

Royaume du Maroc

Ministère de l'Enseignement Supérieur,
de la Recherche Scientifique et de l'Innovation

Université Cadi Ayyad

École Nationale des Sciences Appliquées

Marrakech



المملكة المغربية
وزارة التعليم العالي والبحث العلمي والإبتكار
جامعة القاضي عياض
المدرسة الوطنية للعلوم التطبيقية
مراكش



Présenté par:

- ACHQIBOU Hassania
- AIT EL MOUDEN Hafsa
- BENKSSAB Salma
- NASSIRI Noussaiba

Encadré par:

M. BEKKARI Aissam



SOMMAIRE

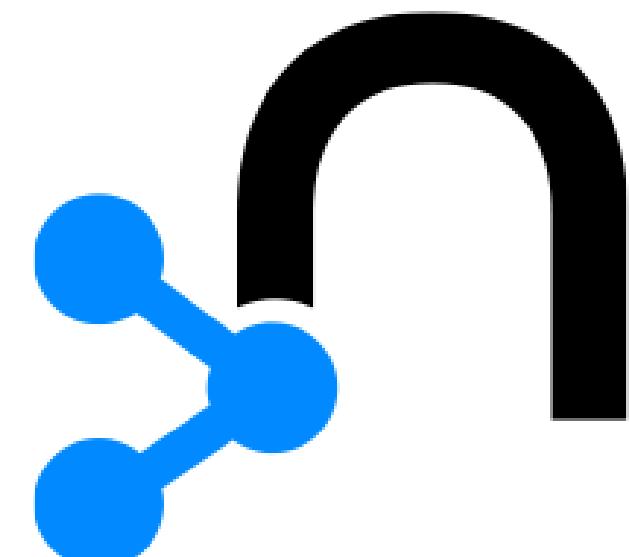
- 1 **Introduction à Neo4j**
- 2 **Modèle de données dans Neo4j**
- 3 **Cypher - Langage de requête de Neo4j**
- 4 **Option d'installation**
- 5 **Schéma et contraintes dans Neo4j**
- 6 **Performance et optimisation dans Neo4j**
- 7 **Exigences de système**
- 8 **l'intégration de Neo4j avec d'autres technologies**
- 9 **Conclusion**

INTRODUCTION À NEO4J

- Présentation de Neo4j
- Importance et cas d'utilisation

INTRODUCTION À NEO4J

- Neo4j est un système de gestion de base de données **orienté graphe**, conçu pour stocker, gérer et analyser des données interconnectées. Contrairement aux bases de données traditionnelles basées sur des tables relationnelles, Neo4j utilise des structures de graphe pour représenter et stocker les données. Dans un graphe, les données sont organisées sous forme de **nœuds**, qui représentent des entités, et de relations, qui décrivent les liens entre ces entités.



CAS D'UTILISATION

- ▶ **Recherche de fraude et gestion des risques** : Les entreprises financières utilisent Neo4j pour détecter les schémas de fraude complexes en analysant les relations entre les transactions, les comptes, les utilisateurs et d'autres entités. Cela leur permet d'identifier rapidement les activités suspectes et de réduire les risques.
- ▶ **Gestion des identités et des autorisations** : Les systèmes d'authentification et de gestion des autorisations utilisent Neo4j pour gérer les relations complexes entre les utilisateurs, les rôles, les permissions et les ressources. Cela permet de garantir un accès sécurisé aux informations sensibles tout en maintenant une gestion flexible des droits.

CAS D'UTILISATION

- ▶ **Recommandations de produits et personnalisation** : Les entreprises de commerce électronique et de diffusion de contenu utilisent Neo4j pour analyser les interactions entre les utilisateurs, les produits, les contenus et les préférences. Cela leur permet de fournir des recommandations personnalisées et d'améliorer l'expérience utilisateur.

MODÈLE DE DONNÉES DANS NEO4J

- Modèle de données graphiques

MODÈLE DE DONNÉES GRAPHIQUES

Le modèle de données graphiques utilisé par Neo4j est fondé sur une structure simple qui repose sur trois concepts principaux : les nœuds, les relations et les propriétés:

1. Nœuds (Nodes)

Les nœuds sont les entités de base dans un graphe. Chaque nœud représente une entité ou un objet dans le domaine de données que vous modélisez. Par exemple, dans un réseau social, les nœuds peuvent représenter des utilisateurs, des publications, des commentaires, etc.

- **Étiquette (Label)** : Les nœuds peuvent avoir un ou plusieurs labels qui définissent leur type ou leur rôle. Par exemple, un nœud avec l'étiquette "Personne" peut représenter un utilisateur dans un réseau social.



MODÈLE DE DONNÉES GRAPHIQUES

2. Relations (Relationships)

Les relations définissent les connexions entre les nœuds. Elles représentent les interactions ou les associations entre les entités.

- **Type** : Chaque relation a un type qui définit la nature de la connexion entre les nœuds. Par exemple, une relation de type "AMIS" peut lier deux nœuds "Personne" pour indiquer une amitié.
- **Direction** : Les relations en Neo4j sont directionnelles, allant d'un nœud de départ à un nœud d'arrivée. La direction peut être utilisée pour spécifier le sens de la relation, bien qu'elle soit souvent interprétée de manière bidirectionnelle dans les requêtes.

MODÈLE DE DONNÉES GRAPHIQUES

3. Propriétés (Properties)

Les propriétés sont des paires clé-valeur qui peuvent être associées à la fois aux nœuds et aux relations. Elles stockent des données additionnelles qui qualifient les entités ou les liens.

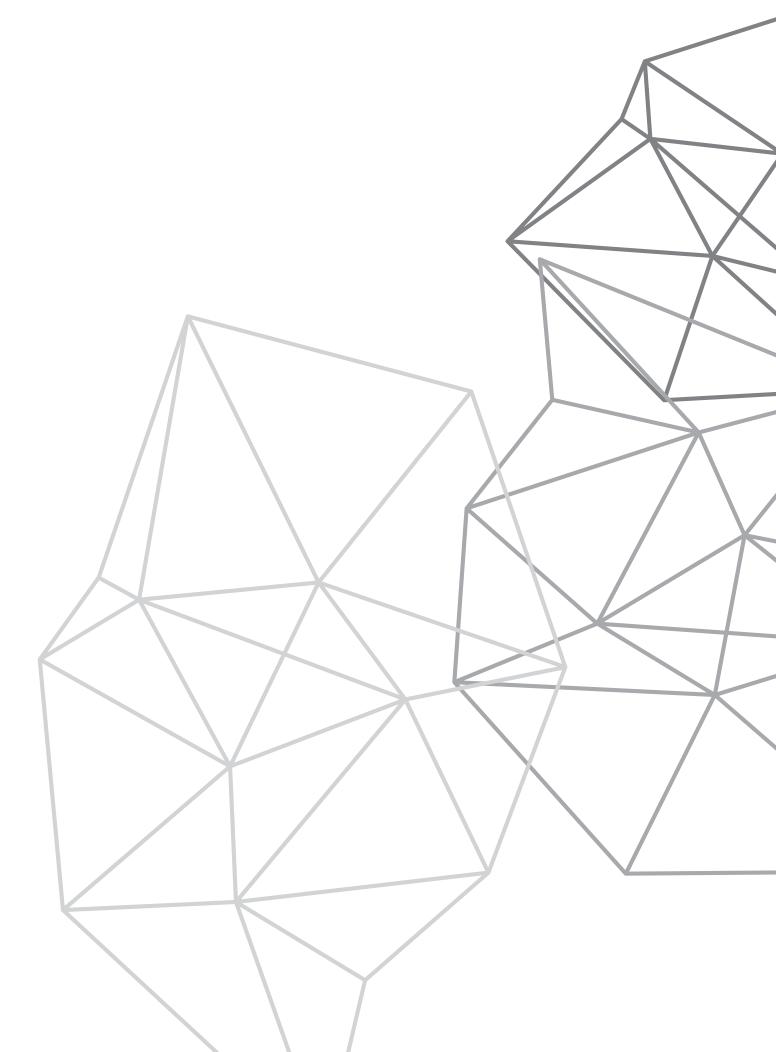
- **Exemple de propriétés de nœud :** Un nœud "Personne" pourrait avoir des propriétés telles que nom: "Alice", âge: 30, et email: "alice@example.com".
- **Exemple de propriétés de relation :** Une relation "TRAVAILLE_POUR" entre un nœud "Personne" et un nœud "Entreprise" pourrait avoir une propriété poste: "Ingénieur" et date_debut: "2020-01-01".

CYPHER - LANGAGE DE REQUÊTE DE NEO4J

- Qu'est-ce que Cypher?
- syntaxe de Cypher

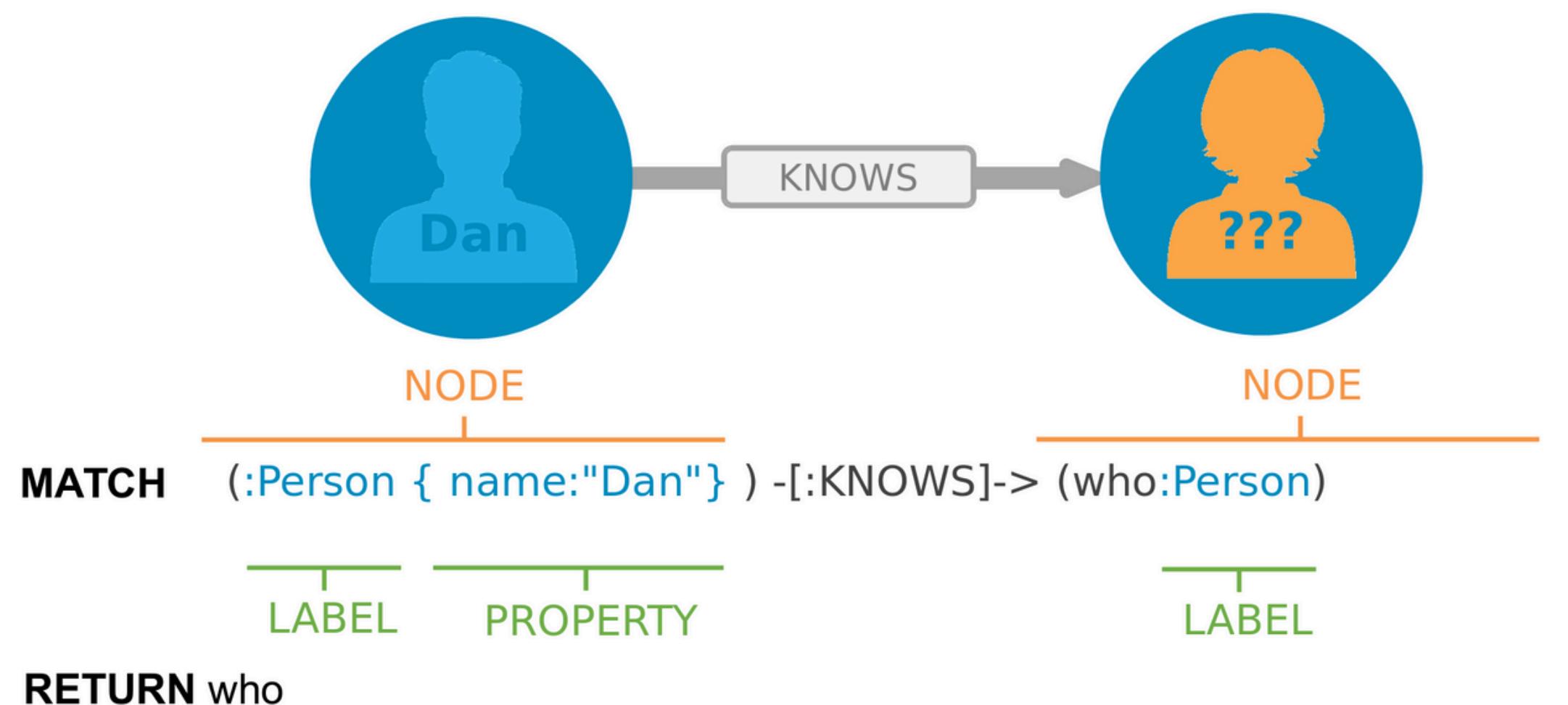
QU'EST-CE QUE CYpher?

- ▶ Cypher est le langage de requête de Neo4j, spécialement conçu pour travailler avec des données de graphes. Sa syntaxe est déclarative et intuitive, ce qui rend les requêtes lisibles et faciles à écrire. Cela permet de se concentrer sur ce que l'on veut trouver sans se soucier de comment le trouver exactement.



SYNTAXE DE CYpher

► Structure Générale d'une Requête Cypher



-> Label : Le type de nœud, ici Person
-> Propriété : Une caractéristique du nœud, ici name: "Dan"
-> Relation : Le type de lien entre les nœuds, ici [:KNOWS]
-> Variable : Un alias pour faire référence au nœud ou à la relation, ici whom



SYNTAXE DE CYpher

Les **nœuds** sont représentés avec des parenthèses

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

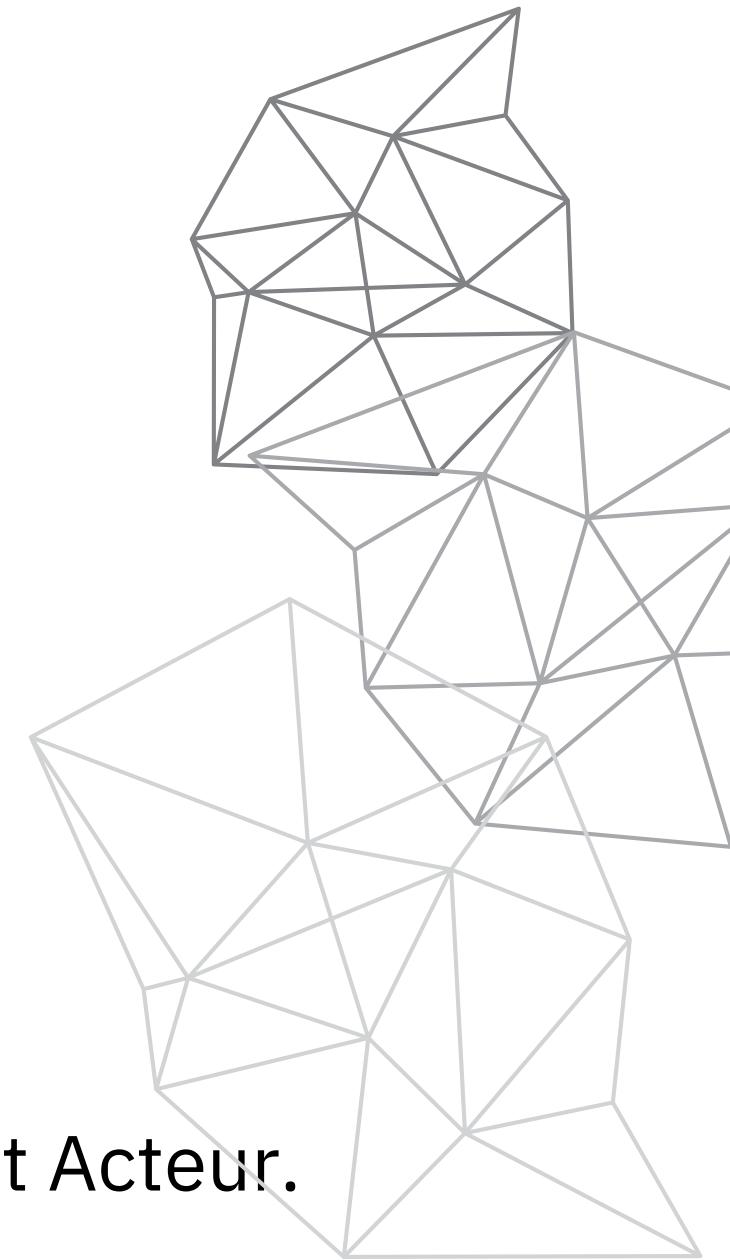
Voici quelques exemples :

0 : 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.



