Introduction à Neo4J

Présentation de Neo4J
Présentation de CQL

Présentation de Neo4J

Neo4j est un système de gestion de base de données basé sur les graphes, développé en Java par la société **Neo technology**.
https://db-engines.com/en/system/Neo4j

Langage de programmation : Java
Date de sortie initiale : 2007
Dernière version : 5.23 (août 2024)
Licence : Open Source (Licence publique générale GNU version 3 et AGPL-3.0

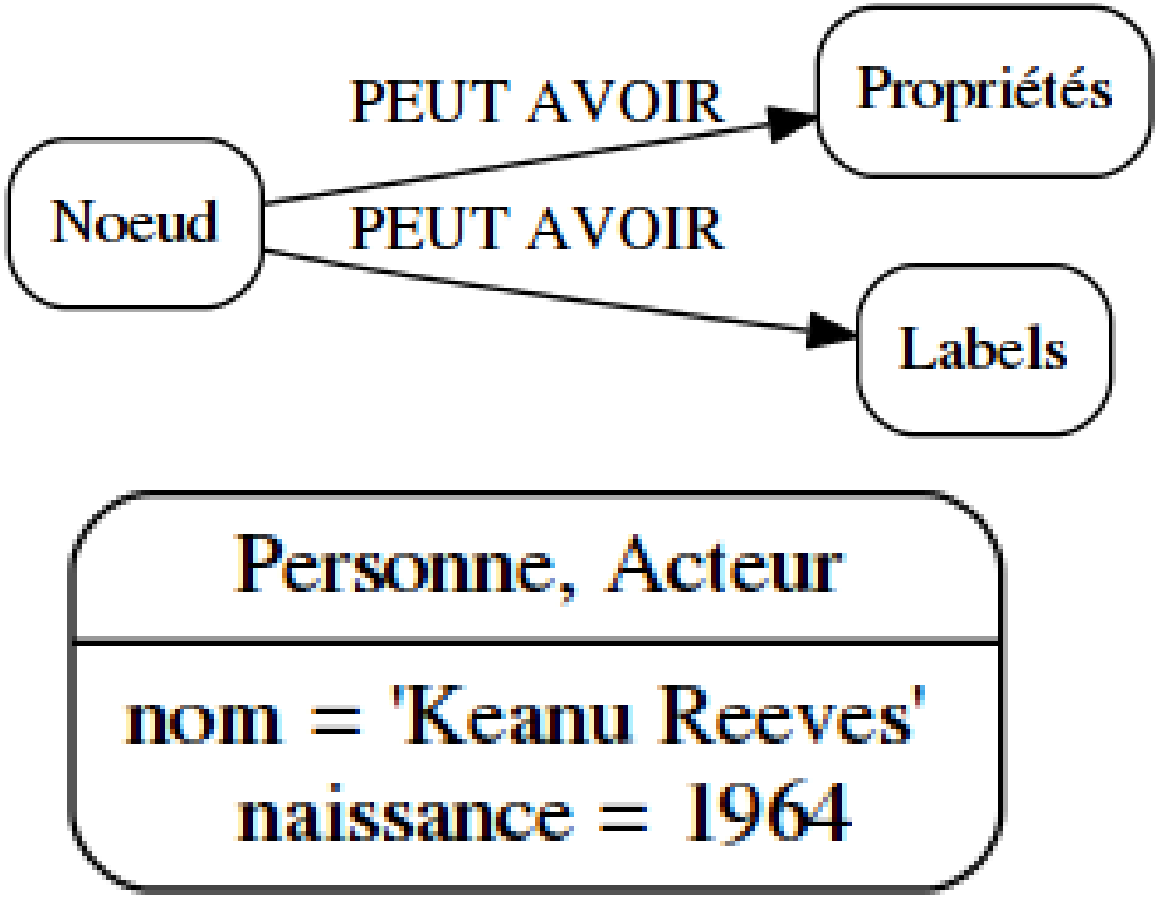
Neo4j dispose d’un langage de requête déclaratif:
Cypher Query Langage (**CQL**)

