



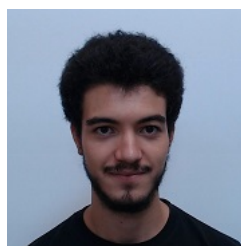
UNIVERSIDADE DO MINHO  
MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA

Projeto Prático

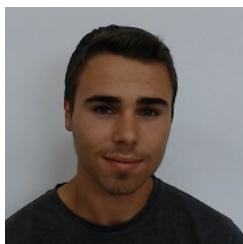
# Bases de dados NoSQL



*Ricardo Cunha - A84302*



*Luis Vila - A84439*



*Luis Ramos - A83930*

Dezembro 2020

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>2</b>
1.1	Enquadramento e Contextualização . . . . .	2
1.2	Problema e Objetivo . . . . .	2
1.3	Estrutura do Relatório . . . . .	2
<b>2</b>	<b>Base de dados Relacional</b>	<b>3</b>
2.1	Modelo de dados . . . . .	3
2.2	Queries . . . . .	3
2.2.1	Query 1 . . . . .	4
2.2.2	Query 2 . . . . .	4
2.2.3	Query 3 . . . . .	5
2.2.4	Query 4 . . . . .	5
2.2.5	Query 5 . . . . .	6
<b>3</b>	<b>Bases de dados Orientadas a Documentos</b>	<b>7</b>
3.1	Modelo de dados . . . . .	7
3.2	Migração de dados . . . . .	8
3.3	Queries . . . . .	9
3.3.1	Query 1 . . . . .	9
3.3.2	Query 2 . . . . .	9
3.3.3	Query 3 . . . . .	10
3.3.4	Query 4 . . . . .	10
3.3.5	Query 5 . . . . .	10
<b>4</b>	<b>Bases de dados Orientadas a Grafos</b>	<b>11</b>
4.1	Modelo de dados . . . . .	11
4.2	Migração de dados . . . . .	12
4.3	Queries . . . . .	14
4.3.1	Query 1 . . . . .	14
4.3.2	Query 2 . . . . .	15
4.3.3	Query 3 . . . . .	15
4.3.4	Query 4 . . . . .	16
4.3.5	Query 5 . . . . .	16
<b>5</b>	<b>Análise Crítica</b>	<b>17</b>
<b>6</b>	<b>Conclusão</b>	<b>18</b>

# 1 Introdução

## 1.1 Enquadramento e Contextualização

Este trabalho surgiu no âmbito da unidade curricular de Base de Dados NoSQL, do 4<sup>o</sup> ano do Mestrado Integrado em Engenharia Informática. Este trabalho aborda principalmente a análise, planeamento e implementação de um SGBD relacional e dois não relacionais.

Para a criação destas, foi necessário recorrer à base de dados relacional **HR**, disponibilizada no *Oracle Database*.

Para a resolução deste trabalho recorreu-se a três tipos de bases de dados diferentes, sendo uma relacional, denominada de **Oracle**, e duas não relacionais, respetivamente, **Neo4J** e **MongoDB**.

## 1.2 Problema e Objetivo

Os principais objetivos deste trabalho prático são logo dados no início do documento fornecido aos alunos. Por nossas palavras, os principais pontos expostos são os seguintes:

- Maior experiência com a utilização de diferentes paradigmas de bases de dados;
- Maior experiência na aplicação, conceção e implementação, dessas bases de dados;
- Pôr em prática os conhecimentos obtidos durante as aulas práticas e teóricas;

O objetivo final passa por demonstrar as diferentes utilizações de bases de dados, assim como as vantagens e desvantagens de cada uma.

