DEVOXX

Fighting Fraud with Graph Databases

Vandy BERTEN

Smals Research

Devoxx – Nov 8th, 2017

Smals

ICT for society

Outline

Motivations

Graph databases

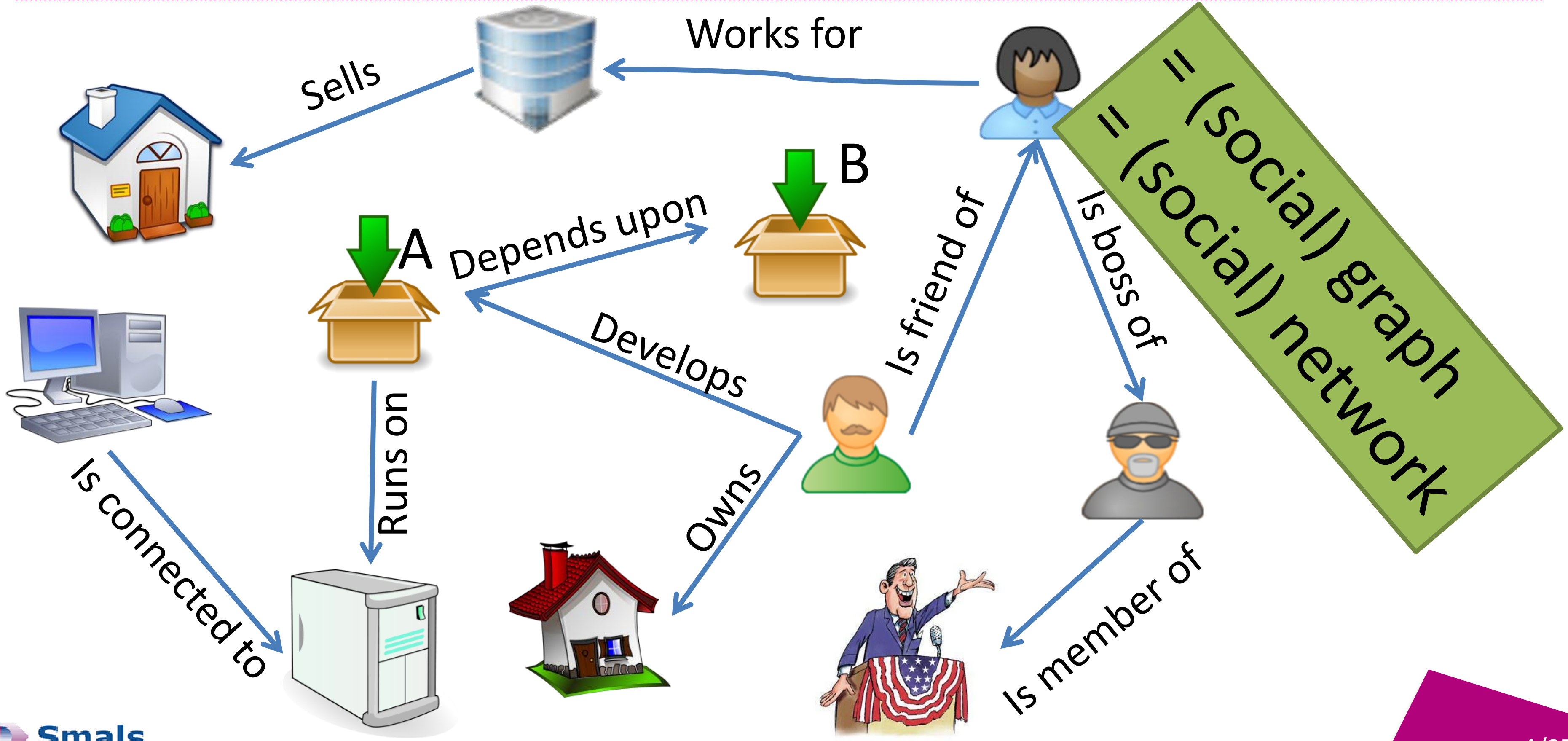
Neo4j

Use case : Social dumping

Conclusions

MOTIVATIONS

Motivations



Starting points

- Graphs excel in model 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

Workers
ID
Name
Employer_ID

Companies
ID
Name

Smals' Employees?

How relationships are implemented

What concerns developers

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

RDBMS : Relational DataBase Management System

RDBMS and relationships

Workers
ID
Name

Works_for
Worker_ID
Company_ID
From_date

Companies
ID
Name

How relationships are implemented

Smals' Employees?

What concerns developers

```
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'
```

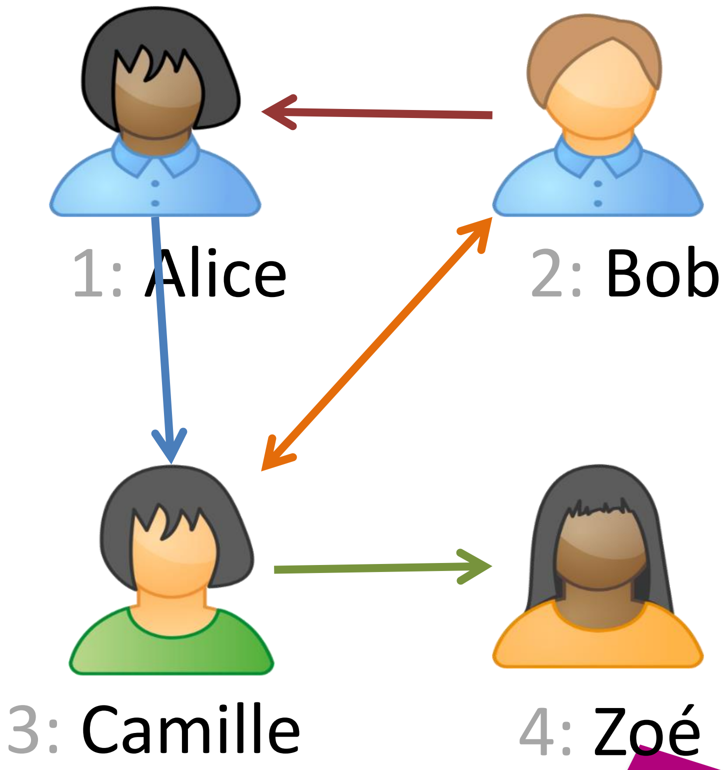
RDBMS and relationships

ID	Name
1	Alice
2	Bob
3	Camille
4	Zoé

Liker_ID	Liked_ID
1	3
2	1
2	3
3	2
3	4

```
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"
```

Who Bob likes?



