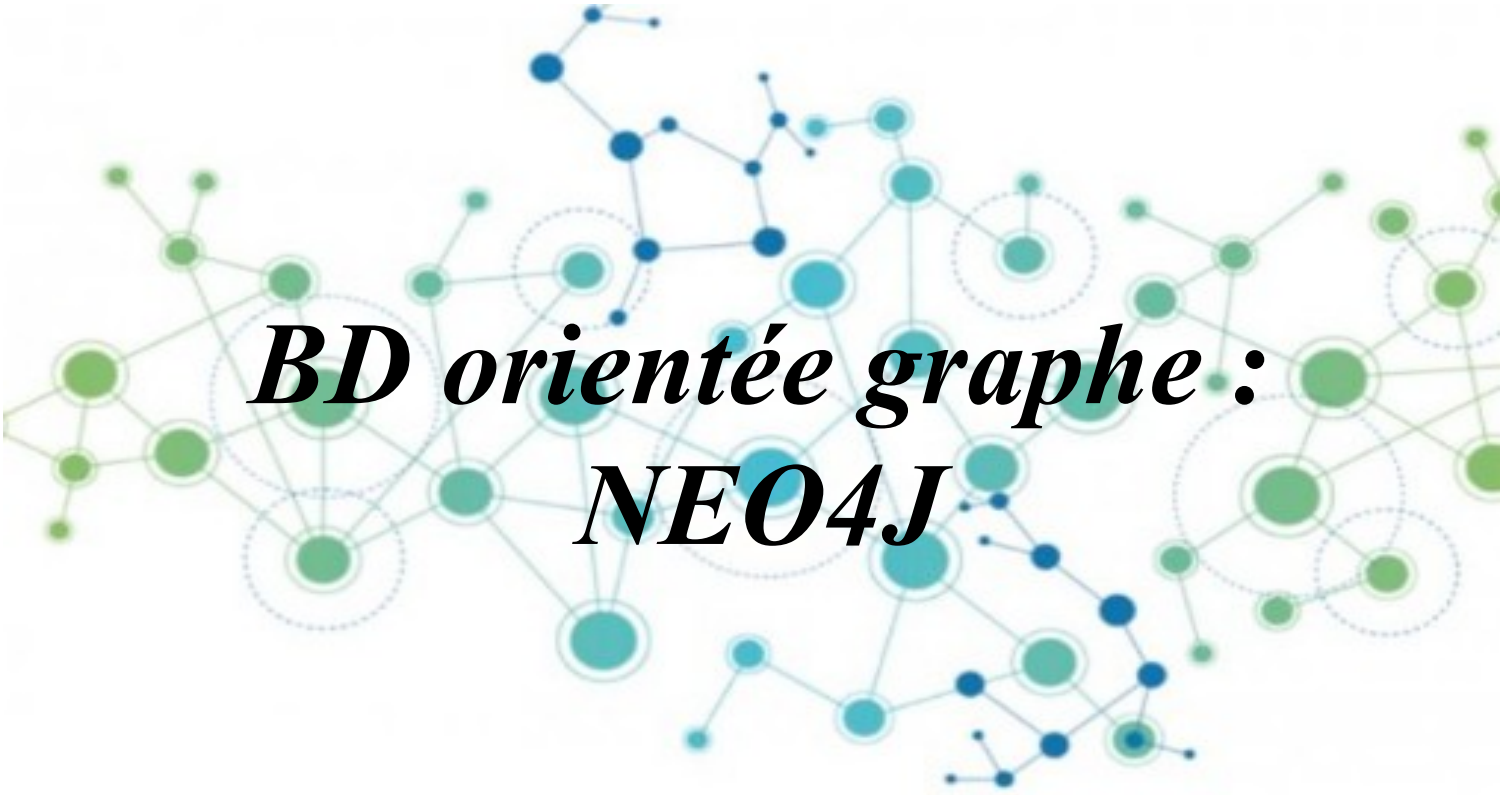


BD orientée graphe : NEO4J

DSEN 2 & GLSI 2

Tasnim Abar: tasnim.abar@supcom.tn

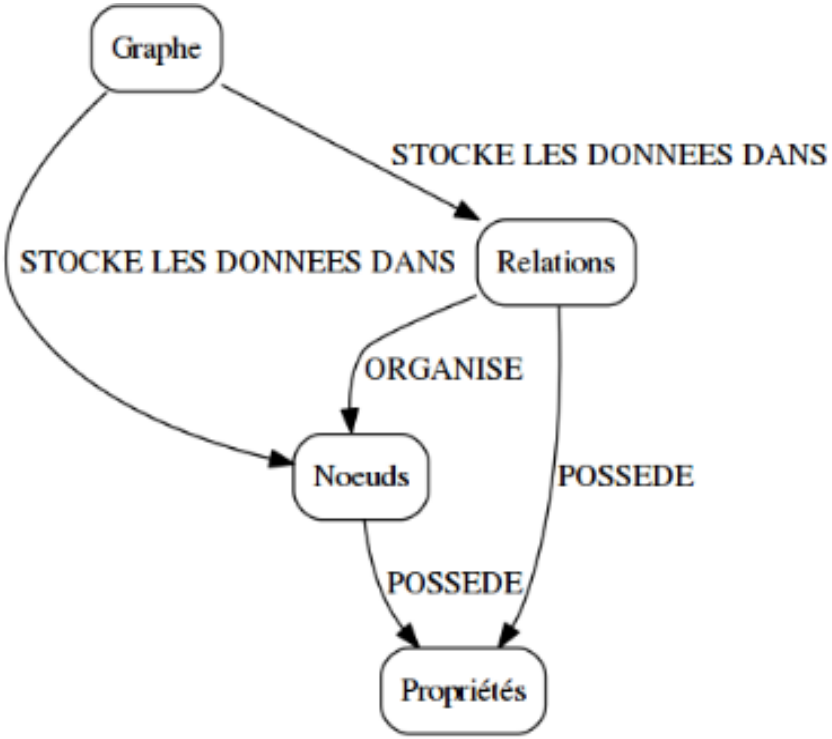
Introduction

- Neo4j est une base de données NoSQL orientée Graph permettant de passage à l'échelle sur de gros graphes.

