



Universidade do Minho

Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2016/2017

Migração de Base de Dados Esteticção de SQL Server para Neo4j

Ana Rita Amorim a74556,

Cláudia Costa a68686,

Daniel Martins a73175,

Diogo Araújo a68695

Janeiro, 2017

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Migração de Base de Dados Estético de SQL Server para Neo4j

**Ana Rita Amorim a74556,
Cláudia Costa a68686,
Daniel Martins a73175,
Diogo Araújo a68695**

Janeiro, 2017

Resumo

Neste relatório está contida a informação relativa à migração do Sistema de Base de Dados de um Centro de Estética Canino, relativo à cadeia Estética, de um modelo de base de dados Relacional (SQL) para um modelo não Relacional (NoSQL). Esta migração surge da necessidade que o nosso cliente expôs de uma maior rapidez de consulta aos registos, daí a implementação de uma *Graph Database*.

Apresentamos ainda uma breve introdução ao Neo4j, o processo completo de migração assim como todo o código utilizado no processo.

Área de Aplicação: Migração de Bases de Dados

Palavras-Chave: SQL, SQL Server, Neo4j, Migração, Grafos, Nodos, Relações, NoSQL

Índice

1. Introdução	1
1.1. Contextualização	1
1.2. Motivação e Objectivos	1
1.3. Estrutura do Relatório	2
2. Bases de Dados não Relacionais: NoSQL	3
3. SQL vs NoSQL - Diferenças de Alto Nível	4
4. Graph Databases	5
4.1. O que são?	5
4.2. Constituintes	5
4.3. Vantagens	5
4.4. Neo4j	6
5. Processo de Migração	7
5.1. Fase 1	7
5.2. Fase 2	8
5.3. Fase 3	10
5.4. Fase 4	11
5.5. Fase 5	12
5.6. Fase 6	13
6. Conclusão	25
7. Bibliografia	26
8. Referências WWW	27

Índice de Figuras

Ilustração 2 - Modelo-base grafos	7
Ilustração 3 - Exportar tabelas csv	8
Ilustração 4 - Guardar tabelas csv	9
Ilustração 5 - Conjunto tabelas exportadas	10
Ilustração 6 - Cria nodo tabela	10
Ilustração 7 - Resultado criação nodo	11
Ilustração 8 - Criação de Relacionamentos	11
Ilustração 9 - Resultado criação Relacionamentos	12
Ilustração 10 - Caso NULL I	12
Ilustração 11 - Caso NULL II	12
Ilustração 12 - Caso NULL III	12
Ilustração 13 - Caso NULL IV	13
Ilustração 14 - Resultados Caso NULL	13
Ilustração 15 - Query nº1	13
Ilustração 16 - Resultado query nº1	14
Ilustração 17 - Query nº2	14
Ilustração 18 - Resultado query nº2	15
Ilustração 19 - Query nº3	15
Ilustração 20 - Resultado query nº3	15
Ilustração 21 - Query nº4	16
Ilustração 22 - Resultado query nº4	16
Ilustração 23 - Query nº5	16
Ilustração 24 - Resultado query nº5	17
Ilustração 25 - Query nº6	17
Ilustração 26 - Resultado query nº6	18
Ilustração 27 - Query nº 7	18
Ilustração 28 - Resultado query nº7	19
Ilustração 29 - Query nº8	19
Ilustração 30 - Resultado query nº8	20
Ilustração 31 - Query nº9	20

Ilustração 32 - Resultado query nº9	20
Ilustração 33 - Query nº10	21
Ilustração 34 - Resultado query nº10	21
Ilustração 35 - Query nº11	22
Ilustração 36 - Resultado query nº11	22
Ilustração 37 - Query nº12	22
Ilustração 38 - Resultado query nº12	23

1. Introdução

1.1. Contextualização

Começamos por recordar que a empresa Estéticão é uma empresa composta por uma cadeia de Centros de Estética para cães espalhada por Portugal. Com o aumento exponencial de clientes, o representante da empresa decidiu criar uma ferramenta de auxílio à gerência de informação das lojas, o que nos levou à criação de uma Base de Dados Relacional que a suportasse.

Recentemente, o cliente voltou a procurar os nossos serviços, expondo a necessidade de um acesso mais rápido aos registos. Daqui surgiu a ideia da passagem da base de dados Relacional para uma *Graph Database (NoSQL)*, isto é, base de dados não relacional.

Para a implementação da *Graph Database* tivemos como suporte de trabalho a ferramenta Neo4j.

1.2. Motivação e Objectivos

Uma vez que a empresa Estéticão se encontra em grande crescimento, surgiu a necessidade de criar uma base de dados que suportasse toda a informação. Contudo, o cliente expôs a necessidade de um acesso mais rápido aos registos, o que nos levou à migração da base de dados de Relacional para não Relacional.

Com a implementação da base de dados não Relacional, espera-se que o tempo de acesso aos registos na base de dados diminua significativamente e estima-se que o contacto do utilizador com a base de dados seja mais *friendly* (isto é, mais simples).

1.3. Estrutura do Relatório

Nos capítulos seguintes deste relatório estará contida toda a informação relativa à migração da base de dados, referente à empresa Estéticão, de relacional para não relacional.

Seguir-se-á, ordenadamente, os capítulos Bases de Dados não Relacionais: NoSQL, SQL vs NoSQL, Graph Databases, Processo de Migração, Conclusão,

2. Bases de Dados não Relacionais: NoSQL

O termo NoSQL (Not Only SQL) foi primeiramente utilizado em 1998 como nome de uma base de dados relacional que não possuía uma interface SQL. Mais tarde, no início de 2009, foi reintroduzido por Eric Evans, funcionário do *Rackspace*, quando Johan Oskarsson da *Last.fm* queria organizar um evento para discutir bases de dados *open source distribuídas*. NoSQL surge então de uma tentativa de descrever o aumento de número de bases de dados não relacionais que não tinham a preocupação de garantir propriedades ACID (atomicidade, consistência, isolamento e durabilidade).

As bases de dados que seguem estes padrões não podem exigir esquemas de tabela fixa, geralmente, não suportam instruções e operações de junção SQL e armazenam os dados com técnicas que visam utilizar escalabilidade horizontal.