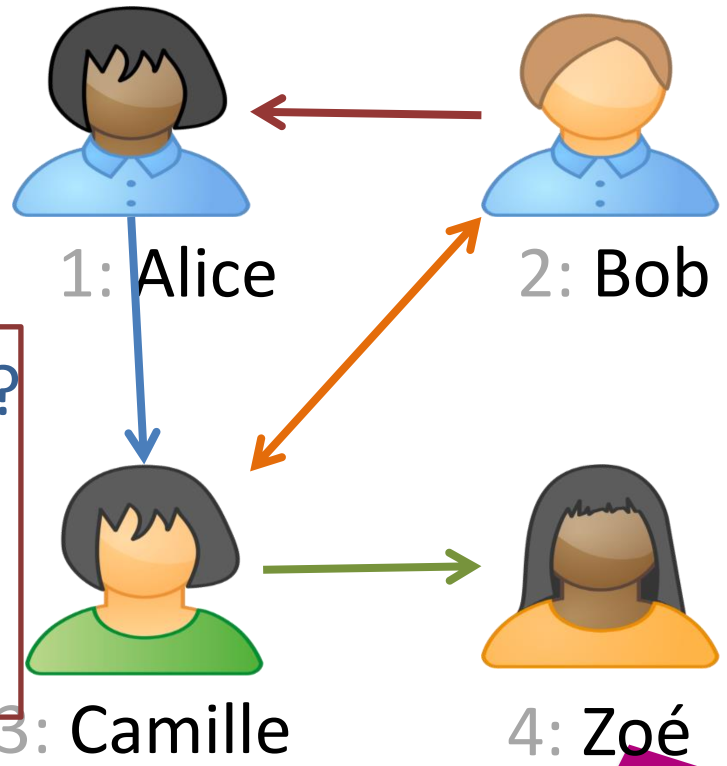
RDBMS and relationships

ID	Name
1	Alice
2	Bob
3	Camille
4	Zoé

Liker_ID	Liked_ID
1	3
2	1
2	3
3	2
3	4

```
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"
```

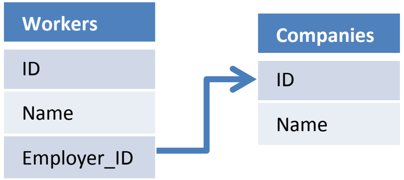
Who Bob $(\text{likes})^2$?



RDBMS and relationships: limitations

1-n relationships

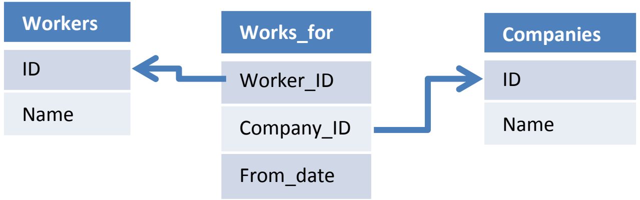
Foreign key:



- Transforms the role of attributes
- No type « Key »

m-n relationships

Join table:



- Transforms the role of tables
- No distinguishable

- Integrity constraints possible (extern to tables)
- Relationship structure to be given to each request
- Join : Heavy in writing code and in execution time

