Neo4j overview

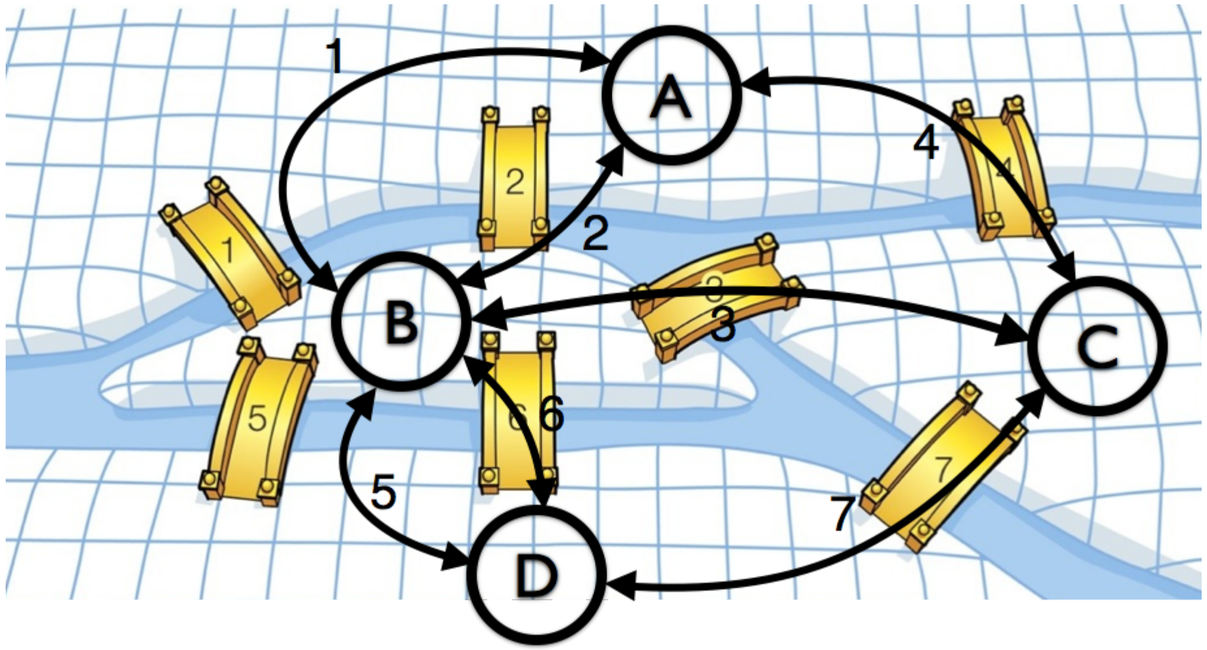
Benoit Simard (@logisima)

