Analyse de Graphes de Données
Les bases de données Graphes l'exemple de Neo4j

Hamida SEBA LAGRAA Hamida.seba@univ-lyon1.fr

Equipe Graphes Algorithmes et Applications GOAL, LIRIS

Ce cours a été réalisé à partir du site de Neo4j et des ressources qui y sont déposées ainsi que du livre Graph databases

Toujours à la recherche de stage?

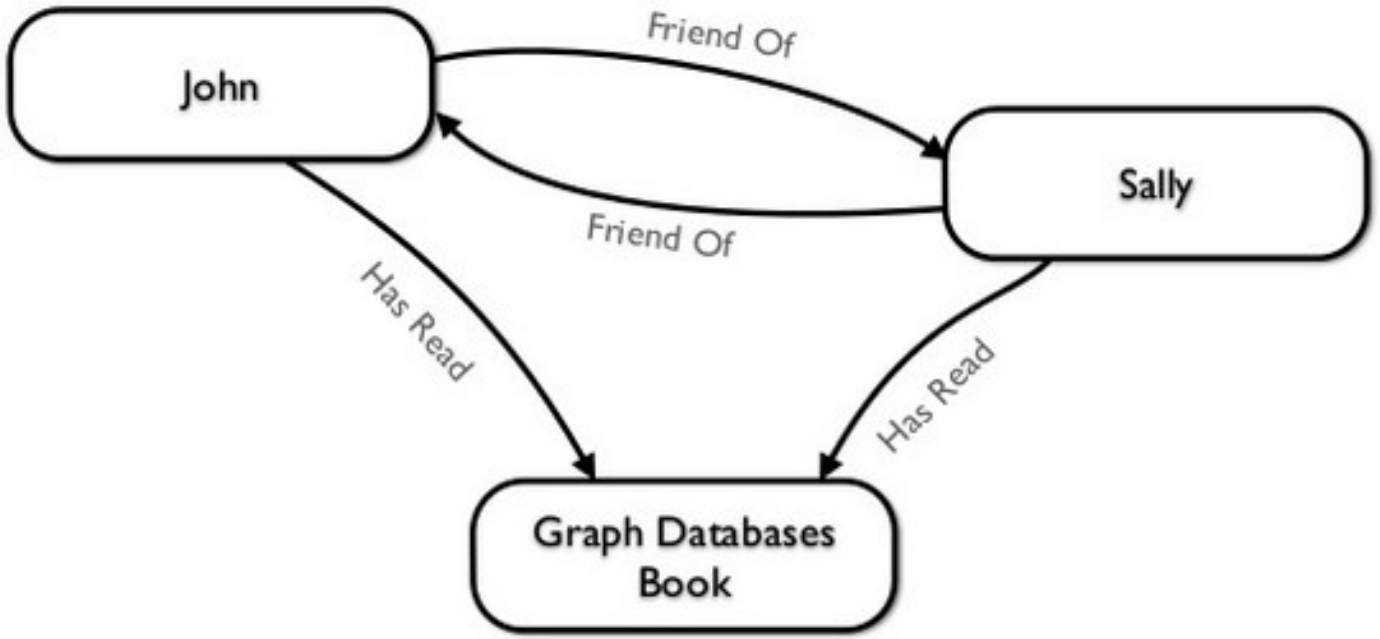
Neo4j

- Neo4j est un SGBD NoSQL orienté Graphes.
  - SGBD: Systèmes de Gestion de Bases de Données
  - NoSQL: Not Only SQL
  - SQL: langage d'interrogation de données dans les bases de données relationnelles (à bases de tables)
- Principe:
  - Dans ce monde hyper-connecté, aucune information n'est isolée
  - Seule un SGBD à base de graphes peut stocker et faire des requêtes efficacement sur des données connectées entre elles.
- Applications: domaines liés
  - Données social (réseaux sociaux)
  - Recommandations
  - Données géo-spatiales
  - Etc.

