



Universidade do Minho

Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2017/2018

GestHotel

A70565 Bruno Arieira

A73974 Daniel Vieira

A74264 Rafael Silva

A74216 Rodrigo Ferreira

Data de Recepção	
Responsável	
Avaliação	
Observações	

GestHotel

A70565 Bruno Arieira

A73974 Daniel Vieira

A74264 Rafael Silva

A74216 Rodrigo Ferreira

Janeiro, 2018

Índice

Índice de Figuras	2
1. Introdução	4
1.1. Motivação e Objetivos	4
1.2. Justificação da Viabilidade do Projeto	4
1.3. Estrutura do Relatório	5
2. Esquema da Base de Dados Relacional	6
3. Migração de Dados	8
3.1. Ficheiros CSV	8
3.2. Importação de Dados	9
3.3. <i>Unique Constraints</i>	16
3.4. Relacionamentos	20
3.4.1. Cartao_Cidadao – Cliente	20
3.4.2. Cliente – Reserva	21
3.4.3. Hotel – Quarto	22
3.4.4. Hotel – Reserva	23
3.4.5. Morada – Cliente	24
3.4.6. Reserva – Tipo_Reserva	25
3.5. Modelo Neo4j	26
4. Queries	27
4.1. Apresenta a informação completa do cliente (<i>querie 1</i>)	27
4.2. Total gasto por cada cliente (<i>querie 2</i>)	28
4.3. Quantidade de reservas por cada cliente (<i>querie 3</i>)	29
4.4. Faturação por cada hotel (<i>querie 4</i>)	30
4.5. Quantidade de quartos reservados por um cliente (<i>querie 5</i>)	31
4.6. Tipo de reserva por cliente (<i>querie 6</i>)	32
5. Conclusão	33

Índice de Figuras

Figura 1- Modelo Conceptual do SBD Relacional	6
Figura 2- Modelo Lógico do Sistema de Bases de Dados Relacional	7
Figura 3- Especificação Tabular da Entidade Hotel no MySQL	8
Figura 4- Ficheiro “hotel.csv”	8
Figura 5- Import do ficheiro “cartao_cidadao.csv”	9
Figura 6- Neo4j – Nodos cartao_cidadao	9
Figura 7- Import do ficheiro “cliente.csv”	10
Figura 8- Neo4j – Nodos cliente	10
Figura 9- Import do ficheiro “hotel.csv”	11
Figura 10- Neo4j – Nodos hotel	11
Figura 11- Import do ficheiro “quarto.csv”	12
Figura 12- Neo4j – Nodos quarto	13
Figura 13- Import do ficheiro “reserva.csv”	13
Figura 14- Neo4j – Nodos reserva	14
Figura 15- Import do ficheiro “tipo_reserva.csv”	14
Figura 16- Neo4j – Nodos tipo_reserva	15
Figura 17- Import do ficheiro “morada.csv”	15
Figura 18- Neo4j – Nodos morada	16
Figura 19- <i>Unique Constraint</i> id_Cartao_Cidadao (Cartao_Cidadao)	17
Figura 20- <i>Unique Constraint</i> id_Cliente (Cliente)	17
Figura 21- <i>Unique Constraint</i> id_Hotel (Hotel)	18
Figura 22- <i>Unique Constraint</i> id_Reserva (Reserva)	18
Figura 23- <i>Unique Constraint</i> id_Tipo_Reserva (Tipo_Reserva)	18
Figura 24- <i>Unique Constraint</i> id_Morada (Morada)	19
Figura 25- Relacionamento entre Cartao_Cidadao e Cliente	20
Figura 26- Neo4j- Relacionamento entre Cartao_Cidadao e Cliente	20
Figura 27- Relacionamento entre Cartao_Cidadao e Cliente	21
Figura 28- Neo4j- Relacionamento entre Cartao_Cidadao e Cliente	21
Figura 29- Relacionamento entre Cartao_Cidadao e Cliente	22
Figura 30- Neo4j- Relacionamento entre Cartao_Cidadao e Cliente	22
Figura 31- Relacionamento entre Cartao_Cidadao e Cliente	23
Figura 32- Neo4j- Relacionamento entre Cartao_Cidadao e Cliente	23
Figura 33- Relacionamento entre Cartao_Cidadao e Cliente	24
Figura 34- Neo4j- Relacionamento entre Cartao_Cidadao e Cliente	24
Figura 35- Relacionamento entre Cartao_Cidadao e Cliente	25
Figura 36- Neo4j- Relacionamento entre Cartao_Cidadao e Cliente	25
Figura 37- Modelo Neo4j	26

Figura 38- <i>Query 1 em MySQL</i>	27
Figura 39- <i>Query 1 em Neo4j</i>	27
Figura 40- <i>Query 2 em MySQL</i>	28
Figura 41- <i>Query 2 em Neo4j</i>	28
Figura 42- <i>Query 3 em MySQL</i>	29
Figura 43- <i>Query 3 em Neo4j</i>	29
Figura 44- <i>Query 4 em MySQL</i>	30
Figura 45- <i>Query 4 em Neo4j</i>	30
Figura 46- <i>Query 5 em MySQL</i>	31
Figura 47- <i>Query 5 em Neo4j</i>	31
Figura 48- <i>Query 6 em MySQL</i>	32
Figura 49- <i>Query 6 em Neo4j</i>	32

1. Introdução

O projeto apresentado neste relatório consiste na migração de um sistema de base de dados relacional, que foi implementado anteriormente através do sistema de gestão de bases de dados MySQL, para um sistema de base de dados não relacional usando o sistema de bases de dados Neo4j orientado a grafos.

Depois da revisão e análise do modelo conceptual e respetivo esquema lógico, prosseguimos para a exportação dos dados das tabelas da base de dados relacional, para que fosse possível fazer a implementação dos respetivos nodos, através da importação de dados dos ficheiros “.csv”. De seguida, identificamos algumas restrições relacionadas com as propriedades dos nodos, criando assim os relacionamentos existentes entre eles.

Por fim, elaborámos um conjunto de queries em Neo4j, que têm o mesmo objetivo das queries criadas no sistema relacional, ou seja, para testar a operacionalidade do Sistema implementado e a consistência dos dados.

Área de Aplicação: Planeamento e execução de sistemas de bases de dados não relacionais relativo a reservas de quartos de hotel

Palavras-Chave: Hotel, Reserva, Quarto, NoSQL, Neo4j

1.1. Motivação e Objetivos

O principal objetivo da realização deste projeto é ter contacto com uma nova perspetiva relativa a base de dados, adquirindo rotinas de planeamento e execução de sistemas de bases de dados não relacionais.

Com o aumento considerável do volume de dados no nosso sistema de bases de dados, foi necessário explorar linguagens NoSQL para que a eficiência não seja limitada. Para isto, foi-nos proposto usar um sistema de bases de dados não relacional orientado por grafos, suportado pela aplicação Neo4j.

1.2. Justificação da Viabilidade do Projeto

Existem algumas vantagens em trocar um sistema de bases de dados relacional por uma solução noSQL. Uma delas é o seu escalonamento. Como este não possui nenhum tipo de esquema pré-definido, o modelo possui maior flexibilidade, facilitando a inserção de novos elementos. Para além disso, outra das vantagens nesta troca é a disponibilidade das bases de dados não relacionais. Como existe uma grande distribuição dos dados, isso origina que uma maior quantidade de solicitações aos dados seja executada pelo sistema num intervalo de tempo menor.

1.3. Estrutura do Relatório

Na parte inicial, é explicado o processo de migração de dados do Sistema de gestão de bases de dados relacional para o Sistema de bases de dados não relacional orientado por grafos, e como se realizou a importação dos mesmos.

Depois disso, segue-se a explicação das restrições utilizadas para garantir a integridade dos dados após a sua inserção, tal como a recriação dos relacionamentos para o novo Sistema de bases de dados.

Por último, é apresentado o modelo final em Neo4j e algumas queries que foram realizadas em SQL no Sistema de gestão de bases de dados relacional, desta vez realizadas em cypher.

2. Esquema da Base de Dados Relacional

Uma vez que este trabalho consiste na migração de um sistema de base de dados relacional para um modelo de dados não relacional, foram identificadas todas as entidades, atributos e relacionamentos através do modelo conceptual, representado na figura.

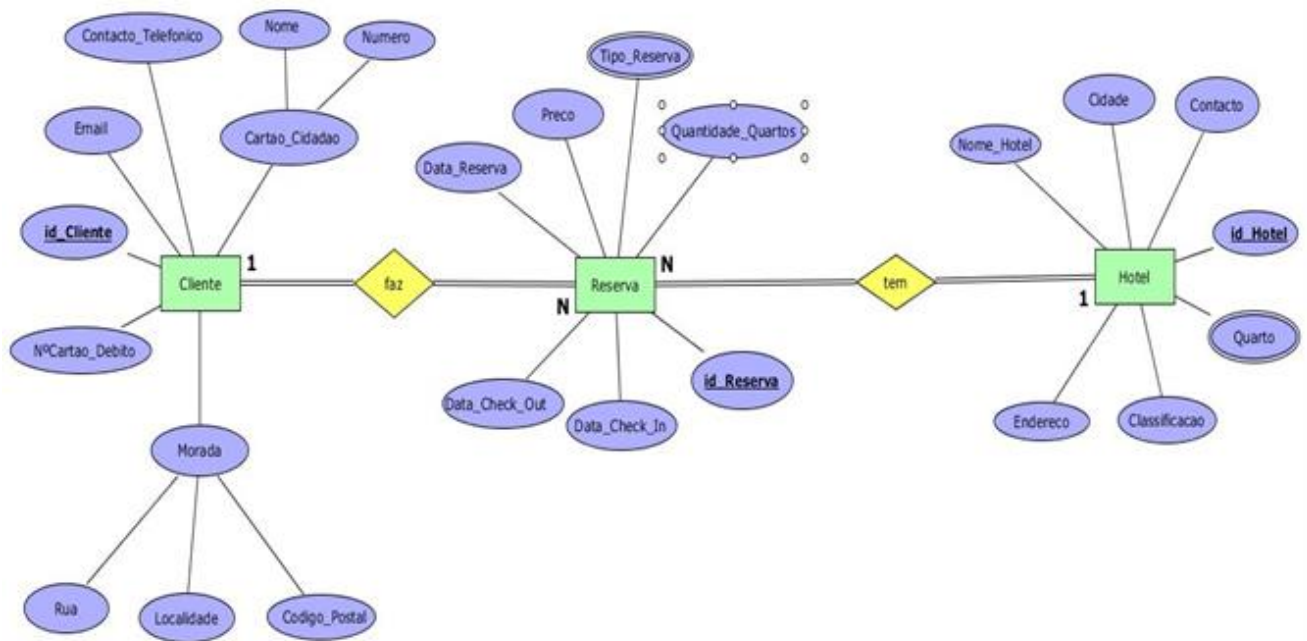


Figura 1- Modelo Conceptual do SBD Relacional

Depois de concluir o modelo conceitual, fizemos a sua conversão para o esquema lógico onde, para além das entidades e atributos, podemos verificar como é preservada a integridade dos dados.

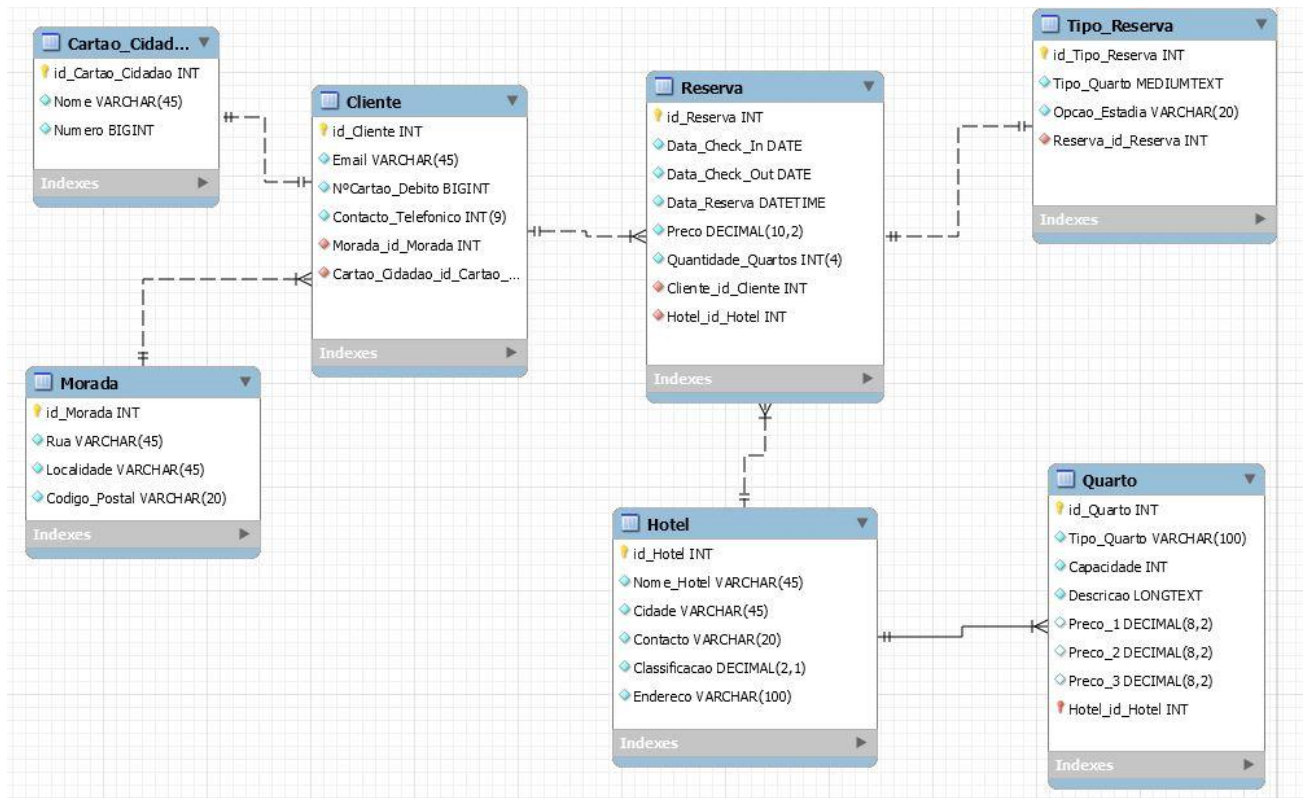


Figura 2- Modelo Lógico do Sistema de Bases de Dados Relacional

3. Migração de Dados

3.1. Ficheiros CSV

Ao inicializarmos o processo de migração de dados, tivemos que exportar cada tabela do modelo relacional que fizemos no projeto para o seu respetivo ficheiro csv.

