INFO834 - TP BD NoSQL Orienté Graphe Neo4j

Britel Mohammed Yassine

Travail à faire

1. Tester au fur et à mesure sur l'exemple des universités

Voici la syntaxe utilisée dans le code pour interagir avec Neo4j, présentée en format texte:

Création de nœuds :

CREATE (nomDuNoeud:Label { propriete1: valeur1, propriete2: valeur2, ... })

Création de relations entre nœuds :

MATCH (start), (end) WHERE id(start) = $start_node_id AND id(end) = $end_node_id
CREATE (start)-[r:RELATION]->(end)

Suppression de tous les nœuds et relations :

MATCH (n) DETACH DELETE n

Requête pour récupérer tous les nœuds :

MATCH (n) RETURN n

Requête pour récupérer un nœud spécifique par son ID :

MATCH (n) WHERE id(n) = $node_id RETURN n

Requête pour récupérer toutes les relations :

MATCH ()-[r]->() RETURN r

Requête pour récupérer une relation spécifique par son ID :

MATCH ()-[r]->() WHERE id(r) = $relationship_id RETURN r

Configuration de propriétés sur une relation :

SET r = $properties

2. Travailler sur de grands graphes

Supprimer tous les nœuds et les relations existants :

```
MATCH (n) DETACH DELETE n
```

Créer une région avec des propriétés spécifiques :

```
MERGE (r:Region {code_region: $code_region}) SET r.nom_region =
$nom_region
```

Créer un département avec des propriétés spécifiques et le relier à une région :

```
MATCH (r:Region {code_region: $code_region}) MERGE (d:Departement
{code_departement: $code_departement}) SET d.nom_departement =
$nom_departement, d.code_region = $code_region MERGE
(r)-[:HAS_DEPARTMENT]->(d)
```

Créer une commune avec des propriétés spécifiques et la relier à un département :

```
MATCH (d:Departement {code_departement: $code_departement}) MERGE
(c:Commune {code_commune_INSEE: $code_commune_INSEE}) SET c.nom_commune
= $nom_commune, c.code_postal = $code_postal, ... MERGE
(d)-[:HAS_COMMUNE]->(c)
```

Créer un équipement avec un nom et un type spécifiques :

```
MERGE (e:Equipment {name: $name}) SET e.type = $type
```

Créer une comptabilité avec une année et un solde spécifiques :

```
MERGE (a:Accounting {year: $year}) SET a.balance = $balance
```

Créer un maire avec un nom spécifique :

```
MERGE (m:Mayor {name: $name})
```

Créer un résident avec un nom et une population spécifiques :

```
MERGE (r:Resident {name: $name}) SET r.population = $population
```

Lier une commune à ses informations d'équipement, de comptabilité, de maire et de résident :

```
MATCH (c:Commune {code_commune_INSEE: $code_commune_INSEE}) MERGE
(c)-[:HAS_EQUIPMENT]->(e) MERGE (c)-[:HAS_ACCOUNTING]->(a) MERGE
(c)-[:HAS_MAYOR]->(m) MERGE (c)-[:HAS_RESIDENT]->(r)
```

Ces requêtes Cypher sont exécutées dans le script Python pour interagir avec la base de données Neo4j et créer les nœuds et les relations nécessaires en fonction des données fournies dans le fichier CSV.