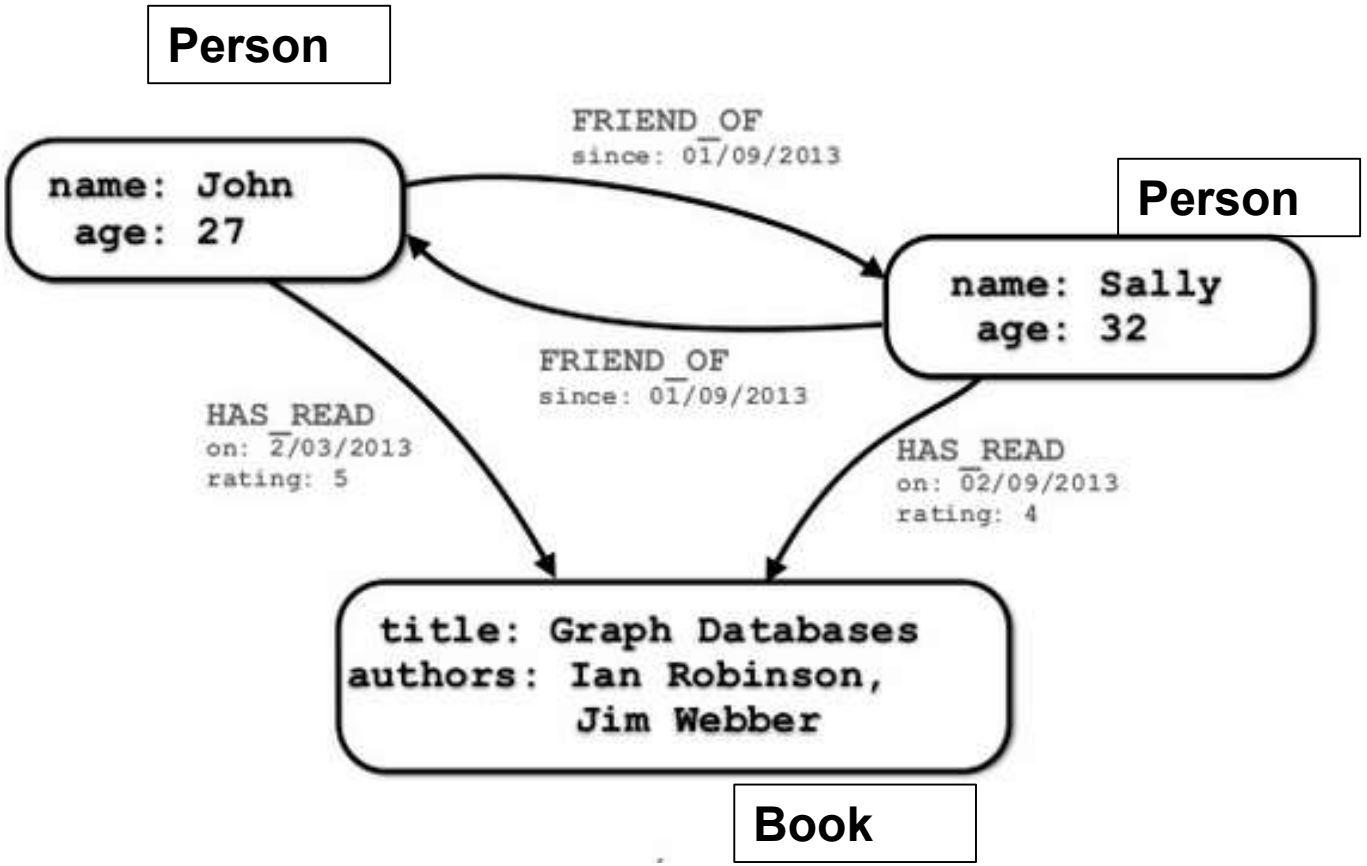
Le modèle de données “Graphe de propriétés” (Property Graph model)