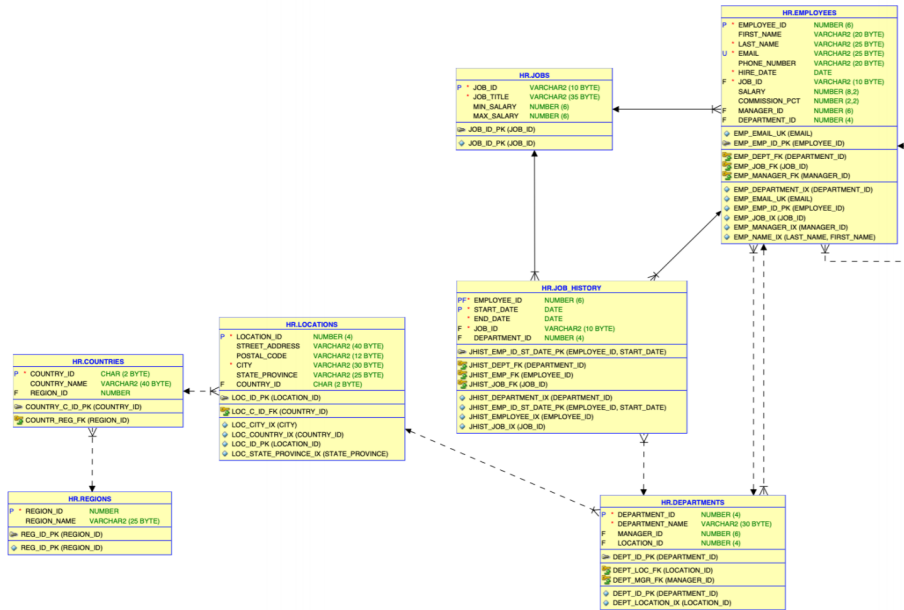
## 1.3 Estrutura do Relatório

Para cada tipo de base de dados distintas, iremos explorar cada um dos tópicos mais aprofundadamente no resto deste relatório:

- **Modelo de dados:** explicamos e demonstramos o modelo de dados da base de dados;
- **Migração de dados:** explicamos em maior detalhe como foi realizada a migração dos dados;
- **Queries:** apresentamos os nossos testes e resultados que realizamos para demonstrar a funcionalidade apropriada a cada base de dados;
- **Conclusão:** avaliação final do nosso trabalho.

## 2 Base de dados Relacional

### 2.1 Modelo de dados



### 2.2 Queries

Nesta secção de queries, optou-se por realizar uma query igual em todos os tipos de bases de dados, como será visível mais à frente neste relatório. Essa query corresponde à query 1.

As restantes queries serão distintas e irão permitir demonstrar a operacionalidade do determinado sistema implementado, neste caso, em Oracle.

De salientar que esta base de dados guarda os dados todos em tabelas, que se encontram relacionadas com outras tabelas, através de chaves estrangeiras e tabelas intermédias.

### 2.2.1 Query 1

Indique os 5 departamentos que ficam mais caros em termos de salários

```
select DEPARTMENT_NAME, sum(SALARY) as soma
from DEPARTMENTS d, EMPLOYEES e
where d.DEPARTMENT_ID = e.DEPARTMENT_ID
group by DEPARTMENT_NAME
order by soma desc
fetch first 5 rows only;
```

	DEPARTMENT_NAME	SOMA
1	Sales	304500
2	Shipping	156400
3	Executive	58000
4	Finance	51608
5	IT	28800

Figura 1: Resposta Query 1

### 2.2.2 Query 2

Indique em que país se encontra o departamento 'Shipping'

```
select COUNTRY_NAME
from COUNTRIES c, DEPARTMENTS d, LOCATIONS l
where d.LOCATION_ID = l.LOCATION_ID
and l.COUNTRY_ID = c.COUNTRY_ID
and d.DEPARTMENT_NAME = 'Shipping';
```

	COUNTRY_NAME
1	United States of America

Figura 2: Resposta Query 2

### 2.2.3 Query 3

Indique quais os empregos que existem em cada departamento

```
select distinct d.DEPARTMENT_NAME, j.JOB_TITLE
from JOBS j
inner join EMPLOYEES e on (j.JOB_ID = e.JOB_ID)
inner join DEPARTMENTS d on (e.DEPARTMENT_ID = d.DEPARTMENT_ID)
order by d.DEPARTMENT_NAME;
```

	DEPARTMENT_NAME	JOB_TITLE
1	Accounting	Accounting Manager
2	Accounting	Public Accountant
3	Administration	Administration Assistant
4	Executive	President
5	Executive	Administration Vice President
6	Finance	Accountant
7	Finance	Finance Manager
8	Human Resources	Human Resources Representative
9	IT	Programmer
10	Marketing	Marketing Manager

