1 Création de nœuds et relations (CREATE)

1.1 Créer un film et des acteurs - "The Departed"

Important : Exécuter tout le code en **un seul bloc** ! Sinon, les instructions CREATE pour les arcs doivent être précédées d'un MATCH pour récupérer les nœuds.

```
CREATE (departed:Movie {title:'The Departed', released:2006,
↪  tagline:'Good and Evil'})
CREATE (leo:Person {name:'Leonardo Di Caprio', born: 1974})
CREATE (matt:Person {name:'Matt Damon', born: 1970})
CREATE (leo)-[:ACTED_IN {roles: ['Billy']}]->(departed)
CREATE (matt)-[:ACTED_IN {roles: ['Colin Sullivan']}]->(departed)
```

Ce que fait ce code :

- Crée un nœud `Movie` avec les propriétés `title`, `released`, `tagline`
- Crée 2 nœuds `Person` avec les propriétés `name`, `born`
- Crée des relations `ACTED_IN` avec la propriété `roles`

1.2 Ajouter une relation à un nœud existant

Si le nœud existe déjà (comme Jack Nicholson), il faut d'abord le récupérer avec `MATCH` :

```
MATCH (jack:Person {name: 'Jack Nicholson'})
MATCH (departed:Movie {title:'The Departed'})
CREATE (jack)-[:ACTED_IN {roles: ['Frank Costello']}]->(departed)
```

Principe clé : `CREATE` crée toujours de nouveaux nœuds. Pour lier des nœuds existants, on doit d'abord les `MATCH`.

2 Visualiser tous les nœuds et relations

Affiche l'ensemble du graphe avec tous les nœuds, relations et leurs connexions.

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

Résultat : 174 nœuds et 256 relations

3 Statistiques du graphe

Le graphe contient :

- **174 nœuds**
- **256 relations**

4 Requêtes sur un film spécifique

4.1 Relations autour de “The Departed”

Trouve toutes les entités connectées au film “The Departed” (acteurs, réalisateurs, etc.).

```
MATCH (m:Movie {title: "The Departed"})-[r]-(n)
RETURN m, r, n
```

Note : Le - (sans flèche) signifie “dans n’importe quelle direction”.

5 Relations Personne-Film

5.1 Toutes les relations entre personnes et films

Affiche toutes les connexions entre les personnes et les films (peu importe le type de relation).

```
MATCH (p:Person)-[r]-(m:Movie)
RETURN p, r, m
```

5.2 Uniquement les acteurs

Filtre pour n’afficher que les personnes ayant joué dans des films.

```
MATCH (p:Person)-[r:ACTED_IN]->(m:Movie)
RETURN p, r, m
```

6 Réalisateurs-Acteurs

6.1 Personnes ayant réalisé ET joué dans un même film

Trouve les personnes qui ont à la fois réalisé et joué dans le même film.

```
MATCH (p:Person)-[:DIRECTED]->(m:Movie)
MATCH (p)-[:ACTED_IN]->(m)
RETURN p, m
```

6.2 Version optimisée avec un seul MATCH

Même requête mais avec une syntaxe plus concise.

```
MATCH (p:Person)-[:ACTED_IN]->(m:Movie)<-[:DIRECTED]-(p)
RETURN p, m
```

Astuce : La flèche <- indique la direction de la relation.

7 Critiques de films

7.1 Tous les critiques

Trouve toutes les personnes ayant critiqué des films.

```
MATCH (p:Person)-[r:REVIEWED]-(m:Movie)
RETURN p
```

7.2 Films avec plusieurs critiques

Identifie les films ayant reçu plus d'une critique.

```
MATCH (m:Movie)-[:REVIEWED]-(reviewer:Person)
WITH m, collect(reviewer) AS reviewers
WHERE size(reviewers) > 1
RETURN m, reviewers
```

Principe : `WITH` permet d'agréger les résultats avant de filtrer.

8 Filtres temporels

8.1 Films récents (après 2010)

Liste tous les films sortis après 2010.

```
MATCH (m:Movie)
WHERE m.released > 2010
RETURN m
```

8.2 Acteurs dans des films récents

Trouve les acteurs ayant joué dans des films sortis après 2010.

```
MATCH (actor:Person)-[r:ACTED_IN]->(m:Movie)
WHERE m.released > 2010
RETURN actor, m
```

9 Co-acteurs

9.1 Paires d'acteurs dans des films récents

Identifie les paires d'acteurs ayant joué ensemble dans des films post-2010.

```
MATCH (a1:Person)-[:ACTED_IN]->(m:Movie)<-[:ACTED_IN]-(a2:Person)
WHERE m.released > 2010 AND elementId(a1) < elementId(a2)
RETURN a1.name AS Acteur1, a2.name AS Acteur2, m.title AS Film,
↪  m.released AS Annee
```

Note : `elementId(a1) < elementId(a2)` évite les doublons (A-B et B-A).

