Graph & Network Analytics
The power of relations

Vandy BERTEN
Section Recherche
6 juin 2017
Définir un réseau
Caractériser un réseau
Visualiser un réseau
Manipuler un réseau
Interroger un réseau
Définir un réseau
Définir un réseau

MOTIVATIONS
"Facebook [...] was founded on the idea that while there’s value in discrete information about people—their names, what they do, etc.—there’s even more value in the relationships between them."
Social Network Analytics

Sociology

Social

Graph theory

Network

(Data) analytics

Analytics

Social Network Analytics : Analyse des structures sociales au travers de la théorie des graphes
(Social) Graph/Network Analytics is not...
7 ponts de Köningsberg (Euler, 1736)
Gartner : BI and Analytics/Data Science

- Définir
- Caractériser
- Visualiser
- Manipuler
- Interroger

Prescriptive Analytics
Real-time Analytics
Graph Analytics

Big Data

Machine Learning
Peak of Inflated Expectations
Predictive Analytics

Plateau of Productivity
Slope of Enlightenment
Text Analytics
Social Analytics

Hype Cycle for Data Science, 2016
Hype Cycle for Business Intelligence and Analytics, 2016
Gartner : applications

- Définir
- Caractériser
- Visualiser
- Manipuler
- Interroger

Route optimization
Epidemiology
Load balancing
Market basket analysis

Intelligence
Fraud detection

Supply chain
Supply chain

Genome
Money laundering
Définir un réseau

EXEMPLES
Définir – Caractériser – Visualiser – Manipuler – Interroger

"Fraud Detection: Discovering Connections with Graph Databases", Neo4j, Sadowski & Rathle
Mariages à Florence

Padgett and Ansell, Robust Action and the Rise of the Medici, 1993
Mariages à Florence

Padgett and Ansell, Robust Action and the Rise of the Medici, 1993
Mariages à Florence

Padgett and Ansell, Robust Action and the Rise of the Medici, 1993
Dépendance de librairies Python
Définir un réseau

DÉFINITIONS
Définir – Caractériser – Visualiser – Manipuler – Interroger

Nœuds, arcs

Nœud, sommet, entité, vertex
Arc, relation, lien, connexion, edge

\[ G = (V, E) \]
Définir – Caractériser – Visualiser – Manipuler – Interroger
Réseau dirigé/non-dirigé

Non dirigé (undirected) : collègue, ami Facebook

Dirigé (directed) : travaille pour, dépend de, follower Twitter
Degré

\[ \forall v \in V \ deg(v) = deg^+(v) + deg^-(v) \]

\[ \sum_{v \in V} \deg(v) = 2 |E| \]

\[ \sum_{v \in V} deg^+(v) = \sum_{v \in V} deg^-(v) \]
Graphe biparti (*bipartite graph*)
Exemple : Dimona

• Déclarations Dimona de 2007 à aujourd’hui
• 7 048 178 nœuds, 20 640 727 relations
• Graphe biparti
• Degré d’un nœud :
  – EMPLOYER : nombre de ses employés
  – WORKER : nombre de ses employeurs
**Exemple : BCE/KBO**

- Fonctions BCE de 2007 à aujourd’hui
- 3 572 876 nœuds, 3 244 227 relations
- Graphe biparti
- Degré d’un nœud :
  - EXECUTOR : nombre de compagnies où il officie
  - COMPANY : nombre de ses « executors » (=mandataires)

---

**Diagramme**

INSS: 227...05 → EXECUTOR
ADM/MGR/FND → ENTITY_ID: 10...35 → COMPANY

**INSS**

**EXECUTOR**

**ADM/MGR/FND**

**COMPANY**

**ENTITY_ID**
Définir – Caractériser – Visualiser – Manipuler – Interroger

Caractériser un réseau
Caractériser un réseau

On peut caractériser :

Le réseau dans son ensemble (Métriques générales)

La relation/distance/similarité entre deux nœuds

Un groupe de nœuds (Clustering)

Un nœud en particulier (Centralité)
Caractériser un réseau

MÉTRIQUES GÉNÉRALES
Définir – Caractériser – Visualiser – Manipuler – Interroger

Densité

\[ \text{Densité} = \frac{\text{Nombre d'arcs}}{\text{Nombres d'arcs possibles}} \]

\[ = \frac{|E|}{(|V|-1)+(|V|-2)+\cdots+1} = \frac{2|E|}{|V| \times (|V|-1)} \]

\[ \frac{10}{10} = 1 \]

\[ \frac{4}{10} = 0,4 \]
Densité ≠ Densité !

