Universidade do Minho
Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2016/2017

Projeto de um Sistema de Bases de Dados não Relacional

Adriana Guedes, Diogo Soares, José Bastos e Ricardo Certo

janeiro, 2017

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Projeto de um Sistema de Bases de Dados não Relacional

Adriana Guedes, Diogo Soares, José Bastos e Ricardo Certo

janeiro, 2017

Resumo

Para esta segunda fase do trabalho tivemos que nos basear na base de dados construída na primeira fase, em SQL, e converte-la para um modelo NoSQL, nomeadamente em Neo4j, que é uma base de dados orientada a grafos.

Primeiramente, criou-se um script em SQL para exportar os dados de cada tabela da nossa base de dados inicial para um outro ficheiro de extensão .csv de modo a podermos importar esses dados para a nossa nova base de dados implementada em Neo4j.

Posteriormente, fizemos vários scripts na linguagem Cypher para conseguirmos criar os nodos e os seus respetivos relacionamentos. Depois de termos efetuado o povoamento, utilizando os scripts mencionados anteriormente, concebemos algumas queries que servem para nos dar alguma informação que esteja presente na nova base de dados criada e para verificar também se a nossa base de dados verifica os requisitos estipulados inicialmente.

Ao longo deste relatório efetuamos uma análise mais detalhada a todo este processo de conversão, desde a explicação da migração de um modelo de base de dados para o outro até à identificação das vantagens e desvantagens dos modelos de base de dados SQL e NoSQL.

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

Palavras-Chave: Relacionamentos, NoSQL, SQL, Neo4j, grafos, nodos, migração, companhia ferroviária, bilhete, sistema de reservas, cliente, viagens.

Índice

Índice de Figuras

1. Introdução

1.1 Contextualização

Uma companhia ferroviária é uma empresa responsável por prestar serviços de deslocação via terra, entre dois pontos geográficos, através de uma vasta rede de linhas férreas que atendam a interesses económicos e sociais. Dentro de uma companhia existe um sistema próprio de controlo de reservas de viagens que permite uma organizada venda de bilhetes.

O início do transporte ferroviário ocorreu no século VI. Os primeiros vestígios da existência de uma linha férrea remontam à Grécia Antiga, por volta do século 6 a.C., servindo, na altura, para o transporte de barcos.

Em 1804 foi criada a primeira ferrovia a operar com uma locomotiva a vapor, mas só em 1825 é que foi criada a primeira companhia ferroviária, operando locomotivas a vapor, a Stockton and Darlington Railway.

O aparecimento do transporte ferroviário esteve estritamente relacionado com a Revolução Industrial, ao longo dos séculos XVIII e XIX. Este meio de transporte emergiu na Europa, mais precisamente na Inglaterra, no século XIX. As locomotivas eram movidas a vapor. Após o surgimento deste inovador transporte, rapidamente a sua tecnologia se alastrou para outros pontos do mundo.

Em Portugal, as primeiras realizações práticas no meio ferroviário, ocorreram nas décadas de 1850 e 1860. A Companhia Central Peninsular dos Caminhos de Ferro de Portugal, mais conhecida como Companhia Peninsular, foi a primeira empresa ferroviária portuguesa, que construiu o troço entre Lisboa e o Carregado da Linha do Norte.

Em 2014 nasceu uma nova empresa ferroviária, Lei Express, pensada e criada por um grupo de jovens empresários dinâmicos e ambiciosos com o objetivo de inovar o transporte dos passageiros por um preço mais acessível do que o mercado oferecia até à data. Para isto ser possível optou-se por comboios com menor capacidade, mas que conseguem fazer grandes percursos, tornando assim o sistema mais acessível e económico para os clientes.

O objetivo desta nova empresa é crescer a nível internacional, melhorando os seus recursos de forma a tentar conquistar o seu lugar entre as grandes empresas do sector, e tornar-se numa das maiores potências da área em questão.

1.2 Apresentação do Caso de Estudo

Nesta segunda fase do trabalho prático o principal objetivo será a implementação de um *Sistema de Base de Dados* (SBD) não relacional tendo como foco o tema apresentado para a primeira fase do trabalho.

As bases de dados relacionais são orientadas a transações e baseiam-se no conceito de transações com as propriedades ACID. Quando as grandes empresas começaram a ter problemas de escala, perceberam que deveriam abrir mão destas propriedades propostas pelo modelo relacional para ganhar escalabilidade.