Figura 3: Resposta Query 3

### 2.2.4 Query 4

Indique qual é o emprego do empregado com id '101'

```
select JOB_TITLE
from JOBS j, EMPLOYEES e
where j.JOB_ID = e.JOB_ID
and e.EMPLOYEE_ID = 101;
```

	JOB_TITLE
1	Administration Vice President

Figura 4: Resposta Query 4

### 2.2.5 Query 5

Indique qual foi a última pessoa a ser contratada, o seu emprego e salário

```
select e.FIRST_NAME, e.LAST_NAME, j.JOB_TITLE, e.SALARY
from EMPLOYEES e, JOBS j
where j.JOB_ID = e.JOB_ID
order by e.HIRE_DATE desc
fetch first 1 rows only;
```

	FIRST_NAME	LAST_NAME	JOB_TITLE	SALARY
1	Amit	Banda	Sales Representative	6200

Figura 5: Resposta Query 5

## 3 Bases de dados Orientadas a Documentos

### 3.1 Modelo de dados

O modelo de dados de MongoDB foi criado de maneira a eliminar as relações existentes.

Desta maneira este vai ser composto por uma série de documentos aninhados.

#### Department:

- *\_id*: Identificação do departamento
- *Manager\_ID*: Identificação do empregado que dá manage ao departamento.
- *Location*: A localização do departamento.
- *Country\_Name*: Nome do país que se localiza o departamento.
- *Region\_Name*: Nome da Região onde se localiza o departamento.
- *Employees*: Empregados que trabalham no departamento. Array de ***Employee***.

#### Location:

- *Street\_Address*: A rua.
- *Postal\_Code*: O código postal.
- *City*: A cidade.
- *State\_Province*: O estado ou província

#### Employee:

- *Employee\_ID*: A identificação do empregado.
- *First\_Name*: O primeiro nome do empregado.
- *Last\_Name*: O último nome do empregado.
- *Email*: O email do empregado.
- *Phone\_Number*: O número de telemóvel do empregado.
- *Hire\_Date*: A data em que o empregado foi contratado.
- *Salary*: O salário do empregado.
- *Manager\_ID*: A identificação do gerente do empregado.
- *Job*: O trabalho do empregado. Objeto ***Job***.
- *Job\_History*: O historial de trabalho do empregado na empresa. Array de ***Job History***.



### Job History:

- *Start\_Date*: A data de começo do trabalho.
- *End\_Date*: A data de conclusão do trabalho.
- *Department\_ID*: O departamento em que foi realizado o trabalho.
- *Job*: O trabalho em si. Objeto **Job**.

### Job:

- *Job\_ID*: A identificação do trabalho.
- *Job\_Title*: O título do trabalho.
- *Min\_Salary*: O salário mínimo que paga.
- *Max\_Salary*: O salário máximo que paga.

## 3.2 Migração de dados

Tendo o modelo, foi usado o programa Studio3T para a migração dos dados.

Primeiro dando load à tabela de departamentos, foi recriado o *schema* que iria ser usado. No final, o programa Studio3T realiza a migração de acordo com o *schema* introduzido.

