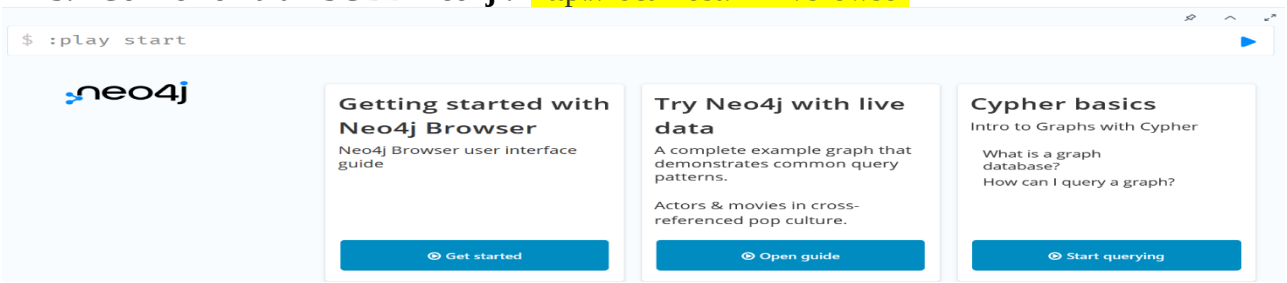
TP n3 en Big Data

Framework Neo4j

1. **Télécharger et installer Neo4j Desktop (https://neo4j.com/docs/desktop-manual/current/installation/download-installation/)**
2. **Télécharger Neo4j community : neo4j-community-4.4.25-windows (https://isolution.pro/fr/t/neo4j/neo4j-quick-guide/neo4j-guide-rapide)**
   Télécharger Neo4j entreprise: neo4j-enterprise-3.5.35-windows
3. **Connexion à un SGBD Neo4j :** http://localhost:7474/browser



L'URL par défaut du navigateur Neo4j est http://localhost:7474/browser
L'URL de connexion par défaut à Neo4j est **bolt://localhost:7687**

Gérer les commandes de connexion

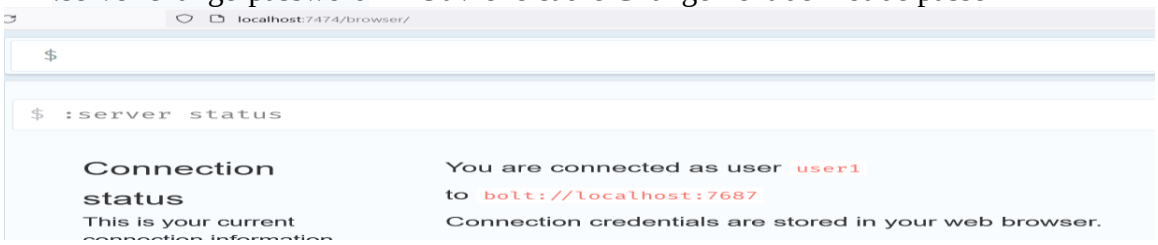
La commande **« server «** permet de gérer la connexion à Neo4j, comme la connexion, la déconnexion et l'affichage des métadonnées de la connexion actuelle

Usage

:server <action>

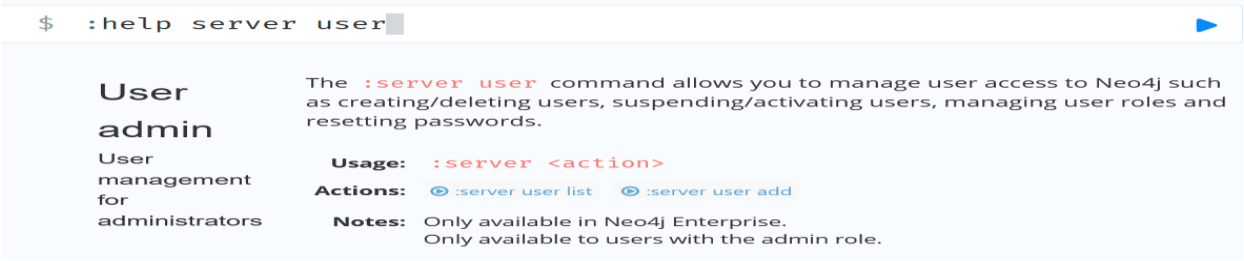
Actions

- :server status — État de la connexion.
- :server change-password — Ouvre le cadre Changement de mot de passe



Utilisateur

:help server user - ouvre le cadre **Administrateur de l'utilisateur**.

