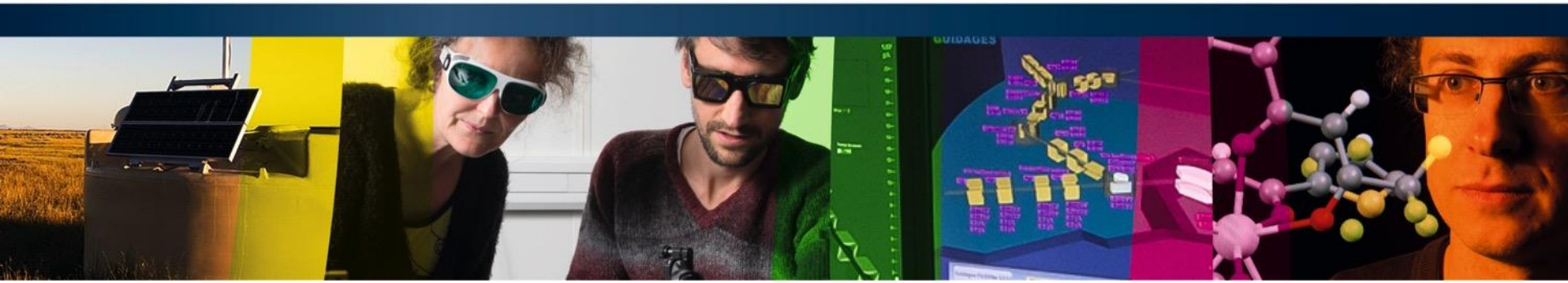
www.cnrs.fr

Neo4J

Une introduction



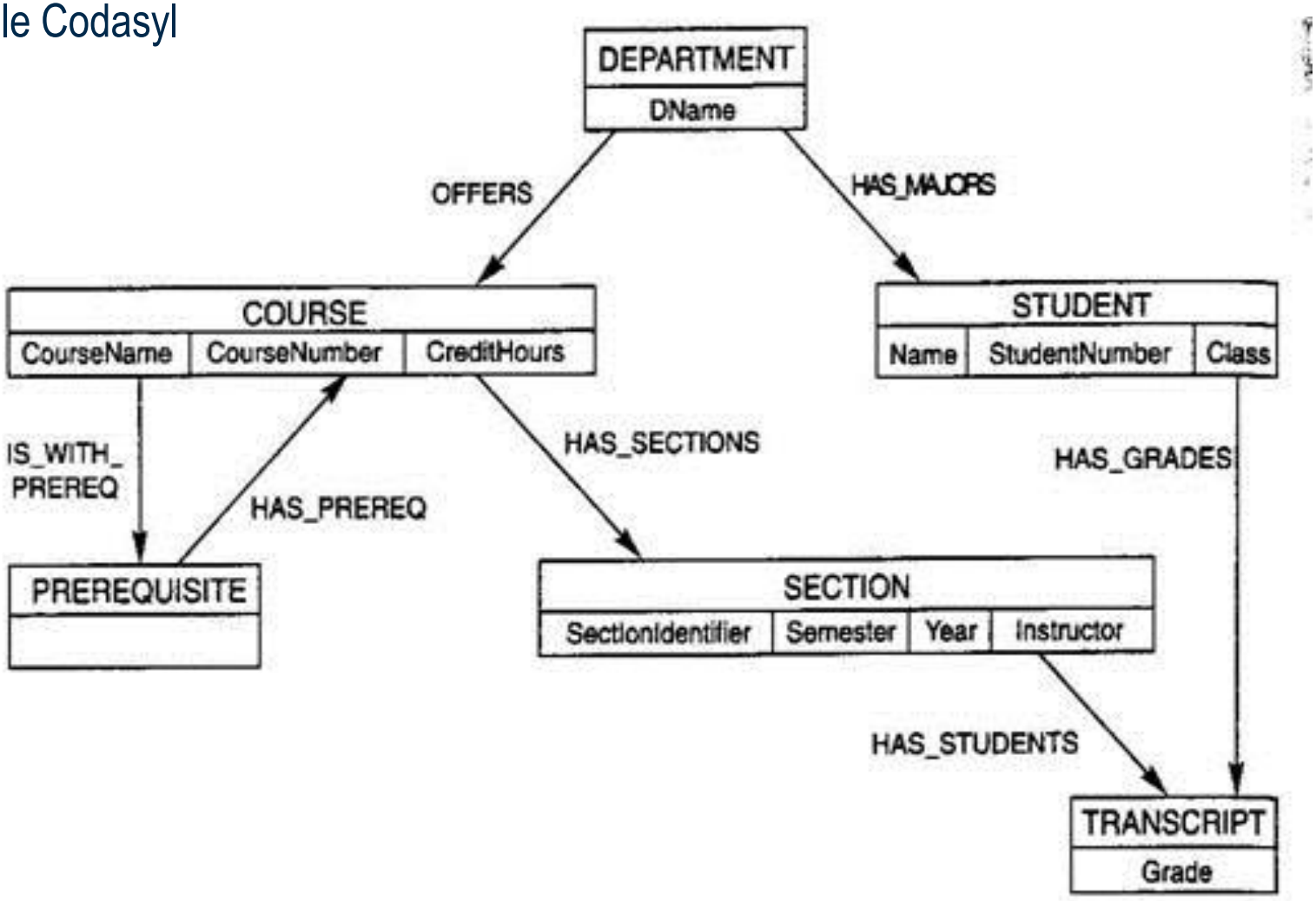
Sommaire

www.cnrs.fr

Historique

Historique

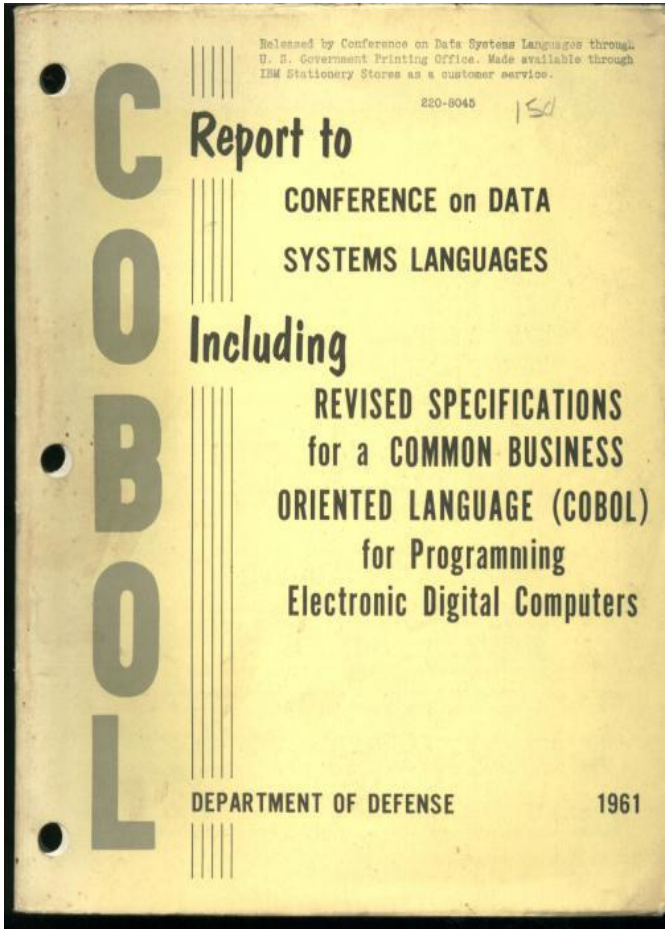
- **Histoire des bases**
  - Modèle Codasyl

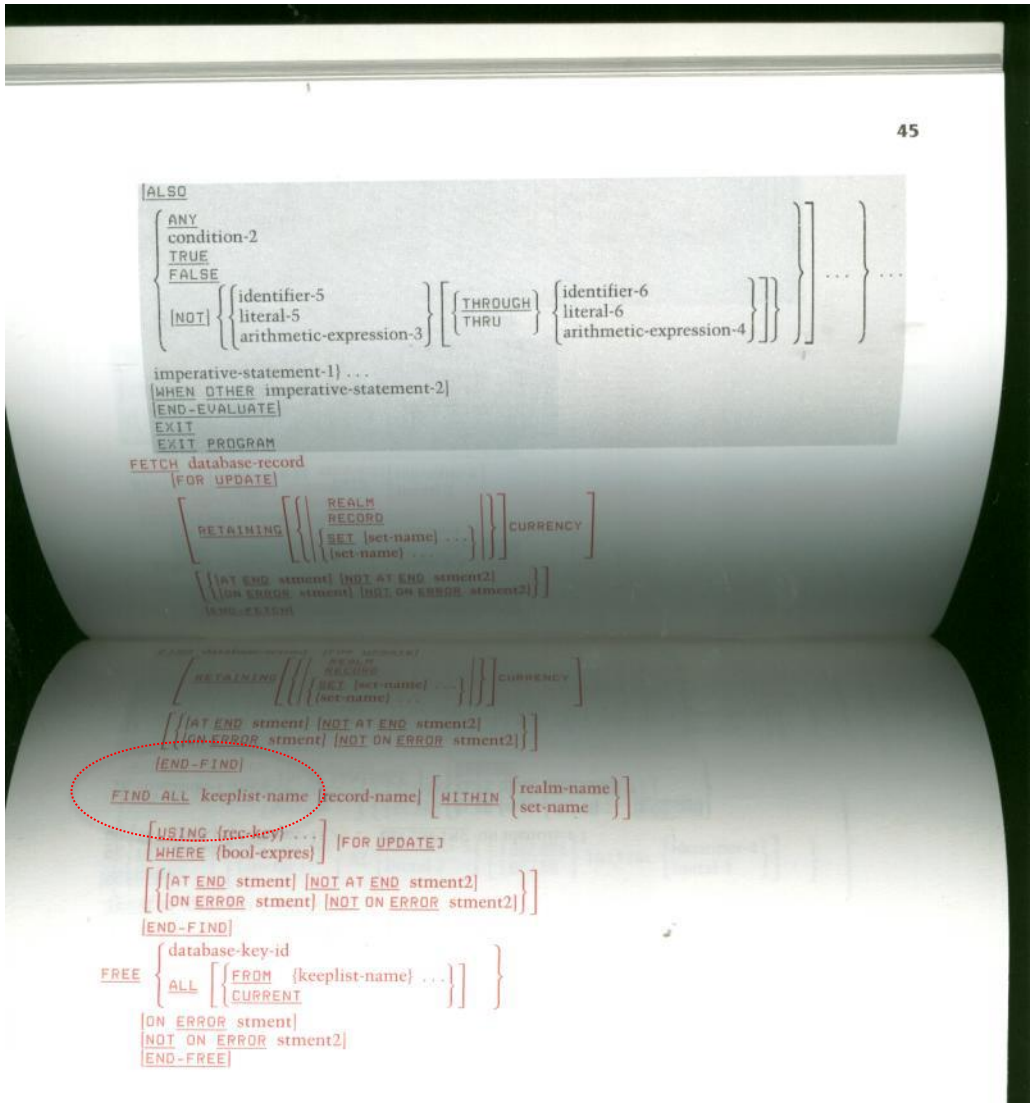


Source : http://www.cs.aucegypt.edu/~csci253/DBConcepts%20v2.htm

Historique

- **Histoire des bases**
  - Cobol





Source : https://www.tomfolio.com/bookssub.asp?subid=1773&lo=2&page=1

Historique

- **Neo4j développé par la société Neo technology depuis 2000**
  - Société suédoise basée aux US
  - Version 1.0 en 2010
- **SGBD NoSQL open source (licence GPL V3.0)**
  - Ecrit en Java
- **Utilisé par des grands comptes comme Cisco, HP, Adidas, SFR, SNCF, Accenture, LinkedIN, Meetic (;-)), etc.**
- **Dernière version stable : 3.0 en avril 2016**
- **Multi-plateformes**
- **Beaucoup d'interfaces avec les langages de programmation**
  - Java évidemment … et Python !

Ecosystème des bases graphes

- **Il existe plusieurs solutions de bases graphe**
  - Titan : http://thinkaurelius.github.io/titan/
  - FlockDB : https://github.com/twitter/flockdb
  - GraphBase (commercial) : http://graphbase.net/
  - InfiniteGraph (commercial) : http://www.objectivity.com/
- **Beaucoup de ces solutions n'ont pas eu de suite**
  - Segmentation des solutions sur un créneau restreint
- **Neo4J reste actuellement une des bases graphe parmi les plus connues**

Ecosystème des bases graphes

- **Une base connue en effet …**

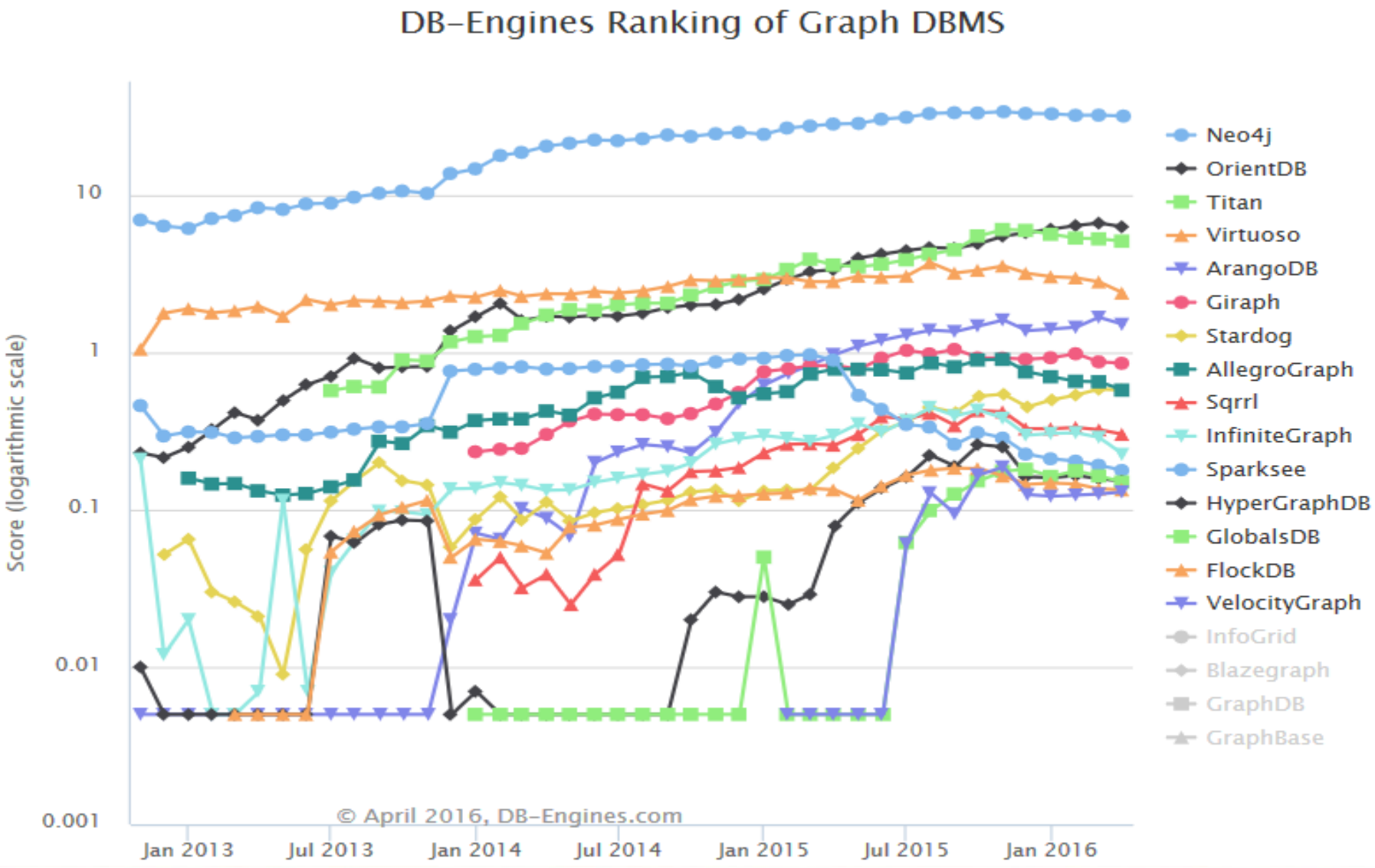
Source : http://neo4j.com/news/neo4j-powers-panama-papers-investigation/

Ecosystème des bases graphes

- **Deux éditions de Neo4j sont disponibles :**
  - Une édition communautaire (Community Edition). Pas de services professionnels ni assistance. Licence gratuite GPLv3
  - Une édition entreprise (Enterprise Edition). Prise en charge de la haute disponibilité, de la gestion, de l'évolutivité et de la production. Licence commerciale par abonnement, assistance incluse. **Licence gratuite sous l'AGPLv3 pour les projets open source**.
- **Pas de prix publics visible en décembre 2015**