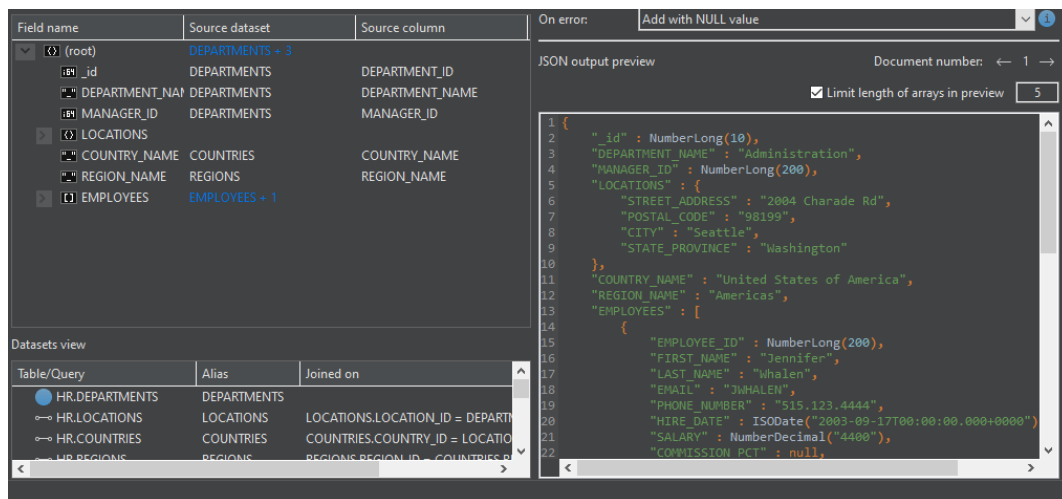


Figura 6: Studio 3T

### 3.3 Queries

Nesta secção de queries, voltou-se a realizar uma query igual às anteriores, sendo esta a query 1.

As restantes queries serão distintas e irão permitir demonstrar a operacionalidade do determinado sistema implementado, neste caso, em MongoDB.

De salientar que esta base de dados é mais adequada a consultas, como será demonstrado abaixo.

#### 3.3.1 Query 1

Indique os 5 departamentos que ficam mais caros em termos salariais.

```
db.DEPARTMENTS.aggregate([{$project:{_id:0, name:"$DEPARTMENT_NAME",
total:{$sum:"$EMPLOYEES.SALARY"} }},{ $sort:{total: -1}}, {$limit: 5}])
```

```
> db.DEPARTMENTS.aggregate([{$project:{_id:0, name:"$DEPARTMENT_NAME",total:{$sum:"$EMPLOYEES.SALARY"} }},{ $sort:{total: -1}}, {$limit: 5}])
{
  "name" : "Sales", "total" : NumberDecimal("304500") }
{
  "name" : "Shipping", "total" : NumberDecimal("156400") }
{
  "name" : "Executive", "total" : NumberDecimal("58000") }
{
  "name" : "Finance", "total" : NumberDecimal("51608") }
{
  "name" : "IT", "total" : NumberDecimal("28800") }
```

Figura 7: Resposta Query 1

#### 3.3.2 Query 2

Indique os departamentos do país "United States of America".

```
db.DEPARTMENTS.find({"COUNTRY_NAME":"United States of America"},
{"_id":0, "DEPARTMENT_NAME":1}).pretty()
```

```
> db.DEPARTMENTS.find({"COUNTRY_NAME":"United States of America"}, {"_id":0, "DEPARTMENT_NAME":1}).pretty()
{ "DEPARTMENT_NAME" : "Administration" }
{ "DEPARTMENT_NAME" : "Purchasing" }
{ "DEPARTMENT_NAME" : "Shipping" }
{ "DEPARTMENT_NAME" : "IT" }
{ "DEPARTMENT_NAME" : "Executive" }
{ "DEPARTMENT_NAME" : "Finance" }
{ "DEPARTMENT_NAME" : "Accounting" }
{ "DEPARTMENT_NAME" : "Treasury" }
{ "DEPARTMENT_NAME" : "Corporate Tax" }
{ "DEPARTMENT_NAME" : "Control And Credit" }
{ "DEPARTMENT_NAME" : "Shareholder Services" }
{ "DEPARTMENT_NAME" : "Benefits" }
{ "DEPARTMENT_NAME" : "Manufacturing" }
{ "DEPARTMENT_NAME" : "Construction" }
{ "DEPARTMENT_NAME" : "Contracting" }
{ "DEPARTMENT_NAME" : "Operations" }
{ "DEPARTMENT_NAME" : "IT Support" }
{ "DEPARTMENT_NAME" : "NOC" }
{ "DEPARTMENT_NAME" : "IT Helpdesk" }
{ "DEPARTMENT_NAME" : "Government Sales" }
```

