

**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](#_Toc472877676)

[1. Introdução 4](#_Toc472877677)

[1.1. Motivação 4](#_Toc472877678)

[1.2. Estrutura do Relatório 4](#_Toc472877679)

[2. Transição do modelo relacional para um modelo orientado por grafos 5](#_Toc472877680)

[2.1. Representação do modelo orientado por grafos 6](#_Toc472877681)

[2.2. Regras de transição 6](#_Toc472877682)

[2.3. Transição de cada tabela para um tipo de nodo 7](#_Toc472877683)

[2.4. Apresentação do esquema geral 10](#_Toc472877684)

[3. Exportação e importação dos dados 11](#_Toc472877685)

[3.1. Exportação dos dados do SGBD MySQL 11](#_Toc472877686)

[3.2. Importação dos dados para o novo SGBD 12](#_Toc472877687)

[4. Implementação de algumas queries 13](#_Toc472877688)

[5. Análise crítica 17](#_Toc472877689)

[6. Conclusão 18](#_Toc472877690)

[Referências 19](#_Toc472877691)

[Lista de Siglas e Acrónimos 20](#_Toc472877692)

Índice de Figuras

[Figura 1 - Modelo lógico do esquema relacional 5](https://d.docs.live.net/1d82eb930983b5c3/Documentos/MiEI/3.º%20ano/1.º%20semestre/Bases%20de%20Dados/Trabalho%20prático/Parte%202%20-%20Base%20de%20dados%20orientada%20por%20grafos%20(NEO4J)/Relatório.docx#_Toc472877693)

[Figura 2 - Representação de um modelo orientado por grafos 6](https://d.docs.live.net/1d82eb930983b5c3/Documentos/MiEI/3.º%20ano/1.º%20semestre/Bases%20de%20Dados/Trabalho%20prático/Parte%202%20-%20Base%20de%20dados%20orientada%20por%20grafos%20(NEO4J)/Relatório.docx#_Toc472877694)

[Figura 3 – Transição da tabela Cliente para o tipo de nodo Cliente 7](#_Toc472877695)

[Figura 4 – Transição da tabela Reserva para o tipo de nodo Reserva 7](#_Toc472877696)

[Figura 5 – Transição da tabela Viagem para o tipo de nodo Viagem 8](#_Toc472877697)

[Figura 6 – Transição da tabela Comboio para o tipo de nodo Comboio 8](#_Toc472877698)

[Figura 7 – Transição da tabela Lugar para o tipo de nodo Lugar 8](#_Toc472877699)

[Figura 8 – Esquema geral da base de dados orientada por grafos 10](#_Toc472877700)

[Figura 9 – Resultado da execução da query SELECT \* FROM cliente 11](#_Toc472877701)

[Figura 10 – Resultado do uso da opção Export do MySQL Workbench 11](#_Toc472877702)

[Figura 11 – Parte do resultado da query 4 em SQL 13](#_Toc472877703)

[Figura 12 – Parte do resultado da query 4 em CQL 14](#_Toc472877704)

[Figura 13 – Parte do resultado da query 8 em SQL 15](#_Toc472877705)

[Figura 14 – Parte do resultado da query 8 em CQL 15](#_Toc472877706)

[Figura 15 – Resultado da query 13 em SQL 16](#_Toc472877707)

[Figura 16 – Resultado da query 13 em CQL 16](#_Toc472877708)