- 4 briques de base:
  - **Nœud**: **entité**
    - ✓ Les propriétés d’une entité sont représentées par des couples (key, value)
    - ✓ En nombre quelconque
  - **Arc**: **association entre entités**
    - ✓ Une association possède une direction, un type, un nœud de départ et nœud d’arrivée
    - ✓ Possibilité d’avoir plusieurs associations entre deux nœuds
    - ✓ Association orientée mais navigable dans les deux directions
  - Propriétés
    - ✓ Attributs des entités, qualité des associations, métadonnées
  - Labels (étiquettes): (**type d’entité ou Classe**)
    - ✓ Groupement d’entités par rôle: user, entreprise, etc.,
    - ✓ Un nœud peut avoir plusieurs label ou aucun
    - ✓ Permet l’association d’index et de contraintes à des groupes de nœuds
- Le schéma de la base est également représenté avec un graphe (nœud=Entité/classe, arc=Association): schéma conceptuel/logique

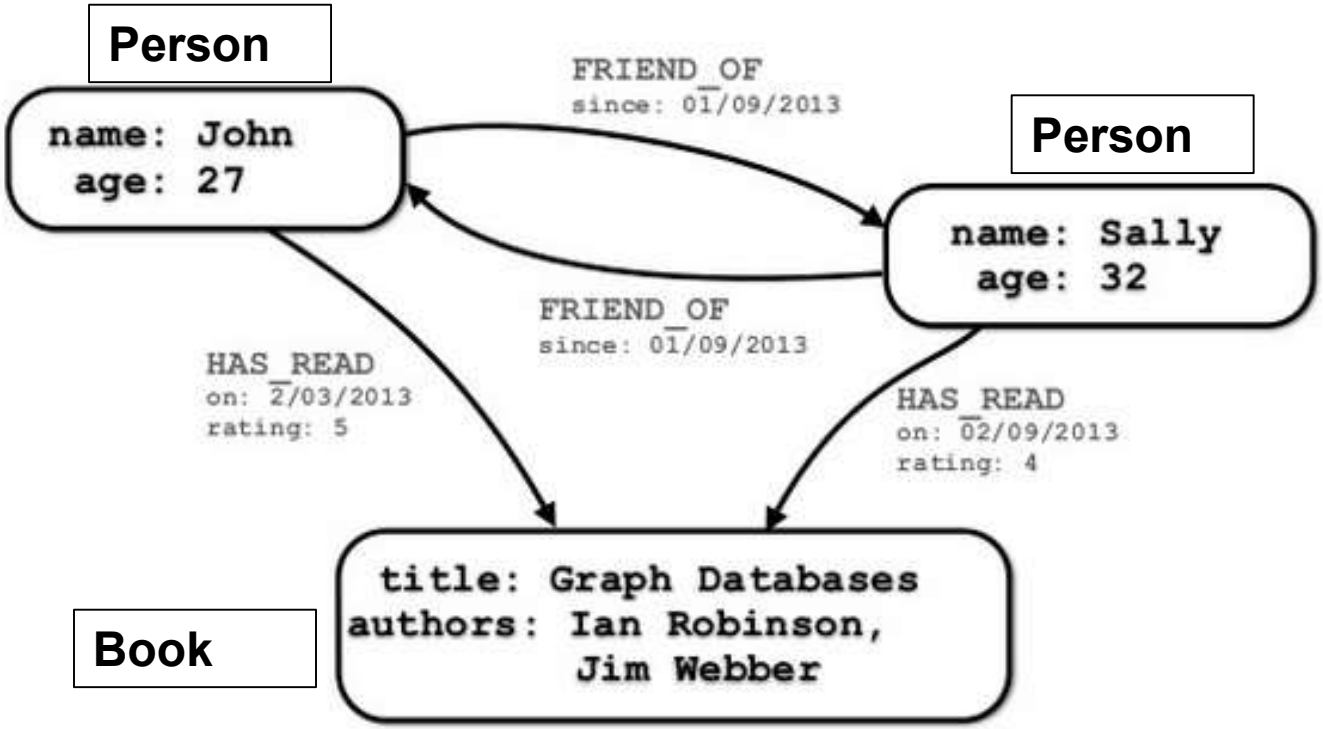
Exemple

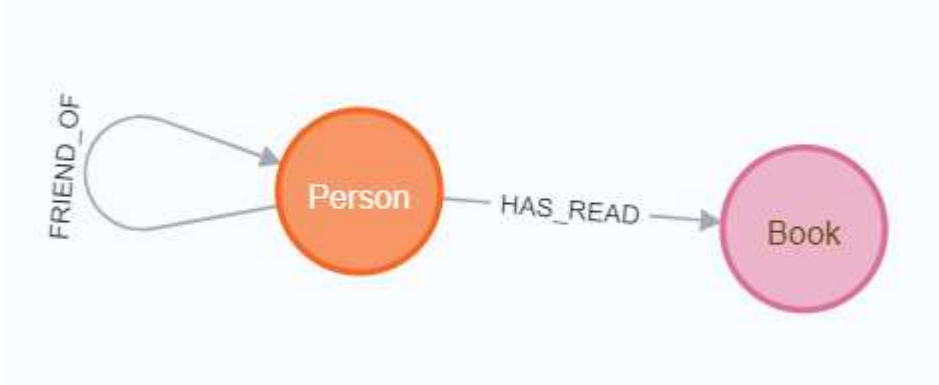

John
Friend Of
Sally
Friend Of
Has Read
Has Read
Graph Databases
Book

Exemple avec les propriétés et les labels



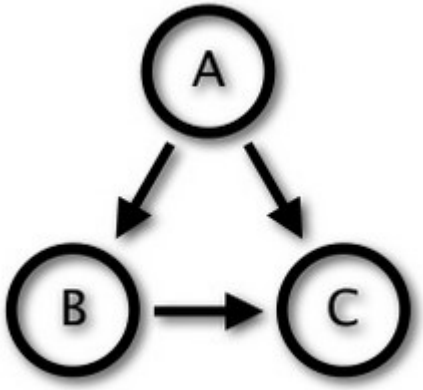
Exemple

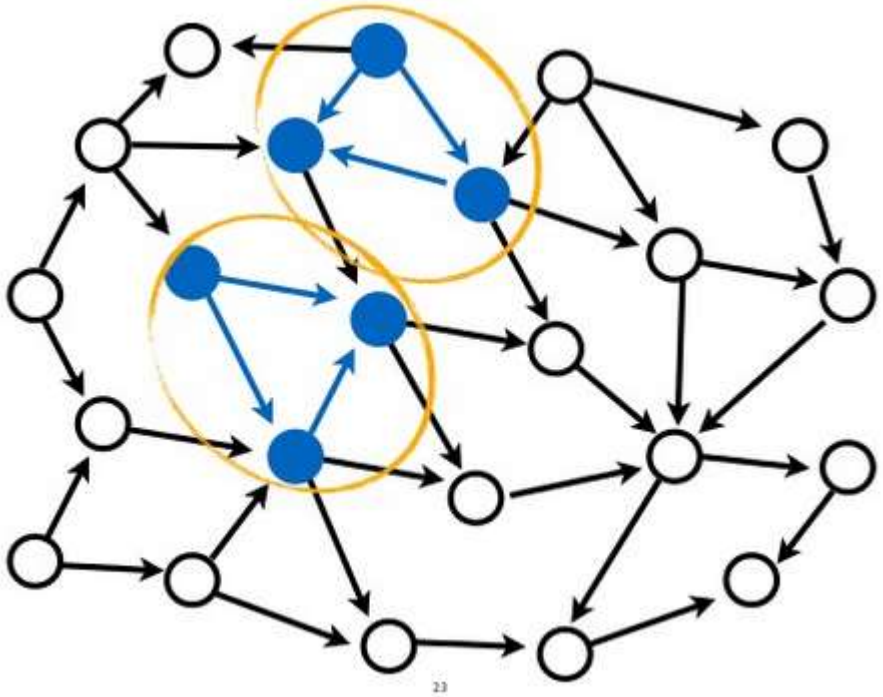

Person
name: John
age: 27
FRIEND_OF
since: 01/09/2013
Person
name: Sally
age: 32
FRIEND_OF
since: 01/09/2013
HAS_READ
on: 2/03/2013
rating: 5
HAS_READ
on: 02/09/2013
rating: 4
Book
title: Graph Databases
authors: Ian Robinson,
Jim Webber


