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

Unidade Curricular de Bases de Dados

Ano Lectivo de 2016/2017

Pedro Dias Imobiliária

João Dias, Luís Silva, Nelson Sá, Tiago Loureiro

Janeiro, 2017

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Pedro Dias Imobiliária

João Dias a 66850, Luís Silva a72113,

Nelson Sá a70925, Tiago Loureiro a71191

Janeiro, 2017

Resumo

Neste trabalho, começamos por fazer a migração da base de dados, criada no projeto anterior, para um sistema de base de dados não relacional (Neo4J).

Depois, passamos para a fase de criação das *queries*, que permitem verificar a operacionalidade da base de dados, que são equivalentes às criadas para o modelo relacional.

Por fim, fazemos uma análise, comparando os modelos, as funcionalidades e as diferenças na implementação dos dois modelos. Assim como as vantagens e desvantagens dos modelos e qual seria o mais adequado para o caso apresentado.

Índice

Índice de Figuras

Índice de Tabelas

Não foi encontrada nenhuma entrada do índice de ilustrações.

1. Introdução

Para modular uma base de dados, além dos modelos relacionais, podemos usar modelos não relacionais como, por exemplo, o Neo4J.

O Neo4J é um sistema de gestão de base de dados não relacional baseado em grafos, altamente escalável, construído especificamente para alavancar não apenas dados, mas também os seus relacionamentos.

Uma base de dados baseada em grafos, apresenta algumas diferenças em relação a outras bases de dados NoSQL. Está diretamente relacionada a um modelo de dados estabelecido, o modelo de grafos. A ideia desse modelo é representar os dados e/ou o esquema dos dados como grafos orientados, ou como estruturas que generalizem a noção de grafos. O modelo baseado em grafos é mais interessante que outros quando "informações sobre a inter-conectividade ou a topologia dos dados são mais importantes, ou tão importante quantos os dados propriamente ditos.

O modelo baseado em grafos possui três componentes básicos: os nós (são os vértices do grafo), os relacionamentos (são as arestas) e as propriedades (ou atributos) dos nós e relacionamentos. Neste caso, a base de dados pode ser visto como um multigrafo rotulado e orientado, onde cada par de nós pode ser conectado por mais de uma aresta.

1.1. Migração

Neste capitulo vamos apresentar passo-a-passo como migramos da nossa base de dados relacional já criada anteriormente para uma base de dados não relacional Neo4J.

1ºPasso

Criar uma pasta com o nome "*import"* dentro da pasta Neo4J.

Para cada tabela criada no MySQL, fazer o seu *export* (ficheiro de extensão .csv) para a pasta criada no passo anterior como mostra a figura em baixo.

```
7
8  use Imobiliaria;
9  select * from Funcionario;
10
11
```

100% 17:8 1 error found

Result Grid | Filter Rows: Search | Edit: | Export/Import:

idFuncionario	Nome	Data_Nasc	idContactoFunciona...
1	Neca Cruz	1975-07-12	1
2	Dolbeth Domingues	1989-02-10	2
3	Castor Oliveira	1967-04-10	3
4	Oliveiros Carvalho	1968-10-25	4
5	Baltazar Miranda	1989-05-08	5
6	Frazine Fernandes	1971-06-10	6
7	Carisa Araujo	1985-05-21	7
8	Marcelo Leal	1990-11-09	8
9	Londrim Rocha	1986-09-21	9
10	Lucete Martins	1981-12-17	10
11	Suzi Vaz	1968-03-07	11
12	Carmezinda Maia	1943-09-06	12
13	Carminho Andrade	1960-10-24	13
14	Diquel Cruz	1943-03-12	14
15	Cardinal Antunes	1981-05-23	15
16	Eleazar Pinto	1992-07-05	16
17	Heldemaro Pinheiro	1966-08-10	17
18	Lecticia Castro	1990-01-06	18
19	Africa Leite	1987-09-15	19
20	Amor Araujo	1987-06-22	20
NULL	NULL	NULL	NULL

Figura 1 Export Tabela

2ºPasso

No Neo4J, executar, para cada ficheiro de extensão .csv criado anteriormente, a *querie* em baixo exemplificada.

