Graph database

Responsables :

- D. Donsez
- GP. Bonneau

RICM5
École Polytech' Grenoble

Sommaire

- Introduction
- Motivations
- Listing des BD sur le marché :
  - Comparaison au BD relationnelles
  - Avantages / Inconvénients
  - Exemple de requête de graphe
- Zoom sur Neo4j
- Conclusion
- Démonstration avec Neo4j

Introduction

La démocratisation des BD en graphe:

- l’explosion des réseaux sociaux (facebook, twitter, linkedIn …)
- le mouvement NoSQL qui a aidé à la diffusion d’autres moyens de stockage

Motivations

Quelles motivations poussant à l'utilisation des BD en graphe ?

- une structure de données en adéquation avec l'application à développer
- une syntaxe de requêtes plus proche de la structure de données manipulée

Cas d'utilisations :

- centralisation des logs
- réalisation d'application en graphe
- ...

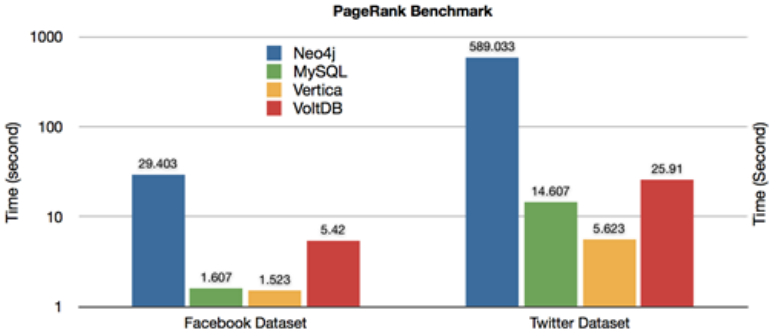
Listings des différents types de BD

- Relationnelle (table) :
  - oritentée colonne : (eg. VerticaDB)
  - orientée ligne : (eg. MySQL)

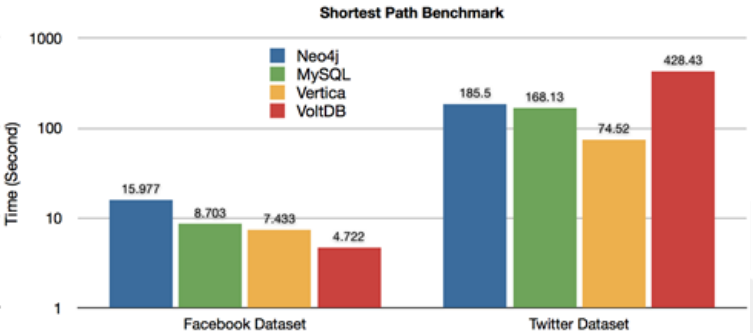
- NotOnlySQL
  - basé sur différents systèmes
    - clé-valeur (eg. Redis)
    - document (eg. MongoDB)
    - …

- Graphe
  - basé sur une structuration en graphe : (eg. Neo4j)
  - basé sur des BD relationnelles : FlockDB, TAO

Comparatif de performance



(a) PageRank on Different Systems



(b) Shortest Paths on Different Systems

Source : http://istc-bigdata.org/index.php/benchmarking-graph-databases/

Un benchmark réalisé par Alekh Jindal : un post doctorant dans le groupe Data base Group du laboratoire MIT CAIL.

Avantages / Inconvénients

BD en graphe :

- Avantages :
  - une interface de requête plus adaptée aux structures en graphe
- Inconvénients:
  - perte en performance en raison de l'implementation sous-jacente (voir Neo4j)

BD relationnelles :

- Avantages :
  - performance (quel que soit le type de requêtes)
- Inconvénients :
  - complexification des requêtes de type graphe
    - perte en temps de développement
    - risque d'erreurs

Solution

Utiliser une interface de requêtes “graph-friendly” qui se mappe au moteur SQL : Giraph, Pregel

- Avantages :
  - abstraction de la syntaxe SQL
- Inconvénients
  - un temps de traitement supplémentaire

Exemples de “wrapper” MySQL : FlockDB (Twitter), TAO (Facebook)

Dataset	VoltDB (SQL)	VoltDB (Vertex-centric)
Facebook Dataset	4.72 sec	14.86 sec
Twitter Dataset	428.43 sec	569.70 sec

Comparaison des interfaces SQL - Vertex-centric (= similaire à Giraph)
Source : http://istc-bigdata.org/index.php/benchmarking-graph-databases/

Zoom sur Neo4j

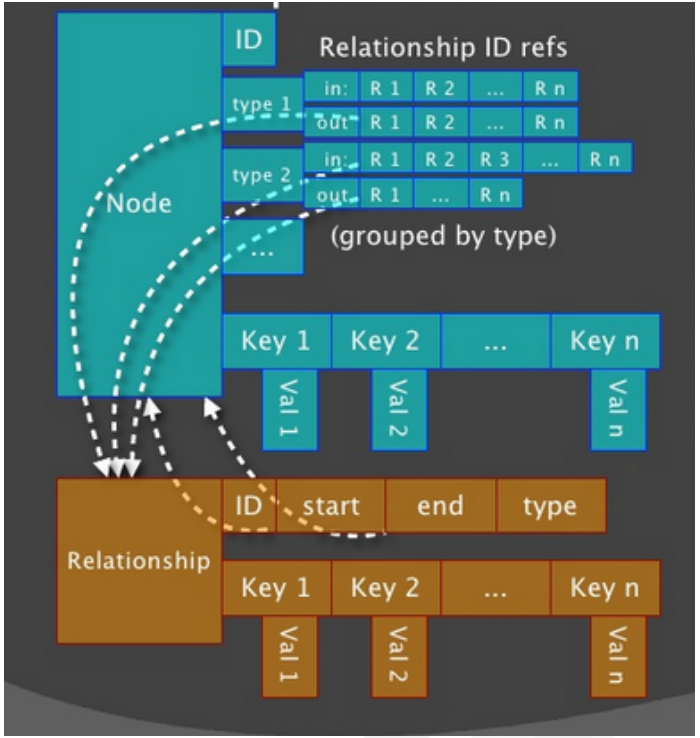
Structure :