FRIEND_OF
Person
HAS_READ
Book

Le langage Cypher

- Cypher
  - Langage déclaratif, inspiré du SQL
  - Décrire des motifs dans les graphes
- Orienté motif (Pattern)





Le langage Cypher: expression des motifs

- Motif de nœuds: `(a)`
  - Simple
  - Avec label `(a:User)`
  - Avec propriétés `(a {name: 'Andres', sport: 'Brazilian Ju-Jitsu'})`
- Motif pour les nœuds avec associations:
  - Le lien entre les nœuds peut être orienté ou pas (les nœuds peuvent être nommés ou pas)

`(a)-->(b)` `(a)-->()<--(c)` `(a)-->(b)<--(c)` `(a)--(b)`

- Motif pour les associations

`(a)-[r]->(b)` `(a)-[r:REL_TYPE]->(b)`

- Motif pour des chemins
  - Longueur fixe ou quelconque

`(a)-[*2]->(b)` `(a)-[*]->(b)` `(a)-[*3..5]->(b)`

Le langage Cypher : requêtes

- Format général d'une requête sur les nœuds:

```
MATCH (node)
RETURN node.property

MATCH (node1{property:value})- -(node2)   /* les - -  signifie en relation avec */
RETURN node2.propertyA

MATCH (node:Label)
WHERE node.property = "value“
RETURN node
```

Exemple:

Les noms de toutes les personnes

```
MATCH (n:Person)
Return n.name
```

Tous les livres lu par john

```
MATCH (n:Person { name:'John' })--(m:Book)
RETURN m.title
```

Des clauses WHERE, ORDER BY ou LIMIT peuvent également être rajoutées

Le langage Cypher : requêtes

- Format général d'une requête sur les associations:

```
MATCH (node1)-[r:REL_TYPE]->(node2)
RETURN r.property
```

```
MATCH (node1:Label1)-[r:REL_TYPE]->(node2:Label2)
RETURN node1, node2, r
```

Exemple:

Les notes attribuées au livre Graph Databases

```
Match (n:Person)-[r:HAS_READ]->(m:BOOK)
Where m.title='Graph Databababses'
return r.score
```

Le langage Cypher : requêtes

- Distinct : supprime les doublons

```
MATCH (p:Person)
return distinct p.name
```

- Mise à jour: commande SET
  - Mettre à jour une propriété ou créer une nouvelle propriété

```
MATCH p = (a:Person)
WHERE a.name = 'John'
SET a.name='Alfred'
```

  - Remplacer toutes les propriétés d'un nœud ou une relation

```
MATCH (p:Person { name: 'Sally' })
SET p = { name: 'Ellen', livesIn: 'London' }
/* l'attribut âge n'existe plus*/
```

  - remplacer qcq propriétés d'un nœud ou une relation

```
MATCH (p:Person { name: 'Sally' })
SET p+= { name: 'Ellen', livesIn: 'London' }
/*changement du nom, ajout d'un attribut et on garde l'age */
```

Le langage Cypher: création de nœuds/associations

- Créer des nœuds:

```
CREATE (node [:label]*[{cle:valeur[,cle:vakeur]*}])

Exemple
CREATE (John:Person { name : 'John', age : '27' })
CREATE (Sally:Person { name : 'Sally', age : '32' })
CREATE (Graphbook:Book {title:'Graph databases', authors:'Ian Robinson'})
```

Chaque nœud créé possède un id géré par le SGBD et que l'on peut accéder avec la fonction ID()
match (n) return ID(n)