Ecosystème des bases graphes

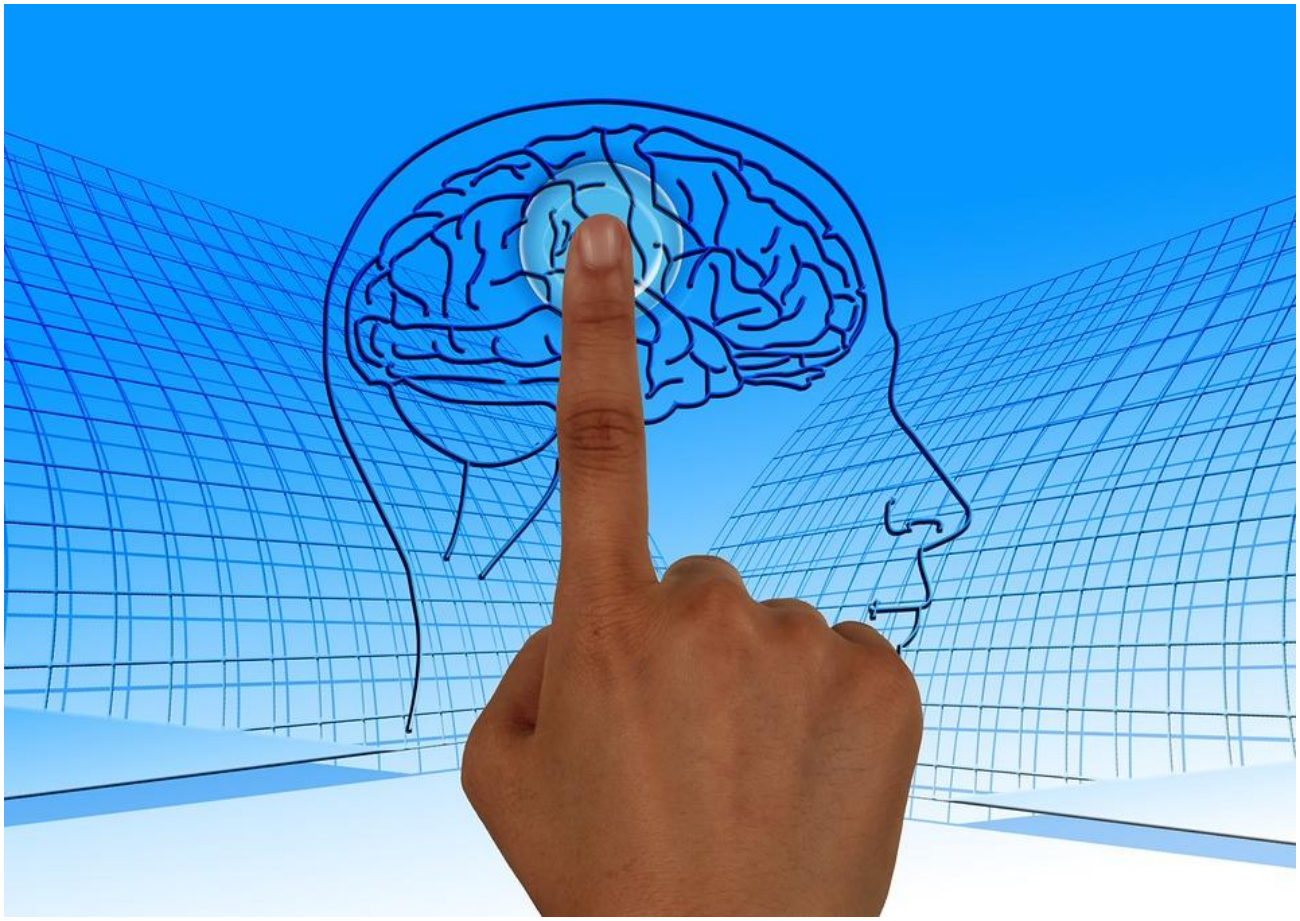
- Positionnement de Neo4J (http://db-engines.com/en/ranking)



Ecosystème des bases graphes

- **Dernières annonces du Graph Connect 2015 sur Neo4J**
  - Disponibilité d'un driver officiel pour tous les langages
    - Vision du codage homogène
  - Procédures stockées
  - OpenCypher
    - Supports
      - DataBricks (soutien de Spark)
      - Oracle (Oracle Spatial et Oracle Graph)
      - Tableau : référence dans l'exploitation des données et le Machine Learning
    - Pourquoi OpenCypher ?
      - Le modèle relationnel a su s'imposer historiquement par rapport aux bases réseaux car il existait un langage commun de requêtage
      - Neo4J essaie de ne pas répéter la même erreur et s'adjoint de grands éditeurs pour normaliser Cypher

Quelques rappels



Pourquoi une base graphe ?

- **Intérêt des données**
  - Connexion, interactions
- **Evolutions actuelles**
  - Volume : souvent cité
  - Connectivité : sous entendu mais plus rarement mis en avant
  - Les deux aspects ont tendance à subir un facteur multiplicatif
- **Complexité = Volume * Connectivité**

Source : http://sharkoliv.blogg.org/mulet-bien-charge-a115760130

Quelques rappels

Les graphes

- Graphe fini G =(S,A)
- Ensemble fini de sommets S = {s1,s2,...,s_n}
- Ensemble fini d'arêtes A = {a1,a2,...,a_m} pour les graphes non orientés
- On parle d'arcs si le graphe est orienté
- Si l'arête 'a' relie les sommets 'a' et 'b', on dira que ces sommets sont adjacents
- On appelle **ordre d'un graphe** le nombre de sommets n de ce graphe
- G, non orienté est **connexe** si, à partir de tout sommet S, il existe une suite d'arêtes permettant de rejoindre tous les autres sommets
- Un graphe est **complet** si tous les sommets sont adjacents
- Il existe plein d'algorithmes de parcours des graphes : en largeur, en profondeur …

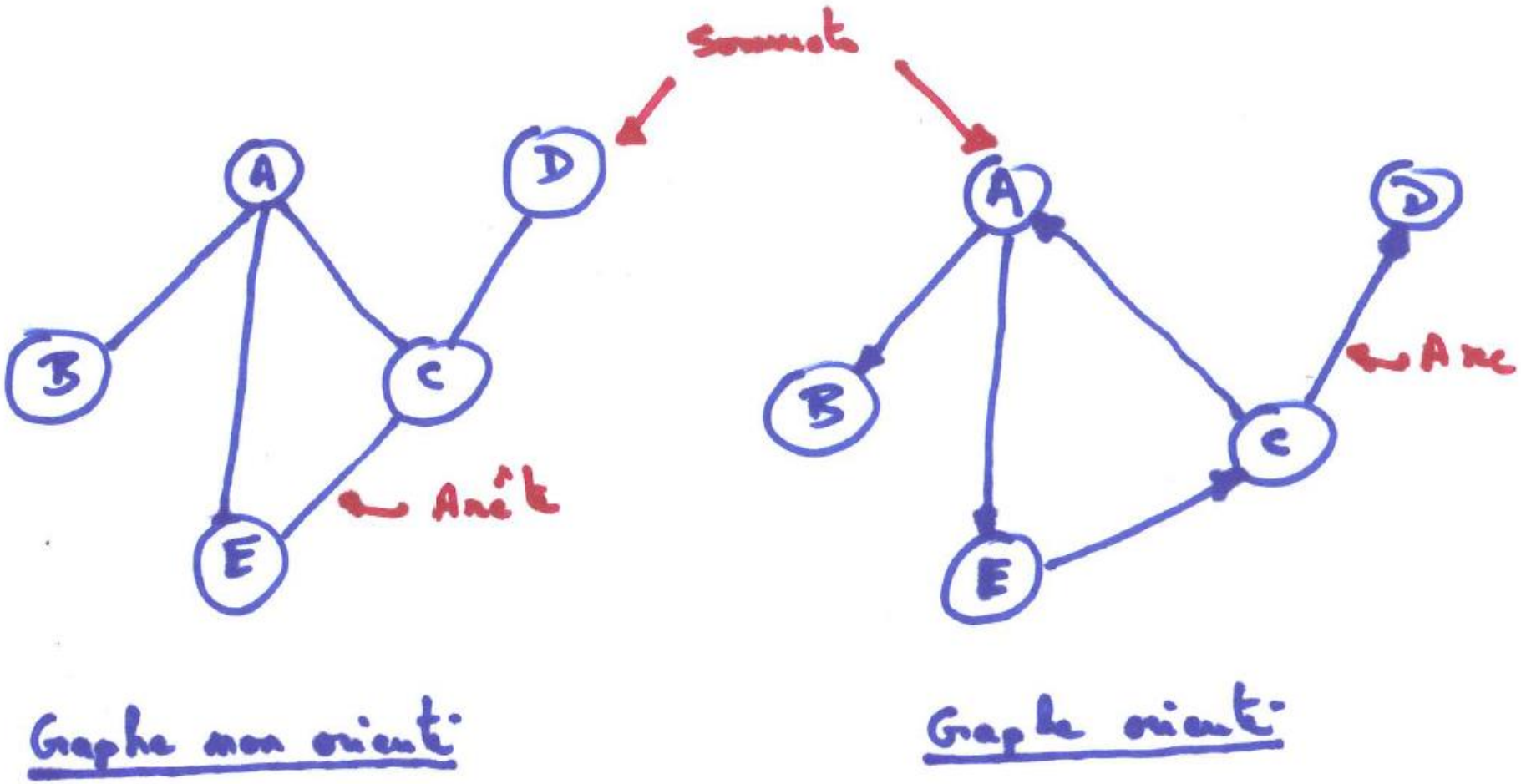
Pour quoi faire déjà ?

- Modéliser des problématiques spécifiques avec forte connectivité
- Recherche de liens optimaux

Source : http://sharkoliv.blogg.org/mulet-bien-charge-a115760130

Quelques rappels

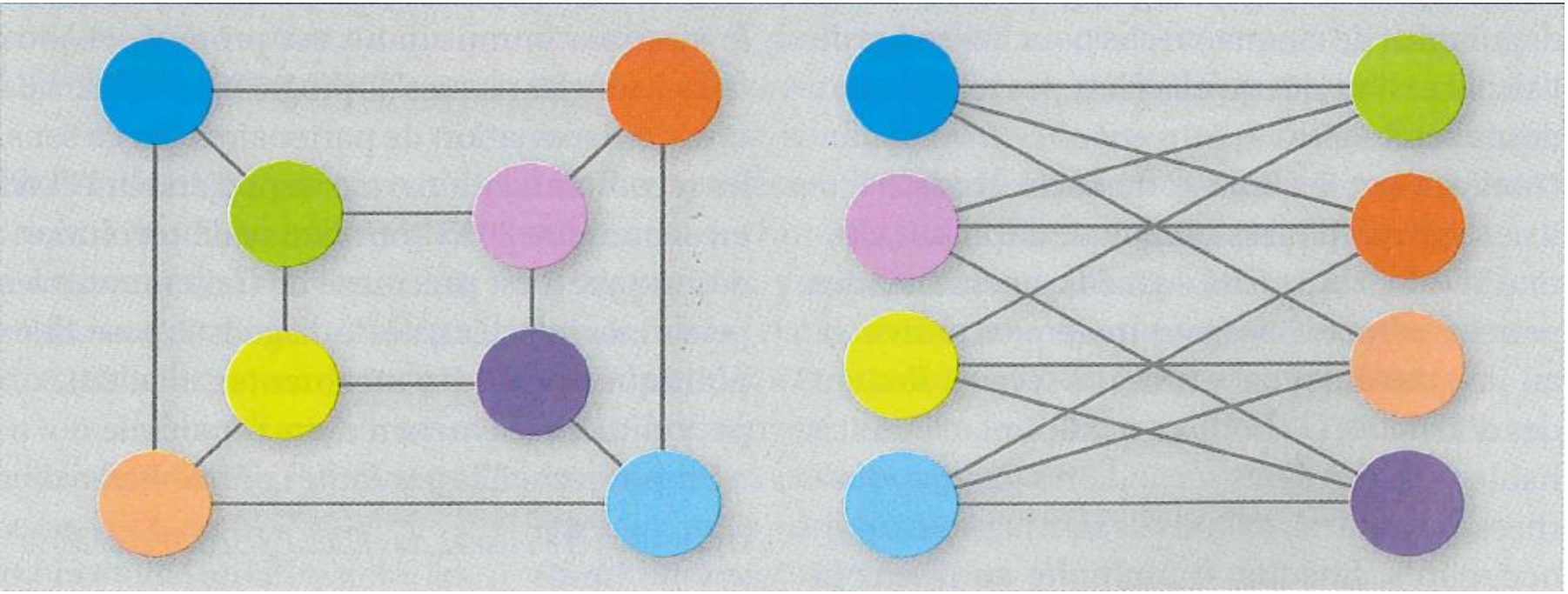
- **Exemples de graphes**



Quelques rappels

Exemples de graphes

▲ László Babai, lauréat du prix Gödel 1993 et du prix Knuth 2015.



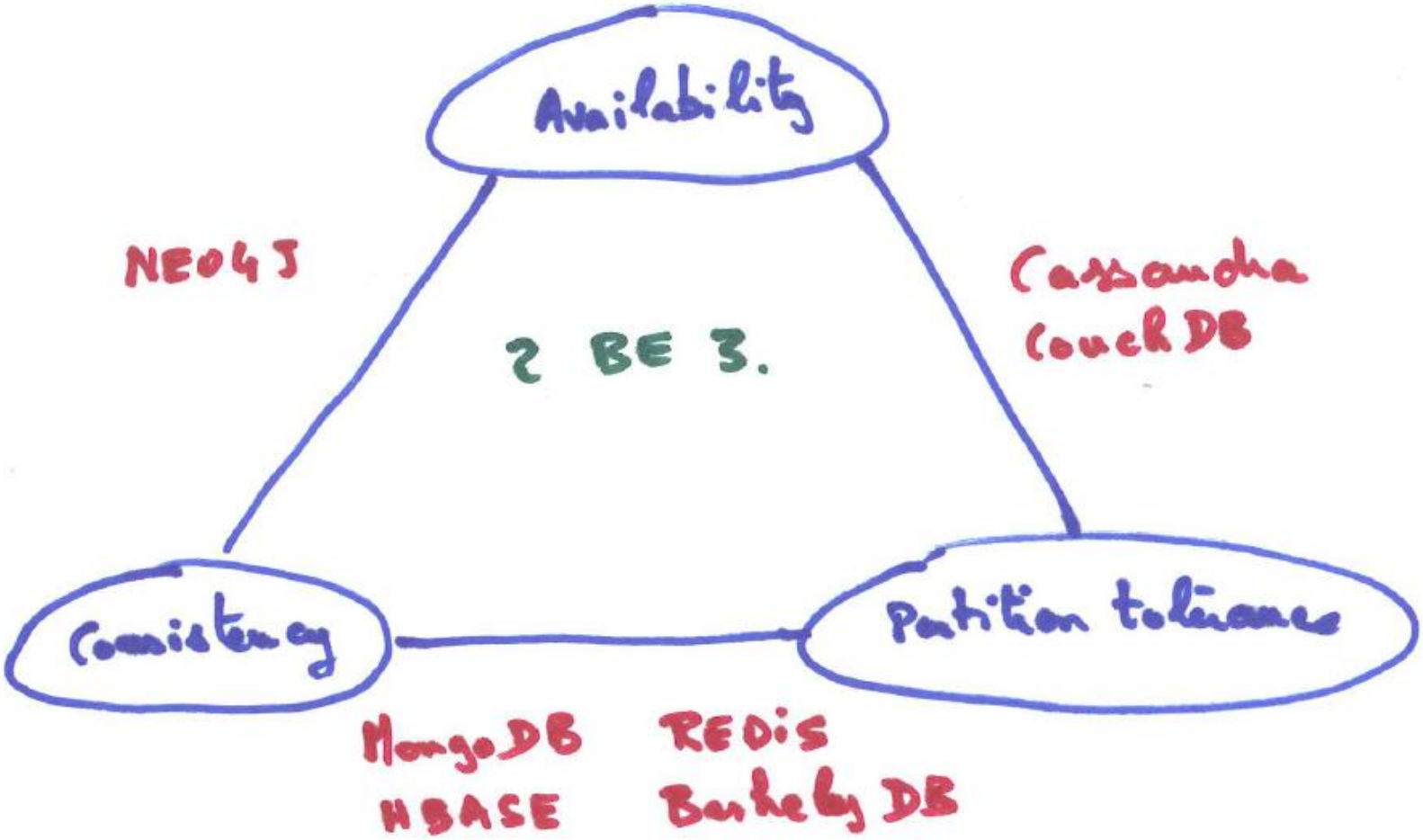
Source : La Recherche – Janvier 2016 – Comparaison de graphes – Philippe Pajot

Généralités sur Neo4J

Les cas d'usage

- **A chaque fois qu'il y a des relations 'many-to-many'**
  - Master Data Management (gestion des données de référence)
  - Infrastructures réseaux
  - Suggestions en temps réel (sites web, achats en ligne, …)
  - Détection de fraude
  - Réseaux sociaux
  - IAM (Identity and Access Management)

Positionnement CAP

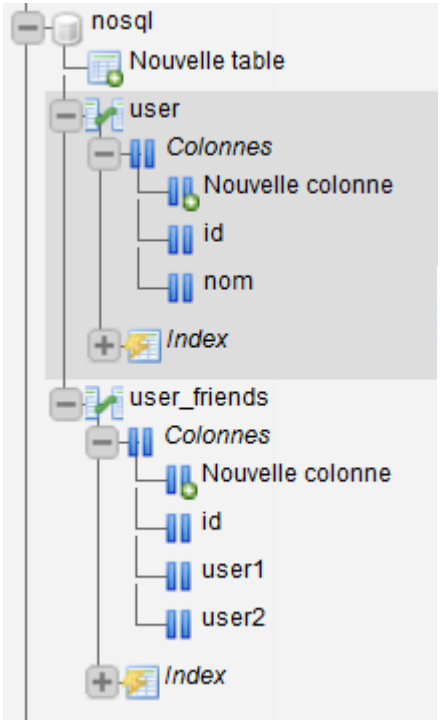


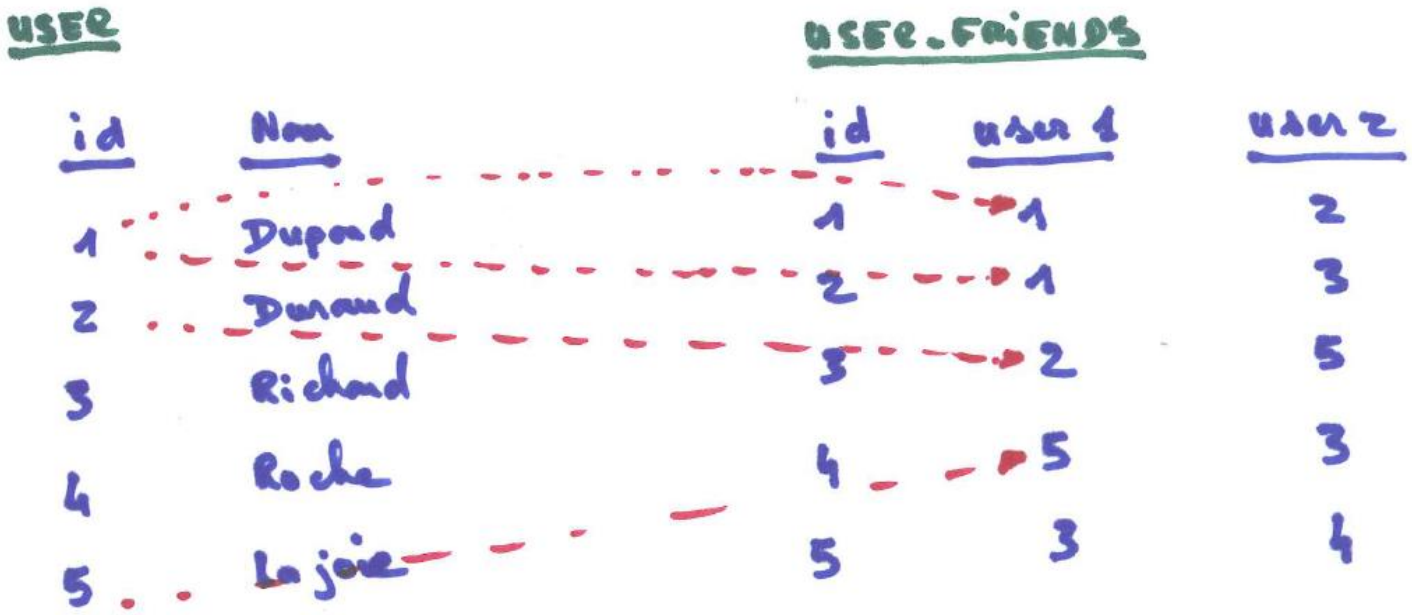
Source : http://blog.grio.com/2015/11/sql-nosql-a-brief-history.html

