



Movie Graphe

Movie Graph est une mini application de graphe contenant des acteurs et des réalisateurs qui sont liés par les films auxquels ils ont collaboré.

- 1) Exécuter la commande dans le fichier *CREATE.txt* cela créera le graphe du film.
- 2) Déterminer le nombre de nœuds et d'arrêtes du graphe. Préciser également le nombre de nœud pour chaque catégorie.
- 3) Trouvez l'acteur nommé "Tom Hanks" ...
- 4) Trouver le film avec le titre " Cloud Atlas " ...
- 5) Trouvez 10 personnes...
- 6) Retrouvez les films sortis dans les années 1990...
- 7) Liste de tous les films de Tom Hanks...
- 8) Qui a réalisé "Cloud Atlas" ?
- 9) Les co-acteurs de Tom Hanks...
- 10) Comment les gens sont liés à l'"Atlas des nuages"...
- 11) Films et acteurs jusqu'à 4e niveau de profondeur de Kevin Bacon
- 12) Le chemin le plus court de Kevin Bacon vers Meg Ryan

Northwind Graphe

Ce TP montre comment migrer d'une base de données relationnelle vers Neo4j. La transformation est itérative et délibérée, mettant l'accent sur le changement conceptuel des tables relationnelles vers les nœuds et les relations d'un graphe.

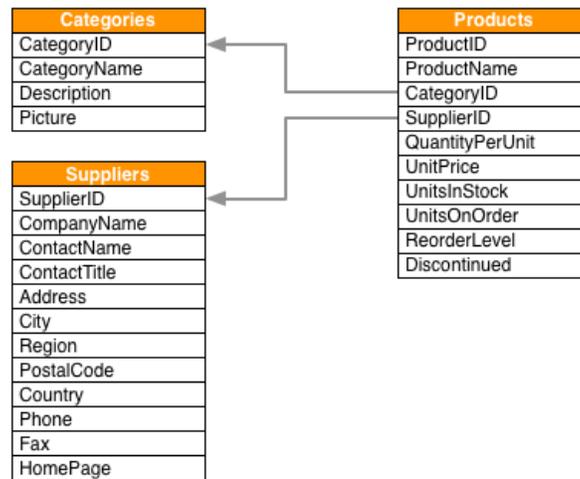
Dans cette partie nous verrons comment :

- Charger : créer des données à partir de fichiers CSV externes
- Indexer : nœuds d'index basés sur l'étiquette
- Relier : transformer les références clés étrangères en relations de données
- Promouvoir : transformer les dossiers d'adhésion en relations

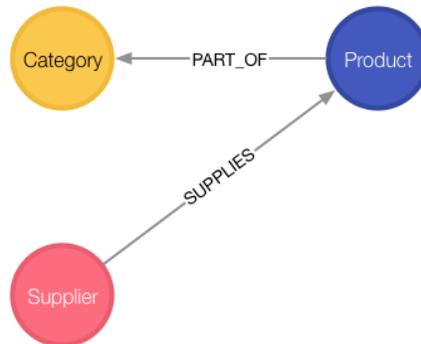
Catalogue Produit

Northwind vend des produits alimentaires (**Product**) dans quelques catégories (**Category**), fournis par des fournisseurs (**Supplier**). Commençons par charger les tableaux des catalogues de produits.

Utiliser la clause **LOAD CSV** récupère un fichier CSV à partir d'une URL valide, en appliquant une instruction Cypher à chaque ligne en utilisant un map nommée (ici nous utilisons le nom "row").



Les produits, les catégories et les fournisseurs sont reliés par des références clés étrangères. Promouvons ces dernières à des relations de données pour réaliser le graphe.



- 1 Charger les enregistrements à partir des fichiers suivants products.csv, categories.csv, suppliers.csv

```
1 LOAD CSV WITH HEADERS FROM "file:///E:/products.csv" AS row
2 CREATE (n:Product)
3 SET n = row,
4 n.unitPrice = toFloat(row.unitPrice),
5 n.unitsInStock = toInteger(row.unitsInStock), n.unitsOnOrder = toInteger(row.unitsOnOrder),
6 n.reorderLevel = toInteger(row.reorderLevel), n.discontinued = (row.discontinued <> "0")
```

```
1 LOAD CSV WITH HEADERS FROM "file:///E:/categories.csv" AS row
2 CREATE (n:Category)
3 SET n = row
```

```
1 LOAD CSV WITH HEADERS FROM "file:///E:/suppliers.csv" AS row
2 CREATE (n:Supplier)
3 SET n = row
```