SYNTAXE DE CYPHER

Les relations sont représentées par deux tirets avec un '>', ce qui ressemble à une flèche : -->

Pour identifier la relation , on peut lui donner un nom comme ceci : **-[maRelation]->**

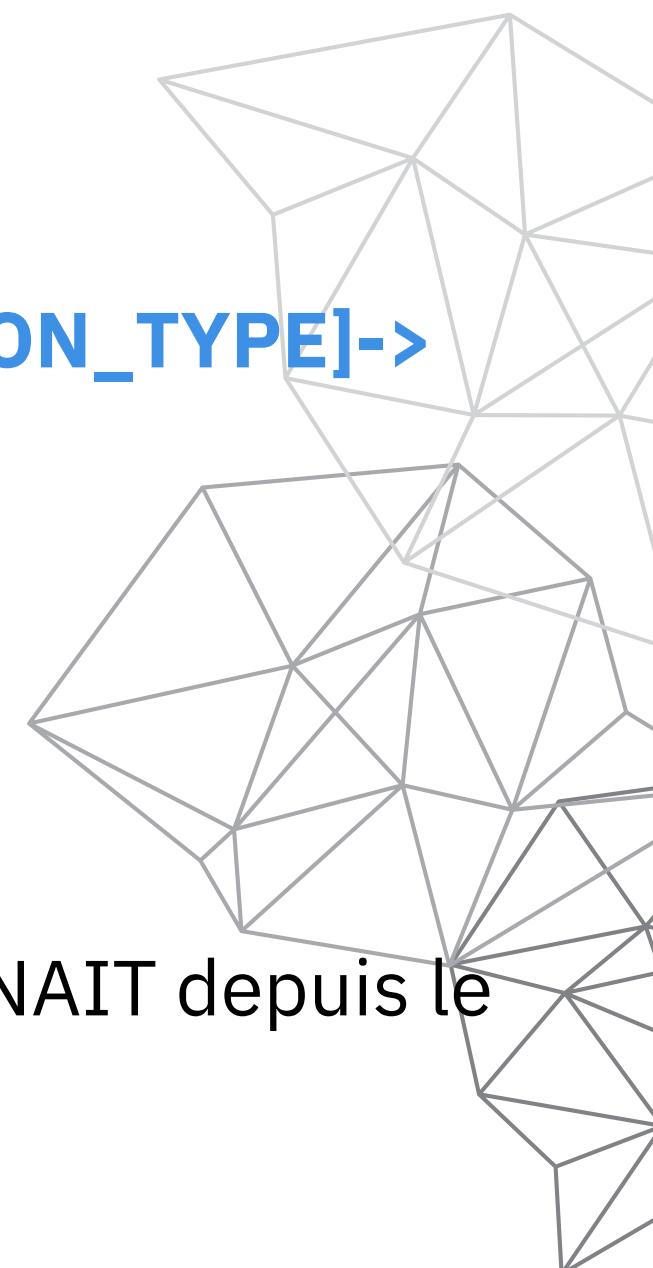
Pour spécifier le type de la relation, il suffit de l'ajouter comme ceci : **-[maRelation:MON_TYPE]->**

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.



SYNTAXE DE CYpher

► Les commandes principales sont :

MATCH : pour rechercher des motifs de graphes.

RETURN : pour retourner les résultats.

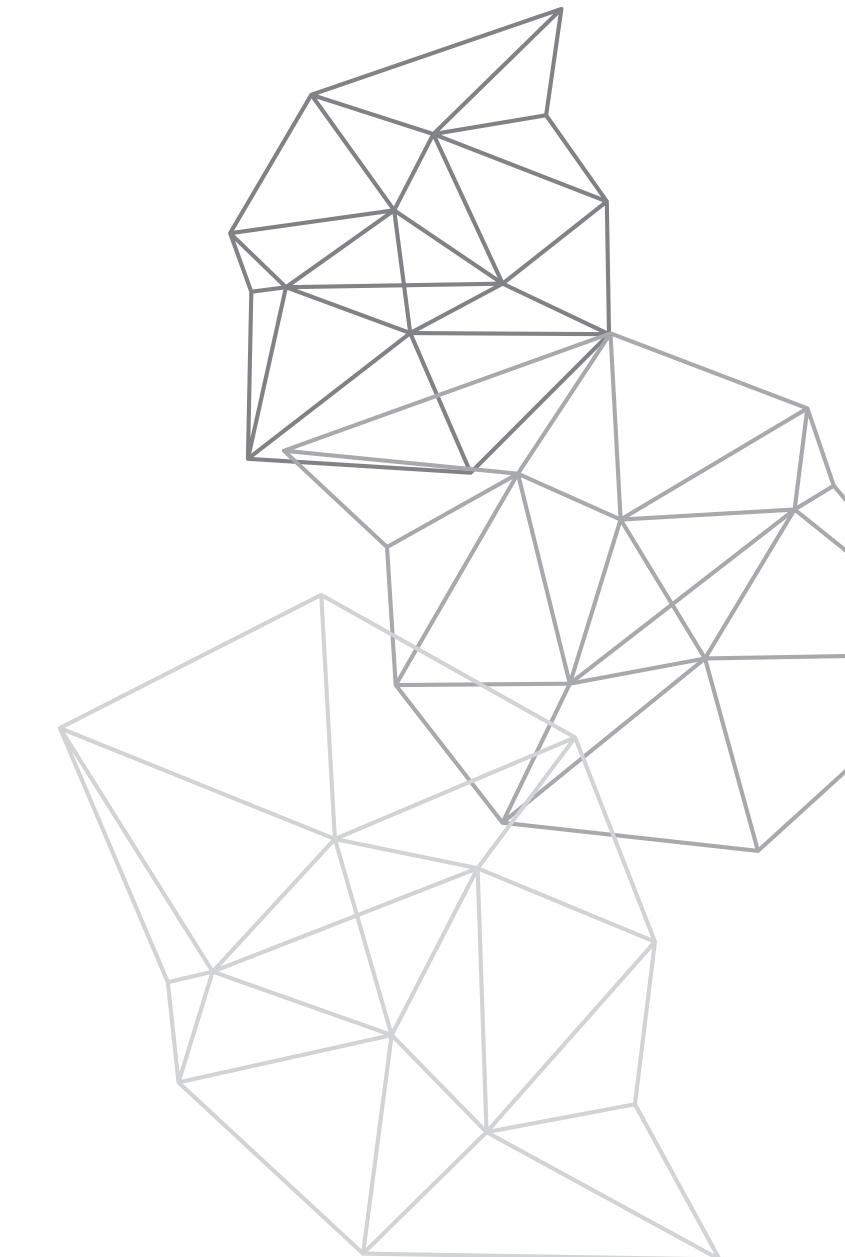
CREATE : pour créer de nouveaux nœuds et relations.

SET : pour mettre à jour des propriétés.

DELETE : pour supprimer des nœuds et relations.

ORDER BY: trie les résultats selon des critères spécifiés.

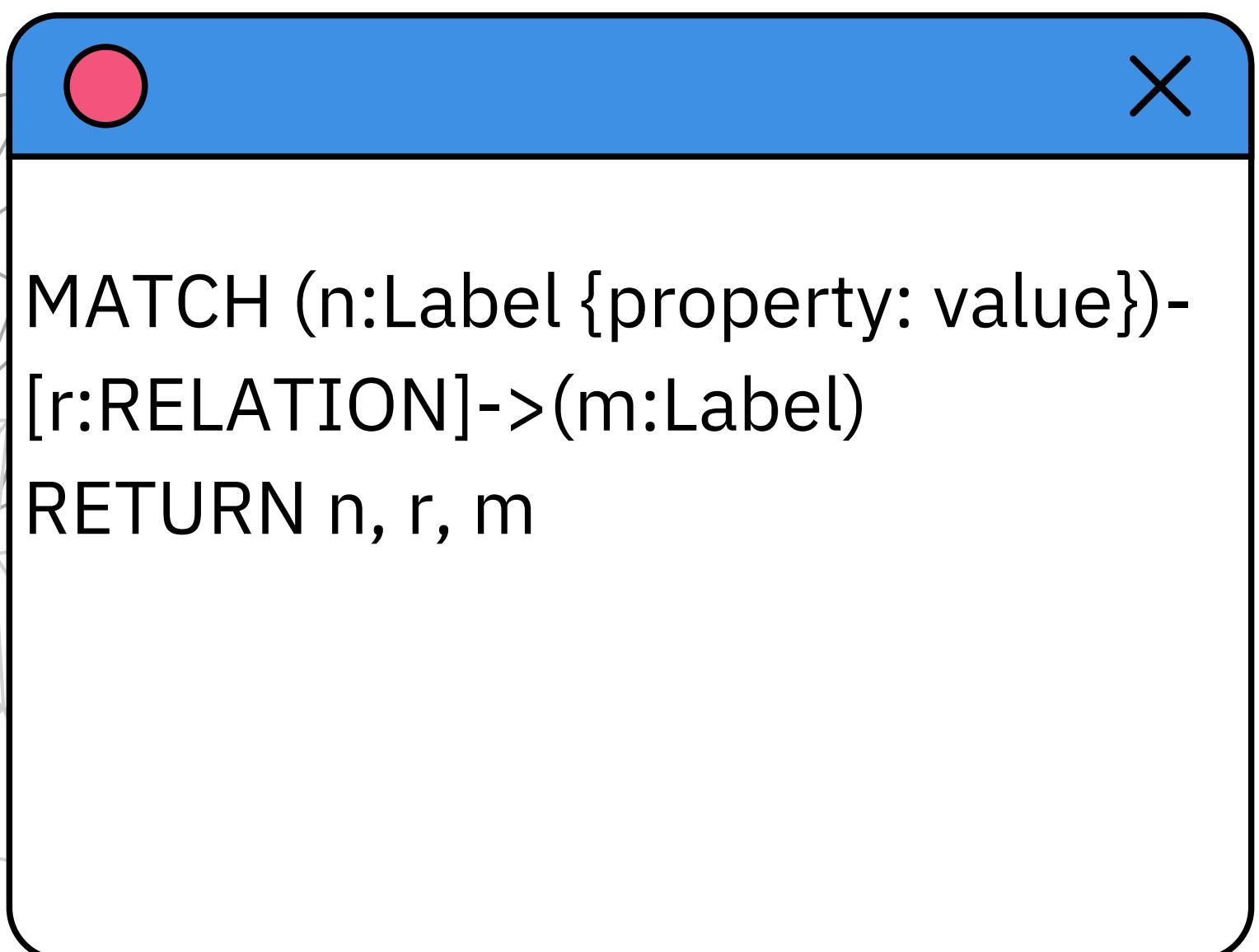
LIMIT: restreint le nombre de résultats retournés.



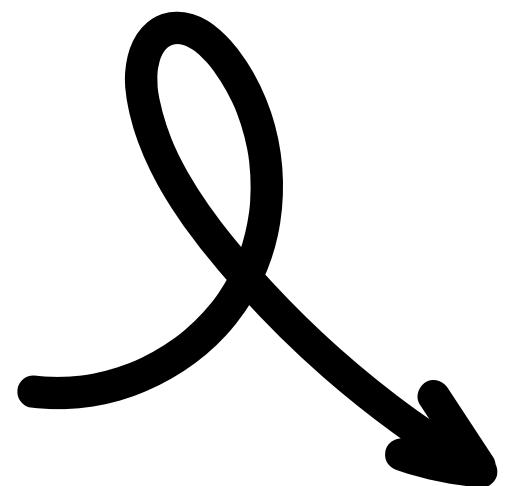
SYNTAXE DE CYpher

► Syntaxe Générale de Cypher

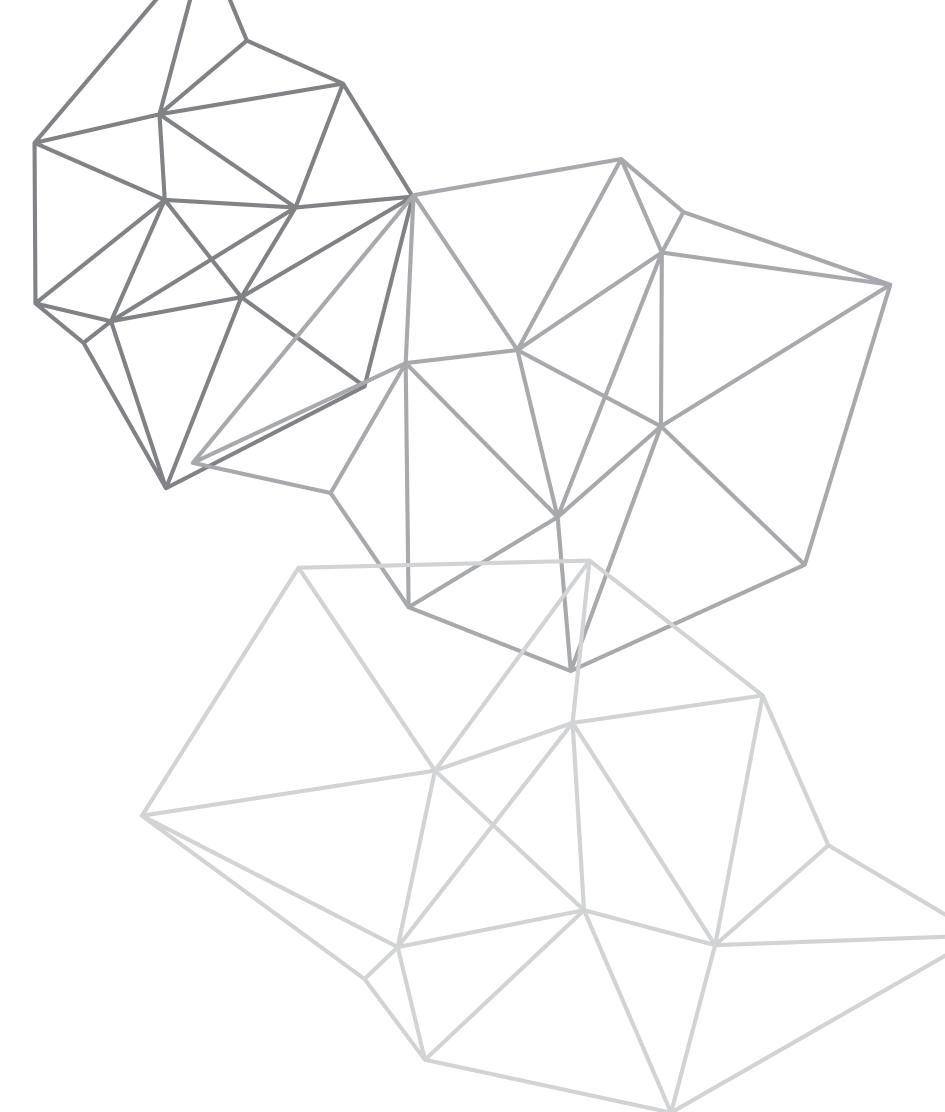
➤ MATCH



```
MATCH (n:Label {property: value})-  
[r:RELATION]->(m:Label)  
RETURN n, r, m
```



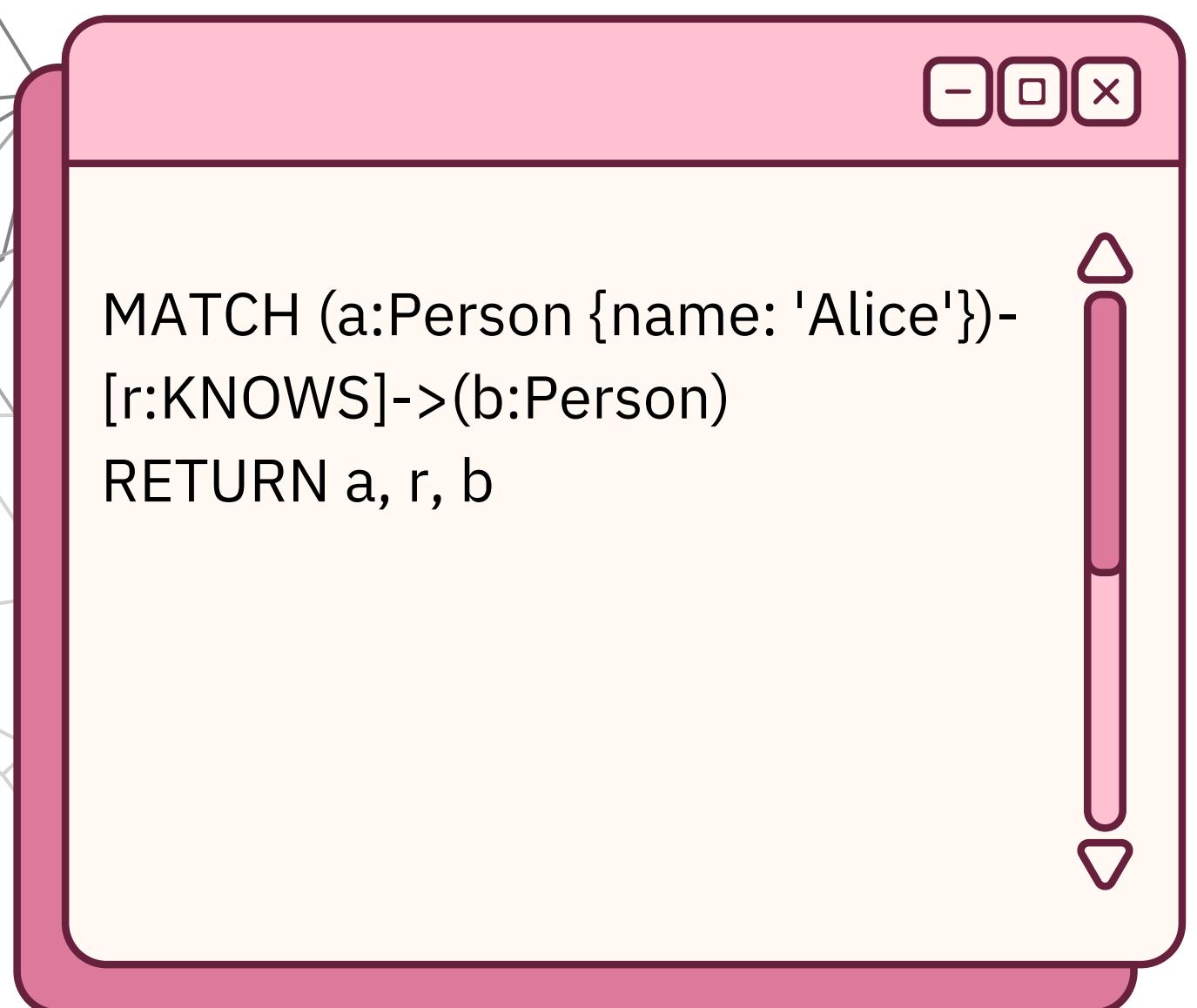
recherche des motifs de graphes spécifiques et retourne les nœuds et les relations correspondants.