- L'un des avantages de Neo4j est sa simplicité d'usage et son langage d'interrogation **Cypher**.

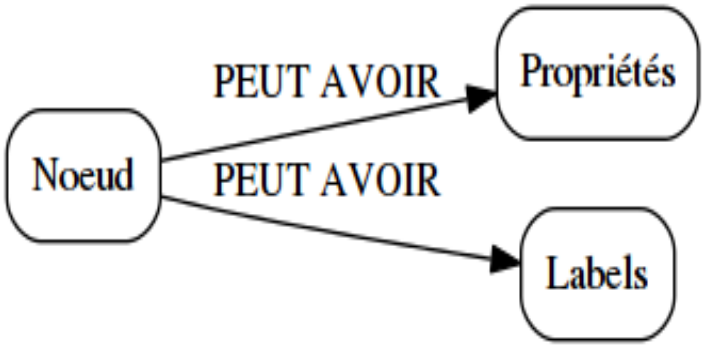
Concept

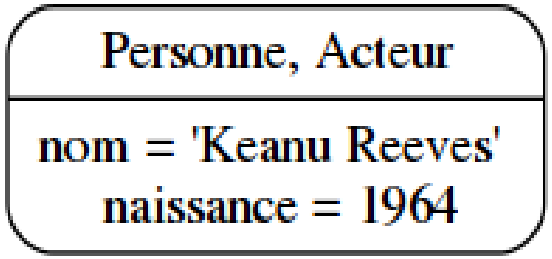
- Les bases de données orientées graphes tournent autour de trois concepts : les nœuds, les relations et leurs propriétés.