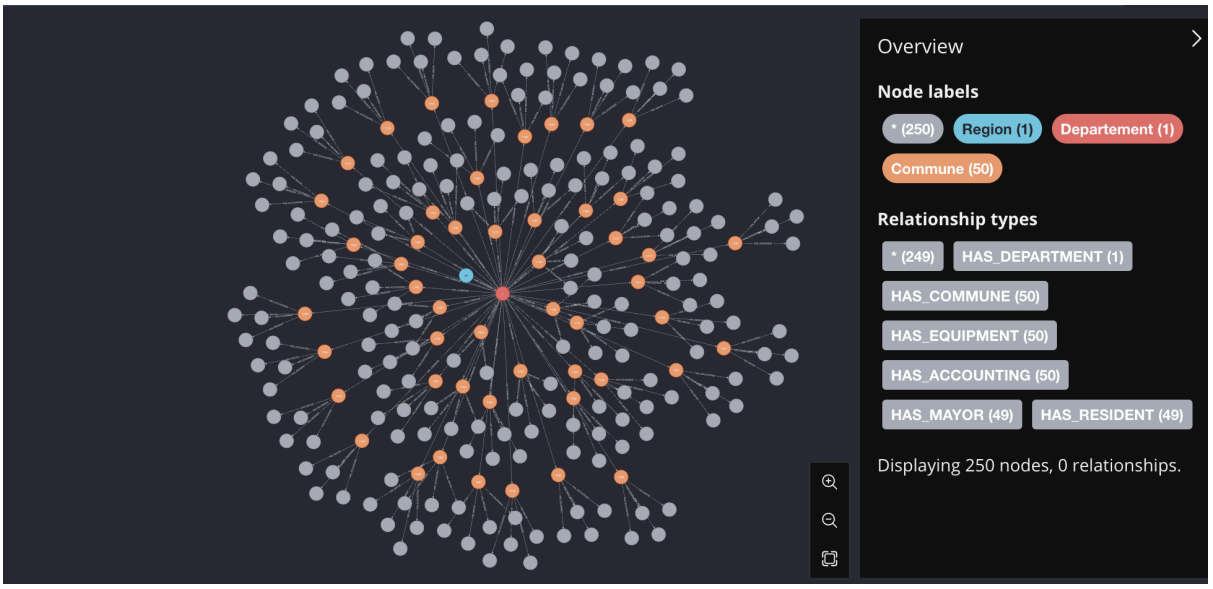
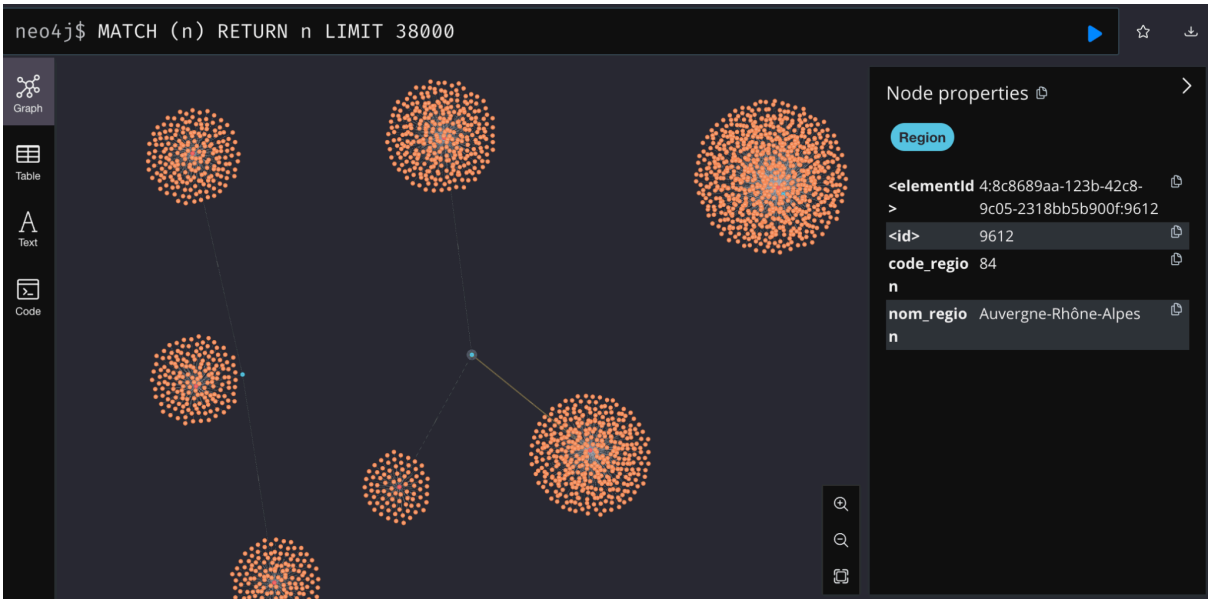
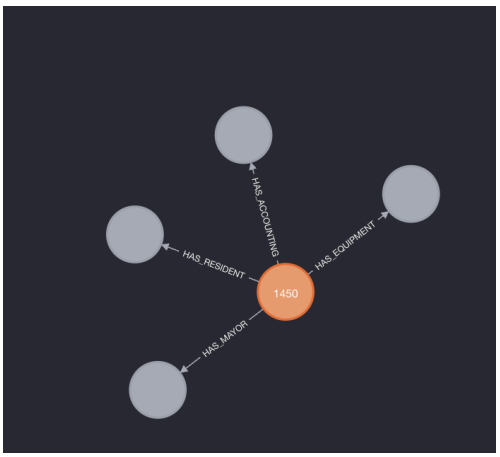
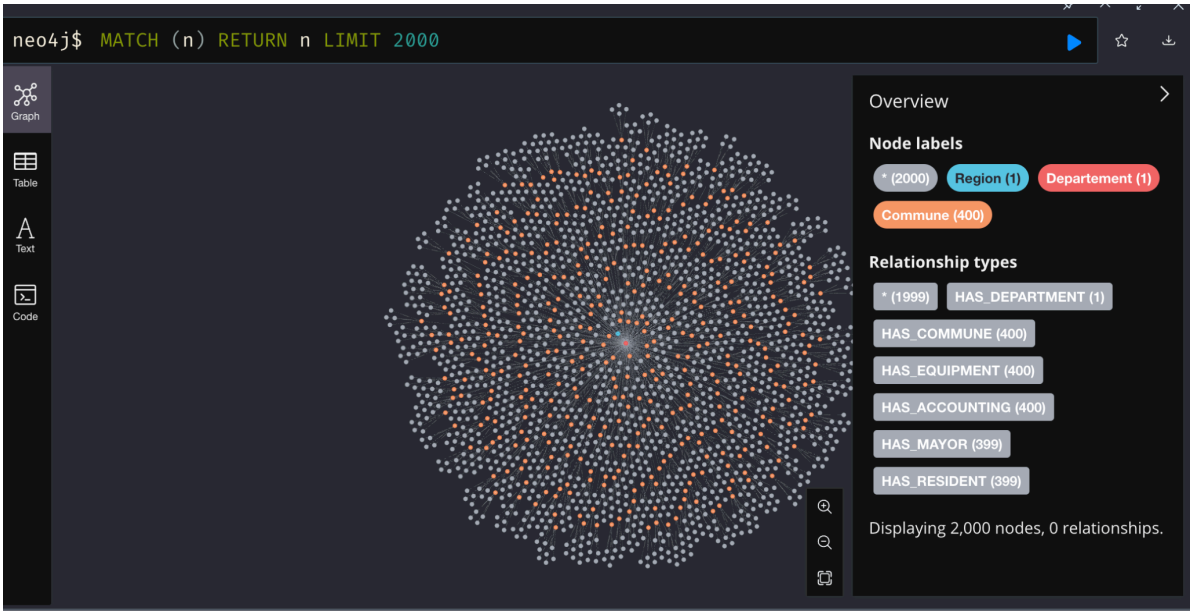
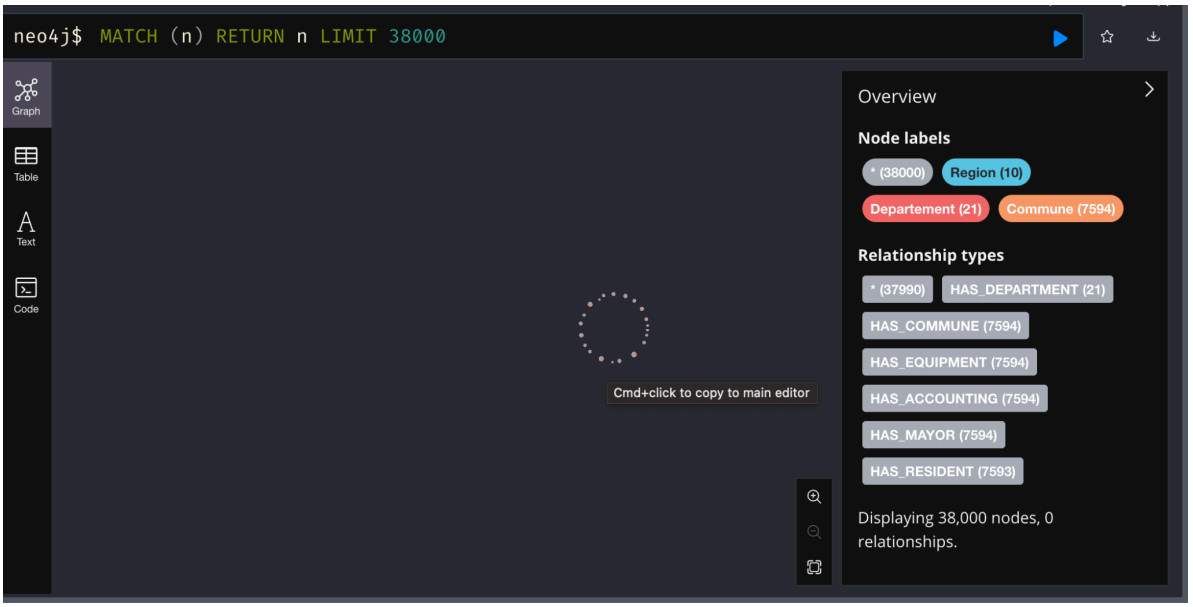