NoSQL solutions

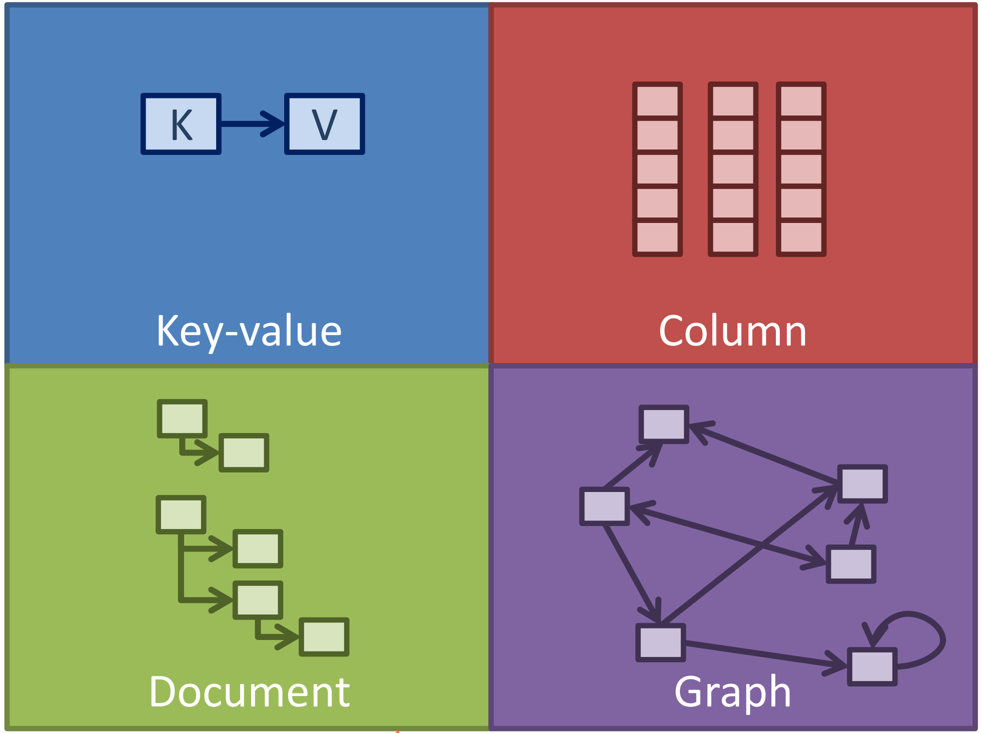
cassandra

redis

ORACLE NOSQL DATABASE

Google BigTable

APACHE HBASE



AllegroGraph

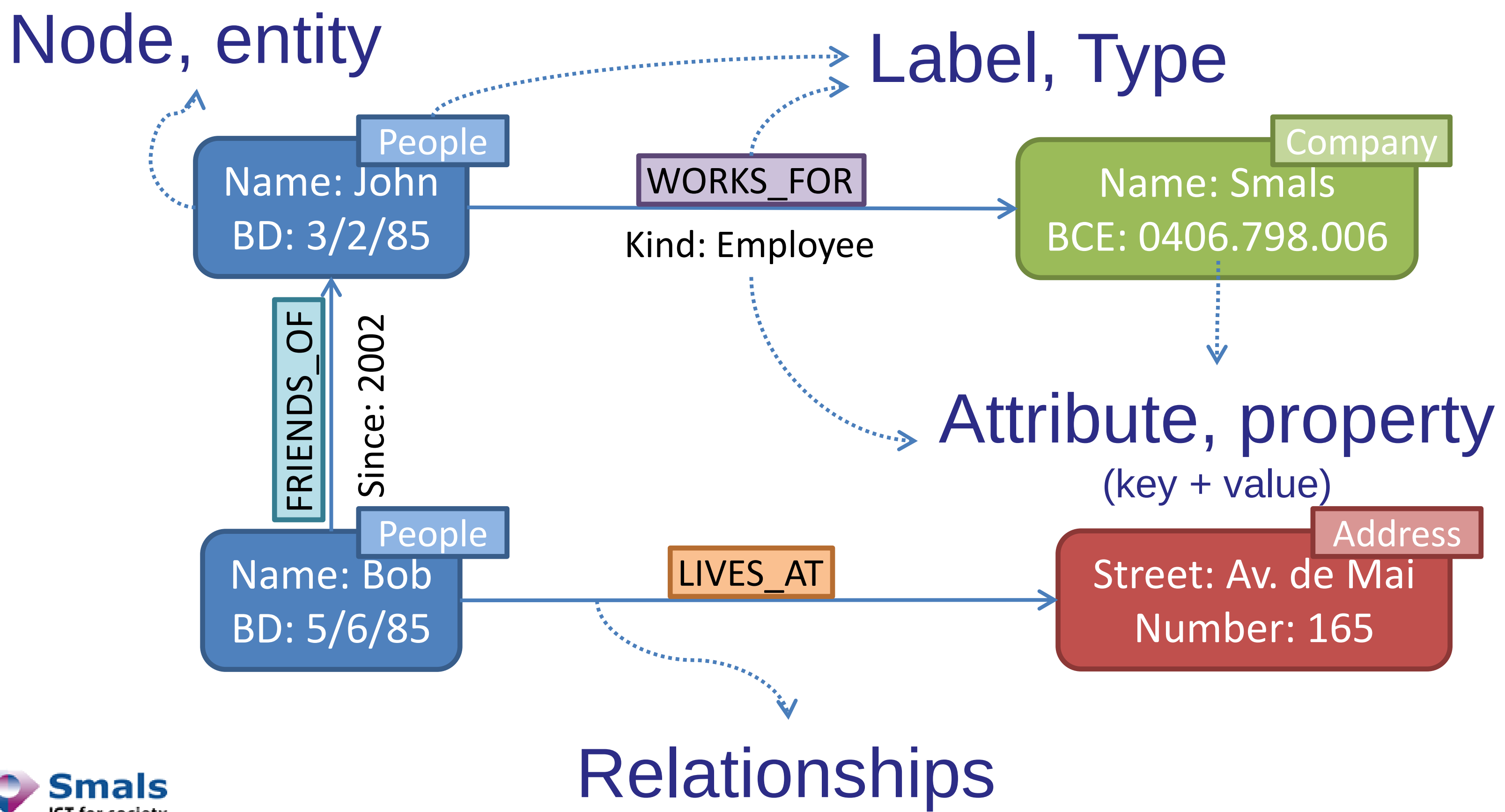
mongoDB

neo4j

InfiniteGraph
Powered by Objectivity

OrientDB

Graph database model



GraphDB: objectives

Objectives of graph databases:

Querying language **splitting apart relationship definition (creation time) and « matching »**

Fast search engine for relationship traversals

```
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"
```

Cypher (Neo4j)

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

GraphDB: less code

```
(SELECT T.directReportees AS directReportees, sum(T.count) AS count
FROM (
SELECT manager.pid AS directReportees, 0 AS count
  FROM person_reportee manager
  WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
UNION
  SELECT manager.pid AS directReportees, count(manager.directly_manages) AS count
FROM person_reportee manager
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
SELECT manager.pid AS directReportees, count(reportee.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee reportee
ON manager.directly_manages = reportee.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
SELECT manager.pid AS directReportees, count(L2Reportees.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_manages = L2Reportees.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
) AS T
GROUP BY directReportees)
UNION
(SELECT T.directReportees AS directReportees, sum(T.count) AS count
FROM (
SELECT manager.directly_manages AS directReportees, 0 AS count
FROM person_reportee manager
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
UNION
SELECT reportee.pid AS directReportees, count(reportee.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee reportee
ON manager.directly_manages = reportee.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
```

```
SELECT depth1Reportees.pid AS directReportees,
count(depth2Reportees.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_manages = L2Reportees.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
) AS T
GROUP BY directReportees)
UNION
(SELECT T.directReportees AS directReportees, sum(T.count) AS count
  FROM(
  SELECT reportee.directly_manages AS directReportees, 0 AS count
FROM person_reportee manager
JOIN person_reportee reportee
ON manager.directly_manages = reportee.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
SELECT L2Reportees.pid AS directReportees, count(L2Reportees.directly_manages)
AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON
JO
ON
WI
GR
) A
GR
UN
(SE
FR
JO
ON
JO
ON
WI
)
```

```
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
```