Concept: Les nœuds

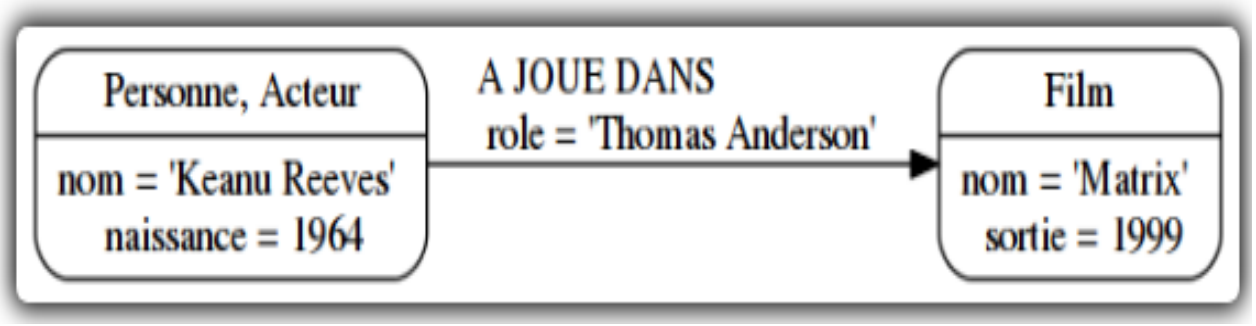
- L'unité fondamentale qui forme un graphe est le nœud.
- Les nœuds sont des enregistrements composés de propriétés de type clef/valeur, sans schéma préétabli. Généralement, ils représentent une entité du modèle.
- Pour différencier les nœuds, Neo4j apporte la notion de **label**. Ceux-ci permettent de donner un rôle ou un type à un nœud (un nœud peut avoir plusieurs labels).

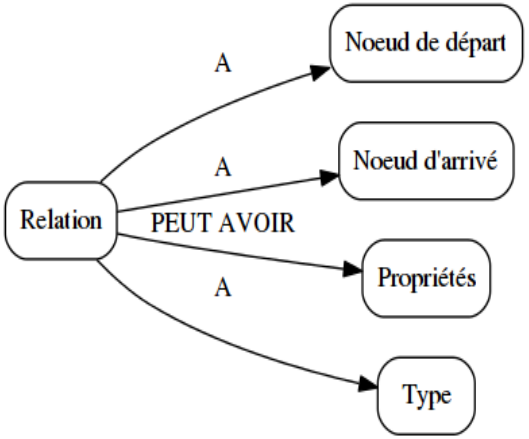




Concept: Les relations

- Les relations entre les nœuds sont la clef de voûte des graphes, c'est ce qui permet de lier des données et de créer des structures comme des listes, des arbres, des maps…
- Neo4j les définit comme étant constituées d'un nœud de départ, d'arrivée (donc une relation avec un sens) et d'un type. De plus, tout comme les nœuds, elles sont également un enregistrement, et donc peuvent avoir des propriétés de type clef/valeur.





Cypher

- Cypher est un langage de requête de graphe déclaratif qui permet une interrogation et une mise à jour expressives et efficaces du graphe.
- Cypher est conçu pour être simple, mais puissant; les requêtes de base de données très compliquées peuvent être facilement exprimées.
- Il s'inspire de différentes approches et s'appuie sur les pratiques établies en matière d'interrogation expressive. De nombreux mots-clés, tels que WHERE et ORDER BY, sont inspirés de SQL

Cypher

- Cypher emprunte sa structure à SQL - les requêtes sont construites à l'aide de différentes clauses.
- Voici quelques clauses utilisées pour lire le graphique:

➢ MATCH: C’est le moyen le plus courant d’obtenir des données du graphique.
➢ WHERE: contrainte bien précise.
➢ RETURN: on retourne quoi comme résultat .

