Fighting Fraud with Graph Databases

Vandy BERTEN
Smals Research
Devoxx – Nov 8th, 2017
Outline

Motivations

Graph databases

Neo4j

Use case: Social dumping

Conclusions
MOTIVATIONS
Motivations

- Sells
- Works for
- Depends upon
- Develops
- Owns
- Runs on
- Is connected to
- Is friend of
- Is boss of
- Is member of

= (social) graph
= (social) network
Starting points

- Graphs excel in modeling fraudulent/criminal behavior.
- RDMS perform poorly in representing graphs (i.e., relationships between entities).
- Graph databases propose a new database model, allowing to query graphs.
- RBMS focus on entities, Graph DB focus on relationships.
GRAPH DATABASES
RDBMS and relationships

Smals' Employees?

SELECT Workers.Name
FROM Workers
JOIN Companies
ON Workers.Employer_ID = Companies.ID
WHERE Companies.Name = 'Smals'

How relationships are implemented

RDBMS: Relational DataBase Management System

What concerns developers

RDBMS and relationships

Workers
ID
Name
Employer_ID

Companies
ID
Name

Smals' Employees?
RDBMS and relationships

Smals’ Employees?

What concerns developers

How relationships are implemented

```
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 and relationships

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

Who Bob likes?

```
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 and relationships

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

RDBMS and relationships: limitations

1-n relationships

- Transforms the role of attributes
- No type « Key »
- Integrity constraints possible (extern to tables)
- Relationship structure to be given to each request
- Join : Heavy in writing code and in execution time

m-n relationships

- Transforms the role of tables
- No distinguishable

Foreign key:

Join table:
NoSQL solutions

- Key-value
- Column
- Document
- Graph
Graph database model

Node, entity

Name: John  
BD: 3/2/85

Name: Bob  
BD: 5/6/85

Label, Type

WORKS_FOR
Kind: Employee

Company
Name: Smals  
BCE: 0406.798.006

Attribute, property
(key + value)

LIVES_AT

Address
Street: Av. de Mai  
Number: 165

Relationships
GraphDB: objectives

Objectives of graph databases:

**Querying language** splitting apart relationship definition (creation time) and « 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"
```

**Fast search engine for relationship traversals**

```cypher
MATCH
  (:People {Name:"Bob"})
-[:Likes*2]->
(p:People)
RETURN p.Name
```

Cypher (Neo4j)
GraphDB: less 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."
Neo4j

• Current Graph Database leader
• Written in Java, multi-platform
• Community Edition (free, GPL) + Enterprise Edition ($, AGPL)
• Can be interfaced by many languages
• ACID-compliant
• Queries in Cypher or Gremlin
• Since 2007
Neo4j request

Node of type « Worker »

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

Variante :

```
MATCH (w:Worker)
-[r:WORKS_FOR]-> (c:Company)
WHERE c.Name = "Smals"
RETURN ... 
```
# How it works

<table>
<thead>
<tr>
<th>RDBMS</th>
<th>Graph DB</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>schema</strong></td>
<td><img src="image1" alt="Diagram of RDBMS schema" /></td>
</tr>
<tr>
<td><strong>WHERE ...</strong></td>
<td><strong>Search in Companies (with index):</strong></td>
</tr>
<tr>
<td></td>
<td>(O(\log(\text{[Table size]})))</td>
</tr>
<tr>
<td><strong>JOIN:</strong></td>
<td><strong>Follow pointers in node:</strong></td>
</tr>
<tr>
<td>- <strong>Search in Works_for:</strong></td>
<td>(O(\log(\text{[Table size]})))</td>
</tr>
<tr>
<td>- <strong>Search in Workers:</strong></td>
<td>(O(\log(\text{[Table size]})) \times [\text{nb res}]))</td>
</tr>
</tbody>
</table>
USE CASE: SOCIAL DUMPING
Limosa declaration: A foreign employer sending (posting) a worker in Belgium

www.limosa.be
DDT: sub-contractors

DDT: Déclaration De Travaux
Aangifte van werken
On a specific worksite (PoW), we have a chain/tree of sub-contractors.
(a:COMPANY)-[:LIM_EMPL]-
(lim_decl:LIM_DECL)-[:LIM_POWER]-
(lim_pow:LIM_POWER)-[:LIM_POWER2COMP]-
(c:COMPANY)-[:DDT_PARENT]-
(ddt_decl:DDT_DECL)-[:DDT_CHILD]-
(b:COMPANY)-[:LIM_CLI]-
(lim_decl)

NOT EXISTS (a)-[:DDT_CHILD]-
(DDT_DECL)-[:DDT_POWER]-
(ddt_pow)
MATCH
(a:COMPANY)-[:LIM_EMPL]-(lim_decl:LIM_DECL)-[:LIM_POW]-
(lim_pow:LIM_POW)-[:LIM_POW2COMP]-(c:COMPANY)-[:DDT_PARENT]-
(ddt_decl:DDT_DECL)-[:DDT_CHILD]-(b:COMPANY)-[:LIM_CLI]-(lim_decl),
(ddt_decl)-[:DDT_POW]-(ddt_pow:DDT_POW)

WHERE
NOT EXISTS ((a)-[:DDT_CHILD]-(:DDT_DECL)-[:DDT_POW]-(ddt_pow)) ...

RETURN
DISTINCT a, b, c,
COUNT(DISTINCT lim_decl),
COUNT(DISTINCT ddt_decl)

ORDER BY COUNT(DISTINCT lim_decl) DESC
Implementation

• We copied in Neo4j 17M nodes, 45M relationships:
  – 800,000 companies, 1,650,000 (foreign) workers
  – 13 millions declarations
  – 185,000 duplication groups
  – 250,000 streets
  – …

• 9 node labels, 15 relationship types
• All companies triplets matching 1st pattern: ~10 seconds (325 results)
In SQL/RDBMS?

- In SQL/RDBMS,
  - Any node and any relationship is a record: ~65 M records
  - Any type/label corresponds to a table: 24 tables
- Any `(n:Type1)-[:REL1]-(m:Type2)` corresponds to:

  ```sql
  FROM Type1
  JOIN REL1 ON Type1.key = REL1.type1_key
  JOIN Type2 ON Type2.key = REL1.type2_key
  \rightarrow Almost 20 joins!
  ```
- Query far more complex to write
- Much longer to run
Data quality problem

• This only works if « B » has been declared with the same ID in both systems!

• Sometimes, the “official ID” is not mandatory → name + address is OK

• We have to deal with this!
Duplicates

Thanks to data quality tools, such as Trillium (IntoDQ), OpenRefine...
Addresses

A standardization is needed! Thanks to data quality tools
Duplicates combinations

A → A1 → DUP GROUP → A2 → Street
Schema with B duplicated
Conclusions

- Graph DBs are the « ideal partner » for many fraud detection problems
- They outperform RDBMS on many aspects (query expressiveness, execution speed...)
- Data quality must be taken into consideration
- Not only for fraud: Infrastructure management, recommendation systems, intelligence...
References: www.smalsresearch.be

• Blogs (in French):
  – https://www.smalsresearch.be/graph-db-vs-rdbms/

• Other:
References

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

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

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