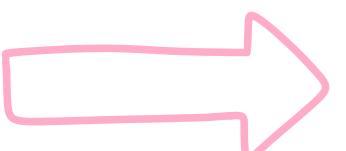
SYNTAXE DE CYpher

► Syntaxe Générale de Cypher

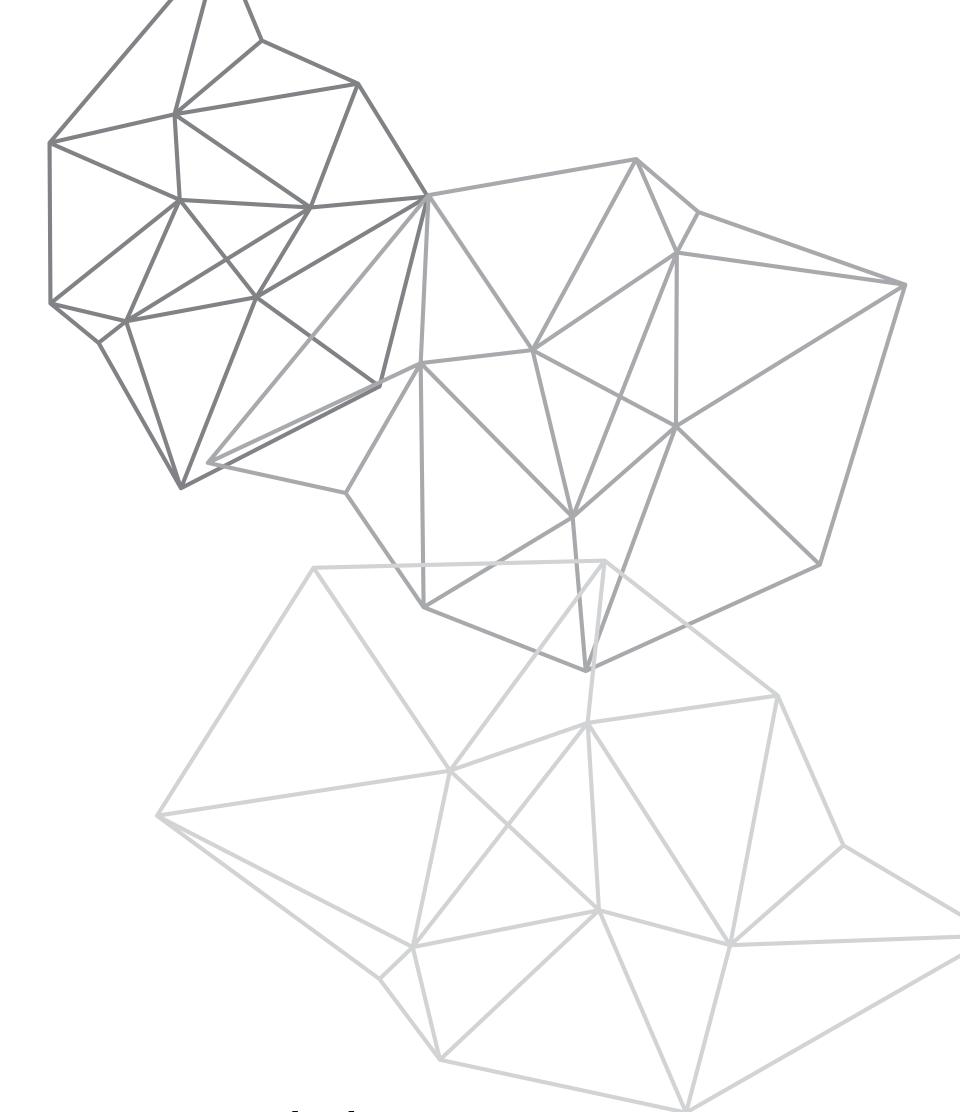
➤ Exemple de MATCH :



```
MATCH (a:Person {name: 'Alice'})-  
[r:KNOWS]->(b:Person)  
RETURN a, r, b
```



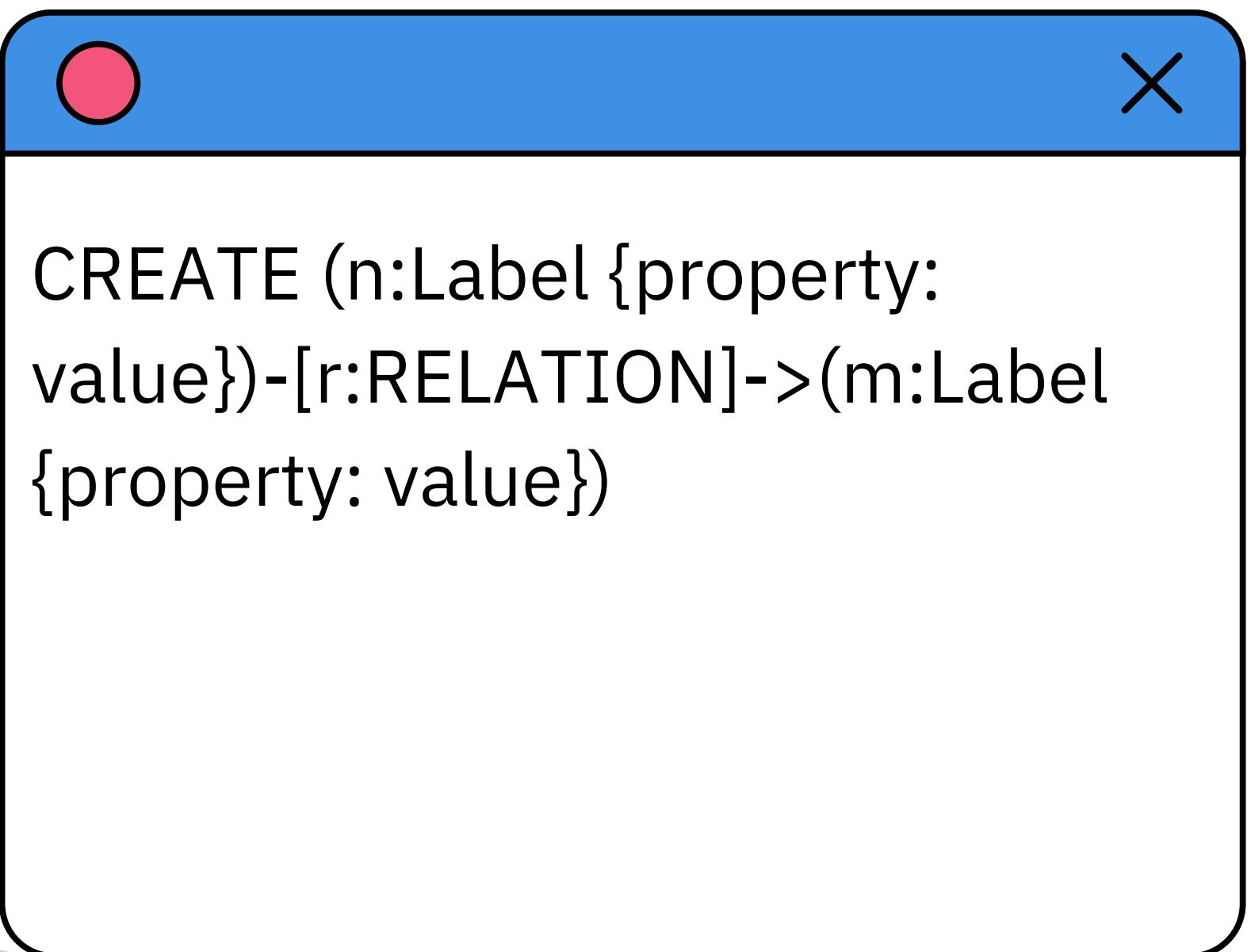
Cette requête recherche un nœud de type Person avec le nom "Alice" et retourne toutes les relations KNOWS entre Alice et d'autres personnes, ainsi que les nœuds connectés.



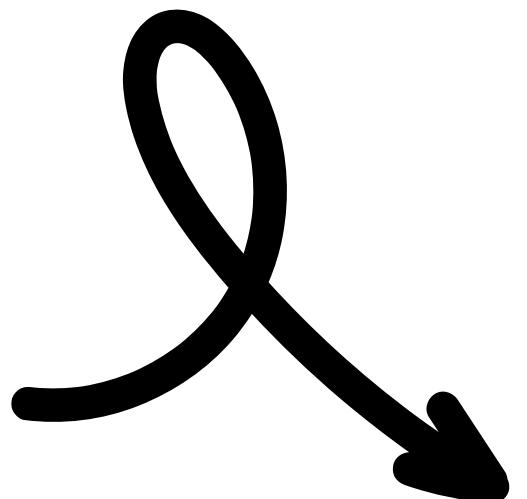
SYNTAXE DE CYpher

► Syntaxe Générale de Cypher

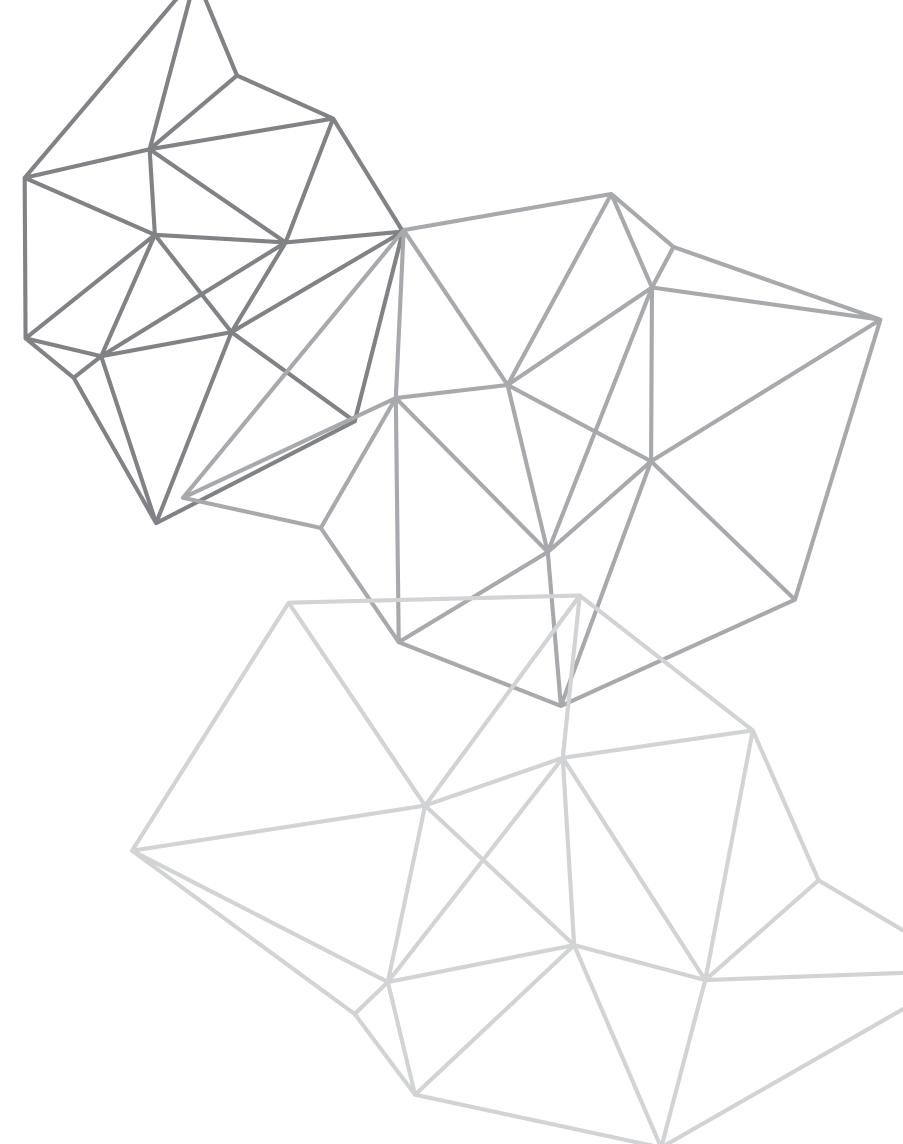
➤ CREATE



```
CREATE (n:Label {property:  
value})-[r:RELATION]->(m:Label  
{property: value})
```

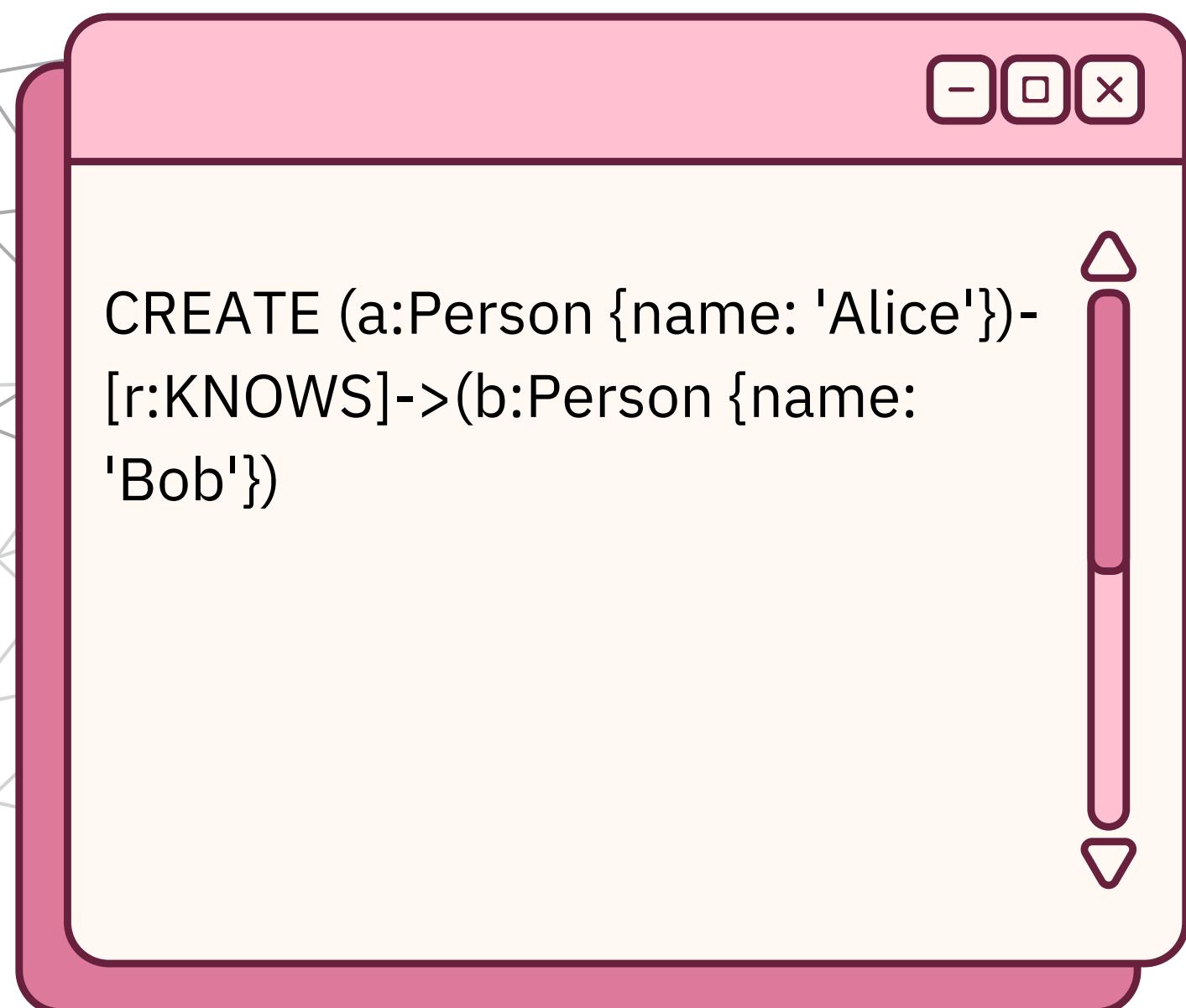


Cela crée de nouveaux nœuds et relations dans la base de données.



