Exploration de Neo4j

Faculté Polydisciplinaire de Ouarzazate
Master Mathématiques Appliquées pour la Science des Données

Présenté par :
Nour Eddine GAJJA
Abdelfattah BOUHLALI
Driss ABATOUR
Mohamed MALLOUKI

Encadré par :
Salma GAOU
Charaf Hamidi

8 mai 2024

PLAN :

Introduction

Introduction à Neo4j

Qu'est-ce que Neo4j ?

Neo4j est une base de données graphique leader qui utilise des structures de données graphiques avec des nœuds, des arêtes, et des propriétés pour représenter et stocker des données. Le modèle graphique permet une représentation flexible, intuitive et riche des relations entre les données.

Concept de bases de données graphiques

Contrairement aux bases de données relationnelles, les bases de données graphiques sont conçues pour traiter les relations entre les données comme aussi importantes que les données elles-mêmes. Elles sont optimales pour gérer des données interconnectées et complexes.

Importance et applications de Neo4j

Pourquoi Neo4j est important ?

Neo4j offre une performance inégalée lorsqu'il s'agit de gérer des données interconnectées, ce qui est crucial dans de nombreux domaines tels que la détection de fraude, la recommandation de produits, et la recherche scientifique.

Applications de Neo4j

Neo4j est utilisé par des entreprises du monde entier pour des applications telles que les réseaux sociaux, les systèmes de recommandation, la gestion de la chaîne logistique, et bien plus encore, exploitant sa capacité à modéliser des données de manière très connectée.

Nœuds, relations

Nœuds

Les nœuds sont les entités fondamentales dans une base de données Neo4j. Ils représentent des entités individuelles telles que des personnes, des lieux ou des objets, et sont souvent utilisés pour stocker des données de manière structurée.

Relations

Les relations sont des liens entre les nœuds qui représentent les connexions ou les interactions entre les entités. Elles permettent de modéliser les relations complexes et les dépendances dans les données.

Propriétés et Étiquettes

Propriétés

Les propriétés sont des paires clé-valeur associées aux nœuds et aux relations, permettant de stocker des informations supplémentaires pertinentes. Elles offrent une flexibilité pour ajouter des détails spécifiques à chaque entité ou relation.

Étiquettes

Les étiquettes sont utilisées pour regrouper des nœuds similaires, facilitant ainsi les requêtes et l'analyse des données. Elles permettent de catégoriser les nœuds en fonction de leurs caractéristiques communes.

Le modèle de données graphique de Neo4j

Modèle de données graphique

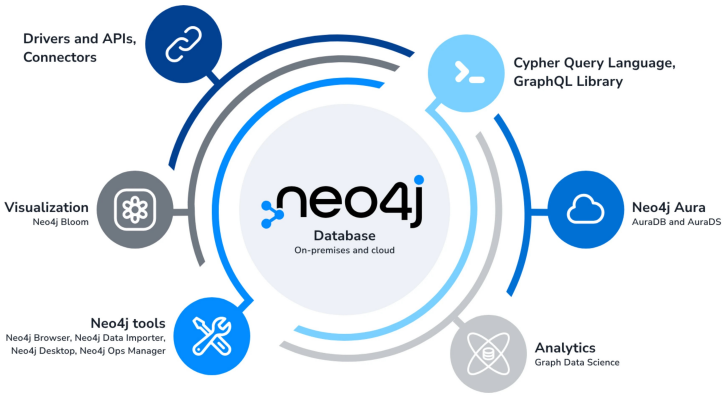
Le modèle de données graphique de Neo4j est conçu pour représenter les données de manière naturelle, en mettant l'accent sur les relations entre les données, ce qui permet une analyse plus intuitive et des requêtes plus performantes.

Avantages du modèle graphique

Ce modèle est particulièrement efficace pour les données interconnectées, offrant une flexibilité et une performance supérieures par rapport aux modèles de données traditionnels, surtout quand il s'agit de naviguer dans des relations complexes.

Architecture de Neo4j

Architecture de Neo4j



Clustering, réplication, et haute disponibilité

Clustering

Le clustering dans Neo4j permet une architecture robuste avec des serveurs qui découvrent les uns les autres et gèrent la charge de manière dynamique.