De seguida, é exemplificado este passo com a exportação de dados da tabela Hotel:

id_Hotel	Nome_Hotel	Cidade	Contacto	Classificacao	Endereco
1	El Palace Barc...	Barcelona	+31 23 12 34...	4.8	Gran Via de les...
2	Hotel Liabeny	Madrid	+32 12 32 111	4.7	Salud, 3, Centr...
3	NH Madrid Nac...	Madrid	+32 45 12 342	4.0	Paseo del Prad...
4	Jupiter Lisboa...	Lisboa	+21 34 21 321	4.2	Avenida da Re...
5	Sheraton Porto...	Porto	+23 23 12 233	5.0	Rua Tenente V...
6	Hotel The Peni...	Paris	+35 32 13 432	5.0	19 avenue Kle...
7	Park Plaza Wes...	Londres	+34 21 31 312	4.3	Westminster B...
8	Meliá Berlin	Berlim	+36 21 32 212	4.5	Friedrichstr. 1...
9	Vincci Gala	Barcelona	+31 32 12 432	4.0	Ronda Sant Pe...
10	Hotel Princesa...	Lisboa	+21 23 21 323	3.5	Rua Gomes Fre...
11	The Church Pal...	Roma	+33 21 34 211	4.6	Via Aurelia 48...
12	Hotel Rialto	Veneza	+30 43 65 676	4.8	Riva del Ferro,...
NULL	NULL	NULL	NULL	NULL	NULL

Figura 3- Especificação Tabular da Entidade Hotel no MySQL

	A	B	C	D	E	F	G	H	I	J	K
1	id_Hotel,Nome_Hotel,Cidade,Contacto,Classificacao,Endereco										
2	1,"El Palace Barcelona",Barcelona,"+31 23 12 34 456",4.8,"Gran Via de les Corts Catalanes, 668, Eixample, 08010 Barcelona, Espanha"										
3	2,"Hotel Liabeny",Madrid,"+32 12 32 111",4.7,"Salud, 3, Centro de Madrid, 28013 Madrid, Espanha"										
4	3,"NH Madrid Nacional",Madrid,"+32 45 12 342",4.0,"Paseo del Prado, 48, Centro de Madrid, 28014 Madrid, Espanha"										
5	4,"Jupiter Lisboa Hotel",Lisboa,"+21 34 21 321",4.2,"Avenida da Republica, 46, Avenidas Novas, 1050-195 Lisboa, Portugal"										
6	5,"Sheraton Porto Hotel & Spa",Porto,"+23 23 12 233",5.0,"Rua Tenente Valadim, 146, Lordelo do Ouro e Massarelos, 4100-476 Porto, Portugal"										
7	6,"Hotel The Peninsula Paris",Paris,"+35 32 13 432",5.0,"19 avenue Kleber, 16Âº arrondissement, 75116 Paris, França"										
8	7,"Park Plaza Westminster Bridge London",Londres,"+34 21 31 312",4.3,"Westminster Bridge Road, Lambeth, Londres, SE1 7UT, Reino Unido"										
9	8,"Meliá Berlin",Berlim,"+36 21 32 212",4.5,"Friedrichstr. 103, Mitte, 10117 Berlin, Alemanha"										
10	9,"Vincci Gala",Barcelona,"+31 32 12 432",4.0,"Ronda Sant Pere, 32, Eixample, 08010 Barcelona, Espanha"										
11	10,"Hotel Princesa Lisboa Centro",Lisboa,"+21 23 21 323",3.5,"Rua Gomes Freire, 130, Arroios, 1150-180 Lisboa, Portugal"										
12	11,"The Church Palace",Roma,"+33 21 34 211",4.6,"Via Aurelia 481, AurÃ©lio, 00165 Roma, Itália"										
13	12,"Hotel Rialto",Veneza,"+30 43 65 676",4.8,"Riva del Ferro, San Marco, 30124 Veneza, Itália"										

Figura 4- Ficheiro "hotel.csv"

3.2. Importação de Dados

Neste tópico é explicado com mais detalhe o procedimento da importação dos ficheiros csv anteriormente referidos. Para criar os seus respetivos nodos (caraterísticas do NEO4J) usamos comandos específicos de Cypher.

Para a criação destes comandos, é necessária a consulta do modelo relacional criado anteriormente, para que haja a importação de todos os dados de cada entidade (verificação dos atributos tem que ser feita).

3.2.1. Cartão de Cidadão

Como podemos verificar no modelo de dados relacional, a tabela `cartao_cidadao` contém uma chave primária (`id_Cartao_Cidadao`) e dois atributos (`Nome` e `Numero`). Assim, ao fazer a importação de dados e criação de nodos, são importados todos os dados do ficheiro “`cartao_cidadao.csv`”.

```
1 LOAD CSV WITH HEADERS FROM 'file:///cartao_cidadao.csv' AS row
2 CREATE (cc: Cartao_Cidadao {id_Cartao_Cidadao: toInt(row.id_Cartao_Cidadao)})
3 SET cc.Nome = toString(row.Nome),
4     cc.Numero = toInt(row.Numero)
5 RETURN cc;
```

Figura 5- Import do ficheiro “`cartao_cidadao.csv`”

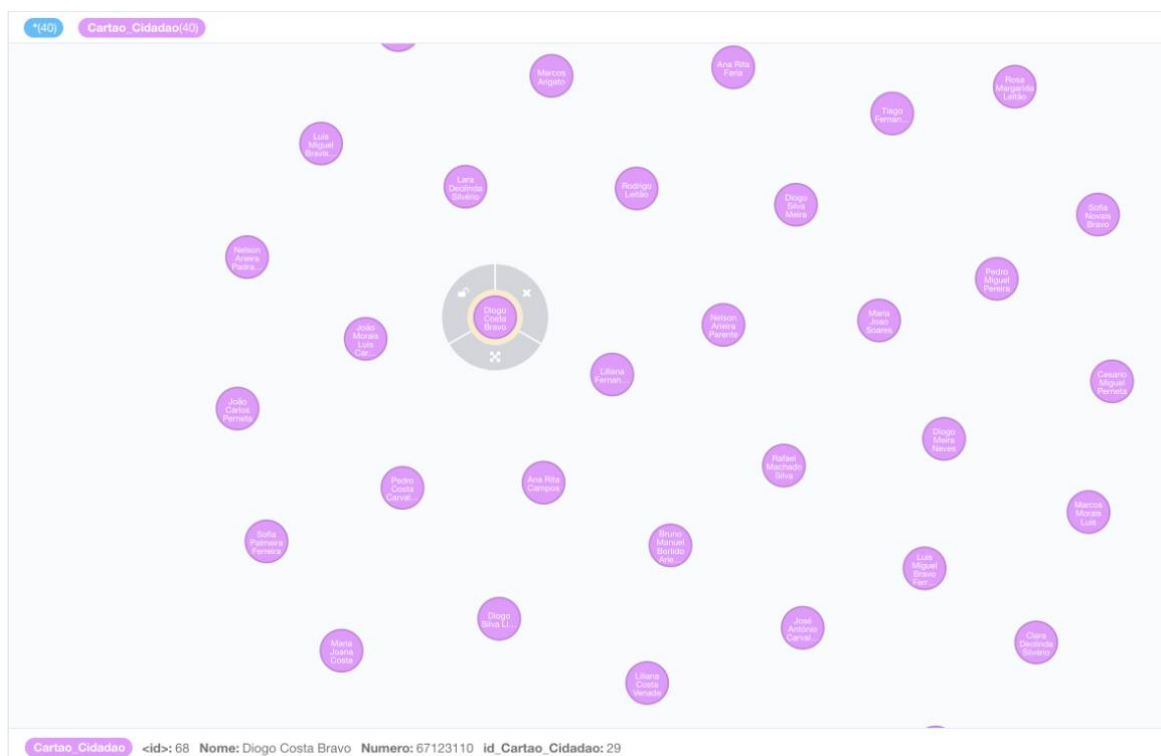


Figura 6- Neo4j – Nodos `cartao_cidadao`

3.2.2. Cliente

Como podemos verificar no modelo de dados relacional, a tabela cliente contém uma chave primária (id_Cliente) e 3 atributos (Email, Contacto_Telefonico, NºCartao_Debito) além das 2 chaves estrangeiras Morada_id_Morada, Cartao_Cidadao_id_Cartao_Cidadao. Assim, ao fazer a importação de dados e criação de nodos, são importados todos os dados do ficheiro “cartao_cidadao.csv”.

```
1 LOAD CSV WITH HEADERS FROM 'file:///cliente.csv' AS row
2 CREATE (c: Cliente {id_Cliente: toInt(row.id_Cliente)})
3 SET c.Email = toString(row.Email),
4     c.Contacto_Telefonico = toInt(row.Contacto_Telefonico),
5     c.NºCartao_Debito = toInt(row.NºCartao_Debito),
6     c.Morada_id_Morada = toInt(row.Morada_id_Morada),
7     c.Cartao_Cidadao_id_Cartao_Cidadao = toInt(row.Cartao_Cidadao_id_Cartao_Cidadao)
8 RETURN c;
9
```

Figura 7- Import do ficheiro “cliente.csv”

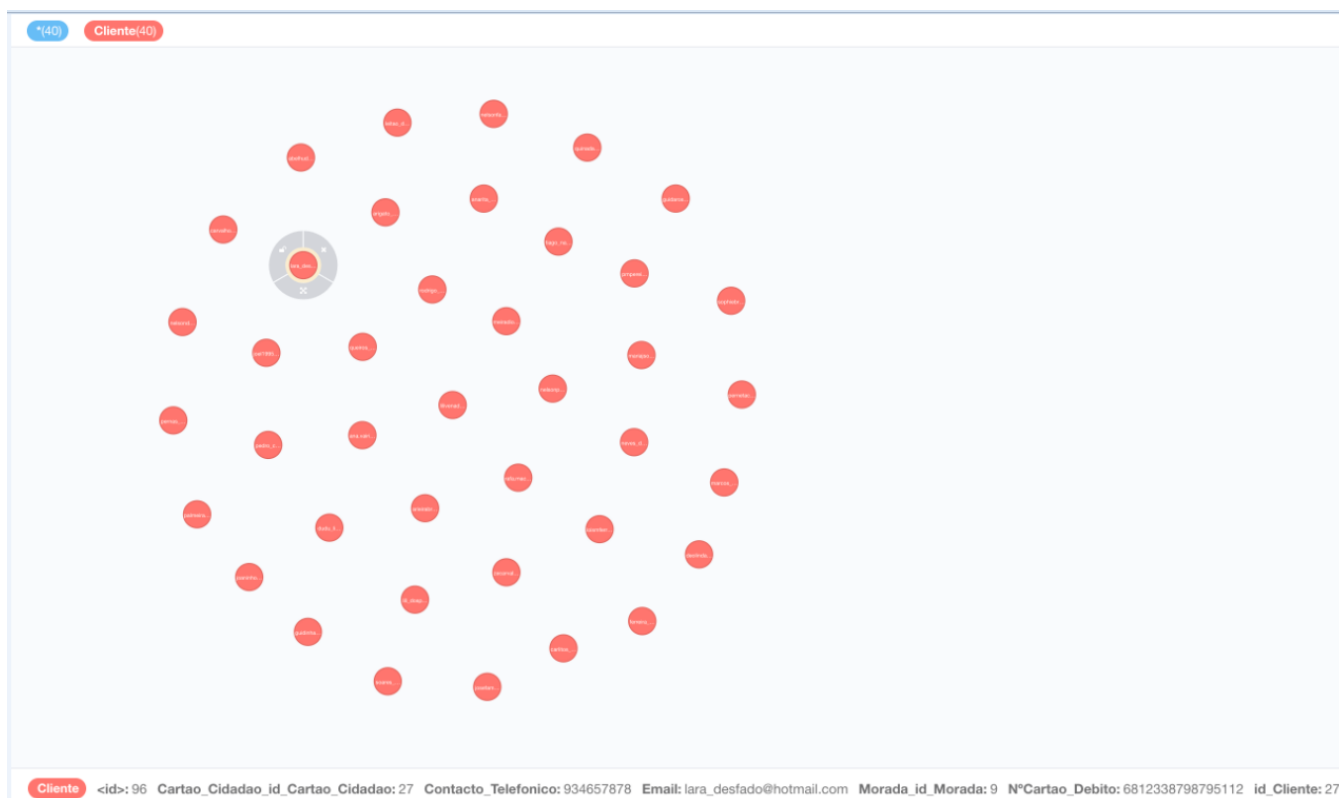


Figura 8- Neo4j – Nodos cliente

3.2.3. Hotel

Como podemos verificar no modelo de dados relacional, a tabela hotel contém uma chave primária (id_Hotel) e cinco atributos (Nome_Hotel, Cidade, Contacto, Classificacao e Endereco). Assim, ao fazer a importação de dados e criação de nodos, são importados todos os dados do ficheiro “hotel.csv”.

```
1 LOAD CSV WITH HEADERS FROM 'file:///hotel.csv' AS row
2 CREATE (h: Hotel {id_Hotel: toInt(row.id_Hotel)})
3 SET h.Nome_Hotel = toString(row.Nome_Hotel),
4     h.Cidade = toString(row.Cidade),
5     h.Contacto = toString(row.Contacto),
6     h.Classificacao = toFloat(row.Classificacao),
7     h.Endereco = toString(row.Endereco)
8 RETURN h;
```

Figura 9- Import do ficheiro “hotel.csv”

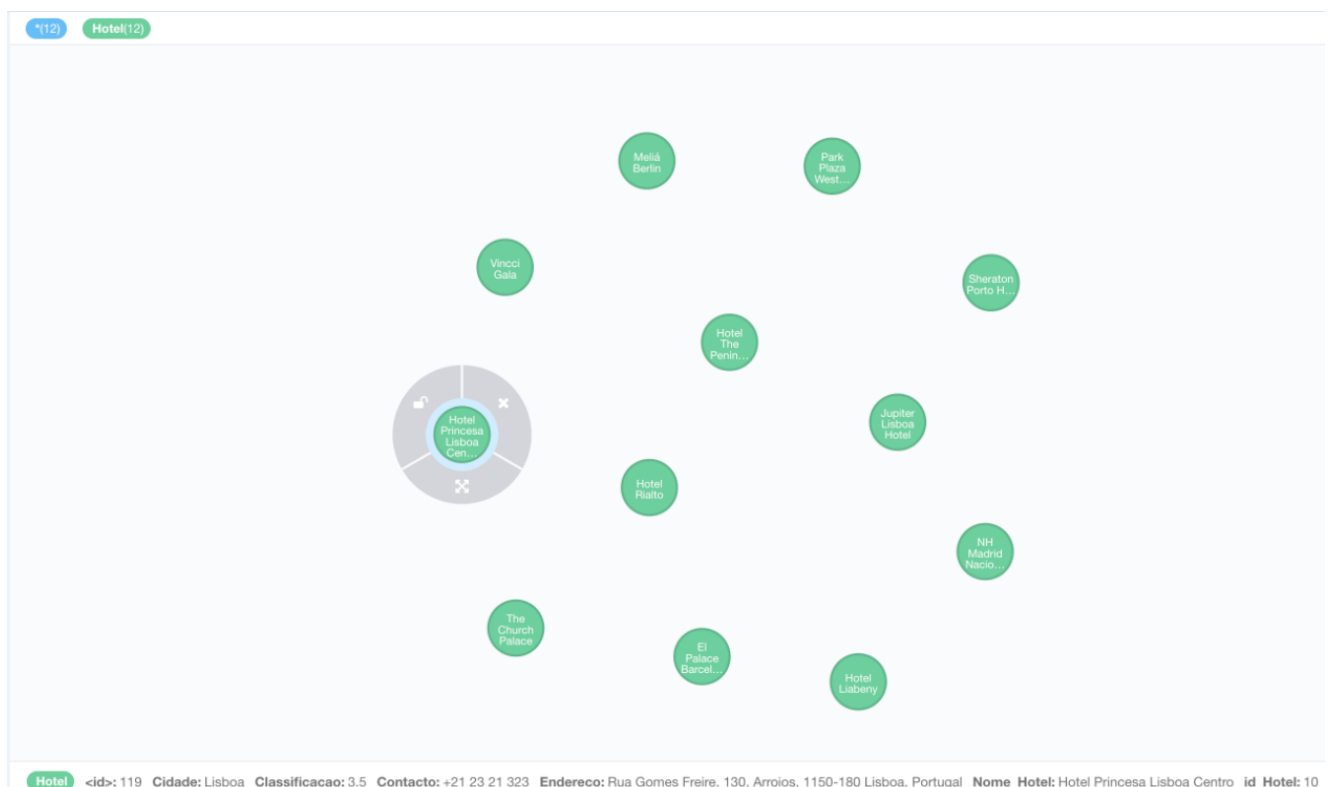


Figura 10- Neo4j – Nodos hotel

3.2.4. Quarto

Como podemos verificar no modelo de dados relacional, a tabela quarto contém uma chave primária (id_Quarto) e seis atributos (Tipo_Quarto, Capacidade, Descricao, Preco_1 (preço do quarto sem nada incluído), Preco_2 (preço do quarto com o pequeno almoço incluído) e Preco_3 (preço do quarto com as refeições todas incluídas)). Há também uma chave estrangeira Hotel_id_Hotel.