Difficile de comparer 2 graphes de tailles différentes !

Définition alternative :

\[
\frac{\text{Nombre d'arcs}}{\text{Nombres de noeuds}}
\]

\[
\frac{2 \times (4 + 4)}{10 \times 9} = 0,177 \ldots
\]
Densité

Dimona:

<table>
<thead>
<tr>
<th>CODE</th>
<th># nœuds</th>
<th># arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>6 474 928</td>
<td>13 722 678</td>
</tr>
<tr>
<td>BCW</td>
<td>332 705</td>
<td>476 003</td>
</tr>
<tr>
<td>IVT</td>
<td>209 166</td>
<td>163 105</td>
</tr>
<tr>
<td>OTH</td>
<td>5 587 049</td>
<td>9 950 130</td>
</tr>
<tr>
<td>STU</td>
<td>1 062 334</td>
<td>1 976 673</td>
</tr>
</tbody>
</table>

BCE:

<table>
<thead>
<tr>
<th>CODE</th>
<th># nœuds</th>
<th># arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>3 572 876</td>
<td>2 933 450</td>
</tr>
<tr>
<td>ADM</td>
<td>923 485</td>
<td>1 055 779</td>
</tr>
<tr>
<td>MGR</td>
<td>1 087 647</td>
<td>781 269</td>
</tr>
<tr>
<td>FND</td>
<td>1 928 211</td>
<td>964 697</td>
</tr>
</tbody>
</table>
Densité alternative

Dimona:

<table>
<thead>
<tr>
<th>CODE</th>
<th># nœuds</th>
<th># arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>6 474 928</td>
<td>13 722 678</td>
</tr>
<tr>
<td>BCW</td>
<td>332 705</td>
<td>476 003</td>
</tr>
<tr>
<td>IVT</td>
<td>209 166</td>
<td>163 105</td>
</tr>
<tr>
<td>OTH</td>
<td>5 587 049</td>
<td>9 950 130</td>
</tr>
<tr>
<td>STU</td>
<td>1 062 334</td>
<td>1 976 673</td>
</tr>
</tbody>
</table>

BCE:

<table>
<thead>
<tr>
<th>CODE</th>
<th># nœuds</th>
<th># arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>3 572 876</td>
<td>2 933 450</td>
</tr>
<tr>
<td>ADM</td>
<td>923 485</td>
<td>1 055 779</td>
</tr>
<tr>
<td>MGR</td>
<td>1 087 647</td>
<td>781 269</td>
</tr>
<tr>
<td>FND</td>
<td>1 928 211</td>
<td>964 697</td>
</tr>
</tbody>
</table>
Densité

- Peut se calculer sur une partie d’un graphe (cluster, communauté, …)
- Donne une idée de l’intensité des relations/contacts entre les nœuds
- Définition 2 : nombre moyen d’arcs par nœuds
Diamètre

Diamètre = plus long « plus court chemin »

Définir – Caractériser – Visualiser – Manipuler – Interroger

1 2 3 4 5 6

→ 3

1 2 3 4 5 6

→ 1

4 5 6

→ ∞
Diamètre
Distribution de degrés

Définir – Caractériser – Visualiser – Manipuler – Interroger
Distribution de degrés (IN vs OUT)

Définir – Caractériser – Visualiser – Manipuler – Interroger
Sur une période de 10 ans

Distribution de degrés : BCE (Adm)

609 041 nœuds de degré 1

Société avec un admin. 
Admin. d’une seule société

1 nœud de degré 400

1 admin. de 400 sociétés ? 
1 société avec 400 admin. ?

Sur une période de 10 ans
Distribution de degrés : BCE (Adm)

Nœuds « people » de degré 1
⇒ Admin. d’une seule société

Nœuds « company » de degré 1
⇒ Avec un seul admin.

Mode :
- 1 société par admin.
- 3 admin. par société

Suppression de 0,5% des nœuds les plus gros
Distribution de degrés : BCE (Adm)

0,003 % des nœuds les plus gros (« outliers »)

1 admin. de 233 sociétés!