Figura 8: Resposta Query 2

### 3.3.3 Query 3

Indique os 3 departamentos com mais empregados.

```
db.DEPARTMENTS.aggregate([{$project:{_id:0, name:"$DEPARTMENT_NAME",
total:{$size:"$EMPLOYEES"}}},{ $sort:{total: -1}}, {$limit: 3}])
```

```
> db.DEPARTMENTS.aggregate([{$project:{_id:0, name:"$DEPARTMENT_NAME",total:{$size:"$EMPLOYEES"}}},{ $sort:{total: -1}}, {$limit: 3}])
{ "name" : "Shipping", "total" : 45 }
{ "name" : "Sales", "total" : 34 }
{ "name" : "Purchasing", "total" : 6 }
```

Figura 9: Resposta Query 3

### 3.3.4 Query 4

Indique em que departamento trabalha o empregado com id: 101.

```
db.DEPARTMENTS.find({"EMPLOYEES.EMPLOYEE_ID":110}, {"DEPARTMENT_NAME":1})
.pretty()
```

```
> db.DEPARTMENTS.find({"EMPLOYEES.EMPLOYEE_ID":110}, {"DEPARTMENT_NAME":1}).pretty()
{ "_id" : NumberLong(100), "DEPARTMENT_NAME" : "Finance" }
```

Figura 10: Resposta Query 4

### 3.3.5 Query 5

Indique o número de empregados total.

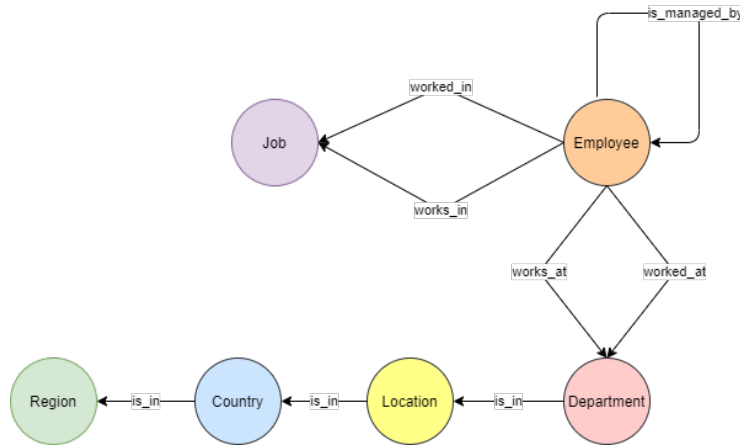
```
db.DEPARTMENTS.aggregate([{$group:{_id:'', "Ammount":{$sum:{$size:"$EMPLOYEES"}}}},
{$project:{_id:0,"total": "$Ammount"}}])
```

```
> db.DEPARTMENTS.aggregate([{$group:{_id:'', "Ammount":{$sum:{$size:"$EMPLOYEES"}}}}, {$project:{_id:0,"total": "$Ammount"}}])
{ "total" : 106 }
```

Figura 11: Resposta Query 5

## 4 Bases de dados Orientadas a Grafos

### 4.1 Modelo de dados



O modelo de dados contruído foi pensado após uma análise do modelo relacional utilizado na base de dados original. Notou-se que as entradas das tabelas podiam ser traduzidas em nodos com relações entre si (as relações percebem-se através da análise das *foreign keys* nas tabelas), à exceção das entradas na tabela *JOB\_HISTORY*, cujo conteúdo será traduzido inteiramente por relações. Assim, um empregado pode relacionar-se ao seu emprego atual ou a empregos que já teve, bem como ao departamento em que trabalha assim como departamentos em que já trabalhou.

Empregados podem também estar relacionados entre si no caso de um deles ser gestor de outro. Para além disso, a relação de um empregado e um departamento consta de uma propriedade que indica se um empregado é gestor do departamento em que trabalha.

Cada departamento está também relacionado com a sua localização, que se encontra relacionado a um país, que por sua vez se encontra relacionado a uma região.

## 4.2 Migração de dados

Para proceder à migração dos dados da base de dados relacional para uma base de dados não relacional de grafos, criamos um *script* em *Python* que se conecta à base de dados relacional e carrega os dados desta para variáveis (uma por cada tabela do modelo relacional).