Tutorial Neo4j : https://youtu.be/-dCeFEqDkUI?t=246

GraphDB : less code

"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/

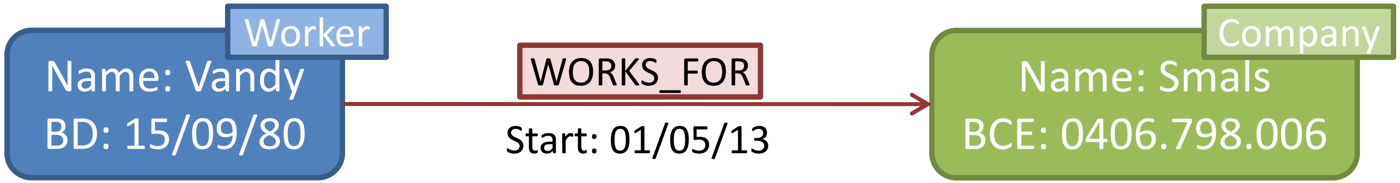
Smals
ICT for society

NEO4J

Neo4j

- Current Graph Database leader
- Written in Java, multi-plateform
- 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 »

Node of type « Company »

Relationship of type « WORKS_FOR »

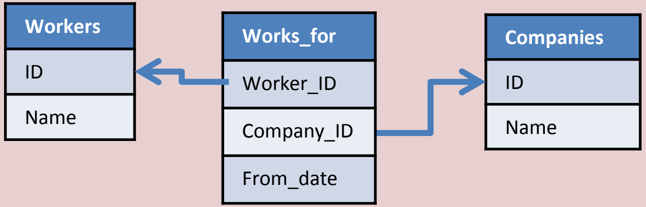
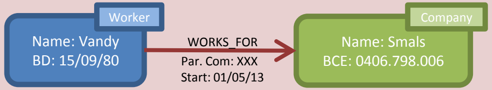
```
MATCH   (w:Worker)  -[r:WORKS_FOR]->  (c:Company)
WHERE   c.Name = "Smals"
RETURN  w.Name, w.BD, r.Start
```

Variante :

```
MATCH
  (w:Worker)
  -[r:WORKS_FOR]->
  (c:Company {Name:"Smals"})
RETURN …
```