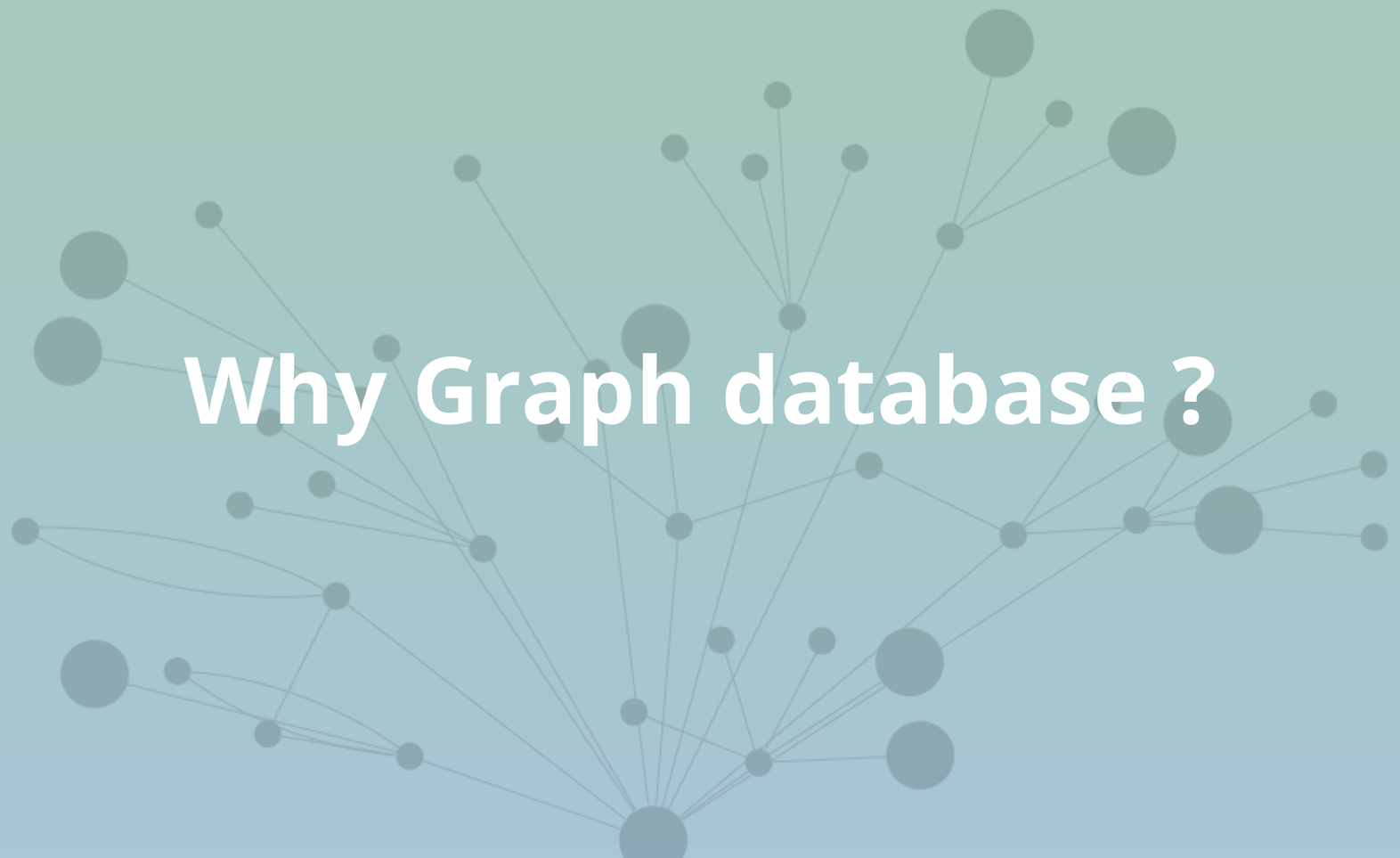
neo4j

History

Leonhard Euler 1707-1783

Konigsbergs

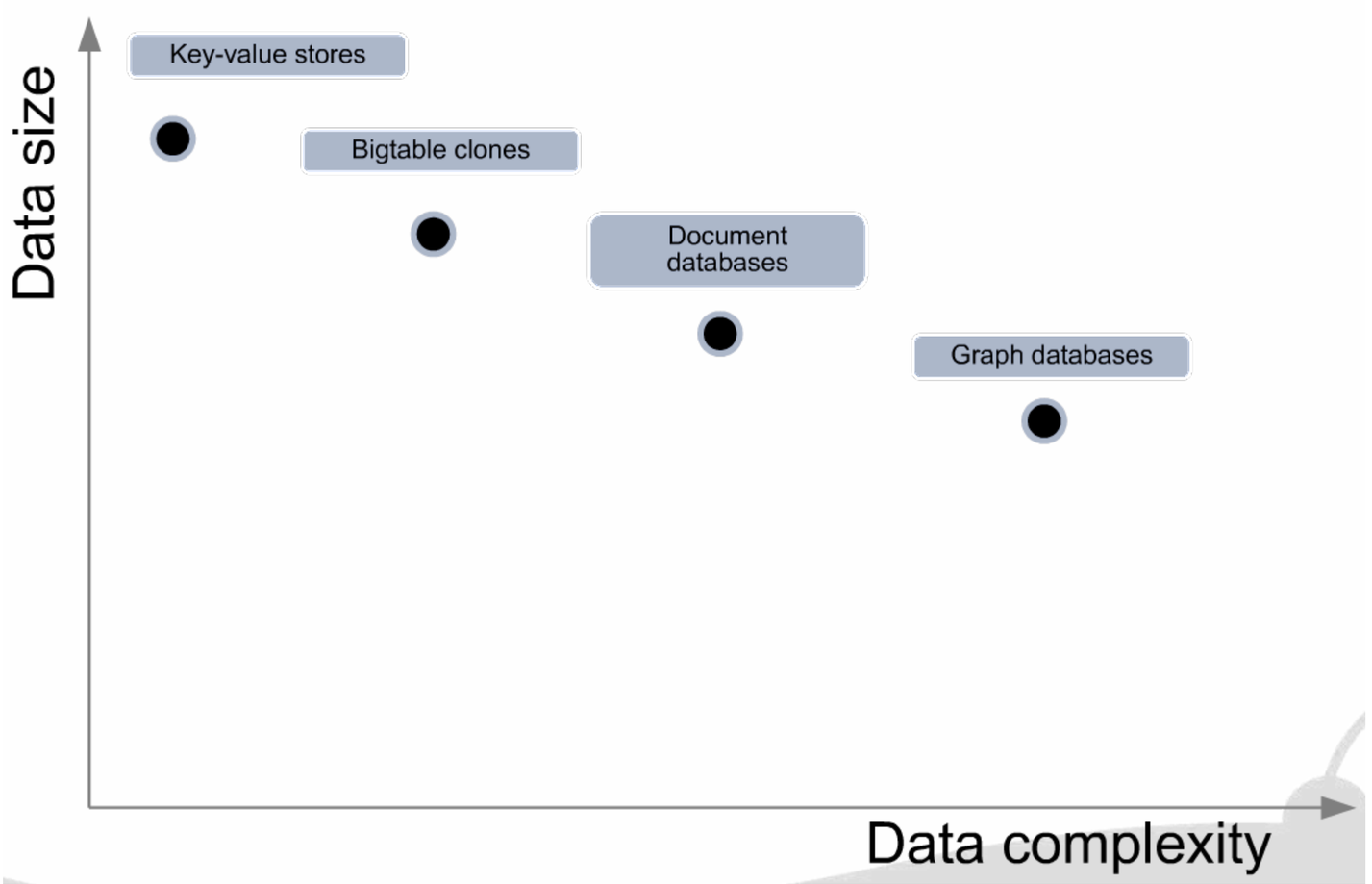




Why Graph database ?

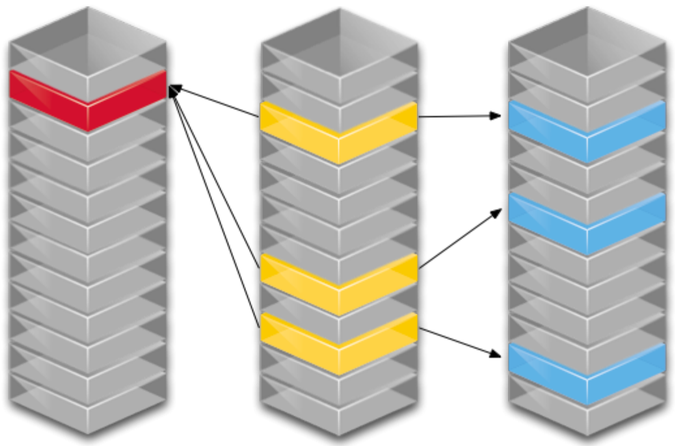
neo4j

Data complexity



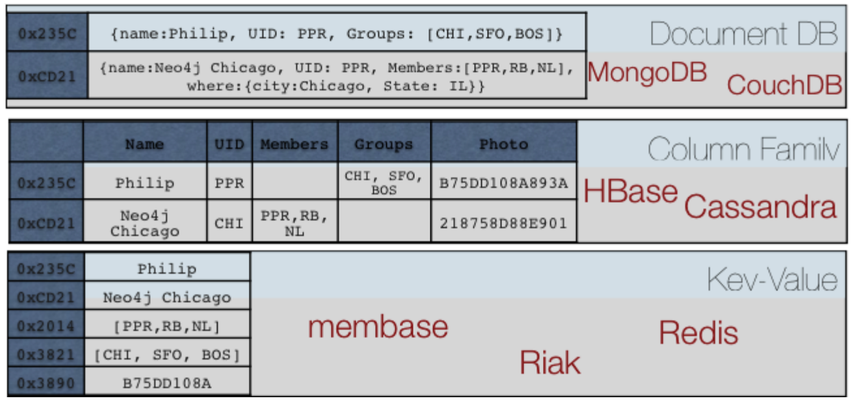
Linked Data

Relation in RDBMS



- schema not flexible
- poor performances on linked data (@see snowflake schema)
- complexe queries

