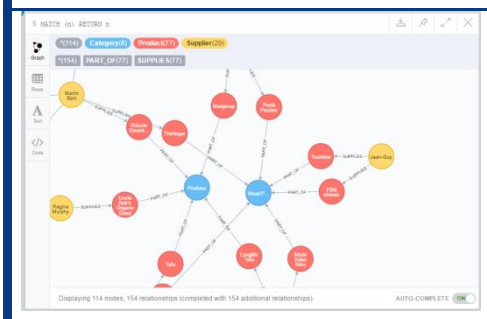
Neo4j 3.03

neo4j https://www.neo4j.com	Graph database management system	
	Système d'exploitation :	Windows/Mac/Linux
	Développé par :	Neo Technology
GPL (version gratuite) ou AGPL (version commerciale)	Personne de concat :	Vandy.Berten@Smals.be

Fonctionnalités



Neo4J est un système de gestion de base de données orienté « graphe ». Dans un tel système, les données sont composées d'une part de nœuds (ou entités), d'autre part de relations entre deux nœuds. Les nœuds et relations peuvent avoir des « labels » (*Person*, *Company*…), et des propriétés (name: « MyName », price: 1.20…).

Pour comparer avec un système de base de données relationnelle classique dans lequel on aurait 3 tables : *products*, *categories* et *suppliers* (un produit ayant une référence vers une catégorie et un fournisseur), nous aurions dans Neo4j des nœuds avec comme label *Product*, *Category* et *Supplier*, et des relations avec comme types « *Supplies* » (entre un fournisseur et un produit) et « *Part of* » (entre un produit et une catégorie).

La base de données peut être interrogée via des requêtes en *Cypher*, un langage proche de SQL. Une requête s'exprimera typiquement en termes de *pattern* (lister les fournisseurs qui ont plusieurs produits appartenant à des catégories différentes) ou en termes de *chemin* (dans un réseau social, quel est le plus court chemin entre une personne A et une personne B).

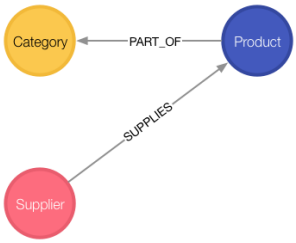
Conclusions & Recommandations

Si les bases de données orientées « graphe » n'ont pas vocation à remplacer les bases de données classiques dans toutes les circonstances, il existe de nombreuses situations où elles peuvent avoir un grand intérêt. En particulier quand on se focalise sur les relations qui existent entre des entités de type différent, plutôt qu'entre les attributs de ces entités. Les analystes et DB architects auraient avantage à connaitre ce genre de technologies, de façon à pouvoir faire des choix éclairés.

Neo4j peut également être un allié précieux pour des Data Scientists, et dans le cadre de la lutte contre la fraude, en permettant de trouver des *patterns* particuliers.

Tests & Résultats

Il existe principalement deux façons d'utiliser Neo4j : soit en exécutant des requêtes Cypher au travers d'un driver au sein d'un programme, comme on le ferait avec des requêtes SQL, soit via une interface Web interactive. Dans ce cas, le résultat d'une requête peut être soit visualisé sous forme d'un tableau, soit, si ce résultat contient des nœuds et des liens, sous forme du sous-réseau résultant.



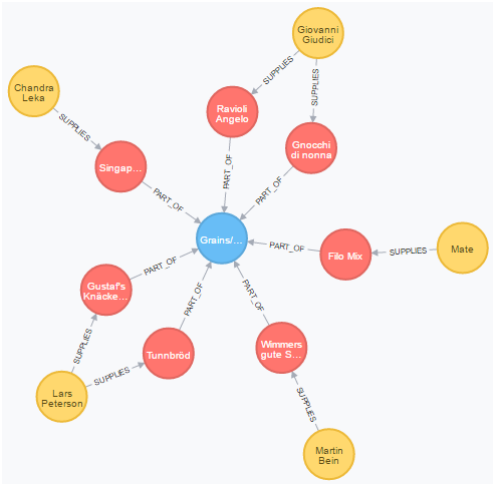
Considérons la requête suivante :

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

La partie « MATCH » décrit le *pattern* qui nous intéresse : un *Supplier* avec une relation vers un *Product*, ayant lui une relation vers une *Category* (comme dans la figure ci-contre). Pour toutes les séquences qui correspondent, on demande dans le « RETURN » de fournir un tableau à deux colonnes : le « companyName » (propriété des nœuds ayant le label Supplier), et la liste des catégories. La même requête en SQL aurait nécessité 3 « joins », dans lesquels il aurait fallu expliciter la façon dont les tables sont liées l'une à l'autre.

La requête Cypher suivante extrait le graphe des fournisseurs de céréales et les produits associés :

```
MATCH (c:Category)<-[:PART_OF]-(p:Product)<-[:SUPPLIES]-(s:Supplier)
WHERE c.categoryName CONTAINS "Cereals"
RETURN s, p, c
```



Les parties entre parenthèses décrivent les nœuds, les parties entre crochets décrivent les relations. On cherche donc une relation de type *PART_OF* entre une *Category* (bleu) et un *Product* (rouge). Idem avec les fournisseurs (en jaune). On a ici 7 résultats (chaîne de 3 nœuds) qui sont combinés dans la visualisation.

L'adoption d'une base de données « orientée graphe » demande de penser ses données d'une façon complètement différente, mais souvent plus intuitive et plus proche de la réalité. Il faut cependant être attentif à des obstacles nouveaux, tels que l'explosion combinatoire : le nombre de chemins possibles entre deux nœuds est exponentiel par rapport au nombre de nœuds. Cependant, pour de nombreux type de requêtes, le temps de réponse de Neo4j peut s'avérer être nettement plus rapide qu'avec un système classique.

Conditions d'utilisation & Budget

La version « Community Edition » est gratuite. Le prix de la version « Enterprise », qui permet d'augmenter les performances, n'est pas public.