A tecnologia *NoSQL* foi originalmente criada e usada por líderes da Internet como *Facebook, Google, Amazon*, e outros que necessitavam de sistemas de gerenciamento de base de dados que pudessem escrever e ler dados em qualquer lugar no mundo e que garantam desempenho a grandes conjuntos de dados e milhões de usuários.

Assim, *NoSQL* surgiu com o objetivo de um alto desempenho perante grandes quantidades de dados, não respeitando as regras impostas pelo modelo relacional, permitindo assim uma maior flexibilidade na construção de uma base de dados consoante o objetivo desta.

O *Neo4j*, ferramenta que iremos utilizar nesta fase do trabalho, pertence à base de dados não relacional e distingue-se por funcionar à base de grafos. Cada nodo e aresta pode ter um número ilimitado de atributos, sendo que estes (nodos e arestas) podem também ser rotulados de forma a restringir pesquisas.

1.3 Motivação e Objetivos

Desde o início que a intenção foi criar um sistema que regista todos os bilhetes reservados na nossa companhia ferroviária. Considerando a contextualização apresentada, sabemos que o sistema de reservas é um sistema de elevado grau de complexidade de gerir. Existem aspetos bastante importantes a ter em conta tais como ter a informação completa e atualizada e sempre disponível para uma gestão de recursos mais eficiente e que seja lucrativa para a empresa.

Uma base de dados bem construída é fundamental para o correto funcionamento do sistema de reservas de uma companhia ferroviária.

Na estruturação do problema e elaboração da lista de requisitos deparamo-nos com a possibilidade de existir uma grande variedade de entidades assim como de relacionamentos. Tendo em conta que o objetivo é ter um sistema simples de base de dados tiveram que ser

tomadas decisões tendo sempre em conta que o sistema teria que possuir o máximo de informação útil que permitisse uma melhor organização.

O grande objetivo desta base de dados é modelar um protótipo de um sistema que permita a organização correta de toda a informação relativa à reserva de bilhetes de uma companhia ferroviária. Para isso é necessário analisar todas as possibilidades à disposição do utilizador para efetuar a reserva e fazer o tratamento desses dados.

A viabilidade presente na elaboração desta base de dados vai depender da possibilidade da sua implementação física a nível de terminais de acesso e também da qualidade da informação na atualização de dados. A inserção dos registos acerca da informação pedida vai ser feita numa plataforma online onde o cliente tem de efetuar o login e ter um saldo superior ao preço da reserva que quer efetuar antes de os dados da reserva serem registados na nossa base de dados.

Esta segunda parte do trabalho tem como objetivo fazer a importação de uma base de dados relacional para uma não relacional. O modelo de base de dados não relacional a ser implementado vai ser o Neo4j que é uma base de dados que utiliza grafos para guardar todas as informações em si inseridas a nível aplicacional. Para isso realizamos uma migração dos dados que possuímos na base de dados relacional para a nova base de dados a ser criada usando um modelo não relacional.

Ao longo da realização desta segunda parte do trabalho fomos adquirindo conhecimentos para fazermos uma diferenciação dos dois modelos usados e saber identificar os pontos fortes e fracos de cada modelo.

1.4 Estrutura do Relatório

Depois de termos apresentado a contextualização, o caso de estudo e a motivação por detrás deste, iremos seguidamente apresentar a nossa análise do caso em estudo.

Antes de avançarmos e focarmos nos na especificação do nosso trabalho, apresentamos a base de dados NoSQL. Sendo este trabalho a construção de uma base de dados deste tipo, falamos das suas principais características e diferenças comparando com a base de dados anteriormente construída, base de dados SQL, mostrando as suas vantagens perante esta, mas também as suas limitações, referindo posteriormente a importância da aprendizagem, para nós alunos, desta nova maneira de construção de base de dados que vem ganhando cada vês mais espaço no universo informático.

De seguida, descrevemos o modelo de base de dados NoSQL baseado em grafos, modelo este que foi implementado na nossa base de dados, usando o Neo4j, apresentando por fim o sistema da nossa base de dados.

Logo depois é apresentado detalhadamente todo o processo da migração de dados que teve de ser feito para o sucesso desta, e a consequente criação de todos os elementos necessários para a formação de uma base de dados final e operacional.

Antes de apresentarmos a conclusão do trabalho e consequente relatório, apresentamos as queries que representam a resposta às perguntas anteriormente apresentadas na análise do caso em estudo.

Por fim, expomos os anexos referentes ao trabalho e as referencias bibliográficas.