- Créer des relations:
  - Pour créer une relation, il faut d'abord charger les deux nœuds concernés

```
MATCH (a:Person),(b:Person)
WHERE a.name = 'Node A' AND b.name = 'Node B'
CREATE (a)-[r:RELTYPE]->(b)RETURN r
Exemple
MATCH (a:Person),(b:Book)
WHERE a.name = 'John' AND b.title = 'Graph databases'
CREATE (a)-[r:HAS_READ{on:date('2013-09-02'), rating:4}]->(b)
RETURN r
MATCH (a:Person),(b:Book)
WHERE a.name = 'Sally' AND b.title = 'Graph databases'
CREATE (a)-[r:HAS_READ{on:date('2013-03-02'), rating:5}]->(b)
RETURN r
```

Pour plus d'info http://neo4j.com/docs/stable/query-create.html

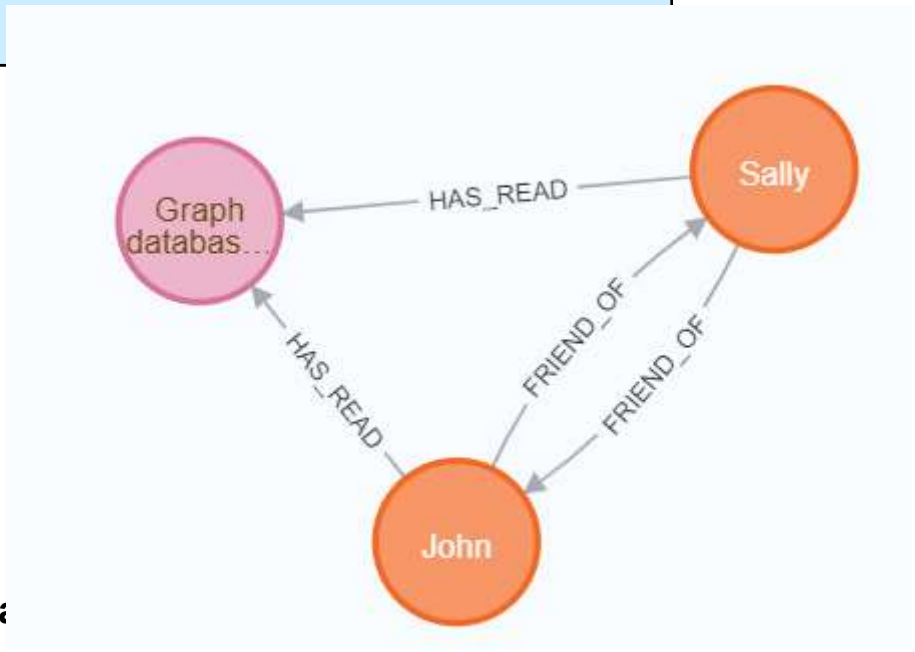
Le langage Cypher : création de nœuds/associations

- Créer des relations:
  - Pour créer une relation, il faut d'abord charger les deux nœuds concernés

Exemple(suite)

```
MATCH (a:Person),(b:Person)
WHERE a.name = 'John' AND b.name = 'Sally'
CREATE (a)-[r:FRIEND_OF{since:date('2013-09-01')}]->(b)
RETURN r

MATCH (a:Person),(b:Person)
WHERE a.name = 'Sally' AND b.name = 'John'
CREATE (a)-[r:FRIEND_OF{since:date('2013-09-01')}]->(b)
RETURN r
```



Le langage Cypher : création de nœuds/associations

- MERGE
  - Soit met en correspondance (match) des nœuds existants et les lie, ou bien crée de nouvelles données et les lie. C'est comme une combinaison de MATCH et CREATE qui permet également de spécifier ce qui se passe si les données ont été mises en correspondance ou créées (VOIR LES CLAUSES ON MATCH et ON CREATE https://neo4j.com/docs/cypher-manual/3.5/clauses/merge/).