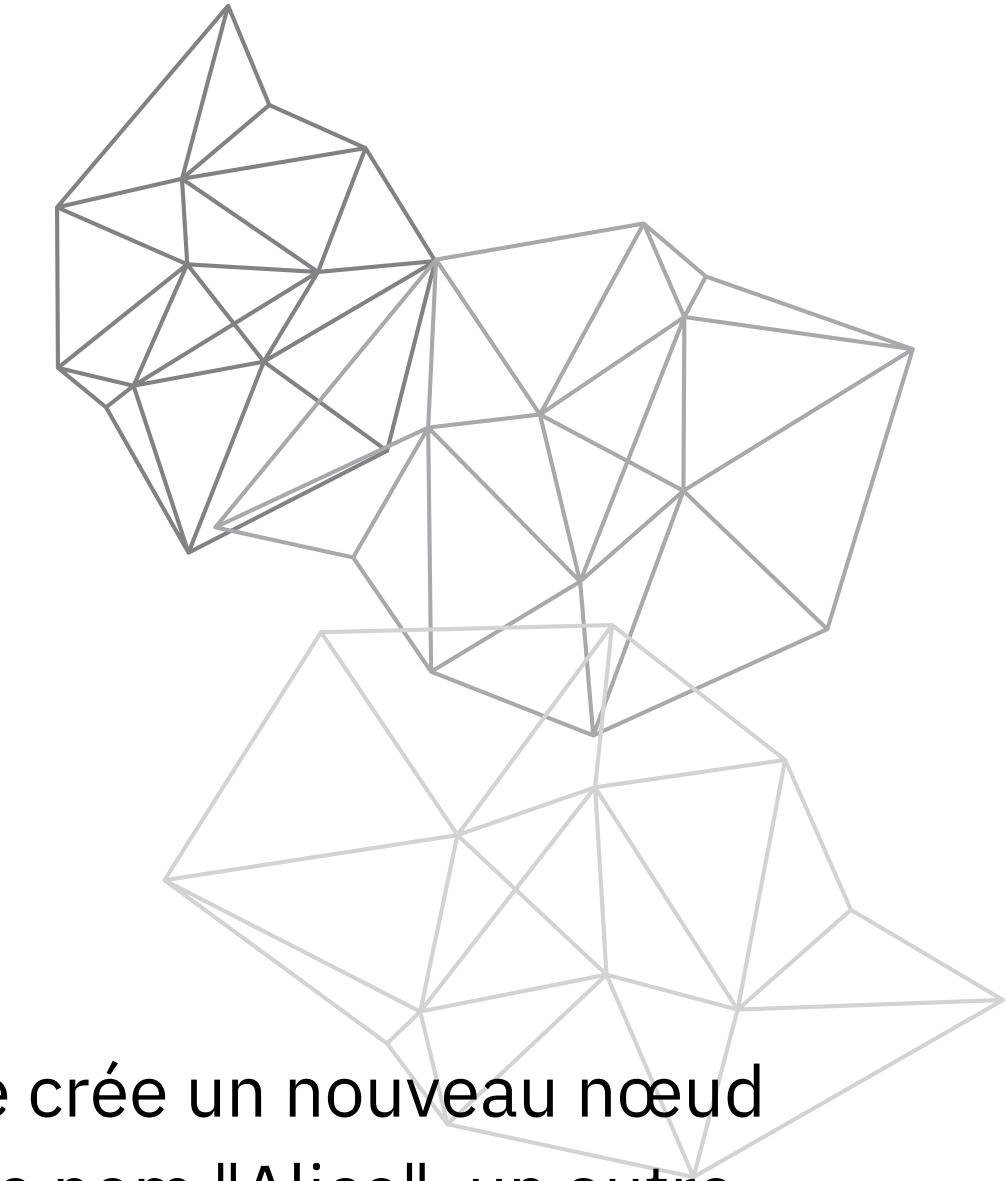
SYNTAXE DE CYpher

- ▶ **Syntaxe Générale de Cypher**
- ▶ **Exemple de CREATE :**



```
CREATE (a:Person {name: 'Alice'})-[r:KNOWS]->(b:Person {name: 'Bob'})
```

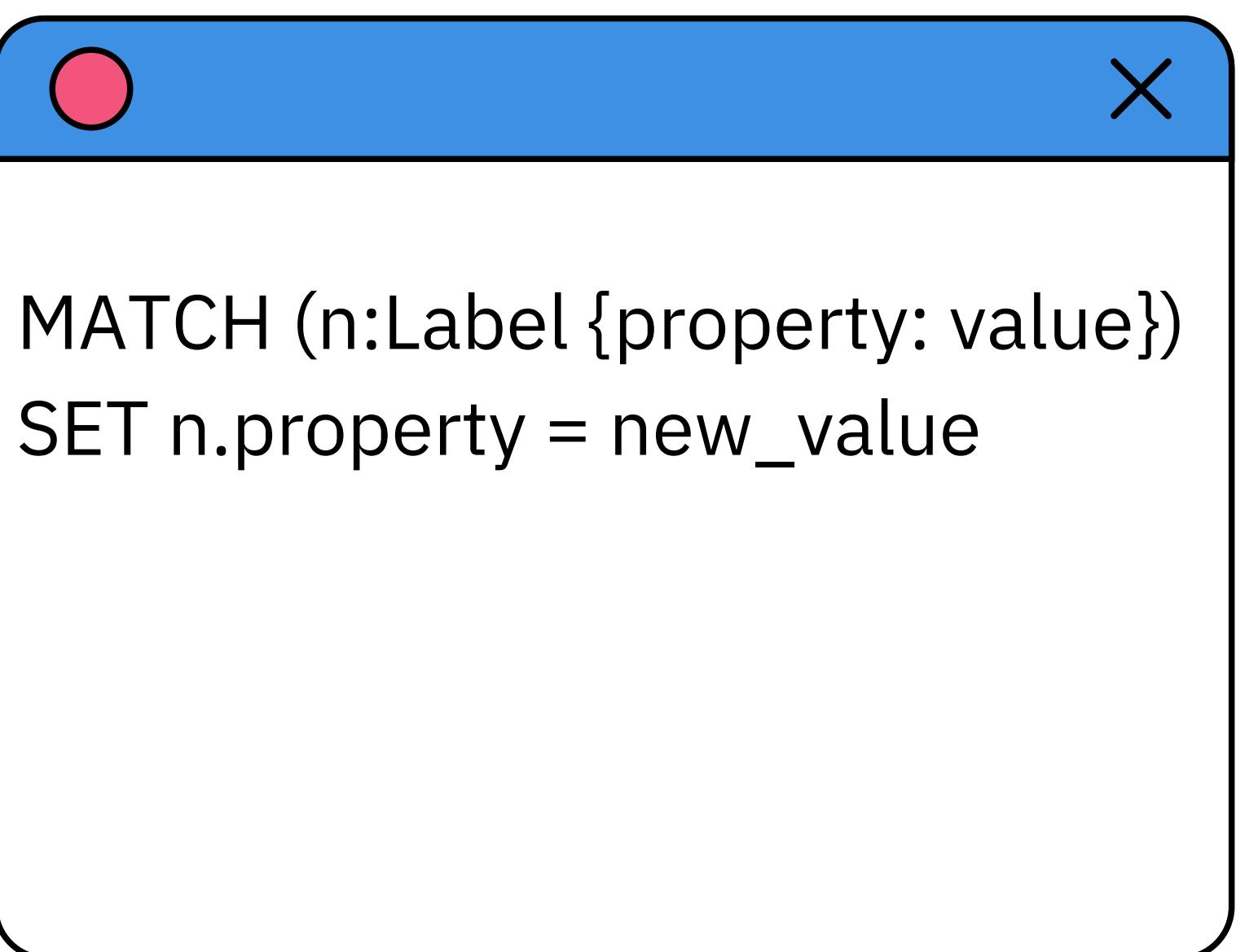
Cette requête crée un nouveau nœud Person avec le nom "Alice", un autre avec le nom "Bob", et une relation KNOWS entre eux.



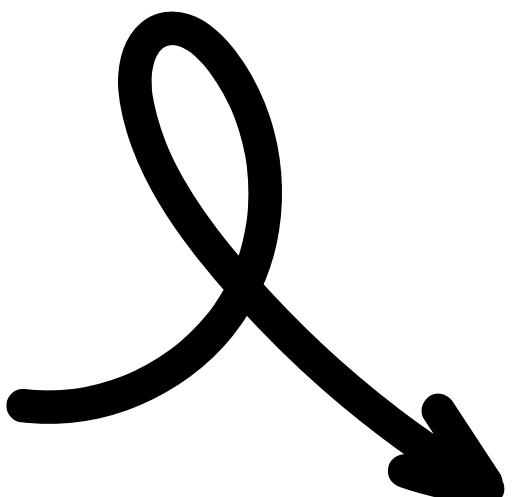
SYNTAXE DE CYpher

► Syntaxe Générale de Cypher

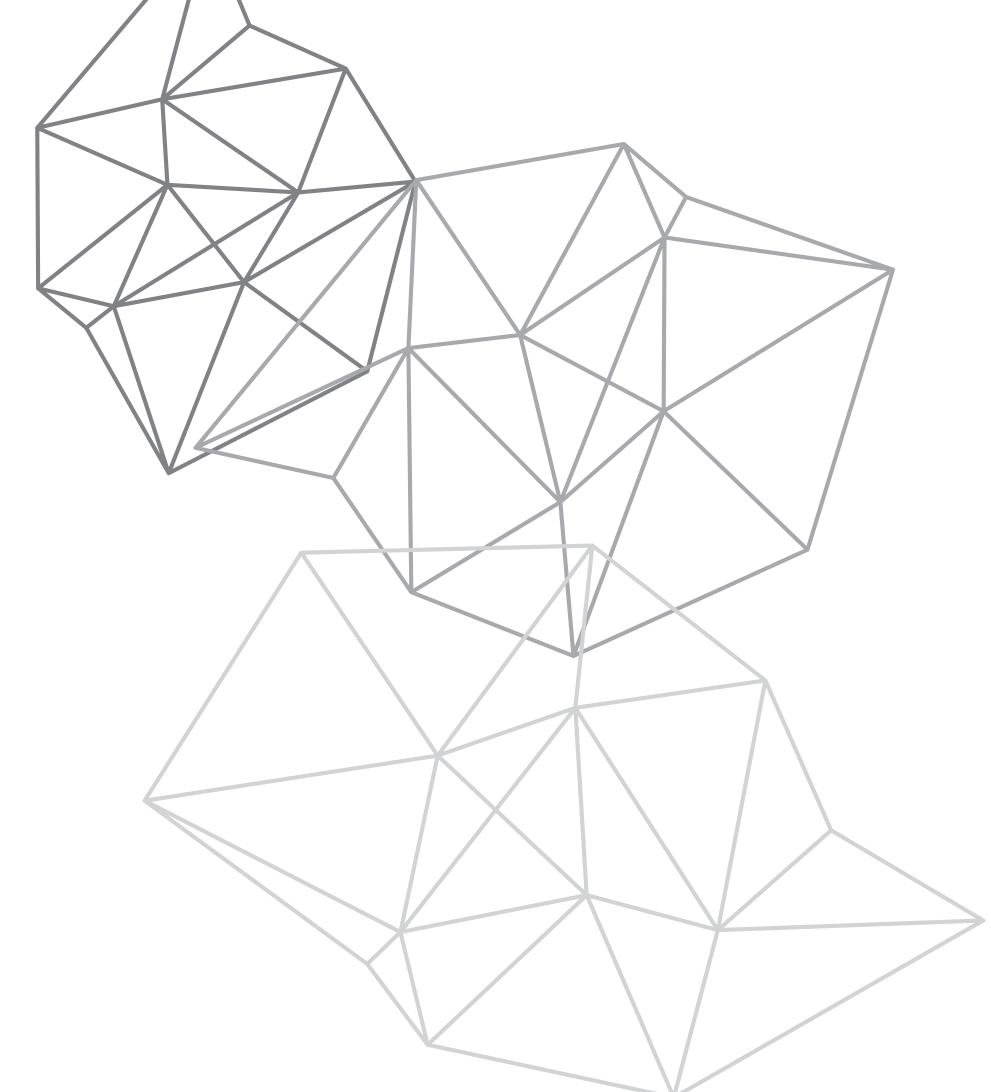
► SET



```
MATCH (n:Label {property: value})
SET n.property = new_value
```



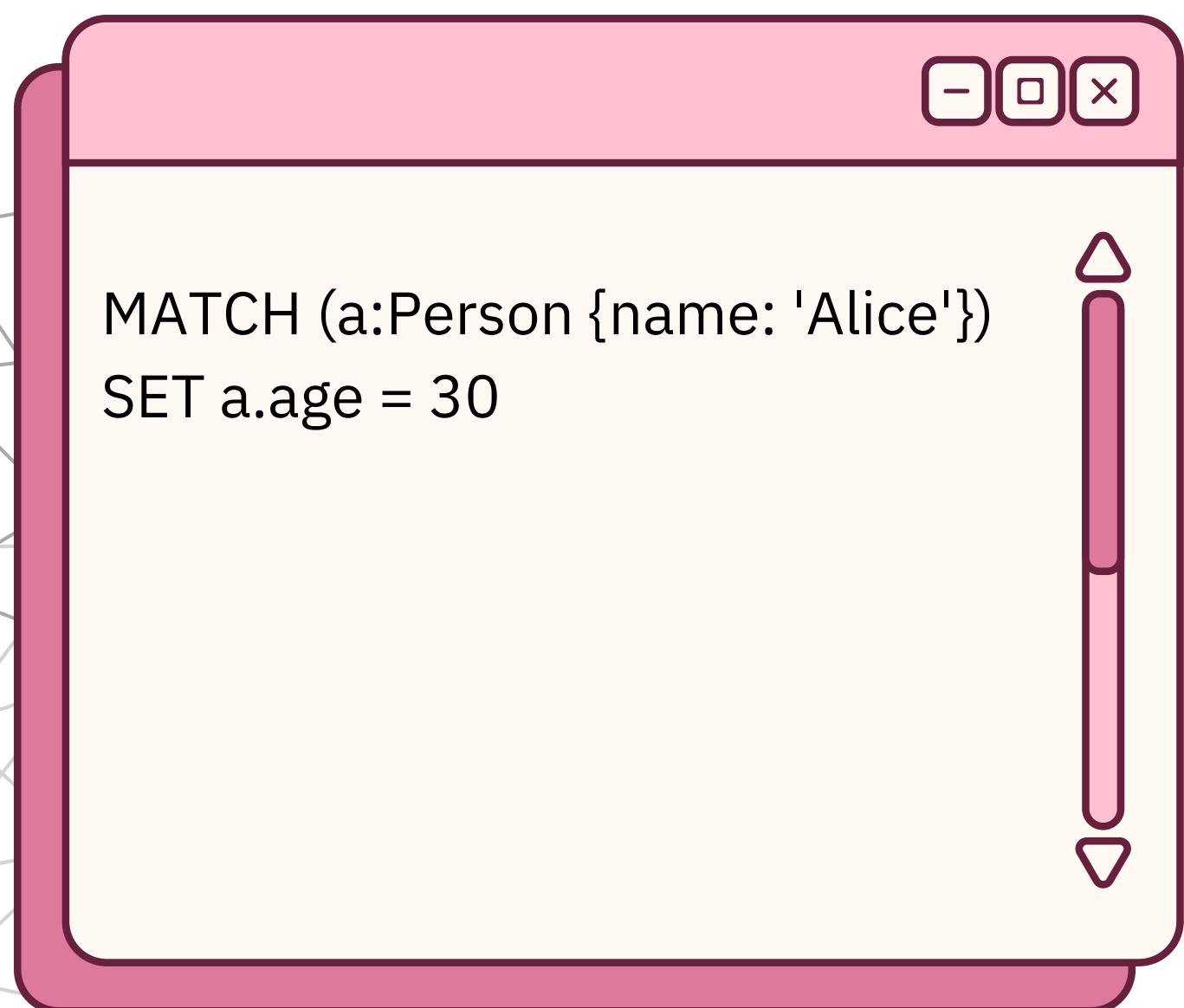
Cette commande met à jour les propriétés d'un nœud existant.