Sociétés avec >250 admin.
Métriques générales

- **Densité** : intensité des relations
- **Diamètre** : étalement du réseau
- **Distribution de degré** : « forme » du réseau
Caractériser un réseau

CENTRALITÉ
Centralité

• Centralité d’un nœud = importance d’un nœud au sein du réseau

• Peut se baser sur des caractéristiques :
  – Du nœud isolé (Degree centrality)
  – Du nœud et de son voisinage immédiat (K-core)
  – De l’ensemble du réseau (Betweenness, Page-rank...)
Centralité de degré

Centralité : degré du nœud
Centralité d’intermédialité (betweenness)

Centralité :
Nombre de « plus courts chemins » passant par le nœud
Centralité de proximité (closeness)

Centralité : Distance moyenne de tous les autres nœuds
Page-rank (Google) : Diffusion de son importance vers ses voisins
Centralité

- Degré : nombre de voisins
- Betweenness : importance en tant que relai, connecteur, intermédiaire
- Page-rank : importance par diffusion
- Quoi utiliser ? Dépend très fort du business case

- Hors présentation :
  - K-coreness : nombre de voisins de même k-coreness
  - Closeness : centralité « topologique »
Caractériser un réseau

DISTANCE/SIMILARITÉ
Plus court chemin/Distance

- **Dijkstra** : Calcul du « plus court chemin » entre deux nœuds
- Chaque lien possède un « *poids* » (longueur, coût...)
- **Distance** = \( \sum \) *poids* (=nbr de liens si non pondéré)
- Variante « *A*⁺ » : pour les réseaux géographiques (utilisé par les GPS)
- Peut ne pas être unique. Le nombre de « plus courts chemins » est aussi une métrique de proximité
Six degrees of separation

- Théorie : distance entre chaque personne sur terre ≤ 6 (via 5 connaissances intermédiaires)
- « Degré de séparation » = plus petite distance
- Facebook : moyenne des moyennes : 4,57

https://research.fb.com/three-and-a-half-degrees-of-separation/
Définir – Caractériser – Visualiser – Manipuler – Interroger

Similarité de Jaccard

Similarité de Jaccard :

\[
\text{# voisins communs} \div \text{# voisins totaux} = 2/5 = 0.4
\]
Distance

• Plus court chemin
• Similarité de Jaccard

• Hors présentation :
  – Connectivité
  – Nombre de plus courts chemins
  – Dépendance sociale (social neighbors)
Caractériser un réseau

CLUSTERING
Composantes connexes

Définir – Caractériser – Visualiser – Manipuler – Interroger

Composante connexe : ensemble (maximal) de nœuds avec un chemin entre chaque paire de nœuds

Composante géante (giant component) : la plus grande CC

Beaucoup d’analyses peuvent (doivent) se faire isolément sur chaque composante
## Composante connexe

### Dimona (all)

<table>
<thead>
<tr>
<th>Taille de la CC</th>
<th>Nbre de CC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>35</td>
<td>48</td>
</tr>
<tr>
<td>36</td>
<td>55</td>
</tr>
<tr>
<td>37</td>
<td>96</td>
</tr>
<tr>
<td>38</td>
<td>6 965 132</td>
</tr>
</tbody>
</table>

### Giant component

- 6,53 millions d’employés (99,3%) sont « collègues de (ex-)collègues de … » via 430K employeurs (92%)

### 25 929 situations avec :
- Un employé sans aucun autre* employeur
- Dans une entreprise sans autre* employé

* Dans la période considérée
Composante géante

- Diamètre : 20
- Médiane : distance = 6

\[ \text{\rightarrow 50 \% des « couples » sont « collègue de collègue de collègue »} \]

- Percentile 95 : distance = 8 (+1 « de collègue »)
- Il y a des « super-connecteurs » : tous les enseignants sont collègues !
Composante connexe : sub-giants

- **Définir**
- **Caractériser**
- **Visualiser**
- **Manipuler**
- **Interroger**

Employeur

Employé « BIS »

Employé « NISS »

[Diagram showing network connections with highlighted nodes for Employer, BIS employee, and NISS employee]
Composante connexe : sub-giants
Composante connexe : sub-giants

- Employeur
- Employé « BIS »
- Employé « NISS »
Définir – Caractériser – Visualiser – Manipuler – Interroger

Cliques

Triangles
Cliques

Ont des gérants en commun

Entreprise

Principe utilisé pour détecter les « spider constructions » (avec une définition de « clique » un peu plus souple)
Communautés

Coefficient de modularité :
- Mesure de la qualité d’une partition
- \([-1, 1]\), liens inter-cluster vs liens intra-cluster
- Bonne partition (coef. ≈ 1):
  liens intra-cluster >> liens inter-cluster
- Optimisation = NP-hard
- Basé sur les liens, pas sur les attributs
Communautés : intérêts

• Différentes communautés ont souvent des propriétés différentes de l’ensemble
• Permet l’analyse des communautés une par une
• Permet l’identification des « missing links » et des « fake links »
• Inférer des caractéristiques (attributs)
• Vue d’ensemble avec un « graphe de clusters » (+drill-down)
Définir – Caractériser – Visualiser – Manipuler – Interroger