Ao falarmos de NoSQL falamos de modelos não-Relacionais de dados sendo portanto introduzidas as *Graph Stores*, *Document Stores* e *Key-Value*.

3. SQL vs NoSQL – Diferenças de Alto Nível

- Bases de dados SQL são chamadas relacionais enquanto que bases de dados NoSQL são chamados de bases de dados não relacionais ou distribuídas;
- Bases de dados SQL representam os dados em forma de tabela enquanto que bases de dados NoSQL representam os dados em forma de documento, par chave-valor ou grafo, dependendo do modelo;
- Bases de dados SQL têm um esquema de dados predefinido enquanto que bases de dados NoSQL têm um esquema de dados dinâmico para dados não estruturados;
- Bases de dados SQL são verticalmente escaláveis enquanto que bases de dados NoSQL são horizontalmente escaláveis;
- Bases de dados SQL usam o SQL (*Structured Query Language*) para definir e manipular dados. Bases de dados NoSQL utilizam uma sintaxe UnQL (*Unstructured Query Language*) que varia conforme o modelo de base de dados;
- Bases de dados SQL estáveis e garantem a atomicidade assim como a integridade dos dados;
- Bases de dados SQL enfatizam propriedade ACID(atomicidade, consistência, isolamento e durabilidade), enquanto que bases de dados NoSQL seguem o teorema de CAP(Consistência, Disponibilidade e Tolerância de Partição).

4. Graph Databases

4.1. O que são?

Graph databases são bases de dados que usam estruturas de grafos para queries semânticas com nodos, relacionamentos e propriedades para representarem e armazenarem informação. O conceito chave do sistema é o grafo (ou relacionamento), o qual relaciona directamente os registos. Os relacionamentos permitem que os registos sejam marcados directamente e em conjunto, e em alguns casos recuperá-los com uma única operação.

O sistema é gerenciado por operações CRUD (Create, Read, Update e Delete), o que significa que a aplicação não terá de conter chaves estrangeiras (*foreign key*).

4.2. Constituintes

- Nodos – representam as entidades tais como pessoas, negócios, contas ou qualquer outra coisa acerca da qual queiramos continuar a ser informados.
- Relacionamentos – são as linhas que ligam os nodos uns aos outros. Representam as relações entre os nodos.
- Propriedades – são as informações pertinentes relativas aos nodos. Por exemplo, se a *Wikipedia* fosse um dos nodos, as propriedades ligadas poderiam ser *website* ou palavra que começa com a letra w ou outros aspectos, dependendo do que fosse pertinente para a base de dados em particular.

4.3. Vantagens

- A estrutura permite modelar todo o tipo de cenários – desde sistemas de estradas, historial médico da população ou qualquer coisa definida por relações;
- *Graph databases* foram construídas para serem usadas em sistemas transaccionais (OLTP) e foram engenhadas com integridade transaccional;

- Para trabalhar com grandes quantidades de informação, *graph databases* têm uma melhor performance a várias ordens de magnitude. A performance mantém-se constante mesmo com a informação crescendo de dia para dia;
- Permite, em vez de modelarem exaustivamente o domínio ao longo do tempo, adicionar informação à estrutura do grafo existente sem interferir com a funcionalidade corrente;
- A agilidade dos *graph databases* permite que qualquer *graph database* se envolva com o resto da aplicação sem requerer quaisquer mudanças.

4.4. Neo4j

Neo4j é um sistema de administração de *graph database* desenvolvido pela Neo Technology, Inc.

É descrita pelos desenvolvedores como sendo ACID e transaccional, com armazenamento e processamento nativo.

Neo4j é a *graph database* mais popular do mercado, segundo o db-engines.com.

5. Processo de Migração

5.1. Fase 1

Primeiramente, numa reunião com os responsáveis da empresa Estético, desenvolvemos um modelo-base em forma de grafo, a partir do qual nos baseamos na migração para Neo4j.

Segue-se esse mesmo modelo.

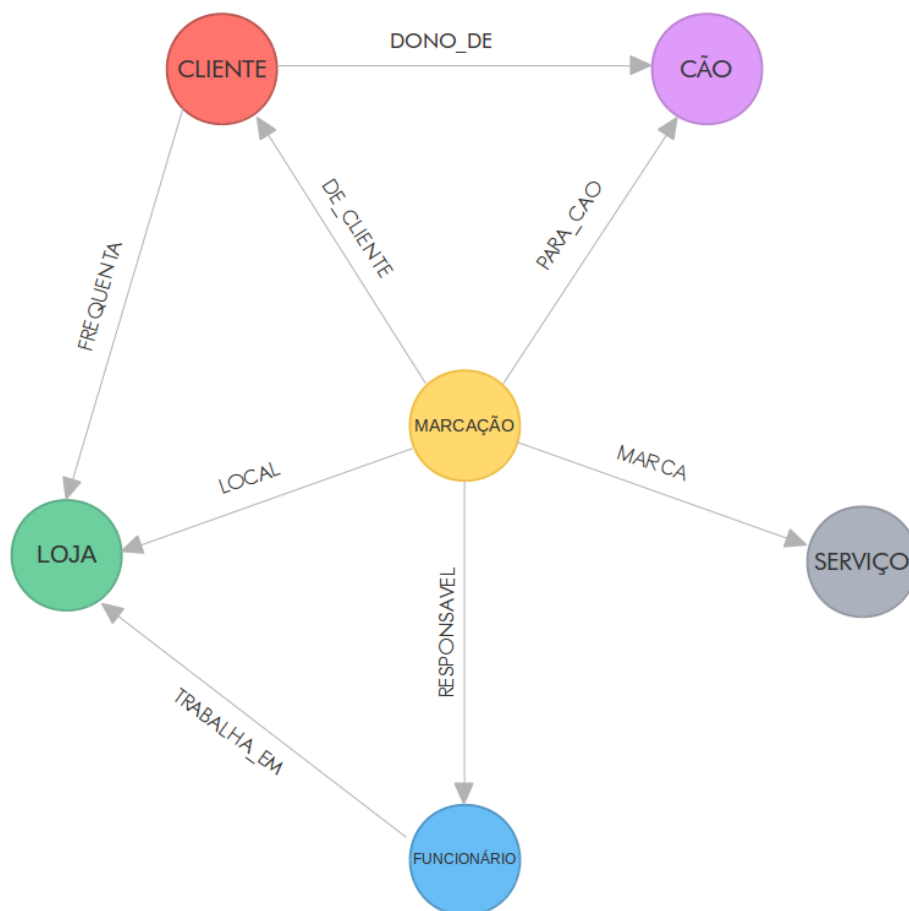


Ilustração 2 - Modelo-base grafos

5.2. Fase 2

Nesta segunda fase começamos por gerar as tabelas no MySQL Workbench e exportá-las como ficheiros csv.

