



**Universidade do Minho**

Mestrado Integrado em Engenharia Informática  
Licenciatura em Ciências da Computação

## **Unidade Curricular de Bases de Dados**

Ano Letivo de 2016/2017

### **Gestão de reservas - InterTrain**

#### **Parte II**

**Ana Fernandes (A74321), Diogo Machado  
(A75399), Miguel Miranda (A74726), Rui  
Leite (A75551)**

Janeiro, 2017

# **BD**

Data de Receção	
Responsável	
Avaliação	
Observações	

## **Gestão de reservas - InterTrain**

### **Parte II**

**Ana Fernandes (A74321), Diogo Machado  
(A75399), Miguel Miranda (A74726), Rui  
Leite (A75551)**

Janeiro, 2017



# Resumo

O projeto realizado consiste no desenvolvimento de uma base de dados que tem assentes os requisitos de funcionamento de uma empresa de reservas de viagens de comboio nacionais e internacionais.

Na primeira parte do projeto foi implementada uma base de dados relacional e nesta segunda parte o objetivo é a transição para uma base de dados não relacional, orientada por grafos.

Numa primeira fase de desenvolvimento é feita a introdução com as motivações da passagem do modelo relacional para o não relacional. De seguida é descrito todo o processo de transição entre um modelo e outro. Depois é explicitada a forma como foram exportados os dados do SGDB anterior para o novo e apresentadas algumas *queries* para demonstrar a viabilidade do modelo orientado por grafos. Por fim é feita uma análise crítica dos resultados.

**Área de Aplicação:** Desenho e arquitetura de Sistemas de Bases de Dados

**Palavras-Chave:** Bases de Dados não Relacionais, Bases de Dados Orientadas por Grafos

# Índice

Resumo	i
1. Introdução	4
1.1. Motivação	4
1.2. Estrutura do Relatório	4
2. Transição do modelo relacional para um modelo orientado por grafos	5
2.1. Representação do modelo orientado por grafos	6
2.2. Regras de transição	6
2.3. Transição de cada tabela para um tipo de nodo	7
2.4. Apresentação do esquema geral	10
3. Exportação e importação dos dados	11
3.1. Exportação dos dados do SGBD MySQL	11
3.2. Importação dos dados para o novo SGBD	12
4. Implementação de algumas queries	13
5. Análise crítica	17
6. Conclusão	18
Referências	19
Lista de Siglas e Acrónimos	20

# Índice de Figuras

Figura 1 - Modelo lógico do esquema relacional	5
Figura 2 - Representação de um modelo orientado por grafos	6
Figura 3 – Transição da tabela Cliente para o tipo de nodo Cliente	7
Figura 4 – Transição da tabela Reserva para o tipo de nodo Reserva	7
Figura 5 – Transição da tabela Viagem para o tipo de nodo Viagem	8
Figura 6 – Transição da tabela Comboio para o tipo de nodo Comboio	8
Figura 7 – Transição da tabela Lugar para o tipo de nodo Lugar	8
Figura 8 – Esquema geral da base de dados orientada por grafos	10
Figura 9 – Resultado da execução da query <code>SELECT * FROM cliente</code>	11
Figura 10 – Resultado do uso da opção Export do MySQL Workbench	11
Figura 11 – Parte do resultado da query 4 em SQL	13
Figura 12 – Parte do resultado da query 4 em CQL	14
Figura 13 – Parte do resultado da query 8 em SQL	15
Figura 14 – Parte do resultado da query 8 em CQL	15
Figura 15 – Resultado da query 13 em SQL	16
Figura 16 – Resultado da query 13 em CQL	16

# 1. Introdução

Esta segunda parte do projeto consiste na implementação de uma base de dados não relacional para a gestão de reservas de bilhetes de viagens na companhia de comboios *InterTrain*.

Na primeira parte foram já apresentadas a contextualização do problema e a apresentação do caso de estudo, bem como os requisitos, a modos que neste capítulo faz-se apenas uma explicitação das motivações para a transição da base de dados relacional para uma base de dados orientada por grafos e a apresentação da estrutura do presente relatório.

## 1.1. Motivação

Apesar das vantagens existentes no modelo relacional, como o seu rigor na consistência dos dados, pode ser vantajoso analisar o desempenho num paradigma não relacional. Por vezes, a arquitetura relacional pode não ser a mais indicada

Motivações para usar bases de dados não relacionais passam essencialmente pela sua escalabilidade (quanto à quantidade de dados ou à quantidade de utilizadores simultaneamente a operar sobre ela), pela flexibilidade (com estruturas de dados flexíveis e possivelmente não definidas, quebrando a dependência a um esquema de inter-relação de tabelas) e também pela boa *performance* que garante.

Nesta segunda parte do trabalho é analisado o comportamento da base de dados numa arquitetura orientada por grafos.