Assim, ao fazer a importação de dados e criação de nodos, são importados todos os dados do ficheiro “quarto.csv”.

```
1 | LOAD CSV WITH HEADERS FROM 'file:///quarto.csv' AS row
2 | CREATE (q: Quarto {id_Quarto: toInt(row.id_Quarto)})
3 | SET q.Tipo_Quarto = toString(row.Tipo_Quarto),
4 |   q.Capacidade = toInt(row.Capacidade),
5 |   q.Descricao = toString(row.Descricao),
6 |   q.Preco_1 = toFloat(row.Preco_1),
7 |   q.Preco_2 = toFloat(row.Preco_2),
8 |   q.Preco_3 = toFloat(row.Preco_3),
9 |   q.Hotel_id_Hotel = toInt(row.Hotel_id_Hotel)
10 | RETURN q;
```

Figura 11- Import do ficheiro “quarto.csv”

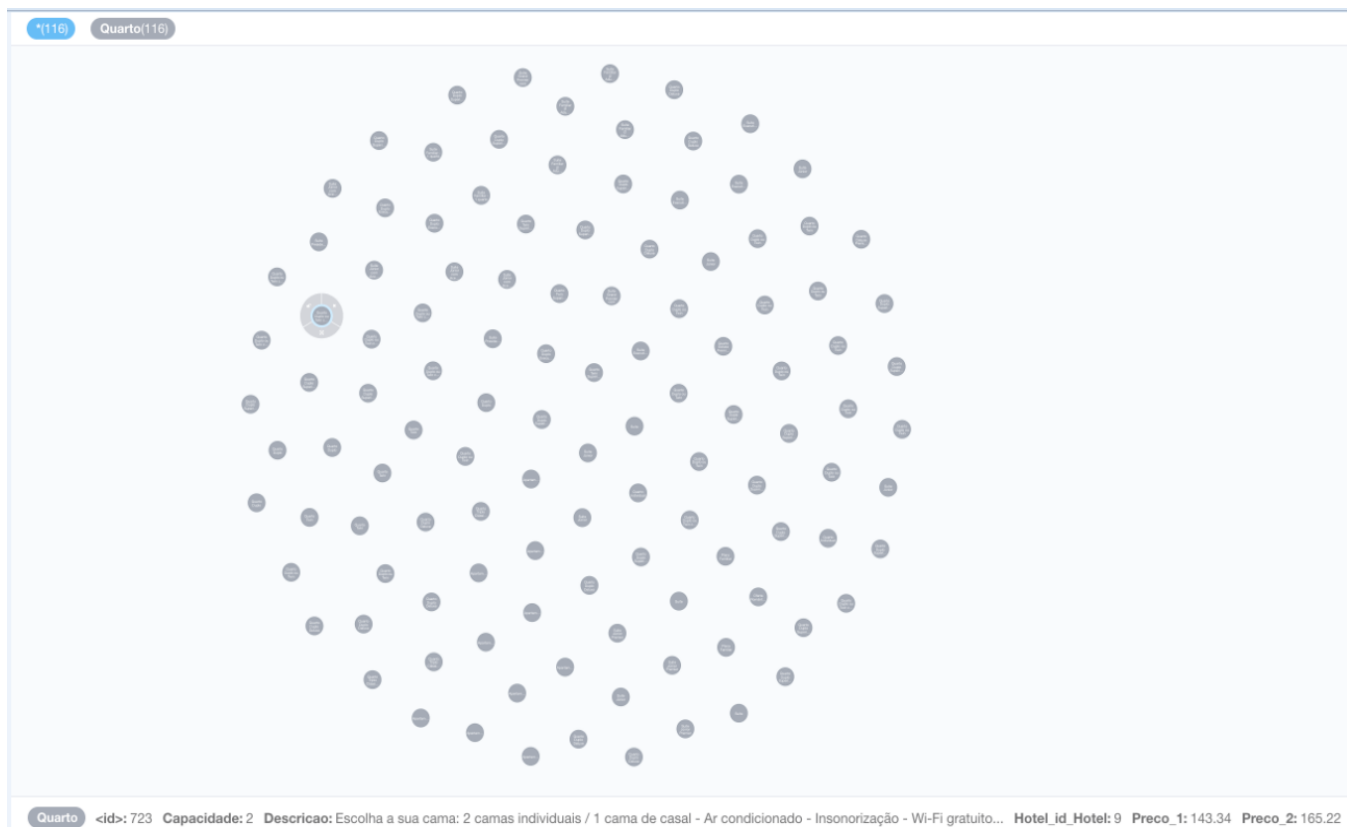


Figura 12- Neo4j – Nodos quarto

3.2.5. Reserva

Como podemos verificar no modelo de dados relacional, a tabela reserva contém uma chave primária (id_Reserva) e cinco atributos (Data_Check_In, Data_Check_Out, Data_Reserva, Preco e Quantidade Quartos). Há também 2 chaves estrangeiras Cliente_id_Cliente e Hotel_id_Hotel.

Assim, ao fazer a importação de dados e criação de nodos, são importados todos os dados do ficheiro “reserva.csv”.

```

1 LOAD CSV WITH HEADERS FROM 'file:///reserva.csv' AS row
2 CREATE (r: Reserva {id_Reserva: toInt(row.id_Reserva)})
3 SET r.Data_Check_In = toString(row.Data_Check_In),
4     r.Data_Check_Out = toString(row.Data_Check_Out),
5     r.Data_Reserva = toString(row.Data_Reserva),
6     r.Preco = toFloat(row.Preco),
7     r.Quantidade Quartos = toInt(row.Quantidade Quartos),
8     r.Cliente_id_Cliente = toInt(row.Cliente_id_Cliente),
9     r.Hotel_id_Hotel = toInt(row.Hotel_id_Hotel)
10 RETURN r;

```

Figura 13- Import do ficheiro “reserva.csv”

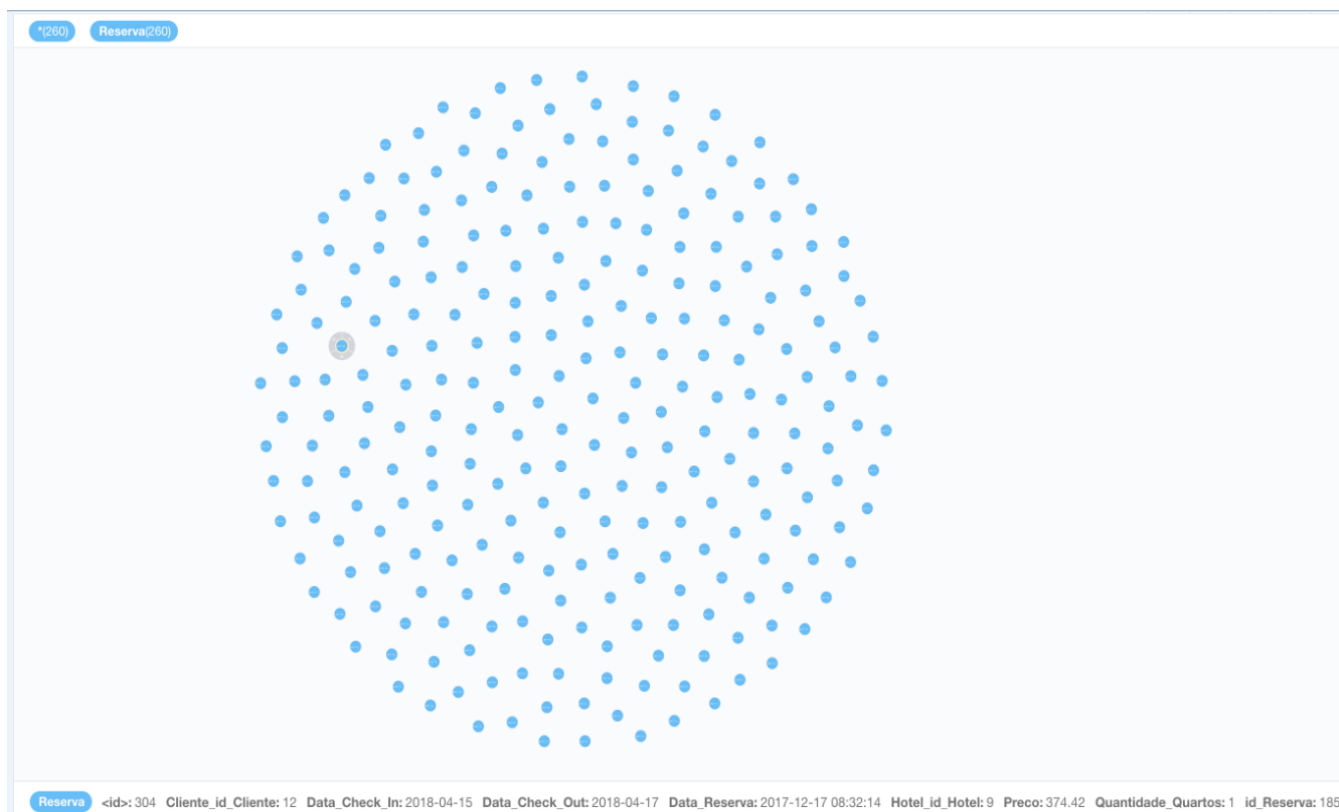


Figura 14- Neo4j – Nodos reserva

3.2.6. Tipo de Reserva

Como podemos verificar no modelo de dados relacional, a tabela `tipo_reserva` contém uma chave primária (`id_Tipo_Reserva`) e dois atributos (`Tipo_Quarto` e `Opcao_Estadia`). Há também 1 chave estrangeira chamada `Reserva_id_Reserva`.

Assim, ao fazer a importação de dados e criação de nodos, são importados todos os dados do ficheiro “`tipo_reserva.csv`”.

```

1 LOAD CSV WITH HEADERS FROM 'file:///tipo_reserva.csv' AS row
2 CREATE (tr: Tipo_Reserva {id_Tipo_Reserva: toInt(row.id_Tipo_Reserva)})
3 SET tr.Tipo_Quarto = toString(row.Tipo_Quarto),
4     tr.Opcao_Estadia = toString(row.Opcao_Estadia),
5     tr.Reserva_id_Reserva = toInt(row.Reserva_id_Reserva)
6 RETURN tr;

```

Figura 15- Import do ficheiro “`tipo_reserva.csv`”

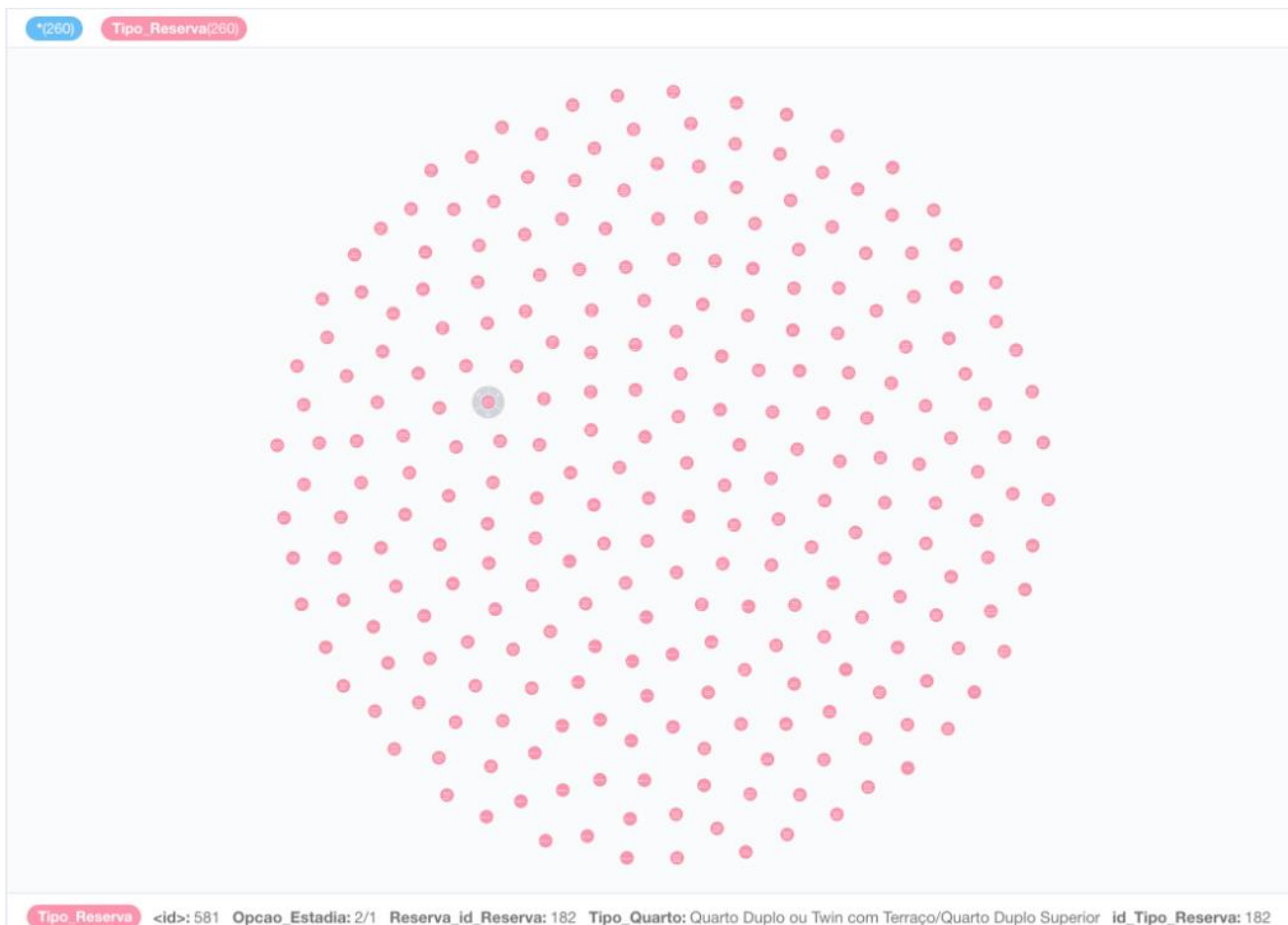


Figura 16- Neo4j – Nodos tipo_reserva

3.2.7. Morada

Como podemos verificar no modelo de dados relacional, a tabela morada (atributo composto) contém uma chave primária (id_Morada) e três atributos (Rua, Localidade e Codigo_Postal). Assim, ao fazer a importação de dados e criação de nodos, são importados todos os dados do ficheiro “morada.csv”.

```

1  LOAD CSV WITH HEADERS FROM 'file:///morada.csv' AS row
2  CREATE (m: Morada {id_Morada: toInt(row.id_Morada)})
3  SET m.Rua = toString(row.Rua),
4      m.Localidade = toString(row.Localidade),
5      m.Codigo_Postal = toString(row.Codigo_Postal)
6  return m;