- Exemple

```
MERGE (p:Person{name:'Robert'})
RETURN p, labels(p)   /* labels() est une function qui renvoie les labels associés à un noeud */
```

  - Résultat
    - ✓ Un nouveau nœud est crée si aucun nœud existant ne lui correspond

Le langage Cypher : suppression

- Suppression d'un nœud et toutes ses relations

```
MATCH (n { property: value })-[r]-()
DELETE n, r

Exemple:
MATCH (n {name:'Alfred'})-[r]-()
DELETE n, r
```

- Suppression d'un nœud isolé
  - Erreur si le nœud a des associations

```
MATCH (n :label { property: value })
DELETE n

Exemple:
MATCH (n:Person {name:'Robert'})
DELETE n
```

- Suppression de toute la base

```
MATCH (n)
OPTIONAL MATCH (n)-[r]-()
DELETE n,r
```

- Afficher toute la base

```
MATCH (n)
OPTIONAL MATCH (n)-[r]-()
RETURN  n,r
Ou
MATCH (n)
RETURN  *
```


Plus d'info http://neo4j.com/docs/stable/query-delete.html

Le langage Cypher : enchainement de requêtes

- La Clause WITH
  - Manipuler la sortie d'une requête avant d'enchainer sur une autre requête.
  - Limiter le nombre d'entrées passer à une autre requête

- Exemple
  - Récupérer un résultat et faire un traitement dessus

```
MATCH (n { name: 'Anders' })--(m)
WITH m
ORDER BY m.name DESC LIMIT 1
MATCH (m)--(o)
RETURN o.name
```

Le langage Cypher : les fonctions

- Les fonctions d'agrégation (analogue au GROUP BY de SQL)
  - L'agrégation peut être calculée sur tous les sous-graphes correspondants ou divisée en introduisant des clés de regroupement.
  - avg(), count(), max(), min(), sum(), collect (), etc.
    - ✓ collect () renvoie une liste contenant les valeurs renvoyées par une expression. L'utilisation de cette fonction permet d'agréger des données en fusionnant plusieurs enregistrements ou valeurs dans une seule liste.

- Exemple
  - Nombre de relations par personne

```
MATCH (n:Person)-[r]->()
RETURN n.name , count(r)
```

n.name	count(r)
"John"	2
"Sally"	2

  - Liste des ages :

```
MATCH (n:Person)
RETURN collect(n.age)
```

collect(n.age)
[13,33,44]

Le langage Cypher : les fonctions

- Fonctions sur les chaines:
  - left(), right(), replace(), toUpper(), toLower(), substring(), …
  - toString() : transforme son entrée en chaine

```
return toString(125)
```

- Fonctions mathématiques: abs(), rand(), log(), cos(), …
- Divers
  - toInteger(): transforme une chaine ou un booléen en entier
  - toFloat(): transforme une chaine ou un entier en Float
  - toBoolean(): transforme une chaine en TRUE/FALSE
  - Type(): retourne le type d'aune association
  - Size(): nombre d'éléments dans une liste/nombre de caractères dans une chaine
  - Length(p): longueur d'un chemin
  - Nodes(p): liste des nœuds d'un chemin
  - Etc.

```
match(n:Person)-[r]-(b:Book)
return distinct type(r)
```

```
MATCH p = (a:Person)--(b:Person)
WHERE a.name = 'John'
RETURN length(p)
```

https://neo4j.com/docs/cypher-manual/current/functions/

Le langage Cypher : les fonctions

- Fonctions de manipulation de dates
  - date(): transforme une chaine ou un Json en date

```
return date({year: 2013, month: 9, day: 1})

return date('2013-09-01')
```

```
"2013-09-01"
```

  - Datetime(): fonctionne comme date() mais avec en plus la prise en compte de l'heure

```
return datetime({year: 2013, month: 9, day: 1, hour:10, minute:20})

return datetime('2013-09-01T10:20:00')
```

```
"2013-09-01T10:20:00Z"
```

- Calcul de durées
  - Duration() : calcule une durée à partir de composants temporels (years, months, weeks, days, hours, minutes, seconds, milliseconds, etc.)

```
return duration({days: 14, hours:16, minutes: 12})
```

```
P14DT16H12M
```

  - duration.between(a, b): calcule la durée entre deux instants (dates par exemple)
  - duration.inMonths(a, b), duration.inDays(a, b), duration.inSeconds(a, b): precise le niveau de détail du résultat

```
return duration.inMonths(date("2022-02-03"), date("2020-10-25"))
return duration.inDays(date("2022-02-03"), date("2021-01-25"))
return duration.inSeconds(date("2022-02-03"), date("2021-10-25"))
```

```
P-1Y-3M
P-1Y-3M-11D
P-1Y-3M-11D
```