## 1.2. Estrutura do Relatório

O presente relatório encontra-se dividido em seis capítulos: o primeiro, a introdução, já apresentada; o segundo, referente à transição entre os modelos; o terceiro, referente à exportação e importação dos dados entre um SGDB e outro; o quarto, referente à implementação de algumas *queries*; o quinto, onde é feita uma análise crítica dos resultados; o sexto e último, onde são feitas as conclusões obtidas com a elaboração do projeto.

## 2. Transição do modelo relacional para um modelo orientado por grafos

Neste capítulo é descrito o processo de transição entre o esquema construído na primeira parte do trabalho prático para um modelo não relacional orientado por grafos. No final é apresentado um grafo representativo do novo “esquema”. Como base para a transição tome-se o esquema lógico do modelo relacional:

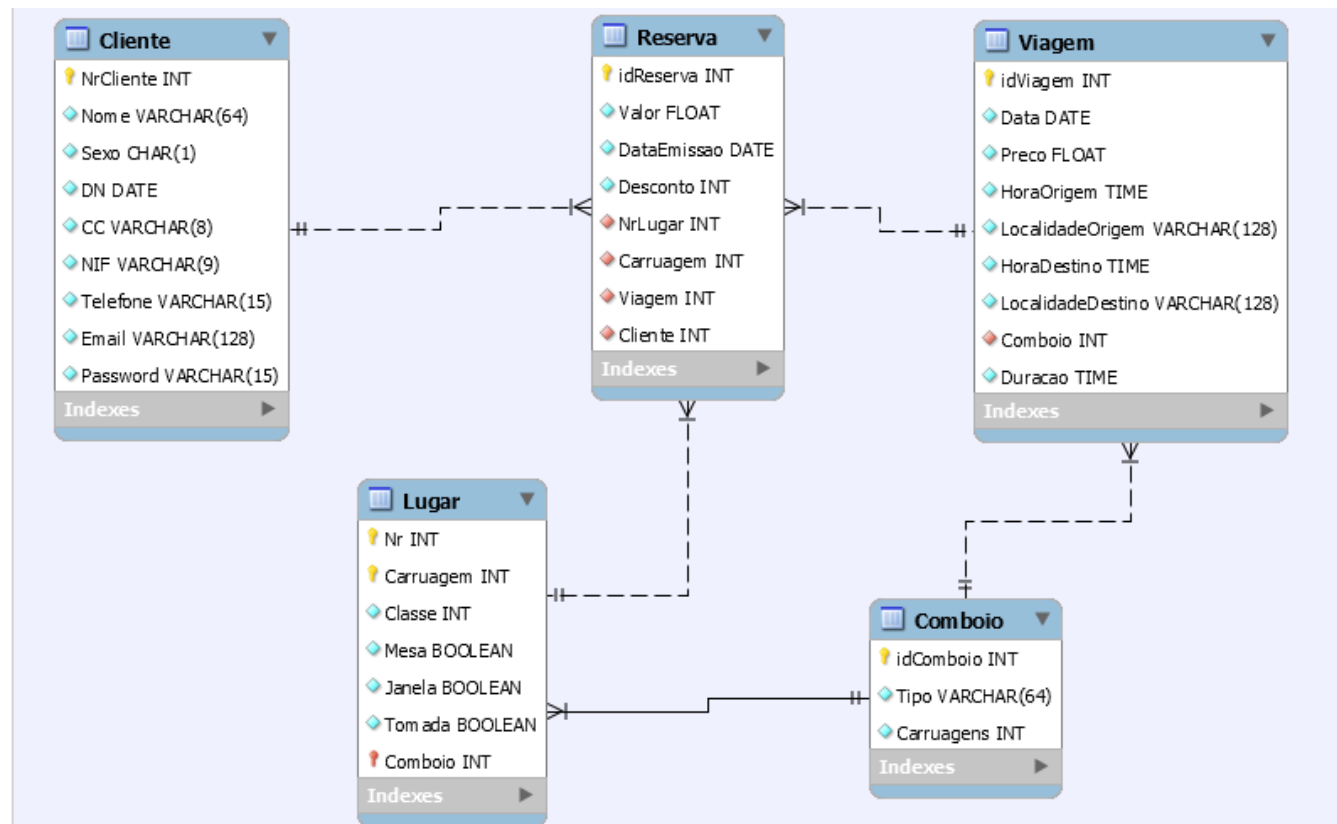


Figura 1 - Modelo lógico do esquema relacional



## 2.1. Representação do modelo orientado por grafos

Como forma de simplificar a explicitação do processo de transição, neste relatório será usada a representação da Figura 2 para o modelo orientado por grafos.

Cada nodo é relativo a um tipo de nodos (*Label name*) e possui as suas propriedades (*Properties*).

Dois nodos podem ser relacionados (*Relation*), relação essa que é definida por um nome.

