

Concepts & utilisation de Neo4j

(Base de données orientée Graphe)

Neo4j (Base de données orientée Graphe)

- **Neo4j** est une base de données orientée graphe. Il existe 2 types d'objets dans le graphe: les nœuds et les relations. Il est possible d'associer des labels et des propriétés à chaque objet (à chaque nœud et relation).
- Le langage **Cypher** permet de lancer les requêtes dans Neo4j. Il permet d'enregistrer des objets (nœuds et relations) et de les retrouver dans le graphe.
- Voici la représentation textuelle des différents objets dans **Cypher** :

	Nœud	Relation
Pattern	<code>()</code>	<code>--></code>
Nom « de la variable »	<code>(a)</code>	<code>-[rel]-></code>
Label / type de relation	<code>(a:Personne)</code>	<code>-[rel:AMI]-></code>
Propriété	<code>(a{nom:'Stettler'})</code>	<code>-[rel{prof:true}]-></code>
Path longueur variable		<code>(a)-[*2]->(b)</code> <code>(a)-[*3..5]->(b)</code>
Exemples	<code>(alex)-->(ben)</code> <code>(alex:Personne)-[:AMI]->(ben)</code> <code>(a:Personne{nom:'Stettler'})-[:AMI{prof:true}]->(b{nom:'Hauri'})</code>	

Langage Cypher : fonctions principales :

- La fonction (l'instruction) **MATCH** permet de définir le modèle de recherche, principalement basé sur les relations ; **WHERE** permet de rajouter des contraintes au modèle.
- Les fonctions (instructions) **CREATE** et **DELETE** permettent de créer et supprimer les nœuds et les relations. **SET** et **REMOVE** sont utilisées pour redéfinir les valeurs des propriétés et affecter des labels aux nœuds.
- **RETURN** permet de retourner un objet ou une valeur : un (ou plusieurs) nœud(s), relation(s) et/ou propriété(s).

Création de la bdd Neo4j :

- Vous pouvez télécharger et installer en local sur votre ordinateur une version de **Neo4j Community Edition**, lancer le serveur `bin\neo4j console`, puis vous connecter via un navigateur sur <http://localhost:7474/> vous permettant de visualiser le contenu sous différentes formes, et exécuter toutes les instructions **Cypher** directement dans cette bdd (il existe également une version **AuraDB** disponible en accès libre dans le cloud que vous pouvez utiliser sans rien installer).

Marche à suivre pour accéder à une bdd Neo4j depuis Java :

- Si vous n'utilisez pas **Maven**, téléchargez le driver Neo4j pour java : <https://neo4j.com/developer/java/>
- Dans un nouveau projet IntelliJ, créez un sous-répertoire `lib` => copiez les drivers (fichiers .jar) dans `lib`, puis indiquez qu'il s'agit d'une librairie Java à utiliser dans votre projet :

File => ProjectStructure => Libraries => + Java => lib

- Dans votre application Java, vous pouvez vous connecter ainsi :

```
Syntaxe : Driver driver = GraphDatabase.driver("uri", AuthTokens.basic("neo4j","password"));
Session session = driver.session();
```

```
Exemple : Session bdd = GraphDatabase.driver("bolt://localhost:7687",
AuthTokens.basic("neo4j","password")).session();
```

- Puis exécuter des instructions **Cypher** à l'aide de la méthode `run`, récupérer les **Result** puis le **Record** :

```
Result res = bdd.run("MATCH (pers:Personne) RETURN pers");
Record rec = res.next();
```

Exemples d'instruction Cypher :

- Pour créer un (ou plusieurs) nœud(s), avec ou sans label(s) et propriété(s) :
CREATE (n)
CREATE (a), (b), (c)
CREATE (p:Personne)
CREATE (p:Personne {nom:'Stettler',prenom:'Christian'})
- Pour créer 2 nœuds ainsi qu'une relation entre eux :
CREATE (*label* :Personne {*propriété* nom:'DaMota'}) -[:*type relation* ASSISTE]-> (*label* :Personne {*propriétés* nom:'Stettler',prenom:'Christian'})
- Pour créer 1 relation entre 2 nœuds existants :
MATCH (p:Personne), (e:Ecole) WHERE p.nom='Stettler' AND e.nom='HEG' CREATE (p) -[:ENSEIGNE]-> (e)
- Pour récupérer les nœuds existants :
MATCH (n) RETURN n
MATCH (p:Personne) RETURN p
MATCH (p:Personne {nom:'Stettler'}) RETURN p
MATCH (p:Personne) WHERE p.nom='Stettler' RETURN p
- Pour récupérer les nœuds en fonction de leur relation :
MATCH (n) -[:ASSISTE]-> (p:Personne {nom:'Stettler'}) RETURN n
- Pour parcourir plusieurs relations (*par exemple, trouver l'ami d'un ami*) :
MATCH (chr {nom:'Stettler'}) -[:AMI]-> () -[:AMI]-> (amidami) RETURN chr, amidami
- Pour trouver un chemin entre 2 nœuds :
MATCH (a{nom:'Stettler'}), (b{nom:'DaMota'}), path = shortestPath((a)-[*..10]-(b)) RETURN path