Les cas d'usage

Un exemple concret en MySQL

- Matérialiser un réseau d'amis





Les cas d'usage

- **Un exemple concret en MySQL**
  - Compter les amis
    - De premier niveau

```
SELECT u1.nom,count(*) FROM user u1 INNER JOIN user_friends uf1 WHERE u1.id = uf1.user1 GROUP BY nom
```

nom	count(*)
Dupont	2
Durant	1
Lajoie	1
Richard	1

    - De deuxième niveau

```
select u1.nom,count(*) FROM user AS u1
  INNER JOIN user_friends AS uf1 on u1.id = uf1.user1
  INNER JOIN user_friends AS uf2 on uf1.user2=uf2.user1
  GROUP BY u1.nom
```

nom	count(*)
Dupont	2
Durant	1
Lajoie	1

Source : http://sharkoliv.blogg.org/mulet-bien-charge-a115760130

Les cas d'usage

Un exemple concret en MySQL

Compter les amis

De troisième niveau

```
select u1.nom,count(*) FROM user AS u1
  INNER JOIN user_friends AS uf1 on u1.id = uf1.user1
  INNER JOIN user_friends AS uf2 on uf1.user2=uf2.user1
  INNER JOIN user_friends AS uf3 on uf2.user2=uf3.user1
  GROUP BY u1.nom
```

nom	count(*)
Dupont	1
Durant	1

De quatrième niveau

```
select u1.nom,count(*) FROM user AS u1
  INNER JOIN user_friends AS uf1 on u1.id = uf1.user1
  INNER JOIN user_friends AS uf2 on uf1.user2=uf2.user1
  INNER JOIN user_friends AS uf3 on uf2.user2=uf3.user1
  INNER JOIN user_friends AS uf4 on uf3.user2=uf4.user1
  GROUP BY u1.nom
```

nom	count(*)
Dupont	1

Source : http://sharkoliv.blogg.org/mulet-bien-charge-a115760130

Les cas d'usage

- **Un exemple concret en MySQL**
  - Compter les amis
    - De cinquième niveau

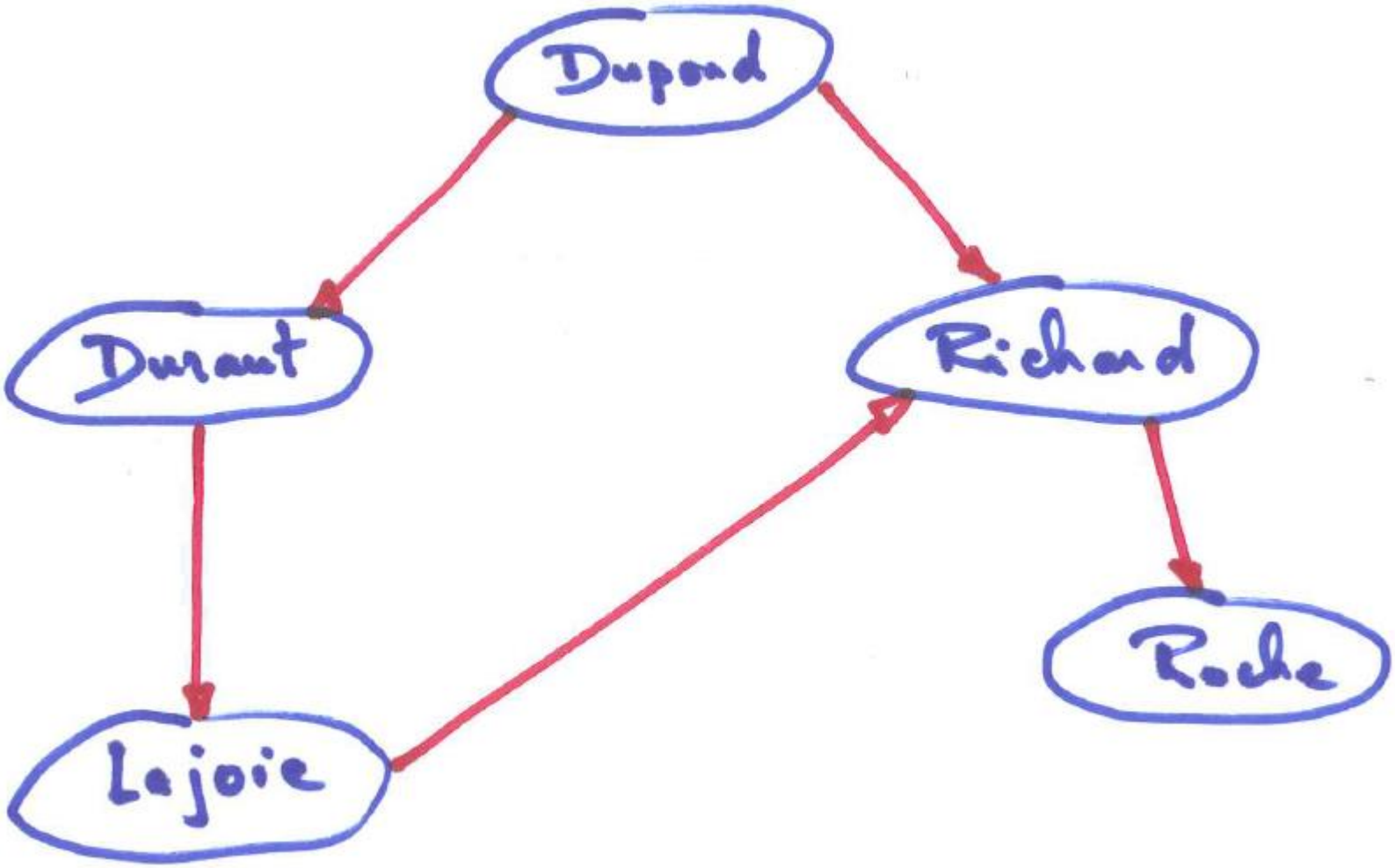
```
select u1.nom,count(*) FROM user AS u1
  INNER JOIN user_friends AS uf1 on u1.id = uf1.user1
  INNER JOIN user_friends AS uf2 on uf1.user2=uf2.user1
  INNER JOIN user_friends AS uf3 on uf2.user2=uf3.user1
  INNER JOIN user_friends AS uf4 on uf3.user2=uf4.user1
  INNER JOIN user_friends AS uf5 on uf4.user2=uf5.user1
  GROUP BY u1.nom
```

    - Le résultat est alors vide
- **Au total :**
  - De multiples jointures sont à réaliser, en fonction du niveau de « profondeur voulu »
  - Coûteux en temps et en mémoire :produit cartésien puis sélection
  - Si le nombres de relations augmente, les jointures deviennent impossibles à exécuter
    - Dans Neo4J in Action [1], avec un Intel I7 et 8 Go de RAM, il faut 92613 s pour 1000 users pour une profondeur de 5
    - Pour 1 million de users, le calcul ne finit jamais …

[1] Neo4J in Action, Aleksa Vukotic and Nicki Watt, Manning, 2015

Les cas d'usage

- La vision « graphe » du même problème est toute différente



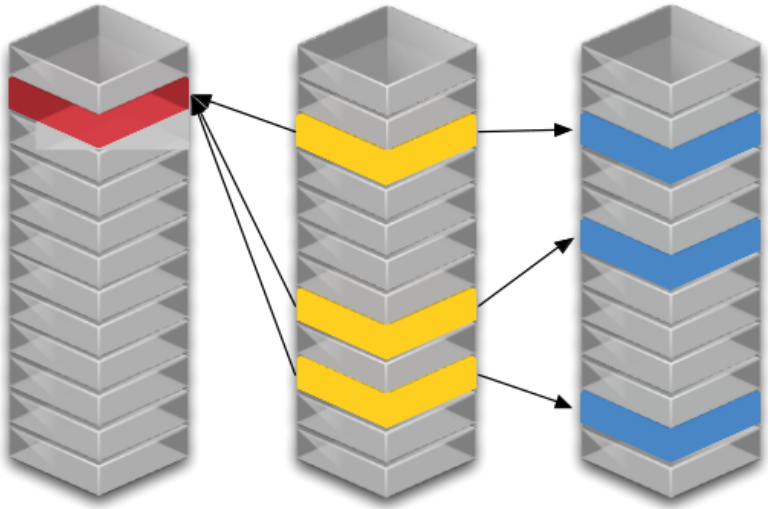
Les cas d'usage

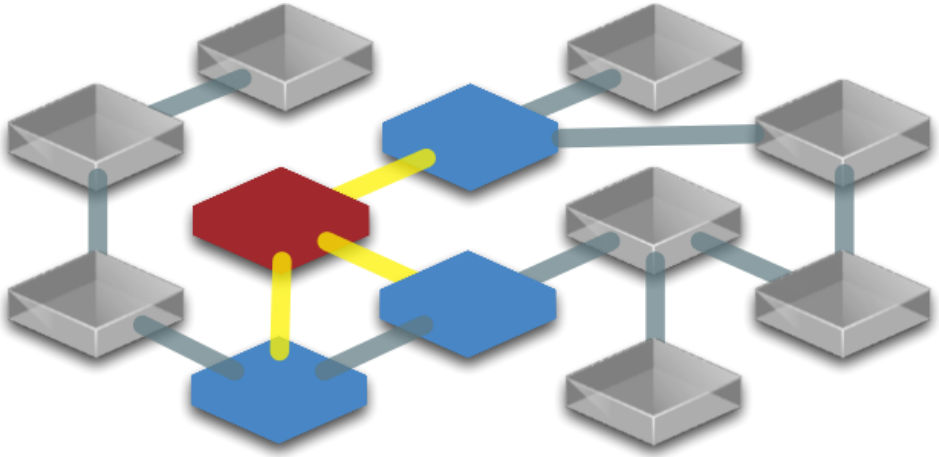
- **Dans ce cas :**
  - Inutile de travailler sur l'ensemble des données comme dans le modèle relationnel
  - Un nœud est choisi et le parcours du graphe se fait à partir de là
  - C'est le principe du **« graph traversal »** de Neo4J
  - Le « **graph traversal** » devient performant lorsque le volume de données est important, volume qui rend les jointures coûteuses voire impossibles
  - L'image qui peut être utilisée est celle du « stade de foot »[1]:
    - Compter le nombre de personnes dans un périmètre de 15 mètres autour de vous dans un stade de 1000 personnes ou 50000 personnes coûte le même temps avec un graphe

[1] Neo4J in Action, Aleksa Vukotic and Nicki Watt, Manning, 2015

Relationnel vs Neo4J

Bases de données relationnelles et bases graphes





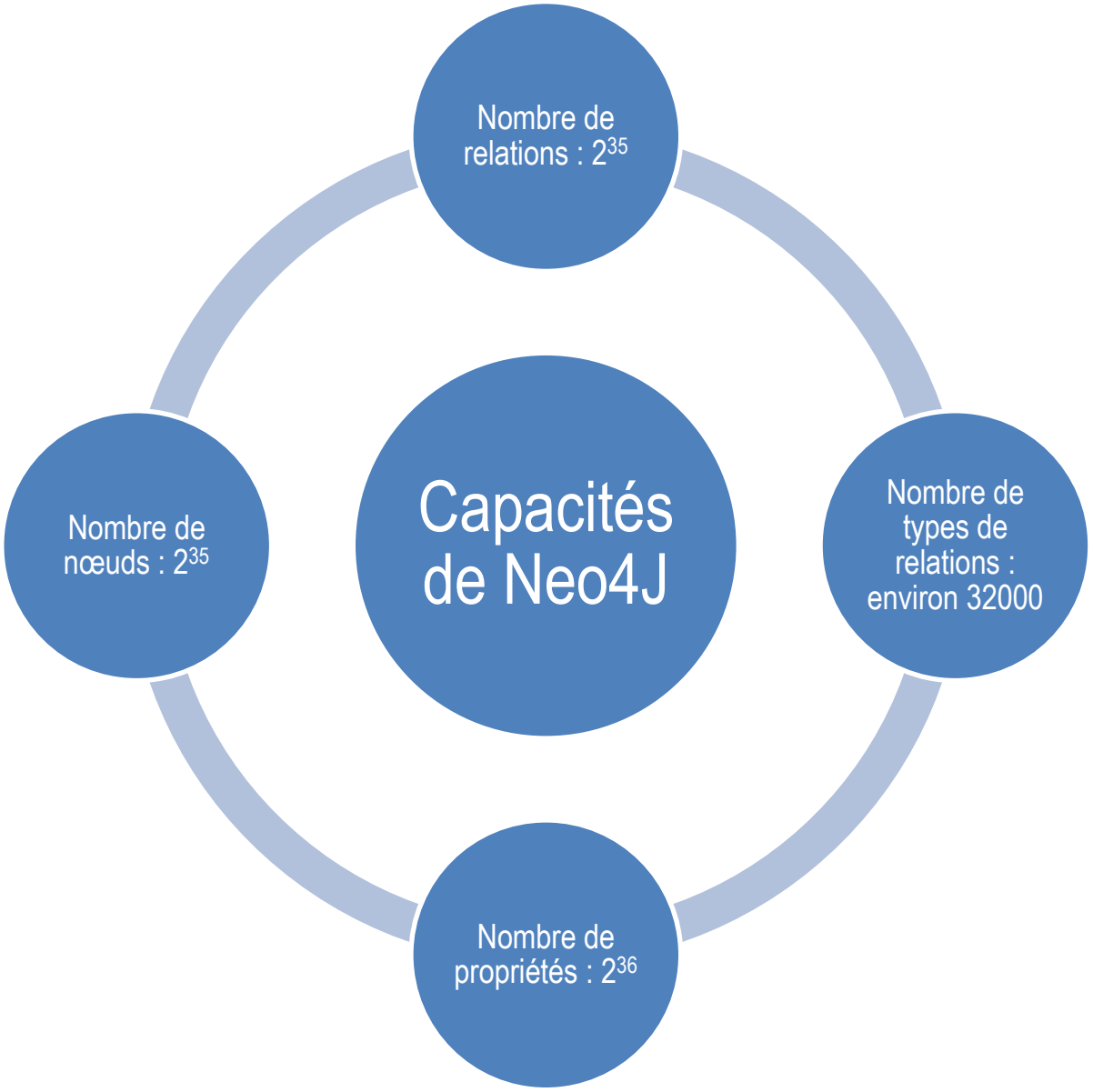
Source : http://blog.humancoders.com/interview-florent-biville-formateur-neo4j-pour-human-coders-formations-705/

En plus, Neo4j est ACID et transactonnel

- **Propriétés ACID**
  - **Atomicité** : opérations atomiques au niveau des sommets et des liens (arcs)
  - **Cohérence** : Neo4J étant transactionnelle, la cohérence est garantie
  - **Isolation** : garanti par le transactionnel
  - **Durabilité**: données écrites sur disque suite à une transaction

- **Propriété « transactionnel »**
  - Neo4J respecte toutes les caractéristiques d'une base de données transactionnelle

Capacités de Neo4J



Installation et première mise en œuvre

Installation

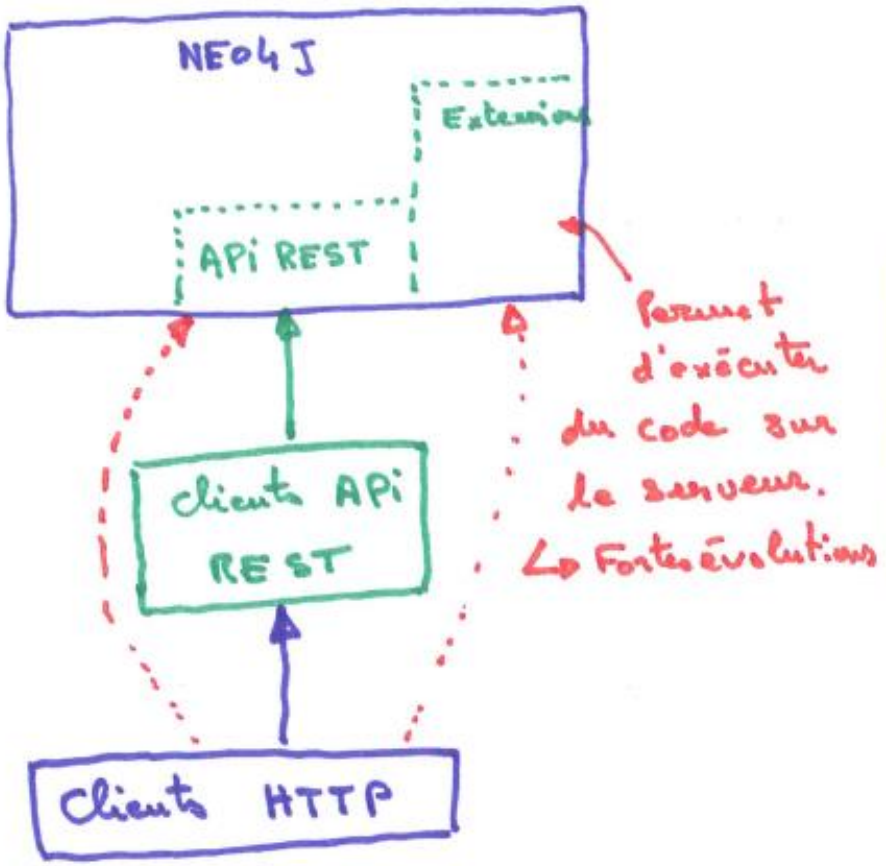
- **Ajout du dépôt officiel pour Ubuntu**
  - wget -O - https://debian.neo4j.org/neotechnology.gpg.key | sudo apt-key add -
  - echo 'deb http://debian.neo4j.org/repo stable/' | sudo tee /etc/apt/sources.list.d/neo4j.list
  - sudo apt-get update
- **Installation depuis le dépôt**
  - sudo apt-get install neo4j -y
- **Lancement**
  - sudo service neo4j start
- **Connection**
  - neo4j-shell
  - http://localhost:7474/browser/ ou http://192.168.56.4:7474/browser/