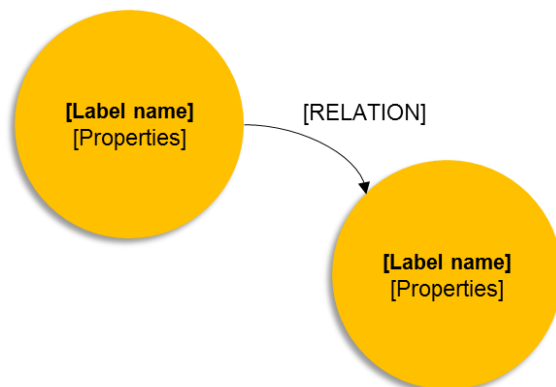


Figura 2 - Representação de um modelo orientado por grafos

## 2.2. Regras de transição

A transição desde o modelo relacional, orientado por tabelas, desenvolvido na primeira parte do trabalho prático para o modelo não relacional orientado por grafos fez-se tendo como base as seguintes regras:

- O nome de uma tabela dá origem a um tipo de nodo;
- Os atributos (colunas de uma tabela) dão origem às propriedades;
- Cada linha de uma tabela origina um nodo do tipo correspondente;
- As relações entre tabelas (pares *primary key/foreign key*) são substituídas por relações entre nodos.

## 2.3. Transição de cada tabela para um tipo de nodo

Nesta secção é apresentada como foi feita a criação de cada tipo de nodo a partir das tabelas do modelo lógico.

### Tabela Cliente → Tipo de nodo Cliente

A tabela Cliente deu origem a um tipo de nodo também ele chamado Cliente. Todos os atributos de Cliente são agora considerados propriedades do nodo Cliente.

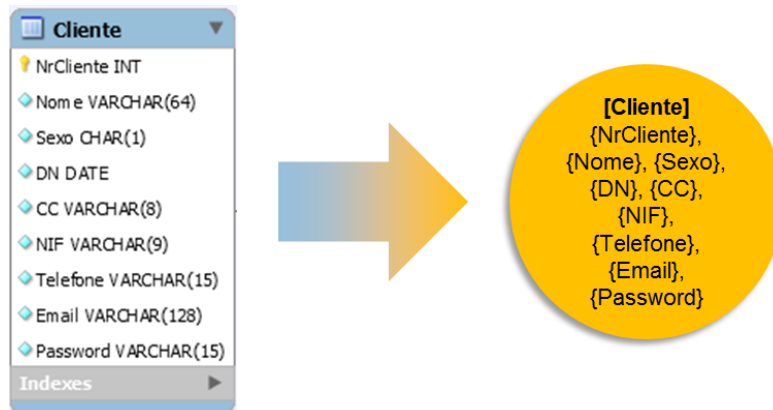


Figura 3 – Transição da tabela Cliente para o tipo de nodo Cliente

### Tabela Reserva → Tipo de nodo Reserva

A tabela Reserva originou um novo tipo de nodo chamado Reserva. Os atributos de Reserva são agora considerados propriedades do nodo Reserva, à exceção das chaves estrangeiras ("NrLugar", "Carruagem", "Viagem" e "Cliente") que passam a ser representadas por relações entre nodos.

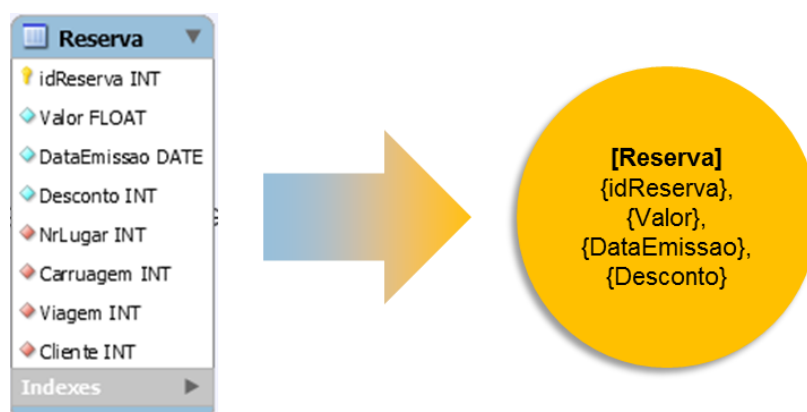


Figura 4 – Transição da tabela Reserva para o tipo de nodo Reserva

### Tabela Viagem → Tipo de nodo Viagem

A tabela Viagem originou um novo tipo de nodo chamado Viagem. Os atributos de Viagem são agora considerados propriedades do nodo Viagem, à exceção da chave estrangeira “Comboio” que passa a ser representada por relações entre nodos.