Algorithme de Louvain (UCL, 2008)

Améliore la « modularity »?
Si oui, migrer

Clustering

- Composantes connexes : groupes « déconnectés »
- Cliques : groupes « totalement connectés »
- Communautés : groupes « sociaux »

Hors présentation : notion de « fortement » ou « faiblement » connecté si on tient compte des réseaux dirigés
Caractériser un réseau

HOMOPHILIE
Homophilie / Assortativité

- Homophilie / Assortativité : formalisation du « qui se ressemble s’assemble »
- Réseau « homophile » : les nœuds similaires sont mieux liés que les nœuds différents
- En détection de fraude/criminalité : plus de chances de trouver des fraudeurs dans le voisinage d’un fraudeur que n’importe où
Questions ?
Pause !
Visualiser un réseau
Visualiser
Visualiser un réseau

SPATIALISATION (LAYOUT)
Spatialisation

- Cas simple : graphe **planaire** = qui peut se dessiner sans croisement
- En général :
  - Minimiser les croisements
  - Rapprocher les nœuds connectés
  - Éloigner les nœuds non-connectés
Spatialisation

- **Force-directed** :
  - Force *attractive* (spring) entre deux nœuds liés
  - Force *répulsive* entre tous les nœuds
- **Spectral** : Utilisation des *vecteurs/valeurs propres* (*eigenvectors/values*)
- **Autres** : Hiérarchiques, circulaires, linéaires...
Spatialisation

Définir – Caractériser – Visualiser – Manipuler – Interroger

Yifan Hu

ForceAtlas2

OpenOrd

Fruchterman Reingold
Visualiser un réseau

SIMPLIFICATION
Simplification

- Au-delà de quelques centaines de nœuds, visualiser un graphe est illusoire
- Plusieurs méthodes permettent une approche
  - Découper le graphe
  - Supprimer des nœuds
  - Fusionner des nœuds
Découper le graphe

- Composantes connexes
  → Méthode naturelle, pas de perte

- Communautés
  → On perd les relations inter-communauté
Filtrer des nœuds

• Supprimer les **super-connecteurs** (degré élevé)
  → augmente le nombre de composantes connexes

• Supprimer les **feuilles** (degré/K-core faible)
  → focus sur le « backbone/core »
Fusionner des nœuds

• Contraction : fusionner des groupes
• Projection bipartite : conversion d’un graphe biparti en 2 graphes unipartis.
Contraction
Contraction

Définir – Caractériser – Visualiser – Manipuler – Interroger
Contraction : BCE (Administrateurs)

- Administrateurs BCE :
  - Composante géante = 655.158 nœuds (sur ~10⁶)
- On applique Louvain :
  - 743 communautés (~10'' en mono-thread)
Contraction : BCE (Administrateurs)

Définir – Caractériser – Visualiser – Manipuler – Interroger
Projection bipartite

Définir – Caractériser – Visualiser – Manipuler – Interroger

Travailleur

Employeurs

Collègues

Travailleurs communs
Visualiser un réseau

OUTIL DESKTOP : GEPHI
Gephi

Définir – Caractériser – Visualiser – Manipuler – Interroger
Visualiser un réseau

