Base de données Graph / Graph database



### Concept

L’idée des bases de données « graphe » est de conserver des entités et leurs interrelations.

Dans le cas ci-dessus, les entités (ou nœud) sont des personnes (ex : Morpheus), et les relations sont KNOWS et CODED\_BY.

Souvent, lorsque l’on peut construire une phrase en cheminant dans le graph, c’est que la base de données est bien conçue. Par exemple : *Morpheus connait Cypher, qui connait l’agent Smith et celui-ci a été codé par l’architecte.*

# Neo4j

Neo4j est une base de données “graphe” codé en Java. Actuellement, c'est le SGBD graphe le plus populaire.



Certains types de bases de données graphe sont en vérité soit construits sur des bases de données relationnelles comme MySQL ou Oracle (Non-native) ou encore dépendent de ressources/librairies externes.

Neo4j est complètement natif, ce qui veut dire qu’il n’est pas basé sur un autre produit pour la persistance et la gestion des données.

FlockDB est, selon la page Graph\_Database de wikipedia, basé sur MySQL.

## Exécution de Neo4j

Il existe deux moyens de démarrer le SGBD Neo4j.

### En mode « imbriqué » (embedded)

Lorsque votre application démarre, il démarre (ou créé) la base de données Neo4j.

### En mode « autonome »

Un peu comme MySQL ou Oracle, il démarre et écoute sur un port et est accessible par REST API (accès via http). Il existe plusieurs connecteurs, ou « drivers », qui permettent à des programmes écrits dans un autre langage (Python, PHP, …) d’interagir avec neo4j.

## Cypher

Dans le film “The Matrix”, le personnage Cypher est un traitre. Par contre, dans Neo4j, il est le moyen par excellence pour interroger la base de données.

Voici à quoi ressemble une requête Cypher dans une base de données de liens entre des villes.

### Exemple #1

Trouver le nom des villes qui sont à côté de Montréal

MATCH (ville)-[:A\_COTE]->(b) Trouve les nœuds qui ont un lien A\_COTE vers un autre nœud

WHERE ville.nom = 'Montréal' Pour ces nœuds, ne retenir que ceux dont la ville est Montréal

RETURN DISTINCT b.nom Retourne au programme le nom de ces villes

Autres détails :

* « ville » et « b » sont des noms arbitraires et est similaire à un ALIAS pour Oracle. On aurait pu écrire « c » et « vil »
* (ville) et (b) sont entre parenthèses, ce qui signifie qu’ils sont des nœuds
* [ :A\_COTE] est entre crochets, ce qui signifie qu’il est une relation de type A\_COTE
* Dans la section MATCH, la flèche est de (ville) vers (b). voir le - et ->
* DISTINCT permet de supprimer les doublons
* La ligne START n’est pas nécessaire dans les cas où on ne veut pas spécifiquement mentionner à Neo4j par où commencer sa recherche. Dans le cas ci-dessus, on dit à Neo4j de commencer à partir des nœuds. Pour dire de commencer à partir des relations, on aurait dit :

START r=relationship(\*)

### Exemple #2

Trouver le nom des villes qui sont à côté de Laval, mais pas de Montréal

MATCH (v)-[:A\_COTE]->(v2)

WHERE

v2.nom = 'Laval' AND

v3.nom = 'Montréal' AND

NOT ((v)-[:A\_COTE]->(v3))

RETURN v

Détails :

* La valeur de retour est le nœud au complet, pas seulement le nom du nœud.

Pour des exemples sur Neo4j, voir : <http://docs.neo4j.org/chunked/milestone/query-match.html>