Présentation de Neo4J

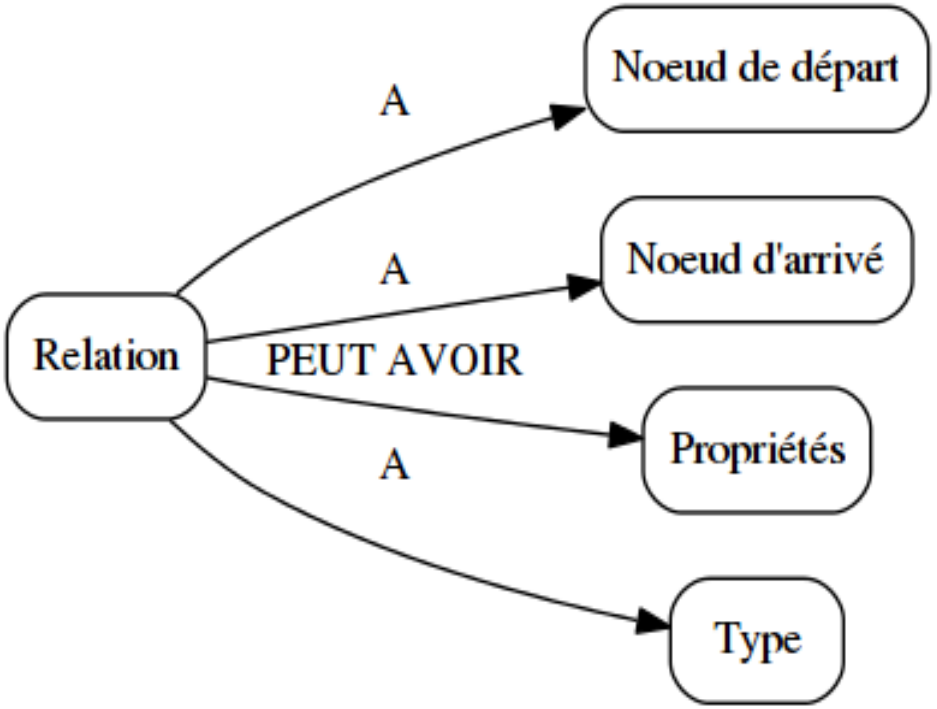
Ses principales caractéristiques sont les suivantes :

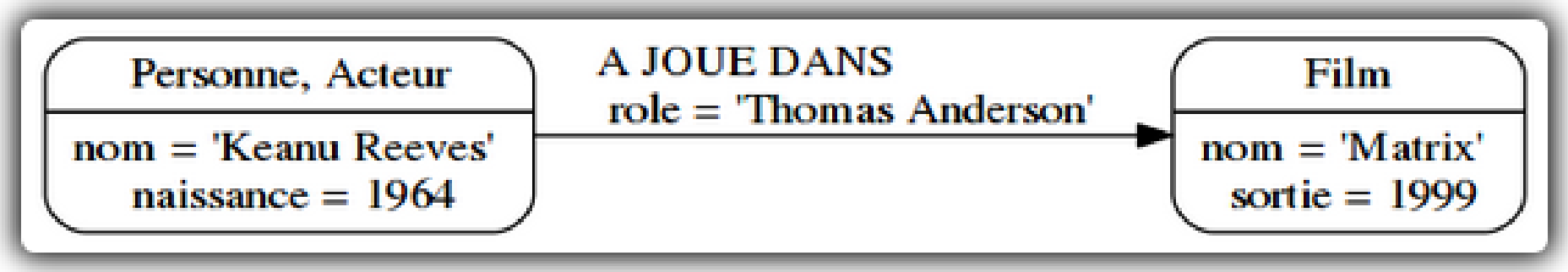
- **Transaction** : Respect des principes ACID ;
- **Haute disponibilité** : via la mise en place d'un cluster ;
- **Volumétrie** : Stockage et Requêtage de milliards de nœuds et de relations ;
- **Cypher**: langage de requête graphe déclaratif, simple et efficace ;
- **Schemaless** : pas de schéma préétabli.