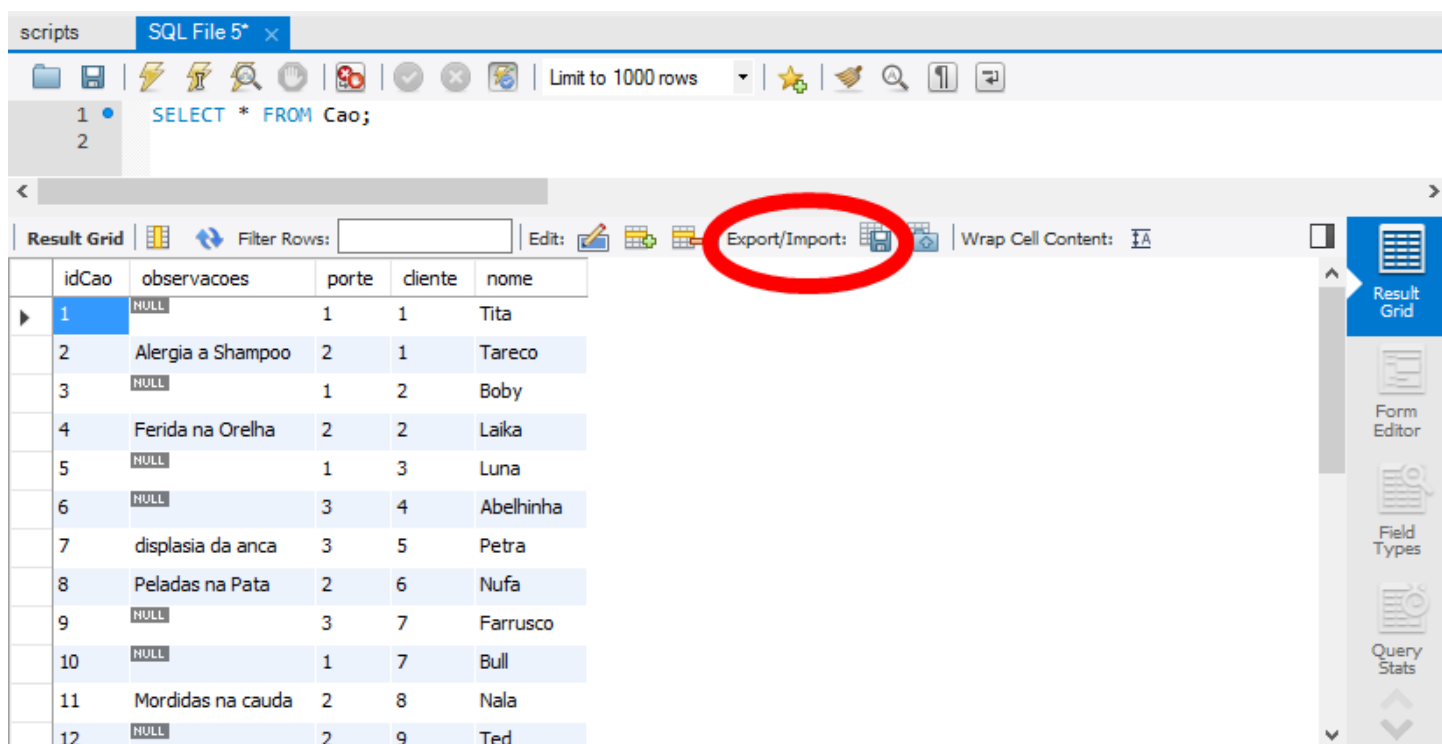


Ilustração 3 - Exportar tabelas csv

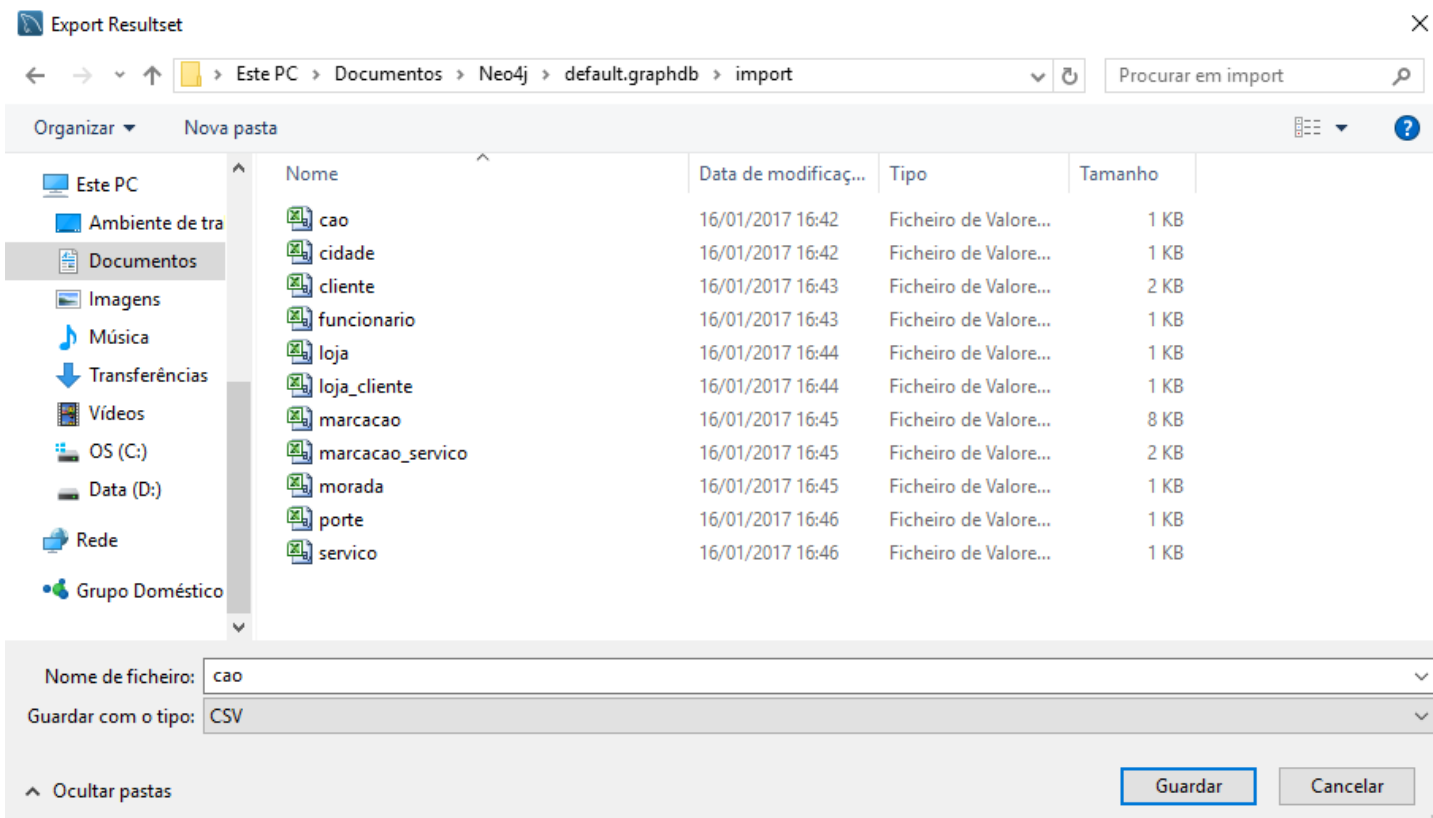


Ilustração 4 - Guardar tabelas csv

Note-se que repetimos o processo para todas as entidades pertencentes à base de dados anteriormente criada.

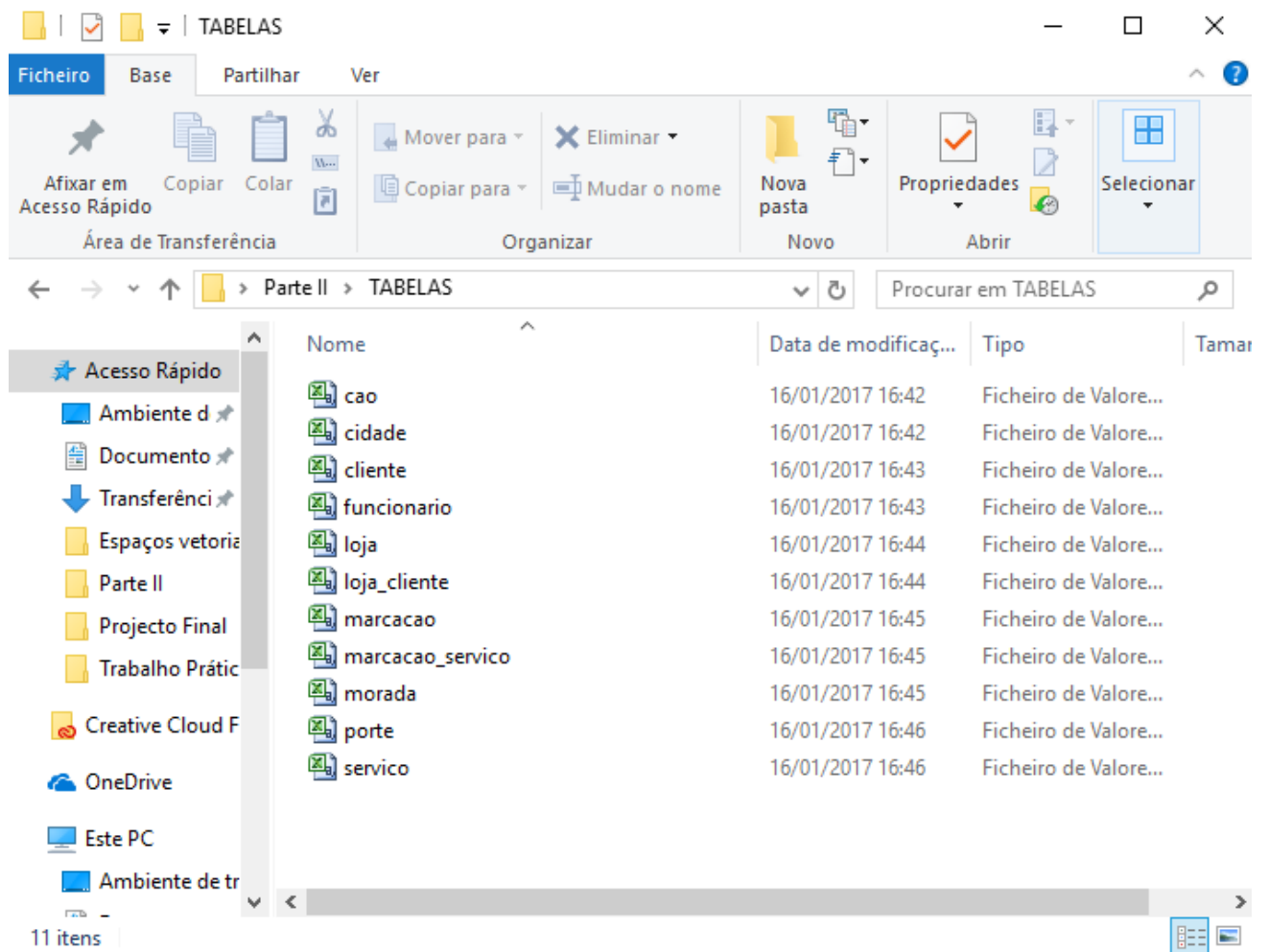


Ilustração 5 - Conjunto tabelas exportadas

5.3. Fase 3

Depois de termos exportado todas as tabelas, procedemos à criação dos nodos das mesmas em Neo4j.

```

1 LOAD CSV WITH HEADERS FROM 'file:///cao.csv' AS line
2 CREATE (cao:Cao {id: TOINTEGER(line.CaoID) })
3 SET cao.nome = line.CaoNome,
4     cao.porte = line.CaoPorte,
5     cao.observacoes = line.CaoObservacoes,
6     cao.cliente = line.CaoCliente
7 RETURN cao:

```

Ilustração 6 - Cria nodo tabela

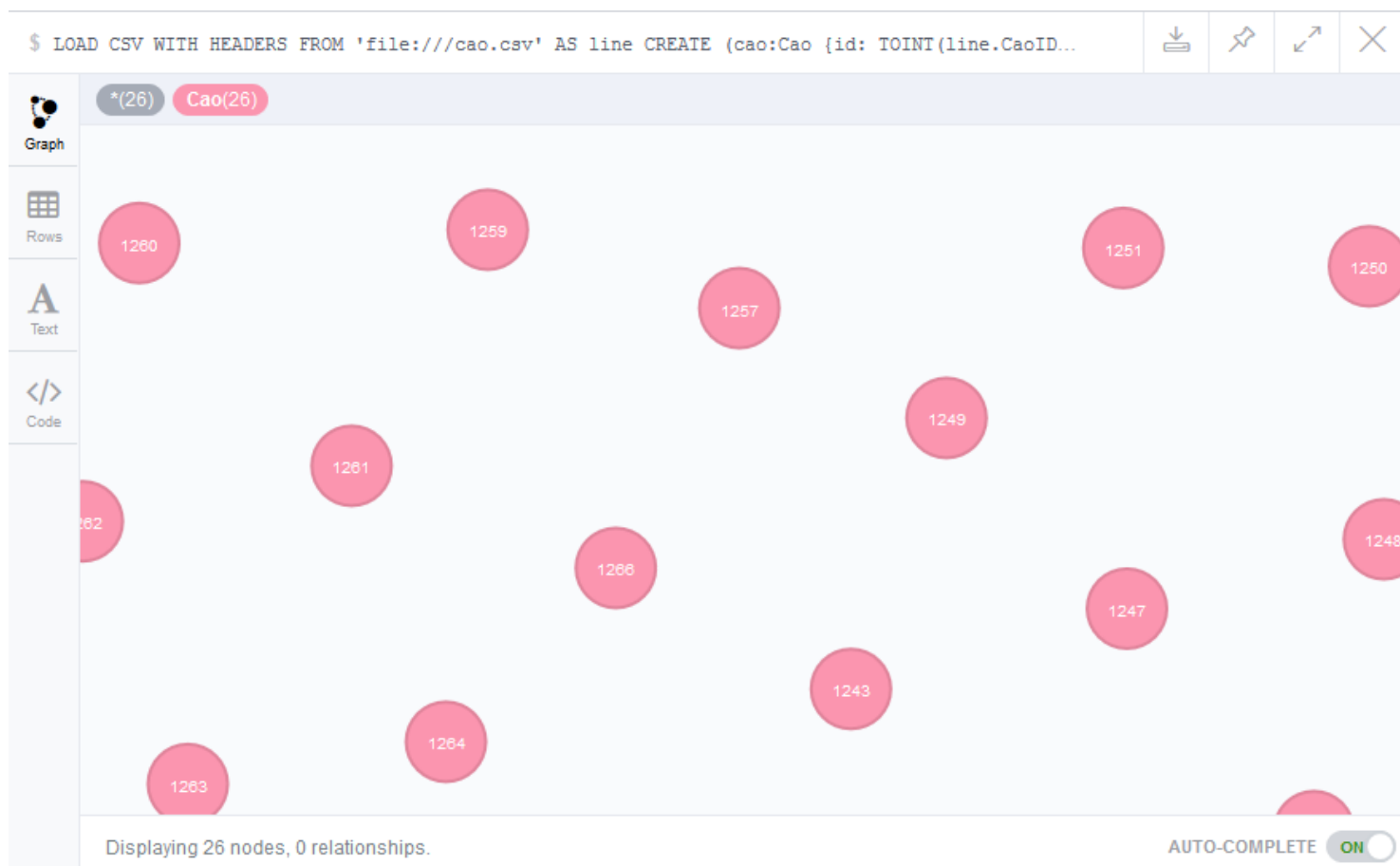


Ilustração 7 - Resultado criação nodo

Note-se que, mais uma vez, repetimos este mesmo processo para todas as tabelas anteriormente exportadas.

5.4. Fase 4

Numa fase 3 procedemos à criação dos relacionamentos entre os nodos, sendo estes os relacionamentos anteriormente criados na base de dados relacional.

```

2 LOAD CSV WITH HEADERS FROM 'file:///cao.csv' AS line
3 MATCH (cliente:Cliente{id:TOINT(line.cliente) })
4 MATCH (cao:Cao{id:TOINT(line.idCao)})
5 CREATE (cliente)-[:DONO_DE]->(cao)
6 RETURN cliente,cao;
7

```

Ilustração 8 - Criação de Relacionamentos

\$

LOAD CSV WITH HEADERS FROM 'file:///cao.csv' AS line MATCH (cliente:Cliente{id:TOINT(line...

Download

Share

Fullscreen

Close

Rows

(no rows)

Code

Returned 0 rows in 158 ms.

Ilustração 9 - Resultado criação relacionamentos

5.5. Fase 5

Aqui, removemos todas as propriedades do nodo que poderiam ser NULL.