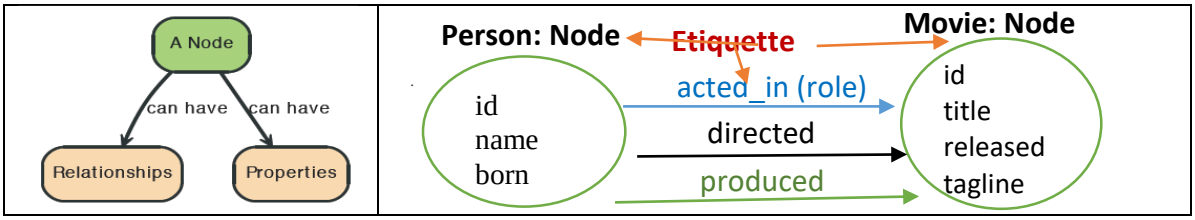


Auth

- :server connect — Ouvre le cadre **Se connecter à Neo4j**.
- :server disconnect — Ouvre le cadre Déconnecté, qui indique que l'utilisateur actuel est **déconnecté** du serveur. Ensuite, il exécute le automatiquement.: server connect

Neo4j cypher-shell : C:\neo4j-community-5.11.0\bin>cypher-shell

Prise en main du langage Cypher (Neo4j)



SQL

```
CREATE TABLE movie (
  id INTEGER,
  title VARCHAR(100),
  released INTEGER,
  tagline VARCHAR(100)
);
CREATE TABLE person (
  id INTEGER,
  name VARCHAR(100),
  born INTEGER
);
CREATE TABLE acted_in (
  role varchar(100),
  person_id INTEGER,
  movie_id INTEGER
);
CREATE TABLE directed (
  person_id INTEGER,
  movie_id INTEGER
);
CREATE TABLE produced (
  person_id INTEGER,
  movie_id INTEGER
);
INSERT INTO movie (id, title, released, tagline) VALUES (
  (1, 'The Matrix', 1999, 'Welcome to the Real World'),
  (2, 'The Devil"s Advocate', 1997, 'Evil has its winning ways'),
  (3, 'Monster', 2003, 'The first female serial killer of America')
);
INSERT INTO person (id, name, born) VALUES (
  (1, 'Keanu Reeves', 1964),
  (2, 'Carrie-Anne Moss', 1967),
  (3, 'Laurence Fishburne', 1961),
  (4, 'Hugo Weaving', 1960),
  (5, 'Andy Wachowski', 1967),
  (6, 'Lana Wachowski', 1965),
  (7, 'Joel Silver', 1952),
  (8, 'Charlize Theron', 1975),
  (9, 'Al Pacino', 1940),
  (10, 'Taylor Hackford', 1944)
);
INSERT INTO acted_in (role, person_id, movie_id) VALUES (
  ('Neo', 1, 1),
  ('Trinity', 2, 1),
  ('Morpheus', 3, 1),
  ('Agent Smith', 4, 1),
  ('Kevin Lomax', 1, 2),
  ('Mary Ann Lomax', 8, 2),
  ('John Milton', 9, 2),
  ('Aileen', 8, 3)
);
INSERT INTO directed (person_id, movie_id) VALUES (
  (5, 1),
  (6, 1),
  (10, 2)
);
INSERT INTO produced (person_id, movie_id) VALUES (
  (7, 1),
  (8, 3)
);
```

