Présentation de NEO4J

Commandes :
FINISH, UNWIND

Fonctions d’agrégation:
AVG, COLLECT, COUNT, MAX, MIN, SUM,

Multiples clauses MATCH
Clause Exists / Not exists
Contraintes

FINISH

Exécute la commande mais sans afficher de résultat
(GQL conformance)
« GQL est le nouveau langage ISO d’interrogation standard pour les bases de type Graphe) » 13/09/2024

Exemple

```
CREATE (p:Person) FINISH;
```

Equivalent à

```
CREATE (p:Person);
```

UNWIND

Permet d'« éclater » une liste d'élément

Exemple

```
WITH [[1,2,3],[4,5,6],[7,8,9]] AS listOne
UNWIND listOne AS listOneElement
RETURN listOneElement;
```

listOneElement
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]

```
WITH [[1,2,3],[4,5,6],[7,8,9]] AS listOne
UNWIND listOne AS listOneElement
RETURN listOneElement =>
UNWIND listOneElement AS element
RETURN element;
```

element
1
2
3
4
5
6
7
8
9

Fonctions d'agrégation

COLLECT

Constitue une liste qui contient la liste des valeurs retournées

Soit des nœuds Personne avec 1 attribut AnneeDeNAissance

```
MATCH (p:Person) RETURN collect(p.born);
```

[1964, 1967, ..., 1963, 1943]

https://neo4j.com/docs/cypher-manual/current/subqueries/collect/#collect-example

```
MATCH (p:Person) RETURN avg(p.age);
MATCH (p:Person) RETURN min(p.age);
MATCH (p:Person) RETURN max(p.age);
MATCH (p:Person) RETURN sum(p.age);
MATCH (n:Movie)  RETURN count(*)
```

Multiple clauses MATCH

Exemple:

```
MATCH (a:Person)-[:ACTED_IN]->(m:Movie)<-[:DIRECTED]-(d:Person)
WHERE m.released  >2010
RETURN a.name as acteur, m.title as film, d.name as réalisateur;
```

OU

```
MATCH (a:Person)-[:ACTED_IN]->(m:Movie)
WHERE m.released  >2010
MATCH (m)<-[:DIRECTED]-(d:Person)
RETURN a.name, m.title, d.name;
```

=> La 1ere syntaxe est plus performante

```
MATCH (n:Person)-[:DIRECTED&ACTED_IN]-> (f:Movie) RETURN n.name;
```

N'est pas correct car une relation n'a qu'un seul type

```
MATCH (n:Person)-[:!DIRECTED&!ACTED_IN]-> (f:Movie) RETURN n.name;
```

est correct

Clauses Exists et Not exists

```
MATCH (p:Person)
WHERE EXISTS { (p)-[:REVIEWED] ->(m:Movie) }
RETURN p.name AS name;

MATCH (p:Person)
WHERE NOT EXISTS {(p)-[:ACTED_IN|DIRECTED|REVIEWED]->(m:Movie) }
ORDER BY p.name RETURN p.name;
```

Contrainte d'existence

Création d'une contrainte d'existence pour un attribut d'un noeud

Exemple:

```
CREATE CONSTRAINT Movie_released_exists IF NOT EXISTS FOR (m:Movie)
REQUIRE m.released IS NOT NULL;
```

Remarques

- l'attribut doit exister pour l'ensemble des nœuds concernés
- La suppression de l'attribut avec la contrainte n'est plus possible

```
CREATE (m:Movie {title: "Au Revoir là-haut"}) RETURN m; KO
CREATE (m:Movie {title: "Au Revoir là-haut", released:2017}) RETURN m; OK
MATCH  (m:Movie {title: "Au Revoir là-haut"})
REMOVE a.released RETURN a ; KO
```

Contrainte d’existence

Création d’une contrainte d’existence pour un attribut d’une relation

Exemple:

```
CREATE CONSTRAINT REVIEWED_rating_exists IF NOT EXISTS
FOR ()-[r:REVIEWED]->()
REQUIRE r.rating IS NOT NULL;

CREATE (n:Person {name:'Jules'})-[r:REVIEWED]->(f:Movie {title:'The
Polar Express'}) RETURN n,r,f;
```

=> KO

Relationship(nnn) with type `REVIEWED` must have the property `**rating**`

Contrainte d'unicité

Création d'une contrainte d'unicité pour un attribut d'un noeud

Exemple:

```
CREATE CONSTRAINT Person_name_uk IF NOT EXISTS
FOR (p:Person)
REQUIRE p.name IS UNIQUE;
```

Remarques

- La contrainte peut porter sur plusieurs attributs du nœud
- La valeur de l'attribut peut être NULL

```
CREATE CONSTRAINT Movie_released_title_UK IF NOT EXISTS
FOR (m:Movie)
REQUIRE (m.released, m.title) IS UNIQUE;

CREATE (m:Movie {title: "Le Comte de Monte-Cristo"}) RETURN m; OK
```

Contrainte d’unicité pour un nœud

Création d’une contrainte clef pour un Noeud
Elle combine les contraintes d’existence et d’unicité

Exemple:

```
CREATE CONSTRAINT Person_nodekey IF NOT EXISTS
FOR (x:Person)
REQUIRE (x.name, x.born) IS NODE KEY
```

Remarque:
Elle combine les contraintes d’existence **et** d’unicité

Gestion des contraintes

Lister les contraintes

```
show constraints
```

Supprimer une contrainte

```
drop constraint < constraint_name >
```