurs nulles dans la base de données).*

Pour importer le fichier, nous avons besoin de :

1. Comprendre toutes les données et leur structure dans le fichier CSV. Analysez-les
2. Modéliser la base de données à partir des données qui sont dans le fichier *Movie.csv* (disponible sur CELENE). Proposez donc un schéma pour votre base. Vous devez remarquer qu'un directeur peut diriger plusieurs films.
3. Comparer le fichier *Movie.csv* et le schéma pour découvrir les différences.
4. Importer des données dans Neo4j selon le schéma modélisé (`https://neo4j.com/developer/guide-import-csv/`). Pour cela suivre les instructions ci-dessous :
   (a) Votre objectif sera de faire une requête pour créer l'instance de la base de données à partir du fichier *Movie.csv*
   (b) Avant d'exécuter cette requête, il faut placer le fichier *Movie.csv* dans le bon répertoire de votre machine virtuelle :
      i. Télécharger le ficher *Movie.csv*, disponible sur CELENE, dans un répertoire de votre machine IUT.
      ii. Copier le fichier *Movie.csv* dans la racine de votre machine virtuelle. Pour cela, exécuter la commande suivante dans le Terminal de la machine IUT, en remplaçant `o012345678` par votre numéro et `172.16.2.xxx` par le numéro de votre machine :

```
scp -o 'ProxyJump o012345678@acces-tp.iut45.univ-orleans.fr' Movie.csv
o012345678@172.16.2.xxx:.
```

Cette commande demandera votre mot de passe à l'iut et ensuite le mot de passe de la machine virtuelle.

iii. Dans un terminal de machine virtuelle, vérifiez si vous voyez bien le fichier copié.

iv. Transférez le fichier *Movie.csv* au bon répertoire de votre machine virtuelle, autrement dit, placer le fichier *Movie.csv* dans le répertoire `/var/lib/neo4j/import`. Pour cela, exécuter la commande suivante dans le terminal de la VM :

```
sudo mv Movie.csv /var/lib/neo4j/import
```

La commande demandera le mot de passe de votre machine virtuelle

(c) Maintenant que vous avez le fichier *Movie.csv* dans le bon répertoire de votre VM, vous pouvez revenir à votre travail avec Neo4J pour construire la requête qui devra importer les données de *Movie.csv* de manière à construire l'instance de la base dont le schéma vous avez conçu.

(d) Ci-dessous nous démarrons la requête (version 1 à compléter) qui doit permettre de faire le chargement de la base à partir du fichier *Movie.csv*.

```
LOAD CSV WITH HEADERS FROM 'file:///Movie.csv' AS movie
WITH movie WHERE movie.boxOffice IS NOT NULL AND movie.class IS NOT NULL
WITH toInteger(movie.idMovie) AS idMovie,
movie.movieName AS movieName,
[continuer en associant les element du fichier csv aux attributs de la BD]
CREATE (m:Movie{idMovie : idMovie,title : movieName,... [et les autres attributs]})
[créer les autres noeuds]
[créer les relations entre les noeuds]
```

(e) Maintenant, observez l'instance que vous avez. Comparez-la avec les données du fichier csv. Que se passe t-il avec les tuples où certaines valeurs étaient nulles ? Le fait qu'un directeur puisse diriger plusieurs films est il pris en considération ? comment cela a été traité ?

(f) Voici une explication de la clause MERGE en Cypher :

> The MERGE clause either matches existing node patterns in the graph and binds them or, if not present, creates new data and binds that. In this way, it acts as a combination of MATCH and CREATE that allows for specific actions depending on whether the specified data was matched or created.
>
> For example, MERGE can be used to specify that a graph must contain a node with a Person label and a specific name property. If there isn't a node with the specific name property, a new node will be created with that name property.

(g) Supprimez votre instance de base de données.

```
MATCH (n) Detach delete n
```

(h) Refaites votre requête en utilisant MERGE à la place de CREATE (version 2).

(i) Analysez le résultat.

(j) Regardez le schéma de votre base.

```
call db.schema.visualization()
```