Relation in No-SQL



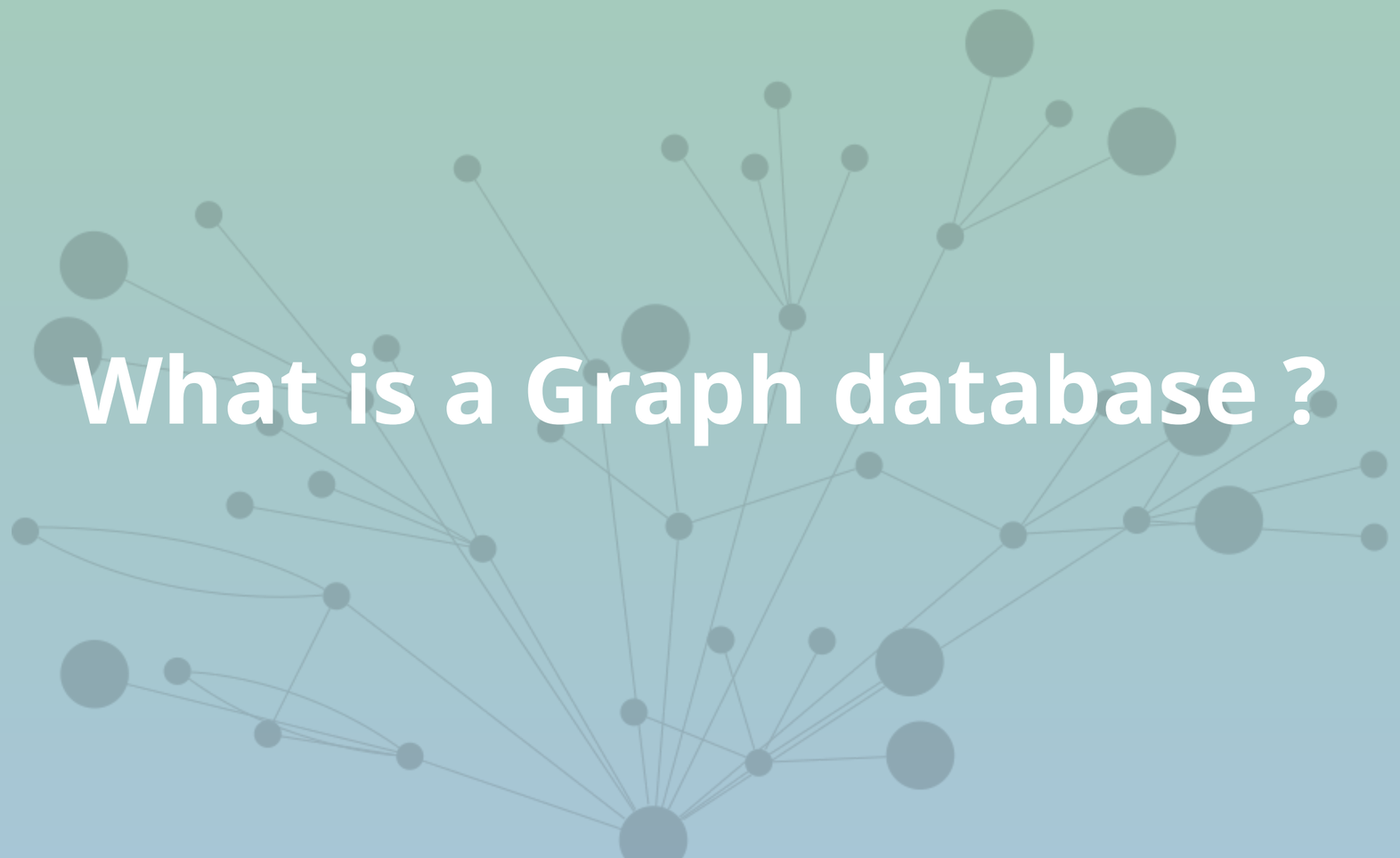
Graph database

When to use it ?

- Do you need a flexible schema ?
- Do your data are highly connected ?
- Do you need near real time ?

Recommendation, fraud detection, MDM, network analysis, impact analysis, ACL, ...

What is a Graph database ?

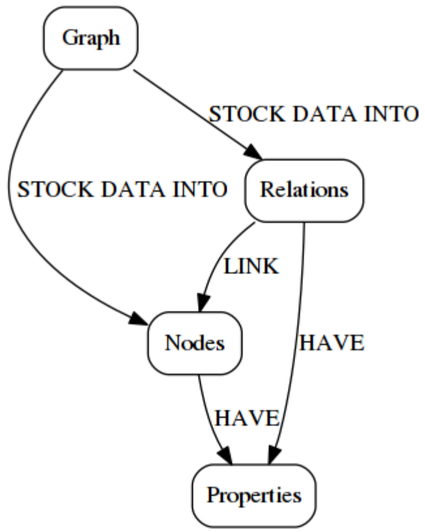


neo4j

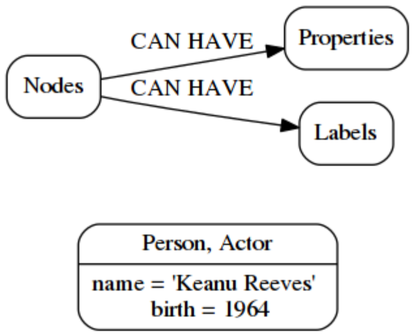
Neo4j

- Since 2010
- License under GPLv3
- Full ACID
- HA
- Schemaless

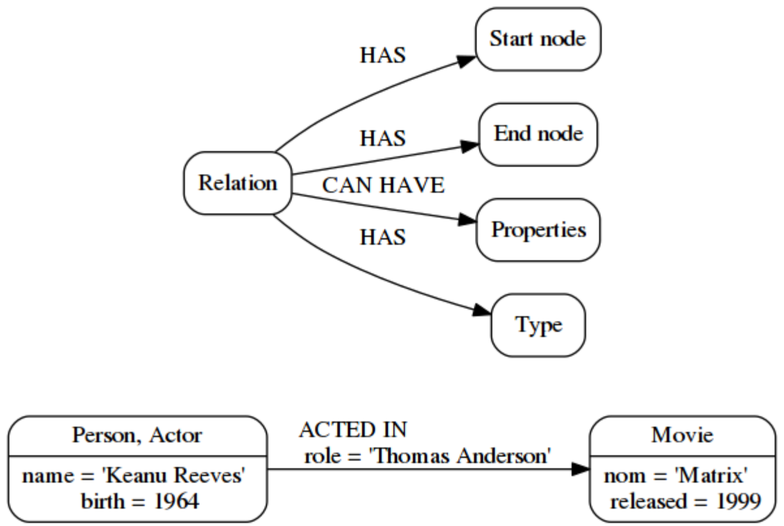
Concept


Graph
STOCK DATA INTO
STOCK DATA INTO
Relations
LINK
Nodes
HAVE
HAVE
Properties

Nodes


CAN HAVE
Properties
Nodes
CAN HAVE
Labels
Person, Actor
name = 'Keanu Reeves'
birth = 1964

Relationships



Index & constraint

Indexes

- on a property for all nodes that have a label

Constraint

- for the uniqueness of a property
- for the existence of a property

Top use-cases

Yeah !