```

Figura 17- Import do ficheiro “morada.csv”

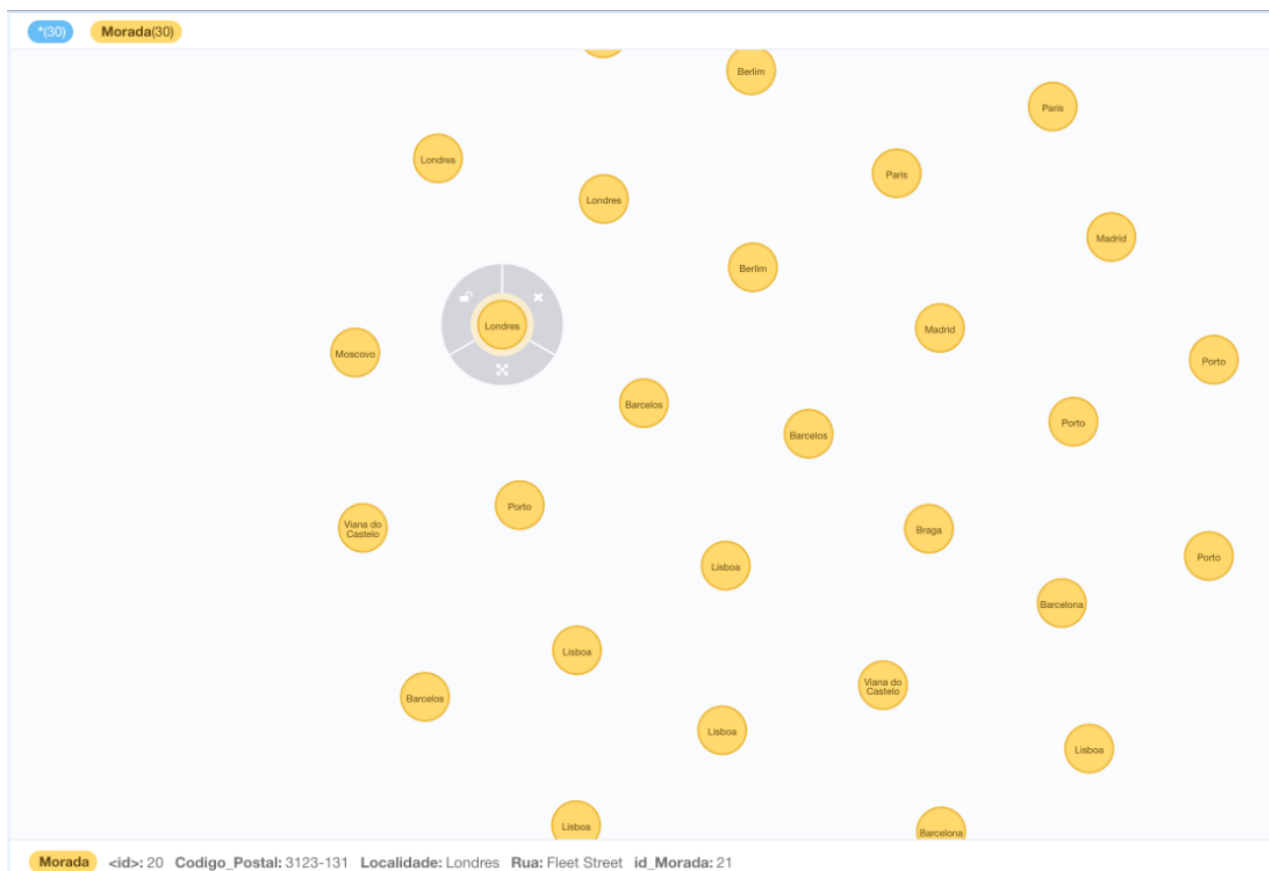


Figura 18- Neo4j – Nodos morada

3.3. *Unique Constraints*

Para uma correta migração do sistema, e de modo a impedir possíveis erros futuros e possibilitar a correta inserção de dados, foram criadas algumas *Unique Constraints*, que servem como garantia de que determinados valores só podem existir uma única vez em cada nodo.

De seguida, apresentámos algumas *Unique Constraint* criadas para o nosso sistema.

Nos nodos de Cartao_Cidadao foi criada uma *Unique Constraint* que garante que não podem existir id_Cartao_Cidadao repetidos:

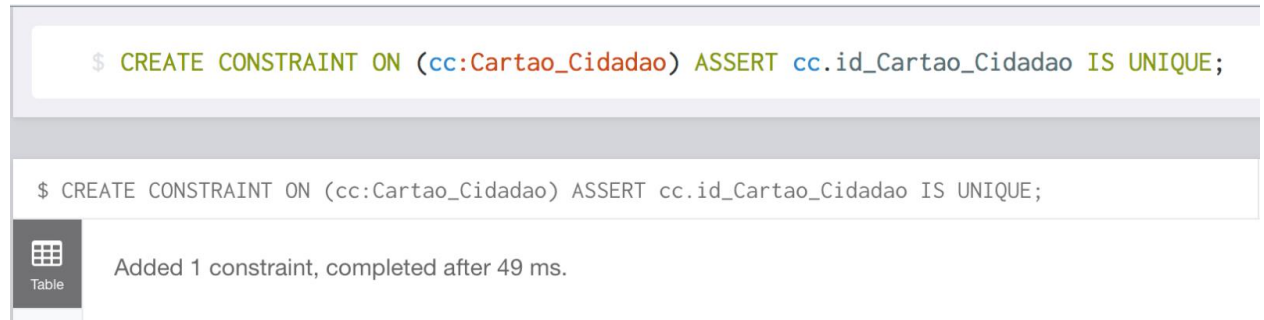


Figura 19- *Unique Constraint* id_Cartao_Cidadao (Cartao_Cidadao)

Nos nodos de Cliente foi criada uma *Unique Constraint* que garante que não existem id_Cliente iguais:

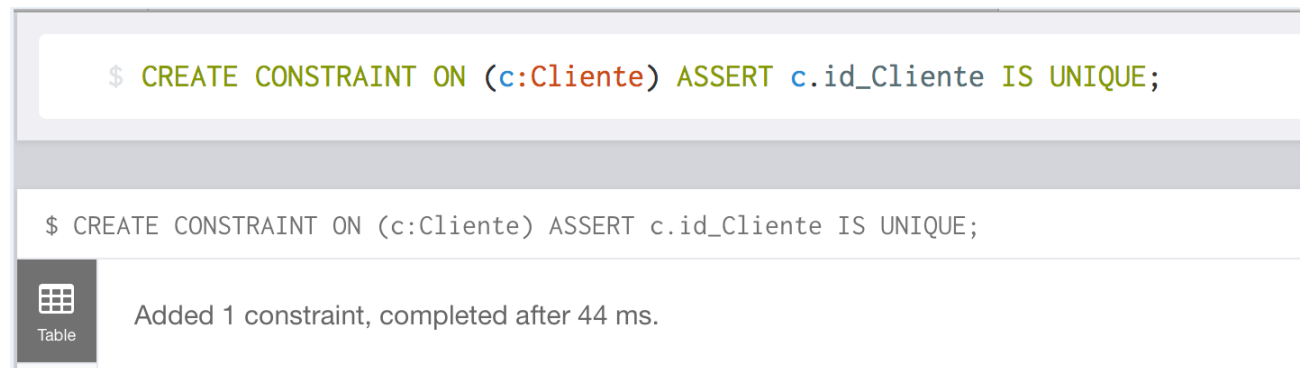


Figura 20- *Unique Constraint* id_Cliente (Cliente)

Nos nodos de Hotel foi criada uma *Unique Constraint* que garante que não existem id_Hotel iguais:

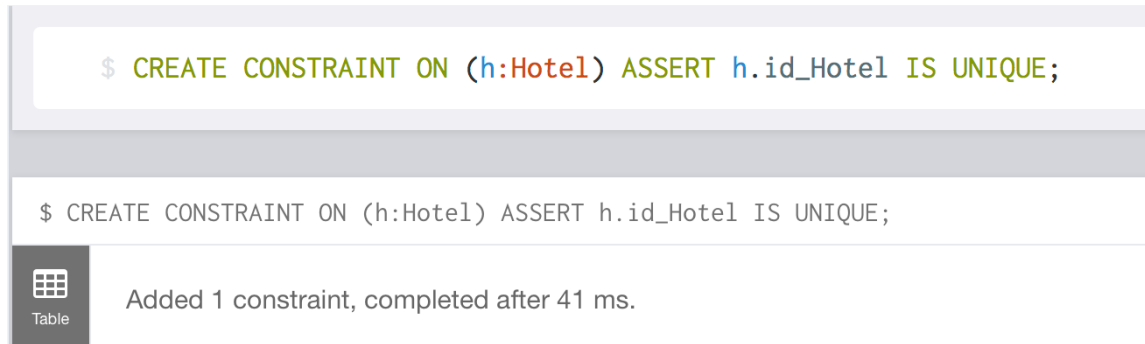


Figura 21- *Unique Constraint* id_Hotel (Hotel)

Nos nodos de Reserva foi criada uma *Unique Constraint* que garante que não existem id_Reserva iguais:

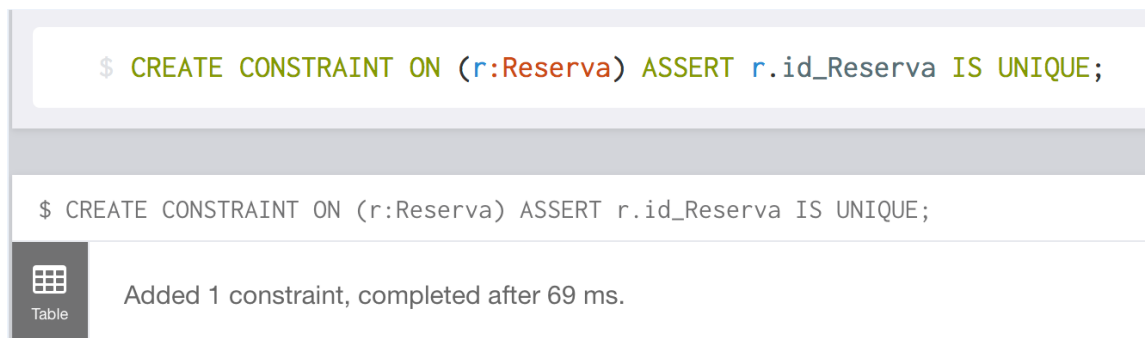


Figura 22- *Unique Constraint* id_Reserva (Reserva)

Nos nodos de Tipo_Reserva foi criada uma *Unique Constraint* que garante que não existem id_Tipo_Reserva iguais:

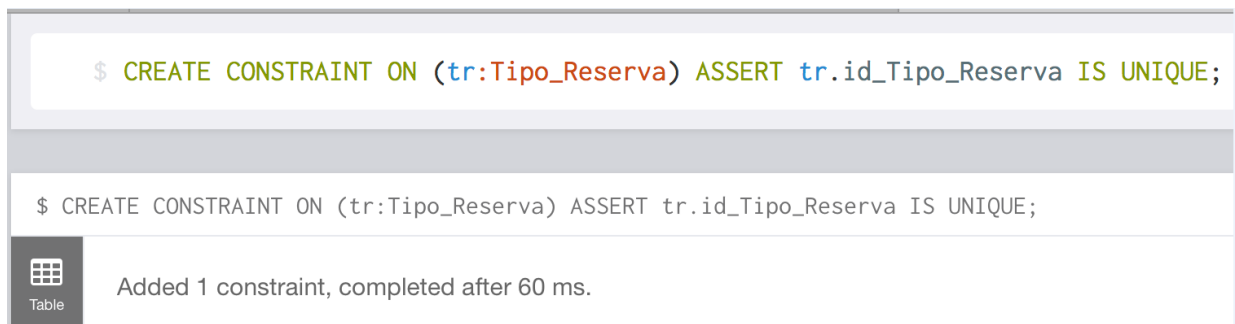


Figura 23- *Unique Constraint* id_Tipo_Reserva (Tipo_Reserva)

Nos nodos de Morada foi criada uma *Unique Constraint* que garante que não existem id_Morada iguais:

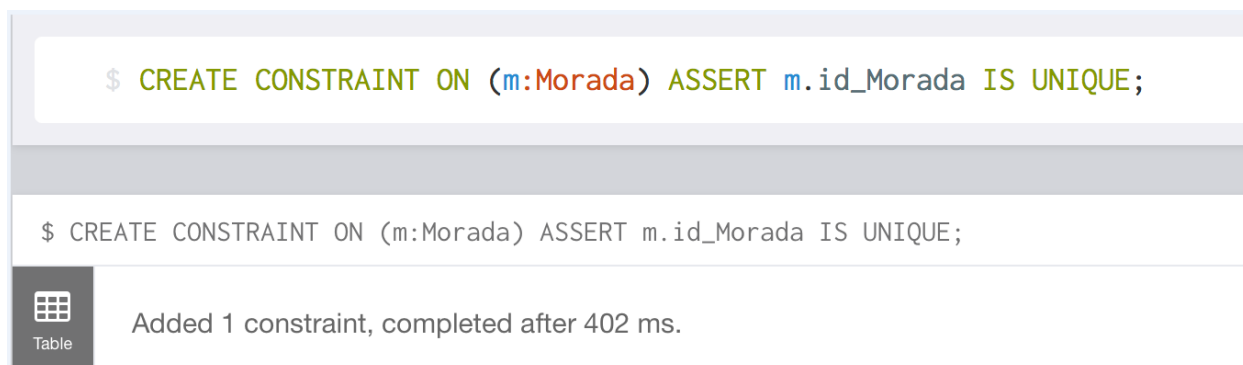


Figura 24- *Unique Constraint* id_Morada (Morada)

3.4. Relacionamentos

Em seguida, será explicado como criamos os diferentes relacionamentos existentes no sistema, com a demonstração dos respectivos comandos utilizados no cypher, bem como o ficheiro Neo4j correspondente.