9.2 Co-acteurs récurrents

Trouve les paires d'acteurs ayant joué ensemble dans plusieurs films.

```
MATCH (a1:Person)-[:ACTED_IN]->(m:Movie)<-[:ACTED_IN]-(a2:Person)
WITH a1, a2, collect(m) AS movies
WHERE elementId(a1) < elementId(a2) AND size(movies) > 1
UNWIND movies AS movie
RETURN a1, a2, movie
```

Principe : `UNWIND` transforme une liste en lignes individuelles.

10 Relations entre critiques

10.1 Réseau de followers (version simple)

Trouve tous les critiques qui suivent d'autres critiques (jusqu'à plusieurs niveaux).

```
MATCH (reviewer1:Person)-[:FOLLOWS*1..]->(reviewer2:Person)
RETURN reviewer1, reviewer2
```

Note : `*1..` signifie "1 relation ou plus" (chemins de longueur variable).

10.2 Réseau de followers (version groupée)

Pour chaque critique, affiche tous les critiques qu'il suit (directement ou indirectement).

```
MATCH (r1:Person)-[:FOLLOWS*1..]->(r2:Person)
RETURN r1.name AS reviewer, collect(DISTINCT r2.name) AS follows_network
```

Amélioration : Cette version regroupe les résultats par critique et élimine les doublons.

11 Exploration du réseau

11.1 Réseau de Clint Eastwood (3 degrés max)

Explore le réseau de Clint Eastwood jusqu'à 3 degrés de séparation.

```
MATCH (clint:Person {name: "Clint Eastwood"})
MATCH path = (clint)-[*1..3]-(n)
RETURN DISTINCT n
LIMIT 12
```

Principe : `[*1..3]` signifie "entre 1 et 3 relations de distance".

12 Calculs d'âge

12.1 Âge des acteurs dans "Apollo 13"

Calcule l'âge de chaque acteur lors du tournage d'Apollo 13.

```
MATCH (actor:Person)-[:ACTED_IN]->(m:Movie {title: "Apollo 13"})
RETURN m, actor.name, m.released - actor.born AS age_lors_tournage
```

12.2 Âge moyen du casting d'Apollo 13

Calcule l'âge moyen des acteurs d'Apollo 13 lors du tournage.

```
MATCH (actor:Person)-[:ACTED_IN]->(m:Movie {title: "Apollo 13"})
RETURN m, avg(m.released - actor.born) AS age_moyen
```

Fonction : `avg()` calcule la moyenne.

13 Analyses statistiques

13.1 Âge moyen par film

Calcule l'âge moyen des acteurs pour chaque film.

```
MATCH (m:Movie)<-[:ACTED_IN]-(actor:Person)
RETURN m, avg(m.released - actor.born) AS age_moyen
```

13.2 Top 10 des films avec les castings les plus jeunes

Classe les films par âge moyen des acteurs (du plus jeune au plus âgé).

```
MATCH (m:Movie)<-[:ACTED_IN]-(actor:Person)
RETURN m, avg(m.released - actor.born) AS age_moyen
ORDER BY age_moyen ASC
LIMIT 10
```

Clauses : `ORDER BY` trie les résultats, `LIMIT` restreint le nombre de lignes.

14 Corrections et bonnes pratiques

14.1 Erreurs courantes corrigées :

1. **Créer des relations sans MATCH préalable** : Si le nœud existe déjà, il faut le récupérer avec `MATCH` avant de créer une relation
2. **Variable `actor` non définie** : Toujours définir la variable dans le `MATCH`
3. **Direction des relations** : `(m:Movie)-[r:ACTED_IN]-(actor)` → Correct : `(m:Movie)<-[:ACTED_IN]-(ac`