(Graph)-[:ARE]→(Everywhere)

Moteur de recommandation en temps réel

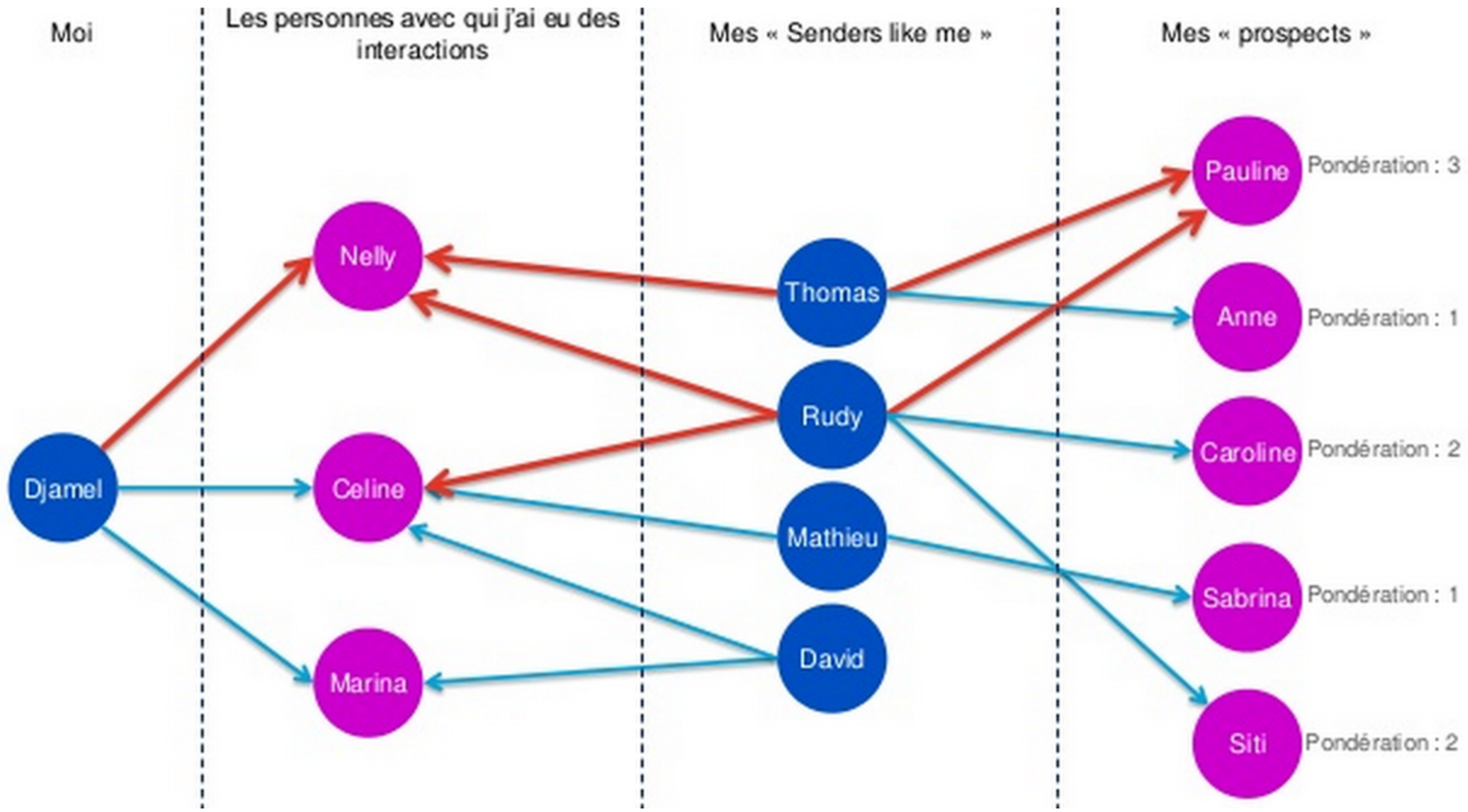
neo4j

La recommandation

Des réseaux sociaux, à la vente au détail, de service ou dans le secteur des médias, la recommandation est partout !

Objectif : offrir une recommandation en temps réel, hautement ciblés et contextuelles afin quelle soit pertinentes pour l'utilisateur

Etude de cas : Meetic



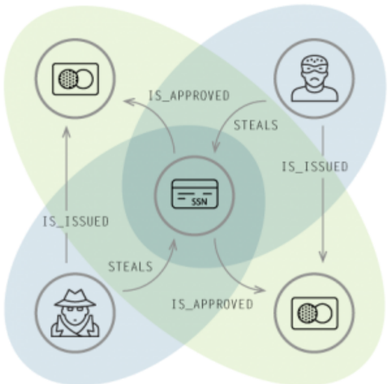
La détection de fraudes

neo4j

Pourquoi du graphe ?

Bien qu'aucune mesure de détection de fraudes ne soit parfaite, il est possible d'introduire des améliorations considérables si l'on se penche sur les connexions qui relient les points de données individuels.

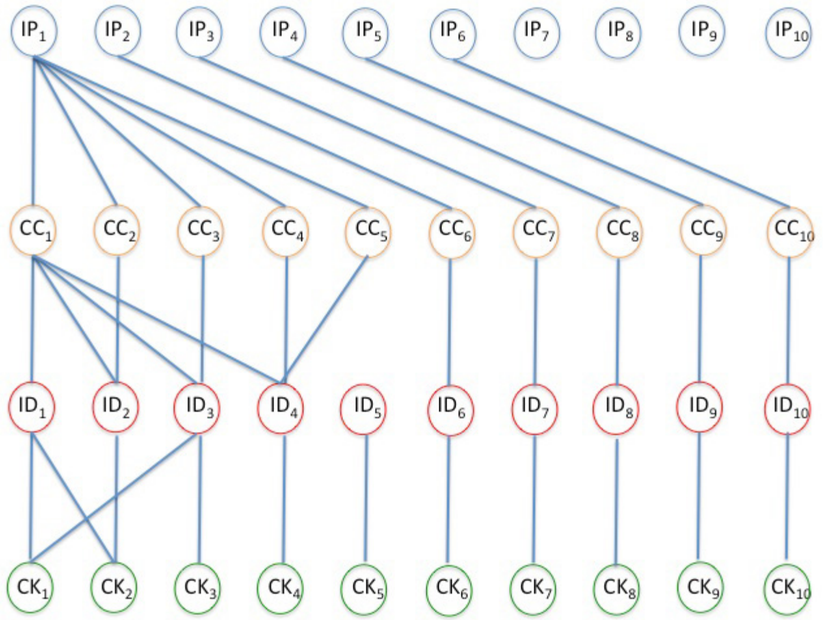
Il faut étudier l'interconnexion des données