Passamos a ilustrar os quatro casos em que isso era possível:

```
1 MATCH (cao:Cao{observacoes:'NULL'})
2 REMOVE cao.observacoes;
3
4
```



Ilustração 10 - Caso NULL I

```
1 MATCH (funcionario:Funcionario{email:'NULL'})
2 REMOVE funcionario.email;
3
4
```

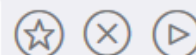


Ilustração 11 - Caso NULL II

```
1 MATCH (cliente:Cliente{nif:'NULL'})
2 REMOVE cliente.nif;
3
```



Ilustração 12 - Caso NULL III

```
1 MATCH (cliente:Cliente{email:'NULL'})
2 REMOVE cliente.email;
```

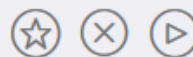


Ilustração 13 - Caso NULL IV

De todas as instruções acima, o resultado foi sempre o mesmo, sendo o registado na imagem abaixo.

```
$ MATCH (cao:Cao{observacoes:'NULL'}) REMOVE cao.observacoes;
```



Rows

(no changes, no rows)



Code

Returned 0 rows in 109 ms.

Ilustração 14 - Resultados Caso NULL

5.6. Fase 6

Nesta sexta, e última, fase procedemos à reestruturação das queries, que tínhamos antes criado em SQL, para a linguagem do Neo4j, Cypher.

- 🚦 Query que devolve as primeiras 10 marcações mais caras, ordenadas por valor descendente.

```
1 MATCH (n:Marcacao)
2 RETURN n
3 ORDER BY (n.valor) DESC
4 LIMIT 10;
5
```

Ilustração 15 - Query nº1

\$ MATCH (n:Marcacao) RETURN n ORDER BY n.valor DESC LIMIT 10;

Graph	n
Rows	<div> <div>valor</div> <div>100</div> </div> <div> <div>data_hora</div> <div>2016-09-03 16:00:00</div> </div> <div> <div>id</div> <div>10</div> </div>
Text	
Code	<div> <div>valor</div> <div>100</div> </div> <div> <div>data_hora</div> <div>2016-09-05 13:00:00</div> </div> <div> <div>id</div> <div>13</div> </div>
	<div> <div>valor</div> <div>100</div> </div> <div> <div>data_hora</div> <div>2016-10-31 16:00:00</div> </div> <div> <div>id</div> <div>152</div> </div>
	Started streaming 10 records after 91 ms and completed after 92 ms.

Ilustração 16 - Resultado query nº1

Query que devolve o número de clientes de cada loja

```

1 MATCH (l:Loja)-[:FREQUENTA]-(c)
2 RETURN l.id, count(c) as numero_clientes
3 ORDER BY numero_clientes DESC;
4

```

Ilustração 17 - Query nº2

\$ MATCH (l:Loja)-[:FREQUENTA]-(c) RETURN l, count(c) as numero_clientes ORDER BY numero_clientes DESC;		
Graph	l	numero_clientes
Rows	<div> <div>cidade</div> <div>Palmela</div> </div> <div> <div>id</div> <div>2</div> </div> <div> <div>rua</div> <div>Rua dos Poços</div> </div>	8
Text		
Code	<div> <div>cidade</div> <div>Pinhal Novo</div> </div> <div> <div>id</div> <div>3</div> </div> <div> <div>rua</div> <div>Bairro 12 de Abril</div> </div>	7
	<div> <div>cidade</div> <div>Setubal</div> </div> <div> <div>id</div> <div>1</div> </div> <div> <div>rua</div> <div>Avenida General Daniel de Sousa</div> </div>	7
Started streaming 3 records after 2 ms and completed after 3 ms.		

Ilustração 18 - Resultado query nº2

🚦 Query que devolve o número de cães de cada porte

```

1 MATCH (p) -[:PORTE] - (c:Cao)
2 RETURN p, count(c) as numero_caes
3 ORDER BY numero_caes DESC;
4
5

```

Ilustração 19 - Query nº 3

\$ MATCH (p)-[:PORTE]-(c:Cao) RETURN p, count(c) as numero_caes ORDER BY numero_caes DESC;		
Graph	p	numero_caes
Rows	<div> <div>modificador_preco</div> <div>1.5</div> </div> <div> <div>designacao</div> <div>Medio</div> </div> <div> <div>id</div> <div>2</div> </div>	9
Text		
Code	<div> <div>modificador_preco</div> <div>1</div> </div> <div> <div>designacao</div> <div>Pequeno</div> </div> <div> <div>id</div> <div>1</div> </div>	9
	<div> <div>modificador_preco</div> <div>2</div> </div> <div> <div>designacao</div> <div>Grande</div> </div> <div> <div>id</div> <div>3</div> </div>	8
Started streaming 3 records after 11 ms and completed after 11 ms.		

Ilustração 20 - Resultado query nº3

🚦 Informação sobre marcações com data valor, cliente, funcionário, cão e loja, ordenado por data decedente (começa pelo mais recente)

```
1 MATCH (m:Marcacao)-[:DE_CLIENTE]-(c:Cliente),
2      (m)-[:RESPONSAVEL]-(f:Funcionario),
3      (m)-[:LOCAL]-(l:Loja),
4      (m)-[:PARA_CAO]-(cao:Cao)
5 RETURN l.id, c.nome, f.nome, cao.nome, m.data_hora, m.valor
6 ORDER BY m.data_hora DESC
7
```

Ilustração 21 - Query nº4

§ MATCH (m:Marcacao)-[:DE_CLIENTE]-(c:Cliente), (m)-[:RESPONSAVEL]-(f:Funcionario), (m)-[:LOCAL]-(l:Loja), (m)-[:PARA_CAO]-(cao:Cao) RETURN l.id, c.nome, f.nome, cao.nome, m.data_hora, m.valor ORDER BY m.data_hora DESC

	l.id	c.nome	f.nome	cao.nome	m.data_hora	m.valor
Rows	2	Diana Lopes	Margarida Rodrigues	Churro	2016-11-25 16:00:00	48
A	3	Gabriela Vaz	Diogo Silva	Patusco	2016-11-25 15:00:00	12
Text	3	Joana Fernadnes	Diogo Silva	Rex	2016-11-25 10:00:00	32.25
</>	1	Juliana Sousa	Paulo cunha	Tita	2016-11-24 16:00:00	12
Code	1	José Maria	Maria Dantas	Luna	2016-11-24 15:00:00	20
	3	Joana Fernadnes	André Henriques	Rex	2016-11-23 14:00:00	75
	2	António Júnior	Maria Dantas	Nufa	2016-11-23 11:00:00	18
	1	Rodrigo Henriques	André Henriques	Laika	2016-11-23 10:00:00	75
	2	Tino Patas	Margarida Rodrigues	Bull	2016-11-22 18:00:00	12
	1	Juliana Sousa	Hugo Melo	Tita	2016-11-22 15:00:00	50
	3	Tiago Marques	Paulo cunha	Pipoca	2016-11-21 18:00:00	12
	2	Diana Lopes	Margarida Rodrigues	Churro	2016-11-21 16:00:00	38.25
	3	Gabriela Vaz	Diogo Silva	Patusco	2016-11-21 15:00:00	12
	1	Diogo Araujo	Maria Dantas	Max	2016-11-19 16:00:00	67
	2	Elísio Fernandes	Ricardo Cunha	Lord Nibbler	2016-11-19 12:00:00	21.5
	3	Leandro Zeferino	Diogo Silva	Kika	2016-11-18 15:00:00	62.25

Started streaming 209 records after 37 ms and completed after 48 ms.