Administration et développements

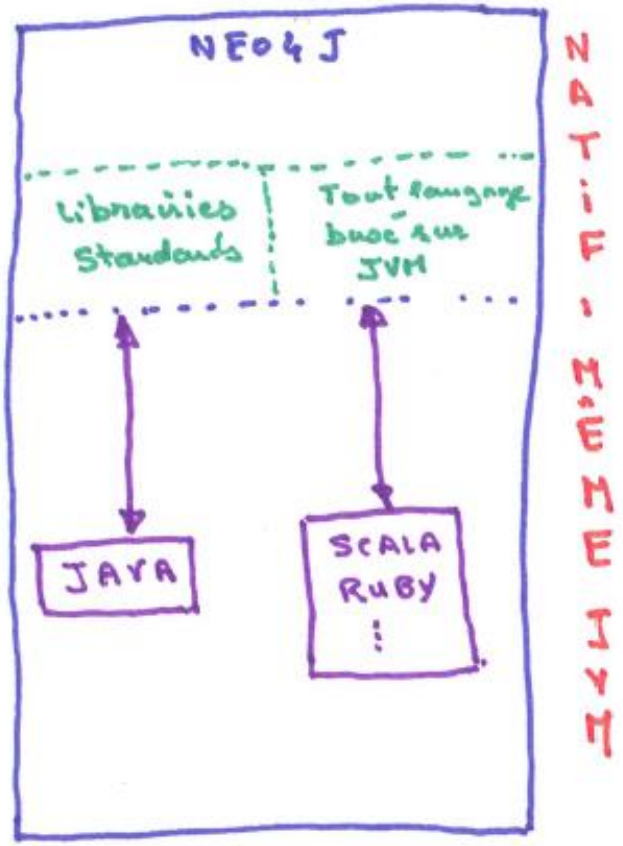
- **Vérifications et tests de quelques commandes**
  - Sur le système : ‘ps auxw | grep –i neo4j’
  - En shell :
    - ‘neo4j-shell’
    - ‘schema’
    - CTRL+D pour quitter
- **Installations pour Python**
  - pip install py2neo
  - pip install ipython-cypher
  - pip install python-igraph
  - pip install plotly
- **A ce stade, Neo4J est opérationnel ainsi que son environnement de développement**
- **2 possibilités de mise en œuvre : ‘embedded’ ou ‘server’**

Administration et développements

- 2 possibilités de mise en œuvre : 'embedded' ou 'server'



Architecture classique

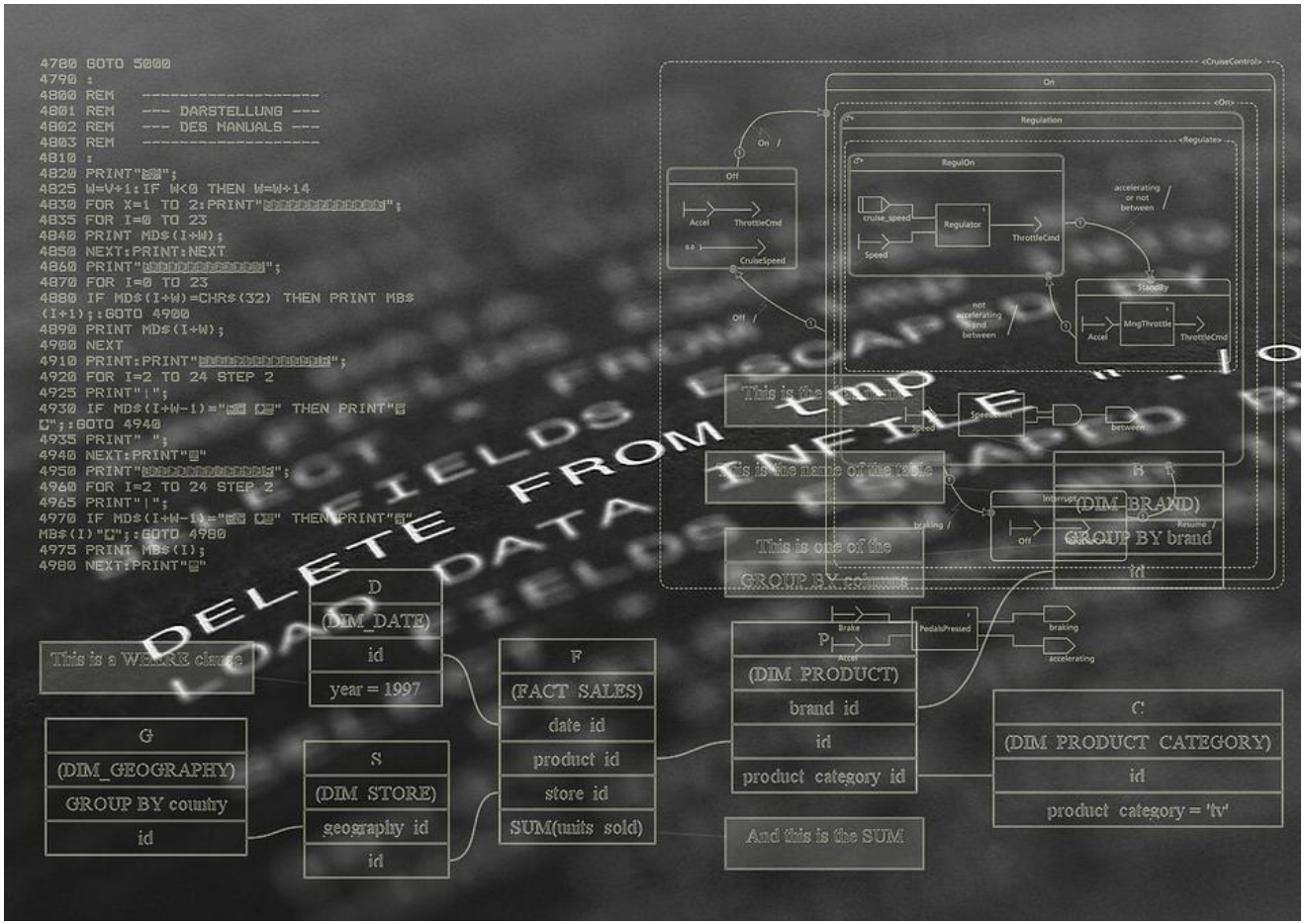


BD et traitements dans la même JVM

Administration et développements

- **2 possibilités de mise en œuvre : ‘embedded’ ou ‘server’**
  - Les extensions permettent d’exécuter du code sur le serveur (~procédures stockées)
  - Ces extensions sont soumises à fortes évolutions et investir de manière pérenne peut poser problème
  - L’idéal reste de séparer les concepts et d’utiliser les API REST avec les drivers par langage

Syntaxe Neo4J

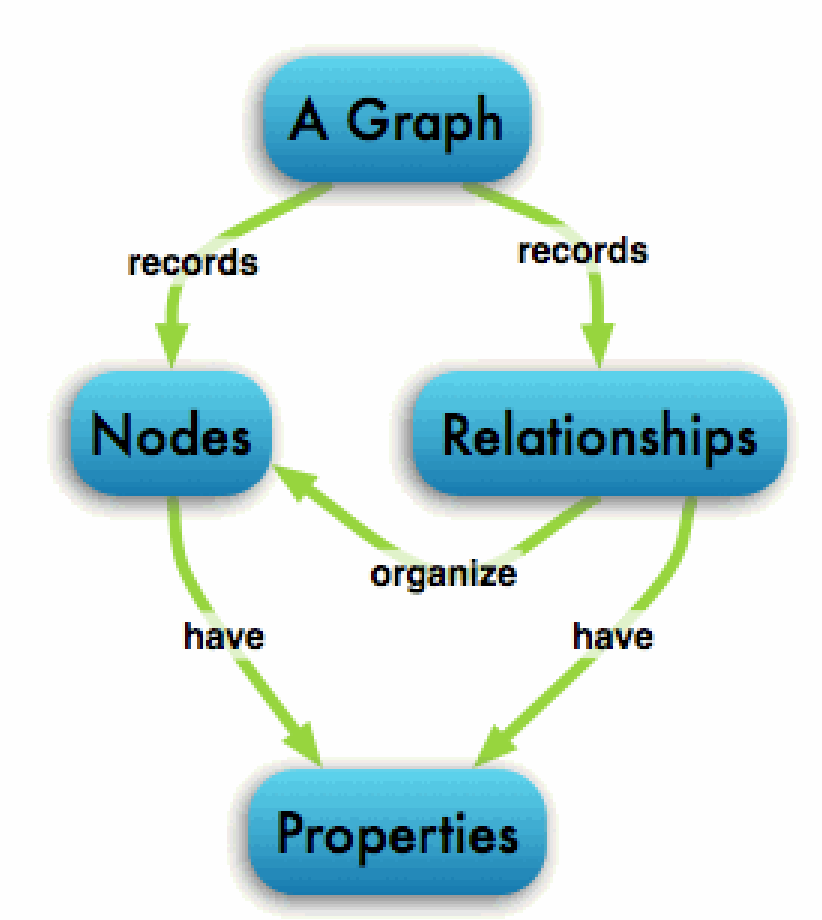


Syntaxe

- Le langage privilégié pour Neo4J est Cypher
- Neo4J a lancé il y a peu une initiative, OpenCypher, visant à définir un langage commun de requête pour les graphes (http://www.opencypher.org/)
- Cypher utilise de « l'ASCII art » et se veut un langage facilement appréhendable
- Cypher est un langage très riche et qui permet toutes les opérations sur les structures de graphes
- La « reference card » de Cypher témoigne de la richesse du langage : http://neo4j.com/docs/stable/cypher-refcard/
- Dans la suite, quelques exemples vont illustrer son utilisation
- Des alternatives existent
  - Gremlin (langage impératif en librairie OpenSource) : plus trop actif
  - https://github.com/tinkerpop/gremlin

Syntaxe

- Des principes simples



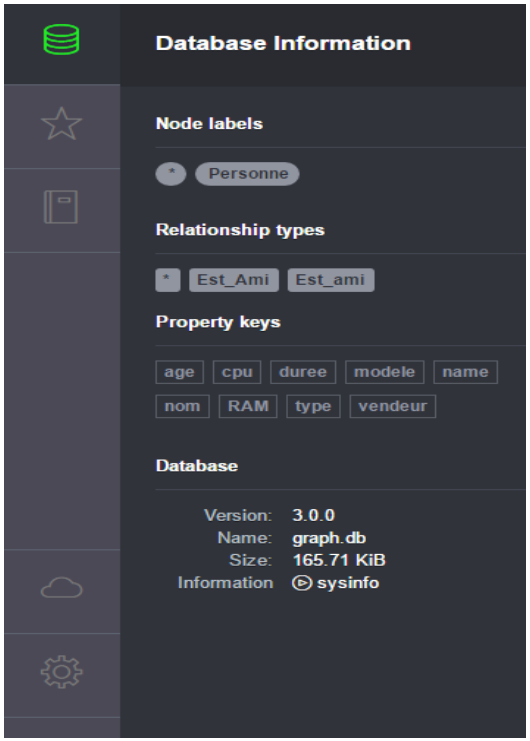
Source : http://assets.neo4j.org/img/propertygraph/graphdb-gve.png

Syntaxe

- **Un nœud**
  - Label
  - Des propriétés
- **Relation**
  - Nœuds de départ et d'arrivée
  - Label
  - Des propriétés
- **Les propriétés utilisent des types Java : int, String, boolean, etc.**

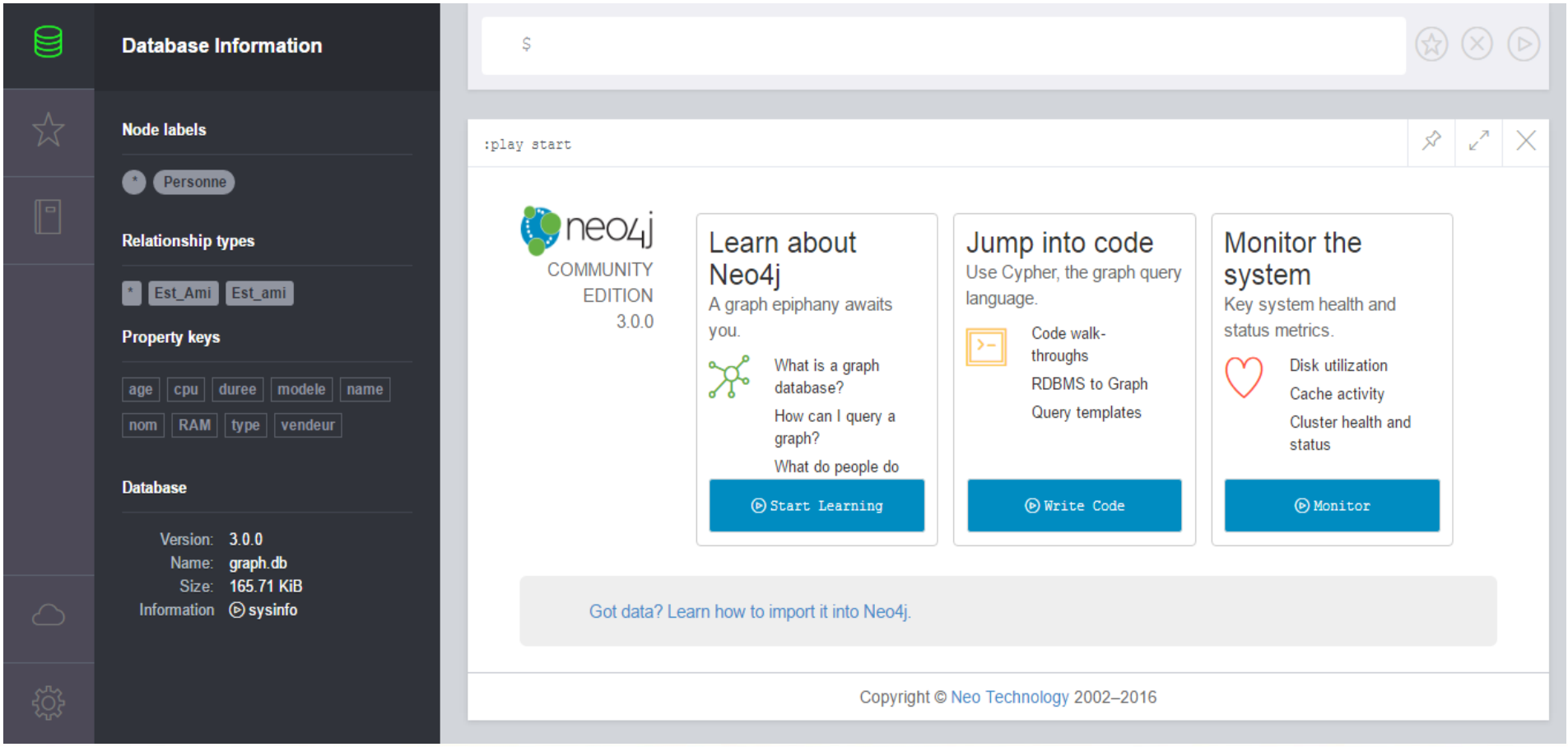
Avant de continuer : les outils

- **Il est toujours possible de travailler en mode Shell**
  - Mais cela devient vite fastidieux lorsque le problème devient complexe
- **Heureusement, il existe une interface, livrée et installée avec Neo4J**
  - L'accès, dans notre cas, se fait via l'URL : http://192.168.56.4:7474/browser/
- **Elle permet déjà de connaitre quelques informations sur l'instance Neo4J**



Avant de continuer : les outils

- **Elle permet aussi**
  - De saisir du code (prompt en haut à droite)
  - D'utiliser des didacticiels
  - De suivre l'activité du système

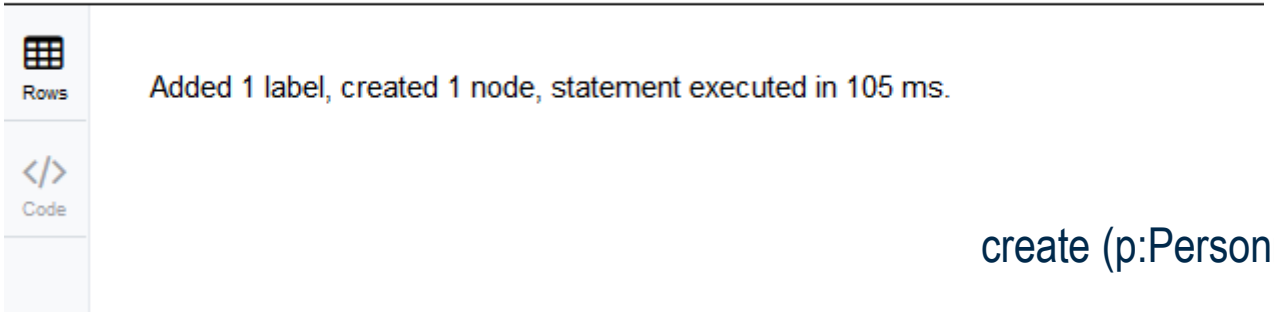


Syntaxe

Commençons par créer un nœud simple :

You can play !!