Réplication

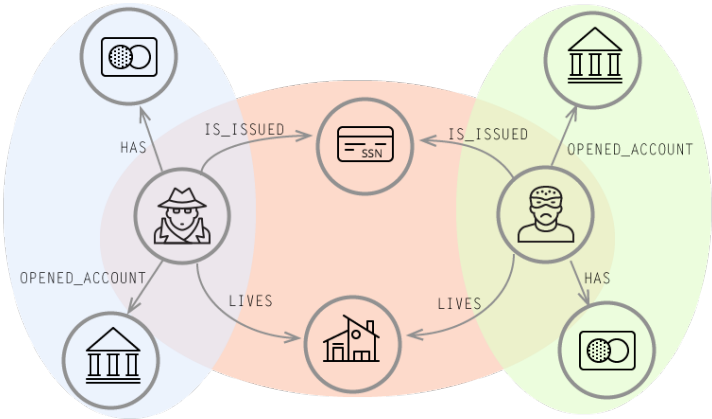
La réplication est réalisée à l'aide du protocole Raft, assurant la cohérence des données à travers le cluster.

Haute disponibilité

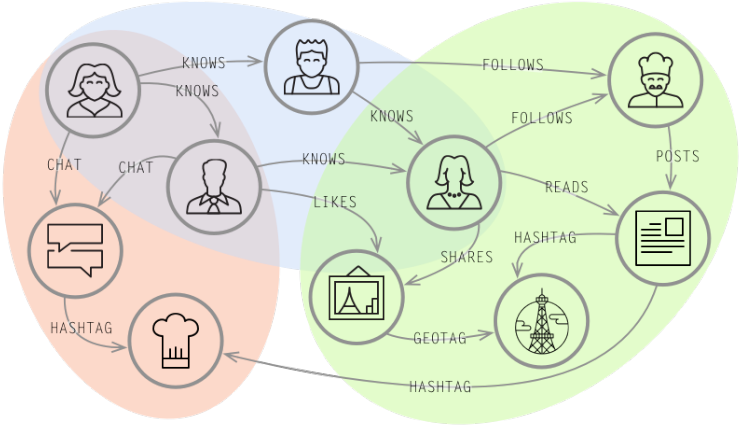
Neo4j est conçu pour être hautement disponible, avec une infrastructure auto-réparatrice et des sauvegardes automatisées pour une récupération rapide en cas de sinistre.

Cas d'Utilisation Réels

Détection de fraude



Reseaux Sociaux



Cypher Query Language

Syntaxe de base et structures de requêtes

Syntaxe de base

Cypher est un langage de requête déclaratif qui permet d'exprimer des motifs et des relations dans les graphes. Les requêtes Cypher se composent généralement de clauses comme MATCH, WHERE et RETURN pour récupérer des données.

Structures de requêtes

MATCH identifie les motifs à trouver dans le graphe.
WHERE filtre les motifs en fonction de conditions spécifiques.
RETURN spécifie les données à récupérer à partir des motifs correspondants.

Exemples de requêtes

Créer un nœud

CREATE (:Person name : 'simo', age : 30)

Créer une relation entre deux nœuds :

MATCH (a :Person), (b :Person) WHERE a.name = 'simo' AND b.name = 'walid' CREATE (a)-[:FRIEND]->(b)

Trouver tous les nœuds d'un type spécifique :

MATCH (p :Person) RETURN p

Démonstration

Création des joueurs et équipes

Création des joueurs

- CREATE (hakimi :Joueur nom : 'Hakimi', prenom : 'Achraf', age : 23) ;
- CREATE (cristiano :Joueur nom : 'Ronaldo', prenom : 'Cristiano', age : 36) ;
- CREATE (messi :Joueur nom : 'Messi', prenom : 'Lionel', age : 34) ;
- CREATE (mbappe :Joueur nom : 'Mbappe', prenom : 'Kylian', age : 23) ;

Création des équipes

- CREATE (psg :Equipe nom : 'PSG') ;
- CREATE (fcb :Equipe nom : 'FCB') ;
- CREATE (alnasser :Equipe nom : 'AlNassr') ;

Création des entraîneurs et des relations entre joueurs et équipes

Création des entraîneurs

- CREATE (entraineur1 :Entraineur nom : 'EntraineurPSG') ;
- CREATE (entraineur2 :Entraineur nom : 'EntraineurFCB') ;
- CREATE (entraineur3 :Entraineur nom : 'EntraineurAlNassr') ;

Création des relations entre joueurs et équipes

- CREATE (hakimi)-[:JOUE_POUR]->(psg) ;
- CREATE (cristiano)-[:JOUE_POUR]->(alnasser) ;
- CREATE (messi)-[:JOUE_POUR]->(fcb) ;
- CREATE (mbappe)-[:JOUE_POUR]->(psg) ;