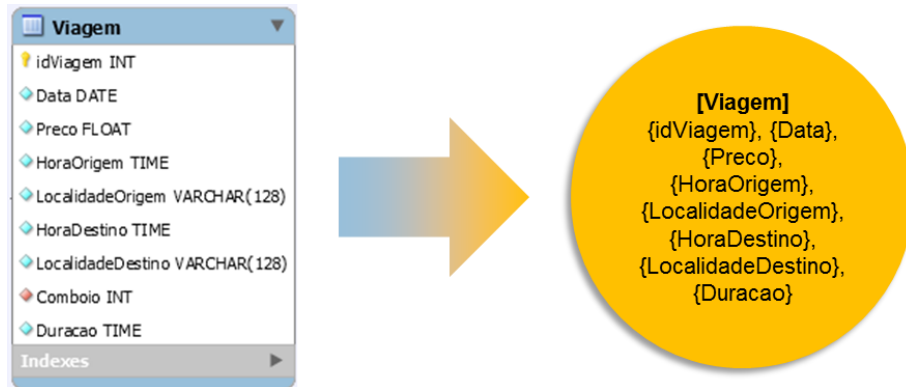


Figura 5 – Transição da tabela Viagem para o tipo de nodo Viagem

### Tabela Comboio → Tipo de nodo Comboio

A tabela Comboio originou um novo tipo de nodo chamado Comboio. Os atributos do Comboio são agora considerados propriedades do nodo Comboio.

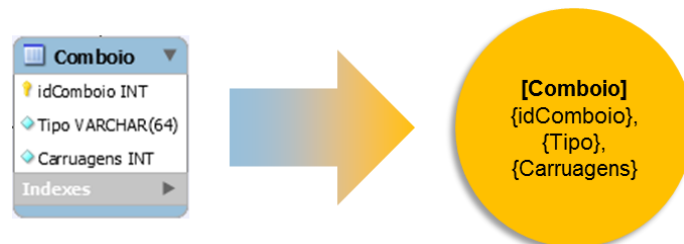


Figura 6 – Transição da tabela Comboio para o tipo de nodo Comboio

### Tabela Lugar → Tipo de nodo Lugar

A tabela Lugar originou um novo tipo de nodo chamado Lugar. Os atributos do Lugar são agora considerados propriedades do nodo Lugar.

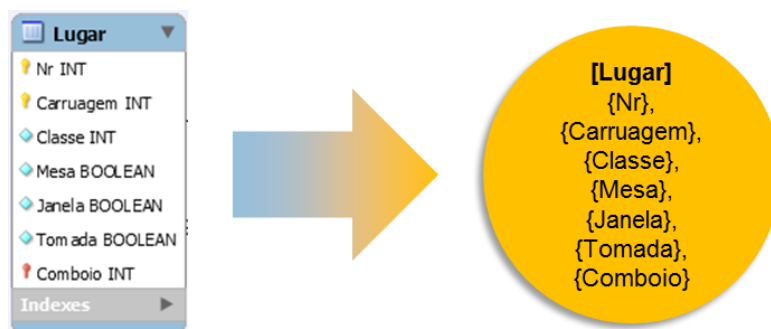


Figura 7 – Transição da tabela Lugar para o tipo de nodo Lugar

No que diz respeito à conversão dos tipos de dados entre o *MySQL* e o *Neo4J* foram feitas as seguintes considerações:

- Os tipos de dados `VARCHAR()` passaram a `Strings`;
- Os tipos de dados `INT` e `FLOAT` continuaram como `INT` e `FLOAT`, respetivamente;
- As datas em *MySQL*, dada a incompatibilidade com o *Neo4J*, foram convertidas para `INT`, através da função `unix_timestamp()` do *MySQL*, ficando do lado da aplicação a responsabilidade de fazer a conversão para o formato `YYYY-MM-DD`.

## 2.4. Apresentação do esquema geral

Pelas regras de transição já apresentadas, foram considerados os relacionamentos provenientes dos pares *primary key/foreign key* das tabelas do esquema lógico (Figura 1). Assim, tem-se o seguinte esquema geral da Base de Dados Orientada por Grafos.