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///Funcionario.csv" AS row
CREATE (:Funcionario {id:row.idFuncionario,nome: row.Nome,data_nascimento:
row.Data_Nasc});
```

Figura 2 Criar Nodos

Que vai criar no Neo4J os nós, por exemplo, de todos os 20 funcionários e a respetiva informação dos mesmos, como mostram as figuras em baixo.

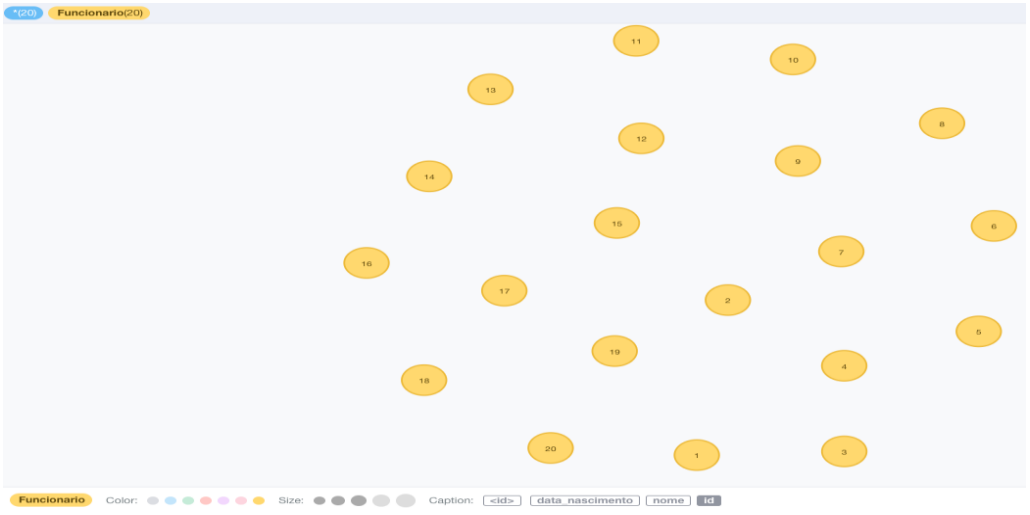

Figura 3 Nodos

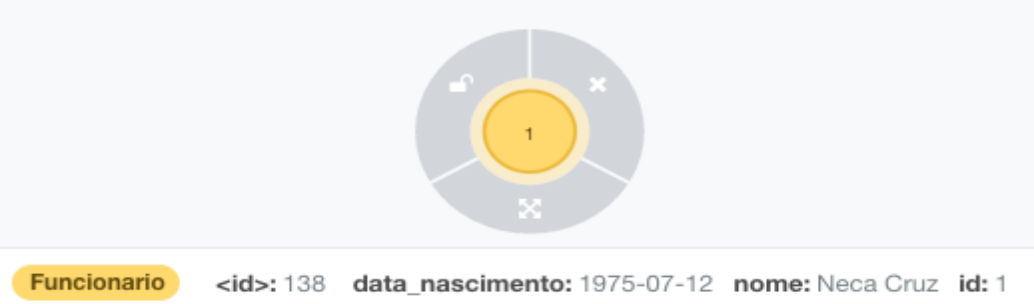

Figura 4 Atributos/Nodo

3ºPasso

Criar os relacionamentos (arestas), no Neo4J, consiste em dar um nome ao relacionamento entre os nós envolvidos, que já foi efetuada no projeto anterior.

Em que fazemos as seguintes *queries* para as relações que existem no nosso trabalho.

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///Cliente.csv" As row
Match (cliente:Cliente{id:row.idCliente})
Match (contactoC:ContactoCliente {id:row.idContactoCliente})
MERGE (contactoC)<-[:Contacto]-(cliente);

USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///Funcionario.csv" As row
Match (funcionario:Funcionario{id:row.idFuncionario})
Match (contactoF:ContactoFuncionario {id:row.idContactoFuncionario})
MERGE (contactoF)<-[:Contacto]-(funcionario);
```