2. Análise do Caso em Estudo

Nesta segunda parte do trabalho baseamo-nos nos requisitos definidos na primeira parte, ou seja, apesar de termos mudado de modelo na implementação da base de dados os requisitos definidos inicialmente vão se manter. Vamos também nos deparar com uma realidade diferente de implementarmos uma base de dados pois passamos de uma implementação relacional para uma implementação não relacional baseada em grafos. Cada uma tem as suas características próprias e seus prós e contras.

As entidades definidas no modelo anterior também se vão manter só que agora vão estar na forma de nodos e estes nodos vão se relacionar com a criação de relacionamentos próprios do Neo4j.

As perguntas que o nosso sistema tem de responder vão ser as mesmas definidas anteriormente pois apesar de mudarmos de modelo de base de dados os requisitos vão-se manter iguais. Relembrando algumas perguntas que o nosso sistema tem de responder:

1) Quantas viagens foram reservadas para um determinado percurso num determinado dia?
2) Quais os lugares desocupados para um certo percurso num dado dia?
3) Quais os possíveis percursos a realizar num dado dia num intervalo de horas dadas?
4) Qual o histórico de reservas de um dado cliente?
5) Quais os lugares desocupados no comboio x no percurso y para deficientes?

3. Base de dados *NoSQL*

O termo NoSQL (*Not Only SQL*) descreve soluções de armazenamento de base de dados não relacionais. Pode assim dizer-se que uma base de dados NoSQL, além de não ter interface SQL, é open source e completamente distinta do modelo relacional tradicional, isto porque as bases de dados NoSQL foram justamente projetadas a partir de necessidades que as bases de dados tradicionais relacionais não satisfaziam, como alta performance e capacidade de expansão. Podemos afirmar que existem quatro tipos diferentes de base de dados NoSQL, que são os seguintes:

- ➔ Chave-Valor: Armazena os seus dados num padrão chave-valor, fazendo lembrar as tabelas de hash. Ex: MemcacheD, Riak, REDIS.
- ➔ Grafos: Armazena os seus dados na forma de grafo, com vértices e arestas. Ex: Neo4j, Sesame.
- ➔ Colunas: Armazena os seus dados em linhas particulares de uma tabela no disco. Ex: Cassandra, Hibase;
- ➔ Documento: armazena os seus dados como documentos, onde um documento pode ser um dado num formato chave-valor (por exemplo o padrão JSON). Ex: MongoDB, CouchDB.

3.1 Principais características

- Alta performance e escalabilidade horizontal

Enquanto que em servidores de base de dados tradicionais possuíam a adição de um maior número de recursos ao servidor, como memória e disco, para suportar mais dados, as bases de dados NoSQL redistribuem-se horizontalmente, ou seja, funcionam como um sistema fracamente acoplado, o que a torna mais económica e escalável. Graças a esse mecanismo, a sua performance provém de não possuir um servidor central com muito poder computacional, mas sim, de vários servidores, que não necessitam de ser de alta performance, conectados e trabalhando em conjunto.

- Replicação

A replicação é uma estratégia que se encaixa perfeitamente na arquitetura do NoSQL, já que ele segue o conceito de sistema fracamente acoplado. Essa ferramenta possibilita o armazenamento de dados e backups independentemente do local físico onde os dados estão armazenados.

- Alta disponibilidade
- API simples para acesso aos dados
- Raízes Open Source

Muitas bases de dados NoSQL tem raízes na comunidade open source. Talvez isso tenha sido fundamental para o rápido crescimento do seu uso e popularidade. Nota-se que as companhias que oferecem versões comerciais de base de dados NoSQL com uma forte estrutura de suporte e serviços. Temos o exemplo do Facebook que já utiliza este modelo de bases de dados para guardar todas as informações.

- Baixo custo operacional

Devido ao peso do open source no NoSQL, o custo para iniciar a utilização dessas bases de dados torna-se muito baixo ou zero. É comum ouvir dizer que a transição relacional -> NoSQL diminuiu muito os custos enquanto obteve um desempenho melhor ou igual ao anterior. As bases de dados relacionais mais elaboradas requerem computadores com excelentes especificações. Com o NoSQL, esse custo também diminui, pois este foi desenvolvido para trabalhar em ambientes distribuídos.

- Armazenamento de dados estruturados ou não estruturados

Permite uma fácil aplicação da escalabilidade e também um aumento na disponibilidade de dados. Na ausência de dados estruturados não existe garantia de haver integridade de dados.