14.2 Conseils Neo4j :

- **`elementId()`** : Pour éviter les doublons dans les paires (A-B vs B-A)
- **`collect()`** : Pour agréger des résultats en liste
- **`DISTINCT`** : Élimine les doublons dans les résultats
- **`*1..` ou `*1..3`** : Pour les chemins de longueur variable
- **`WITH`** : Pour enchaîner des étapes de traitement (agrégation puis filtrage)
- **`UNWIND`** : Pour transformer une liste en lignes
- **CREATE en bloc** : Toujours créer nœuds + relations en un seul bloc si possible

14.3 Syntaxe des relations :

- `(a)-[r]->(b)` : relation dirigée de a vers b
- `(a)-[r]-(b)` : relation dans n'importe quelle direction
- `(a)<-[r]-(b)` : relation de b vers a
- `(a)-[:TYPE]->(b)` : relation avec un type spécifique
- `(a)-[*1..3]->(b)` : chemin de 1 à 3 relations

15 Résumé des fonctions utiles

Fonction	Usage	Exemple
count()	Compte les éléments	RETURN count(n)
avg()	Moyenne	RETURN avg(m.released)
sum()	Somme	RETURN sum(m.budget)
min() / max()	Min / Max	RETURN max(m.released)
collect()	Agrège en liste	collect(actor.name)
size()	Taille d'une liste	size(movies)
elementId()	ID unique du nœud	elementId(n)

16 Anti-patterns : Erreurs à éviter absolument

16.1 Erreur 1 : Créer au lieu de lier

MAUVAIS :

```
CREATE (p:Person {name:"Jack"})-[:ACTED_IN]->(m:Movie {title:"Film"})
```

Problème : Crée TOUJOURS de nouveaux nœuds, même s'ils existent déjà → doublons garantis !

BON :

```
MATCH (p:Person {name:"Jack"})
MATCH (m:Movie {title:"Film"})
CREATE (p)-[:ACTED_IN]->(m)
```

16.2 Erreur 2 : Oublier d'éviter les doublons dans les paires

MAUVAIS :

```
MATCH (a1:Person)-[:ACTED_IN]->(m)<-[:ACTED_IN]-(a2:Person)
RETURN a1, a2, m
```

Problème : Retourne Tom-Jerry ET Jerry-Tom (doublon)

BON :

```
MATCH (a1:Person)-[:ACTED_IN]->(m)<-[:ACTED_IN]-(a2:Person)
WHERE elementId(a1) < elementId(a2)
RETURN a1, a2, m
```

16.3 Erreur 3 : Mauvaise direction de relation

MAUVAIS :

```
MATCH (m:Movie)-[:ACTED_IN]->(p:Person)  // Film joue dans personne ???
RETURN p
```

BON :

```
MATCH (p:Person)-[:ACTED_IN]->(m:Movie)  // Personne joue dans film
RETURN p
```

16.4 Erreur 4 : Filtrer avant d'agréger

MAUVAIS :

```
MATCH (m:Movie)<-[:REVIEWED]-(r:Person)
WHERE count(r) > 1  // count() ne fonctionne pas ici !
RETURN m
```

BON :

```
MATCH (m:Movie)<-[:REVIEWED]-(r:Person)
WITH m, collect(r) AS reviewers
WHERE size(reviewers) > 1
RETURN m
```

17 A savoir

17.1 Absolument :

1. **Créer un film et des acteurs** → Section 0
2. **Trouver tous les acteurs d'un film** → Section 4.2
3. **Personnes ayant réalisé ET joué** → Section 5
4. **Paires d'acteurs ayant joué ensemble** → Section 8.1
5. **Films avec critiques multiples** → Section 6.2
6. **Calculer un âge moyen** → Section 11-12

17.2 bon à savoir :

7. **Exploration réseau avec chemins variables** → Section 10
8. **Réseau de followers** → Section 9
9. **Top N avec ORDER BY + LIMIT** → Section 12.2

17.3 Piège :

Question : "Ajoutez l'acteur Brad Pitt au film Fight Club"

Réponse piège :

```
CREATE (brad:Person {name:"Brad Pitt"})-[:ACTED_IN]->(fight:Movie
↪ {title:"Fight Club"})
```

Bonne réponse :

```
MATCH (brad:Person {name:"Brad Pitt"})
MATCH (fight:Movie {title:"Fight Club"})
CREATE (brad)-[:ACTED_IN]->(fight)
```

Dernière astuce : Si je bloque, je peux me demander :

1. Qu'est-ce que je cherche ?
2. D'où je pars ?
3. Quelle relation relie ces nœuds ?