Présentation de Neo4J (Nœud)



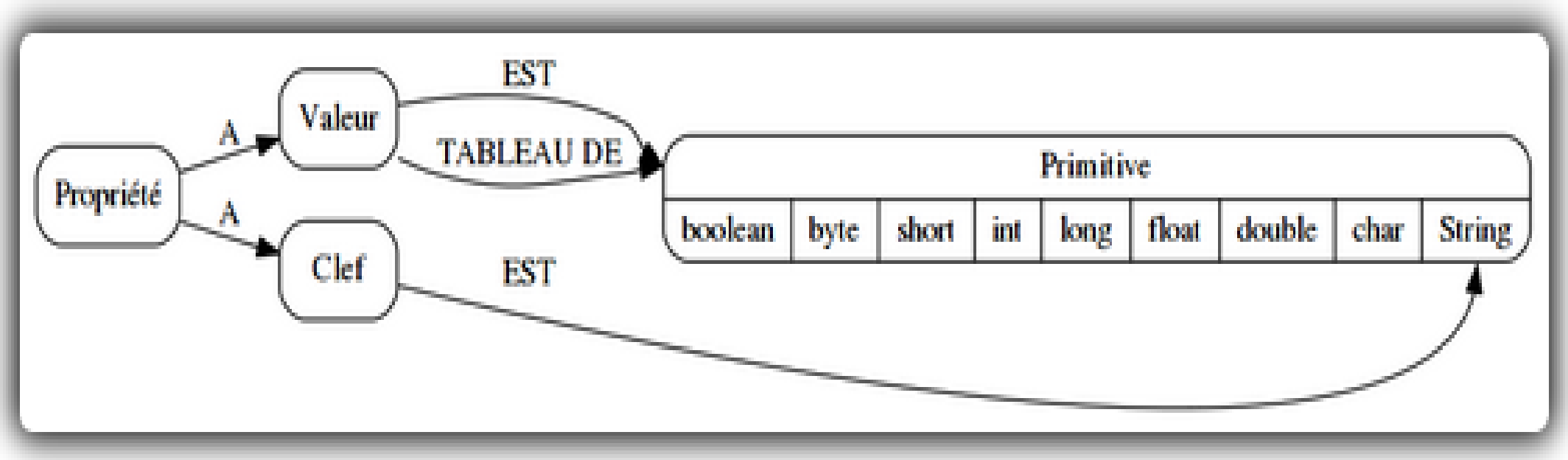
Présentation de Neo4J (Relation)





Présentation de Neo4J (Propriété)

Les types possibles des **propriétés** des **nœuds** et **relations** correspondent aux types primitifs de Java, ou à un tableau de type primitif.



Présentation de CQL

Cypher Query Langage (**CQL**) est un langage déclaratif permettant de requêter et mettre à jour le graphe.

Il est inspiré du SQL, on y retrouve beaucoup de concepts familiers, comme les clauses WHERE, ORDER BY, SKIP, LIMIT…

CQL fournit un moyen visuel pour représenter les nœuds et les relations d’un graphe.

Deux clauses principales en Cypher pour construire des requêtes

- CREATE pour créer une nouvelle entité
- MATCH pour chercher/récupérer des entités

Présentation de CQL

Cypher s'appuie sur le type de syntaxe **ascii-art ¯_(ツ)_/¯.**
L'**art ASCII** consiste à réaliser des images uniquement à l'aide des lettres et caractères spéciaux contenus dans le code ASCII

Nœud: Représenté avec des parenthèses : **()**
Relation: Représentée par : **[ ]**, **-** et **< ou >**

-[ ]-> <-[ ]-
Le sens de la flèche indique la direction de la relation

Exemple :
Affiche tous les nœuds en relation dans la base

```
MATCH (n)-[r]->(m) RETURN n,r,m;
```

Présentation de CQL

NOEUD

Syntaxe:

```
CREATE (n:label1:label2:.. {prop1:'value', prop2:'value'...})
[RETURN n];
```

Le nœud est créé par l'intermédiaire d'un alias (ici "**n**") qui n'est pas enregistré dans la base mais sert à désigner le nœud.