3.2 SQL vs NoSQL

3.2.1 Armazenamento de Dados

- SQL

O armazenamento é num modelo relacional, em tabelas, em que as linhas são as entradas/entidades e as colunas os atributos das respetivas entidades.

- NoSQL

O NoSQL abrange uma série de base de dados, cada uma com a sua estrutura de armazenamento de dados.

3.2.2 Estrutura e Flexibilidade

- SQL

Alterações que sejam feitas a nível de estrutura implicam alterar toda a base de dados.

- NoSQL

Estruturas dinâmicas. Informações podem ser adicionadas facilmente e permite haver entradas em colunas da tabela vazias.

3.2.3 Escalabilidade

- SQL

A escalabilidade é vertical. Assim, mais dados implica haver um servidor maior o que poderá ser dispendioso. É possível escalar uma base de dados em vários servidores, mas isso é um processo difícil e demorado.

- NoSQL

A escalabilidade é horizontal, como já referido anteriormente. Mais económico.

3.2.4 Propriedades ACID

- SQL

A grande maioria dos bancos de dados relacionais são compatíveis com ACID.

- NoSQL

Varia de acordo com as tecnologias, mas muitas soluções NoSQL sacrificam a compatibilidade ACID para desempenho e escalabilidade.

3.3 Vantagens

- Dados sempre disponíveis;
- Custo mais reduzido que uma base de dados relacional;
- Base de dados orientada a objetos flexíveis;
- Facilidade em introduzir novos dados;
- Excelente maneira de lidar com o problema de dados em massa.

3.4 Limitações

- Mais recentes que as bases de dados relacionais, não contendo tantas alternativas;
- Não irão resolver problemas de escalabilidade de um website;
- Situações onde o modo como os dados estão estruturados é importante, como é o caso de contas bancárias, uma base de dados relacional é muito mais adequada.

3.5 Modelo de Dados *NoSQL* baseado em grafos

As bases de dados relacionais não permitem que os dados sejam representados através de grafos de uma forma natural, com isso, algumas pesquisas tornam-se

extremamente complexas, ou até mesmo inconcebíveis nas bases de dados relacionais.

Uma base de dados baseada em grafos utiliza o modelo de grafos para representar o esquema. Esse modelo apresenta três componentes básicas: nodos (vértices), os relacionamentos entre nodos (arestas) e as propriedades dos nodos ou dos relacionamentos existentes entre eles. Assim, nas bases de dados baseadas em grafos basta apenas navegarmos pela estrutura de dados desse grafo para conseguimos obter os resultados pretendidos.

Este modelo de dados está diretamente relacionado à teoria de grafos e suporta a utilização de restrições de integridade de dados garantindo assim as relações entre elementos de forma consistente.

3.6 Importância da aprendizagem de um modelo não relacional

Cada vez mais no mercado a criação de base de dados não relacionais tem vindo a aumentar exponencialmente devido ás suas características referidas a cima e também pela sua facilidade de utilização.

As bases de dados não relacionais representam entre 10% a 20% do mercado mundial, parecem números baixos, mas se virmos isto na perspetiva mundial estas valores são na ordem dos milhões de utilizadores. Ao fazermos o trabalho numa primeira fase num modelo relacional e depois nesta segunda fase num modelo não relacional permitiu-nos ter um maior conhecimento sobre os diferentes tipos de base de dados que podemos vir a implementar no futuro.

3.7 Novo sistema de base de dados

Na figura que se segue podemos ter uma melhor perceção de como vai ficar implementada a nossa base de dados em Neo4j, identificando para isso todos os tipos de nodos presentes e todos os relacionamentos entre os nodos. Essa figura vai funcionar tipo um esboço de modo a facilitar a perceção do conteúdo de dados que a nossa base de dados vai suportar.