Création des R. entre entraîneurs et équipes - R. d'amitié

Création des relations entre entraîneurs et équipes

- CREATE (entraineur1)-[:ENTRAINE]->(psg) ;
- CREATE (entraineur2)-[:ENTRAINE]->(fcb) ;
- CREATE (entraineur3)-[:ENTRAINE]->(alnasser) ;

Création des relations d'amitié

- CREATE (mbappe)-[:AMI]->(hakimi) ;
- CREATE (messi)-[:AMI]->(cristiano) ;

Exécution de requêtes en direct

Trouver tous les joueurs d'une équipe

- MATCH (j :Joueur)-[:JOUE_POUR]->(e :Equipe nom : 'PSG') ;
- RETURN j.nom, j.prenom, j.age ;

Cette requête retourne les noms, prénoms et âges de tous les joueurs qui jouent pour le PSG.

Trouver l'entraîneur d'une équipe

- MATCH (e :Equipe nom : 'FCB')<-[:ENTRAINE]-(entraineur) ;
- RETURN entraineur.nom ;

Cette requête retourne le nom de l'entraîneur du FC Barcelone.

Travailler avec les Données dans Neo4j

Importation de données dans Neo4j à partir de différentes sources

Importation à partir de fichiers CSV

Vous pouvez importer des données à partir de fichiers CSV en utilisant la commande LOAD CSV dans Cypher. Cela fonctionne bien pour les données tabulaires ou structurées.

Structures de requêtes

LOAD CSV WITH HEADERS FROM 'file :///fichier.csv' AS ligne
CREATE (:TypeDeNoeud propriete1 : ligne.valeur1, propriete2 : ligne.valeur2, ...).

Example Importation à partir de fichiers CSV

Fichier CSV :

name	ville	age
hassan	casa	25
walid	fes	23
fatima	agadir	65

cypher

```
LOAD CSV WITH HEADERS FROM 'file :///employees.csv' AS row
CREATE (:person  name : row.name, ville : row.ville, age :
toInteger(row.age))
```

Importation de données dans Neo4j à partir de différentes sources

Importation à partir de fichiers JSON

Pour importer des données à partir de fichiers JSON dans Neo4j, vous pouvez utiliser l'outil apoc.load.json

Structures de requêtes

CALL apoc.load.json('file :///chemin/vers/fichier.json') YIELD value
CREATE (:TypeDeNoeud propriete1 : value.propriete1, propriete2 : value.propriete2, ...)

Example Importation à partir de fichiers JSON

fichiers JSON

```
{
  "products": [
    {
      "id": 1,
      "name": "Laptop",
      "category": "Electronics",
      "price": 1200
    },
    {
      "id": 2,
      "name": "Smartphone",
      "category": "Electronics",
      "price": 800
    },
    {
      "id": 3,
      "name": "Chair",
      "category": "Furniture",
      "price": 100
    }
  ]
}
```

cypher

CALL apoc.load.json(”file :///products.json”) YIELD value UNWIND value.products AS product MERGE (p :Product id : product.id) SET p.name = product.name, p.category = product.category, p.price = product.price ;

Exportation de données depuis Neo4j pour analyse et reporting

Exportation vers des fichiers CSV :

Vous pouvez exporter des données depuis Neo4j vers des fichiers CSV à l'aide de la commande 'EXPORT CSV' dans Cypher. Cette méthode est utile si vous souhaitez analyser les données dans des outils d'analyse de données ou des tableurs

Structures de requêtes

CALL apoc.export.csv.all('/chemin/vers/fichier.csv',)

Exportation de données depuis Neo4j pour analyse et reporting

Exportation vers des fichiers JSON :

Vous pouvez également exporter des données depuis Neo4j vers des fichiers JSON à l'aide de la fonction apoc.export.json.query. Cela peut être utile si vous avez besoin de manipuler les données dans des applications qui acceptent le format JSON.

Structures de requêtes

CALL apoc.export.json.query("MATCH (n) RETURN n", '/chemin/vers/fichier.json',)

Administration et Optimisation

introduction

neo4j-adminet neo4jsont des outils de ligne de commande pour gérer et administrer un SGBD Neo4j. Les deux sont installés dans le cadre du produit et peuvent être exécutés avec un certain nombre de commandes. Les neo4jcommandes sont équivalentes aux commandes les plus importantes de la neo4j-admincatégorie serveur.