2 Créer des index pour l'ensemble de noeuds

```
CREATE INDEX ON :Category(categoryID)
```

```
$ CREATE INDEX ON :Product(productID)
```

```
CREATE INDEX ON :Supplier(supplierID)
```

3 Créer des relations de données. Notez qu'il vous suffit de comparer les valeurs des nœuds des clés leur servant à une identification de façon unique des nœuds.

```
1 MATCH (p:Product),(c:Category)
2 WHERE p.categoryID = c.categoryID
3 CREATE (p)-[:PART_OF]->(c)
```

```
1 MATCH (p:Product),(s:Supplier)
2 WHERE p.supplierID = s.supplierID
3 CREATE (s)-[:SUPPLIES]->(p)
```

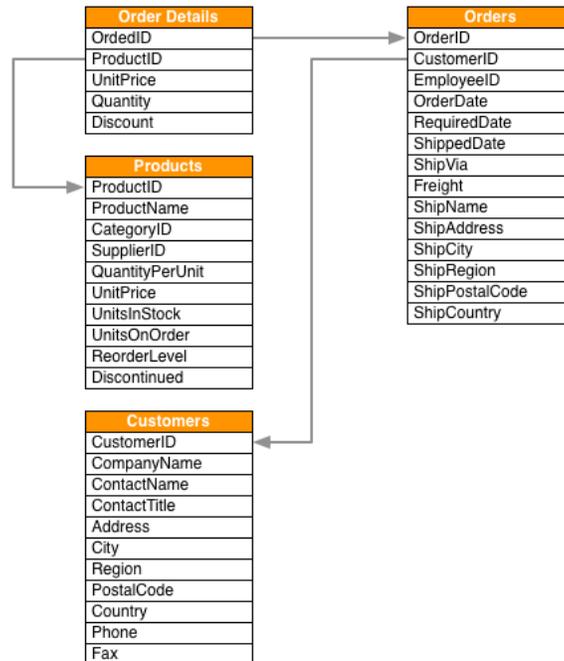
4 Listez les catégories de produits fournies par chaque fournisseur.

```
1 MATCH (s:Supplier)-->(p:Product)-->(c:Category)
2 RETURN s.companyName as Company, collect(distinct c.categoryName) as Categories
```

5 Trouvez les fournisseurs de produits.

```
MATCH (c:Category {categoryName:"Produce"})<--(:Product)<--(s:Supplier)
RETURN DISTINCT s.companyName as ProduceSuppliers
```

Les clients de Northwind passent des commandes qui peuvent être relayées au détail de plusieurs produits.



- 6 Charger et créer des index pour l'ensemble de nœuds entre les clients, ordre de commandes et produits

```
1 LOAD CSV WITH HEADERS FROM "file:///E:/customers.csv" AS row
2 CREATE (n:Customer)
3 SET n = row
```

```
1 LOAD CSV WITH HEADERS FROM "file:///E:/orders.csv" AS row
2 CREATE (n:Order)
3 SET n = row
```

```
CREATE INDEX ON :Customer(customerID)
```

```
CREATE INDEX ON :Order(orderID)
```

- 7 Créer des relations de données.

```
1 MATCH (c:Customer),(o:Order)
2 WHERE c.customerID = o.customerID
3 CREATE (c)-[:PURCHASED]->(o)
```



Notez que les détails de la commande font toujours partie d'une commande et qu'ils relient la commande à un produit - c'est une table commune. Les tables jointes sont toujours le signe d'une relation de données, indiquant un partage d'informations entre deux autres enregistrements.

Ici, nous allons directement enregistrer chaque detail d'ordre dans une relation du graphe.



8 Charger et indexer les enregistrements

```
LOAD CSV WITH HEADERS FROM "http://order-details.csv" AS row
MATCH (p:Product), (o:Order)
WHERE p.productID = row.productID AND o.orderID = row.orderID
CREATE (o)-[details:ORDERS]->(p)
SET details = row,
    details.quantity = toInteger(row.quantity)
```

9 Rechercher la quantité de produits achetés par chaque clients

```
MATCH (cust:Customer)-[:PURCHASED]->(o:Order)-[o:ORDERS]->(p:Product),
      (p)-[:PART_OF]->(c:Category {categoryName:"Produce"})
RETURN DISTINCT cust.contactName as CustomerName, SUM(o.quantity) AS TotalProductsPurchased
```