Smals' employees

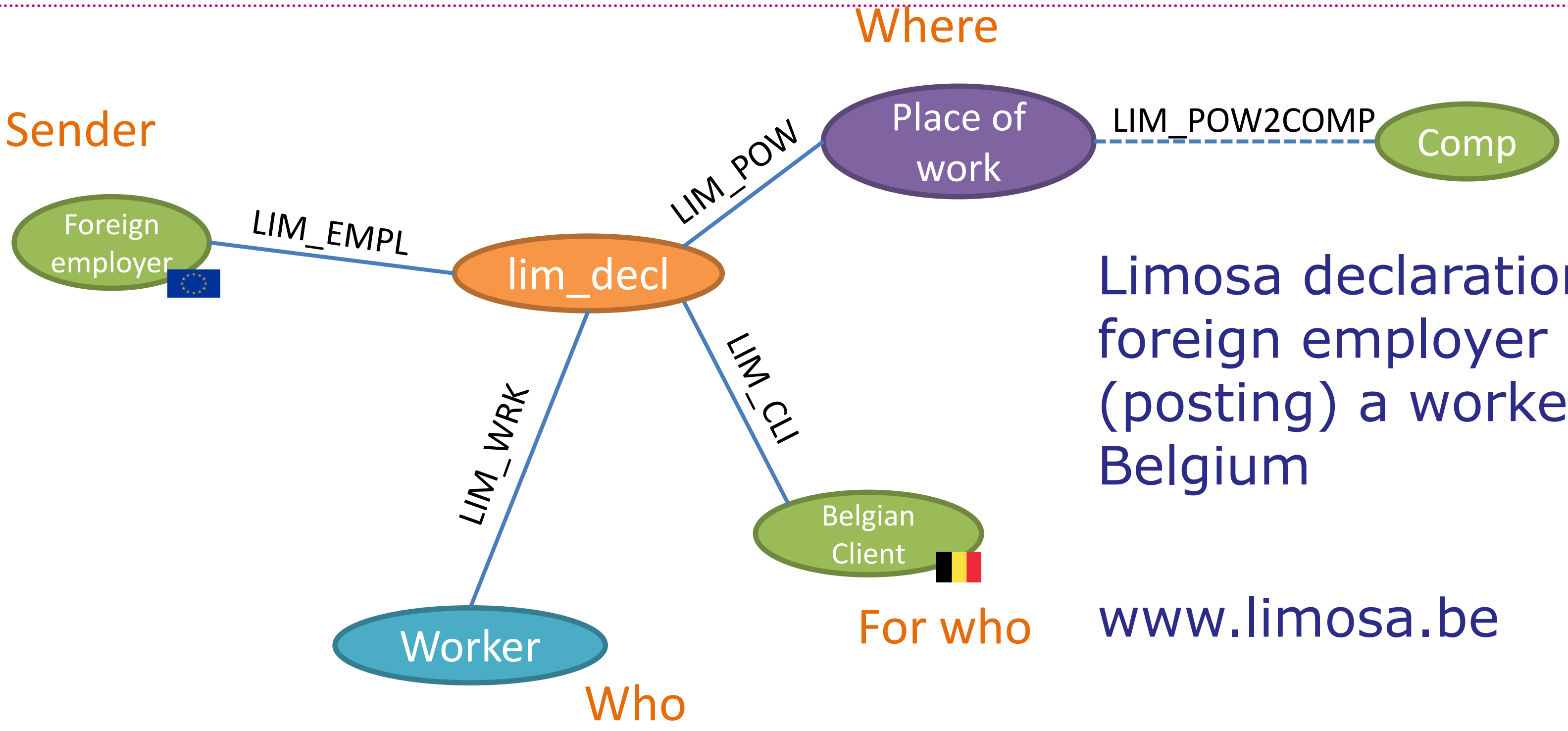
How it works

	RDBMS	Graph DB
schema		
WHERE …	Search in Companies (with index): O(log([Table size])	Search in nodes (with index) : O(log([DB size])
Relation	JOIN : - Search in Works_for : O(log([Table size])) - Search in Workers : O(log([Table size]) x [nb res])	Follow pointers in node : O([nb of employees])

Smals ICT for society

USE CASE: SOCIAL DUMPING

Limosa: detachment declaration



Limosa declaration: A foreign employer sending (posting) a worker in Belgium

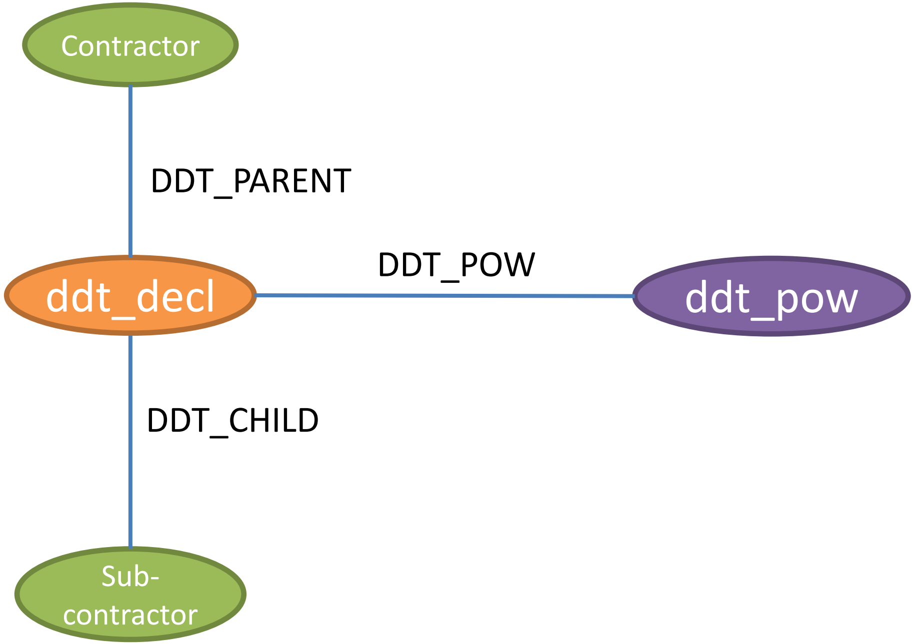
www.limosa.be

Limosa : Landenoverschrijdend Informatiesysteem ten behoeve van MigratieOnderzoek bij de Sociale Administratie

Syst. d'infor. transfrontalier en vue de la recherche en matière de migration auprès de l'adm. sociale