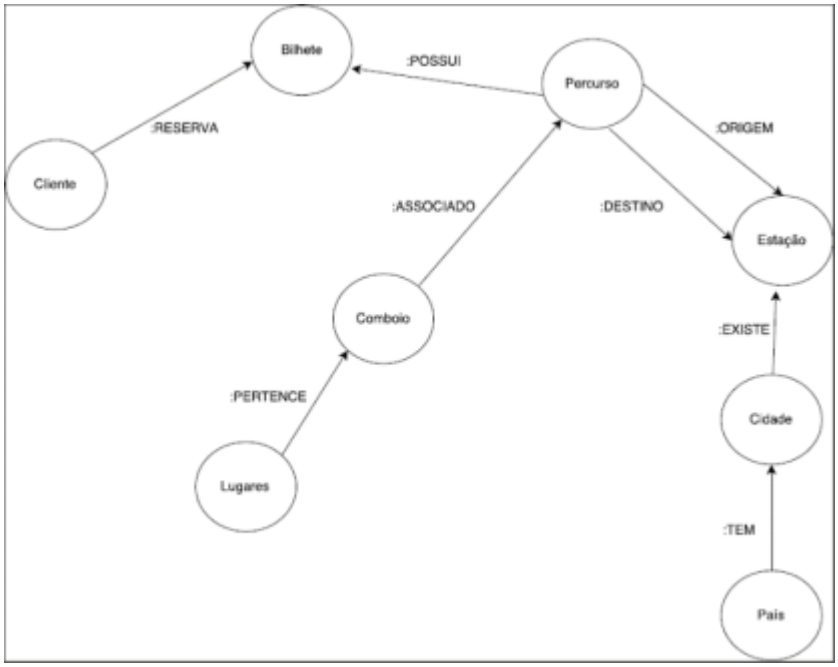

Figura 1 - Esquema tipo do novo sistema de base de dados

Em tópicos mais à frente do relatório iremos aprofundar mais o novo sistema de Base de dados.

4. Migração de Dados

Esta migração consiste no processo de exportação dos dados que temos na nossa base de dados em MySQL, que antes tinha sido povoada na primeira parte do projeto, para ficheiros com extensão .csv. Cada tabela vai ser exportada para um ficheiro diferente com o respetivo nome. Neste processo de migração tivemos que importar os documentos para uma diretoria específica pertencente ao MySQL Server devido às permissões deste.

4.1 Importação dos ficheiros csv para o Neo4j

```
USE leiExpress;

select * FROM bilhete
into outfile 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/bilhete.csv'
fields enclosed by '"' terminated by ',' escaped by ''
lines terminated by '\r\n';

select * FROM cidades
into outfile 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/cidades.csv'
fields enclosed by '"' terminated by ',' escaped by ''
lines terminated by '\r\n';

select * FROM cliente
into outfile 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/cliente.csv'
fields enclosed by '"' terminated by ',' escaped by ''
lines terminated by '\r\n';

select * FROM comboio
into outfile 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/comboio.csv'
fields enclosed by '"' terminated by ',' escaped by ''
lines terminated by '\r\n';

select * FROM estação
into outfile 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/estacao.csv'
fields enclosed by '"' terminated by ',' escaped by ''
lines terminated by '\r\n';

select * FROM lugares
into outfile 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/lugar.csv'
fields enclosed by '"' terminated by ',' escaped by ''
lines terminated by '\r\n';

select * FROM paises
into outfile 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/pais.csv'
fields enclosed by '"' terminated by ',' escaped by ''
lines terminated by '\r\n';

select * FROM percurso
into outfile 'C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/percurso.csv'
fields enclosed by '"' terminated by ',' escaped by ''
lines terminated by '\r\n';
```

Figura 2 - Migração de dados do MySQL Server para um ficheiro .csv

Depois de termos os ficheiros .csv, fizemos vários scripts na linguagem Cypher para criar os nodos e os seus respetivos relacionamentos. Tivemos de fazer vários scripts devido à incompatibilidade da versão utilizada do Neo4j que não permite a criação dos nodos todos no mesmo script, e por isso temos de ter vários scripts.

Utilizamos um ‘LOAD’ para carregarmos o ficheiro .csv com as informações da tabela para o Neo4j. Com um ‘CREATE’ criamos os respetivos nodos e com um ‘MERGE’ os relacionamentos presentes entre nodos.