3.4.1. Cartao_Cidadao – Cliente

```
MATCH(cc:Cartao_Cidado),(c:Cliente)
WHERE cc.id_Cartao_Cidado = c.Cartao_Cidado_id_Cartao_Cidado
CREATE (c)-[:PERTENCE]->(cc);
```

Figura 25- Relacionamento entre Cartao_Cidado e Cliente

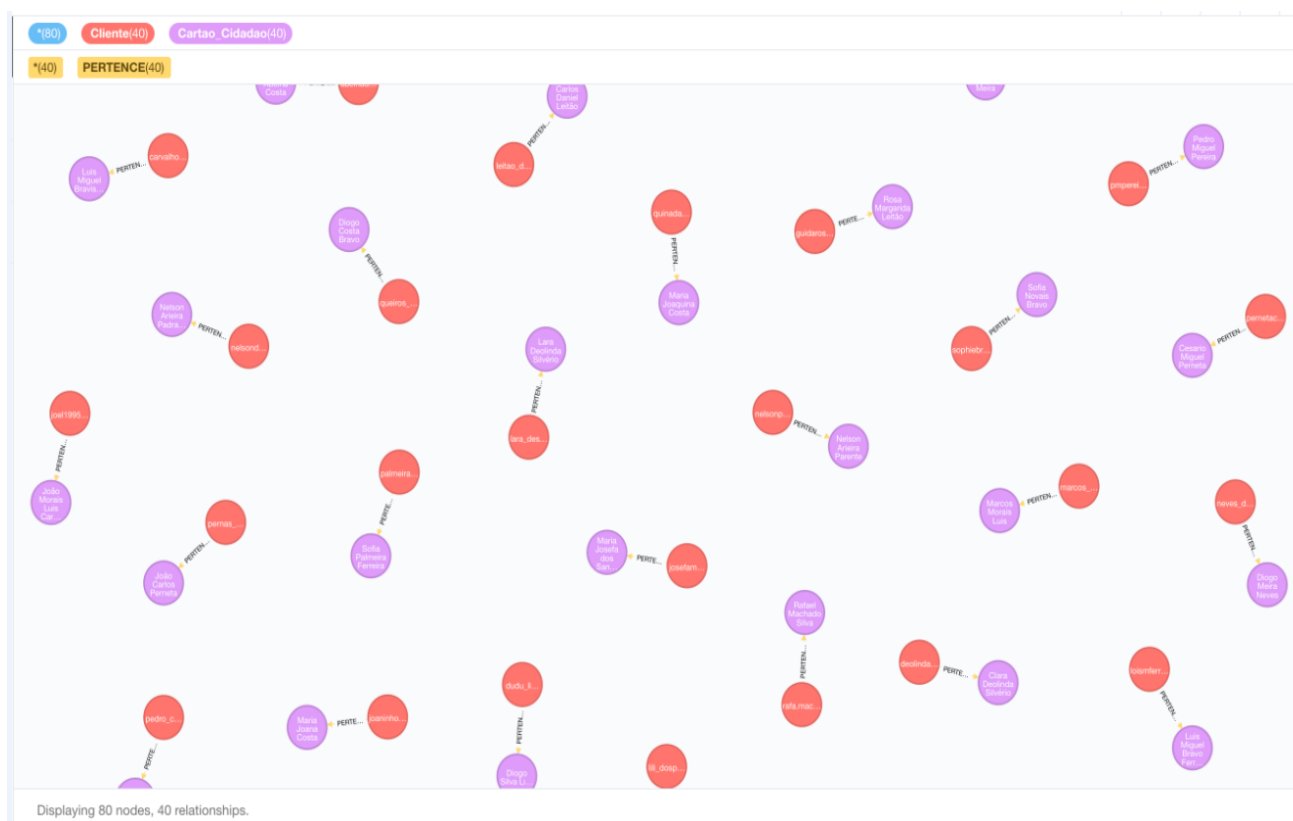


Figura 26- Neo4j- Relacionamento entre Cartao_Cidado e Cliente

3.4.2. Cliente – Reserva

```
MATCH(c:Cliente),(r:Reserva)
WHERE c.id_Cliente = r.Cliente_id_Cliente
CREATE (c)-[:EFETUA]->(r);
```

Figura 27- Relacionamento entre Cartao_Cidadao e Cliente

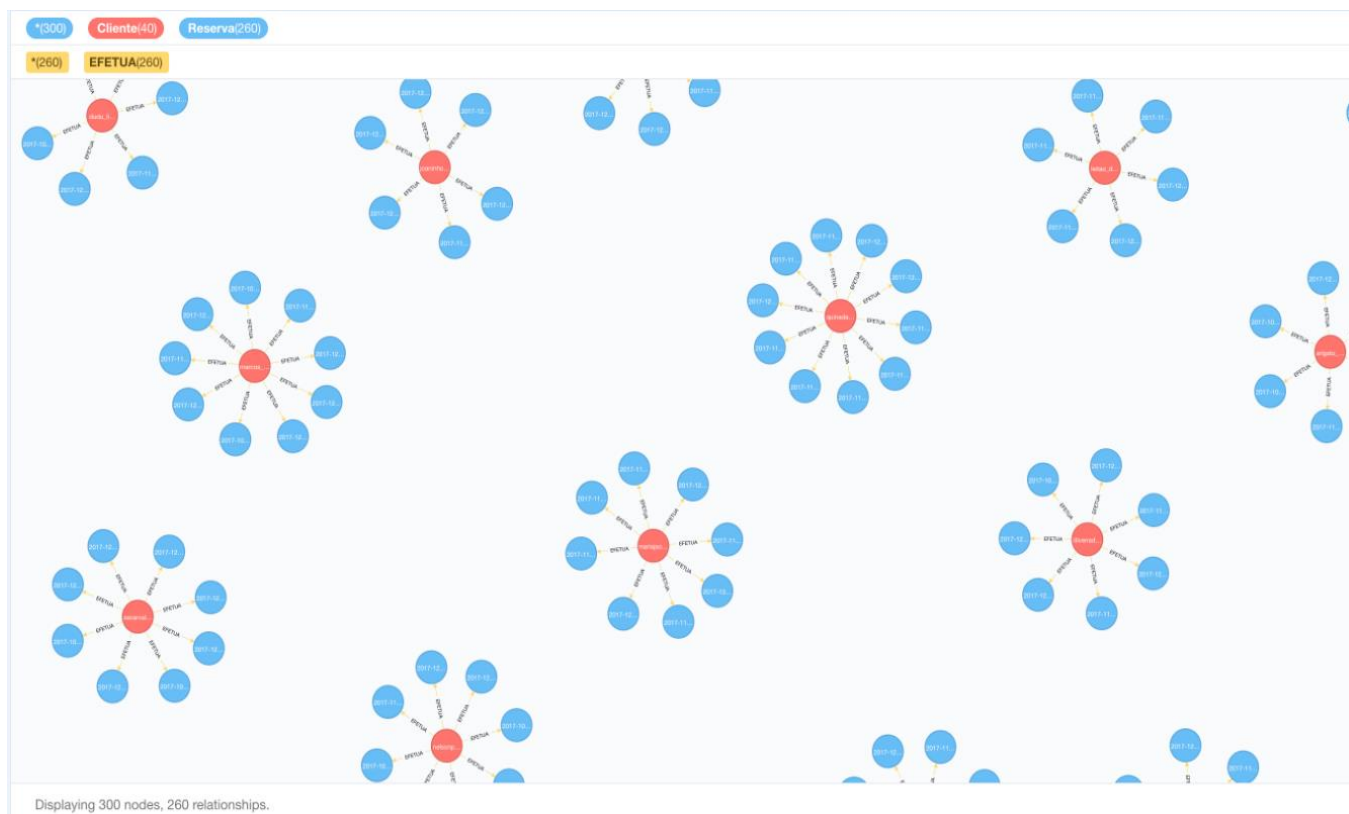


Figura 28- Neo4j- Relacionamento entre Cartao_Cidadao e Cliente

3.4.3. Hotel – Quarto

```
MATCH(q:Quarto),(h:Hotel)
WHERE q.Hotel_id_Hotel = h.id_Hotel
CREATE (h)-[:CONSTITUIDO]->(q);
```

Figura 29- Relacionamento entre Cartao_Cidadao e Cliente

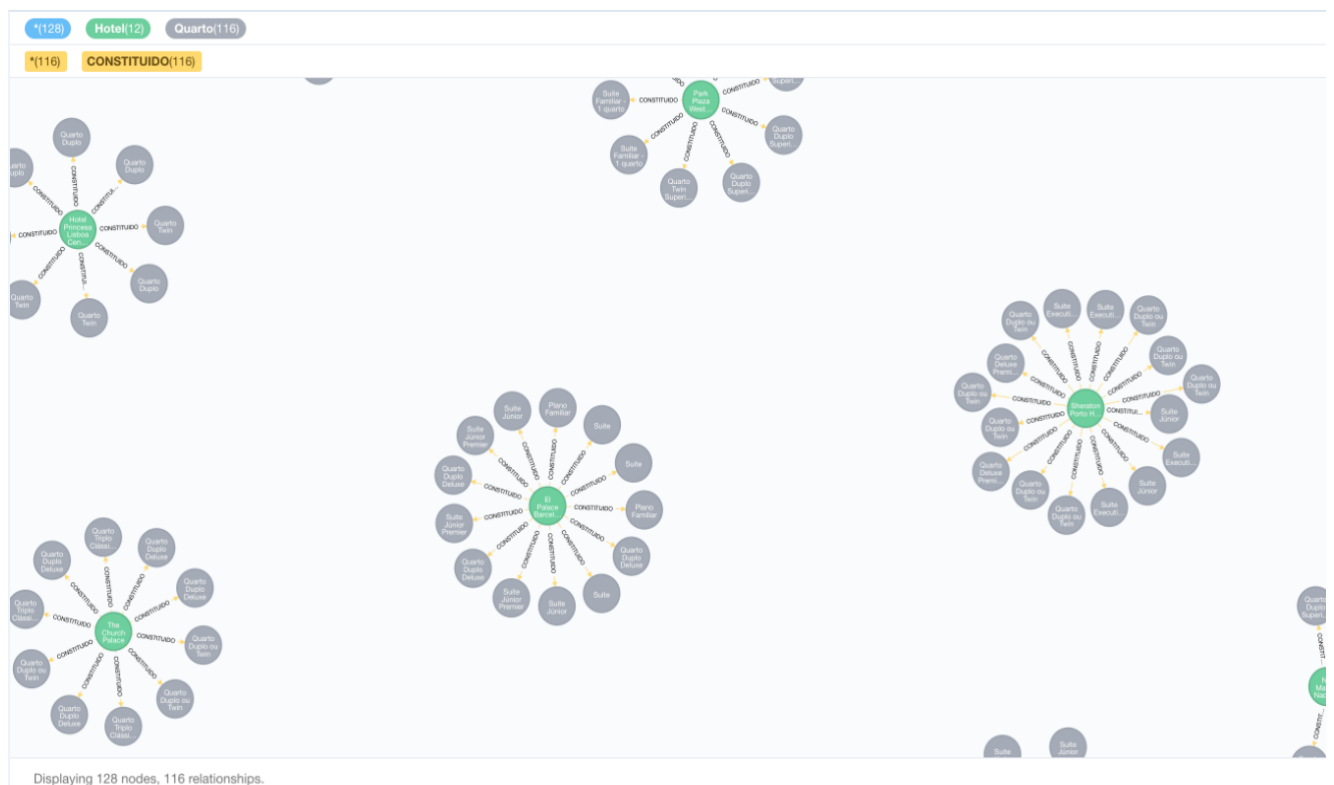


Figura 30- Neo4j- Relacionamento entre Cartao_Cidadao e Cliente

3.4.4. Hotel – Reserva

```
MATCH(h:Hotel),(r:Reserva)
WHERE h.id_Hotel = r.Hotel_id_Hotel
CREATE (h)-[:POSSUI]->(r);
```

Figura 31- Relacionamento entre Cartao_Cidadao e Cliente

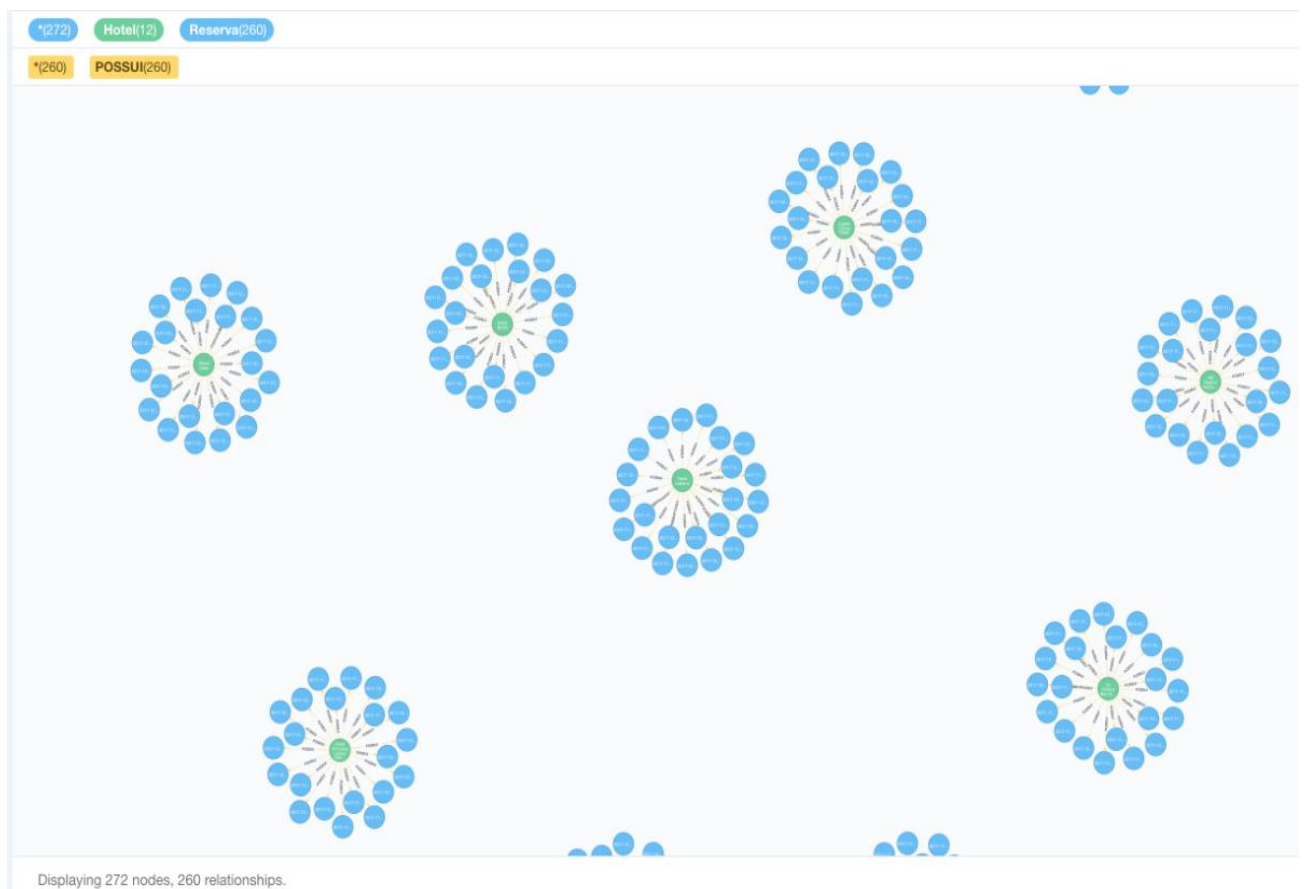


Figura 32- Neo4j- Relacionamento entre Cartao_Cidadao e Cliente

3.4.5. Morada – Cliente

```
MATCH(m:Morada),(c:Cliente)
WHERE m.id_Morada = c.Morada_id_Morada
CREATE (c)-[:MORA]->(m);
```