Smals
ICT for society

DDT: sub-contractors

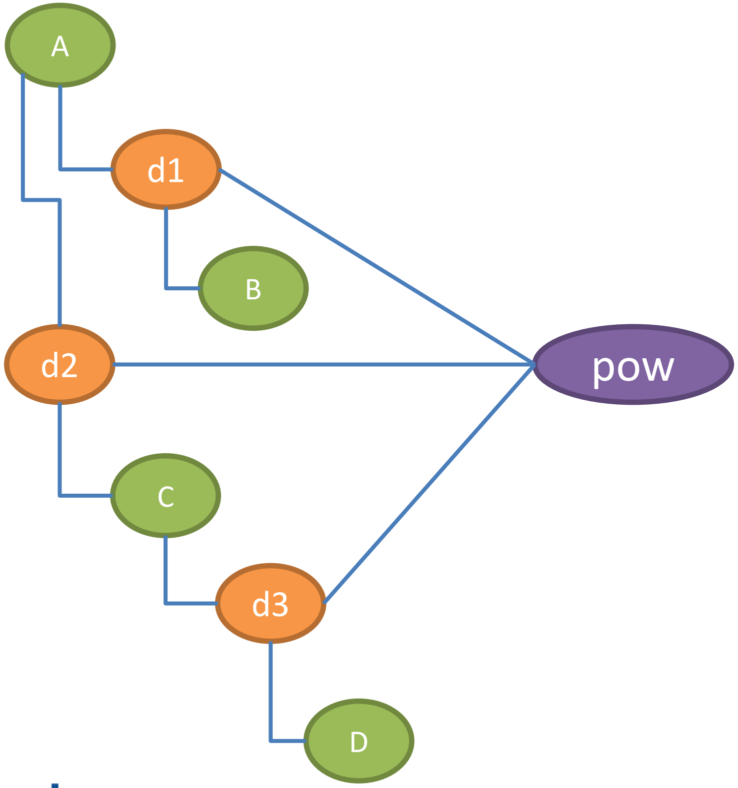


DDT: Déclaration De Travaux

Aangifte van werken

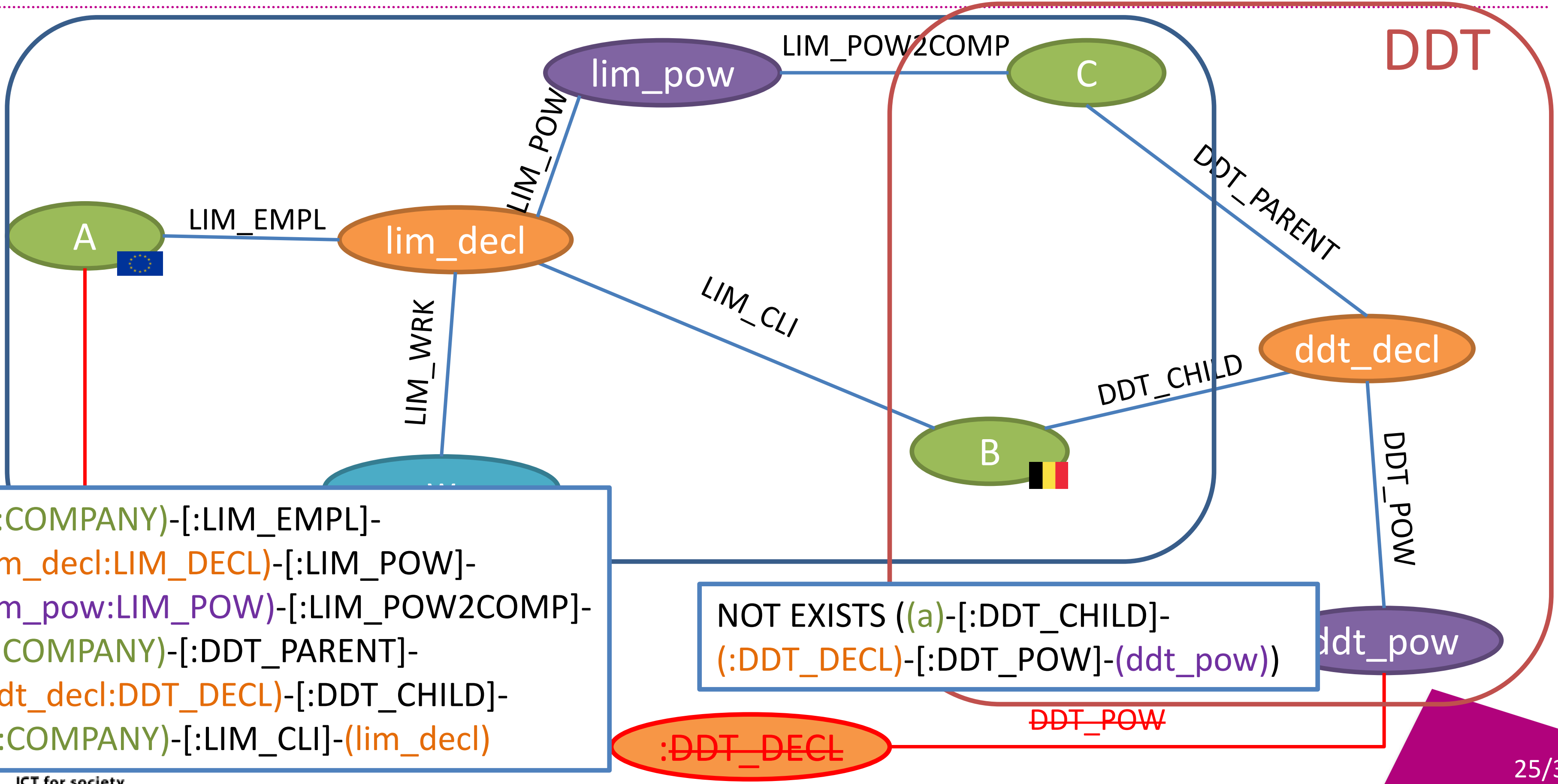
Smals
ICT for society

DDT: sub-contractors



On a specific worksite (PoW), we have a chain/tree of sub-contractors

Full schema



(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)

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

Full query

```
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 :