CLI Neo4j Command Line Interface

La CLI Neo4j est un outil de ligne de commande qui permet aux utilisateurs d'interagir avec une instance de base de données Neo4j via une interface en ligne de commande. Cela permet aux utilisateurs d'exécuter diverses opérations sur la base de données, telles que la création et la gestion de données, l'exécution de requêtes Cypher, etc., sans avoir besoin d'une interface graphique. **Les principales fonctionnalités de la CLI Neo4j peuvent inclure :**

- Exécution de requêtes Cypher.
- Gestion des utilisateurs et des autorisations.
- Importation et exportation de données.
- configuration de l'instance de base de données

CLI Neo4j Command Line Interface

cli neo4 j

CLI Neo4j (Command Line Interface) :

- Pour exécuter une requête Cypher, vous pouvez utiliser la commande cypher suivie de votre requête. Par exemple : cypher ”MATCH (n) RETURN n”
- Pour gérer les utilisateurs et les autorisations, vous pouvez utiliser les commandes user et role.
- Pour importer des données, utilisez la commande import.
- Pour surveiller et configurer votre instance Neo4j, vous pouvez utiliser des commandes telles que info, metrics, set, etc.

rôles d'administrateur

Les administrateurs (ou "admins") peuvent avoir différents rôles et responsabilités en fonction du système ou de l'organisation dans laquelle ils opèrent. Voici quelques-uns des rôles d'administrateur courants que l'on trouve souvent :

les diffirents rolesrôles d'administrateur

- **Super Administrateur**
- **Administrateur Système**
- **Administrateur de Base de Données (DBA)**
- **Administrateur Réseau**
- **Administrateur d'Application**
- **Administrateur de Sécurité**

les différents types d'autorisations

Dans un système d'autorisation, il existe différents types d'autorisations qui déterminent les actions qu'un utilisateur peut effectuer sur les ressources du système. Voici quelques-uns des types d'autorisations courants

l es types d'autorisations

- Lecture (Read)
- Écriture (Write)
- Exécution (Execute)
- Accès (Access)
- Création (Create)
- Modification (Update)
- Administration (Admin)
- Suppression (Delete)
- Propriétaire (Owner)
- Partage (Share)

Ressources et communauté

Ressources

Documentation et Tutoriels

Guides pour débutants, manuels Cypher, et tutoriels variés sur Neo4j Docs.

Assistance Technique

Base de connaissances et support pour clients entreprise sur Neo4j Support.

Communauté et Événements

Communauté Neo4j

Forums, collaborations, et événements sur Neo4j Community.

Événements Neo4j

Conférences et webinaires à venir sur Neo4j Events.

Conclusion

Conclusion

Récapitulatif

Nous avons parcouru les bases de Neo4j et le concept de bases de données graphiques, mettant en lumière l'importance et les applications de cette technologie dans divers domaines.

Concepts Clés

Les concepts de base tels que les nœuds, les relations, les propriétés et les étiquettes, ainsi que le modèle de données de graphique de Neo4j, ont été présentés en détail.

Conclusion

Cypher Query Language

Nous avons exploré la syntaxe et les structures de requêtes de Cypher, avec des exemples concrets pour interroger et manipuler des données graphiques de manière efficace.

Architecture et Performances

L'architecture de Neo4j, incluant le stockage de données, l'indexation et les techniques d'optimisation des performances, ainsi que des considérations sur le clustering, la réplication et la haute disponibilité, ont été abordées.

Conclusion

Cas d'utilisation

Nous avons examiné plusieurs cas d'utilisation réels illustrant la polyvalence et l'efficacité de Neo4j dans divers scénarios, démontrant ainsi son utilité dans le monde réel.

Démonstration et Ressources

Enfin, nous avons réalisé une démonstration pratique de la création d'un petit graphique de données et de l'exécution de requêtes en direct pour mettre en évidence la puissance de Neo4j. Nous avons également discuté des ressources disponibles et de la communauté dynamique de Neo4j pour continuer votre apprentissage et développement.

Merci pour votre attention !

Si vous avez des questions ou des commentaires,
nous serons heureux de vous répondre

Webographie

- Neo4j Official Website : www.neo4j.com
- Neo4j Documentation : www.neo4j.com/docs/
- Neo4j Developer Guides : www.neo4j.com/developer/guide/
- Neo4j Community : www.community.neo4j.com
- Neo4j on GitHub : www.github.com/neo4j
- Neo4j Blog : www.neo4j.com/blog/
- Top 10 Use Cases : wwww.go.neo4j.com/rs/710-RRC-335/images/Neo4j-Top-10-Use-Cases-FR-A4.pdf