4.2 Criação de Nodos

```
USING PERIODIC COMMIT
LOAD CSV FROM "file:///cliente1.csv" AS line
CREATE (c:Cliente { ClienteID: (line[0]),
Nome: (line[2]),
Email: (line[1]),
Password: (line[3]),
Saldo: TOFLOAT(line[4])})

USING PERIODIC COMMIT
LOAD CSV FROM "file:///cidades1.csv" AS line
CREATE (ci:Cidade { CidadeID: (line[1]),
Nome: (line[0]),
Paises: (line[2])})

USING PERIODIC COMMIT
LOAD CSV FROM "file:///comboio1.csv" AS line
CREATE (co:Comboio {ComboioID: (line[0]),
Capacidade: (line[1])})

USING PERIODIC COMMIT
LOAD CSV FROM "file:///estacao1.csv" AS line
CREATE (e:Estacao {EstacaoID: (line[1]),
Nome: (line[0]),
Cidade: (line[2])})

USING PERIODIC COMMIT
LOAD CSV FROM "file:///paises1.csv" AS line
CREATE (pa:Pais {PaisID: (line[1]),
Nome: (line[0])})
```

```
USING PERIODIC COMMIT
LOAD CSV FROM "file:///percurso1.csv" AS line
CREATE (pe:Percurso { PercursoID: (line[0]),
Partida: (line[1]),
Chegada: (line[2]),
Linha: (line[3]),
Preco: (line[4]),
Comboio: (line[5])})

USING PERIODIC COMMIT
LOAD CSV FROM "file:///lugares1.csv" AS line
CREATE (l:Lugar { NumeroLugar: (line[0]),
Carruagem: (line[1]),
Tipo: (line[2]),
Comboio: (line[3])})

USING PERIODIC COMMIT
LOAD CSV FROM "file:///bilhete1.csv" AS line
CREATE (b:Bilhete { BilheteID: (line[0]),
Carruagem: (line[1]),
Lugar: (line[2]),
Preco: (line[3]),
Tipo: (line[4]),
Data: (line[5]),
Viagem: (line[6]),
Cliente: (line[7]),
Percurso: (line[8])})
```

Figura 3 - Script de criação de nodos

Como podemos observar nas duas imagens anteriores temos os scripts que são

responsáveis por efetuarem a criação dos nodos com base nos valores contidos no ficheiro.csv. Para além de criarmos os respetivos nodos inserimos toda a informação que o ficheiro .csv continha sobre esse nodo em específico.

4.3 Criação dos Indexes

```
CREATE INDEX ON :Cliente(ClienteID);
CREATE INDEX ON :Bilhete(BilheteID);
CREATE INDEX ON :Percurso(PercursoID);
CREATE INDEX ON :Comboio(ComboioID);
CREATE INDEX ON :Estacao(EstacaoID);
CREATE INDEX ON :Pais(PaisID);
CREATE INDEX ON :Cidade(CidadeID);
CREATE INDEX ON :Lugar(NumeroLugar);
CREATE INDEX ON :Lugar(ComboioID);
```

Figura 4 - Script de criação de indexes

Estes Indexes (Índices) vão ser uma cópia redundante das informações da nossa base de dados e tem como objetivo tornar a pesquisa de dados mais eficiente. Isto requer um espaço extra de armazenamento e gravações mais lentas, pelo que este processo desempenha um papel importante e muita das vezes não trivial.

A linguagem Cypher permite a criação de índices sobre uma dada propriedade para todos os nodos que têm um rótulo em comum. Uma vez criado o índice, ele será automaticamente gerenciado e atualizado pela base de dados, sempre que o gráfico for alterado.

4.4 Criação de Relacionamentos

```
USING PERIODIC COMMIT
LOAD CSV FROM "file:///bilhete1.csv" AS row
MATCH (bil:Bilhete {BilheteID: row[0]})
MATCH (cli:Cliente {ClienteID: row[7]})
MERGE (cli) -[:RESERVA]-> (bil);

USING PERIODIC COMMIT
LOAD CSV FROM "file:///bilhete1.csv" AS row
MATCH (bil:Bilhete {BilheteID: row[0]})
MATCH (per:Percurso {PercursoID: row[8]})
MERGE (per) -[:POSSUI]-> (bil);

USING PERIODIC COMMIT
LOAD CSV FROM "file:///lugares1.csv" AS row
MATCH (lug:Lugar {NumeroLugar: row[0]})
MATCH (comb:Comboio {ComboioID: row[3]})
MERGE (lug) -[:PERTENCE]-> (comb);

USING PERIODIC COMMIT
LOAD CSV FROM "file:///cidades1.csv" AS row
MATCH (ci:Cidade {CidadeID: row[1]})
MATCH (pa:Pais {PaisID: row[2]})
MERGE (pa) -[:TEM]-> (ci);

USING PERIODIC COMMIT
LOAD CSV FROM "file:///estacao1.csv" AS row
MATCH (e:Estacao {EstacaoID: row[1]})
MATCH (ci:Cidade {CidadeID: row[2]})
MERGE (ci) -[:LOCALIZADA]-> (e);
```

Figura 5 - Script da criação de relacionamentos

Como podemos verificar nas imagens anexadas em cima vamos possuir 8 diferentes tipos de relacionamentos entre os nodos. Estes são:

- RESERVA

É o relacionamento entre cada cliente e cada bilhete que foi reservado pelo mesmo.