- Et voici des exemples de clauses utilisées pour mettre à jour le graphique:

➢ CREATE(et DELETE): Créer (et supprimer) des nœuds et des relations.
➢ SET(et REMOVE): définir des valeurs sur des propriétés et ajoutez des étiquettes sur les nœuds à l'aide de SETet utilisez REMOVE pour les supprimer.
➢ MERGE: Correspond à des nœuds et modèles existants ou en crée de nouveaux. Ceci est particulièrement utile avec des contraintes uniques.

Cypher

- Les nœuds sont représentés avec des parenthèses, ce qui ressemble à des cercles : ()
- Si vous avez besoin d'identifier le nœud dans votre requête (dans une clause WHERE par exemple), il suffit de lui donner un nom : (monNoeud)

- Pour spécifier un label, il suffit de l'ajouter comme ceci : (monNoeud:monLabel)

- Voici quelques exemples :

➢ () : n'importe quel nœud ;
➢ (:Personne) : un nœud avec le label *Personne* ;
➢ (n:Personne) : un nœud identifié dans la variable *n* avec le label *Personne* ;
➢ (n:Personne:Acteur) : un nœud identifié dans la variable *n* avec le label *Personne* et *Acteur*.

Cypher

- Les relations sont représentées par deux tirets avec un '>', ce qui ressemble à une flèche : -->
- Si vous avez besoin d'identifier la relation dans votre requête, vous pouvez lui donner un nom comme ceci : -[maRelation]->

(moi) -[:AMI]-> (mesAmis) -[:AMI]-> (amisDeMesAmis)

- Pour spécifier le type de la relation, il suffit de l'ajouter comme ceci : -[maRelation:MON_TYPE]->

- Voici quelques exemples :

➢ (a)--(b) : n'importe quelle relation entre le nœud *a* et *b* (peu importe la direction) ;
➢ (a)-[:AMI]->(b) : relation de type *AMI* depuis le nœud *a* vers le nœud *b* ;
➢ (a)-[r:AMI|CONNAIT]->(b) : relation identifiée dans la variable *r* de type *AMI* ou *CONNAIT* depuis le nœud *a* vers le nœud *b*.

Les Clauses principales

- MATCH: est utilisée pour rechercher le modèle décrit dans celle-ci.
- Exemple 1: obtenir tous les nœuds:

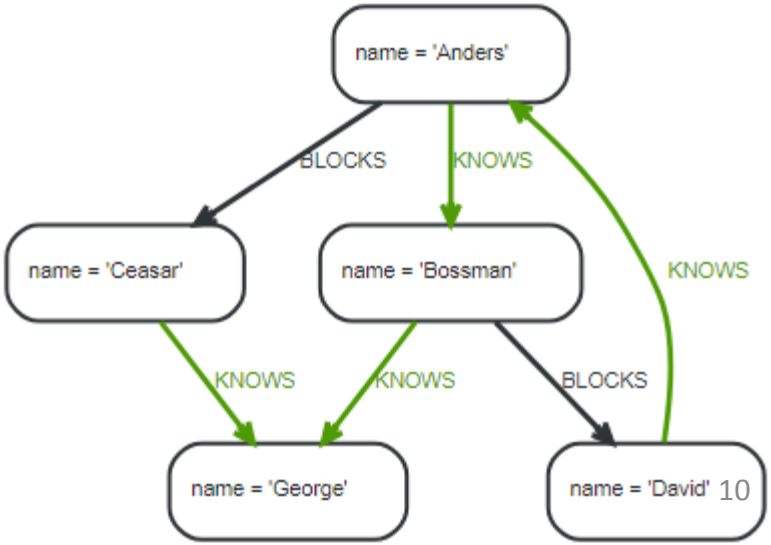
```
MATCH (n)
RETURN n
```

- Exemple 2: obtenir tous les films de la base

```
MATCH (movie:Movie)
RETURN movie.title
```

- WITH: manipuler la sortie avant de la transmettre.

```
MATCH (david { name: 'David' })--(otherPerson)-->()
WITH otherPerson, count(*) AS foaf
WHERE foaf > 1
RETURN otherPerson.name
```



Les Clauses principales

- ORDER BY : trier les résultats avant la transmission.

```
MATCH (n)
RETURN n.name, n.age
ORDER BY n.age, n.name
```

```
MATCH (n)
RETURN n.name, n.age
ORDER BY n.name DESC
```

On ajoutant « DESC » après ORDER BY, le tri se fera dans l'ordre inverse.