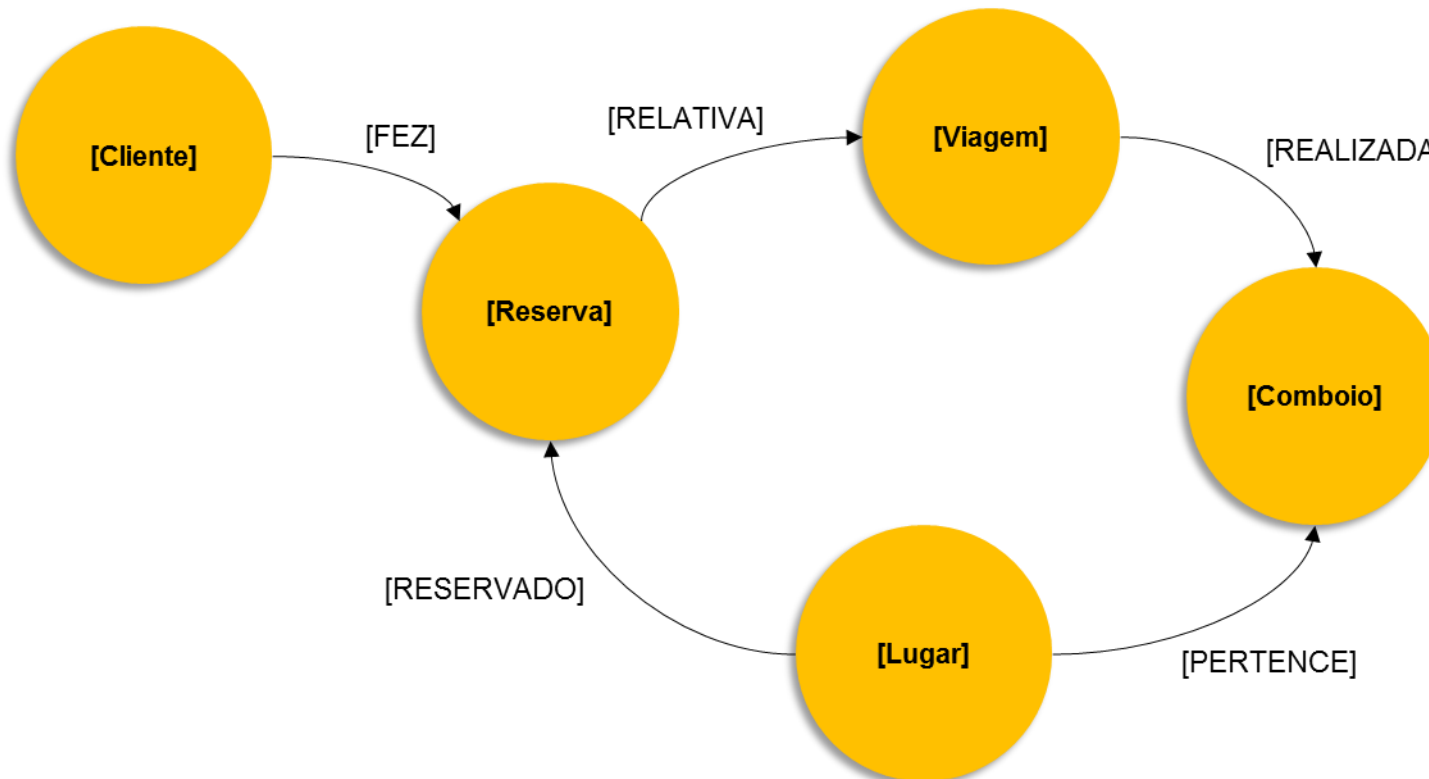



Figura 8 – Esquema geral da base de dados orientada por grafos

### 3. Exportação e importação dos dados

Neste capítulo é apresentada a forma como foram exportados os dados da base de dados relacional implementada no SGBD *MySQL* e como foram importados para o SGBD *Neo4J*.

#### 3.1. Exportação dos dados do SGBD *MySQL*

Para a exportação dos dados a partir do SGBD implementado na primeira parte do trabalho prático em *MySQL*, foi utilizada uma ferramenta do *MySQL WorkBench* através do seguinte procedimento:

- Para cada uma das tabelas executar uma *query* de seleção de todos os dados;
- Utilizar a ferramenta do “Export recordset to an external file”:  e guardar o ficheiro no formato CSV.

##### Exemplo

```
SELECT * FROM cliente;
```

Resultado:







Result Grid		Filter Rows: <input type="text"/>		Edit:   		Export/Import:  		Wrap Cell Content: 	
	NrCliente	Nome	Sexo	DN	CC	NIF	Telefone	Email	Password
▶	1	Lúgia Napoleão Barrocas	F	1979-09-03	84540663	902426160	904400412	isabellehc_spears@outlook.com	uATuu56gsStrN44
	2	Eliseu Celeste Castelo Branco	M	1975-08-05	32431999	926913817	996924604	lsah_57@hotmail.com	3db3Wlysm0vY3Zx
	3	Morgana Cássia Figueira	F	1994-05-14	40399458	799959330	928886272	concepcion.skt@aol.com	Cjvy27dAudBpvfC
	4	Margarida Neuza Marins	F	1976-02-18	64774557	786597535	904170913	justine_mb_webb.8@outlook.com	vsvzciESEMvLINb
	5	Laurinda Hermígio Ramos	F	1983-10-10	11149975	912696983	994444394	alice.kc_oneill@yahoo.com	ncPBFzbbo14K5rK
	6	Bráulio Almerinda Toscano	M	1991-07-27	98601893	270240829	951866420	myrnabm.bryan_82@gmx.com	QvYRSO86RrJEk8z
	7	Norberto Gláuber Graça	M	1982-04-27	81717392	257338047	958345592	lora_54@gmail.com	XOIZcH6job6hFU4
	8	Evandro Bartolomeu Freiria	M	1969-04-08	42374448	218438287	979537118	lessie_bhp95@gmx.com	ZCwP7m5btGTJ5s
	9	Alzira Priscila Bulhões	F	2004-08-20	48458901	105771613	971320505	sybilwar.42@gmx.com	Ns2F2WYH2joiehz
	10	Nádia Carlota Vilarinho	F	1963-09-14	15247220	360560802	973864829	ilavmg_60@gmx.com	5XKCVsu037CwRf