Figura 33- Relacionamento entre Cartao_Cidadao e Cliente

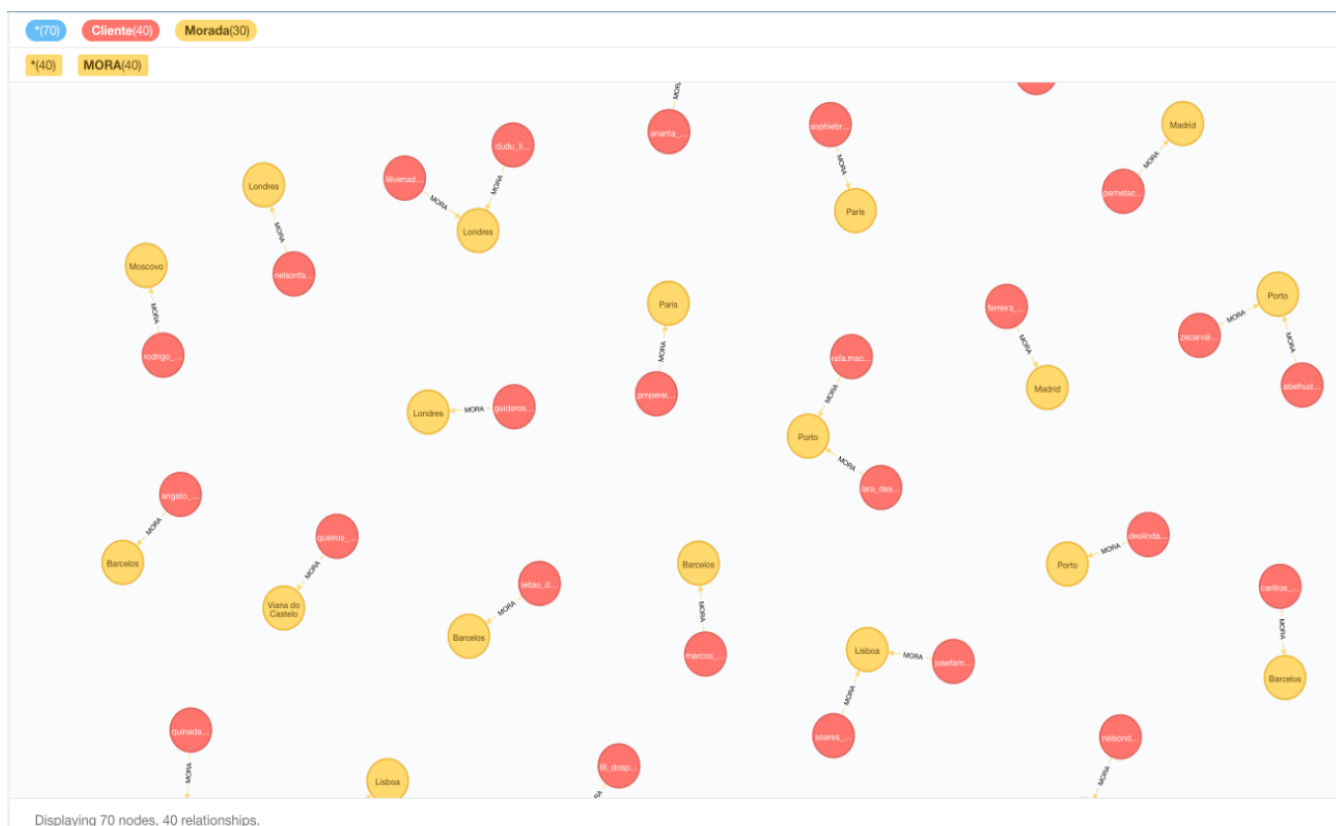


Figura 34- Neo4j- Relacionamento entre Cartao_Cidadao e Cliente

3.4.6. Reserva – Tipo_Reserva

```
MATCH(r:Reserva),(tr:Tipo_Reserva)
WHERE r.id_Reserva = tr.Reserva_id_Reserva
CREATE (r)-[:DEFINIDA]->(tr);
```

Figura 35- Relacionamento entre Cartao_Cidadao e Cliente

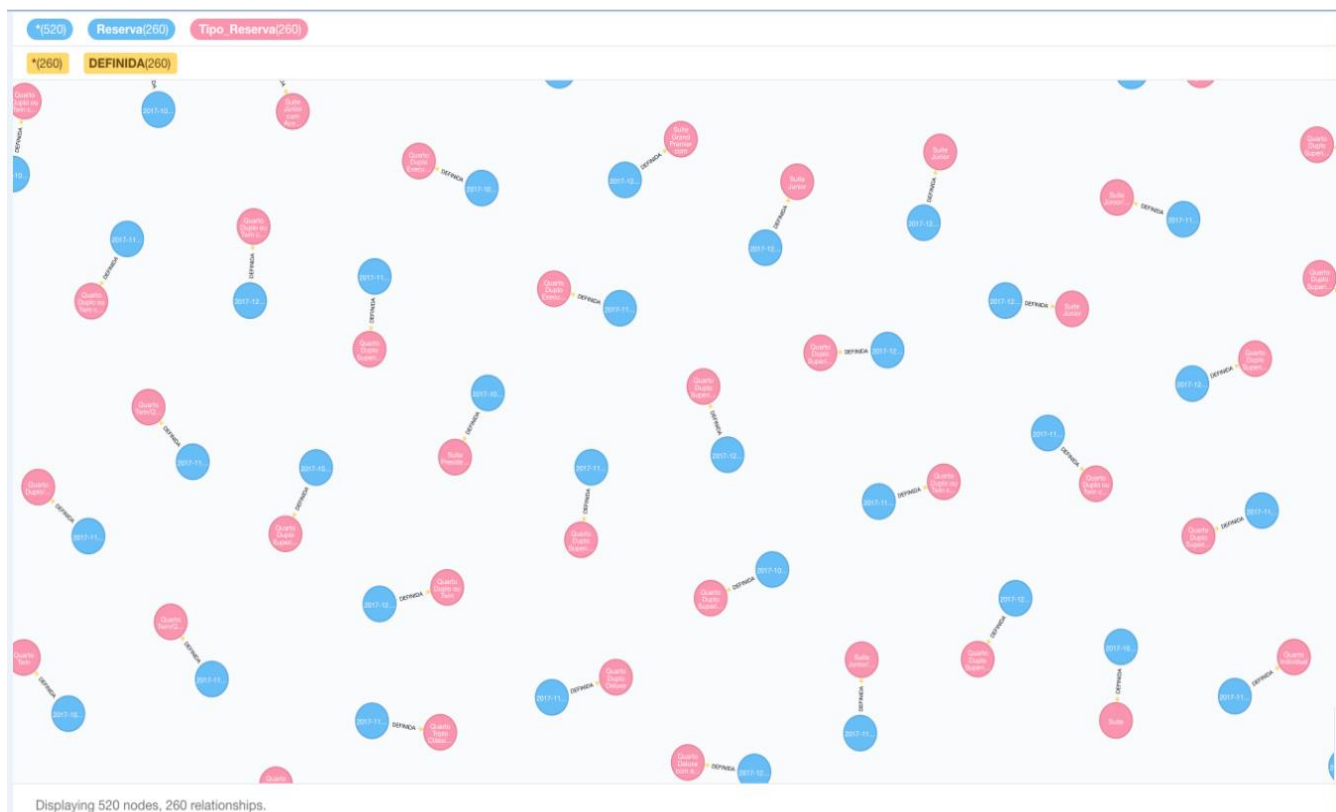


Figura 36- Neo4j- Relacionamento entre Cartao_Cidadao e Cliente

3.5. Modelo Neo4j

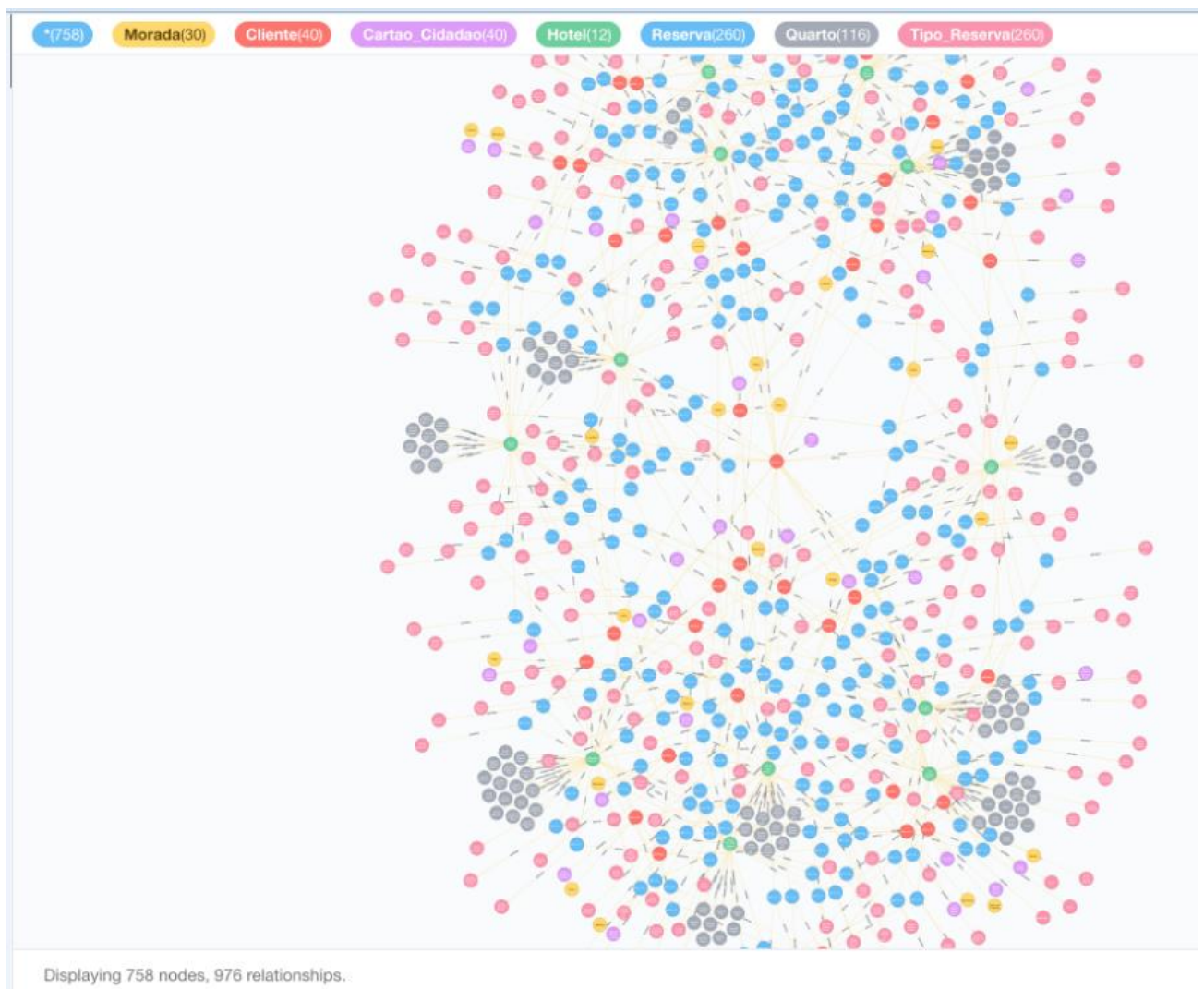
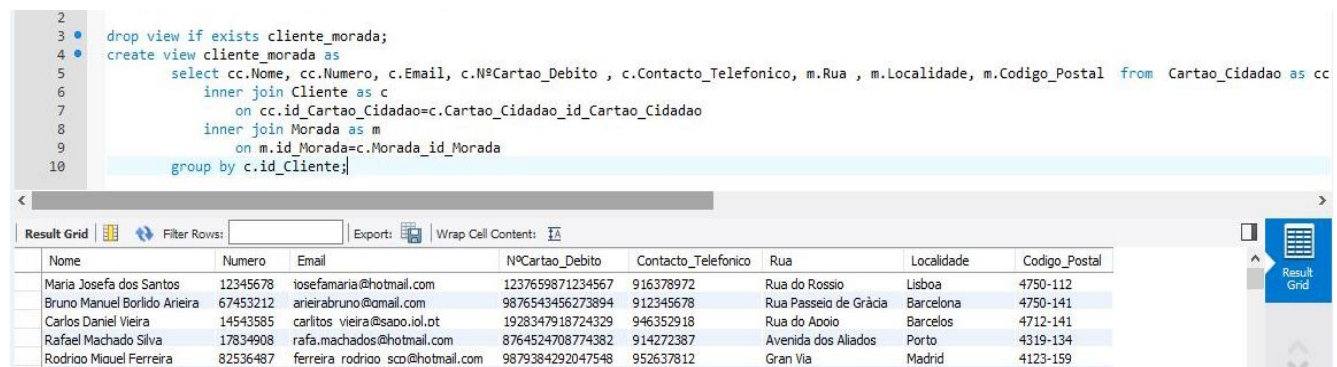


Figura 37- Modelo Neo4j

4. Queries

De modo a demonstrar a operacionalidade do sistema implementado, serão exemplificadas de seguida um conjunto de *queries*, utilizando a linguagem de interrogação do Neo4j (*cypher*). Para além disso, apresentámos as mesmas *queries* no modelo relacional, utilizando o MySQL, para que possa ser feita a comparação.