LIBRAIRIE JS: VIS.JS
```html
<div id="graph"></div>
<script type="text/javascript">
var nodes = new vis.DataSet([{
    id: 1, label: 'Node 1'
}, {
    id: 2, label: 'Node 2'
}, {
    id: 3, label: 'Node 3'
}]);

var edges = new vis.DataSet([{
    from: 1, to: 3
}, {
    from: 1, to: 2
}, {
    from: 3, to: 3
}]);

var container = document.getElementById('graph');
var data = {
    nodes: nodes,
    edges: edges
};

var options = {};
var network = new vis.Network(container, data, options);
</script>
http://visjs.org/examples/network/basicUsage.html
Visualiser un réseau

OUTIL WEB : LINKURIOUS
Définir – Caractériser – Visualiser – Manipuler – Interroger
Visualiser un réseau

AUTRES OUTILS
Outils desktop

- Gephi [https://gephi.org](https://gephi.org) os
- TouchGraph [http://www.touchgraph.com](http://www.touchgraph.com)
- Cytoscape [http://www.cytoscape.org](http://www.cytoscape.org) os
- Maltego [https://www.paterva.com](https://www.paterva.com)
- NodeXL [https://nodexl.codeplex.com](https://nodexl.codeplex.com) os
- SAS Network Analytics €
- IBM i2 €
Outils Cloud

- Linkurio.us  https://linkurio.us €
- Keylines  https://cambridge-intelligence.com/keylines €
Librairies web

- Vis.js  [http://visjs.org](http://visjs.org)  os
- Cytoscape.js  [http://js.cytoscape.org](http://js.cytoscape.org)  os
- Sigma.js  [http://sigmajs.org](http://sigmajs.org)  os
- D3.js  [https://d3js.org](https://d3js.org)  os
- OGMA (Linkurio.us SDK)  €
Visualiser un réseau

USE CASE : SOCDUMPV2
SocDumpv2 : Contexte

• Pour différents services d’inspection, on veut visualiser le réseau d’un groupe de personnes ou entreprises (suspectées de dumping social ou autre fraude)

• À partir de données de (principalement) :
  – Dimona : relations employeurs – employés (en Belgique)
  – BCE/KBO : relations sociétés – mandataires
  – Limosa : Travailleurs étrangers travaillant en Belgique
  – DDT : Sociétés – Chantiers – Sous-traitants
  – CAW : Chantiers – Travailleurs – Sociétés
Définir – Caractériser – Visualiser – Manipuler – Interroger

SocDumpv2 : modules

1. Dimona
2. BCE/KBO
3. Limosa
4. CAW

Modules : Dimona, BCE/KBO, Limosa, CAW

BCE/KBO
SocDumpv2 : modules

• Chaque module produit 2 types d’outputs :
  – Des nœuds : personnes, entreprises, POW
  – Des relations

• Les nœuds d’outputs de chaque module peuvent servir d’input pour un autre

• Exemple :

  NISS –[BCE]–> Entreprises –[Dimona]–> Employés
SocDumpv2 : qualité

• La qualité n’est pas toujours au rendez-vous !
• Une entreprise peut être désignée soit par un identifiant « source authentique » (numéro BCE, matricule...), soit par une un numéro interne à l’application (principalement DDT, Limosa)
• Idem pour les personnes : numéro NISS, ou numéro interne
SocDumpv2 : Flux

Définir – Caractériser – Visualiser – Manipuler – Interroger
SocDumpv2

• Après la recherche des données, on va :
  – Collecter les adresses des entreprises
  – Enrichir les nœuds (nom des entreprises et des personnes, dates de naissance)
  – Détecter et (éventuellement) fusionner les doublons
  – Calculer une série de métriques (distance, centralité…)

• On visualise ensuite le résultat : Gephi, TouchGraph…
Manipuler un réseau

Définir – Caractériser – Visualiser – Manipuler – Interroger
Manipuler un réseau

IGRAPH (PYTHON/R)
Définir – Caractériser – Visualiser – Manipuler – Interroger

iGraph

- iGraph : librairie C de création, manipulation et analyse de graphes
- Interfacé avec R, Python, Mathematica
- Open Source, GNU GPL v2
- Très populaire dans la communauté scientifique

+ : standard, multi-langage, natif, très riche
- : mono-cpu, installation pas toujours aisée
iGraph : fonctions

- **Génération** : de nombreux modèles
- **Analyse** : centralité, chemins, communauté, recherche de pattern/motifs, comparaison isomorphe...
- **Modification** : projection bipartite, extraction
- **Visualisation** : layout, génération d’image
- **Import/Export**
**iGraph, Python vs R.**

**Python**

```python
import igraph as ig

g = ig.Graph()
g.add_vertices(['a', 'b', 'c'])
g.add_edges([('a', 'b'), ('a', 'c')])