- SKIP définit à partir de quelle ligne commencer, en incluant les lignes dans la sortie.

```
MATCH (n)
RETURN n.name
ORDER BY n.name
SKIP 3
```

Les trois premiers nœuds sont ignorés

LIMIT: limite le nombre de lignes dans la sortie.

```
MATCH (n)
RETURN n.name
ORDER BY n.name
LIMIT 3
```

Les Clauses principales

- UNWIND: développe une liste en une séquence de lignes.
- WHERE: ajoute des contraintes aux modèles. On peut utiliser les opérateurs booléens AND. OR. XOR et NOT

```
MATCH (n)
WHERE n.name = 'Peter' XOR (n.age < 30 AND n.name = 'Timothy') OR NOT (n.name = 'Timothy' OR n.name = 'Peter')
RETURN n.name, n.age
```

- La CREATE: est utilisée pour créer des nœuds et des relations:
- CREATE (n)
- CREATE (n),(m): La création de plusieurs nœuds se fait en les séparant par une virgule.
- CREATE (n:Person): ajouter un label lors de la création d'un nœud
- CREATE (n:Person:Swedish): ajouter 2 labels
- CREATE (n:Person { name: 'Andy', title: 'Developer' }): Lors de la création d'un nouveau nœud avec des labels, vous pouvez ajouter des propriétés en même temps.

Les Clauses principales

- Pour créer une relation entre deux nœuds, on obtient d'abord les deux nœuds. Une fois les nœuds chargés, on crée simplement une relation entre eux:

```
MATCH (a:Person),(b:Person)
WHERE a.name = 'A' AND b.name = 'B'
CREATE (a)-[r:RELTYPE]->(b)
RETURN type(r)
```

type (r)
"RELTYPE"
1 rangée Relations créées: 1

- La définition des propriétés sur les relations s'effectue de la même manière que lors de la création de nœuds:

```
MATCH (a:Person),(b:Person)
WHERE a.name = 'A' AND b.name = 'B'
CREATE (a)-[r:RELTYPE { name: a.name + '<->' + b.name }]->(b)
RETURN type(r), r.name
```

type (r)	r.nom
"RELTYPE"	"A<->B"
1 ligne Relations créées: 1 Ensemble de propriétés: 1	

Les Clauses principales

- DELETE: est utilisée pour supprimer des nœuds, des relations ou des chemins:
- MATCH (n:Person { name: 'UNKNOWN' }) DELETE n
- MATCH (n) DETACH DELETE n : supprimer un noeud avec tous ses relations.
- SET: est utilisée pour mettre à jour les labels sur les nœuds et les propriétés sur les nœuds et les relations.

```
MATCH (n { name: 'Andy' })
SET n.surname = 'Taylor'
RETURN n.name, n.surname
```

- REMOVE: est utilisée pour supprimer les propriétés des nœuds et des relations et pour supprimer les labels des nœuds.