# 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.

## 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.

## 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.

# 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:

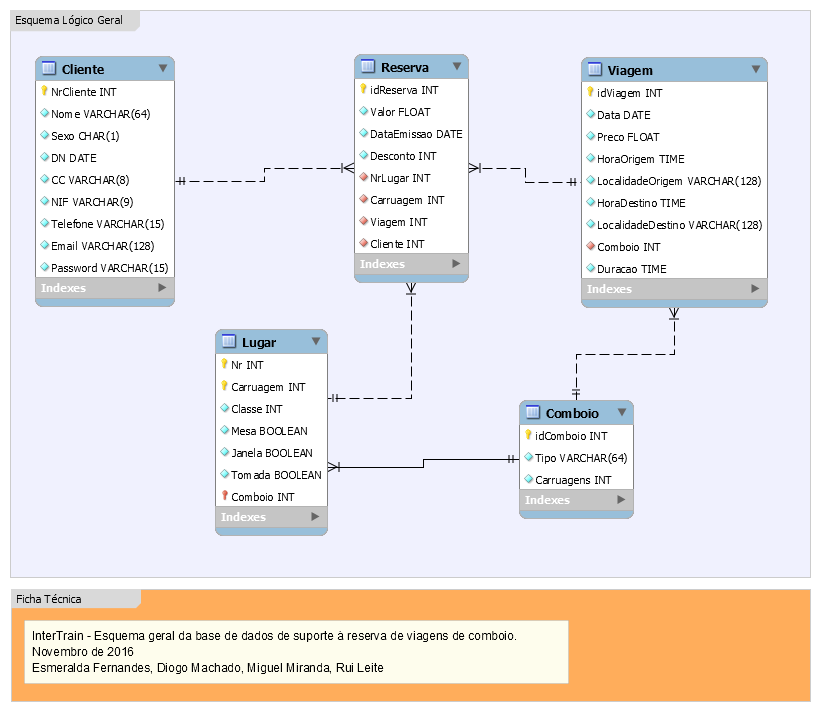


Figura 1 - Modelo lógico do esquema relacional

## 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.

Figura 2 - Representação de um 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.

## 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.

## 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.

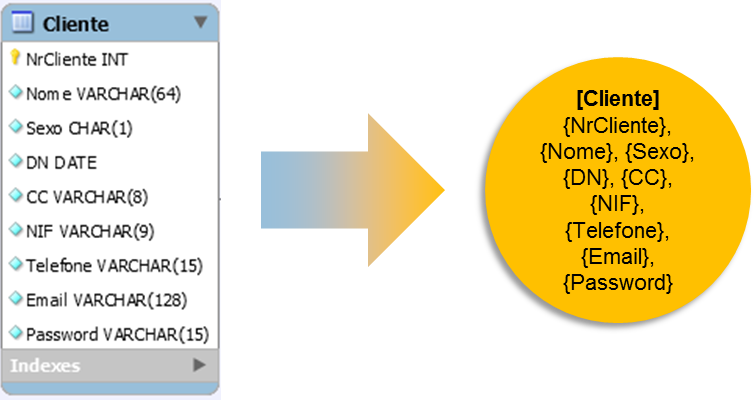
****

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.

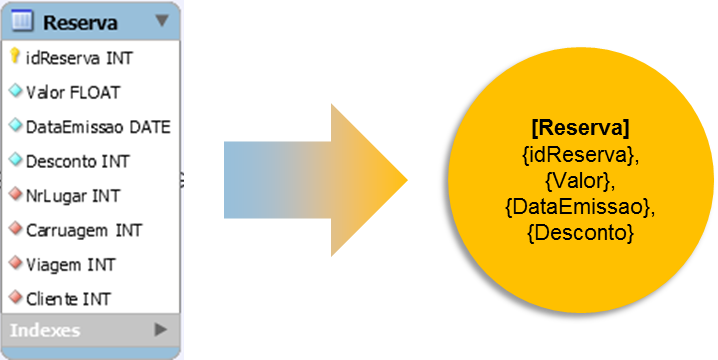
****

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.

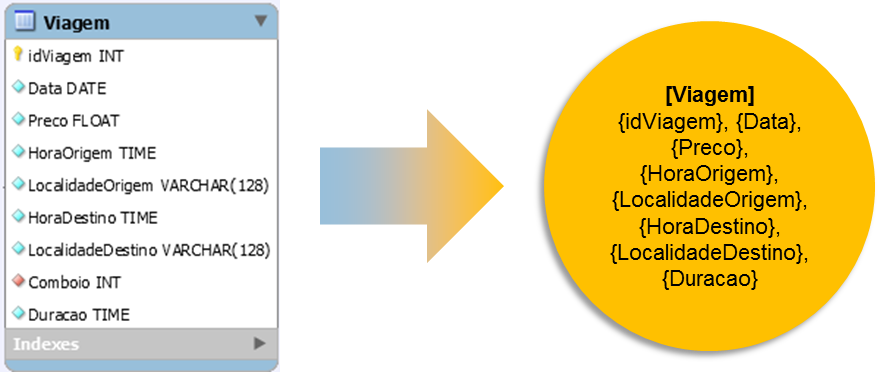
****

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.