Neo4j : Cypher

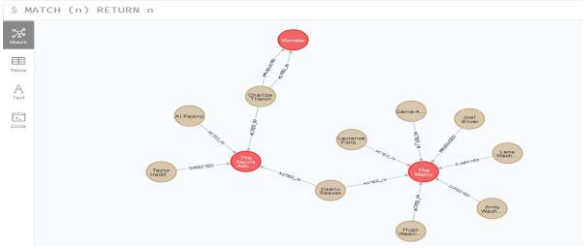
```
CREATE (TheMatrix:Movie{title:'The Matrix',released:1999,tagline:
'Welcome to the Real World'})
CREATE (TheDevilsAdvocate:Movie{title:"The Devil's Advocate",
released:1997,tagline:'Evil has its winning ways'})
CREATE (Monster:Movie{title:'Monster',released: 2003,tagline:'The
 first female serial killer of America'})

CREATE (Keanu:Person{name:'Keanu Reeves',born:1964})
CREATE (Carrie:Person{name:'Carrie-Anne Moss',born:1967})
CREATE (Laurence:Person{name:'Laurence Fishburne',born:1961})
CREATE (Hugo:Person{name:'Hugo Weaving',born:1960})
CREATE (AndyW:Person{name:'Andy Wachowski',born:1967})
CREATE (LanaW:Person{name:'Lana Wachowski',born:1965})
CREATE (JoelS:Person{name:'Joel Silver',born:1952})
CREATE (Charlize:Person{name:'Charlize Theron',born:1975})
CREATE (Al:Person{name:'Al Pacino',born:1940})
CREATE (Taylor:Person{name:'Taylor Hackford',born:1944})

CREATE
  (Keanu)-[:ACTED_IN {roles:['Neo']}]->(TheMatrix),
  (Carrie)-[:ACTED_IN {roles:['Trinity']}]->(TheMatrix),
  (Laurence)-[:ACTED_IN {roles:['Morpheus']}]->(TheMatrix),
  (Hugo)-[:ACTED_IN {roles:['Agent Smith']}]->(TheMatrix),

  (AndyW)-[:DIRECTED]->(TheMatrix),
  (LanaW)-[:DIRECTED]->(TheMatrix),

  (JoelS)-[:PRODUCED]->(TheMatrix)

CREATE
  (Keanu) -[:ACTED_IN {roles:['Kevin Lomax']}]-
>(TheDevilsAdvocate),
  (Charlize)-[:ACTED_IN {roles:['Mary Ann Lomax']}]-
>(TheDevilsAdvocate),
  (Al) -[:ACTED_IN {roles:['John Milton']}]->(TheDevilsAdvocate),

  (Taylor)-[:DIRECTED]->(TheDevilsAdvocate),

  (Charlize) -[:ACTED_IN {roles:['Aileen']}]->(Monster),

  (Charlize) -[:PRODUCED {roles:['Aileen']}]->(Monster)
```

Exemples de requetes :

SQL	Neo4j : Cypher
SELECT movie.title FROM movie;	MATCH (movie:Movie) RETURN movie.title;
SELECT movie.title FROM movie WHERE movie.released > 2000;	MATCH (movie:Movie) WHERE movie.released > 2000 RETURN movie.title;
Énumérons toutes les personnes et les films dans lesquels elles ont joué. SELECT person.name, movie.title FROM person JOIN acted_in AS acted_in ON acted_in.person_id = person.id JOIN movie ON acted_in.movie_id = movie.id;	MATCH (person:Person)-[:ACTED_IN]->(movie:Movie) RETURN person.name, movie.title;
Co-acteurs de Keanu Reeves SELECT DISTINCT co_actor.name FROM person AS keanu JOIN acted_in AS acted_in1 ON acted_in1.person_id = keanu.id JOIN acted_in AS acted_in2 ON acted_in2.movie_id = acted_in1.movie_id JOIN person AS co_actor ON acted_in2.person_id = co_actor.id AND co_actor.id <> keanu.id WHERE keanu.name = 'Keanu Reeves';	MATCH (keanu:Person)-[:ACTED_IN]->(movie:Movie), (coActor:Person)-[:ACTED_IN]->(movie) WHERE keanu.name = 'Keanu Reeves' RETURN DISTINCT coActor.name;
Qui sont les acteurs/producteurs SELECT person.name FROM person WHERE person.id IN (SELECT person_id FROM acted_in) AND person.id IN (SELECT person_id FROM produced)	MATCH (person:Person) WHERE (person)-[:ACTED_IN]->() AND (person)-[:PRODUCED]->() RETURN person.name
Agrégation SELECT director.name, count(*) FROM person keanu JOIN acted_in ON keanu.id = acted_in.person_id JOIN directed ON acted_in.movie_id = directed.movie_id JOIN person AS director ON directed.person_id = director.id WHERE keanu.name = 'Keanu Reeves' GROUP BY director.name ORDER BY count(*) DESC	MATCH (keanu:Person {name: 'Keanu Reeves'})-[:ACTED_IN]->(movie:Movie), (director:Person)-[:DIRECTED]->(movie) RETURN director.name, count(*) ORDER BY count(*) DESC
Suppression des proprietes Alter table Drop column nom_colonne	MATCH (p:Post) WHERE exists(p.text) REMOVE p.text
Suppression des noeuds et des relations Drop table nom_table;	MATCH (n) OPTIONAL MATCH (n)-[r]-() DELETE n,r