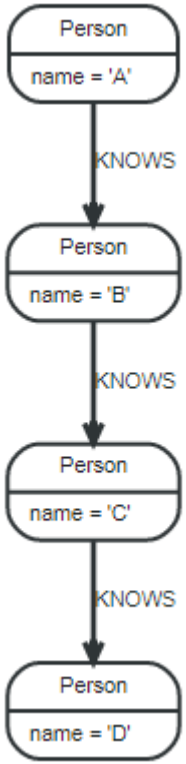
```
MATCH (a { name: 'Andy' })
REMOVE a.age
RETURN a.name, a.age
```

Les Clauses principales

- FOREACH: est utilisée pour mettre à jour les données d'une liste, qu'il s'agisse des composants d'un chemin ou du résultat de l'agrégation.

```
MATCH p =(begin)-[*]->(END )
WHERE begin.name = 'A' AND END .name = 'D'
FOREACH (n IN nodes(p)| SET n.marked = TRUE )
```

Cette requête définira la propriété marked sur true sur tous les nœuds d'un chemin.



Les Clauses principales

- LOAD CSV est utilisé pour importer des données à partir de fichiers CSV:

artistes.csv.

```
1,ABBA,1992
2,Roxette,1986
3,Europe,1979
4,The Cardigans,1992
```

Requete.

```
LOAD CSV FROM 'https://neo4j.com/docs/cypher-manual/3.5/csv/artists.csv' AS line
CREATE (:Artist { name: line[1], year: toInteger(line[2])})
```

- "WITH HEADERS" : va permettre de récupérer la première ligne du CSV comme des identifiants de colonne
- Enfin, la fonction "CREATE" va ici permettre de créer des nœuds dans graphe avec comme paramètre les la première colonne du CSV (données pointées par "csvLine.CSVLigne1Colonne1)

artistes-with-headers.csv.

```
Id,Name,Year
1,ABBA,1992
2,Roxette,1986
3,Europe,1979
4,The Cardigans,1992
```

Requete.

```
LOAD CSV WITH HEADERS FROM 'https://neo4j.com/docs/cypher-manual/3.5/csv/artists-with-headers.csv' AS line
CREATE (:Artist { name: line.Name, year: toInteger(line.Year)})
```

Les Clauses principales

- Pour importer de grandes quantités de données, on peut utiliser le USING PERIODIC COMMIT pour indiquer à Neo4j d'effectuer une validation après un certain nombre de lignes. Cela réduit la surcharge de mémoire liée à l'état de la transaction. Par défaut, la validation aura lieu toutes les 1000 lignes.

```
USING PERIODIC COMMIT
LOAD CSV FROM 'https://neo4j.com/docs/cypher-manual/3.5/csv/artists.csv' AS line
CREATE (:Artist { name: line[1], year: toInteger(line[2])})
```

Exemples

```
1- MATCH (ville)-[:A_COTE]->(b)
2- WHERE ville.nom = 'Montréal'
3- RETURN DISTINCT b.nom
```

1- Trouve les nœuds qui ont un lien A_COTE vers un autre nœud
2- Pour ces nœuds, ne retenir que ceux dont la ville est Montréal
3- Retourne au programme le nom de ces villes

(ville) et (b) sont entre parenthèses, ce qui signifie qu'ils sont des nœuds
[:A_COTE] est entre crochets, ce qui signifie qu'il est une relation de type A_COTE
Dans la section MATCH, la flèche est de (ville) vers (b).
DISTINCT permet de supprimer les doublons

Les fonctions Cypher

- Les fonctions prédicats: *Les prédicats sont des fonctions booléennes qui renvoient true ou false pour un ensemble d'entrées non nulles. Ils sont le plus souvent utilisés pour filtrer les sous-graphes dans la partie* WHERE *d'une requête.*

fonction	Description	Exemple
all()	Teste si le prédicat est valable pour tous les éléments d'une liste.	MATCH p =(a)-[*1..3]->(b) WHERE a.name = 'Alice' AND b.name = 'Daniel' AND ALL (x IN nodes(p) WHERE x.age > 30) RETURN p
any()	Teste si le prédicat est valable pour au moins un élément d'une liste.	MATCH (a) WHERE a.name = 'Eskil' AND ANY (x IN a.array WHERE x = 'one') RETURN a.name, a.array
exists()	Renvoie true s'il existe une correspondance pour le modèle dans le graphique ou si la propriété spécifiée existe dans le nœud, la relation ou la mappe.	MATCH (n) WHERE exists(n.name) RETURN n.name AS name, exists((n)-[:MARRIED]->()) AS is_married