4.1. Apresenta a informação completa do cliente (*querie 1*)

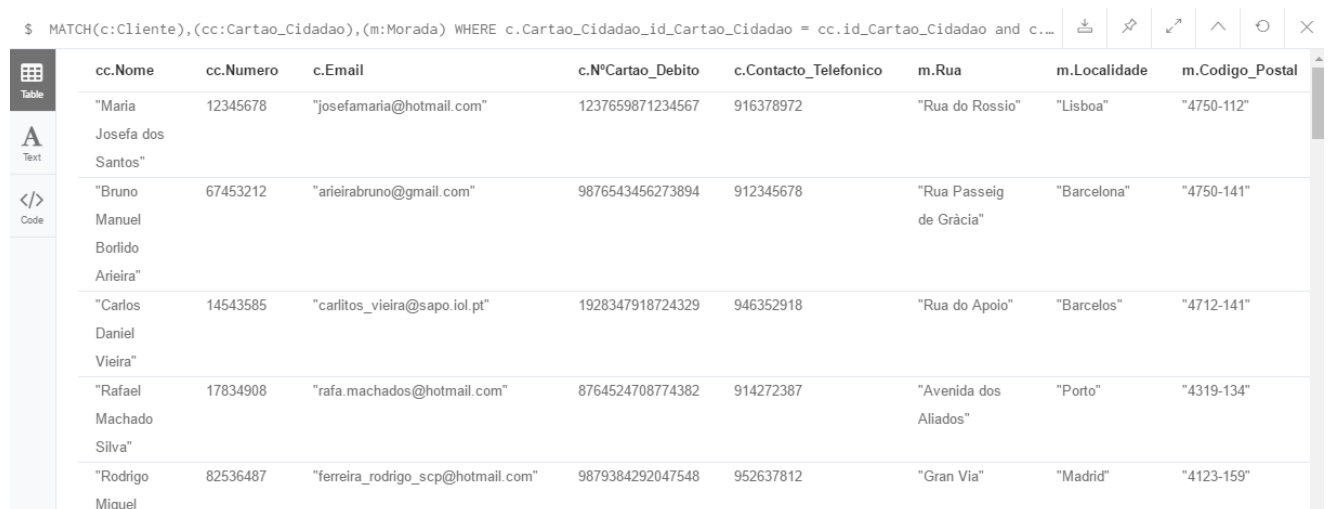


```
2
3 drop view if exists cliente_morada;
4 create view cliente_morada as
5     select cc.Nome, cc.Numero, c.Email, c.NºCartao_Debito, c.Contacto_Telefonico, m.Rua, m.Localidade, m.Codigo_Postal from Cartao_Cidadao as cc
6         inner join Cliente as c
7             on cc.id_Cartao_Cidadao=c.Cartao_Cidadao_id_Cartao_Cidadao
8         inner join Morada as m
9             on m.id_Morada=c.Morada_id_Morada
10    group by c.id_Cliente;
```

Nome	Numero	Email	NºCartao_Debito	Contacto_Telefonico	Rua	Localidade	Codigo_Postal
Maria Josefa dos Santos	12345678	josefamaria@hotmail.com	1237659871234567	916378972	Rua do Rossio	Lisboa	4750-112
Bruno Manuel Borlido Arieira	67453212	arieirabruno@gmail.com	9876543456273894	912345678	Rua Passeio de Grácia	Barcelona	4750-141
Carlos Daniel Vieira	14543585	carlitos_vieira@sapo.iol.pt	1928347918724329	946352918	Rua do Apoio	Barcelos	4712-141
Rafael Machado Silva	17834908	rafa.machados@hotmail.com	8764524708774382	914272387	Avenida dos Aliados	Porto	4319-134
Rodrioo Miquel Ferreira	82536487	ferreira_rodrigo_scp@hotmail.com	9879384292047548	952637812	Gran Via	Madrid	4123-159

Figura 38- Querie 1 em SQL

```
1 //Query 1 (Informação completa de cliente)
2
3 MATCH(c:Cliente),(cc:Cartao_Cidadao),(m:Morada)
4 WHERE c.Cartao_Cidadao_id_Cartao_Cidadao = cc.id_Cartao_Cidadao and c.Morada_id_Morada = m.id_Morada
5 Return cc.Nome, cc.Numero, c.Email, c.NºCartao_Debito, c.Contacto_Telefonico, m.Rua, m.Localidade,
   m.Codigo_Postal ;
```



```
$ MATCH(c:Cliente),(cc:Cartao_Cidadao),(m:Morada) WHERE c.Cartao_Cidadao_id_Cartao_Cidadao = cc.id_Cartao_Cidadao and c...
```

cc.Nome	cc.Numero	c.Email	c.NºCartao_Debito	c.Contacto_Telefonico	m.Rua	m.Localidade	m.Codigo_Postal
"Maria Josefa dos Santos"	12345678	"josefamaria@hotmail.com"	1237659871234567	916378972	"Rua do Rossio"	"Lisboa"	"4750-112"
"Bruno Manuel Borlido Arieira"	67453212	"arieirabruno@gmail.com"	9876543456273894	912345678	"Rua Passeig de Grácia"	"Barcelona"	"4750-141"
"Carlos Daniel Vieira"	14543585	"carlitos_vieira@sapo.iol.pt"	1928347918724329	946352918	"Rua do Apoio"	"Barcelos"	"4712-141"
"Rafael Machado Silva"	17834908	"rafa.machados@hotmail.com"	8764524708774382	914272387	"Avenida dos Aliados"	"Porto"	"4319-134"
"Rodrigo Miquel Ferreira"	82536487	"ferreira_rodrigo_scp@hotmail.com"	9879384292047548	952637812	"Gran Via"	"Madrid"	"4123-159"

Figura 39- Querie 1 em Neo4j

4.2. Total gasto por cada cliente (querie 2)

```

2
3 drop view if exists total_gasto;
4 create view total_gasto as
5     select cc.Nome, cc.Numero, c.Email, Sum(Preco) as Gasto from Cartao_Cidadao as cc
6         inner join Cliente as c
7             on cc.id_Cartao_Cidadao=c.Cartao_Cidadao_id_Cartao_Cidadao
8         inner join Reserva as r
9             on c.id_Cliente=r.Cliente_id_Cliente
10        group by c.id_Cliente;
11

```

Nome	Numero	Email	Gasto
João Carlos Perneta	33213218	pernas_carlos@gmail.com	5727.97
Cesario Miguel Perneta	31546548	pernetacesario@sapo.iol.pt	8640.00
Pedro Miguel Pereira	36925847	pmopereira@gmail.com	23723.50
Diooo Costa Bravo	67123110	queiros didi@yahoo.com	4275.05
Maria Joaquina Costa	12312331	ouinadascostas@gmail.com	8968.13
Rafael Machado Silva	17834908	rafa.machados@hotmail.com	31896.43

Figura 40- Querie 2 em SQL

```

1 //Query 2 (Total gasto por cada cliente)
2
3 MATCH (c:Cliente),(cc:Cartao_Cidadao),(r:Reserva)
4 WHERE c.Cartao_Cidadao_id_Cartao_Cidadao = cc.id_Cartao_Cidadao and r.Cliente_id_Cliente=c.id_Cliente
5 Return cc.Nome, cc.Numero, c.Email, Sum(r.Preco) as Gasto;

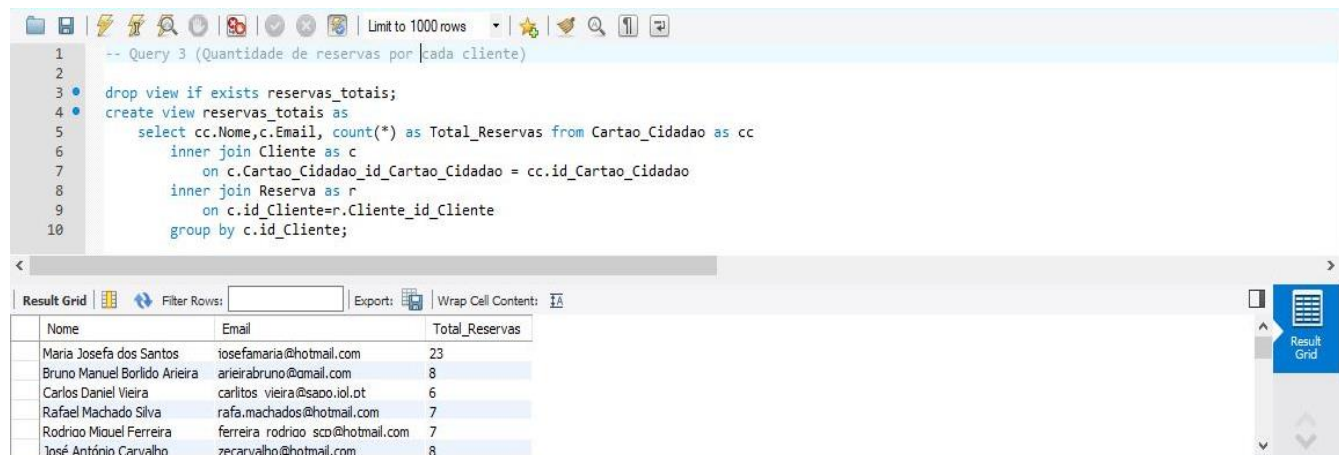
```

\$ MATCH (c:Cliente),(cc:Cartao_Cidadao),(r:Reserva) WHERE c.Cartao_Cidadao_id_Cartao_Cidadao = cc.id_Cartao_Cidadao and ...

cc.Nome	cc.Numero	c.Email	Gasto
"Cesario Miguel Perneta"	31546548	"pernetacesario@sapo.iol.pt"	8640
"Liliana Fernandes Venade"	21657843	"lilivenade@hotmail.com"	32056.4999999999996
"Pedro Costa Carvalho"	36312327	"pedro_costado@sapo.iol.pt"	4285.02
"Diogo Silva Meira"	32145698	"meiradioguinho@gmail.com"	5393.17
"Rosa Margarida Arieira Silva"	75213128	"guidinha_arias@hotmail.com"	2743.41
"Lara Deolinda Silvério"	81233887	"lara_desfado@hotmail.com"	10595.2200000000001
"Ana Rita Faria"	94381675	"anarita_parente@yahoo.com"	14076.25
"João Carlos Perneta"	33213218	"pernas_carlos@gmail.com"	5727.9699999999999
"Rafael Machado Silva"	17834908	"rafa.machados@hotmail.com"	31896.4299999999993
"Liliana Costa Venade"	21315444	"lili_dospresuntos@hotmail.com"	5311.01
"Rosa Margarida Leitão"	75391568	"guidarosa@hotmail.com"	14702.08
"Nelson Arieira Parente"	19897864	"nelsonparente@gmail.com"	15057.2699999999999
"Maria Joao Soares"	85269341	"mariajsoars14@hotmail.com"	12305.95
"Diogo Costa Bravo"	67123110	"queiros_didi@yahoo.com"	4275.05
"Sofia Palmeira Ferreira"	72313296	"palmeira_sofia@sapo.iol.pt"	13767.2

Figura 41- Querie 2 em Neo4j

4.3. Quantidade de reservas por cada cliente (querie 3)



The screenshot shows a SQL query in the 'Query 3 (Quantidade de reservas por cada cliente)' window. The query is as follows:

```
-- Query 3 (Quantidade de reservas por cada cliente)
1
2
3 drop view if exists reservas_totais;
4 create view reservas_totais as
5     select cc.Nome,c.Email, count(*) as Total_Reservas from Cartao_Cidadao as cc
6         inner join Cliente as c
7             on c.Cartao_Cidadao_id_Cartao_Cidadao = cc.id_Cartao_Cidadao
8         inner join Reserva as r
9             on c.id_Cliente=r.Cliente_id_Cliente
10        group by c.id_Cliente;
```

Below the query, the 'Result Grid' shows the following data:

Nome	Email	Total_Reservas
Maria Josefa dos Santos	iosefamaria@hotmail.com	23
Bruno Manuel Borlido Arieira	arieirabruno@gmail.com	8
Carlos Daniel Vieira	carlitos_vieira@sapo.iol.pt	6
Rafael Machado Silva	rafa.machados@hotmail.com	7
Rodrioo Miquel Ferreira	ferreira_rodrioo_scp@hotmail.com	7
José António Carvalho	zecarvalho@hotmail.com	8

Figura 42- Querie 3 em SQL

```
1 //Query 3 (Quantidade de reservas por cada cliente)
2
3 MATCH (cc:Cartao_Cidadao),(c:Cliente)-[:EFETUA]->(r:Reserva)
4 WHERE c.Cartao_Cidadao_id_Cartao_Cidadao = cc.id_Cartao_Cidadao and
5        r.Cliente_id_Cliente=c.id_Cliente
6 Return cc.Nome, c.Email, Count(*) as NrReservas;
```



The screenshot shows a Cypher query in the Neo4j interface. The query is as follows:

```
$ MATCH (c:Cliente),(cc:Cartao_Cidadao),(r:Reserva) WHERE c.Cartao_Cidadao_id_Cart...
```

Below the query, the 'Table' view shows the following data:

cc.Nome	c.Email	NrReservas
"Tiago Fernandes Duarte"	"soares_dos_frangos@gmail.com"	4
"Diogo Meira Neves"	"neves_diogo@hotmail.com"	6
"Bruno Abelha Costa"	"abelhudo123@gmail.com"	5
"Clara Deolinda Silvério"	"deolinda_desfado@yahoo.com"	6
"Diogo Costa Bravo"	"queiros_didi@yahoo.com"	5
"Carlos Daniel Leitão"	"leitao_daniel@yahoo.com"	6
"Maria Josefa dos Santos"	"josefamaria@hotmail.com"	23
"Maria Joaquina Costa"	"quinadascostas@gmail.com"	10
"Marcos Arigato"	"arigato_marquicos@sapo.iol.pt"	6
"Maria Joana Costa"	"joaninho.maria@yahoo.com"	6
"Rosa Margarida Arieira Silva"	"guidinha_arias@hotmail.com"	5
"Sofia Palmeira Ferreira"	"palmeira_sofia@sapo.iol.pt"	6
"Ana Rita Campos"	"ana.vairita@sapo.iol.pt"	5
"Sofia Novais Bravo"	"sophiebravo@sapo.iol.pt"	8
"José António Carvalho"	"zecarvalho@hotmail.com"	8

Figura 43- Querie 3 em Neo4j

4.4. Faturação por cada hotel (querie 4)

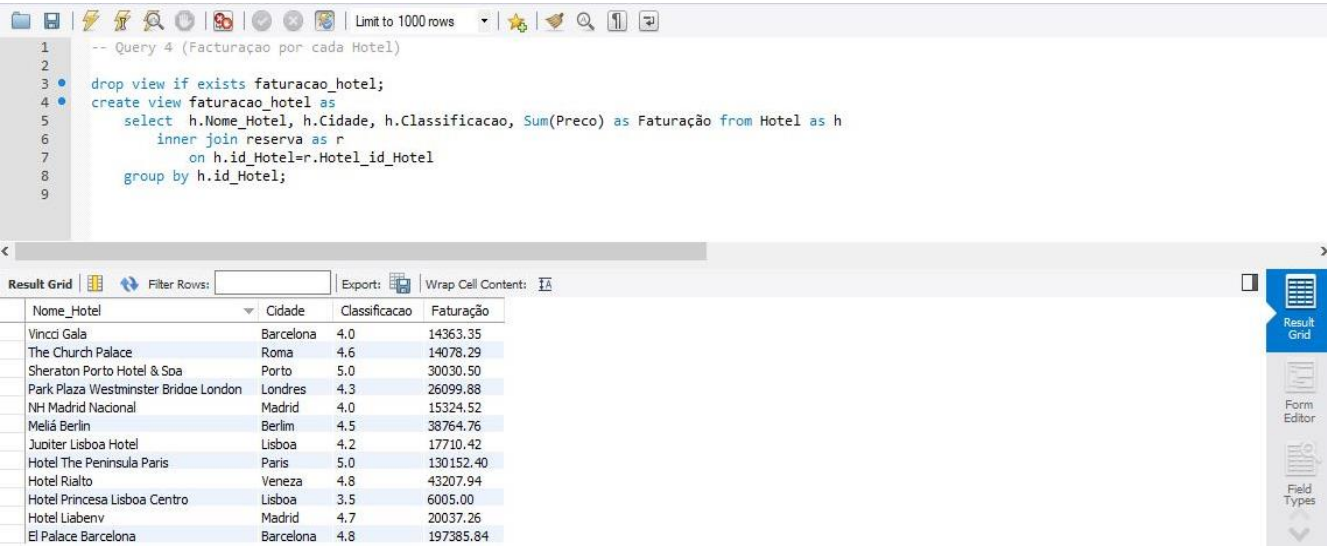


Figura 44- Querie 4 em SQL

```
1 // Query 4 (Facturação por cada Hotel)
2
3 MATCH (h:Hotel),(r:Reserva)
4 WHERE h.id_Hotel = r.Hotel_id_Hotel
5 Return h.Nome_Hotel, h.Cidade, h.Classificacao, Sum(r.Preco) as Faturacao;
```