Exercice 1.

1. En utilisant Neo4j Desktop, créer un nouvel utilisateur (user1/user1)
2. Accéder à l'interface utilisateur via votre navigateur préféré : http://localhost:7474/
3. Reprendre les requetes du tableau « Prise en main du langage Cypher (Neo4j) » pour créer le graphe suivant
4. Afficher le schéma actuel du graphique.



5. Afficher la liste des nœuds personnes
6. Afficher les noms des 3 premieres personnes
7. Récupérer le nœud de l'acteur Keanu Reeves
8. Afficher la liste actuelle des clés de propriété dans le graphe
9. Afficher la liste des co-acteurs de Keanu Reeves
10. Afficher tous les nœuds connectés au film, The Matrix, ainsi que leurs relations.
11. Afficher les titres des films de Keanu Reeves ainsi que les rôles qu'il a joué
12. Qui a réalisé le film " The Matrix " ?
13. Comment les gens sont-ils reliés au film "The Matrix" (le nom et le role de chaque personne)
14. Afficher les noms des acteurs avec le nombre de films pour chacun. Trier les résultats par nombre de films dans l'ordre décroissant
15. Retourner le nombre d'acteurs
16. Retourner le nombre de personnes
17. Retourner le nombre d'acteurs et le nombre de personnes
18. Ajouter la propriété « gender » au nœud Person.

Exercice 2.

1. Donner les instructions Cypher qui permettent de réaliser les opérations suivantes :
   1.1. Création de nœuds ayant les caractéristiques spécifiées dans le tableau 1.
   1.2. Création de relations ayant les caractéristiques spécifiées dans le tableau 2.

Id	Label	Attribut 1	Attribut 2
1	Personne	Nom :Smith	Age :45
2	Personne	Nom :Alice	Age :55
3	Personne	Nom :Mark	Age :36
4	Logement	Type :Appartement	Modele :3 pieces
5	Logement	Type :Maison	Modele :R+2
6	Logement	Type : Appartement	Modele :2 pieces

Table 1 : Nœuds a inserer

De	Label	Vers	Attribut
1	Posseder	4	Depuis :2014
2	Posseder	5	Depuis :2008
3	Posseder	6	Depuis :2010
1	Vendre	4	En :2022
2	Vendre	5	En :2021
3	Vendre	6	En :2020
2	Acheter	4	En :2022

Table 2 : Relations a inserer

2. Ecrire et tester les requêtes suivantes :
   2.1. Retourner le nombre de personnes
   2.2. Afficher les differents modeles des logements de type Appartement
   2.3. Qui possède un logement depuis au moins 2010 ? Vous ne renverrez que l'attribut Nom de la ou des personnes concernées
   2.4. Quels sont les propriétaires de maison ? Vous ne renverrez que l'attribut Nom de la ou des personnes concernées.
   2.5. Qui achette logement vendu par Smith
   2.6. Afficher les noms des personnes avec le nombre de logements pour chacun. Trier les résultats par nombre de logements dans l'ordre décroissant
3. Effectuer les mises à jour suivantes :
   3.1. Modifier le nom de Smith par Mark et augmenter son âge de 10 ans ;
   3.2. Ajouter un attribut superficie valant 100m2 aux logements de type maison.