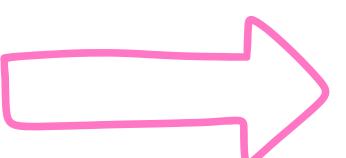
SYNTAXE DE CYpher

► Syntaxe Générale de Cypher

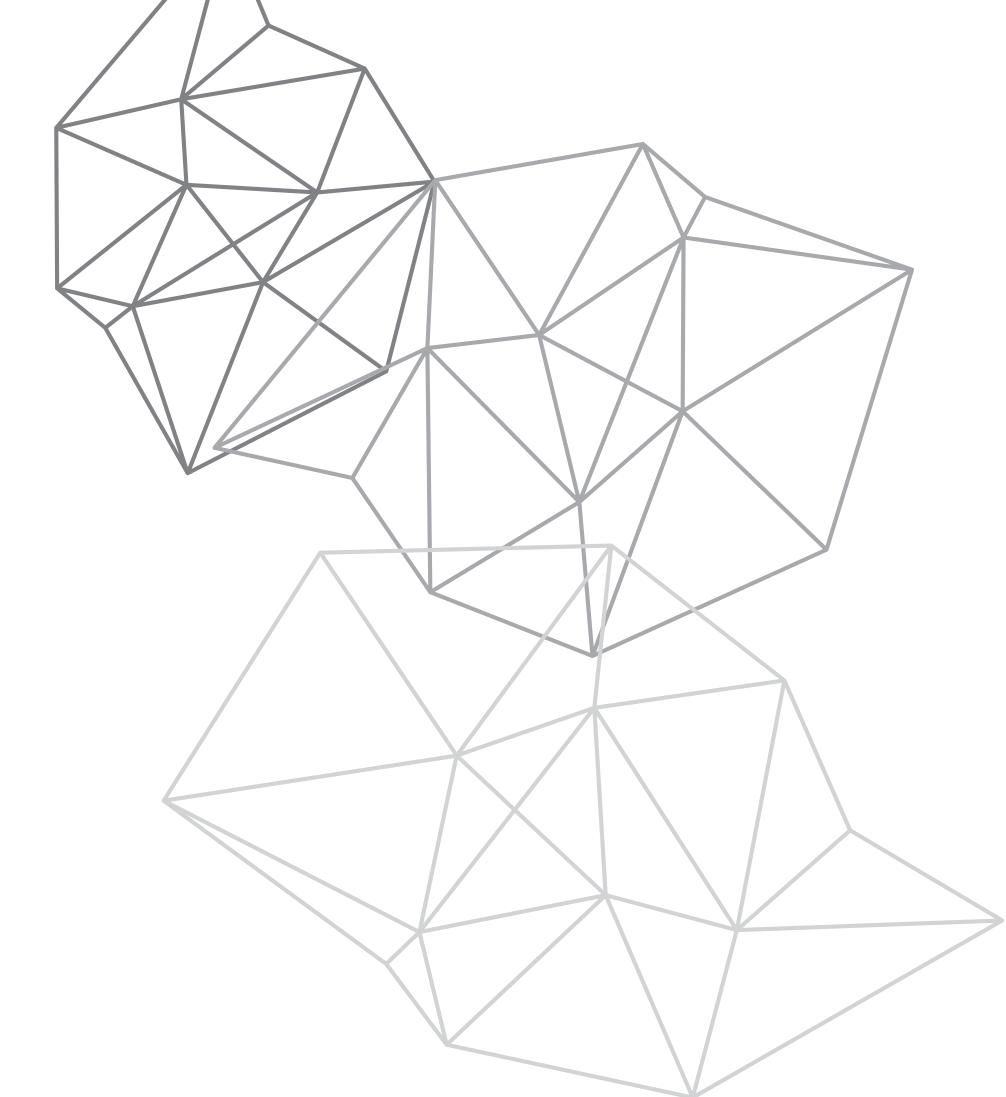
► Exemple de SET



```
MATCH (a:Person {name: 'Alice'})  
SET a.age = 30
```



Cette requête met à jour la propriété age du nœud Person nommé "Alice" pour la définir à 30 ans.

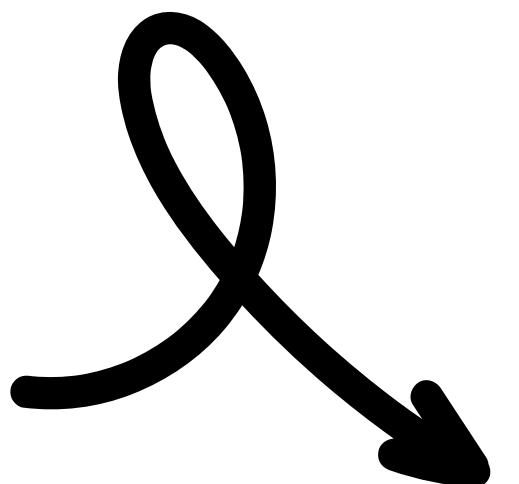


SYNTAXE DE CYpher

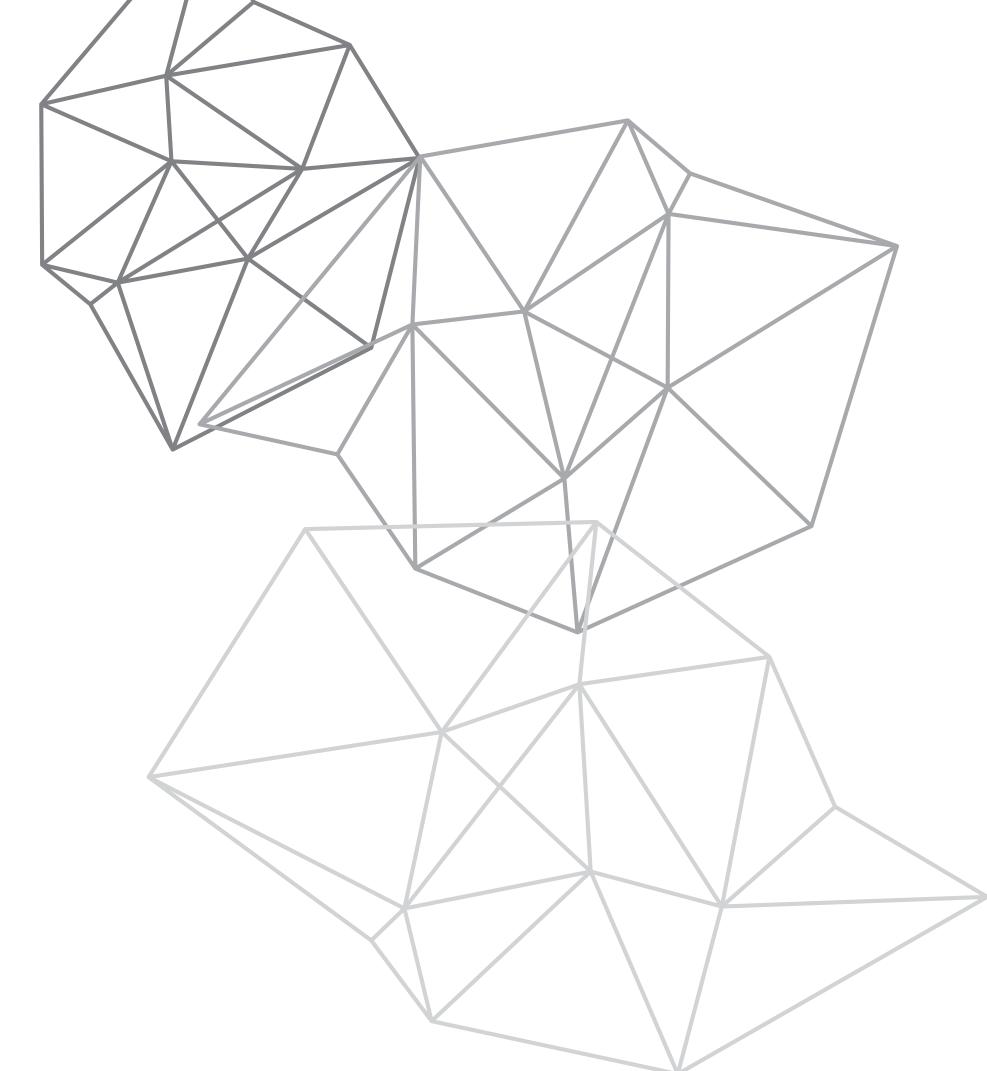
► Syntaxe Générale de Cypher

➤ **DELETE**

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



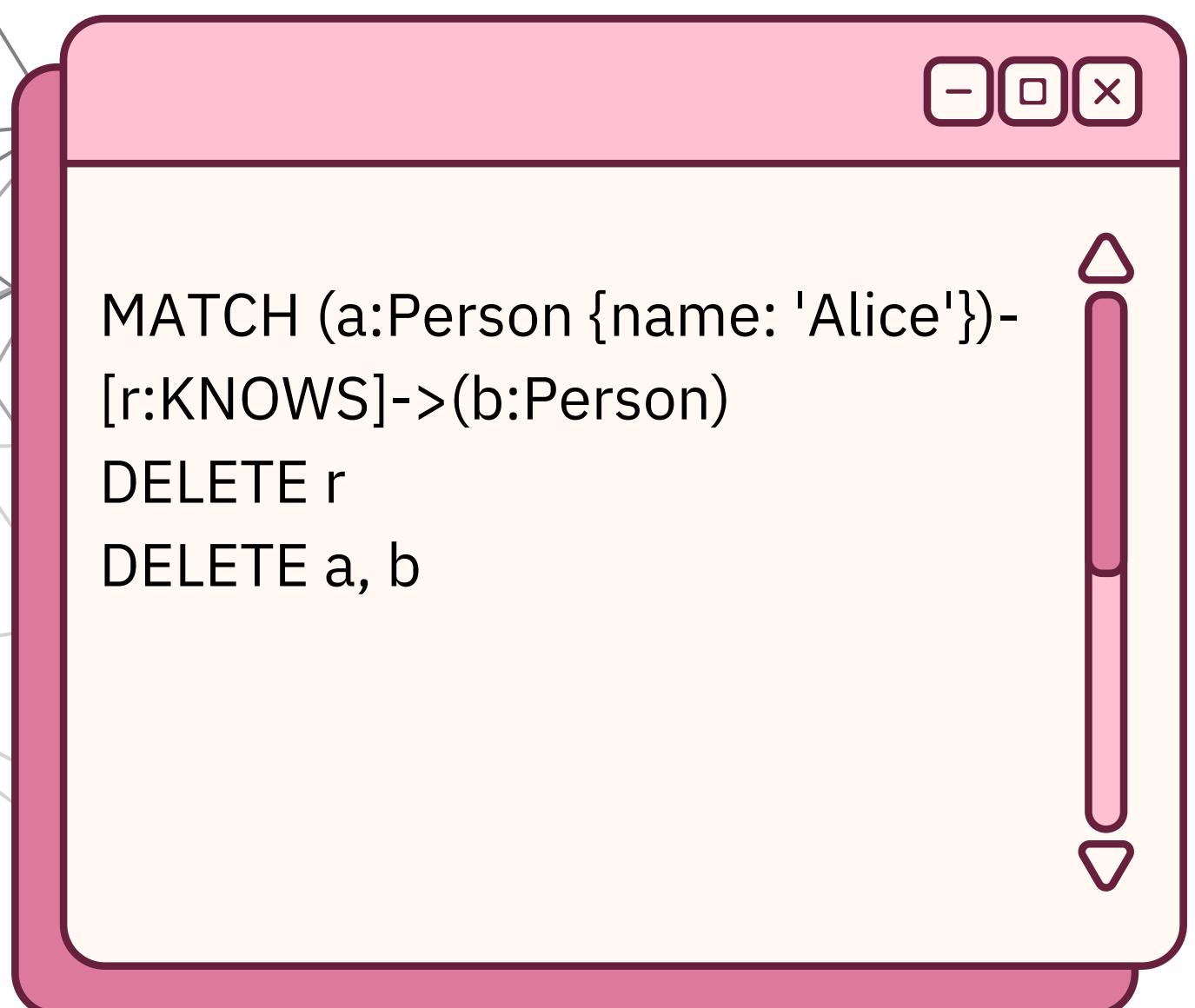
Ceci supprime les relations entre les nœuds, puis les nœuds eux-mêmes.



SYNTAXE DE CYpher

► Syntaxe Générale de Cypher

► Exemple de DELETE

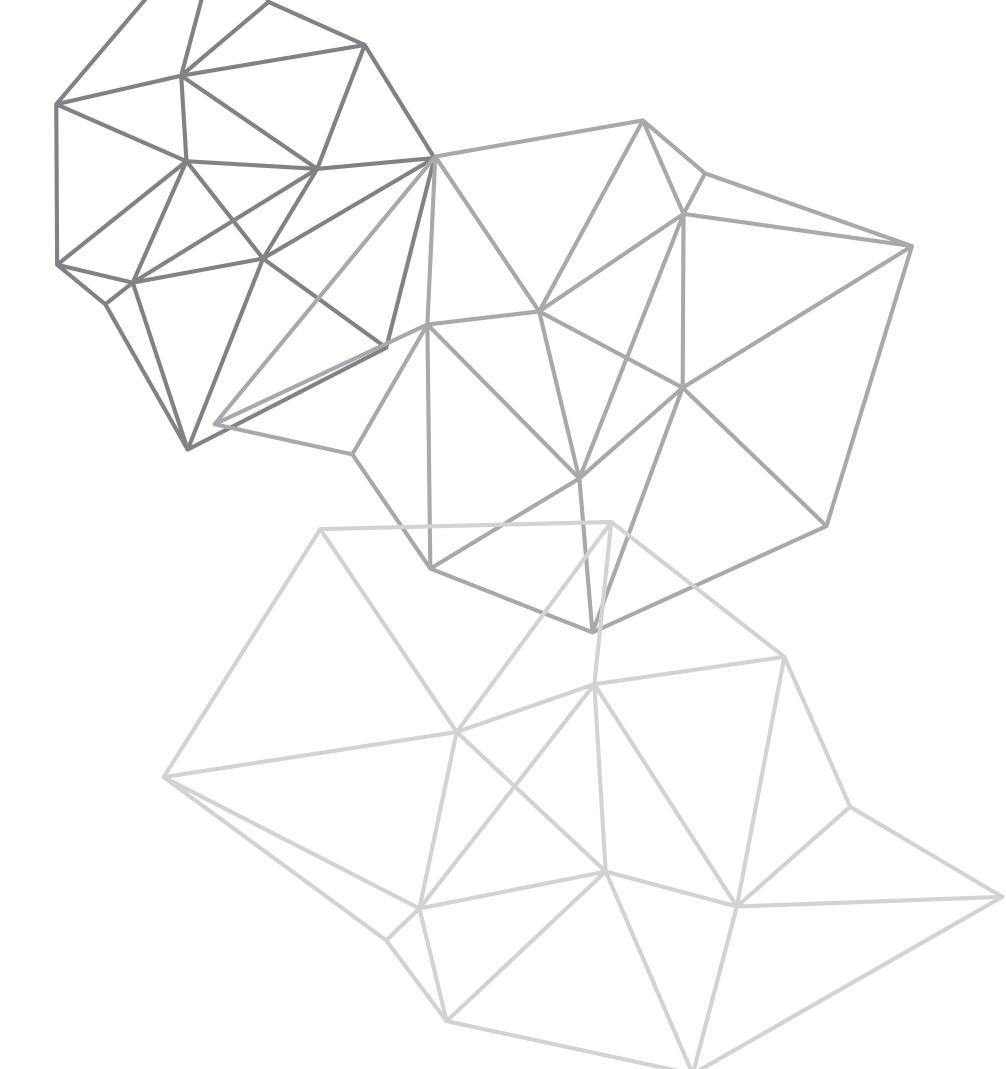


A screenshot of a Cypher editor window. The window has a pink header bar with three icons: a minus sign, a square, and an 'X'. The main area contains the following Cypher code:

```
MATCH (a:Person {name: 'Alice'})-  
[r:KNOWS]->(b:Person)  
DELETE r  
DELETE a, b
```

The code is highlighted with a vertical pink selection bar on the right side. A pink arrow points from the explanatory text below to this selection bar.