Ilustração 22 - Resultado query nº4

🚦 Query que ordena os clientes consoante o número de marcações que já realizaram

```
1 MATCH (c:Cliente)-[]-(m:Marcacao)
2 RETURN c, count(m) AS num_marcacoes
3 ORDER BY num_marcacoes DESC;
4
5
```

Ilustração 23 - Query nº5

```
MATCH (c:Cliente)-[]-(m:Marcacao) RETURN c, count(m) AS num_marcacoes ORDER BY num_marcacoes DESC;
```

c		num_marcacoes
<div><div><div>cidade</div><div>Setubal</div></div><div><div>contacto</div><div>945890234</div></div><div><div>nif</div><div>235728194</div></div><div><div>nome</div><div>Juliana Sousa</div></div><div><div>id</div><div>1</div></div><div><div>email</div><div>juliana@gmail.com</div></div><div><div>rua</div><div>Rua do carvalho 24</div></div></div>		36
<div><div><div>contacto</div><div>915841236</div></div><div><div>cidade</div><div>Pinhal Novo</div></div><div><div>nif</div><div>216839174</div></div><div><div>nome</div><div>Rodrigo Henriques</div></div><div><div>id</div><div>2</div></div></div>		23

Started streaming 22 records after 6 ms and completed after 8 ms.

Ilustração 24 - Resultado query nº5

Query que selecciona os clientes da Loja 1

```
MATCH (c:Cliente)-[]-(l:Loja(id:1))
RETURN c;
```

Ilustração 25 - Query nº6

```
$ MATCH (c:Cliente)-[]-(l:Loja{id:1}) RETURN c;
```

Graph	c														
Rows															
Text															
Code															
	<table><tr><td>cidade</td><td>Setubal</td></tr><tr><td>contacto</td><td>922546879</td></tr><tr><td>nif</td><td>235579989</td></tr><tr><td>nome</td><td>Francisca Fernandes</td></tr><tr><td>id</td><td>16</td></tr><tr><td>email</td><td>francisca98@gmail.com</td></tr><tr><td>rua</td><td>Rua Caslopo 22</td></tr></table>	cidade	Setubal	contacto	922546879	nif	235579989	nome	Francisca Fernandes	id	16	email	francisca98@gmail.com	rua	Rua Caslopo 22
cidade	Setubal														
contacto	922546879														
nif	235579989														
nome	Francisca Fernandes														
id	16														
email	francisca98@gmail.com														
rua	Rua Caslopo 22														
	<table><tr><td>contacto</td><td>922316485</td></tr><tr><td>cidade</td><td>Setubal</td></tr><tr><td>nome</td><td>Diogo Araujo</td></tr><tr><td>id</td><td>20</td></tr><tr><td>email</td><td>diogoaraujo98@gmail.com</td></tr></table>	contacto	922316485	cidade	Setubal	nome	Diogo Araujo	id	20	email	diogoaraujo98@gmail.com				
contacto	922316485														
cidade	Setubal														
nome	Diogo Araujo														
id	20														
email	diogoaraujo98@gmail.com														
Started streaming 7 records after 16 ms and completed after 20 ms															

Ilustração 26 - Resultado query nº6

🚦 Query que lista os funcionários de todas as lojas

```
1 // Lista Funcionarios
2 MATCH (f:Funcionario) RETURN f;
```

Ilustração 27 - Query nº7

§ MATCH (f:Funcionario) RETURN f;

Graph	f										
Rows	<table> <tr><td>funcao</td><td>Recepcionista</td></tr> <tr><td>contacto</td><td>912343423</td></tr> <tr><td>nome</td><td>Júlio Fernandes</td></tr> <tr><td>id</td><td>1</td></tr> <tr><td>email</td><td>j_fernandes@esteticao.pt</td></tr> </table>	funcao	Recepcionista	contacto	912343423	nome	Júlio Fernandes	id	1	email	j_fernandes@esteticao.pt
funcao	Recepcionista										
contacto	912343423										
nome	Júlio Fernandes										
id	1										
email	j_fernandes@esteticao.pt										
Text											
Code											
	<table> <tr><td>funcao</td><td>Tosquiador</td></tr> <tr><td>contacto</td><td>96787374</td></tr> <tr><td>nome</td><td>André Henriques</td></tr> <tr><td>id</td><td>2</td></tr> <tr><td>email</td><td>a_henriques@esteticao.pt</td></tr> </table>	funcao	Tosquiador	contacto	96787374	nome	André Henriques	id	2	email	a_henriques@esteticao.pt
funcao	Tosquiador										
contacto	96787374										
nome	André Henriques										
id	2										
email	a_henriques@esteticao.pt										
	<table> <tr><td>funcao</td><td>Pet Groomer</td></tr> </table>	funcao	Pet Groomer								
funcao	Pet Groomer										

Started streaming 12 records after 1 ms and completed after 2 ms.