- POSSUI

É o relacionamento entre cada percurso e cada bilhete associado a esse percurso.

- PERTENCE

É o relacionamento entre cada lugar e um comboio específico, ou seja, é a representação dos lugares de um dado comboio.

- TEM

É o relacionamento entre cada país e cada cidade, pois cada país pode possuir várias cidades a si associadas.

- EXISTE

É o relacionamento entre cidade e estação, pois em cada cidade pode existir uma dada estação.

- ASSOCIADO

É o relacionamento entre percurso e comboio, pois cada percurso vai ter a si um comboio associado.

- ORIGEM

É o relacionamento entre percurso e estação de origem, pois cada percurso vai possuir uma estação de origem.

- DESTINO

É o relacionamento entre percurso e estação de destino, pois cada percurso vai possuir uma estação de destino.

Estes relacionamentos que foram mencionados em cima vão desempenhar um papel importante na elaboração de queries futuras, pois vai ser através deles que vamos navegar no grafo de modo a obtermos a informação desejada.

5. *Queries*

A elaboração de *queries* em qualquer modelo de Base de Dados é um processo importante na maneira de obtermos os dados pretendidos e tirar conclusões através desses resultados obtidos sobre se a elaboração da nossa base de dados cumpre todos os requisitos pedidos. As queries no modelo Neo4j baseiam-se na linguagem *Cypher*. De seguida apresentamos algumas queries e a sua respetiva explicação que efetuamos na nossa base de dados criada nesta segunda fase do projeto.

- MATCH (n:Cliente) RETURN n.Email

Com esta *query* podemos mostrar que a base de dados seleciona todos os clientes, retornando os respetivos emails.

- MATCH (Cliente {ClienteID:'1'})-[:RESERVA]->(Bilhete)

RETURN Cliente.Saldo, Bilhete.Preco

Selecionado o Cliente cujo ClienteID é 1 e que tem um relacionamento de reserva com o bilhete retorna o seu saldo e o preço dos respetivos bilhetes.

- MATCH (n:Cliente)
  WHERE (n.Saldo>200)
  RETURN n.Nome,n.Saldo

Esta *query* retorna o nome e o saldo de todos os clientes cujo saldo é superior a 200€.

6. Versão final da Base de Dados em Neo4j

O Neo4j na parte da visualização gráfica da base de dados, fornece-nos uma identificação colorida dos diferentes tipos de nodos presentes na base dados o que facilita a interpretação dos relacionamentos e tipos de nodos nela presentes.

Como podemos verificar na figura que se segue cada tipo de nodo vai ter uma cor específica que o identifica.