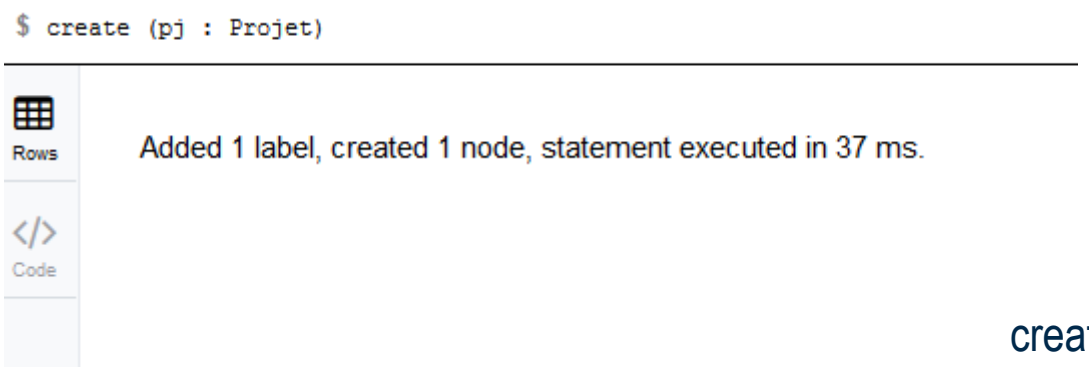
```
$ create (p : Personne)
```



create (p:Personne)

Créons un deuxième nœud :

```
$ create (pj : Projet)
```

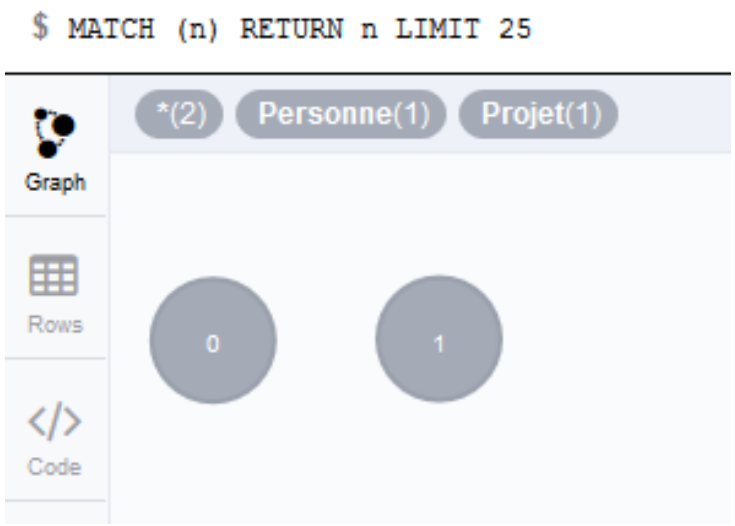


create (pj:Projet)

A ce stade, il existe 2 nœuds avec les labels Personne et Projet

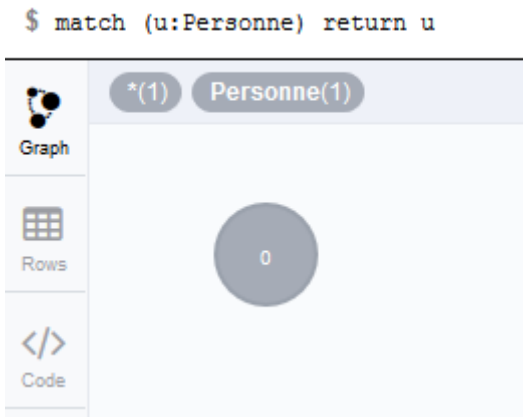
Syntaxe

Voyons le résultat graphique :



match (n) return n limit 25

Comment retrouver le nœud relatif à la personne :

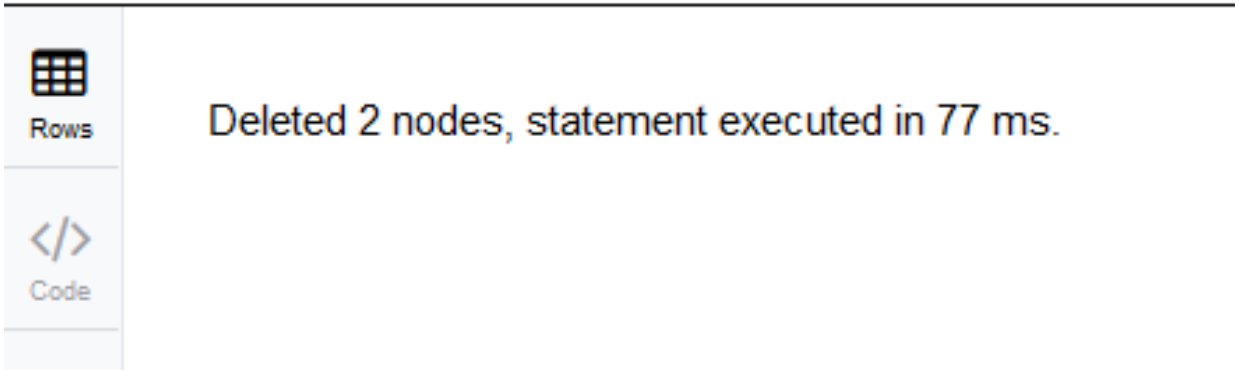


match (u:Personne) return u

Syntaxe

- **Tout détruire (à utiliser maintenant ;-)) :**

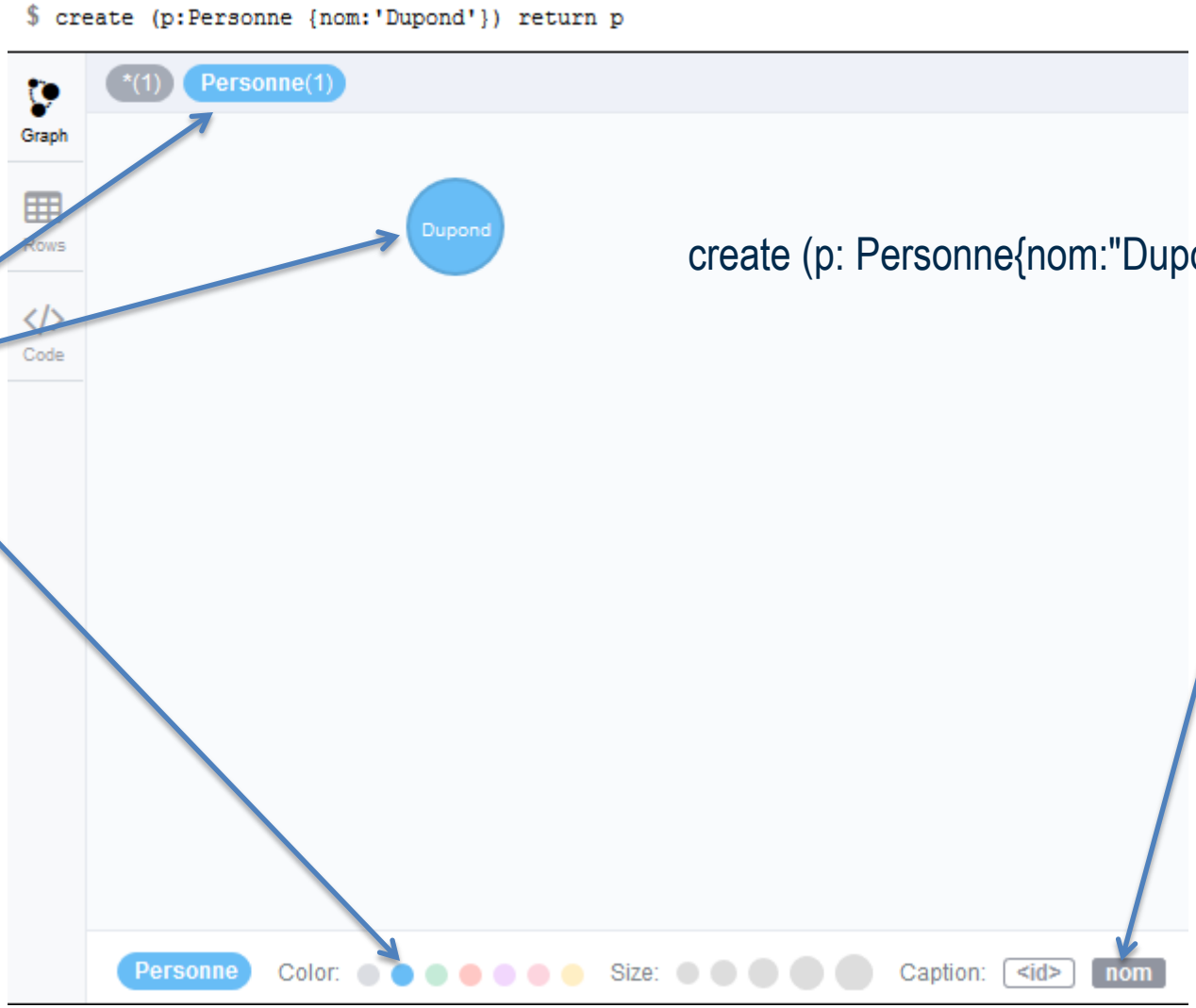
```
$ MATCH (n) OPTIONAL MATCH (n)-[r]-() DELETE n,r
```



match (n)optional match (n)-[r]-() delete n,r

Syntaxe

- Créons les mêmes nœuds mais un peu plus complexes :



create (p: Personne{nom:"Dupond"}) return p

Code couleur

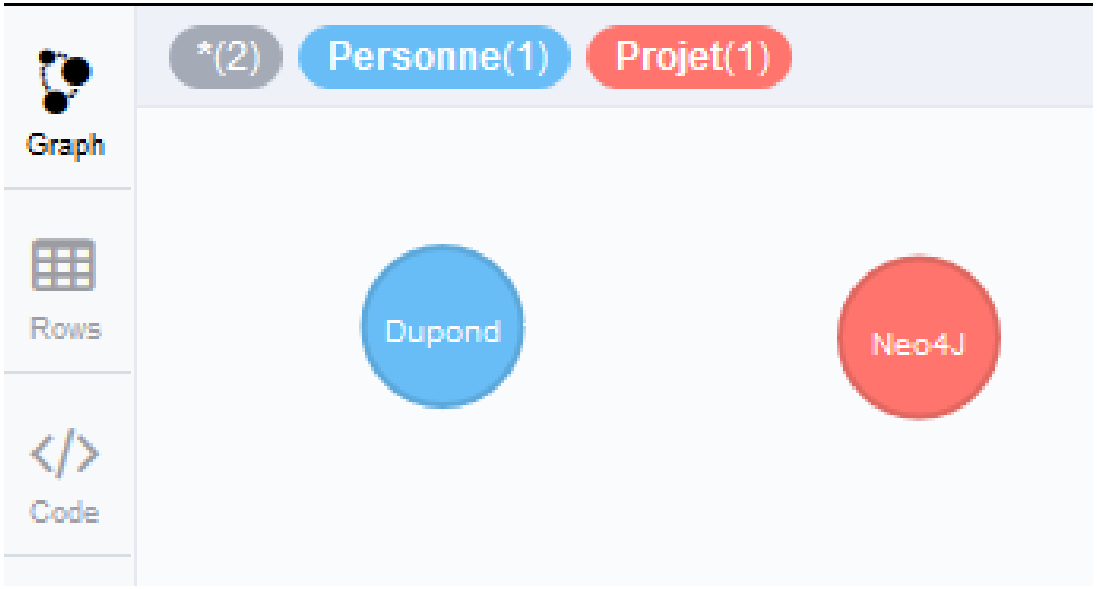
Libellé

Syntaxe

- **Même action pour le projet :**

Match permet de faire un select ….

```
$ MATCH (n) RETURN n LIMIT 25
```

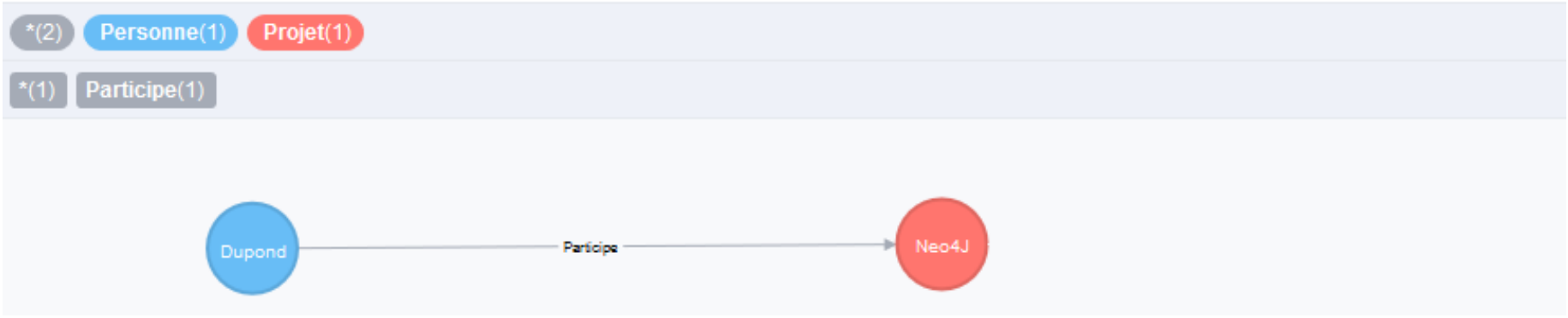



```
create (pj:Projet{nom:"neo4j"})
match (n) return n limit 25
```

Syntaxe

Créons maintenant une relation entre ‘p’ et ‘pj’ :

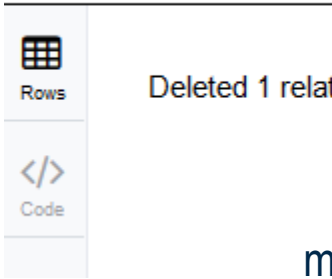
```
$ match (p:Personne {nom:'Dupond'}),(pj:Projet {nom:'Neo4J'}) create (p)-[relation:Participe]->(pj) return relation,p,pj
```



match (p:Personne{nom:'Dupond'}),(pj:Projet{nom:'neo4j'}) create (p)-[relation:Participe]->(pj) return relation, p,pj

Détruisons cette relation entre ‘p’ et ‘pj’ :

```
$ match (p:Personne {nom:'Dupond'})-[r:Participe]->(pj:Projet {nom:'Neo4J'}) delete r
```

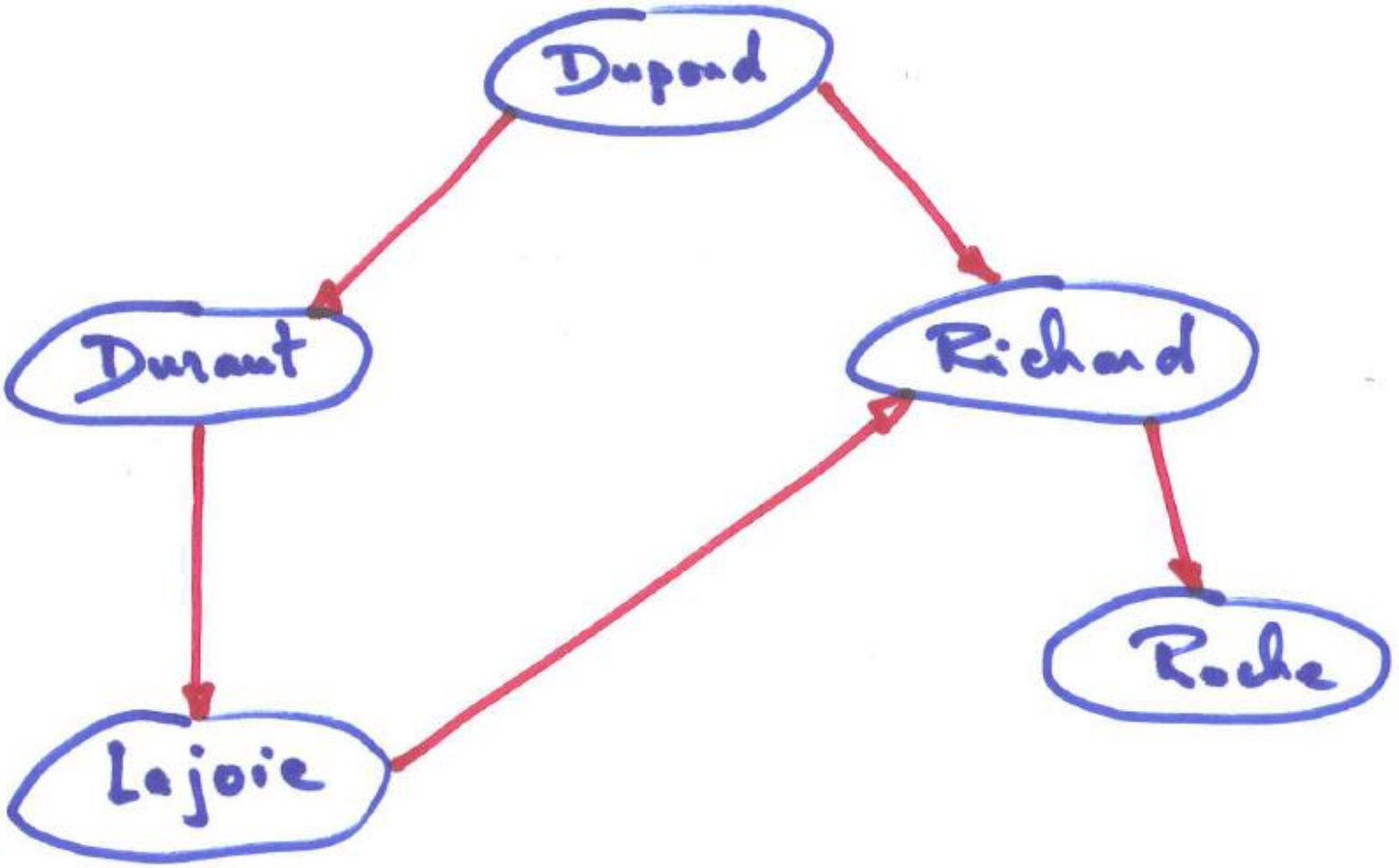


Deleted 1 relationship, statement executed in 511 ms.

match (p:Personne{nom:'Dupond'})-[r:Participe]->(pj:Projet{nom:'neo4j'}) delete r

Syntaxe

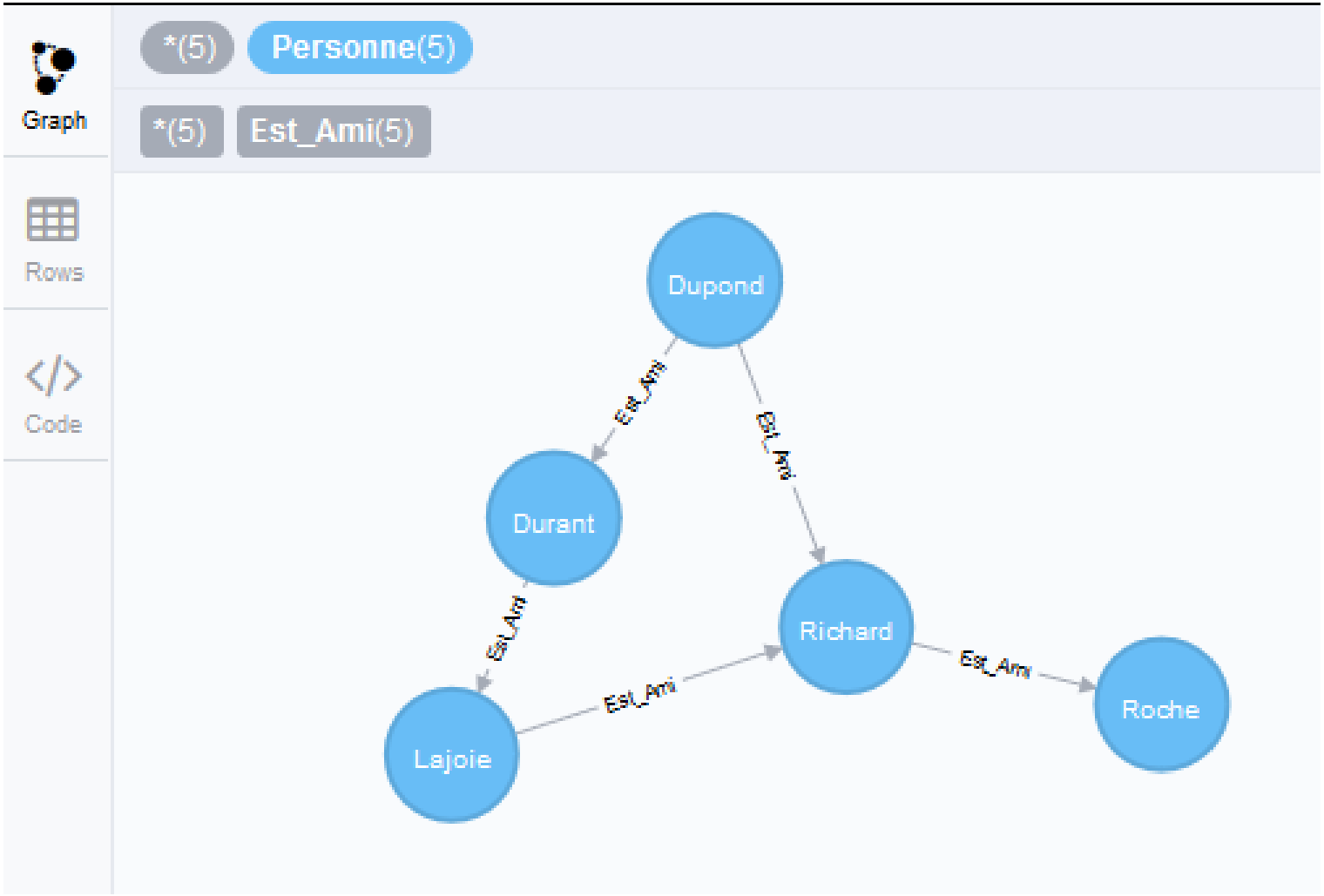
- Créons maintenant le graphe qui a servi d'exemple SQL