De seguida, com os dados que já havíamos carregado para as diferentes variáveis, estabelecemos uma conexão com a base de dados *Neo4j* e carregamos para esta os dados lidos da base de dados relacional. Isto resultou assim na criação de vários nodos, sendo que cada nodo corresponderia a uma entrada nas tabelas da base de dados relacional, excetuando da tabela *JOB\_HISTORY*, cujas entradas se traduzirão em relações entre nodos.

O carregamento dos nodos é efetuado com recurso à execução de várias *queries Cypher*, como exemplifica o bloco de código seguinte. Para cada nodo inserido, criamos também atributos que são os mesmos que os atributos das tabelas relacionais, à exceção das *foreign keys*, que não faz sentido incluir numa base de dados de grafos, porque as relações entre os nodos já estarão estabelecidas e serão também guardadas na base de dados.

```
for c in countries:
    cypher = 'CREATE (n:Country {country_id: \'' + str(c[0]) + '\', \
country_name: \'' + str(c[1]) + '\'})'
    session.run(cypher)
print('countries inserted')
```

Seguidamente tratamos de criar as relações entre os nodos existentes, em que interpretamos os relacionamentos expressos pelas *keys* (*primary* e *foreign*) presentes nas tabelas relacionais, a partir das quais conseguimos criar relações entre os nodos já criados, como se exemplifica seguidamente:

```
# Relações entre countries e regions
def relations_coun_reg():
    for c in countries:
        for r in regions:
            if(str(c[2]) == str(r[0])):
                q='MATCH (a:Country),(b:Region) WHERE \
a.country_id = \'' + str(c[0]) + '\ ' \
AND b.region_id = ' + str(r[0]) + ' \
CREATE (a)-[i:is_in]->(b) RETURN type(i)'
                session.run(q)
```

Também utilizamos os dados presentes na tabela *JOB\_HISTORY* para estabelecer relações entre nodos já criados, como se demonstra em seguida:

```
# job history
def relations_jobHist():
    for j in job_history:
        q1='MATCH (a:Employees),(b:Job) WHERE a.employee_id = ' + \
            str(j[0]) + ' AND b.job_id = \'' + str(j[3]) + '\'' CREATE \
            (a)-[w:worked_in {start: \'' + str(j[1]) + '\', finish: \
            \'' + str(j[2]) + '\'}]->(b) RETURN type(w), w.start, \
            w.finish'
        session.run(q1)
        q2='MATCH (a:Employees),(b:Department) WHERE a.employee_id \
            = ' + str(j[0]) + ' AND b.department_id = ' + str(j[4]) + \
            ' CREATE (a)-[w:worked_at {start: \'' + str(j[1]) + '\', \
            finish: \'' + str(j[2]) + '\'}]->(b) RETURN type(w), \
            w.start, w.finish'
        session.run(q2)
```

## 4.3 Queries

Nesta secção de queries, voltou-se a realizar uma query igual às anteriores, sendo esta a query 1.

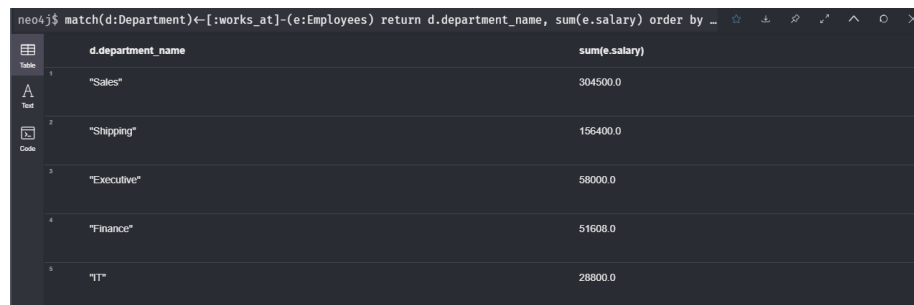
As restantes queries serão distintas e irão permitir demonstrar a operacionalidade do determinado sistema implementado, neste caso, em Neo4J.

De salientar que esta base de dados é mais adequada a pesquisas relacionais, como será demonstrado abaixo.