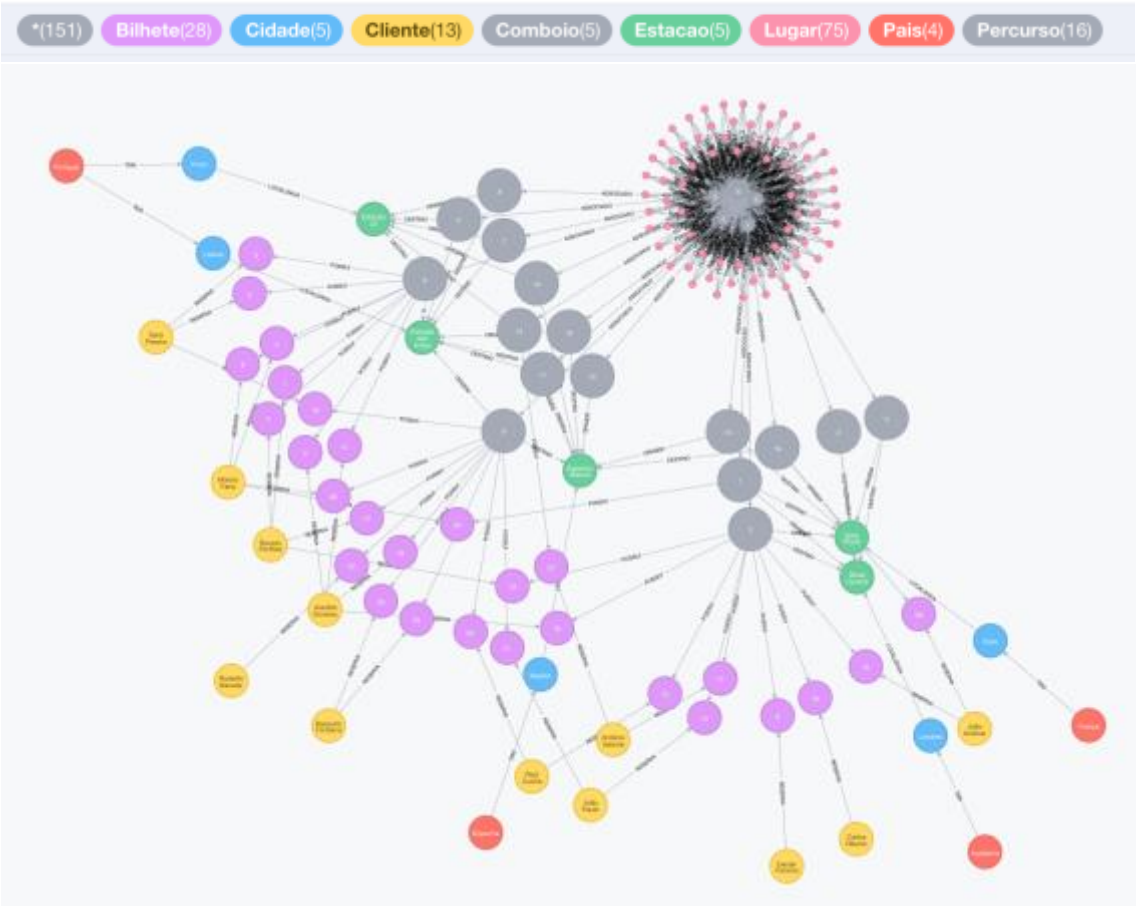

Figura 6 - Modelo final da nova base de dados

O Neo4j oferece uma API simples para acesso aos dados, como podemos verificar quando olhamos para o resultado gráfico da BD podemos logo identificar todos os tipos de dados nela presentes de uma forma mais fácil. Para além disso trata-se de uma base de dados de custo mais reduzido em comparação com uma base de dados relacional e também facilita a introdução de novos dados.

7. CONCLUSÃO

A realização deste trabalho permitiu nos adquirir competências a trabalhar com uma base de dados NoSQL assim como compreender de uma forma prática as principais diferenças de uma base de dados relacional e não relacional.

Nesta segunda parte do trabalho não tivemos de passar pela construção dos modelos necessários à base de dados, mas sim adaptar estes anteriormente feitos ao agora pretendido, passando assim por varias etapas e percebendo o que é necessário para a passagem de uma base de dados relacional a uma não relacional.

O uso de NoSQL e nomeadamente do modelo Neo4J deu-nos outra perspetiva sobre as diferentes maneiras da criação de uma base de dados, pois esta é constituída por grafos e nodos, algo completamente diferente daquilo que tínhamos feito numa primeira parte do projeto.

Além disso tivemos de nos adaptar a outra linguagem na criação da nova base de dados, que serviu para criar nodos, criar relacionamentos e realizar as queries para obtermos a informação necessária da nossa base de dados.

Posto isto, confirmamos aquilo que nos foi transmitido sobre as Base de dados NoSQL, tendo achado as principais vantagens e mudanças ao utilizar o Neo4j o facto de ele possuir um elevado grau de distribuição dos dados o que propicia um maior número de solicitações aos dados, evitando assim que o sistema fique menos tempo não disponível, achando por outro lado que o modelo relacional era mais consistente, pois as regras de consistência presentes proporcionam-lhe um grau de rigor elevado quanto á consistência das suas informações.

Ao realizarmos este trabalho que consistia em realizarmos a mesma base de dados num modelo relacional e num modelo não relacional, adquirimos uma ideia de como ambas funcionam e respetiva importância que cada uma têm no mercado de trabalho referente a esta área. As bases de dados relacionais são as mais usadas no mundo de trabalho, mas á mediada que o tempo vai avançando as não relacionais também vão ganhando o seu espaço no mercado devido ás características referidas a cima. Ou seja, ao realizarmos estre trabalho ficamos também com um sentido crítico para num projeto futuro decidirmos qual o modelo de

base de dados a implementar consoante os requisitos que nos vão ser apresentados.

Referências

[01] Website do Neo4j. https://neo4j.com/

Lista de Siglas e Acrónimos

BD .....Base de Dados
SGBD ..... Sistema de Gestão de Base de Dados
ACID.....Atomicidade, Consistência, Isolamento e Durabilidade