Ilustração 28 - Resultado query nº7

🚀 Query que devolve o número de clientes de cada cidade

```

2 MATCH (c:Cliente)
3 RETURN c.cidade, count(c) as numero_clientes
4 ORDER BY numero_clientes DESC

```

Ilustração 29 - Query nº8

```
$ MATCH (c:Cliente) RETURN c.cidade, count(c) as numero_clientes ORDER BY nume
```

	c.cidade	numero_client
Rows	Pinhal Novo	8
A	Palmela	8
Text	Setubal	6
</>		

Ilustração 30 - Resultado query nº8

🚦 Query que devolve o montante total resultante do somatório das vendas referente ao mês de Setembro de 2016

```
2 MATCH (m:Marcacao)
3 WHERE m.data_hora STARTS WITH '2016-09-'
4 RETURN sum(m.valor) AS Total_Vendas;
```

Ilustração 31 - Query nº9

```
$ MATCH (m:Marcacao) WHERE m.data_hora STARTS WITH '2016-09-' RETURN sum(m.valor)
```


	Total_Vendas
Rows	2690.5
A	
Text	
</>	
Code	

Ilustração 32 - Resultado query nº9

🚦 Query que devolve as marcações da Loja 2 referentes ao dia 2016-11-11

```
MATCH (m:Marcacao)-[:LOCAL]-(l:Loja{id:2})
WHERE m.data_hora STARTS WITH '2016-11-11'
RETURN m;
```

Ilustração 33 - Query nº10



The screenshot shows a database query result interface. At the top, the query is displayed: `$ MATCH (m:Marcacao)-[:LOCAL]-(l:Loja{id:2}) WHERE m.data_hora STARTS WITH '2016-11-11' RETURN m;`. Below the query, there are three records displayed in a table-like format. Each record is shown in a light gray box with a white background for the fields. The fields are 'valor', 'data_hora', and 'id'. The first record has 'valor' 12, 'data_hora' 2016-11-11 18:00:00, and 'id' 175. The second record has 'valor' 12, 'data_hora' 2016-11-11 15:00:00, and 'id' 174. The third record has 'valor' 32.25, 'data_hora' 2016-11-11 19:00:00, and 'id' 176. On the left side of the interface, there are icons for 'Graph', 'Rows', 'Text', and 'Code'. At the bottom, a status message reads: 'Started streaming 3 records after 1 ms and completed after 6 ms.'

m	
valor	12
data_hora	2016-11-11 18:00:00
id	175
valor	12
data_hora	2016-11-11 15:00:00
id	174
valor	32.25
data_hora	2016-11-11 19:00:00
id	176

Started streaming 3 records after 1 ms and completed after 6 ms.

Ilustração 34 - Resultado query nº10

🚦 Query que dá como resultado os funcionários com o salário mais alto

```
3 MATCH (f:Funcionario)
4 RETURN f, f.salario
5 ORDER BY f.salario DESC
```

Ilustração 35 - Query nº11

\$ MATCH (f:Funcionario) RETURN f, f.salario ORDER BY f.salario DESC

f		f.salario
Graph	<div><div>funcao</div><div>Gerente</div></div> <div><div>contacto</div><div>923554329</div></div> <div><div>salario</div><div>1120</div></div> <div><div>nome</div><div>Tito Dantas</div></div> <div><div>id</div><div>4</div></div> <div><div>email</div><div>t_dantaas@esteticao.pt</div></div>	1120
Rows	<div><div>funcao</div><div>Gerente</div></div> <div><div>contacto</div><div>921637289</div></div> <div><div>salario</div><div>1000</div></div> <div><div>nome</div><div>Daniel Braga</div></div> <div><div>id</div><div>12</div></div> <div><div>email</div><div>d_braga@esteticao.pt</div></div>	1000
Text		
Code		

Started streaming 12 records after 2 ms and completed after 3 ms.

Ilustração 36 - Resultado query nº11

🚦 Query que dá como resultado o somatório dos salários de cada loja

```
3 MATCH (l:Loja) - [:TRABALHA_EM] - (f:Funcionario)
4 RETURN l, sum(f.salario) AS Soma_Salario
5 ORDER BY Soma_Salario DESC
```

Ilustração 37 - Query nº12

\$ MATCH (l:Loja)-[:TRABALHA_EM]-(f:Funcionario) RETURN l, sum(f.salario) AS Soma_Salario ORDER BY Soma_Salario DESC

Graph

Rows

Text

Code

I

cidade

Setubal

id

1

rua

Avenida General Daniel de Sousa

2720

cidade

Pinhal Novo

id

3

rua

Bairro 12 de Abril

2653

cidade

Palmela

id

2

rua

Rua dos Poços

2650

Started streaming 3 records after 6 ms and completed after 7 ms.

Ilustração 38 - Resultado query nº12

6. Conclusão

O nosso cliente Estético consultou-nos expondo a necessidade de um acesso mais rápido aos registos.

Reunindo a equipa, surgiu a ideia da migração da base de dados de SQL para NoSQL.

Depois de criarmos um plano sólido e ideias fixas, reunimos de novo com o cliente e propusemos o novo modelo. Tendo o cliente ficado satisfeito com o plano criado, avançamos para o processo de migração.

Esperamos que com esta alteração o cliente obtenha o resultado esperado, que a eficácia e fiabilidade perdurem e que continue satisfeito com o desempenho da equipa.

Consideramos, portanto, que a proposta foi bem conseguida e ficamos ansiosos por futuras colaborações.

7. Bibliografia

[01] Connolly, T., Begg, C., Database Systems, A Practical Approach to Design, Implementation, and Management, Addison-Wesley, 4ª Edição, 2004, ISBN-10: 0321210255, ISBN-13: 978-0321210258

8. Referências WWW

[01] <http://docplayer.com.br/2298476-Neo4j-aprendendo-conceitos-por-tras-do-neo4j-sem-sql-apresentacao-por-que-grafos-por-que-agora-por-que-grafos-por-que-agora.html>

[02] <http://neo4j.com/docs/developer-manual/current/extending-neo4j/>

[03] <http://movidoashell.blogspot.pt/2014/05/diferencas-entre-banco-de-dados-sql-vs.html>

[04] http://ccsl.ime.usp.br/w/images/2/20/NoSQL_Vantagens_Desvantagens_e_Compromissos.pdf

[05] <https://pt.wikipedia.org/wiki/NoSQL>

[06] https://en.wikipedia.org/wiki/Graph_database

[07] <https://neo4j.com/why-graph-databases/>

[08] <http://db-engines.com/en/>