Figura 9 – Resultado da execução da *query* `SELECT * FROM cliente`

Ao escolher a opção *Export* ():

Export Resultset

← → ↕

« Parte 2 - Base de dado... » Povoamento

Procurar em Povoamento

Nome de ficheiro:

Cliente.csv

Guardar com o tipo:

CSV

Procurar em pastas

Guardar

Cancelar

Figura 10 – Resultado do uso da opção *Export* do *MySQL Workbench*

## 3.2. Importação dos dados para o novo SGBD

A importação dos dados para o SGBD *Neo4J* foi feita através da criação de uma *query* em CQL, para cada tipo de nodo que “percorre” todo o ficheiro em CSV e que cria os nodos, com o tipo correto, mediante a informação de cada linha do ficheiro (corresponde a uma linha na tabela respeitante).

### Exemplo

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:/cliente.csv" AS row
CREATE (:Cliente {NrCliente: toInt(row.NrCliente),
                  Nome: row.Nome,
                  Sexo: row.Sexo,
                  DN: row.DN,
                  CC: row.CC,
                  NIF: row.NIF,
                  Telefone: row.Telefone,
                  Email: row.Email,
                  Password: row.Password});
```

Tendo sido todos os nodos, e todas as propriedades de cada um, inseridos na base de dados, é necessário relacioná-los conforme o esquema da Figura 8. Para tal, voltou-se a percorrer cada um dos ficheiros CSV necessários e implementou-se uma *query* que construa a relação entre cada dois nodos.

### Exemplo

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:/viagem.csv" AS row
MATCH (viagem:Viagem {idViagem: toInt(row.idViagem)})
MATCH (comboio:Comboio {idComboio: toInt(row.Comboio)})
MERGE (viagem)-[r:REALIZADA]->(comboio);
```

## 4. Implementação de algumas *queries*

Neste capítulo são apresentadas algumas das queries que foram definidas na primeira parte do trabalho prático e que agora foram novamente implementadas na linguagem de interrogação *Cypher*.

**4. Para efeitos de faturação, a empresa deverá saber o número de contribuinte e número do CC de cada um dos seus clientes, a data em que cada reserva foi emitida e o respetivo preço.**

EM SQL:

```
SELECT C.NrCliente, C.NIF, C.CC, R.DataEmissao AS 'Data de emissão da  
reserva', R.Valor  
FROM cliente  
INNER JOIN reserva R ON R.Cliente = C.NrCliente  
ORDER BY C.NrCliente,R.DataEmissao;
```

	NrCliente	NIF	CC	Data de emissão da reserva (ms)	Valor
▶	1	902426160	84540663	1456358400000	171.457
	1	902426160	84540663	1479772800000	18.2
	1	902426160	84540663	1493852400000	20
	1	902426160	84540663	1493852400000	63.15
	1	902426160	84540663	1501542000000	59.61
	1	902426160	84540663	1506553200000	43.815
	1	902426160	84540663	1512864000000	161.236
	2	926913817	32431999	1455580800000	171.457
	2	926913817	32431999	1479513600000	47.32
	2	926913817	32431999	1494543600000	164.19
	2	926913817	32431999	1501110000000	79.48
	2	926913817	32431999	1512259200000	165.37
	3	799959330	40399458	1494457200000	126.3
	3	799959330	40399458	1501974000000	103.324
	3	799959330	40399458	1512172800000	124.027
	4	786597535	64774557	1455580800000	131.89
	4	786597535	64774557	1480118400000	36.4
	4	786597535	64774557	1501714800000	39.74