Etude de cas : Fraude dans le e-commerce

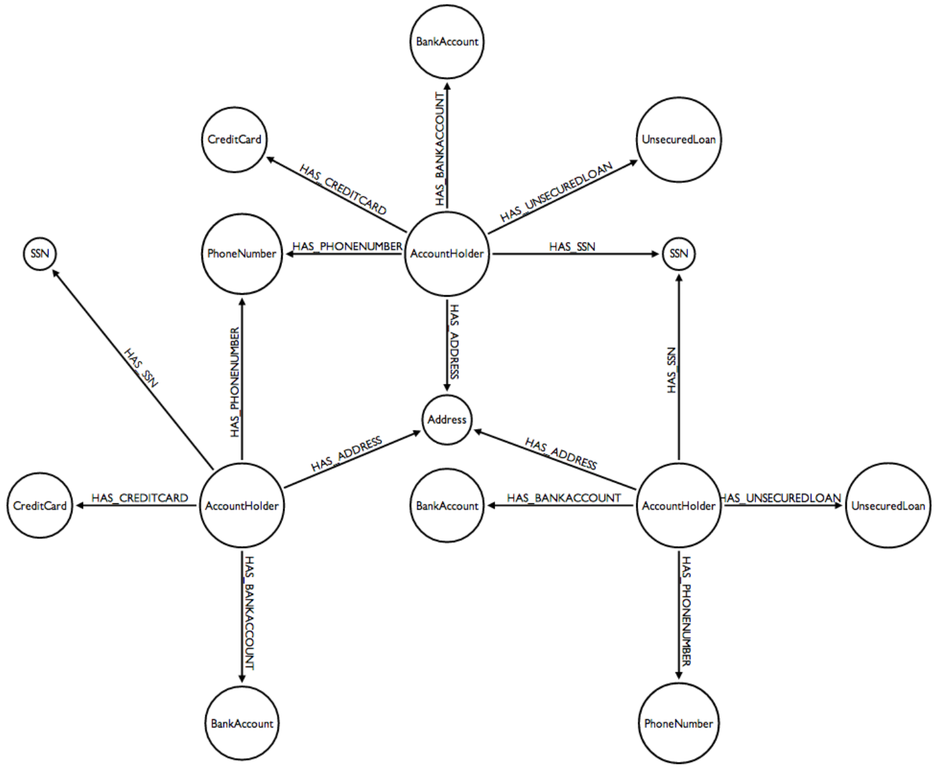
Graphe d'une série de transactions effectuée avec fraude probable se produisant à l'adresse IP1.

(IP : Adresse IP, CC : carte de crédit, ID : Identifiant de l'utilisateur, CK : Adresse de livraison)



Etude de cas : La détection d'anneau

Constituer un ensemble de fausses identités à partir d'identités volées, dans l'objectif de commettre un acte frauduleux