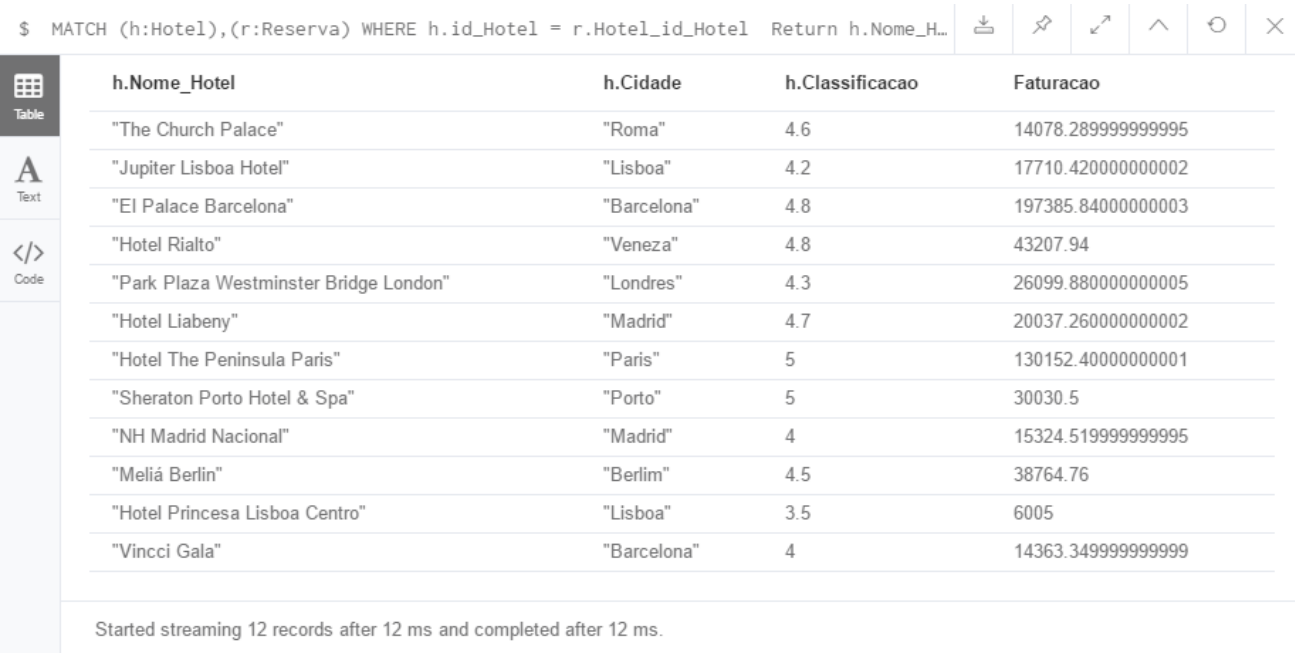
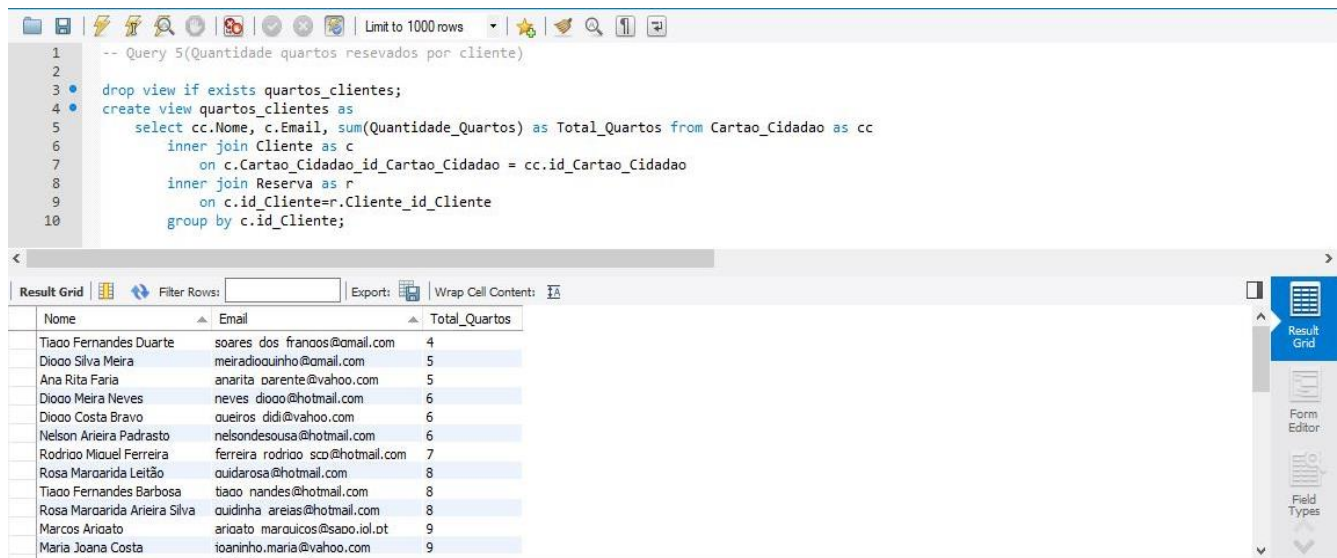


Figura 45- Querie 4 em Neo4j

4.5. Quantidade de quartos reservados por um cliente (querie 5)



```

1  -- Query 5(Quantidade quartos resevados por cliente)
2
3  drop view if exists quartos_clientes;
4  create view quartos_clientes as
5      select cc.Nome, c.Email, sum(Quantidade Quartos) as Total Quartos from Cartao_Cidadao as cc
6          inner join Cliente as c
7              on c.Cartao_Cidadao_id_Cartao_Cidadao = cc.id_Cartao_Cidadao
8          inner join Reserva as r
9              on c.id_Cliente=r.Cliente_id_Cliente
10         group by c.id_Cliente;

```

Nome	Email	Total Quartos
Tiago Fernandes Duarte	soares dos franoos@gmail.com	4
Diogo Silva Meira	meiradiouinho@gmail.com	5
Ana Rita Faria	anarita_parente@yahoo.com	5
Diogo Meira Neves	neves_diogo@hotmail.com	6
Diogo Costa Bravo	queiros_didi@yahoo.com	6
Nelson Arieira Padrao	nelsondesousa@hotmail.com	6
Rodrio Miquel Ferreira	ferreira_rodrio_scp@hotmail.com	7
Rosa Maroarida Leitão	quidarosa@hotmail.com	8
Tiago Fernandes Barbosa	tiao_nandes@hotmail.com	8
Rosa Maroarida Arieira Silva	quidinha_arias@hotmail.com	8
Marcos Arioato	arioato_marquicos@sapo.iol.pt	9
Maria Joana Costa	ioaninho.maria@yahoo.com	9

Figura 46- Querie 5 em SQL

```

1 // Query 5 (Quantidade quartos reservados por cliente)
2
3 MATCH (cc:Cartao_Cidadao),(c:Cliente) - [:EFETUA] -> (r:Reserva)
4 WHERE cc.id_Cartao_Cidadao = c.Cartao_Cidadao_id_Cartao_Cidadao and
5       c.id_Cliente = r.Cliente_id_Cliente
6 Return cc.Nome, c.Email, Sum(r.Quantidade Quartos) as Total Quartos;

```

\$ MATCH (cc:Cartao_Cidadao),(c:Cliente) - [:EFETUA] -> (r:Reserva) WHERE cc.id_Car...

cc.Nome	c.Email	Total Quartos
"Tiago Fernandes Duarte"	"soares_dos_frangos@gmail.com"	4
"Diogo Meira Neves"	"neves_diogo@hotmail.com"	6
"Bruno Abelha Costa"	"abelhudo123@gmail.com"	11
"Clara Deolinda Silvério"	"deolinda_desfado@yahoo.com"	16
"Diogo Costa Bravo"	"queiros_didi@yahoo.com"	6
"Carlos Daniel Leitão"	"leitao_daniel@yahoo.com"	10
"Maria Josefa dos Santos"	"josefamaria@hotmail.com"	38
"Maria Joaquina Costa"	"quinadascostas@gmail.com"	13
"Marcos Arigato"	"arigato_marquicos@sapo.iol.pt"	9
"Maria Joana Costa"	"joaninho.maria@yahoo.com"	9
"Rosa Margarida Arieira Silva"	"guidinha_arias@hotmail.com"	8
"Sofia Palmeira Ferreira"	"palmeira_sofia@sapo.iol.pt"	13
"Ana Rita Campos"	"ana.vairita@sapo.iol.pt"	9
"Sofia Novais Bravo"	"sophiebravo@sapo.iol.pt"	20

Figura 47- Querie 5 em Neo4j

4.6. Tipo de reserva por cliente (querie 6)

```

1  -- Query 6 (Tipo de reserva por cliente)
2
3  drop view if exists tipo_reserva_cliente;
4  create view tipo_reserva_cliente as
5      select r.id_Reserva, cc.Nome, c.Email, h.Nome_Hotel, tr.Tipo_Quarto, tr.Opcao_Estadia from Tipo_Reserva as tr
6          inner join Reserva as r
7              on tr.id_Tipo_Reserva=r.id_Reserva
8          inner join Hotel as h
9              on h.id_Hotel=r.Hotel_id_Hotel
10         inner join Cliente as c
11             on c.id_Cliente = r.Cliente_id_Cliente
12         inner join Cartao_Cidadao as cc
13             on cc.id_Cartao_Cidadao=c.Cartao_Cidadao_id_Cartao_Cidadao
14     group by r.id_Reserva;

```

id_Reserva	Nome	Email	Nome_Hotel	Tipo_Quarto	Opcao_Estadia
13	Maria Josefa dos Santos	josefamaría@hotmail.com	The Church Palace	Quarto Triplo Clássico	2
14	Maria Josefa dos Santos	josefamaría@hotmail.com	Hotel Rialto	Apartamento de luxo com Quarto e Vista	3
15	Maria Josefa dos Santos	josefamaría@hotmail.com	Hotel Liabenv	Oferta Romântica	3
16	Maria Josefa dos Santos	josefamaría@hotmail.com	NH Madrid Nacional	Quarto Duplo Superior com Vista/Quarto Duplo ...	1/1/1
17	Maria Josefa dos Santos	josefamaría@hotmail.com	Juiter Lisboa Hotel	Quarto Duplo Superior/Quarto Duplo Superior/O...	1/1/1
18	Maria Josefa dos Santos	josefamaría@hotmail.com	Sheraton Porto Hotel & Soa	Quarto Duplo ou Twin Deluxe /Quarto Deluxe Pr...	2/2/2
19	Maria Josefa dos Santos	josefamaría@hotmail.com	Hotel The Peninsula Paris	Quarto Deluxe com acesso ao Spa e Acesso Wi...	3/3
20	Maria Josefa dos Santos	josefamaría@hotmail.com	Park Plaza Westminster Bridge London	Quarto Duplo Superior com Vista Interna/Quart...	1/1/1
21	Maria Josefa dos Santos	josefamaría@hotmail.com	Melá Berlin	Suite Presidencial	3
22	Maria Josefa dos Santos	josefamaría@hotmail.com	Vinci Gala	Quarto Duplo Superior	1
23	Bruno Manuel Borlido Ar...	arieirabruno@gmail.com	El Palace Barcelona	Quarto Duplo Deluxe	3
24	Carlos Daniel Vieira	carlitos_vieira@sapo.iol.pt	El Palace Barcelona	Suite Junior/Quarto Duplo Deluxe	3/3

Figura 48- Querie 6 em SQL

```

1  // Query 6 (Tipo de reserva por cliente)
2
3  MATCH (cc:Cartao_Cidadao), (c:Cliente) - [:EFETUA] -> (r:Reserva), (h:Hotel), (tr:Tipo_Reserva)
4  Where cc.id_Cartao_Cidadao = c.Cartao_Cidadao_id_Cartao_Cidadao
5        and h.id_Hotel = r.Hotel_id_Hotel
6        and r.id_Reserva = tr.Reserva_id_Reserva
7  Return r.id_Reserva, cc.Nome, c.Email, h.Nome_Hotel, tr.Tipo_Quarto, tr.Opcao_Estadia;

```

\$ MATCH (cc:Cartao_Cidadao), (c:Cliente) - [:EFETUA] -> (r:Reserva), (h:Hotel), (tr:Tipo_Reserva) Where cc.id_Cartao_Cid...

r.id_Reserva	cc.Nome	c.Email	h.Nome_Hotel	tr.Tipo_Quarto	tr.Opcao_Estadia
195	"Maria Josefa dos Santos"	"josefamaría@hotmail.com"	"Vinci Gala"	"Quarto Duplo Superior"	"2"
18	"Maria Josefa dos Santos"	"josefamaría@hotmail.com"	"Sheraton Porto Hotel & Spa"	"Quarto Duplo ou Twin Deluxe /Quarto Deluxe Premium/Quarto Duplo ou Twin Clube com acesso ao Salão"	"2/2/2"
17	"Maria Josefa dos Santos"	"josefamaría@hotmail.com"	"Jupiter Lisboa Hotel"	"Quarto Duplo Superior/Quarto Duplo Superior/Quarto Duplo Superior"	"1/1/1"
20	"Maria Josefa dos Santos"	"josefamaría@hotmail.com"	"Park Plaza Westminster Bridge London"	"Quarto Duplo Superior com Vista Interna/Quarto Duplo Superior com Vista Interna/Quarto Duplo Superior com Vista Interna"	"1/1/1"
19	"Maria Josefa dos Santos"	"josefamaría@hotmail.com"	"Hotel The Peninsula"	"Quarto Deluxe com acesso ao Spa e Acesso Wi-Fi Gratuito/Quarto Deluxe com acesso ao Spa e Acesso Wi-Fi"	"3/3"

Figura 49- Querie 6 em Neo4j

5. Conclusão

Este trabalho prático foi bastante importante e enriquecedor, pois permitiu-nos ter um contacto mais direto com a compreensão e execução de projetos de sistemas de bases de dados não relacionais.

Inicialmente foi necessário observar o sistema relacional que tínhamos construído anteriormente, para identificar todas as entidades e atributos, de maneira a compreender o comportamento da base de dados. Assim, conseguimos ter uma noção mais exata do que era necessário para a sua conversão numa base de dados não relacional.

De seguida, foi definido todo o processo de migração dos dados do sistema relacional para o não relacional, usando os ficheiros “.csv” e usando a linguagem de interrogação do Neo4j (cypher).

Por último, foram implementadas e exemplificadas algumas *queries*, que já tinham sido implementadas no sistema relacional, de modo a demonstrar a operacionalidade e execução do sistema, comparando os resultados das mesmas. Neste passo, ocorreu um pequeno problema para o qual não conseguimos encontrar resolução, mas que achámos não ter uma grande influência nos objetivos do projeto prático: as *queries* apresentam os mesmos resultados, mas não com a mesma ordem de apresentação.

Em suma, podemos concluir que com a realização deste trabalho percebemos que um sistema de bases de dados não relacional é mais eficaz, rápido e flexível na organização de dados do que um sistema de bases de dados relacional, que foi usado na primeira parte do projeto.