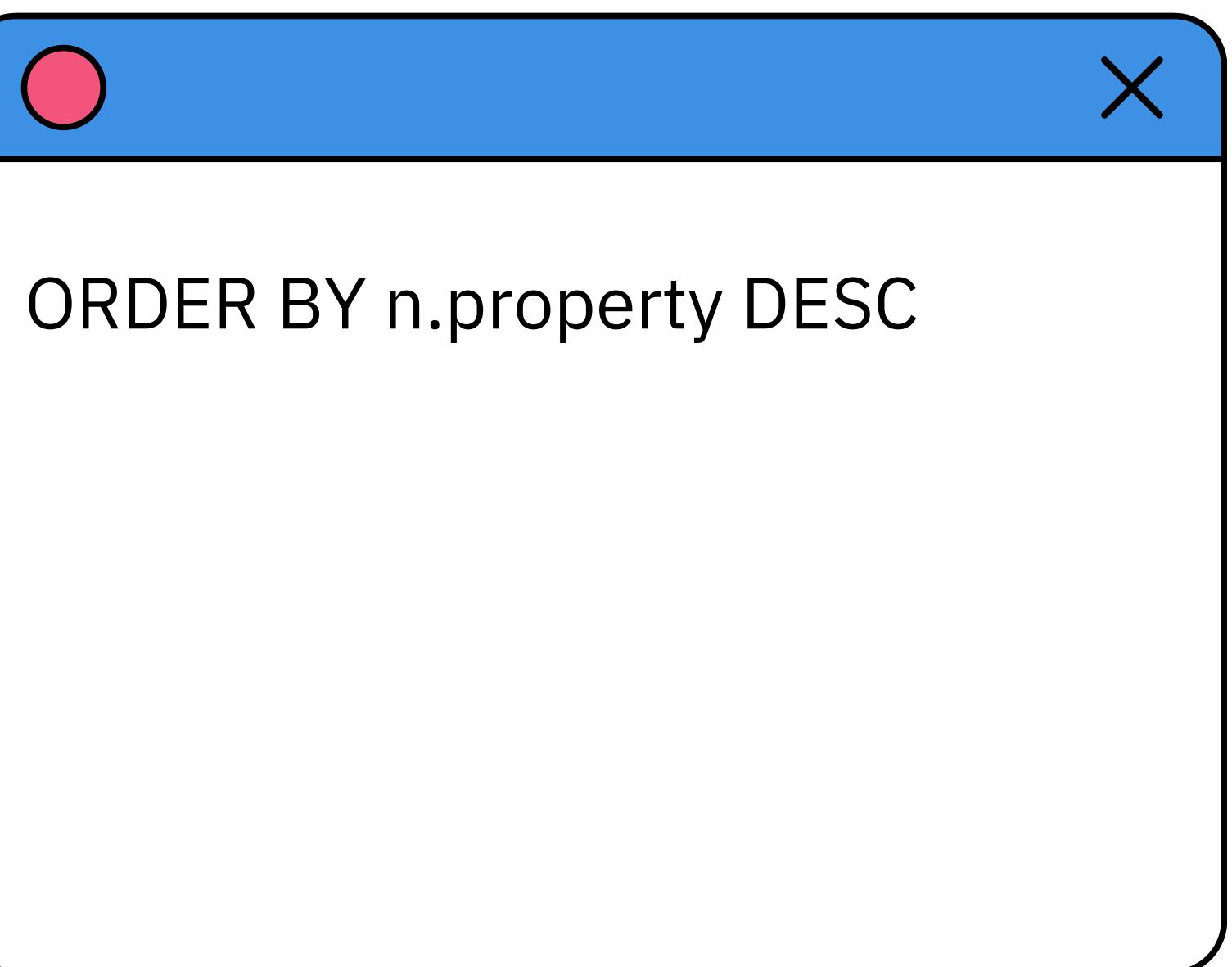
Cette requête supprime la relation KNOWS entre Alice et une autre personne, puis supprime les nœuds eux-mêmes.



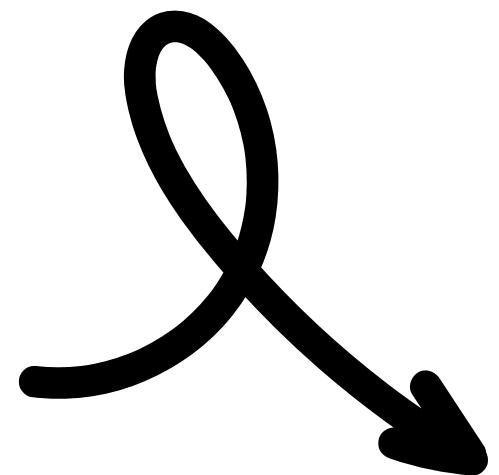
SYNTAXE DE CYpher

► Syntaxe Générale de Cypher

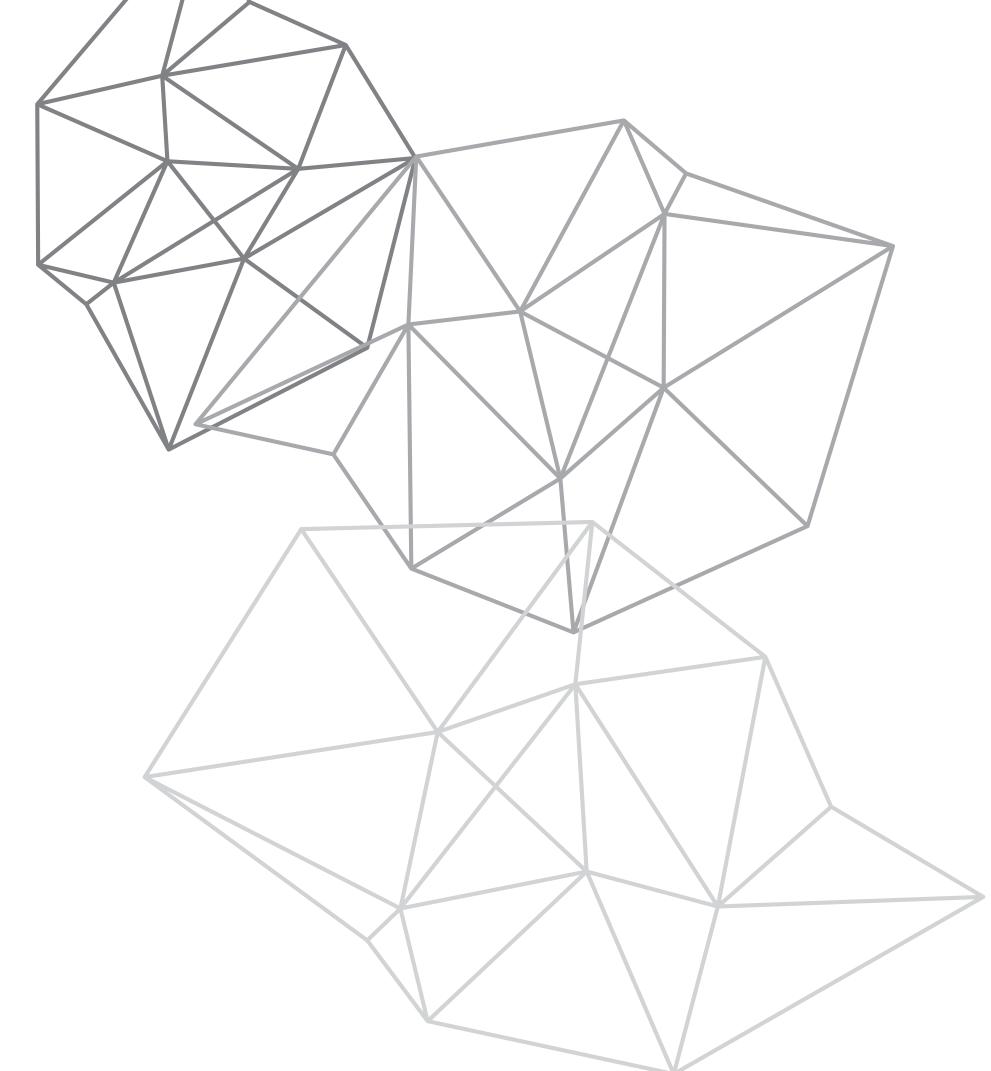
➤ ORDER BY



```
ORDER BY n.property DESC
```



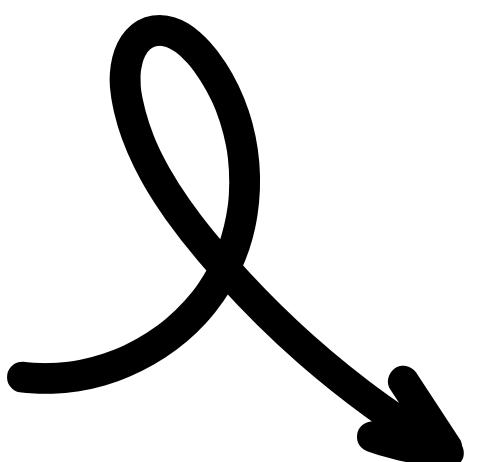
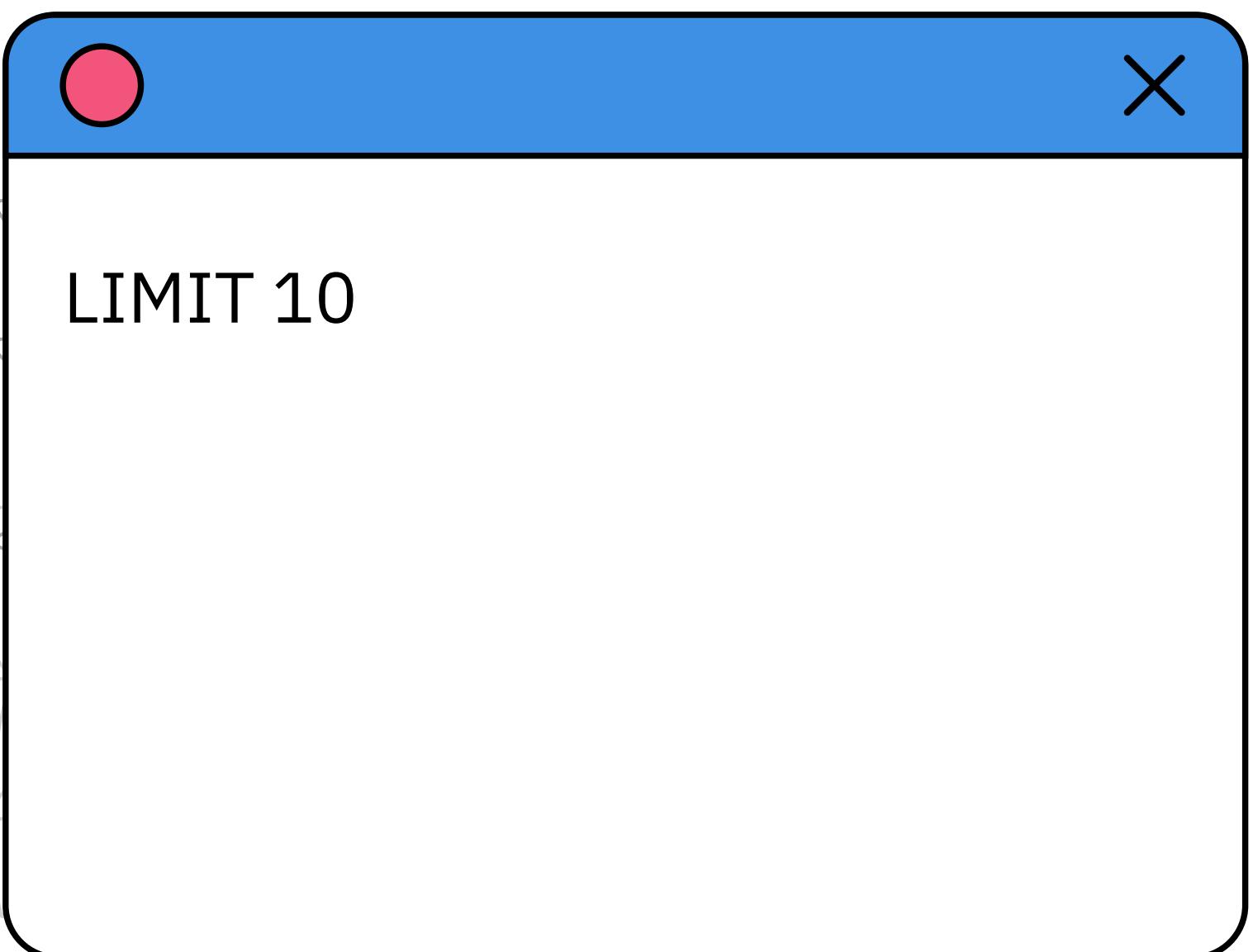
Cela trie les résultats en ordre décroissant selon une propriété.