g.vs['label'] = ['x', 'y', 'z']
g.vs[2]['name'] = 'C'
```

**R**

```r
g <- graph(edges = c('a', 'b', 'a', 'c'))
V(g)$label = c('x', 'y', 'z')
V(g)$name[2] = 'C'
```

**Définir – Caractériser – Visualiser – Manipuler – Interroger**
Définir – Caractériser – Visualiser – Manipuler – Interroger

BIG DATA: GIRAPHE & GRAPHX

Manipuler un réseau
GraphX

http://spark.apache.org/graphx/

• Librairie de manipulation de graphes pour Spark (framework open source de calcul distribué/Big Data)

• Spark :
  – Souvent utilisé sur Hadoop (pour HDFS)
  – En Scala, Java ou Python
  – GraphX : Uniquement en Scala

• Basé sur les RDDs (non-modifiables)

• + : Permet la manipulation de très gros graphes

• - : Librairie très limitée (à ce stade)
GraphX

// Create an RDD for the vertices
val users: RDD[((VertexId, (String, String)))] =
sc.parallelize(Array(((3L, ("rxin", "student")),
(7L, ("jgonzal", "postdoc")),
(5L, ("franklin", "prof")))))

// Create an RDD for edges
val relationships: RDD[Edge[String]] =
sc.parallelize(Array(Edge(3L, 7L, "collab"),
Edge(5L, 3L, "advisor"),
Edge(5L, 7L, "pi")))

Graph val graph = Graph(users, relationships, defaultUser)
Giraph

"Apache Giraph is an iterative graph processing system built for high scalability."

- Extension de Pregel (Google)
- Utilisé par Facebook
- Basé sur MapReduce (Hadoop)

http://giraph.apache.org
GraphX vs Giraph

- GraphX créé pour être plus performant que Giraph (moins de passage par le disque)
- Benchmark par GraphX largement en sa faveur...
- Mais des benchmarks de Facebook montrent le contraire

https://code.facebook.com/posts/319004238457019/
Interroger un réseau

Définir – Caractériser – Visualiser – Manipuler – Interroger

www.neo4j.com
Interroger un réseau

GRAPH DATABASES
**RDBMS et relations**

**Employés de Smals ?**

Ce qui intéresse le développeur

```
SELECT Workers.Name
FROM Workers
JOIN Companies
ON Workers.Employer_ID = Companies.ID
WHERE Companies.Name = 'Smals'
```
RDBMS et relations

Employés de Smals ?

Ce qui intéresse le développeur

```sql
SELECT Workers.Name
FROM Workers
JOIN Works_for
    ON Workers.ID = Works_for.Worker_ID
JOIN Companies
    ON Works_for.Company_ID = Companies.ID
WHERE Companies.Name = 'Smals'
```
RDBMS et relations

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
</tr>
<tr>
<td>3</td>
<td>Camille</td>
</tr>
<tr>
<td>4</td>
<td>Zoé</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Liker_ID</th>
<th>Liked_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Qui Bob aime ?

```sql
SELECT p1.Name
FROM People p1
JOIN Likes
ON Likes.Liked_ID = p1.ID
JOIN People p2
ON Likes.Liker_ID = p2.ID
WHERE p2.Name = "Bob"
```
RDBMS et relations

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
</tr>
<tr>
<td>3</td>
<td>Camille</td>
</tr>
<tr>
<td>4</td>
<td>Zoé</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Liker_ID</th>
<th>Liked_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Qui Bob (aime)² ?

SELECT p1.Name
FROM People p1
JOIN Likes l1
  ON l1.Liked_ID = p1.ID
JOIN Likes l2
  ON l1.Liker_ID = l2.Liked_ID
JOIN People p2
  ON l2.Liker_ID = p2.ID
WHERE p2.Name = "Bob"
RDBMS et relations : limitations

Relations 1-n

- Détourne le rôle d’un attribut
- Pas de type « Key »

Relations m-n

- Détourne le rôle d’une table
- Pas distinguishable des autres

- Contrainte d’intégrité possible (externe à la table)
- Structure de la relation à préciser à chaque requête
- Join : Lourd en écriture et en exécution
Solutions NoSQL

Définir – Caractériser – Visualiser – Manipuler – Interroger
GraphDB : objectifs

Objectifs des bases de données orientées graphes :

**Querying language** qui sépare la définition des relations (à la création) et la recherche de « matching »

```sql
SELECT p1.Name
FROM People p1
JOIN Likes l1
    ON l1.Liked_ID = p1.ID
JOIN Likes l2
    ON l1.Liker_ID = l2.Liked_ID
JOIN People p2
    ON l2.Liker_ID = p2.ID
WHERE p2.Name = "Bob"
```

**Moteur efficace pour le parcours des relations**

Cypher (Neo4j)

```cypher
MATCH (:People {Name:"Bob"}) -[:Likes*2]->
    (p:People)
RETURN p.Name
```
Définir – Caractériser – Visualiser – Manipuler – Interroger

GraphDB: moins de code

MATCH (boss)-[MANAGES*0..3]->(sub), (sub)-[MANAGES*1..3]->(report)
WHERE boss.name = "John Doe"
RETURN sub.name AS Subordinate, count(report) AS Total
“We found Neo4j to be literally thousands of times faster than our prior MySQL solution, with queries that require 10-100 times less code. Today, Neo4j provides eBay with functionality that was previously impossible.”

https://neo4j.com/news/graph-databases-provide-crystal-ball-online-recommendations-wal-mart-ebay/
Interroger un réseau

NEO4J
Liste des employés de Smals

MATCH (w:Worker) -[r:WORKS_FOR]-(c:Company)
WHERE c.Name = "Smals"
RETURN w.Name, w.BD, r.Start

Variante :
MATCH (w:Worker) -[r:WORKS_FOR]- (c:Company {Name:"Smals"})
RETURN ...
Requête Neo4j

MATCH
(w1:Worker)-[[:WORKS_FOR]->(c:Company {Name:"Smals"}),
(w2:Worker)-[[:WORKS_FOR]->(c),
(w1)-[[:RELATIVES]->(w2)
RETURN w1.Name, w2.Name

Liste des employés de Smals apparentés

Définir – Caractériser – Visualiser – Manipuler – Interroger
Définir – Caractériser – Visualiser – Manipuler – Interroger

Requête Neo4j