```
FROM Type1
JOIN REL1  ON Type1.key = REL1.type1_key
JOIN Type2 ON Type2.key = REL1.type2_key
```

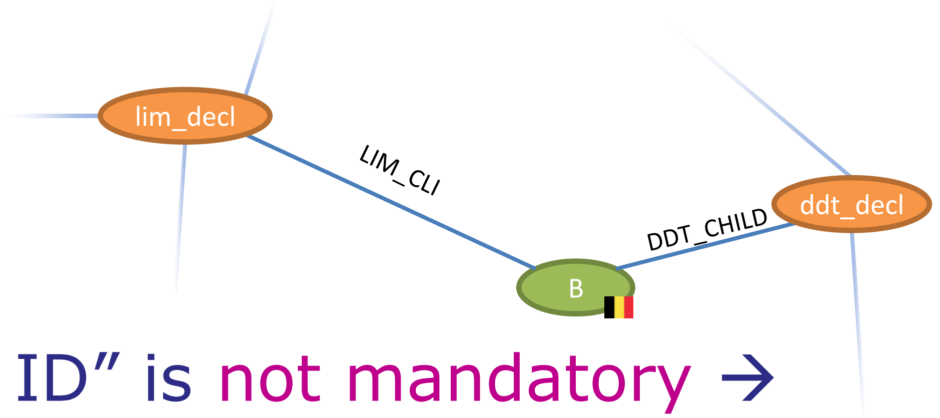
→ Almost 20 joins!

- → Query far more complex to write
- → Much longer to run

Smals
ICT for society

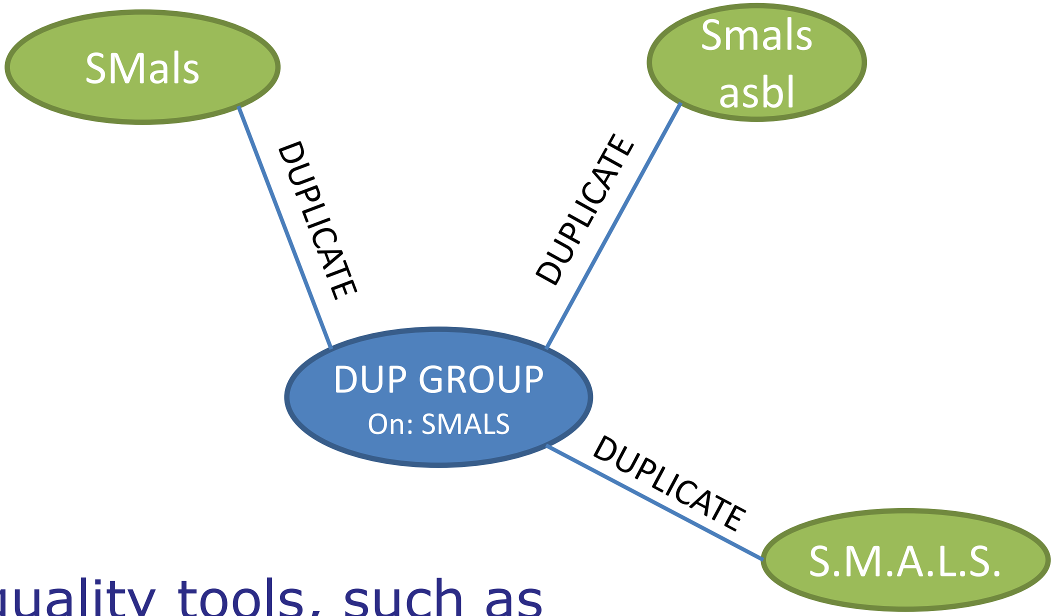
Data quality problem

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



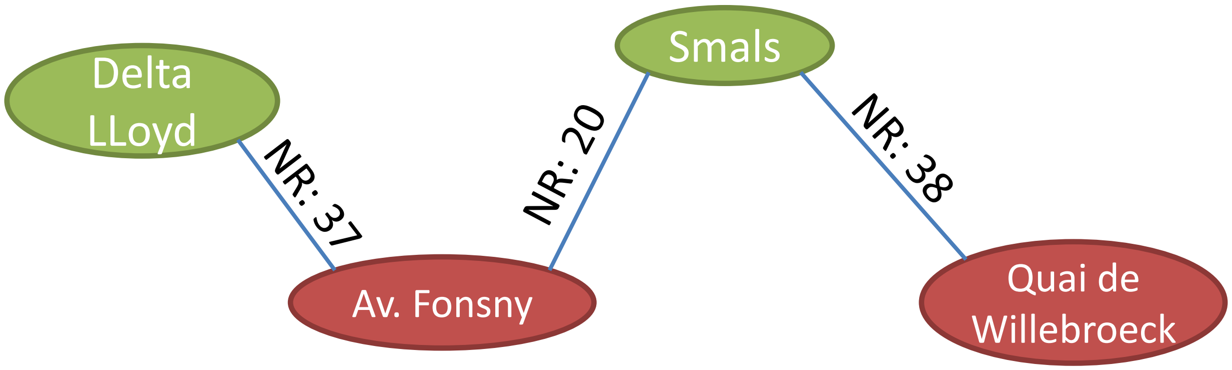
- Sometimes, the “official ID” is not mandatory → name + address is OK
- We have to deal with this!

Duplicates



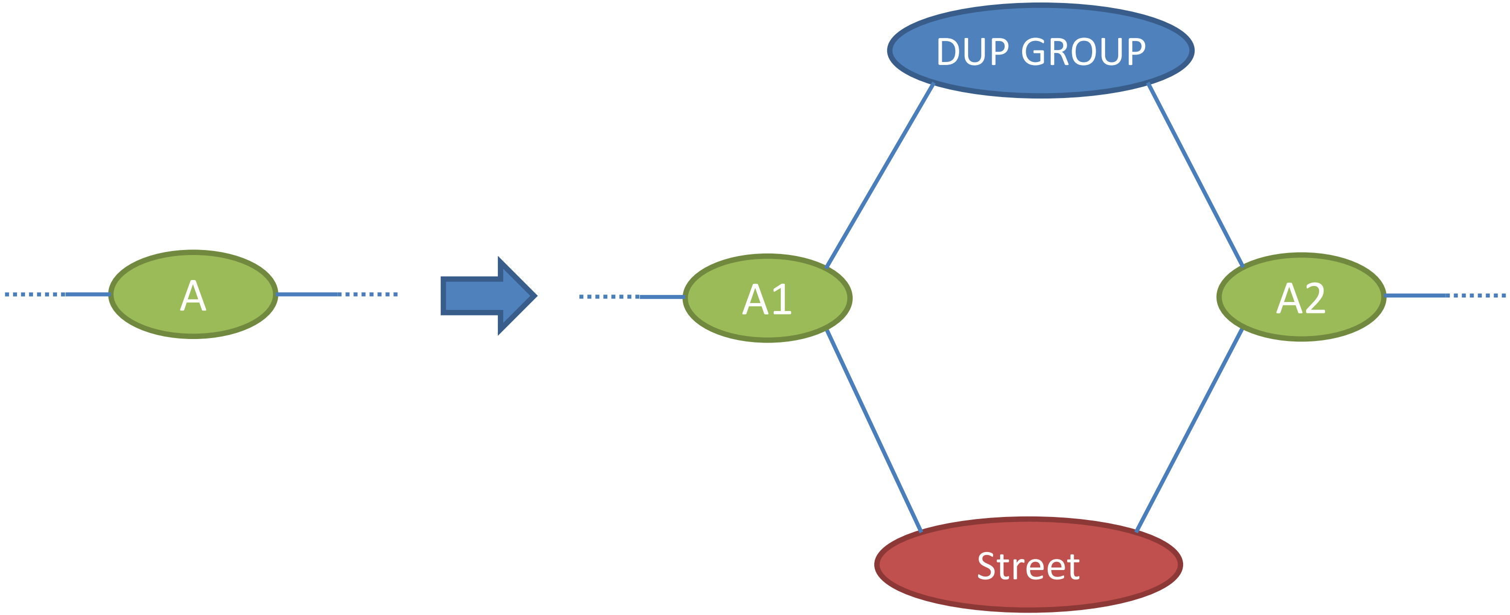
Thanks to data quality tools, such as Trillium (IntoDQ), OpenRefine…

Addresses

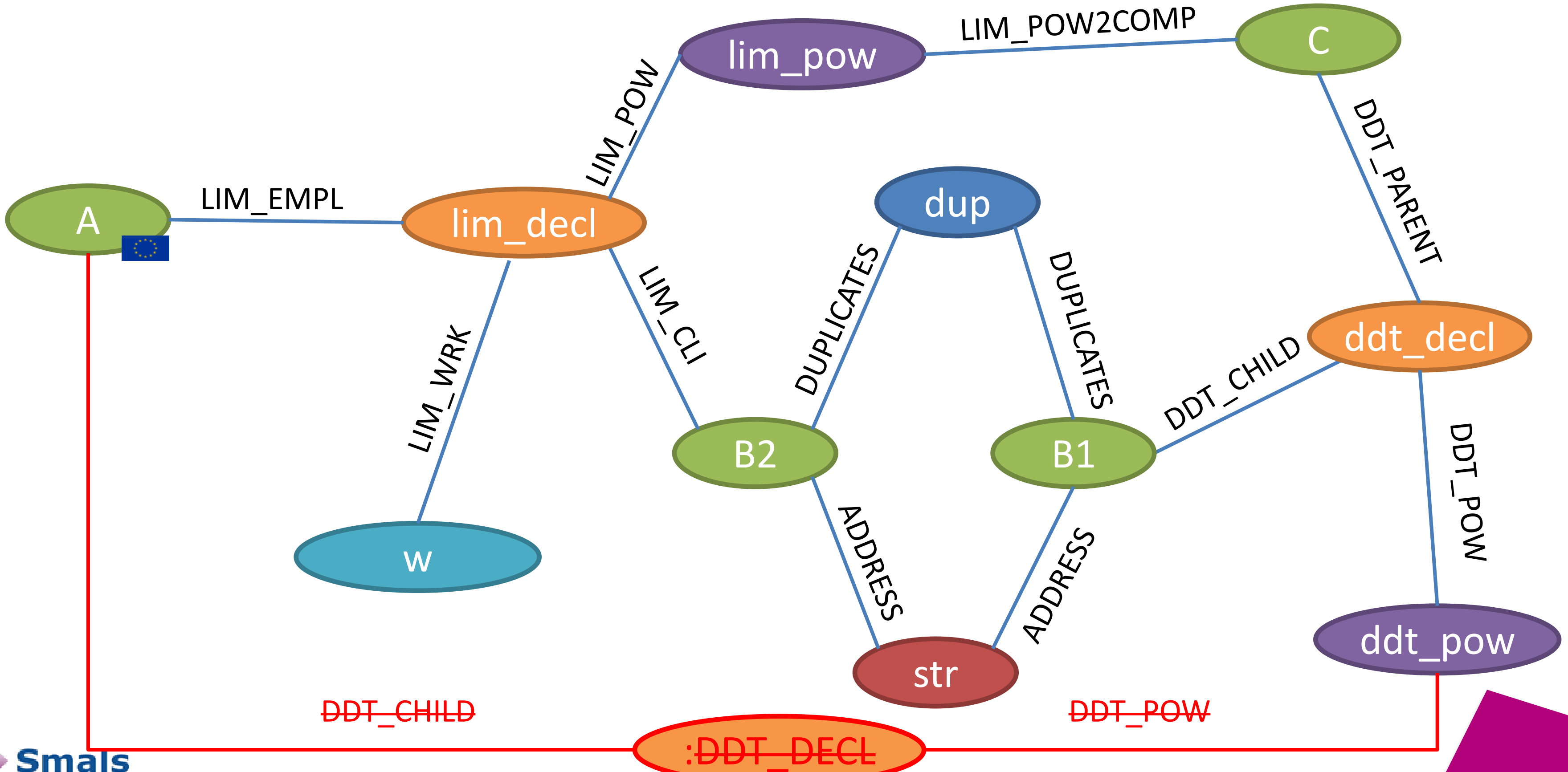


A standardization is needed! Thanks to data quality tools

Duplicates combinations



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…

Smals
ICT for society

References : www.smalsresearch.be

- Blogs (in French):
  - https://www.smalsresearch.be/un-fraudeur-ne-fraude-jamais-seul/
  - https://www.smalsresearch.be/un-fraudeur-ne-fraude-jamais-seul-partie-2/
  - https://www.smalsresearch.be/bases-de-donnees-relationnelles-adequates-pour-des-relations/
  - https://www.smalsresearch.be/graph-db-vs-rdbms/
- Other:
  - NoSQL - Hype ou Innovation, Grégory Ogonowski, https://www.smalsresearch.be/publications/document/?docid=59

References

- **Graph Databases**, *Robinson & all*, O'Reilly 2015
- **Fraud Detection: Discovering Connections with Graph Databases**, *Neo4j*, Sodawksi & Rathle
- **Fraud Analytics Using Descriptive, Predictive, and Social Network Techniques: A Guide to Data Science for Fraud Detection**, *Bart Baesens, Veronique Van Vlasselaer, Wouter Verbeke*, Willey, 2015

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