Dataswati
INDUSTRIES DATA-DRIVEN

Initiation au NoSQL

Etape 3 – BDD Graphe Neo4j – Fonctionnement et Cypher

A propos

L'objectif de ce support de cours est de comprendre le fonctionnement de Neo4j et d'aborder un premier niveau de prise en main via le langage Cypher (GQL).

Plan

- Définition
- Le graphe en quelque mots
- Le fonctionnement de neo4j
- Le langage Cypher
  - Les nœuds
  - Les relations
  - Les propriétés
  - Les requêtes
- Reference Card

Contenu

Définition

Parmi les bases de données NoSQL, celles orientées « graphes » connaissent un succès fulgurant du fait de leur souplesse et de leur évolutivité extrême. Ne reposant sur aucun schéma, un graphe peut accepter de nouveaux flux de données sans nécessiter d'interventions lourdes. À l'heure où les entreprises ont compris que la fusion de leurs silos de données (CRM, ERP, Web, etc.) est une condition sine qua non pour créer de la valeur (solutions d'intelligence prédictive, moteurs de recommandation), cette propriété du graphe se révèle essentielle.

En considérant les relations entre objets comme des données de premier ordre, de nouvelles perspectives d'exploration et d'analyse apparaissent et favorisent la découverte d'informations inédites. L'absence de barrières logiques ouvre également la porte à des modalités d'exploitation de la donnée en profondeur, sans nuire aux performances. Installés depuis de nombreuses années au cœur des infrastructures des géants du Web (Google, Facebook, LinkedIn), les graphes sont désormais accessibles au plus grand nombre.

Extrait Bases de données Orientées Graphe - Eyrolles

Propriété Dataswati ® - Tous droits réservés

Le graphe en quelques mots

Historique

C'est en 1735 que Leonhard Euler, introduit à l'académie de Saint-Pétersbourg le premier graphe schématisant le problème des 7 ponts de Königsberg. Ce problème consistait à trouver une promenade au travers de la ville de Königsberg permettant de partir et de revenir au même point en passant 1 seule fois par chaque pont. La schématisation sous forme de graphe a permis de prouver que la seule solution à ce problème était de construire un nouveau pont.
La théorie des graphes s'est alors développée durant les trois siècles consécutifs à cette découverte.

Définition générique

Un graphe est la représentation de plusieurs objets ayant une relation binaire entre eux, c'est-à-dire que chaque objet peut être reliée à un autre objet par l'intermédiaire d'un lien.

Les objets sont nommés nœuds (En : node) et les liens relations (En : relation/edge). Chaque relation n'ayant d'existence propre que par son ancrage entre deux nœuds.

Chaque relation peut être soit non orientée soit orientée, c'est-à-dire que la relation peut ou non déterminer un sens entre les deux nœuds.

Dans le premier cas, le **graphe sera dit non orienté** et dans le deuxième, le **graphe sera dit orienté.**

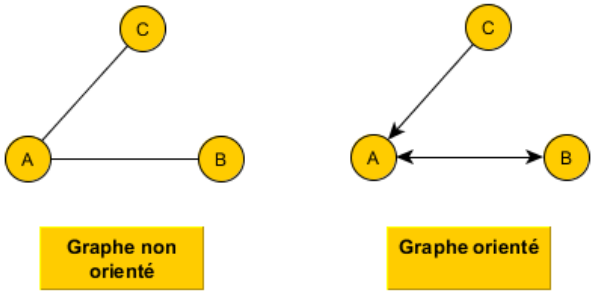

Figure 1 Différence entre graphe orienté et non orienté

Il est possible dans certains graphes d'associer une ou plusieurs valeurs aux nœuds et/ou aux relations. Ces valeurs sont notées attributs et le graphe est qualifié de **graphe attribué**. Pour exemple, dans la figure 1, les nœuds sont attribués par des lettres mais pas les relations.

> *L'**hypergraphe** définit un graphe sur lequel les relations ne sont plus binaires, c'est-à-dire qu'une même relation peut réunir plus de 2 nœuds.*
>
> *L'**hypernoeud** définit un graphe dans lequel les nœuds peuvent contenir eux-mêmes un graphe. Il s'agit dans ce cas d'une abstraction mathématique.*
>
> *(Marie-Aude Aufaure, 2011)(Aufaure, 2011) (Aufaure, 2011)*

Nous considérerons dans le reste du cours que le terme graphe recouvre la notion de **graphe orienté** et **attribué.**

Définition IT

Le terme graphe est un raccourci de la notion de base de données graphe (ou graph databases) proposant une technologie alternative de stockage et de requêtage de données aux systèmes de gestion de bases de données (SGBD) de type relationnel.

Les bases de données graphe s'inscrivent comme un sous-groupe du mouvement NoSQL entamé depuis le milieu des années 2000.

Cette technologie utilise le concept de graphe pour stocker la donnée sous forme de nœud et de relation. Elle change donc complètement le rapport traditionnel que nous avons de la donnée à travers des modèles relationnels.

Par exemple, il est possible de l'exploiter en utilisant la théorie mathématique des graphes. Mais nous aurons l'occasion tout au long de cette thèse de montrer les cas d'usages et leur pertinence.

Quelques technologies recensées en 2013

Ce tableau présente une liste non exhaustive de différentes technologies proposant un traitement graphe recensées en 2013.