Figura 11 – Parte do resultado da *query* 4 em SQL



Em CQL:

```
MATCH (c:Cliente) -[:FEZ]-> (r:Reserva)
RETURN c.NrCliente, c.NIF, c.CC, r.DataEmissao AS `Data de emissão da
Reserva`, r.Valor
ORDER BY c.NrCliente, r.DataEmissao;
```

c.NrCliente	c.NIF	c.CC	Data de emissão da Reserva	r.Valor
1	902426160	84540663	1456358400000	171.457
1	902426160	84540663	1479772800000	18.2
1	902426160	84540663	1493852400000	63.15
1	902426160	84540663	1493852400000	20
1	902426160	84540663	1501542000000	59.61
1	902426160	84540663	1506553200000	43.815
1	902426160	84540663	1512864000000	161.236
2	<a href="#">926913817</a>	32431999	1455580800000	171.457
2	<a href="#">926913817</a>	32431999	1479513600000	47.32
2	<a href="#">926913817</a>	32431999	1494543600000	164.19
2	<a href="#">926913817</a>	32431999	1501110000000	79.48
2	<a href="#">926913817</a>	32431999	1512259200000	165.37
3	799959330	40399458	1494457200000	126.3
3	799959330	40399458	1501974000000	103.324
3	799959330	40399458	1512172800000	124.027
4	786597535	64774557	1455580800000	131.89

Figura 12 – Parte do resultado da *query* 4 em CQL

**8. O preço de uma reserva é dado pelo preço base que todas as viagens têm associado, eventualmente com a possibilidade de um desconto.**

EM SQL:

```
SELECT R.idReserva, R.Cliente, R.Valor, V.Preco, R.Desconto, L.Classe,
R.Viagem
FROM reserva R
INNER JOIN viagem V ON R.Viagem = V.idViagem
INNER JOIN lugar L ON (R.NrLugar = L.Nr AND R.Carruagem = L.Carruagem
AND L.Comboio = V.Comboio)
ORDER BY R.idReserva;
```

	idReserva	Nome	Valor	Preco	Desconto	Classe	Viagem
▶	1	Norberto Gláuber Graça	18.2	36.4	50	2	1
	2	Margarida Neuza Marins	36.4	36.4	0	2	1
	3	Ligia Napoleão Barrocas	18.2	36.4	50	2	1
	4	Nádia Carlota Vilarinho	36.4	36.4	0	2	1
	5	Bráulio Almerinda Toscano	36.4	36.4	0	2	1
	6	Evandro Bartolomeu Freiria	27.3	36.4	25	2	1
	7	Laurinda Hermígio Ramos	47.32	36.4	0	1	1
	8	Alzira Priscila Bulhões	27.3	36.4	25	2	1
	9	Eliseu Celeste Castelo Branco	47.32	36.4	0	1	1
	10	Alzira Priscila Bulhões	79.48	79.48	0	2	2
	11	Bráulio Almerinda Toscano	79.48	79.48	0	2	2
	12	Evandro Bartolomeu Freiria	77.493	79.48	25	1	2
	13	Laurinda Hermígio Ramos	103.324	79.48	0	1	2
	14	Norberto Gláuber Graça	39.74	79.48	50	2	2
	15	Nádia Carlota Vilarinho	79.48	79.48	0	2	2

Figura 13 – Parte do resultado da *query* 8 em SQL

Em CQL:

```

MATCH (l:Lugar) -[:RESERVADO]-> (r:Reserva) -[:RELATIVA]-> (v:Viagem)
MATCH (r:Reserva) <-[:FEZ]- (c:Cliente)
RETURN r.idReserva, c.Nome, r.Valor, v.Preco, r.Desconto, l.Classe,
v.idViagem
ORDER BY (r.idReserva) ASC;

```

Reserva	Nome	Valor	Preço	Desconto	Classe	Viagem
1	Norberto Gláuber Graça	18.2	36.4	50	2	1
2	Margarida Neuza Marins	36.4	36.4	0	2	1
3	Ligia Napoleão Barrocas	18.2	36.4	50	2	1
4	Nádia Carlota Vilarinho	36.4	36.4	0	2	1
5	Bráulio Almerinda Toscano	36.4	36.4	0	2	1
6	Evandro Bartolomeu Freiria	27.3	36.4	25	2	1
7	Laurinda Hermígio Ramos	47.32	36.4	0	1	1
8	Alzira Priscila Bulhões	27.3	36.4	25	2	1
9	Eliseu Celeste Castelo Branco	47.32	36.4	0	1	1
10	Alzira Priscila Bulhões	79.48	79.48	0	2	2
11	Bráulio Almerinda Toscano	79.48	79.48	0	2	2
12	Evandro Bartolomeu Freiria	77.493	79.48	25	1	2
13	Laurinda Hermígio Ramos	103.324	79.48	0	1	2
14	Norberto Gláuber Graça	39.74	79.48	50	2	2
15	Nádia Carlota Vilarinho	79.48	79.48	0	2	2

Figura 14 – Parte do resultado da *query* 8 em CQL

**13. O cliente deve ter a capacidade de poder escolher o lugar que mais lhe agrada para uma determinada viagem. Para tal deve ter acesso à lista de lugares que ainda não foram reservados numa dada viagem e escolher um desses lugares. O mais importante para os clientes na escolha do lugar é a classe em que vão viajar assim como saber se esse lugar se encontra à janela, se tem tomada elétrica disponível e se tem mesa.**