### 4.3.1 Query 1

Indique os 5 departamentos que ficam mais caros em termos de salários

```
match(d:Department)<-[:works_at]-(e:Employees)
return d.department_name, sum(e.salary)
order by sum(e.salary) desc
limit 5;
```



The screenshot shows the Neo4J interface with a Cypher query executed. The results are displayed in a table view with two columns: 'd.department\_name' and 'sum(e.salary)'. The results are ordered by the sum of salaries in descending order, showing the top 5 departments.

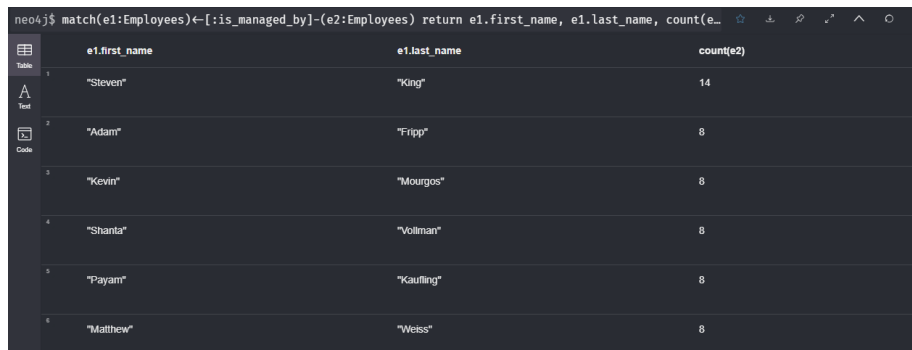
	d.department_name	sum(e.salary)
1	"Sales"	304500.0
2	"Shipping"	156400.0
3	"Executive"	58000.0
4	"Finance"	51608.0
5	"IT"	28800.0

Figura 12: Resposta Query 1

### 4.3.2 Query 2

Indique os 10 administradores que gerem o maior número de empregados

```
match(e1:Employees)<-[:is_managed_by]-(e2:Employees)
return e1.first_name, e1.last_name, count(e2)
order by count(e2) desc
limit 10;
```



	e1.first_name	e1.last_name	count(e2)
1	"Steven"	"King"	14
2	"Adam"	"Fripp"	8
3	"Kevin"	"Mourgos"	8
4	"Shanta"	"Vollman"	8
5	"Payam"	"Kauffing"	8
6	"Matthew"	"Weiss"	8

Figura 13: Resposta Query 2

### 4.3.3 Query 3

Indique qual é o empregado que gere cada departamento

```
match(d:Department)<-[:w:works_at{manages:'yes'}]-(e:Employees)
return d, w, e;
```

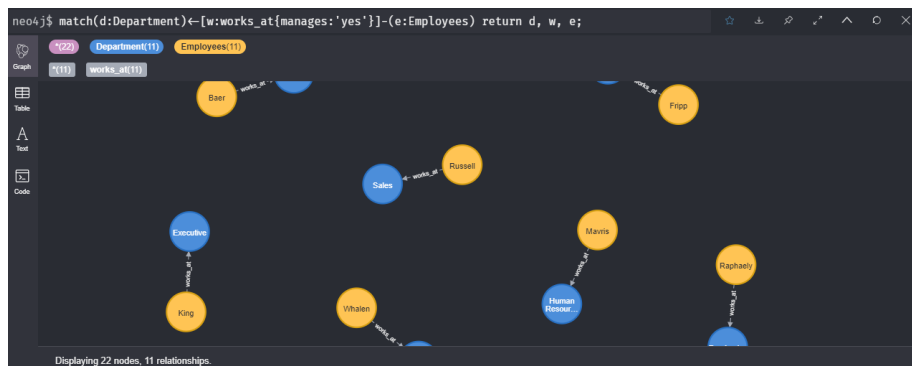


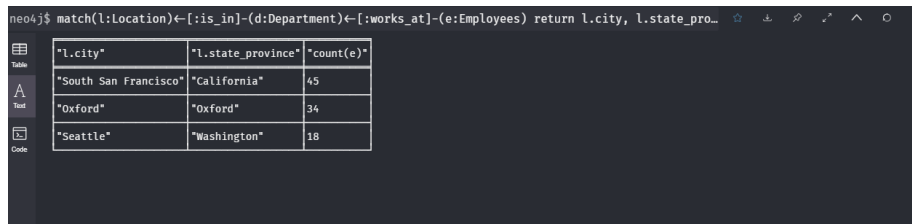
Figura 14: Resposta Query 3



#### 4.3.4 Query 4

Indique qual as 3 localizações com mais empregados.

```
match(l:Location)<-[:is_in]-(d:Department)<-[:works_at]-(e:Employees)
return l.city, l.state_province, count(e)
order by count(e) desc
limit 3;
```