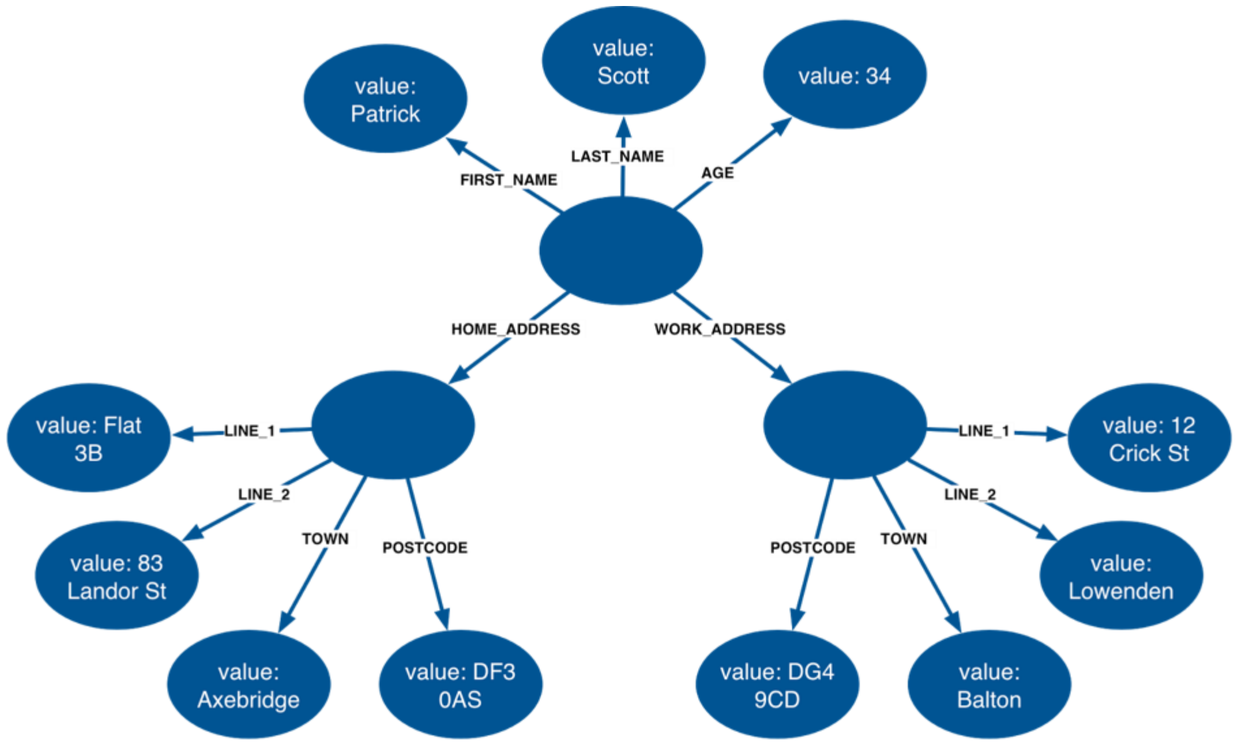


La gestion de données de référence

neo4j

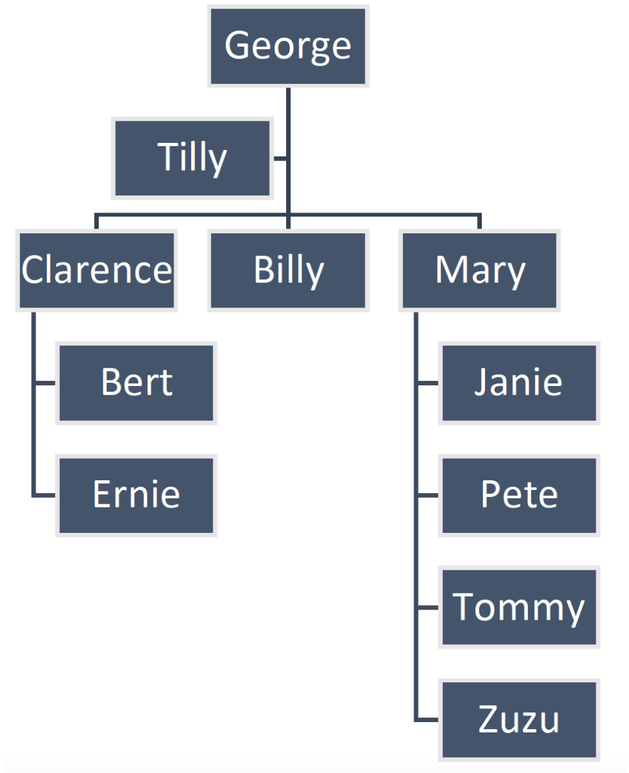
Reconciliation des données

Réconcialiation des données par degré de similitude



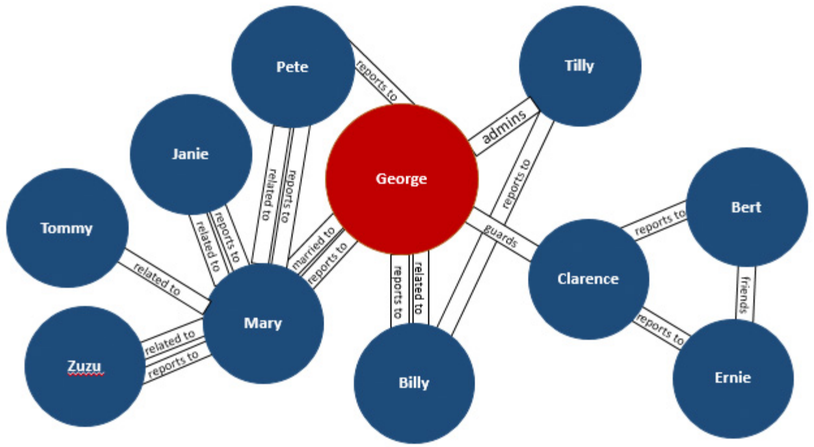
La modélisation RH d’une entreprise

Hiérarchie de données de référence illustrant des structures de rapport et de supervision d’employé.



La modélisation RH d’une entreprise

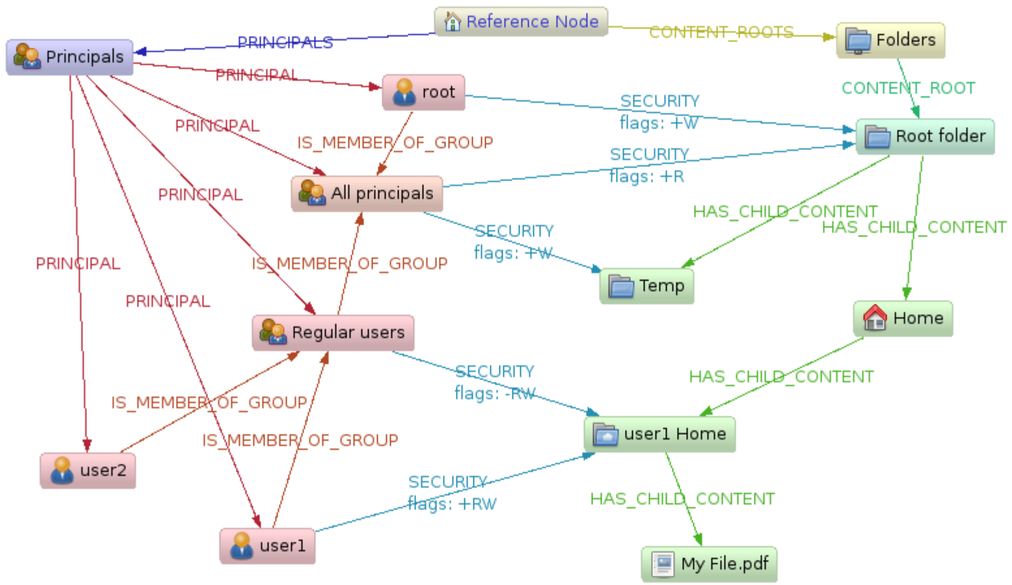
Réseau de données de référence détaillant les relations de rapport et de supervision d’employé, ici avec une plus grande complexité du **monde réel**.



La gestion des identités et des accès

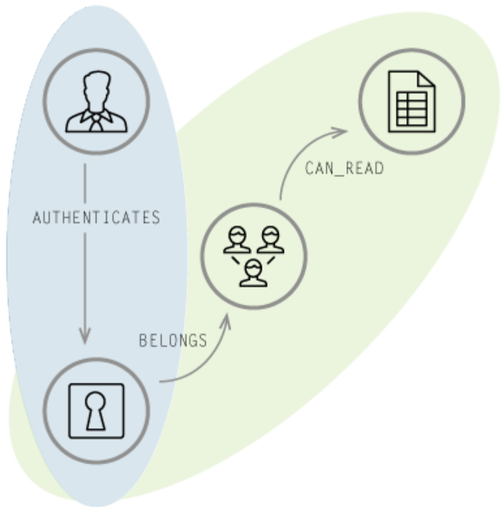
neo4j

La gestion des identités et des accès



Pourquoi du graphe ?

C’est un graphe !