Image creates a relation between the Department his region and all comunes in it. And after i made de relation for the node communes with it Mayor,Account,Equipment and Residence.




HAS_ACCOUNTING
HAS_RESIDENT
HAS_EQUIPMENT
1450
HAS_MAYOR


neo4j$ MATCH (n) RETURN n LIMIT 2000
Graph
Table
Text
Code
Overview
Node labels
* (2000)
Region (1)
Departement (1)
Commune (400)
Relationship types
* (1999)
HAS_DEPARTMENT (1)
HAS_COMMUNE (400)
HAS_EQUIPMENT (400)
HAS_ACCOUNTING (400)
HAS_MAYOR (399)
HAS_RESIDENT (399)
Displaying 2,000 nodes, 0 relationships.


neo4j$ MATCH (n) RETURN n LIMIT 38000
Graph
Table
Text
Code
Cmd+click to copy to main editor
Overview
Node labels
* (38000)
Region (10)
Departement (21)
Commune (7594)
Relationship types
* (37990)
HAS_DEPARTMENT (21)
HAS_COMMUNE (7594)
HAS_EQUIPMENT (7594)
HAS_ACCOUNTING (7594)
HAS_MAYOR (7594)
HAS_RESIDENT (7593)
Displaying 38,000 nodes, 0 relationships.