Le langage Cypher : fonctions

- Les Prédicats:
  - Les prédicats sont des fonctions booléennes qui renvoient vrai ou faux pour un ensemble donné d'entrées non nulles.
  - Ils sont le plus souvent utilisés pour filtrer les sous-graphes dans la partie WHERE d'une requête. https://neo4j.com/docs/cypher-manual/current/functions/predicate/
    - ✓ all()
    - ✓ any()
    - ✓ exists()
    - ✓ Etc.
- Exemple : Tous les nœuds des chemins renvoyés auront une propriété(ic age >30 ans)

```
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
```

Le langage Cypher : les opérateurs

- Pour les chaines (sensibles à la casse)
  - STARTS WITH: recherche de préfixe sur les chaînes
  - ENDS WITH: recherche de suffixe sur les chaînes
  - CONTAINS: recherche d'inclusion sur les chaînes

- Expressions régulières: =~

```
Match (a: Person)
WHERE a.name =~ '.*ous.*'
RETURN a.name
```

  - Les expressions régulières suivent le modèle java https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/regex/Pattern.html

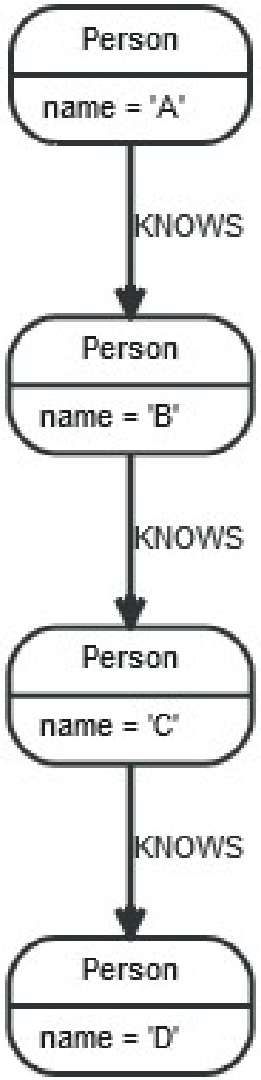
- Les autres opérateurs sont semblables à SQL

Le langage Cypher

- La clause FOREACH
  - utilisée pour mettre à jour les données d'une liste, qu'il s'agisse des composants d'un chemin ou du résultat d'une agrégation
- Exemple
  - Mettre l'attribut *marked* à vrai pour tous les nœuds d'un chemin

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

  - Résultats:



Le langage Cypher : les Listes

- Création d'une liste
  - Littéralement :

```
RETURN [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] AS list
```

  - Avec range: les crochets permettent de sélectionner des éléments de l'intervalle.

```
RETURN range(0,10)[..5]  /* résultat:    [0,1,2,3,4]
```

  - Avec collect()

```
MATCH (n:Person)
RETURN collect(n.age)
 /* résultat:    collect(n.age)
[20, 24, 24, 28, 27, 31, 42, 19,]
```

  - Liste en compréhension (par filtrage d'autres listes):

```
RETURN [x IN range(0,10) WHERE x % 2 = 0 | x^3] AS result
/* résultat: [0.0,8.0,64.0,216.0,512.0,1000.0] */
```

  - Motif en compréhension (filtrage de motifs et projection)

```
MATCH (a:Person { name: 'alice' })
RETURN [(a)-->(b) where b:Person AND b.name<>a.name | b.age] AS ages
```

    - ✓ | sépare la partie sélection de la partie action et sighifie « pour ceux-ci faire »

Le langage Cypher

- UNWIND
  - développe une liste en une séquence de lignes ou de valeurs
- Exemple

```
UNWIND [1, 2, 3, NULL ] AS x
RETURN x, 'val' AS y
```

  - Résultats :

x	y
1	"val"
2	"val"
3	"val"
<null>	"val"

4 rows