```
MATCH
  (w1:Worker)-[:WORKS_FOR]->(c:Company {Name:"Smals"})
  <-[:WORKS_FOR]- (w2:Worker)-[:RELATIVES]->(w1)
RETURN w1.Name, w2.Name
```

Liste des employés de Smals apparentés

- **Name:** Vandy
  - **BD:** 15/09/80
  - **Type:** cousin

- **Name:** Victor
  - **BD:** 30/03/85

**Company:** Smals
- **Name:** Smals
- **Par. Com:** XXX
- **BCE:** 0406.798.006
- **Start:** 01/05/13

**Company:** Smals
- **Name:** Smals
- **Par. Com:** YYY
- **BCE:** 0406.798.006
- **Start:** 15/09/10
### Fonctionnement

<table>
<thead>
<tr>
<th>RDBMS</th>
<th>Graph DB</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>schema</strong></td>
<td><img src="image" alt="Diagram of RDBMS schema" /> <img src="image" alt="Diagram of Graph DB" /></td>
</tr>
<tr>
<td><strong>WHERE ...</strong></td>
<td>Search in Companies (with index): $O(\log([\text{Table size}])$)</td>
</tr>
<tr>
<td><strong>JOIN :</strong></td>
<td>Search in Works_for : $O(\log([\text{Table size}])$)</td>
</tr>
<tr>
<td>- Search in Workers :</td>
<td>$O(\log([\text{Table size}]) \times [\text{nb res}])$</td>
</tr>
</tbody>
</table>

**Définir** – Caractériser – Visualiser – Manipuler – **Interroger**

129/149
Performance

- Réseau social de $10^6$ de nœuds
- ~50 voisins par nœuds
- Recherche des « amis d’amis »

<table>
<thead>
<tr>
<th>Depth</th>
<th>MySQL</th>
<th>Neo4j</th>
<th># results</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0,02</td>
<td>0,01</td>
<td>2 500</td>
</tr>
<tr>
<td>3</td>
<td>30,27</td>
<td>0,17</td>
<td>110 000</td>
</tr>
<tr>
<td>4</td>
<td>1543,51</td>
<td>1,36</td>
<td>600 000</td>
</tr>
<tr>
<td>5</td>
<td>Unfinished</td>
<td>2,13</td>
<td>800 000</td>
</tr>
</tbody>
</table>

Scénario très favorable aux graph DB !

Graph Databases, I. Robinson et al.
Neo4j : Création des données

CREATE (:Person {Name:"Vandy", BD:"15/09/80"})
CREATE (:Company {Name:"Smals", BCE:"0406..."})
MATCH (p:Person {Name: "Vandy"}),
     (c:Company {Name: "Smals"})
CREATE (p)-[:WORKS_FOR {Start:...}]->(c)
OU:
CREATE (:Person {Name:"Vandy", BD:"15/09/80"})
   -[:WORKS_FOR {Start:...}]->(:Company {Name:"...", ...})
Neo4j : importation de données

• Dans la pratique : importation de fichiers CSV (à préparer !!)
• Via une commande Cypher (LOAD CSV FROM...)
  – Facile pour des petits fichiers
  – Lent à l’exécution
• En command line
  – Beaucoup plus rapide
  – Mal documenté
Définir – Caractériser – Visualiser – Manipuler – Interroger

Interroger un réseau

GRAPHDB : LA SOLUTION ?
RDBMS ou GraphDB ?

Dans un monde idéal et utopique

1. SQL

2. SQL

3. SQL

Définir – Caractériser – Visualiser – Manipuler – Interroger
GraphDB : Avantages

• Langage plus proche du modèle
• Pas de clés à gérer, ni de contrainte d’intégrité
• Pas de structures de relations à chaque requête
• Très efficace pour le parcours de relations
• Beaucoup de champs d’application : recommandation, fraude, infra, MDM, KM…
• Neo4J: ACID, schema-less
GraphDB : limitations

- Pas la maturité des RDBMS (robustesse, haute dispo., communauté...)
- Pas de standard
- Survie à l’effet de mode ? (cf OO DB)
- Pas optimal pour des recherches, agrégation, batch process, transactionnel
- Pas une alternative, plutôt un complément (ne résout pas les mêmes problèmes)
Interroger un réseau

USE CASE : SUPERBONUS