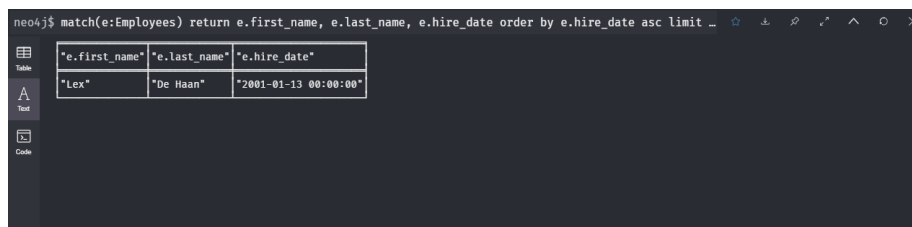
"l.city"	"l.state_province"	"count(e)"
"South San Francisco"	"California"	45
"Oxford"	"Oxford"	34
"Seattle"	"Washington"	18

Figura 15: Resposta Query 4

#### 4.3.5 Query 5

Indique qual o empregado que trabalha há mais tempo na empresa

```
match(e:Employees) return e.first_name, e.last_name, e.hire_date
order by e.hire_date asc limit 1;
```



"e.first_name"	"e.last_name"	"e.hire_date"
"Lex"	"De Haan"	"2001-01-13 00:00:00"

Figura 16: Resposta Query 5

## 5 Análise Crítica

Neste projeto, foi proposto que se migrasse uma base de dados relacional para duas bases de dados NoSQL, sendo uma delas orientada grafos e outra orientada a documentos.

Assim, decidiu-se migrar a base de dados relacional para uma base de dados *Neo4j*, orientada a grafos, e para uma base de dados *MongoDB*, orientada a documentos. A escolha destas baseou-se no conforto do grupo com estas bases de dados, devido a utilização prévia destas ferramentas, e também devido à popularidade destas no mundo empresarial.

Bases de dados NoSQL são adequadas ao armazenamento de grandes quantidades de dados. A base de dados com que se trabalhou é relativamente pequena, no entanto uma previsão futura da sua expansão justifica uma migração para uma base de dados não relacional.

A vantagem da base de dados orientada a grafos é o facto de esta não necessitar de calcular relações que possam existir entre dados presentes na base de dados. Caso um nodo na base de dados se relacione com outro, esta relação ficará guardada, o que representa uma vantagem sobre o modelo relacional, em que os dados são relacionados por chaves associadas a cada entrada na base de dados. Assim, é necessário um maior esforço computacional para calcular relações entre dados presentes em tabelas diferentes.

Já no caso da base de dados orientada a documentos, a sua maior vantagem encontra-se na sua capacidade de encapsular objetos. Assim, no caso do trabalho desenvolvido previamente, foi possível encapsular dados, o que representou uma vantagem devido ao conteúdo presente na base de dados. Nesta estão inseridos departamentos, nos quais se inserem empregados, que por sua vez têm associados a si empregos, etc. Assim, a estruturação permitida pela base de dados orientada a documentos mostrou-se também vantajosa no caso da migração desta base de dados.

## 6 Conclusão

Culminada a elaboração deste trabalho prático, importa referir que a execução do mesmo permitiu aos elementos do grupo compreender o funcionamento e a utilização de cada base de dados.

Assim sendo, foi possível perceber que para se escolher a base de dados ideal, é necessário ter em atenção as características que se quer privilegiar e apurar quais serão as interrogações mais frequentes.

No ímpeto geral o desenvolvimento deste trabalho decorreu como planeado, alcançando os objectivos delineados pelo enunciado.