- Une liste chainée de noeud

- Chaque noeud contient :
  - une référence sur une liste chainée de relations, classé par types
  - une référence sur une liste chainée de propriétés (clé-valeur)

Inconvénients :

- coût en temps et en nombre d'accès mémoire



Structure des données dans Neo4j

Conclusion

L’application à développer déterminera le choix du type de BD :

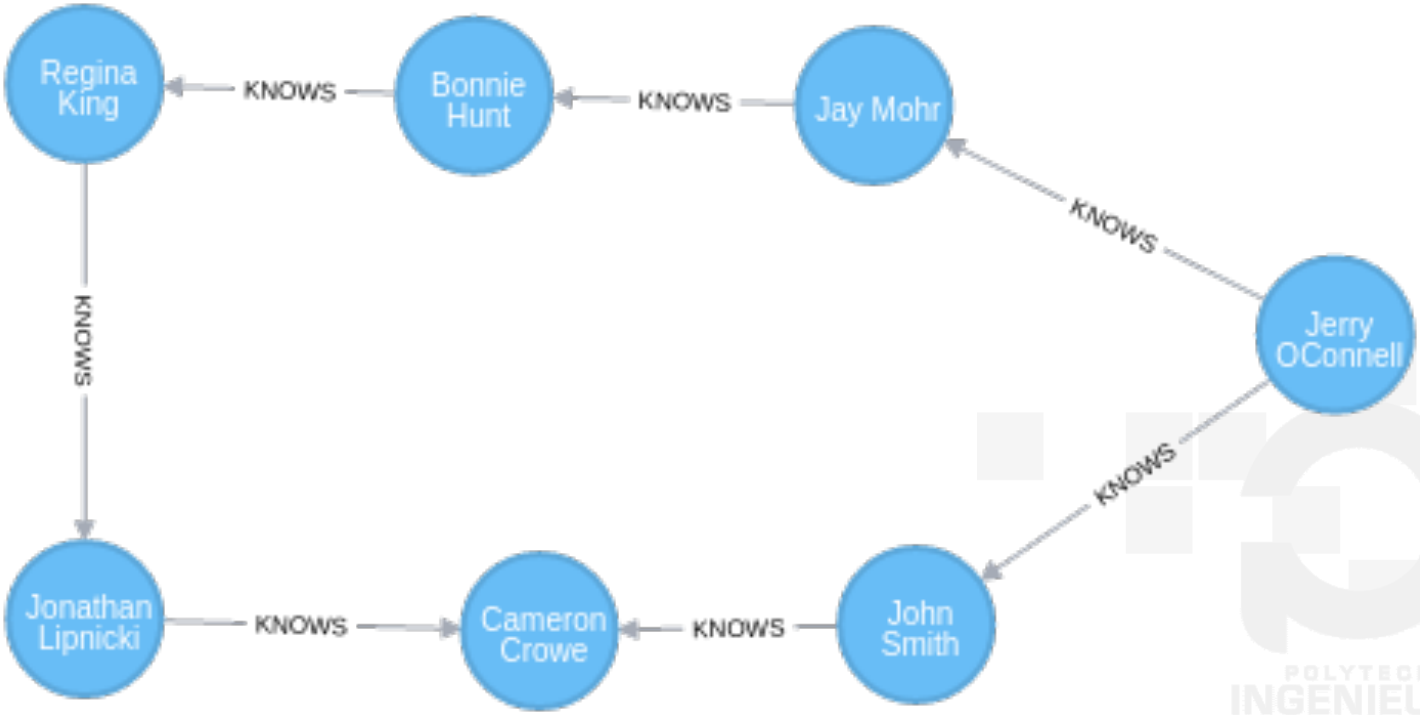
- performances → NoSQL
- propriétés ACID → RDBMS
- facilité de syntaxe des requêtes, pour un développement plus rapide
  → utiliser des “wrapper” de requêtes SQL (Giraph, Pegasus, …)
  → utiliser des BD dédiés : Neo4j (Cypher)

Fin

Des questions ?

Démonstration

Exemple de requête de graphe



Exemple de requête de graphe

Requête de plus court chemin Neo4j (Syntaxe Cypher)

```
MATCH
(jerry:Person { name:"Jerry OConnell" }),
(cameron:Person { name:"Cameron Crowe" }),
p = shortestPath((jerry)-[*..15]-(cameron))
RETURN p
```

Requête de plus court chemin BD relationnelle (Syntaxe SQL)

```
Algorithme récursif sur les connaissances
de 'Jerry OConnell'

1ere ronde
SELECT * FROM KNOWS WHERE name='Jerry
OConnell';

2eme ronde
SELECT * FROM KNOWS WHERE name='John
Smith';

SELECT * FROM KNOWS WHERE name='Jay Mohr';

3eme ronde
On s'arrête, Cameron Crowe est trouvée
```