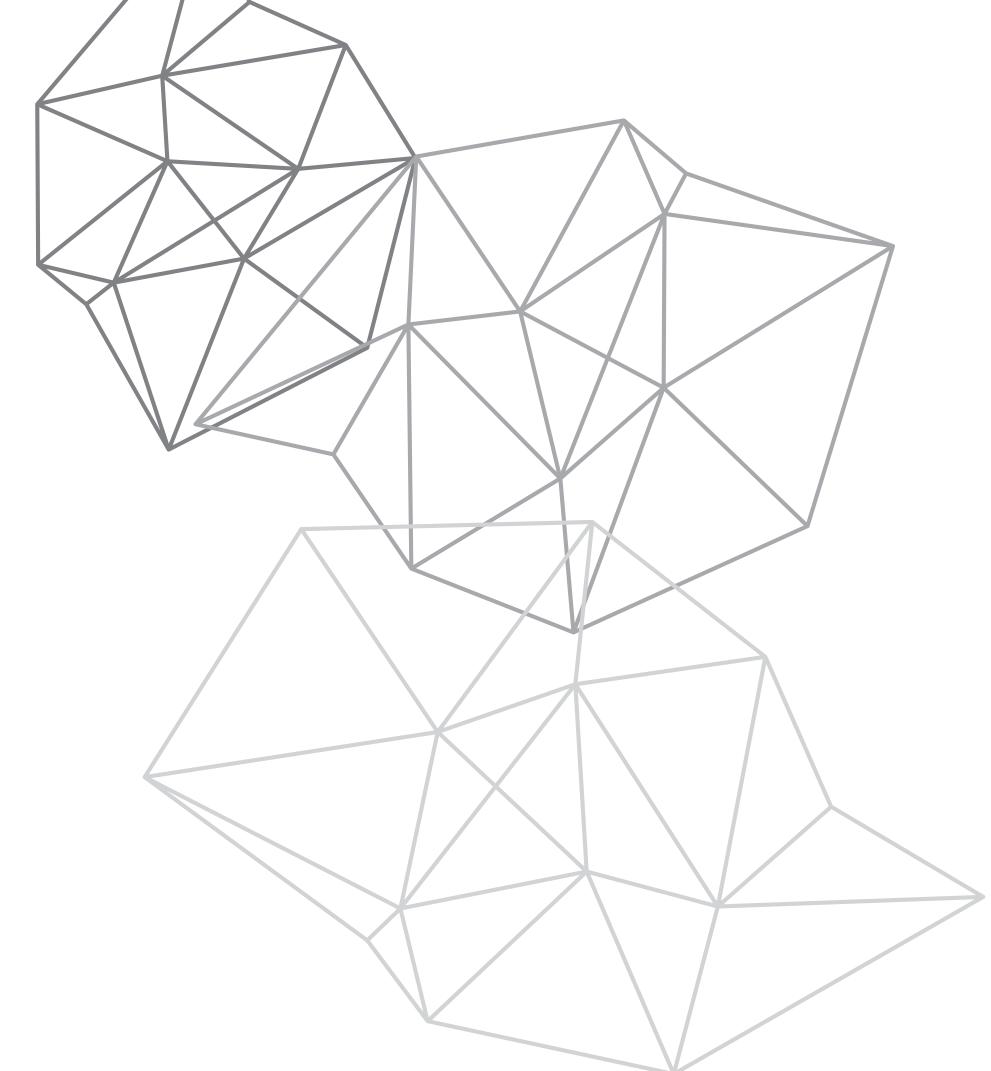
SYNTAXE DE CYpher

► Syntaxe Générale de Cypher

LIMIT



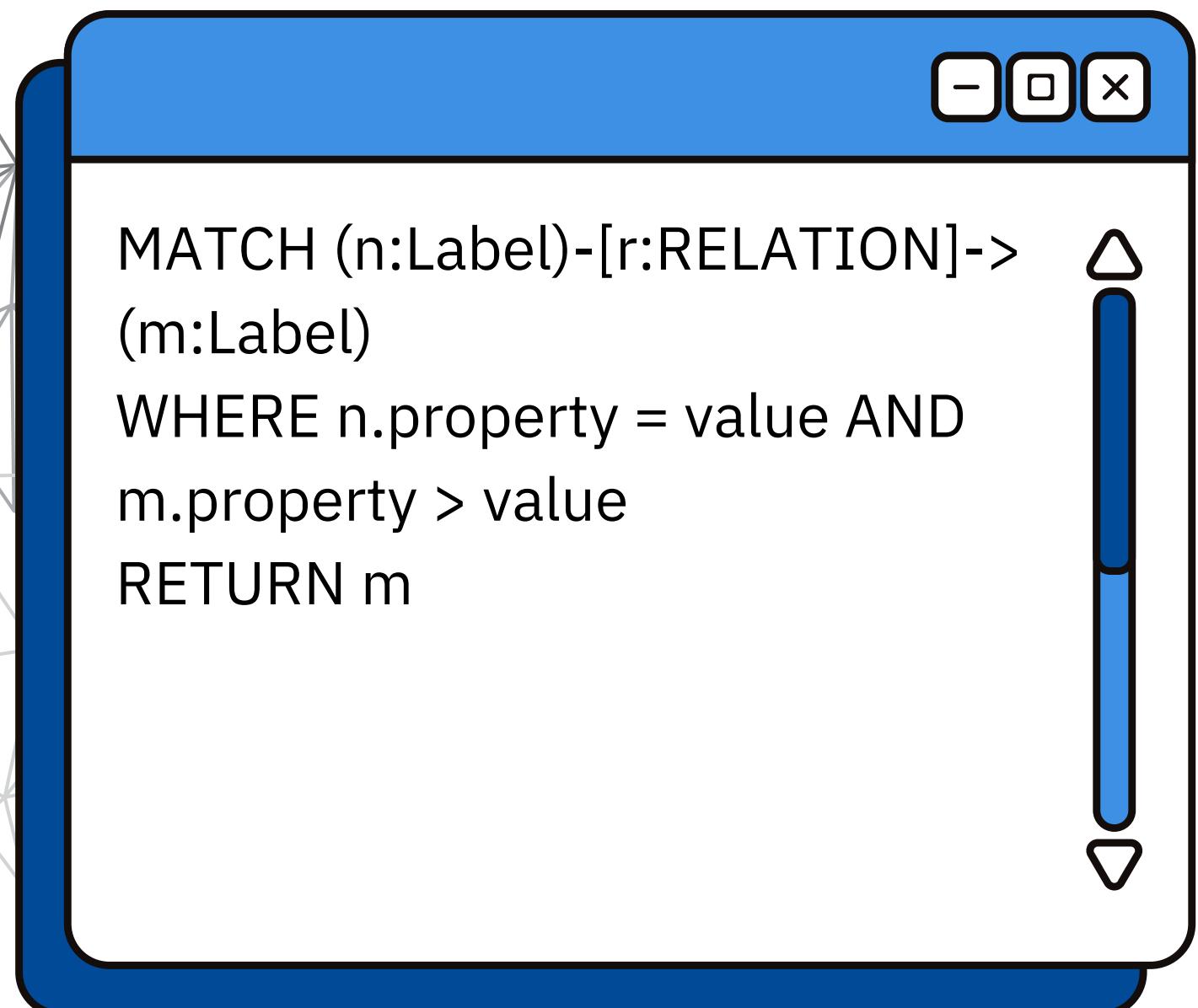
Cela retourne uniquement les 10 premiers résultats.



SYNTAXE DE CYPHER

► Fonctions Avancées

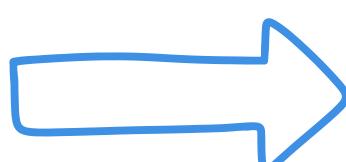
► utilisation de filtres avec WHERE



The screenshot shows a blue-themed web application window with three control buttons (-, square, X) at the top right. Inside, there is a code editor area containing the following Cypher query:

```
MATCH (n:Label)-[r:RELATION]->(m:Label)
WHERE n.property = value AND
      m.property > value
RETURN m
```

A vertical scrollbar is visible on the right side of the code editor.

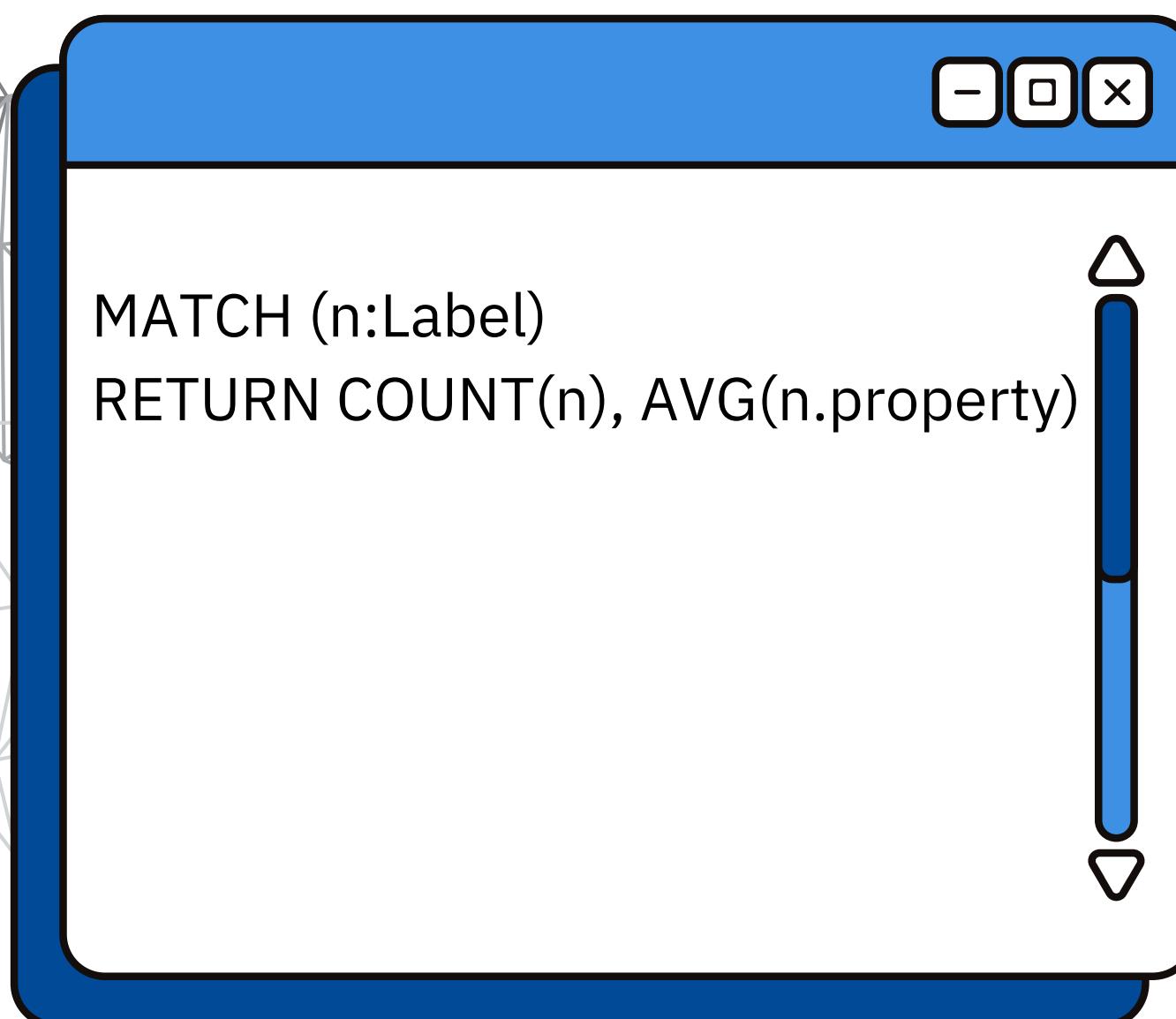


Cette requête recherche des nœuds n de type Label reliés par une relation RELATION à des nœuds m de type Label, où n a une propriété égale à une certaine valeur et m a une propriété supérieure à une autre valeur. Les nœuds m correspondant aux critères sont retournés.

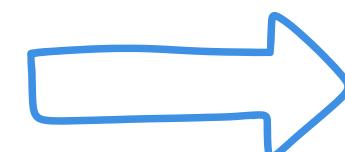
SYNTAXE DE CYPHER

► Fonctions Avancées

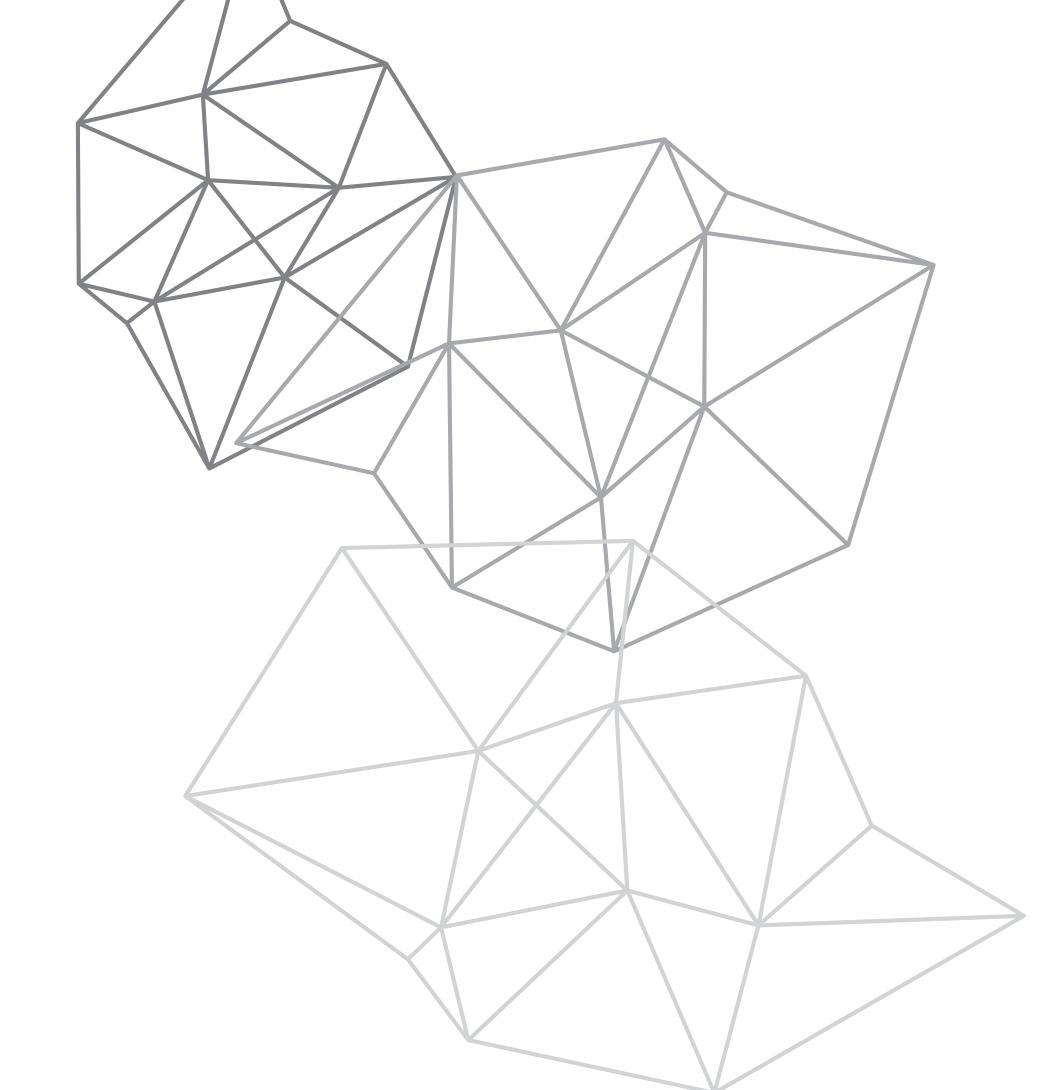
➤ Fonctions d'Aggrégation comme COUNT et AVG



```
MATCH (n:Label)
RETURN COUNT(n), AVG(n.property)
```



Cette requête sélectionne tous les nœuds n de type Label et **retourne** le nombre total de ces nœuds ainsi que la moyenne d'une de leurs propriétés.



INSTALLATION

- Options d'installation

OPTIONS D'INSTALLATION

► Installation locale :

Neo4j Desktop :

Télécharger et installer Neo4j Desktop pour une utilisation locale. Idéal pour le développement et les tests, offrant une interface conviviale pour interagir avec les bases de données.

Neo4j Community Edition :

Utiliser la version gratuite de Neo4j pour des projets de petite à moyenne envergure. Fournit toutes les fonctionnalités de base nécessaires pour explorer et développer des bases de données graphes.



OPTIONS D'INSTALLATION

► **Déploiement cloud :**

Neo4j Aura :

Utiliser Neo4j Aura pour des déploiements scalables et gérés. Solution cloud éliminant les soucis de maintenance et de gestion, permettant une mise à l'échelle facile selon les besoins du projet.

Déploiements sur AWS, Azure, et GCP :

Déployer Neo4j sur des infrastructures cloud populaires comme AWS, Azure, et Google Cloud Platform. Offrent flexibilité et robustesse, adaptées aux entreprises de toutes tailles.