Nom	Stockage Graph	Traitement Graph	Avantages
*dex	Natif	Natif	
AffinitY	Natif	Natif	
Atep Services	Non Natif *(Système RGBD pour le stockage)*	Natif *(Moteur d'induction pour transposition SQL - GQL)*	Langage de requêtée GQL.
FaceBook Graph	-	-	Intégré directement aux données Facebook. Le Graph Search a été ouvert dans le courant de l'automne 2013.
Franz' AllegroGraph	Natif	Non-Natif	
Google Pregel	-	-	Microcosme Google pour le développement.
InfiniteGraph	Non-Natif	Natif	
Neo4j	Natif	Natif	Langage de requête (Cypher) REST ACID Transaction
OrientDB	Natif	Natif	
Sones GraphDB	-	-	
Twiter's Flock DB	Non-Natif	Non Natif	Intégré directement aux données Twitter

Tableau 1 Liste non exhaustive des technologies graphe 2013. Inspiration (Ian Robinson, 2013)

Le fonctionnement de Neo4j

Principe

Neo4j est une base de graphe dont la structure de stockage est nativement en graphe, c'est-à-dire qu'elle va permettre de gérer les relations nativement sans passer par des tables d'association, mais également proposer des outils performant pour la recherche dans des graphes (calcul de distance, identification de chemins, ...).

Propriétés

Neo4j fonctionne comme un serveur de base de données traditionnel et fournit des services standards:

- Propriétés ACID (atomicité, cohérence, isolation et durabilité)
- Une interface Web (accessible par défaut sur le port 7474) permettant:
  - L'administration de la base
  - Requêtes directes dans la bdd
- API REST
  - Permettant d'accélérer les requêtes
- Interface compatibles
  - NodeJS
  - Java
  - PHP
  - Python

Langage

Pour simplifier son approche, Neo4j a inventé un langage de requête nommé Cypher (ou GQL) très proche du SQL et structuré symboliquement comme du graphe le rendant extrêmement intuitif.

Le langage Cypher

Les noeuds

En Cypher un nœud est représenté par des parenthèses : **()**

Le nœud peut être typé ce type permettra de le différentier des autres : **(:TYPE_NOEUD)**
Exemple : (:ACTOR), (:PERSON), ...

Les relations

Une relation est obligatoirement liée à 2 nœuds (pas plus) et orientée.

En Cypher elle est représentée par une flèche et d'accolades : **()-[]->()**

A l'instar du nœud, la relation peut également être typée ce type permettra de le différentier des autres : **()- [:TYPE_NOEUD] ->()**
Exemple : **(:ACTOR)- [:PLAY] ->(:MOVIE)**

Les propriétés

Les nœuds et relations peuvent être composés de propriétés (appelé aussi attributs).

La propriété est disposée sur un format clé:valeur libre dans les relations ou dans les nœuds.

Si la valeur n'est pas numérique, elle doit être entourée de guillemets.

Propriété d'un nœud	Propriété d'une relation
(:NŒUD{propriete:valeur})	**()-[:RELATION{propriete.valeur}]-()**

Note

Une base Neo4j n'est pas portée par un schéma directeur, autrement dit, chaque nœud et chaque relation sont indépendants et ce même si ils ont le même type. Il est donc possible de trouver 2 nœuds ou 2 relations n'ayant pas les mêmes propriétés.

L'absence de schéma directeur est volontaire et il permet de traiter des cas unitaires très facilement (ce qui est souvent le cas en graphe).

Néanmoins, cette liberté peut parfois amener à des comportement inattendus (absence de propriété mettant à l'écart certains objets lors de leur manipulation).

Ainsi, il est préférable, lorsque la topographie est posée et que l'on se dirige vers un développement en production, de mettre en place des moyens de contrôle d'intégrité des objets au niveau de la couche de développement.

Les requêtes

Une requête Cypher permets soit de collecter, de créer, de modifier ou encore de supprimer des objets.

Elle est composée de 2 parties, l'identification des objets à sélectionner et les actions à entreprendre sur ces objets.

	Expression Cypher
Identification	**MATCH ()-[]-()**

	Type	Expression Cypher
Action	Affichage d'objets	**RETURN**
	Modification	**MERGE, SET**
	Suppression	**DELETE**
	Création	**CREATE**

Variables

Il est possible d'utiliser des variables pour identifier les nœuds ou les relations à manipuler. L'identification de la variable se situe soit seule dans le nœud ou la relation **(n) ou [r]** soit avant les : lorsque le nœud ou la relation est typée **(n:NOEUD) ou [r:RELATION]**

!!! Une variable déclarée dans un match doit être unique !!!

Accès aux propriétés

L'accès aux propriétés d'un nœud ou d'une relation se fait par la déclaration d'un **.** soit **n.propriete ou r.propriete**.

Particularité

Il est possible de faire un CREATE ou un MERGE sans utiliser de MATCH dans ce cas la syntaxe se fait de la façon suivante:

```
CREATE / MERGE (:NŒUD{propriete1:valeur})-[:RELATION{propriete2:valeur}]->(:NŒUD{propriete3:valeur})
```

Exemples

Requête simple	**MATCH ()-[]-() RETURN ***	
Ajout d'un nœud	**CREATE ({propriete:valeur})**	
Ajout d'une relation (sans création de nœud)	**MATCH (a), (b) CREATE (a)-[{propriete:valeur}]->(b)**	
Modification d'un nœud	**MATCH (n) WHERE n.propriete = valeur SET n.propriete = valeur2**	**MATCH (n {n.propriete}) SET n.propriete = valeur2**
Modification d'une relation	**MATCH (n)-[r]-() WHERE n.propriete = valeur SET r.propriete = valeur2**	**MATCH ()-[r{propriete:valeur}]-() SET r.propriete = valeur2**
Suppression d'un nœud	**MATCH (n) WHERE n.propriete = valeur DELETE n**	**MATCH (n {n.propriete}) DELETE n**

Reference Card

Pour toutes informations sur Neo4j et surtout pour disposer de l'ensemble des requêtes sous forme synthétique: https://neo4j.com/docs/cypher-refcard/current/