Les fonctions Cypher

- Les fonctions scalaires:

fonction	Description	Exemple
Head()	Retourne le premier élément de la liste.	MATCH (a) WHERE a.name = 'Eskil' RETURN a.array, head(a.array)
Type()	Renvoie la représentation sous forme de chaîne du type de relation.	MATCH (n)-[r]->() WHERE n.name = 'Alice' RETURN type(r)
Id()	Retourne l'id d'une relation ou d'un nœud.	MATCH (a) RETURN id(a)
Length()	Retourne la longueur d'un chemin.	MATCH p =(a)-->(b)-->(c) WHERE a.name = 'Alice' RETURN length(p)
timestamp()	Renvoie la différence, exprimée en millisecondes, entre l'heure actuelle et minuit, e 1er janvier 1970 UTC	RETURN timestamp()

Les fonctions Cypher

- Les fonctions d'agrégation

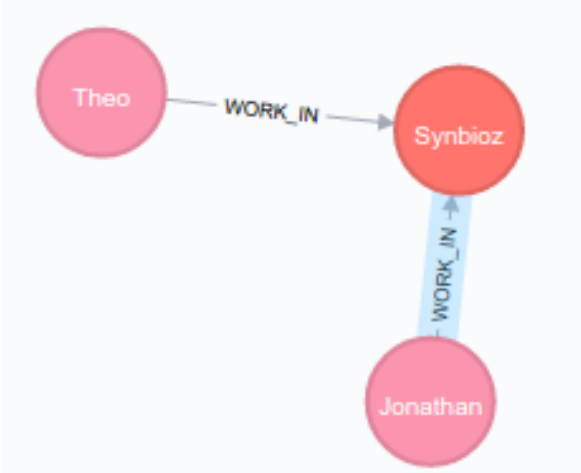
fonction	Description	Exemple
avg()	Renvoie la moyenne d'un ensemble de valeurs numériques..	MATCH (n:Person) RETURN avg(n.age)
min()	Renvoie la valeur minimale dans un ensemble de valeurs.	RETURN min(val)
max()	Renvoie la valeur maximale dans un ensemble de valeurs.	RETURN max(val)
sum()	Retourne la somme d'un ensemble de valeurs numériques.	MATCH (n:Person) RETURN sum(n.age)

Exemples

- MATCH (actor:Person)-[:ACTED_IN]->(movie:Movie)
- WHERE movie.title STARTS WITH "T"
- RETURN movie.title AS title, collect(actor.name) AS cast ORDER BY title ASC LIMIT 10;

- Les labels sont : person, Movie.
- Les nœuds sont : actor, movie.
- La relation: ACTED_IN
- La condition: le titre de film doit commencer par T.
- Le résultat : - le titre de film
  - la liste des acteurs pour le film triée par ordre croissant
  - on se limite à 10 sorties.

- **CREATE** (Theo:Staff {title:'Theo', joined:2014, job: 'Developer', city: 'Nantes'})
- **CREATE** (Jon:Staff {title:'Jonathan', joined:2013, job: 'Project Manager', city: 'Lille'})
- **CREATE** (Synbioz:Company {name:'Synbioz', born:2007})
- **MATCH** (n:Staff) **RETURN** n **LIMIT** 25
- **MATCH** (n:Staff) **MATCH** (m:Company {name: 'Synbioz'}) **CREATE** (n)-[:WORK_IN]->(m)



Références

- https://neo4j.com/docs/cypher-manual/current/clauses/
- https://www.irit.fr/~Thierry.Millan/MemoiresENG221/Nicolas_vergnes.pdf
- https://www.infoq.com/fr/articles/graph-nosql-neo4j/