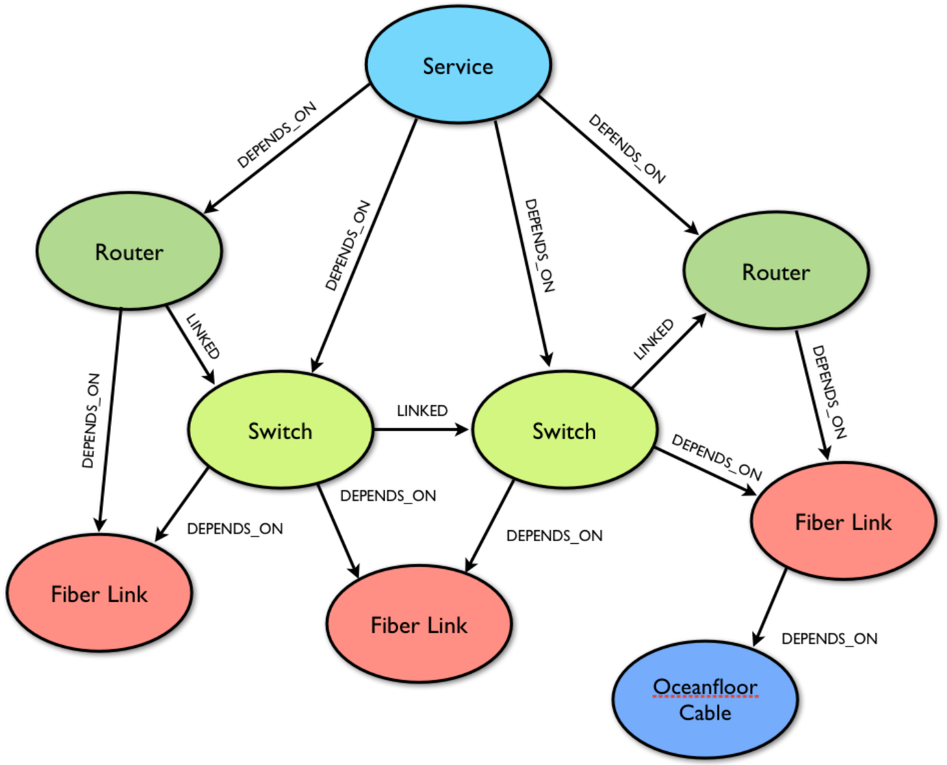


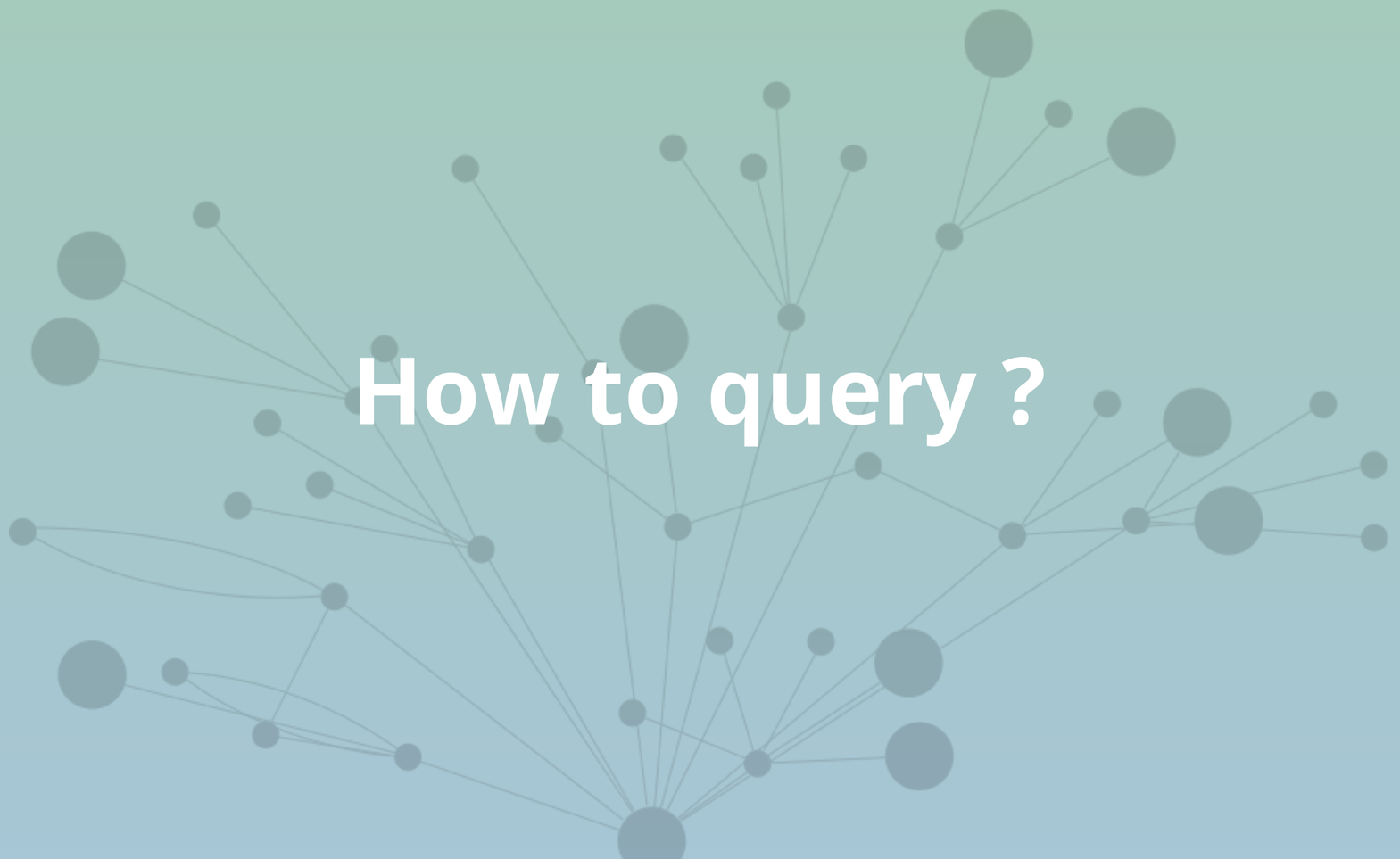
Les réseaux informatiques

neo4j

Analyse d’impact chez SFR

Modélisation de leur reseau physique dans Neo4j.