Exemple:

```
CREATE (charlie:Personne:Acteur {Prenom: 'Charlie', Nom:
'Sheen'}),
(oliver:Personne:Realisateur {Prenom: 'Oliver', Nom: 'Stone'});
```

Présentation de CQL

RELATION

Création des deux nœuds et de la relation entre eux
CREATE (**n**:label {...}) **-[r:label {...}]->** (**m**:label {...}) **[RETURN n,r,m];**
OU
Création d'une relation entre un nœud existant et un nœud créé
MATCH (a:label) where a.xxx="…"
CREATE (a) **-[r:label {...}]->** (**m**:label {...}) **[RETURN n,r,m];**

Exemple:
Créer une relation JoueDans entre Charlie Sheen et Wall Street

```
MATCH (a:Acteur) WHERE a.Nom="Sheen"
CREATE (a) -[:JoueDans {role: 'Bud Fox'}]->(wallStreet:Film {titre:
'WallStreet'});
```

Visualiser le graphe concernant les noeud liés à la relation créée

```
MATCH (n)-[r:JoueDans]->(m) RETURN n,r,m;
```

Présentation de CQL

RELATION

Exemple:

Créer une relation **Realise** entre Oliver Stone et le film Wall Street

```
MATCH (a:Realisateur), (b:Film)
WHERE a.Nom="Stone" AND b.titre ="WallStreet"
CREATE (a)-[r:Realise]-> (b);
```

Visualiser le graphe concernant les noeud liés à la relation créée

```
MATCH (n)-[r:Realise]->(m) RETURN n,r,m;
```

Visualiser le graphe concernant les noeud liés aux relations créées

```
MATCH (n)-[r]->(m) RETURN n,r,m;
```

OU

```
MATCH (n)-[r:Realise|JoueDans]->(m) RETURN n,r,m;
```

Présentation de CQL

Compter le nombre de nœuds du graphe

```
MATCH (n) RETURN count(*);
```

Compter le nombre de participants pour chaque relation du graphe

```
MATCH (n)-[r]->() RETURN type(r), count(*);
```

Affiche les nœuds de la base

```
MATCH (n) RETURN n;
```

Affiche les nœuds en relations de la base

```
MATCH (n)-[r]->(m) RETURN n,r,m;
```

Affiche les nœuds (avec ou sans) relations de la base

```
MATCH (n) OPTIONAL MATCH (n)-[r]-(m) RETURN n, r,m;
```

Supprimer tous les noeuds et relations créés

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

Présentation de CQL: Exercice

Un événement important vient de bouleverser le monde de **Game Of Thrones**.

Lors d'un combat épique, Oberyn Martell, dit The Viper et Gregor Clegane, dit The Mountain, se sont affrontés.
Gregor Clegane a tué Oberyn Martell dans un duel

Créer les deux nœuds représentant les personnages.
Créer la relation entre les personnages.

Présentation de CQL : Enchaînement

```
CREATE (a:Personne:Acteur {Prenom:'Anthony',Nom:'Hopkins'})-
[r:JoueDans {role: 'Hannibal Lecter'}]->(SilenceDesAgneaux:Film
{titre: 'Le Silence des agneaux'})<-[:Realise]-
(j:Personne:Realisateur {Prenom: 'Jonathan', Nom:'Demme'})
RETURN a,r,j;
```

Création des 2 nœuds et des relations associées

```
MATCH (n)-[r:Realise|JoueDans]->(m) RETURN n,r,m; => OU Inclusif
MATCH (n)-[r:!JoueDans]->(m) RETURN n,r,m; => Exclusion
```

Présentation de CQL : Exercice

Rechercher le nom du réalisateur du film Wall Street

Rechercher le nom des acteurs ayant joué dans le film Le Silence des agneaux