EM SQL (para a viagem 1):

```
SELECT L.Nr, L.Carruagem, L.Classe, IF (L.Janela,'Sim','Não') AS
'Janela', IF(L.Tomada,'Sim','Não') AS 'Tomada', IF(L.Mesa,'Sim','Não')
AS 'Mesa'FROM lugar L
INNER JOIN comboio C ON L.Comboio = C.idComboio
INNER JOIN viagem V ON C.idComboio = V.Comboio
WHERE V.idViagem = 1 AND
(L.Nr, L.Carruagem) NOT IN (
SELECT R.NrLugar, R.Carruagem
FROM reserva R
INNER JOIN viagem V ON R.Viagem = V.idViagem
WHERE V.idViagem = 1);
```

	Nr	Carruagem	Classe	Janela	Tomada	Mesa
►	1	1	2	1	1	1
	1	2	2	1	0	0
	2	1	2	0	1	0
	4	1	1	1	0	0
	4	2	2	1	0	1
	5	1	2	1	1	0

Figura 15 – Resultado da *query* 13 em SQL

Em CQL (para a viagem 1):

```
MATCH (l:Lugar) -[:RESERVADO]-> (r:Reserva) -[:RELATIVA]-> (v:Viagem
{idViagem: 1})
WITH collect(l) AS lugaresReservados
MATCH (l:Lugar) -[:PERTENCE]-> (c:Comboio) <-[:REALIZADA]- (v:Viagem
{idViagem: 1})
WHERE NOT l IN lugaresReservados
RETURN l.Nr, l.Carruagem, l.Classe, l.Janela, l.Tomada, l.Mesa
```

I.Nr	I.Carruagem	I.Classe	I.Janela	I.Tomada	I.Mesa
2	1	2	0	1	0
1	2	2	1	0	0
5	1	2	1	1	0
1	1	2	1	1	1
4	2	2	1	0	1
4	1	1	1	0	0

Figura 16 – Resultado da *query* 13 em CQL

## 5. Análise crítica

Após a realização deste trabalho, conseguimos destacar algumas vantagens e desvantagens face aos modelos SQL e *NoSQL* utilizados.

Perante o modelo não-relacional, destacamos o facto de a linguagem de interrogação utilizada no *Neo4J* ser simples de aprender e intuitiva no modo como se enunciam as relações entre os dados.

Na linguagem SQL, reconhecemos a importância da organização e consistência dos dados. Na arquitetura orientada por grafos usada, a consistência é, de certa forma, implícita pelas relações que se fazem, mas o cuidado na inserção deve ser maior para não injetar erros nos dados. Também é possível fazer uma comparação com o modelo relacional na medida em que no SGBD *Neo4J* é possível definir os tipos de nodos, relacionados com as tabelas em *MySQL*.

O ponto negativo no SGBD não relacional prende-se com a falta de várias funções essenciais para o tratamento de determinados dados, como datas, o que dificultou a transição do *MySQL*.

Pela a dimensão da base de dados usada, qualquer uma das implementações se mostrou bastante rápida, não havendo diferenças significativas.

## 6. Conclusão

Numa análise final e global do trabalho desenvolvido, fazemos um balanço positivo do projeto, tendo a sua elaboração seguido todos os passos da metodologia e cumprindo todos os requisitos.

Destacamos a exportação dos dados da base de dados relacional para a *Neo4J* como sendo a tarefa de mais fácil realização. Como dificuldades sentidas na realização deste projeto, frisamos a “conversão” de algumas das *queries* mais substanciais de SQL para a linguagem *Cypher*.

Devido ao profundo conhecimento do contexto e dos requisitos da base de dados e pela definição do modelo conceptual e lógico, pela realização da primeira fase do projeto, a realização desta segunda implementação foi, no geral, mais simples e fluida.

Em relação ao sistema *Neo4J* e à linguagem *Cypher* utilizada no SGBD, compreendemos e adquirimos os conhecimentos essenciais para a sua utilização, gestão e manipulação de dados.

## Referências

Neo4j.com. (2017). *Tutorials - The Neo4j Manual v2.3.3*. [online] Available at: <http://neo4j.com/docs/stable/tutorials.html> [Accessed 17 Jan. 2017].

Neo4j Graph Database. (2017). *For Relational Database Developers: A SQL to Cypher Guide*. [online] Available at: <https://neo4j.com/developer/guide-sql-to-cypher/> [Accessed 17 Jan. 2017].

# Lista de Siglas e Acrónimos

<b>BD</b>	Base de Dados
<b>SGBD</b>	Sistema de Gestão de Bases de Dados
<b>CQL</b>	<i>Cypher Query Language</i>
<b>SQL</b>	<i>Structured Query Language</i>