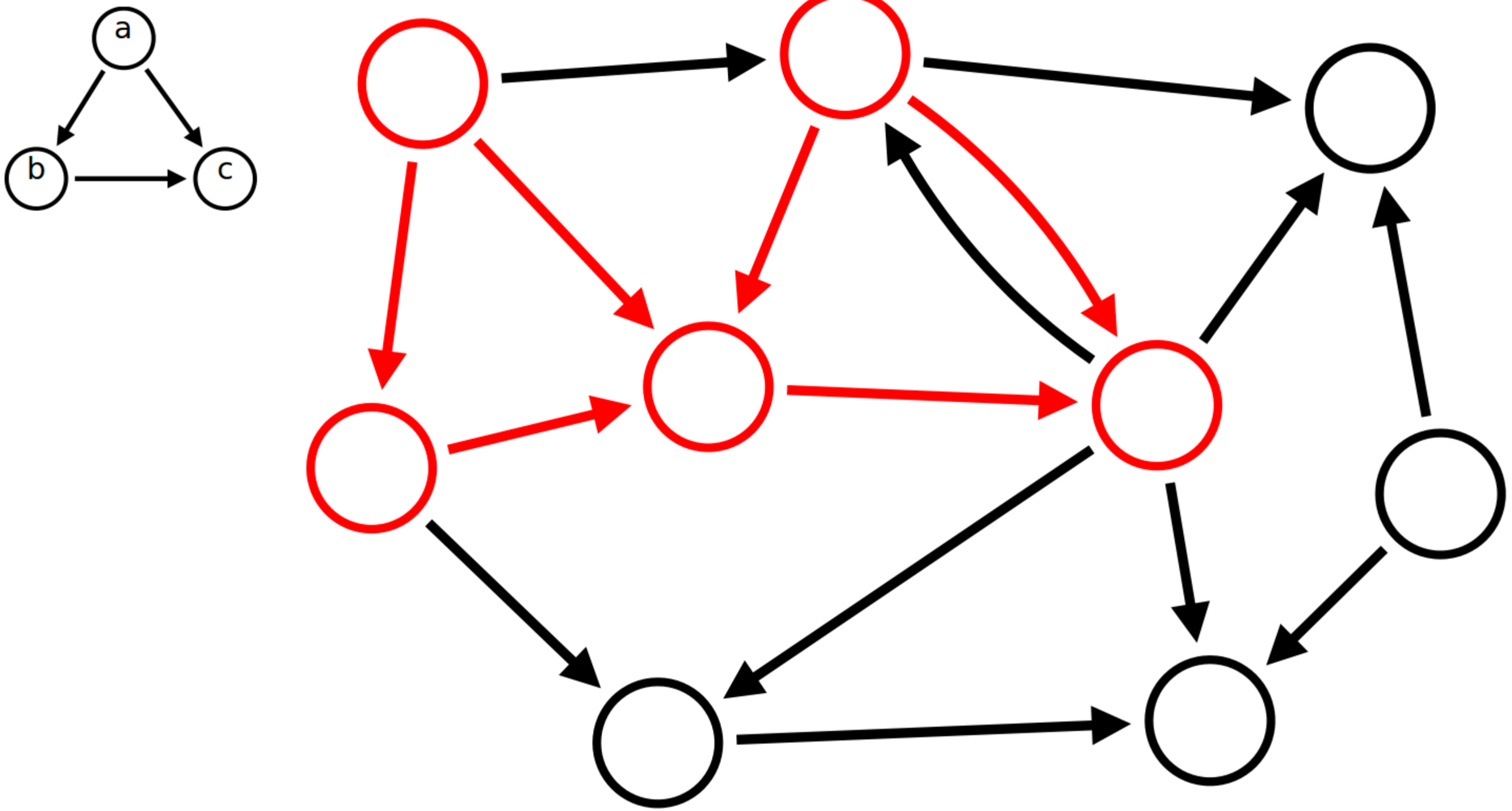
How to query ?

Cypher

Cypher is Neo4j's graph query language

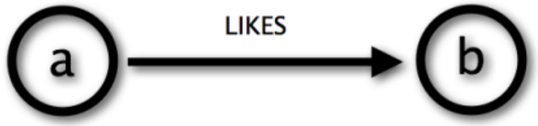
- Declarative Pattern-Matching language
- SQL-like syntax
- Designed for graphs

All is pattern



Ascii Art

Cypher using relationship 'likes'



Cypher

```
(a) -[:LIKES]-> (b)
```

Let see it in action

Indexes

```
1 CREATE INDEX ON :Person(name)
2 CREATE INDEX ON :Movie(title)
```

Query

```
MATCH (a)-[:ACTED_IN]->(m)<-[:DIRECTED]-(a)
RETURN a.name, m.title;
```

List all the characters in the movie “The Matrix”

```
MATCH (matrix:Movie)<-[r:ACTED_IN]-(actor)
WHERE matrix.title="The Matrix"
RETURN r.roles, actor.name
```

Query

```
MATCH (tom:Person)-[:ACTED_IN]->(movie:Movie)
WHERE tom.name="Tom Hanks"
AND movie.released < 1992
RETURN movie.title
```

Query

```
MATCH (actor:Person)-[r:ACTED_IN]->(movie)
WHERE actor.name="Keanu Reeves"
AND "Neo" IN (r.roles)
RETURN movie.title
```

Query

```
MATCH (tom:Person)-[:ACTED_IN]->(movie),
      (a:Person)-[:ACTED_IN]->(movie)
WHERE tom.name="Tom Hanks"
  AND a.born < tom.born
RETURN DISTINCT a.name, (tom.born - a.born) AS diff
```

Query

```
MATCH (keanu:Person)-[:ACTED_IN]->(movie),
      (n)-[:ACTED_IN]->(movie),
      (hugo:Person)
WHERE keanu.name="Keanu Reeves" AND
      hugo.name="Hugo Weaving"
AND NOT (hugo)-[:ACTED_IN]->(movie)
RETURN DISTINCT n.name
```

Who are the five busiest actors?

```
MATCH (a:Person)-[:ACTED_IN]->()
RETURN a.name, count(*) AS count
ORDER BY count DESC
LIMIT 5
```

Recommend 3 actors that Keanu Reeves should work with (but hasn't).

```
MATCH (keanu:Person)-[:ACTED_IN]->()<-[:ACTED_IN]-(c),
      (c)-[:ACTED_IN]->()<-[:ACTED_IN]-(coc)
WHERE keanu.name="Keanu Reeves"
  AND coc <> keanu
  AND NOT((keanu)-[:ACTED_IN]->()<-[:ACTED_IN]-(coc))
RETURN coc.name, count(coc)
ORDER BY count(coc) DESC
LIMIT 3
```

Add KNOWS relationships between all actors who acted in the same movie

```
1 MATCH (a:Person)-[:ACTED_IN]->()<-[:ACTED_IN]-(b:Person)
2 MERGE (a)-[:KNOWS]-(b);
```

Friends-of-Friends

```
1 MATCH (keanu:Person)-[:KNOWS*2]-(fof)
2 WHERE keanu.name="Keanu Reeves"
3 RETURN DISTINCT fof.name
```

Query

```
MATCH (keanu:Person)-[:KNOWS*2]-(fof)
WHERE keanu.name="Keanu Reeves"
  AND keanu <> fof AND NOT (keanu)-[:KNOWS]-(fof)
RETURN DISTINCT fof.name
```

Query

```
MATCH p=shortestPath(
  (charlize:Person)-[:KNOWS*]-(bacon:Person)
)
WHERE charlize.name="Charlize Theron" AND
      bacon.name="Kevin Bacon"
RETURN length(rels(p))
```

Return the names of the people joining Charlize to Kevin.

```
MATCH p=shortestPath((charlize:Person)-[:KNOWS*]-(bacon:Person))
WHERE charlize.name="Charlize Theron" AND
      bacon.name="Kevin Bacon"
RETURN [n in nodes(p) | n.name] AS names;
```

QUESTION ?

PROCHAIN ÉVÈNEMENTS

- **Graph Day Paris :** mercredi 21 Septembre
- **Training Graph Modeling :** mardi 25 Octobre à Paris