NoSQL project report

Team 7

Bouchoux Adrien
GoutteFarde Léo
Nahyl Othmane

Wednesday 5th December 2016

Contents

1 NoSQL system choice

For this project, we chose the Neo4j system.

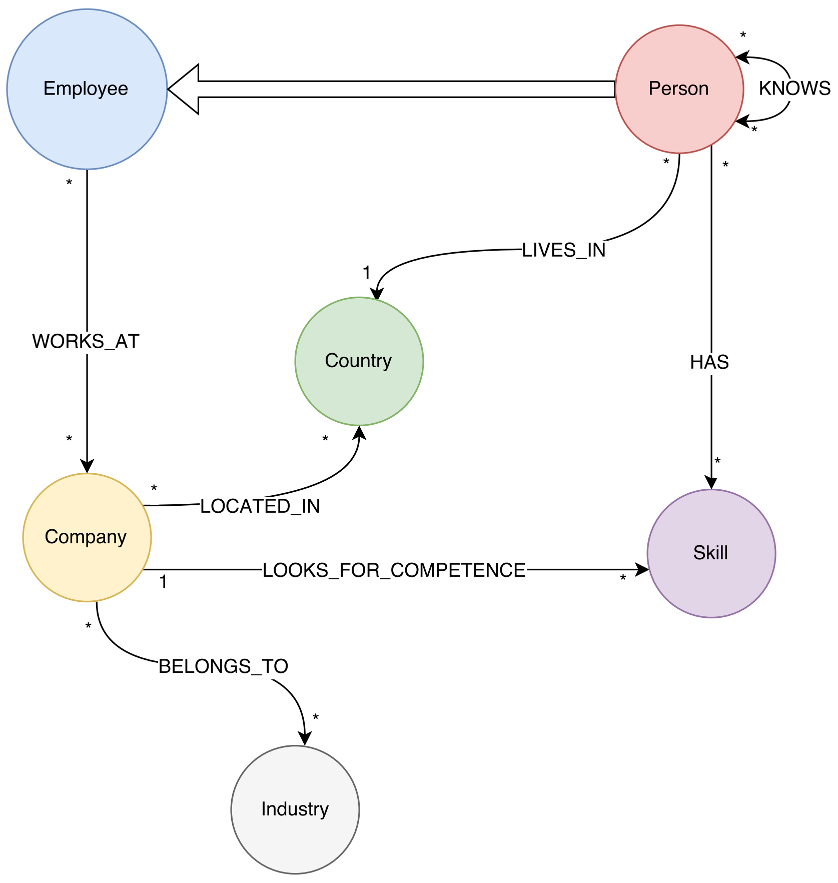
2 Data stored in the database

The database describes relationships between persons and companies established in several industries. Each person has a name and a birth date attribute. A person can have several skills with a specific level, and lives in a country (whose only attribute is the name of the country). Any person can also be an employee, ending up having 2 labels.

A company has a name as attribute, and is specialized in one or several industries (eg. Oil, Farming, ...) and established in one or several countries. Some companies are looking for specific skills. Every employee may work in a company. In this case, the start date and the salary can be stored in the relation between the person and the company.

Persons may know each other. The relationship level can be stored as a relation property.

For more informations, please look at the figure bellow :



3 Indexes

We use an index on each node type's name property, and another index for the birth date in nodes of type Person. Indeed we use a name attribute on each node type to identify every instance, therefore an index will speed up identification requests a lot.

4 Queries

4.1 Query 1

Description

This query retrieves companies and the number of experts and intermediates (all skills included) that they are looking for.

Code

```
MATCH (cmp:Company)-[r:LOOKS_FOR_COMPETENCE]->(s:Skill)
MATCH (company_country:Country)<-[]-(cmp)
MATCH (prs:Person)-[h:HAS]->(s)
MATCH (person_country:Country)<-[]-(prs)
WHERE h.level IN ['Intermediate', 'Expert']
AND person_country.name = company_country.name
RETURN cmp AS Company, count(prs) AS Number;
```

Index impact

The number of country nodes compared by name will be reduced thanks to the index on the country name property.

Advantages and disadvantages of this implementation

The different MATCH layers allow us to filter the different nodes before applying the last query layer with the two comparisons. The index on the name property helps ignoring some data.

This implementation is good because all used properties are indexed correctly, bringing excellent performance. No specific disadvantage appears here.

4.2 Query 2

Description

For the Apple company, get the list of persons who have an affinity with somebody working in the same company, and who have at least one skill of the same level in common with the latter.

Code

```
MATCH (c:Company {name: 'Apple'})-[WORKS_AT]-(e:Employee)-[KNOWS]-(ee:Employee)
MATCH (s:Skill)<-[h:HAS]-(e)
OPTIONAL MATCH (ss:Skill)<-[hh:HAS]-(ee)-[WORKS_AT]-(c)
WHERE h.level = hh.level
RETURN ee AS Employee;
```

Index impact

The index established on the name of the company reduces the number of nodes retrieved by the first MATCH part of the query.

Advantages and disadvantages of this implementation

The index enables to ignore inconsistent data such as people working in other companies. The filtering of nodes at the first line reduces the amount of nodes and as a result the number of comparisons executed. A disadvantage however, is that the level attributes have to be compared as strings for each matching pair of Person nodes. Unnecessary nodes can be matched.

4.3 Query 3

Description

Retrieve the list of companies working in 'Software Engineering' and 'Oil' who employ persons specialised in 'Finance'.

Code

```
MATCH (ss:Skill {name: 'Finance'})-[HAS]-(ee:Person:Employee)-
[WORKS_AT]-(cc:Company)-[BELONGS_TO]-(ii:Industry {name: 'Software Engineering'})
OPTIONAL MATCH (cc)-[BELONGS_TO]-(jj:Industry {name: 'Oil'})
RETURN cc AS Company;
```

Index impact

The created indexes allow to speed up name matching for the Skill, Industry and second Industry nodes in the MATCH commands.

Advantages and disadvantages of this implementation

This implementation is good because all used properties are indexed correctly, bringing excellent performance. No specific disadvantage appears here.

4.4 Query 4

Description

Increase by 200% the salary of all employees experts in 'Neo4j' for 'Software Engineering' companies.

Code

```
MATCH (ss:Skill {name: 'Neo4j'})-[HAS {level: 'Expert'}]-(ee:Person:Employee)-
[ww:WORKS_AT]-(cc:Company)-[BELONGS_TO]-(ii:Industry {name: 'Software Engineering'})
SET ww.salary = 3 * ww.salary;
```

Index impact

The created indexes allow to speed up name matching for the Skill and Industry nodes.

Advantages and disadvantages of this implementation

This implementation uses several indexes, which brings good speed up in performance. However the skill relationship level property is a non indexed string element, therefore there may be a better way to do that part efficiently.

4.5 Query 5

Description

Fire (delete the work relationship) all employees in finance who have a beginner level in all of their skills or no skill at all.

Code

```
MATCH (bb:Skill)-[HAS {level: 'Beginner'}]-(ee:Employee)-
[ww:WORKS_AT]-(cc:Company)-[BELONGS_TO]-(ii:Industry {name: 'Finance'})
OPTIONAL MATCH (ee)-[HAS]-(ss:Skill)
WITH ww, count(bb) AS bbCount, count(ss) AS ssCount
WHERE bbCount = ssCount
DELETE ww;
```

Index impact

The created indexes allow to speed up name matching for the Industry node, which brings some better performance.

Advantages and disadvantages of this implementation

This implementation is still good because it uses an index on the Industry name matching. However the skill relationship level property is a non indexed string element, therefore there may be a better way to do that part efficiently.