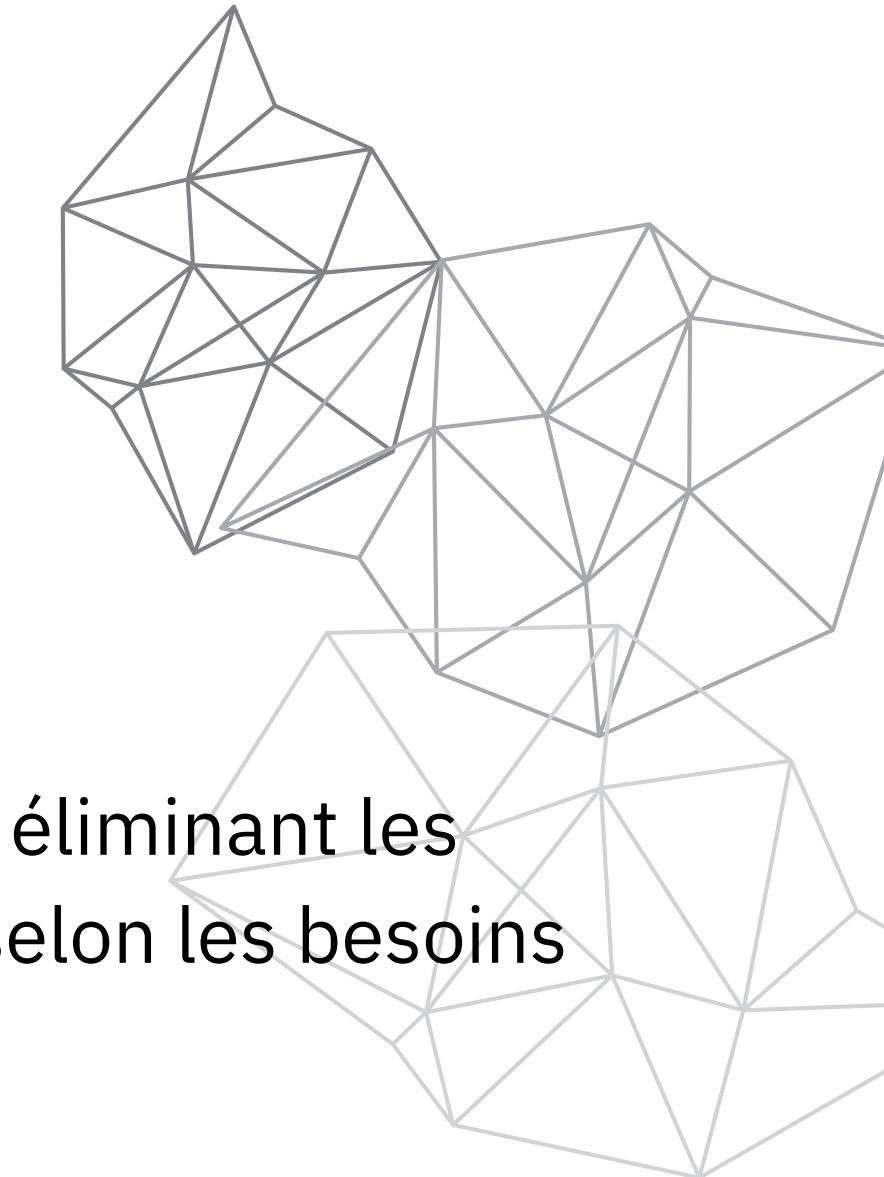


SCHÉMA ET CONTRAINTES DANS NEO4J

- Modélisation de données dans Neo4j
- Contraintes de données
- Filtrage et aggregation

MODÉLISATION DE DONNÉES DANS NEO4J

● Analyse du domaine

Commencez par une analyse approfondie de votre domaine d'application pour identifier les entités principales et leurs relations. Cela implique de comprendre les objets, les concepts et les interactions qui sont significatifs dans votre système.

● Identification des entités

Une fois que vous avez une vision claire de votre domaine, identifiez les entités principales qui jouent un rôle central. Ce sont généralement des personnes, des objets, des événements ou des concepts qui ont une existence distincte et des caractéristiques uniques.

MODÉLISATION DE DONNÉES DANS NEO4J

● **Représentation des entités sous forme de nœuds**

Chaque entité identifiée peut être représentée sous forme de nœud dans le graphe de données. Les nœuds servent de points de données et peuvent contenir des propriétés pour décrire les caractéristiques de chaque entité. Par exemple, un nœud représentant une personne peut avoir des propriétés telles que le nom, l'âge, le sexe, etc.

● **Identification des relations**

Ensuite, identifiez les relations entre les entités. Les relations décrivent les liens ou les connexions entre les entités et peuvent être unidirectionnelles ou bidirectionnelles. Par exemple, dans un réseau social, une relation "AMI" peut relier deux nœuds représentant des personnes.

MODÉLISATION DE DONNÉES DANS NEO4J

● **Représentation des relations**

Les relations sont représentées par des arêtes (ou des liens) entre les nœuds correspondants dans le graphe. Les arêtes peuvent également contenir des propriétés pour décrire la nature ou les attributs de la relation. Par exemple, une relation "TRAVAILLE_AVEC" entre deux nœuds représentant des employés peut avoir une propriété indiquant la date de début de la collaboration.

CONTRAINTES DE DONNÉES

Les contraintes de données dans Neo4j jouent un rôle crucial pour garantir l'intégrité des données et maintenir la cohérence du modèle de données.

● **Unicité des nœuds et des relations**

Une des contraintes les plus courantes dans Neo4j est la contrainte d'unicité, qui garantit qu'un nœud ou une relation particulière ne peut être créé qu'une seule fois dans le graphe. Cela permet d'éviter les doublons et assure l'intégrité des données. Pour appliquer cette contrainte, vous pouvez utiliser la clause CREATE CONSTRAINT suivie du type de contrainte (NODE ou RELATION) et des propriétés sur lesquelles vous souhaitez appliquer l'unicité.

```
CREATE CONSTRAINT ON (n:Person) ASSERT n.email IS UNIQUE;
```

CONTRAINTES DE DONNÉES

● Existence des nœuds et des relations

Une autre contrainte importante est la contrainte d'existence, qui garantit qu'une relation ne peut être créée que si les nœuds sources et cibles existent déjà dans le graphe. Cela évite les relations orphelines qui pourraient compromettre l'intégrité des données. Cette contrainte peut être appliquée en utilisant des clauses de vérification lors de la création de relations.

```
MATCH (source:Person {name: 'John'}), (target:Person {name: 'Jane'})  
CREATE (source)-[:FRIENDS_WITH]->(target);
```

CONTRAINTES DE DONNÉES

● Contraintes de validation personnalisées

En plus des contraintes prédéfinies, Neo4j permet également de définir des contraintes de validation personnalisées pour garantir des règles spécifiques définies par l'utilisateur. Cela peut inclure des contraintes telles que des valeurs minimales ou maximales pour certaines propriétés, des expressions régulières pour la validation de formats de données, etc.

```
CREATE CONSTRAINT my_constraint ON (n:Person) ASSERT n.age >= 18;
```

FILTRAGE ET AGGREGATION

● Filtrage simple

Le filtrage simple consiste à sélectionner des éléments spécifiques d'un ensemble de données en fonction de critères prédéfinis.

```
MATCH (p:Person) WHERE p.age > 30 RETURN p;
```

● Agrégation simple

L'agrégation simple consiste à regrouper des données similaires et à effectuer une opération d'agrégation (comme le comptage, la sommation, la moyenne, etc.) sur ces données regroupées.

```
MATCH (p:Person) RETURN COUNT(p) AS totalPersons;
```

FILTRAGE ET AGGREGATION

● Filtrage avec agrégation

Le filtrage avec agrégation consiste à filtrer les données en fonction de certains critères, puis à regrouper et à effectuer une opération d'agrégation sur les données filtrées.

```
MATCH (p:Person) WITH p.age / 10 AS ageGroup, COUNT(p) AS numberOfPeople  
RETURN ageGroup * 10 AS ageRange, numberOfPeople;
```

● Filtrage et agrégation avec des relations

Le filtrage et l'agrégation avec des relations impliquent de filtrer les données en fonction de critères liés aux relations entre les nœuds, puis d'effectuer des opérations d'agrégation sur les données filtrées.

```
MATCH (a:Author)-[:WROTE]->(b:Article)  
WITH a, COUNT(b) AS numArticles WHERE numArticles > 3  
RETURN a.name AS author, numArticles;
```

PERFORMANCE ET OPTIMISATION DANS NEO4J

- Optimisation des requêtes
- Indexation et autres techniques

OPTIMISATION DES REQUÊTES

● Objectifs

L'optimisation des requêtes Cypher vise à rédiger des requêtes efficaces qui s'exécutent rapidement et utilisent de manière optimale les ressources de la base de données.

OPTIMISATION DES REQUÊTES

● Les bonnes pratiques

● Utiliser des clauses de filtrage dès que possible

Limitez les résultats de votre requête en utilisant des clauses WHERE ou FILTER pour restreindre les données dès le début de la requête.

● Utiliser des indexes

Utilisez des indexes sur les propriétés que vous utilisez fréquemment dans les clauses WHERE pour accélérer la recherche.

● Limiter les résultats avec LIMIT

Utilisez la clause LIMIT pour limiter le nombre de résultats renvoyés par la requête lorsque cela est possible.

OPTIMISATION DES REQUÊTES

● Les bonnes pratiques

● Utiliser des variables de chemin

Utilisez des variables de chemin pour limiter la taille des chemins dans les requêtes, en spécifiant des limites sur la longueur des chemins à travers le graphe.

● Utiliser des fonctions Cypher efficaces

Choisissez judicieusement les fonctions Cypher pour vos besoins spécifiques. Par exemple, préférez les fonctions de correspondance plus efficaces telles que MATCH plutôt que les fonctions moins efficaces comme OPTIONAL MATCH.

INDEXATION ET AUTRES TECHNIQUES

● Objectifs

L'indexation et d'autres techniques d'optimisation de la performance visent à améliorer la vitesse d'accès aux données en créant des structures de données supplémentaires ou en utilisant des algorithmes plus efficaces.

INDEXATION ET AUTRES TECHNIQUES

● Techniques

● Partitionnement de données

Divisez les données en partitions logiques pour répartir la charge de travail et améliorer les performances.

● Mise en cache

Utilisez des mécanismes de mise en cache pour stocker en mémoire les résultats fréquemment consultés et éviter des accès répétés à la base de données.

● Tuning de la mémoire

Ajustez les paramètres de mémoire de Neo4j pour optimiser l'utilisation des ressources système et améliorer les performances globales.

INDEXATION ET AUTRES TECHNIQUES

● Techniques

● Optimisation du schéma

Révisez et optimisez régulièrement le modèle de données pour garantir qu'il reflète efficacement les besoins de votre application et minimise les opérations inutiles.

● Analyse du plan d'exécution

Utilisez les outils de diagnostic fournis par Neo4j pour analyser les plans d'exécution des requêtes et identifier les goulots d'étranglement potentiels.

EXIGENCES DE SYSTÈME

- **Exigences Matériel**
- **Exigences Logiciel**

EXIGENCES DE SYSTÈME

Les exigences de système désignent les spécifications techniques minimales et recommandées nécessaires pour installer et faire fonctionner le logiciel de manière optimale

EXIGENCES MATÉRIEL

Processeur

Neo4j est assez gourmand en ressources CPU, donc un processeur multicœur est recommandé pour des performances optimales.

Réseau

Une connexion réseau rapide est recommandée, surtout si vous prévoyez de déployer Neo4j dans un environnement distribué.

Stockage

Le stockage SSD est fortement recommandé pour des performances optimales, car Neo4j accède fréquemment aux données

Mémoire

Plus vous avez de mémoire, mieux c'est. Au moins 8 Go de RAM sont recommandés pour les petites charges de travail, mais pour des performances optimales, 16 Go ou plus sont préférables

EXIGENCES LOGICIEL

Système d'exploitation

Neo4j est compatible avec plusieurs systèmes d'exploitation, y compris Linux, Windows et macOS. Assurez-vous de vérifier la compatibilité avec la version spécifique de Neo4j que vous envisagez d'utiliser.

Java

Neo4j est écrit en Java, donc une installation de Java est nécessaire sur la machine hôte. Assurez-vous d'utiliser une version compatible de Java selon les recommandations de Neo4j

L'INTÉGRATION DE NEO4J AVEC D'AUTRES TECHNOLOGIES

- JAVA
- PYTHON

JAVA

Étant donné que Neo4j est écrit en Java, il dispose d'une excellente intégration avec ce langage. L'API Java fournie par Neo4j permet aux développeurs d'interagir avec la base de données de manière native.

Exemple simple de comment utiliser l'API Java fournie par Neo4j pour interagir avec la base de données :

● ETAPE 1

Assurez-vous d'avoir téléchargé et installé Neo4j, et que le serveur Neo4j soit en cours d'exécution sur votre machine locale.

● ETAPE 2

Vous devez inclure la dépendance Maven (si vous utilisez Maven) ou la bibliothèque jar (si vous utilisez un autre système de gestion de dépendances) dans votre projet Java.

Si vous utilisez Maven, vous pouvez ajouter la dépendance suivante à votre fichier pom.xml :

```
<dependency>
    <groupId>org.neo4j.driver</groupId>
    <artifactId>neo4j-java-driver</artifactId>
    <version>4.3.1</version> <!-- Mettez à jour la version en fonction de votre besoin -->
</dependency>
```

Exemple

```
import org.neo4j.driver.*;
import org.neo4j.driver.Record;
public class Neo4jExample {
    public static void main(String[] args) {
        // Configuration de la connexion à la base de données Neo4j
        String uri = "bolt://localhost:7687"; // L'URI de connexion à la base de données
        String username = "neo4j"; // Nom d'utilisateur de la base de données (par défaut "neo4j")
        String password = "password"; // Mot de passe de la base de données (par défaut "password")

        // Création d'une session Neo4j
        try (Driver driver = GraphDatabase.driver(uri, AuthTokens.basic(username, password));
            Session session = driver.session()) {
            // Exécution d'une requête Cypher
            String query = "MATCH (n) RETURN count(n) as count";
            Result result = session.run(query);

            // Traitement des résultats de la requête
            while (result.hasNext()) {
                Record record = result.next();
                long count = record.get("count").asLong();
                System.out.println("Nombre total de nœuds dans la base de données : " + count);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

PYTHON

Neo4j propose également un pilote officiel pour Python, appelé "Neo4j Python Driver", qui permet aux développeurs d'utiliser Python pour interagir avec la base de données.

● ETAPE 1

1. Assurez-vous d'avoir téléchargé et installé Neo4j, et que le serveur Neo4j soit en cours d'exécution sur votre machine locale.

● ETAPE 2

Installer le pilote Python pour Neo4j. Vous pouvez le faire en utilisant pip

```
(pythonProject3) PS C:\Users\hp\Desktop\pythonProject3> pip install neo4j
Collecting neo4j
  Downloading neo4j-5.19.0.tar.gz (202 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 203.0/203.0 kB 202.2 kB/s eta 0:00:00
Installing build dependencies ... done
Getting requirements to build wheel ... done
Installing backend dependencies ... done
Preparing metadata (pyproject.toml) ... done
Collecting pytz
  Using cached pytz-2024.1-py2.py3-none-any.whl (505 kB)
Building wheels for collected packages: neo4j
  Building wheel for neo4j (pyproject.toml) ... done
  Created wheel for neo4j: filename=neo4j-5.19.0-py3-none-any.whl size=280768 sha256=223dd59c9d3b5f33900ba691830449777793a93296e8643b9fcf444416688af1
  Stored in directory: c:\users\hp\appdata\local\pip\cache\wheels\8d\44\2c\02104e871d6023b947989dcf81b35b308ead0ee1e50a059c0e
Successfully built neo4j
Installing collected packages: pytz, neo4j
Successfully installed neo4j-5.19.0 pytz-2024.1
```

Exemple

```
from neo4j import GraphDatabase

# Configuration de la connexion à la base de données Neo4j
uri = "bolt://localhost:7687" # L'URI de connexion à la base de données
username = "neo4j" # Nom d'utilisateur de la base de données (par défaut "neo4j")
password = "password" # Mot de passe de la base de données (par défaut "password")

# Définition de la fonction pour exécuter une requête Cypher
def run_query(query):
    with GraphDatabase.driver(uri, auth=(username, password)) as driver:
        with driver.session() as session:
            result = session.run(query)
            for record in result:
                print(record)

# Exemple d'exécution d'une requête Cypher
query = "MATCH (n) RETURN count(n) AS count"
run_query(query)
```

AUTRES LANGAGES

Bien que Neo4j propose des pilotes officiels pour Java et Python, il existe également des bibliothèques tierces pour d'autres langages populaires tels que JavaScript, Ruby, et bien d'autres

CONCLUSION

Dans cette présentation, nous avons couvert les principaux aspects de Neo4j, une base de données orientée graphe puissante et flexible. En commençant par une introduction à Neo4j, nous avons exploré son modèle de données graphiques, son langage de requête Cypher, ainsi que les méthodes d'installation, de configuration et d'optimisation.

CONCLUSION

En utilisant des cas d'utilisation variés, nous avons souligné l'importance de Neo4j dans le contexte des bases de données orientées graphe, mettant en évidence sa capacité à modéliser des structures complexes et des relations entre les données de manière efficace.

Nous avons également abordé des aspects avancés tels que la modélisation de données, l'optimisation des performances et l'intégration avec d'autres technologies, offrant ainsi une vue d'ensemble complète de l'écosystème Neo4j.



MERCI !!!

Royaume du Maroc

Ministère de l'Enseignement Supérieur,
de la Recherche Scientifique et de l'Innovation

Université Cadi Ayyad

École Nationale des Sciences Appliquées

Marrakech



المملكة المغربية
وزارة التعليم العالي والبحث العلمي والإبتكار
جامعة القاضي عياض
المدرسة الوطنية للعلوم التطبيقية
مراكش



Présenté par:

- ACHQIBOU Hassania
- AIT EL MOUDEN Hafsa
- BENKSSAB Salma
- NASSIRI Noussaiba

Encadré par:

M. BEKKARI Aissam