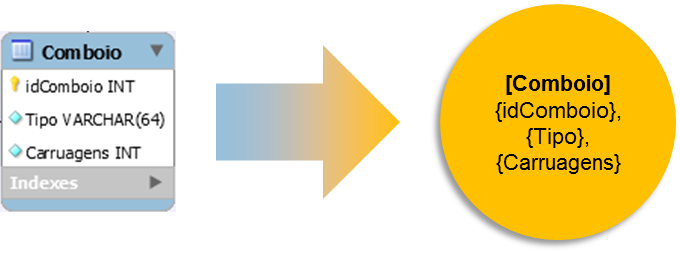
****

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.

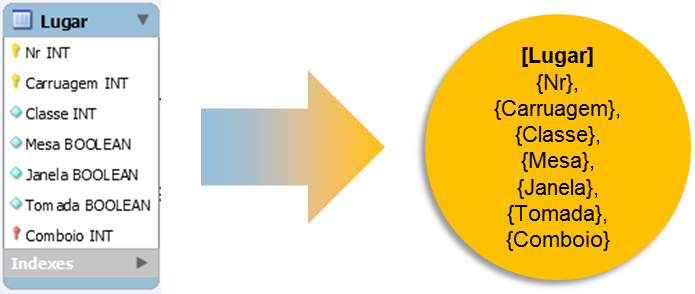
****

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.

## 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.

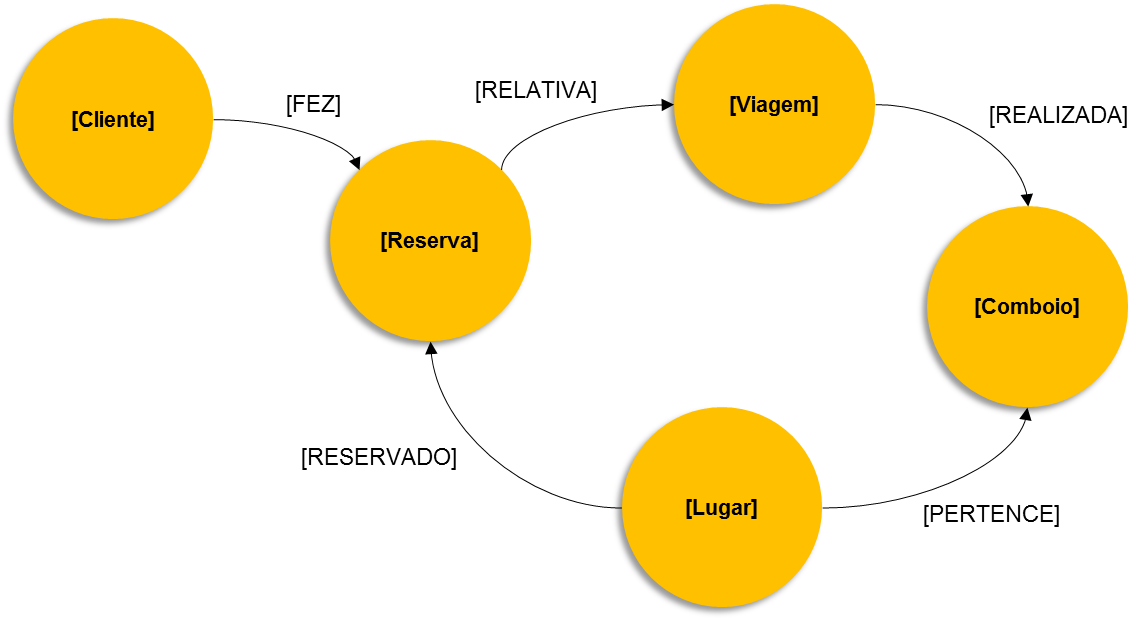


Figura 8 – Esquema geral da base de dados orientada por grafos

# 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*.

## 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:

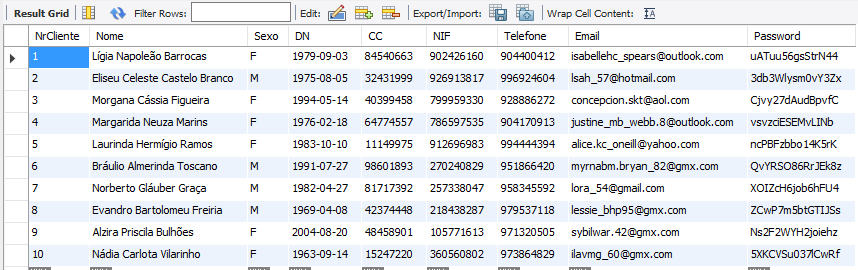


Figura 9 – Resultado da execução da *query* SELECT \* FROM cliente

Ao escolher a opção *Export* ():