Syntaxe

- Cela donne :

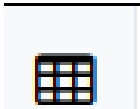
```
$ MATCH (n:Personne) RETURN n LIMIT 25
```



Syntaxe

Qui est ami avec Dupond ?

```
$ match (p:Personne)-[:Est_Ami]->(amis) where p.nom='Dupond' return amis.nom
```



amis.nom
Richard
Durant

```
match (p:Personne)-[:Est_ami]->(amis:Personne) where p.nom='Dupond' return amis.nom
```

Ce sont les amis de « premier niveau »

Syntaxe

Qui est ami avec Dupond ?

```
$ match (p:Personne)-[:Est_Ami*2]->(amis) where p.nom='Dupond' return amis.nom
```

amis.nom
Lajoie
Roche

Rows

Code

La subtilité est ici !

```
match (p:Personne)-[:Est_ami*2]->(amis:Personne) where p.nom='Dupond' return amis.nom
```

Ce sont les amis de « deuxième niveau »

Syntaxe

- Quels sont tous les amis de Dupond (premier et deuxième niveau) ?

```
$ match (p:Personne)-[:Est_Ami*1..2]->(amis) where p.nom='Dupond' return amis.nom
```

Rows

Code

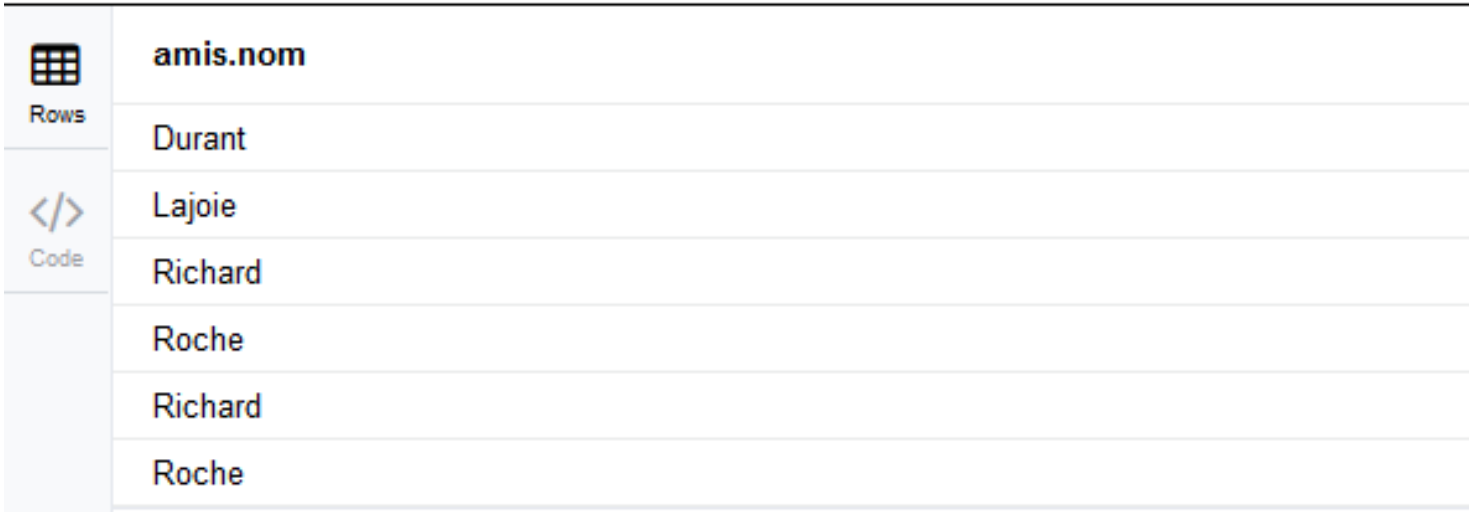
amis.nom
Durant
Lajoie
Richard
Roche

match (p:Personne)-[:Est_ami*1..2]->(amis:Personne) where p.nom='Dupond' return amis.nom

Syntaxe

Quels sont tous les amis de Dupond ?

```
$ match (p:Personne)-[:Est_Ami*]->(amis) where p.nom='Dupond' return amis.nom
```



match (p:Personne)-[:Est_ami*]->(amis:Personne) where p.nom='Dupond' return amis.nom

Il existe des doublons qui sont éliminés simplement :

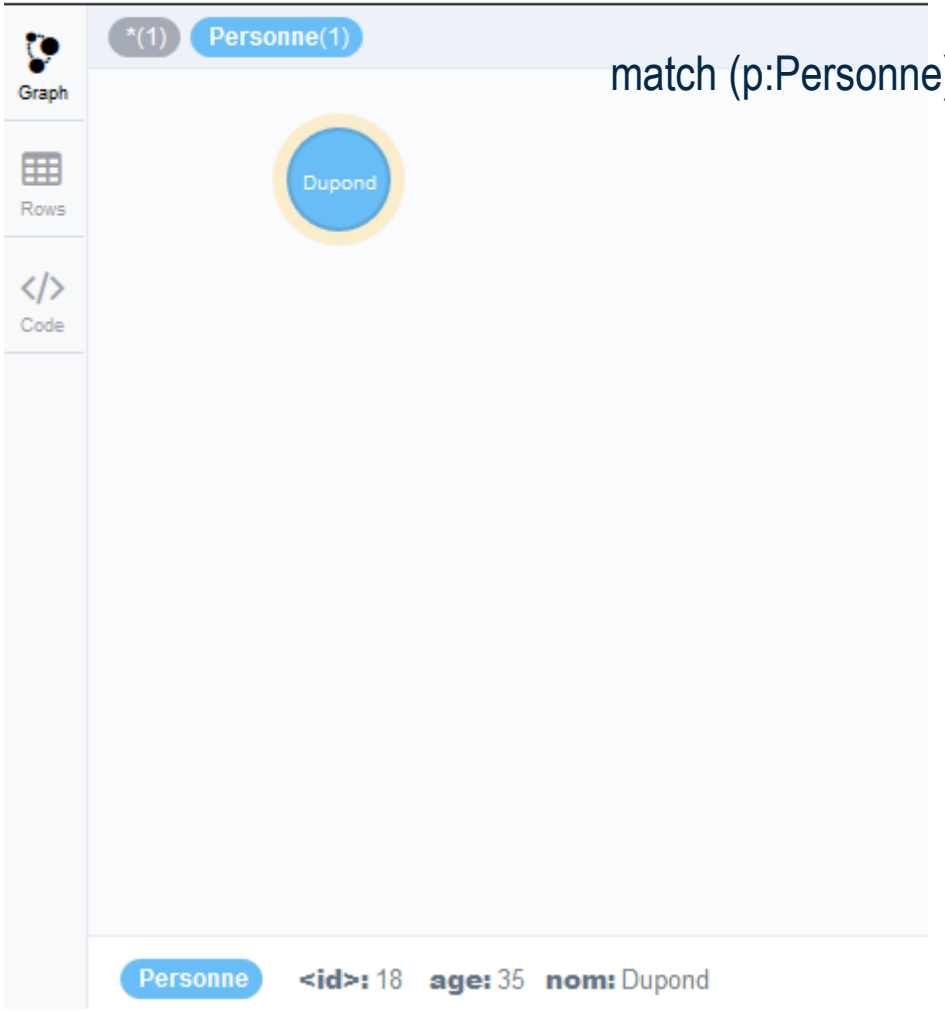
```
$ match (p:Personne)-[:Est_Ami*]->(amis) where p.nom='Dupond' return distinct amis.nom
```

match (p:Personne)-[:Est_ami*]->(amis:Personne) where p.nom='Dupond' return distinct amis.nom

Syntaxe

Ajoutons quelques propriétés (nœuds) :

```
$ match (p:Personne) where p.nom='Dupond' set p.age=35 return p
```



match (p:Personne)where p.nom='Dupond' set p.age=35 return p

Syntaxe

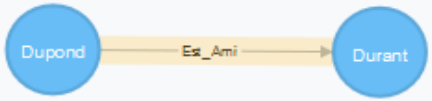
Ajoutons quelques propriétés (relations) :

```
$ match (p1:Personne)-[r:Est_Ami]->(p2:Personne) where p1.nom='Dupond' and p2.nom='Durant' set r.duree=30 return r
```

*(2) Personne(2)

*(1) Est_Ami(1)

match (p1:Personne)-[r:Est_ami]->(p2:Personne) where p1.nom='Dupond' and p2.nom='Durand' set r.duree=30 return r

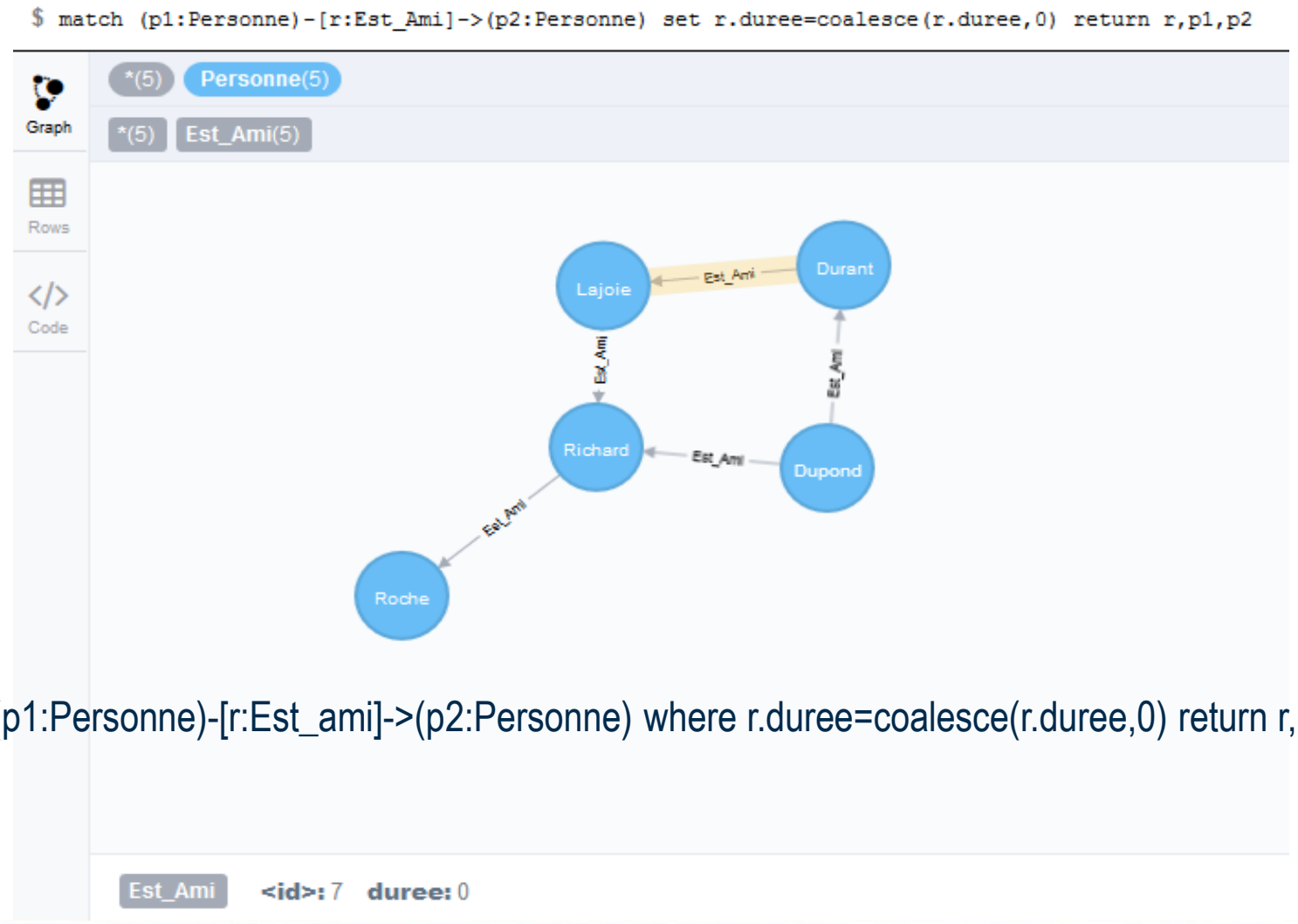


Des amis de 30 ans ….

Est_Ami <id>: 6 duree: 30

Syntaxe

- **Mettons à zéro cette propriété pour toutes les autres relations qui n'ont pas de valeurs:**



match (p1:Personne)-[r:Est_ami]->(p2:Personne) where r.duree=coalesce(r.duree,0) return r,p1,p2

Syntaxe

Sélectionnons les amitiés de plus de 10 ans :

```
$ match (p1:Personne)-[r:Est_Ami]->(p2:Personne) return p1,p2,r
```

Graph | Rows | Code

p1	p2	r
age 35 nom Dupond	nom Richard	duree 3
age 35 nom Dupond	nom Durant	duree 30
nom Durant	nom Lajoie	duree 24
nom Richard	nom Roche	duree 6
nom Lajoie	nom Richard	duree 28

Returned 5 rows in 438 ms.

match (p1:Personne)-[r:Est_ami]->(p2:Personne) return r,p1,p2

Syntaxe

Sélectionnons les amitiés de plus de 10 ans :

```
$ match (p1:Personne)-[r:Est_Ami]->(p2:Personne) where r.duree >=10 return r,p1,p2
```

Graph | Rows | Code

r	p1	p2
duree 30	age 35 nom Dupond	nom Durant
duree 24	nom Durant	nom Lajoie
duree 28	nom Lajoie	nom Richard

Returned 3 rows in 1515 ms.

match (p1:Personne)-[r:Est_ami]->(p2:Personne) where r.duree >= 10 return r,p1,p2

Syntaxe

Comptons, par personne, le nombre d'amitiés :

```
$ match (p1:Personne)-[r:Est_Ami]->(:Personne) with p1,count(r) as nombre order by nombre desc return p1,nombre
```

p1	nombre
age 35 nom Dupond	2
nom Lajoie	1
nom Durant	1
nom Richard	1

match (p1:Personne)-[r:Est_ami]->(:Personne) with p1, count(r) as nombre
order by nombre desc return p1,nombre

Syntaxe

Classement par la moyenne des durées d'amitiés :

```
$ match (p1:Personne)-[r:Est_Ami]->(:Personne) with p1,avg(r.duree) as moy_duree order by moy_duree desc return p1,moy_duree
```

p1	moy_duree
nom Lajoie	28
nom Durant	24
age 35 nom Dupond	16.5
nom Richard	6

match (p1:Personne)-[r:Est_ami]->(:Personne) with p1, avg(r.duree) as moy_duree order by moy_duree desc return p1,moy_duree

Syntaxe : récapitulatif

- **Les bases**
  - Les nœuds sont représentés par : ()
    - Il est aussi possible de spécifier : (monNoeud:monLabel)
  - Les relations entre nœuds sont représentées par : --[]->
    - Il est aussi possible de spécifier : -[maRelation:mon_Type]->

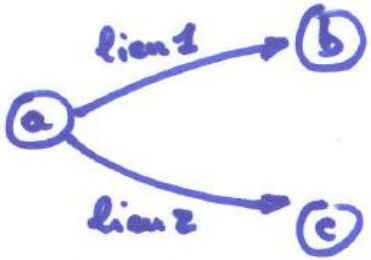
Syntaxe : récapitulatif

- Quelques exemples de graphes simples

Simple

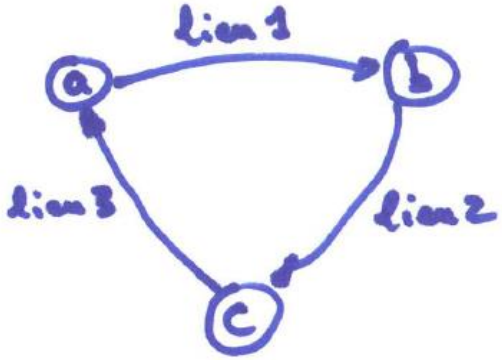
(a)-[:lien]->(b)

Double



(a)-[:lien1]->(b), (a)-[:lien2]->(c)

Cycle



(a)-[:lien1]->(b)-[:lien2]->(c)-[:lien3]->(a)

Note : la relation est dans les 2 sens

()-[:lien]->() ou ()<-[:lien]-()