Superbonus : contexte

• Avec le TaxShift (2016), des avantages sont donnés aux nouveaux employeurs
• Beaucoup d’employeurs existants se recréent pour profiter de ces avantages
• Le « Network Analytics » permet d’identifier beaucoup de ces cas
Data collection

DmfA: **SuperBonus** requestors
11078 employers

Dimona: **Worker in Periods** 2015-2016
27056 workers

Dimona: **Employers in Periods** 2015
15416 employers

BCE: **Executors** + other companies

BCE: **Addresses** (3296 addr.) + cotenant
Définir – Caractériser – Visualiser – Manipuler – Interroger

Full transfert

\[ \text{COUNT}(W) > \text{Em.Worker}_\text{Cnt}_\text{2015} \times 80\% \]
\[ \text{COUNT}(W) > \text{SB.Worker}_\text{Cnt}_\text{2016} \times 80\% \]

MATCH (c_sb:SB)--(w:Worker)--(c_old:Company)
WHERE c_sb <> c_old
OPTIONAL MATCH (c_sb)--(e:Executor)--(c_old)
OPTIONAL MATCH (c_sb)--(a:Address)--(c_old)
WITH c_sb, c_old,
    \[ \text{COUNT(DISTINCT } w\text{)} \text{ AS } w\text{_count} \]
WHERE
    \[ w\text{_count} > c\_sb\.\text{WORKER}_\text{CNT}_\text{16} \times 0.8 \text{ AND} \]
    \[ w\text{_count} > c\_old\.\text{WORKER}_\text{CNT}_\text{15} \times 0.8 \]
RETURN c_sb, c_old, w_cnt

\( W \) \hspace{1cm} \( \text{Em} \) \hspace{1cm} Employer without SB \hspace{1cm} \( \text{SB} \) \hspace{1cm} Employer with SB
\( A \) \hspace{1cm} Executor
COUNT(W) > 0 OR COUNT(A) > 0

MATCH (sb:SB) --> (e:Executor) --> (comp:Company)
WHERE sb <> comp
OPTIONAL MATCH (sb) --> (w:Worker) --> (comp)
OPTIONAL MATCH (sb) --> (a:Address) --> (comp)
WITH sb, comp,
  COUNT(DISTINCT w) AS Workers_count,
  COUNT(DISTINCT a) AS Addresses_count
WHERE (Workers_count > 0 OR Addresses_count > 0)
RETURN sb, comp, ...
Conclusions
En résumé

**Caractériser** (théorie) : tout un réseau, un nœud (centralité), deux nœuds (distance), un groupe de nœuds (clustering)

**Visualiser** : Desktop, Javascript, Cloud

**Manipuler** (coding) : Environnement classiques ou Big Data

**Interroger** (graph DB) : focus sur les relations

**Application** : fraude, gestion des infrastructures/logicielles, recommandation, intelligence, KM...
Conclusions

• De la place pour les graph analytics/graph DB dans nos institutions
• Pas uniquement pour la fraude !
• Pour les data-scientists/BI :
  ➢ nouvelle façon de réfléchir, centré sur les relations
• Pour les DBA/développeurs :
  ➢ nouvelle famille de DB, complémentaires
• Pour les analystes/business :
  ➢ inspiration pour de nouveaux use cases
Conclusions : expérience personnelle

- Préparation des données :
  - R/Python + igraph/graphx
- Première analyse :
  - Gephi/TouchGraph
- Production :
  - GraphDB (Neo4j...)
- Pour le client :
  - Interactif (futur) : libraire JS / outils « à la » Linkurio.us
  - Statique : Gephi/TouchGraph, igraph → Excel, PDF...
Références : www.smalsresearch.be

• Blogs :
  – https://www.smalsresearch.be/graph-db-vs-rdbms/

• Autres :
Références

• **Graph Databases**, *Robinson & all*, O’Reilly 2015

• **Fraud Detection: Discovering Connections with Graph Databases**, *Neo4j*, Sodawkski & Rathle

Définir – Caractériser –  – Visualiser – Manipuler – Interroger

Vandy Berten
02/787.57.32
vandy.berten@smals.be

More on Smals Research :
Website : www.smalsresearch.be
Blog : www.smalsresearch.be/blog
Twitter : @SmalsResearch