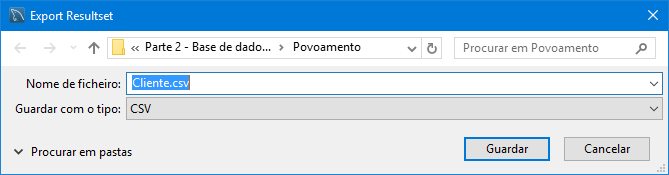


Figura 10 – Resultado do uso da opção *Export* do *MySQL Workbench*

## 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);

# 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;

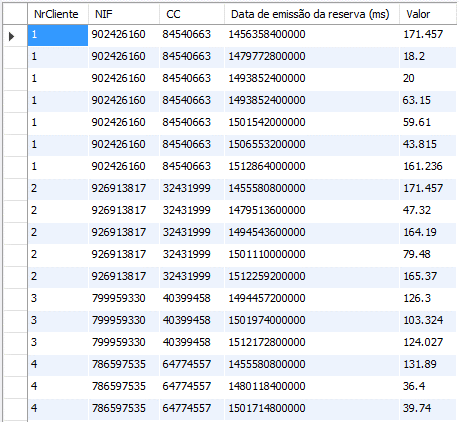


Figura 11 – Parte do resultado da *query* 4 em SQL

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;



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;

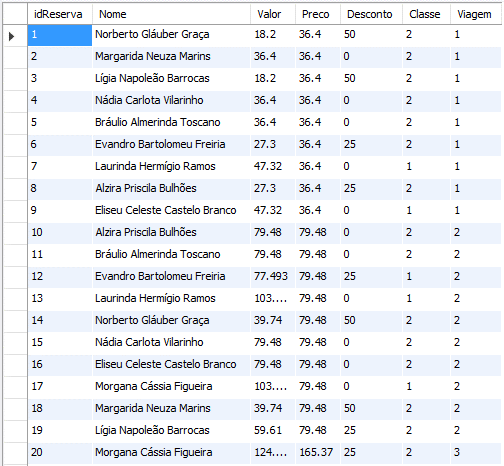


Figura 13 – Parte do resultado da *query* 8 em SQL

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;

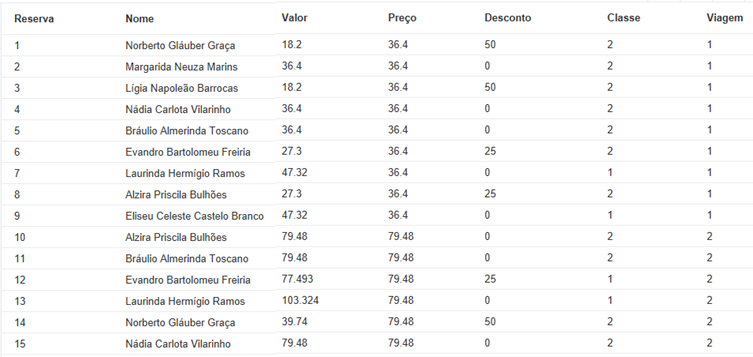


Figura 14 – Parte do resultado da *query* 8 em CQL

**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);

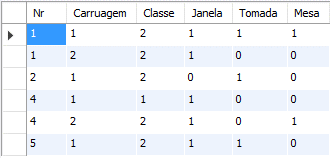


Figura 15 – Resultado da *query* 13 em SQL

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

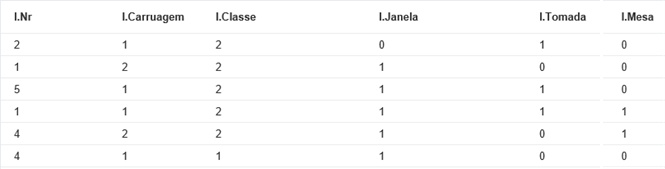


Figura 16 – Resultado da *query* 13 em CQL

# 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 SGDB 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.

# 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

**BD** Base de Dados

**SGBD** Sistema de Gestão de Bases de Dados

**CQL** *Cypher Query Language*

**SQL** *Structured Query Language*