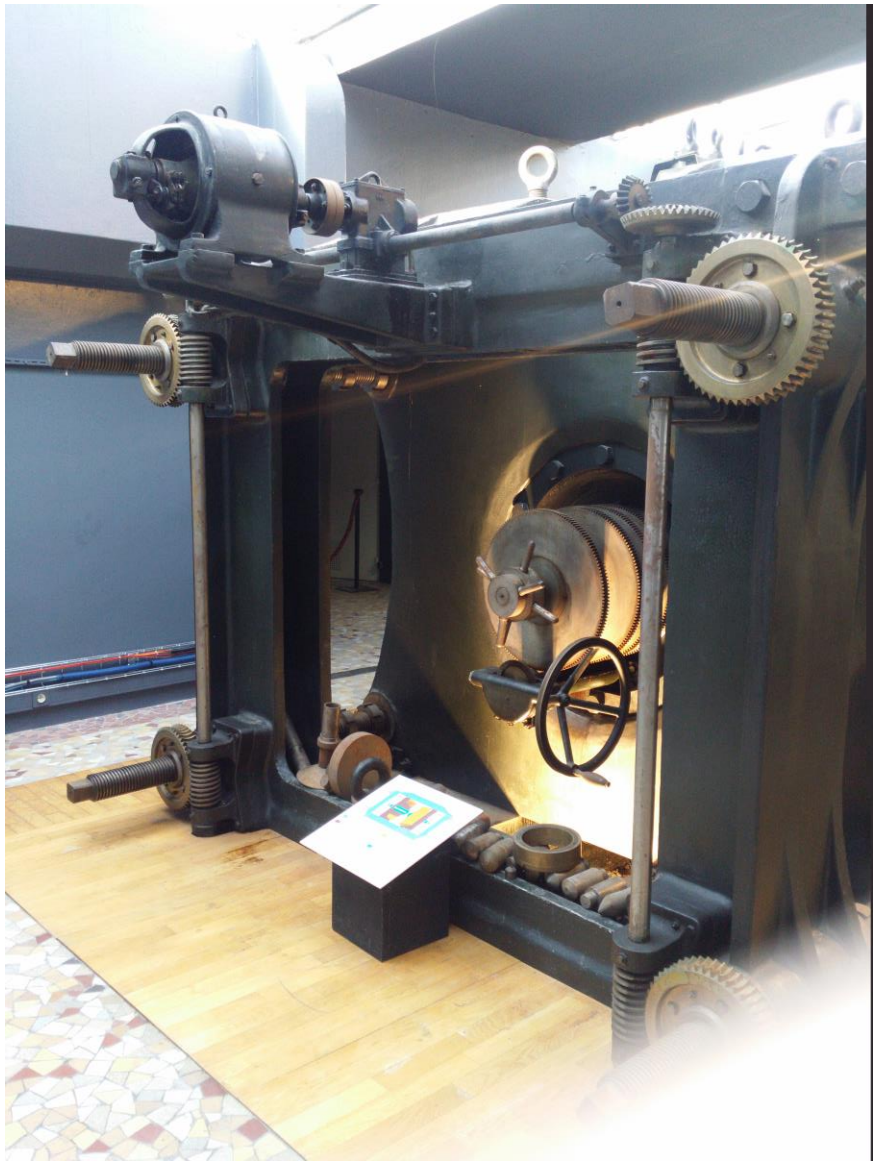
Syntaxe : récapitulatif

- **Il y a ensuite des opérations de base**
  - Match/where :
    - MATCH (n)-->(m) WHERE NOT (m)-->()
  - Return : identique au 'select' en SQL
    - RETURN personne.nom as userName ORDER BY userName LIMIT 3
  - With : identique à Return mais avec la capacité de poursuivre le traitement
    - MATCH (n:Personne)-[:Ami]-(a) WITH count(a) as s, n MATCH (n)-[:Ami]-()-[:Ami]-(fof) RETURN n, s, fof
  - Create
    - CREATE (n1)-[:Lien]->(n2)
  - Agrégation : count, sum, avg, …
    - MATCH (n1)-->(n2) RETURN n, count(1)
  - Etc.
- **Le langage subit quand même de fortes évolutions**
  - Il existe même la possibilité la version de Cypher qui doit être utilisée
  - CYPHER 1.8 MATCH (p)-[r]->() ….

Mode de parcours du graphe

- **Dans les graphes, il existe 2 modes de parcours**
  - En profondeur d'abord
  - En largeur d'abord
- **Avec Cypher, le mode du langage étant déclaratif, le mode de parcours est totalement délégué**
  - Il n'y a pas à s'en préoccuper
- **Dans un langage impératif, il y a possibilité de spécifier le mode de parcours du graphe**
  - Trouvé en Java
    - TraversalDescription traversalDescription = graphDb.traversalDescription() .depthFirst()
  - Pas en Python …
    - Il faudra compter sur l'harmonisation des API

Manips en cours …

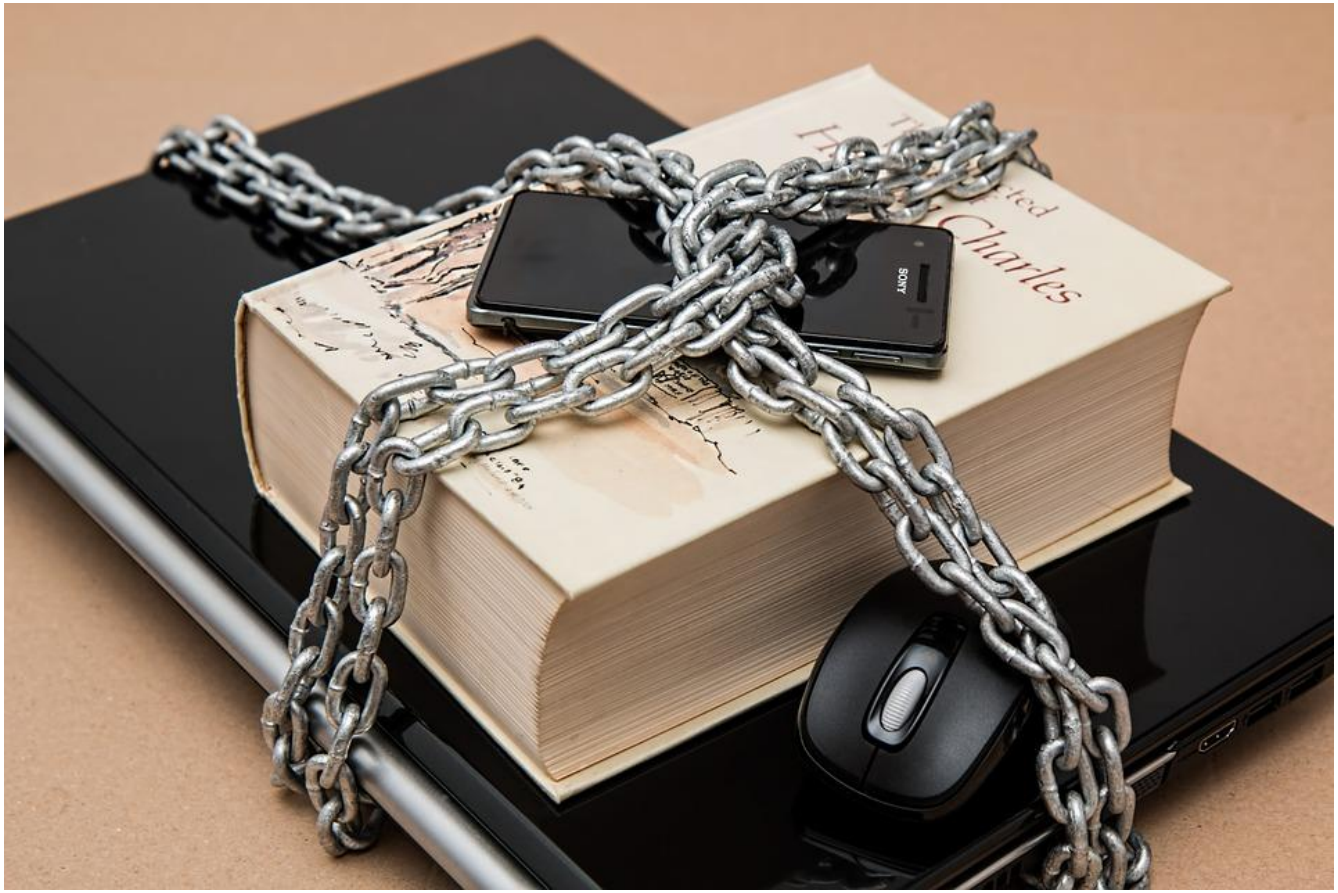


Electro-aimant Meudon – Nov. 2015

Objectifs et outils

- **Poursuite de l'exemple**
  - Les amis font connaissance ;-)
  - Utilisation de Cypher (interface Neo4J ou Jupyter)
  - Programmation en Python via Jupyter

La sécurité



La sécurité

- **La sécurité est assez peu intégrée dans Neo4J (comme dans MongoDB)**
- **Neo4J est basé sur des API REST et tout le problème consiste donc à sécuriser ces API (accès HTTP)**
  - Py2neo permet est basé sur les API REST et en cache la complexité au développeur
- **Il est possible de définir nativement des comptes et des mots de passe mais des accès restent basiques (granularité des accès)**
- **Cette option peut paraître « osée » dans le contexte sécurité actuel**
  - Mais performances et sécurité font rarement bon ménage …
- **Il faut donc sécuriser les accès via des moyens externes à Neo4J**

Première étape : les sauvegardes

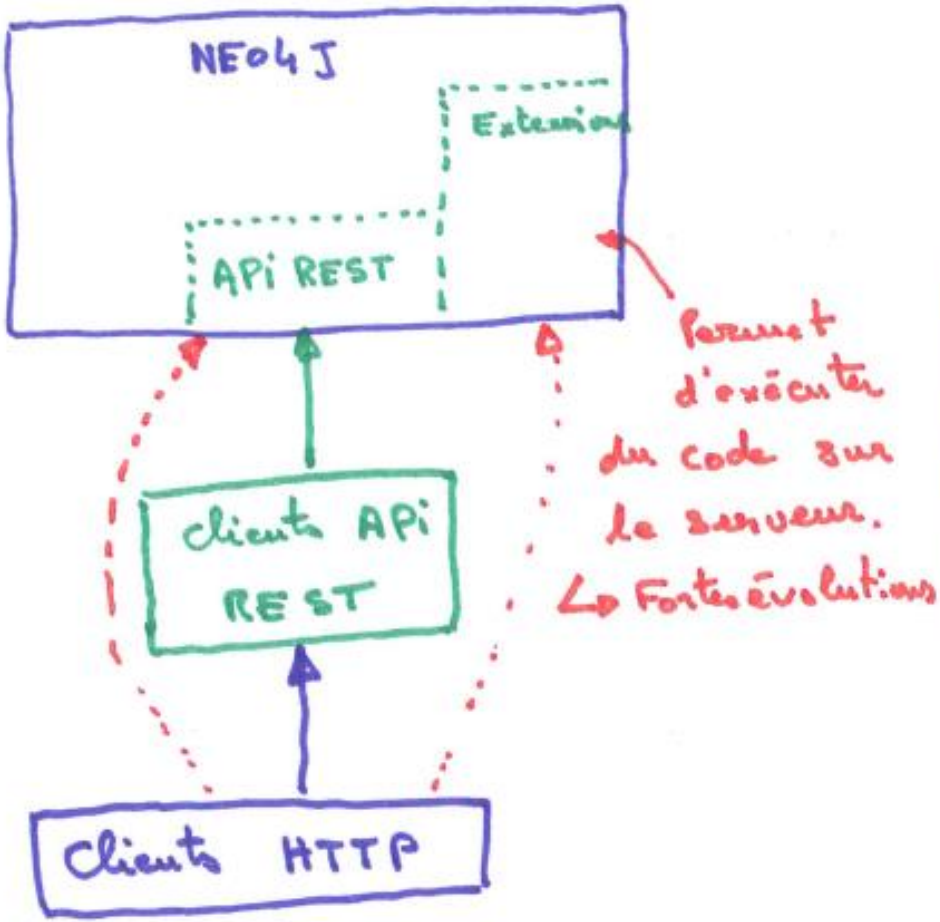
- **Méthode 1: copie des fichiers de la base**
  - cp -R data/graph.db data/backup
- **Méthode 2: utiliser la commande de sauvegarde**
  - http://neo4j.com/docs/stable/backup-embedded-and-server.html
  - mkdir /…./backup/neo4j-backup
  - neo4j-backup -host 192.168.56.1 -to /…./backup/neo4j-backup
- **Méthode 3: utiliser la commande Cypher Dump**
  - neo4j-shell -c 'dump MATCH (n) OPTIONAL MATCH (n)-[r]-() RETURN r,n' > export.cql
  - Pour réimporter le graphe (après avoir détruit ce qui existait auparavent): neo4j-shell -file export.cql
- **Peu de méthodes natives éprouvées …**

La sécurité : les accès

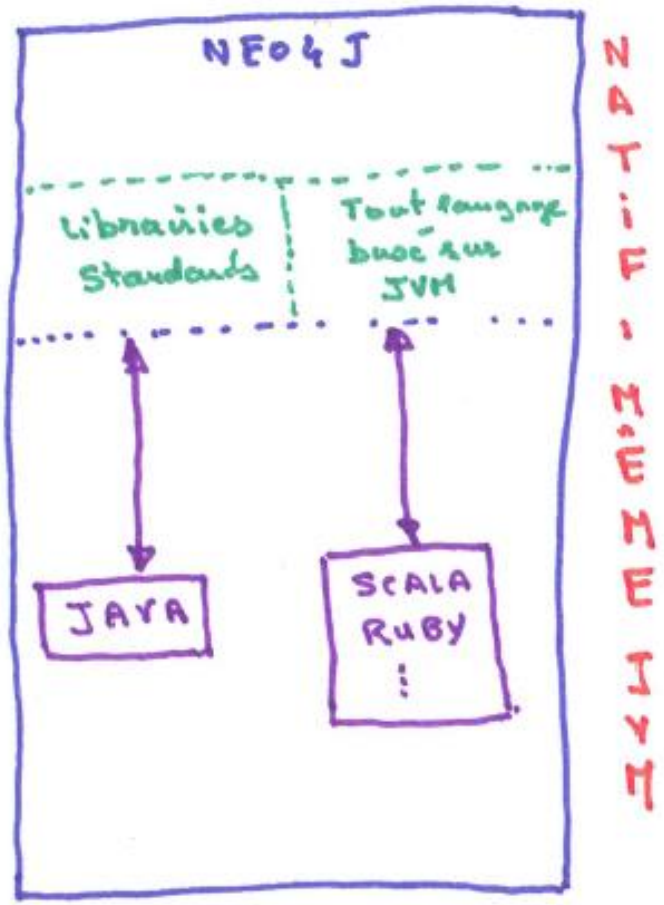
- **Les moyens de sécurisation de base**
  - Sécuriser l'accès à Neo4J revient à sécuriser des API REST
  - Utilisation de proxy (HA-Proxy par exemple) : http://www.haproxy.org/
  - Utilisation de SSL/TLS
  - Restreindre si possible l'accès à un domaine voire à un sous ensemble défini d'applications clientes
  - Positionner Neo4J dans un réseau dédié
  - Tracer, logger, consigner, filtrer, exploiter, ….

Architecture

Rappel : Embedded vs Server

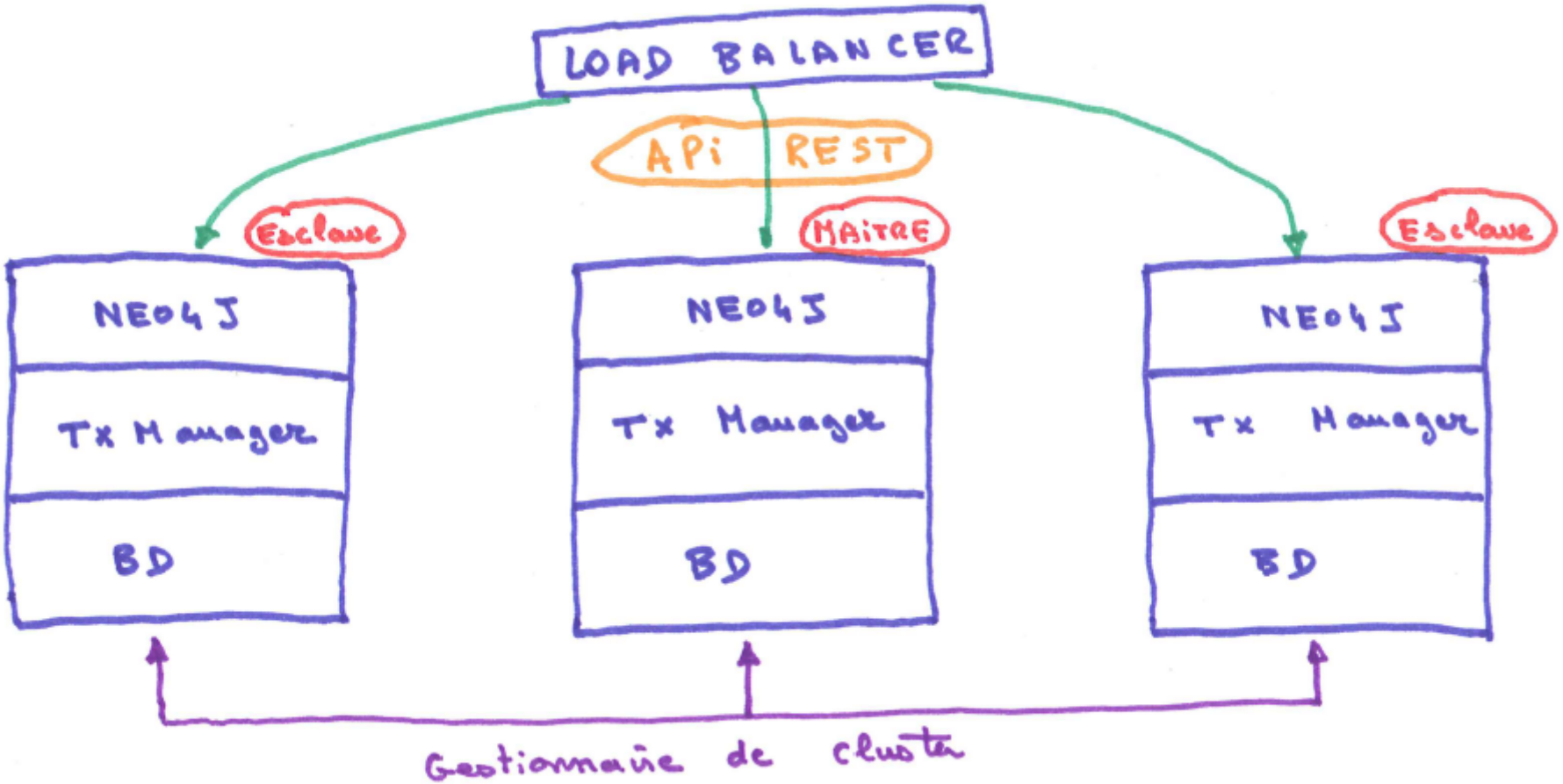


Architecture classique



BD et traitements dans la même JVM

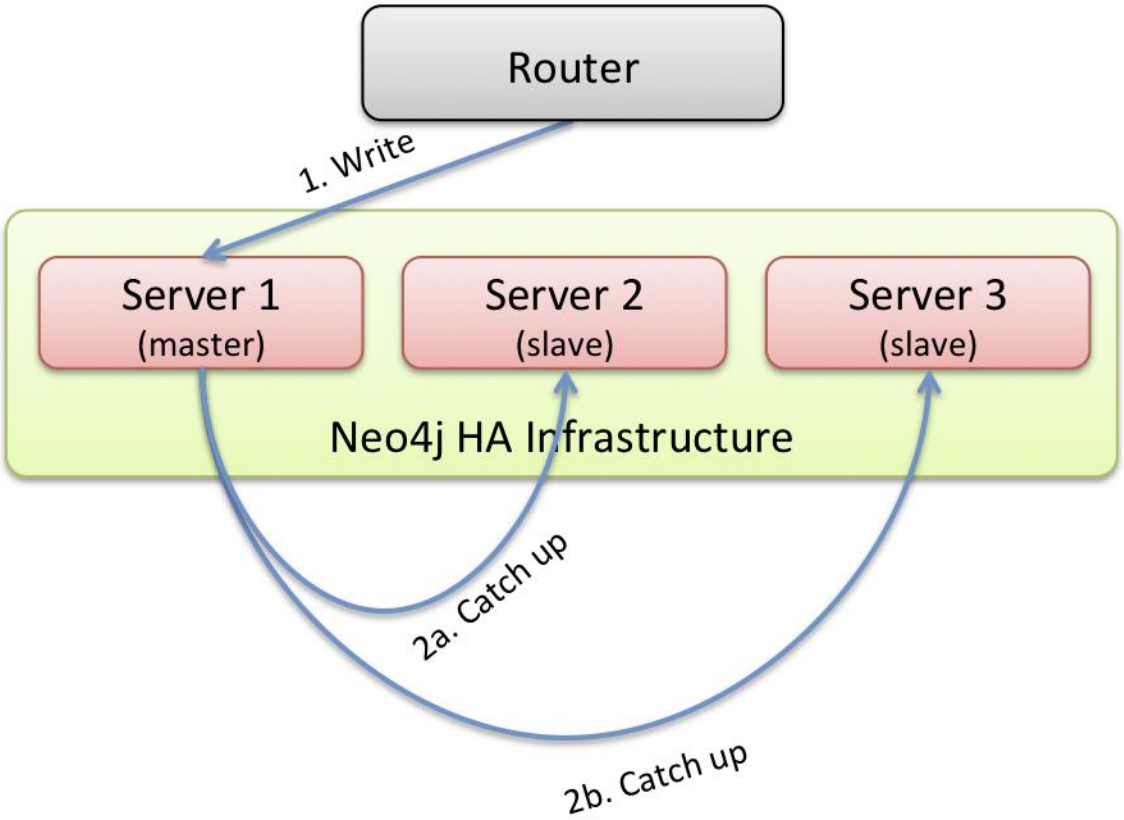
Mise en œuvre haute-disponibilité



Mise en œuvre haute-disponibilité

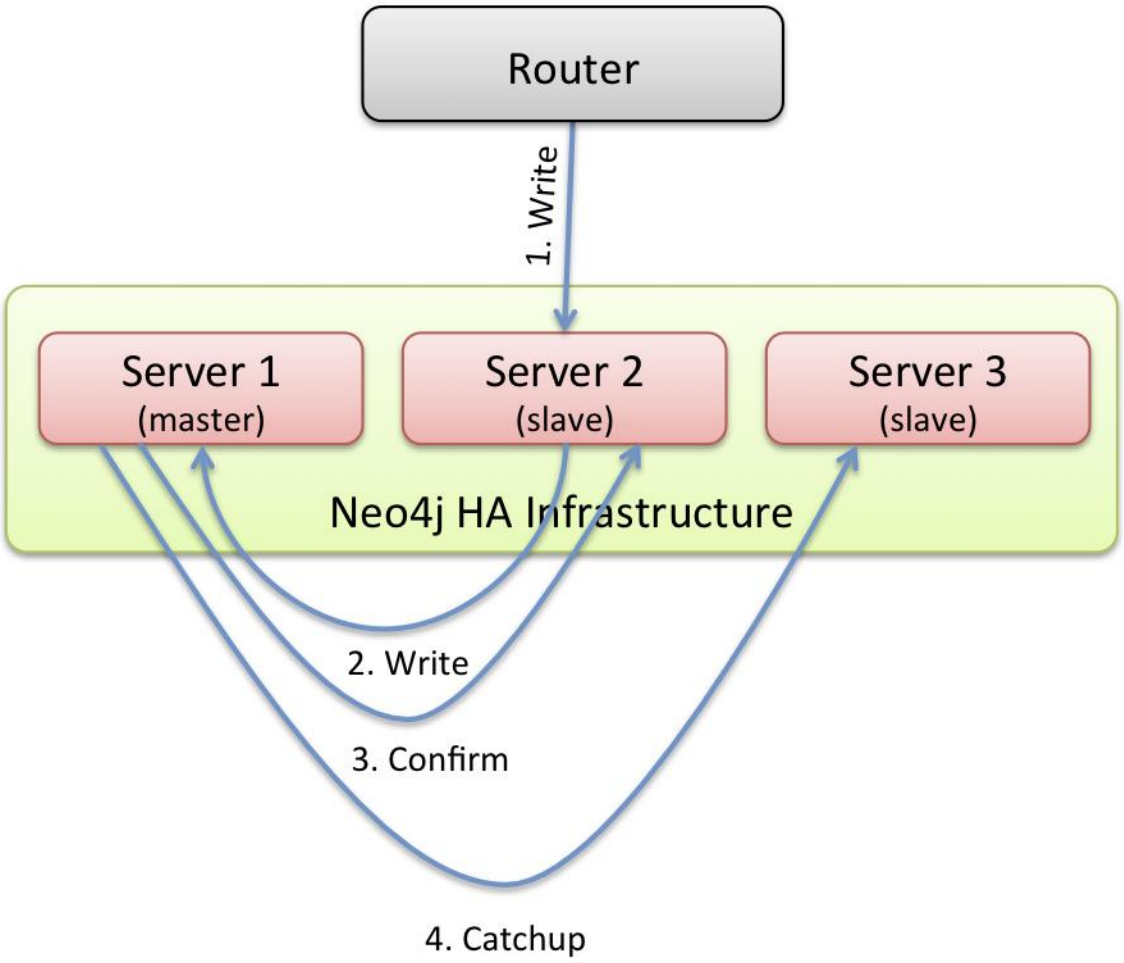
- **Cette configuration nécessite la version Enterprise**
- **La mise en œuvre n'est pas très complexe mais nécessite de la ressource**
  - Intel Core I7 recommandé
  - Filesystem Ext4
  - 8 à 12 Go de RAM minimum (par nœud)
  - OpenJDK 8 recommandé
- **Voici l'URL de référence : http://neo4j.com/blog/graphs-to-production-at-scale/**
  - Contient même la configuration de Ha-Proxy

La haute-disponibilité et la lecture/écriture



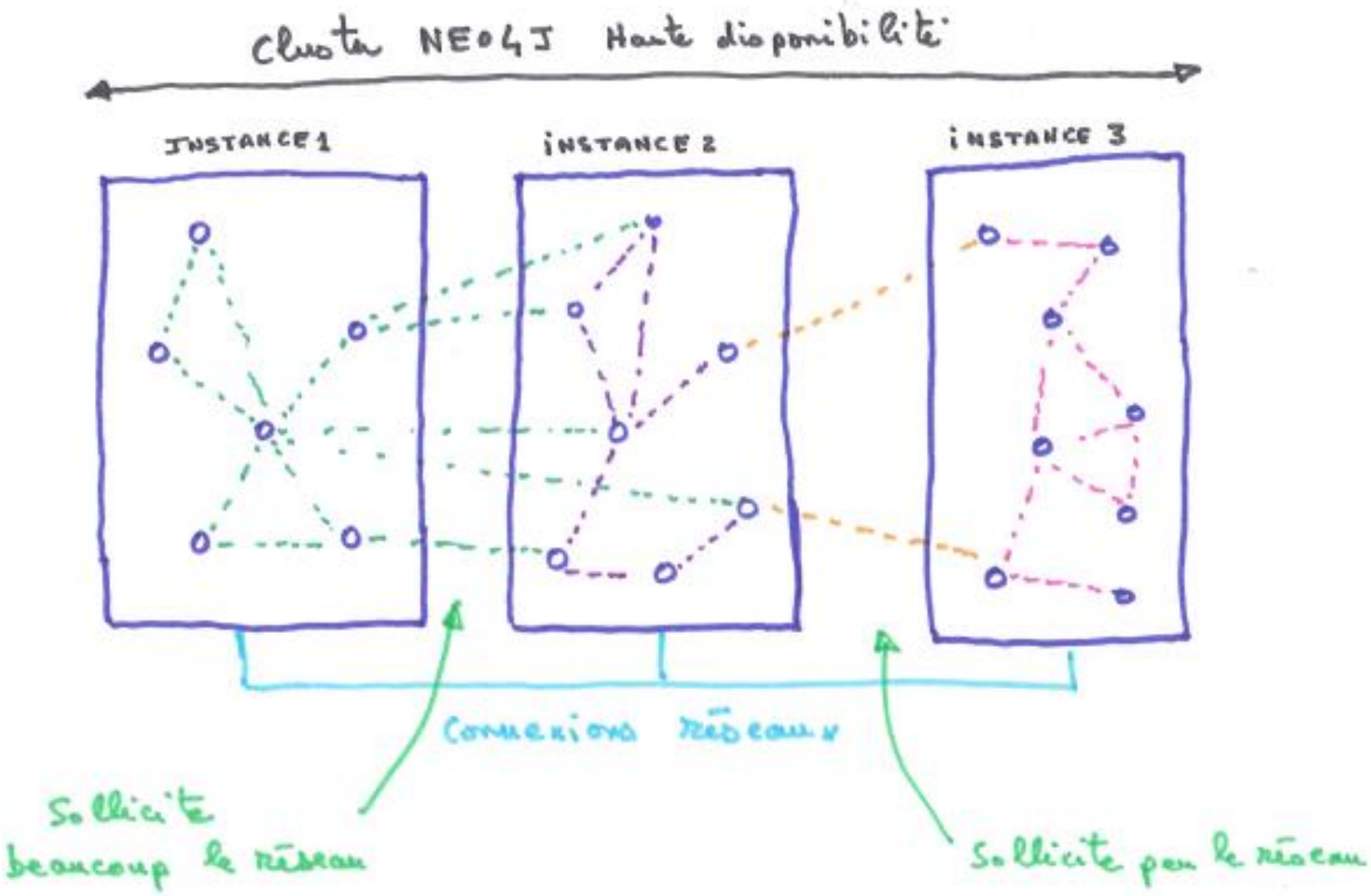
Source : http://jimwebber.org/2011/02/scaling-neo4j-with-cache-sharding-and-neo4j-ha//

La haute-disponibilité et la lecture/écriture

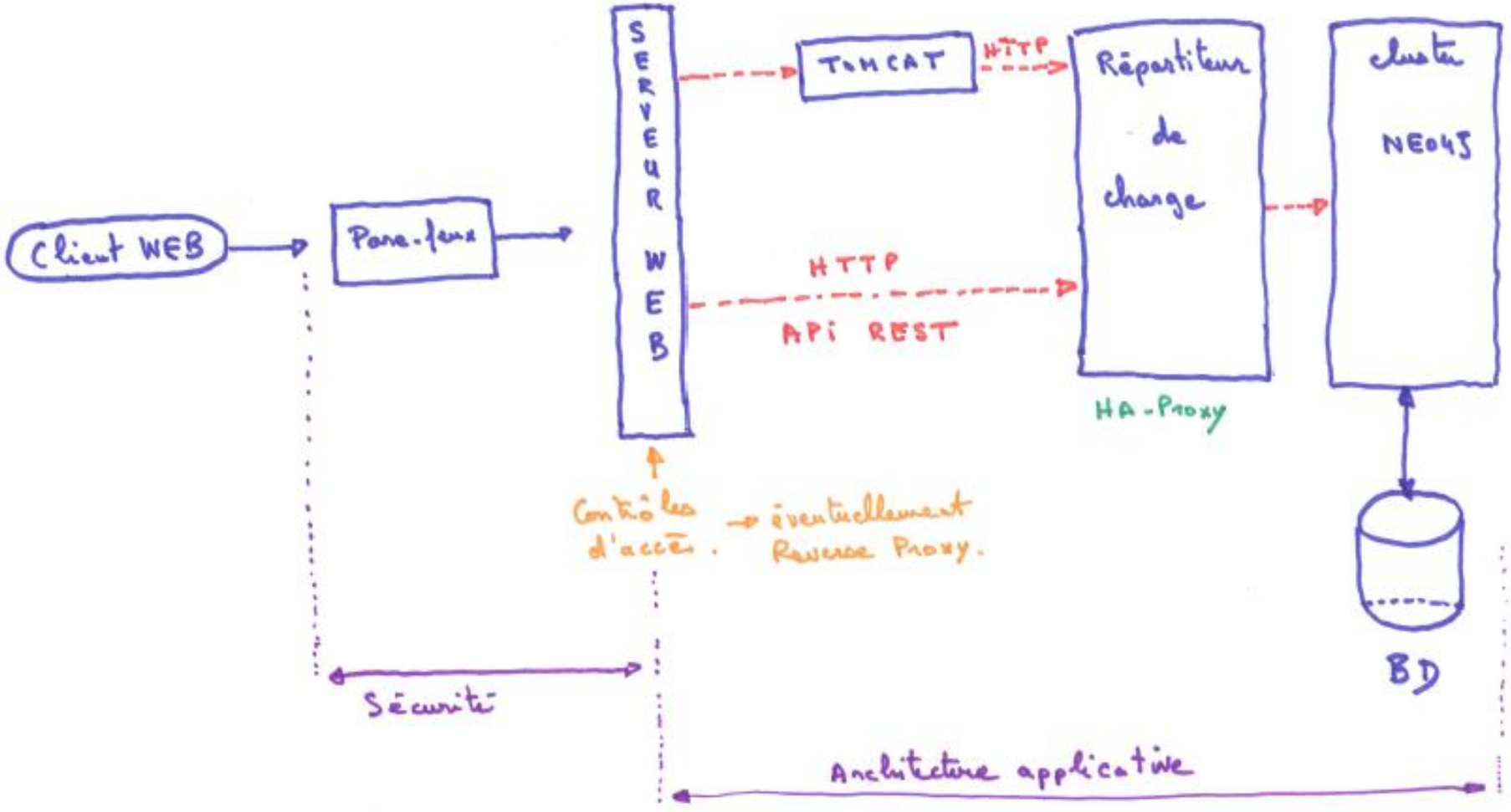


Source : http://jimwebber.org/2011/02/scaling-neo4j-with-cache-sharding-and-neo4j-ha//

La haute-disponibilité : répartition d'un graphe

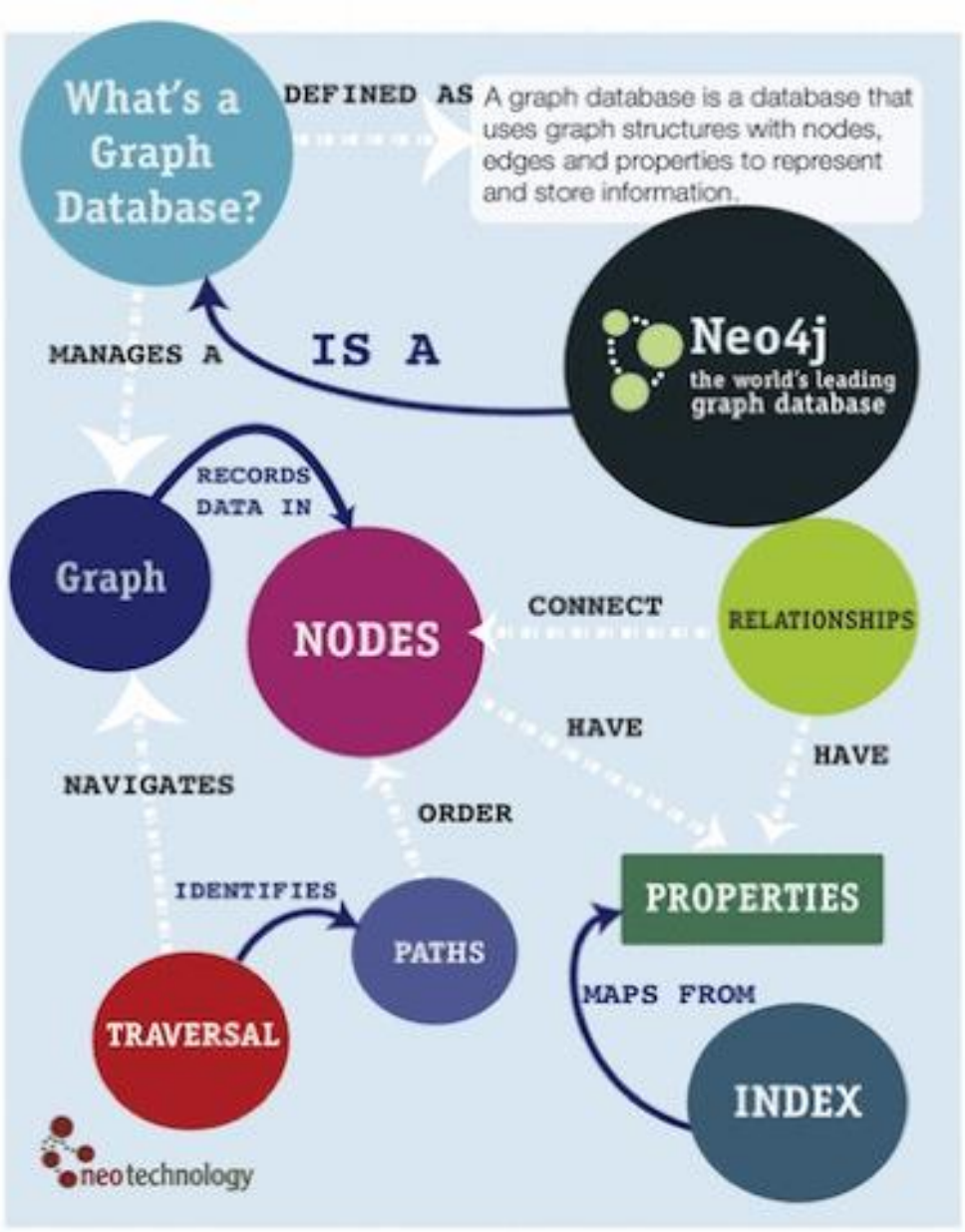


Exemple de mise en œuvre complète



Des questions ?

Résumé des concepts



Source : http://i1-mac.softpedia-static.com/screenshots/Neo4j_2.jpg