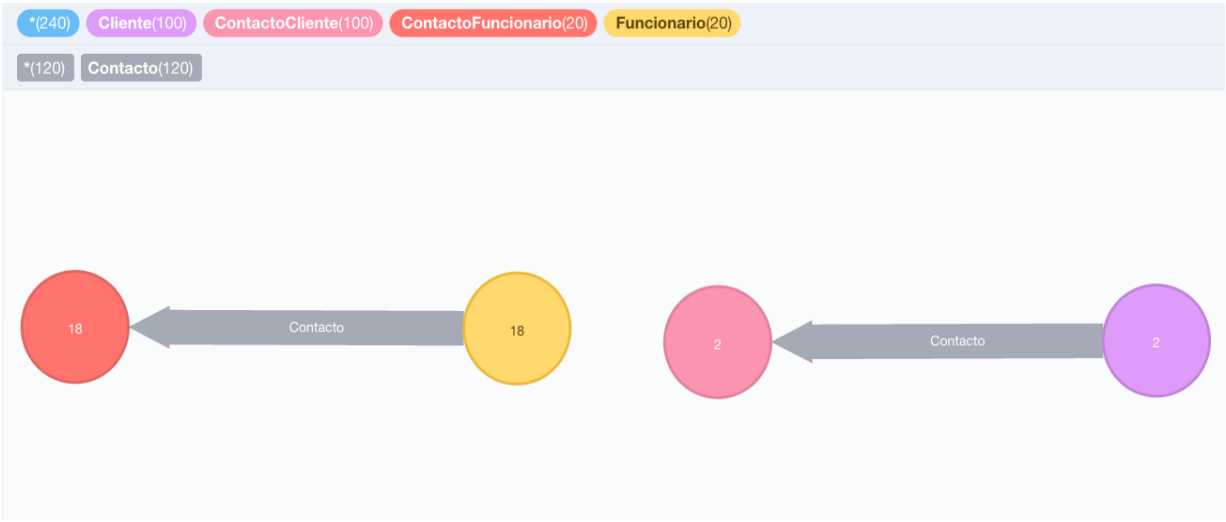

Figura 5 Relação Contacto

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///Transacao.csv" As row
Match (cliente:Cliente{id:row.idCliente})
Match (transacao:Transacao {id:row.idTransacao})
MERGE (transacao)<-[:Efectua]-(cliente);
```

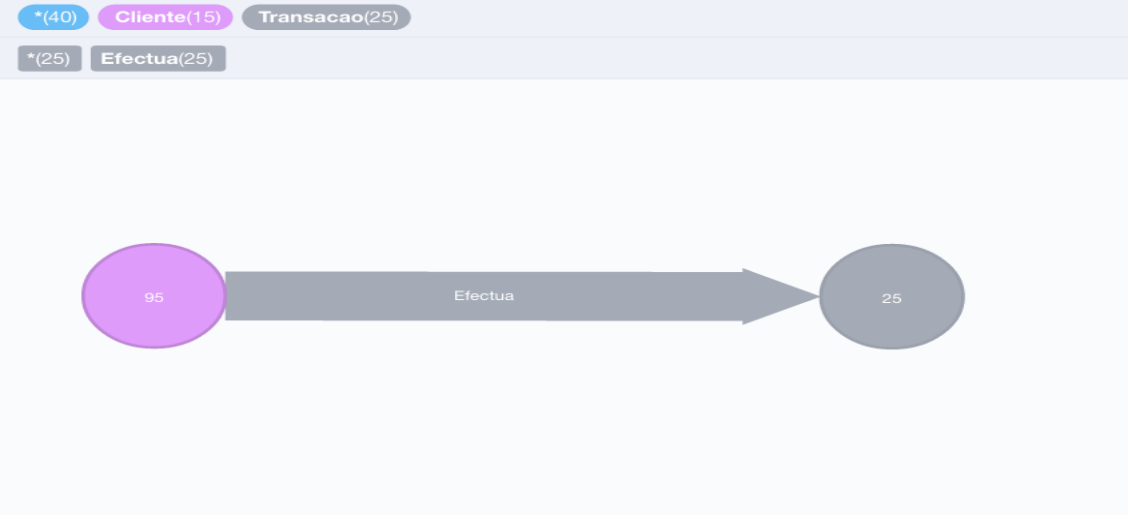

Figura 6 Relação Efectua

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///Imovel.csv" As row
Match (imovel:Imovel{id:row.idImovel})
Match (cliente:Cliente {id:row.idCliente})
MERGE (cliente)-[:Proprietario]->(imovel);
```

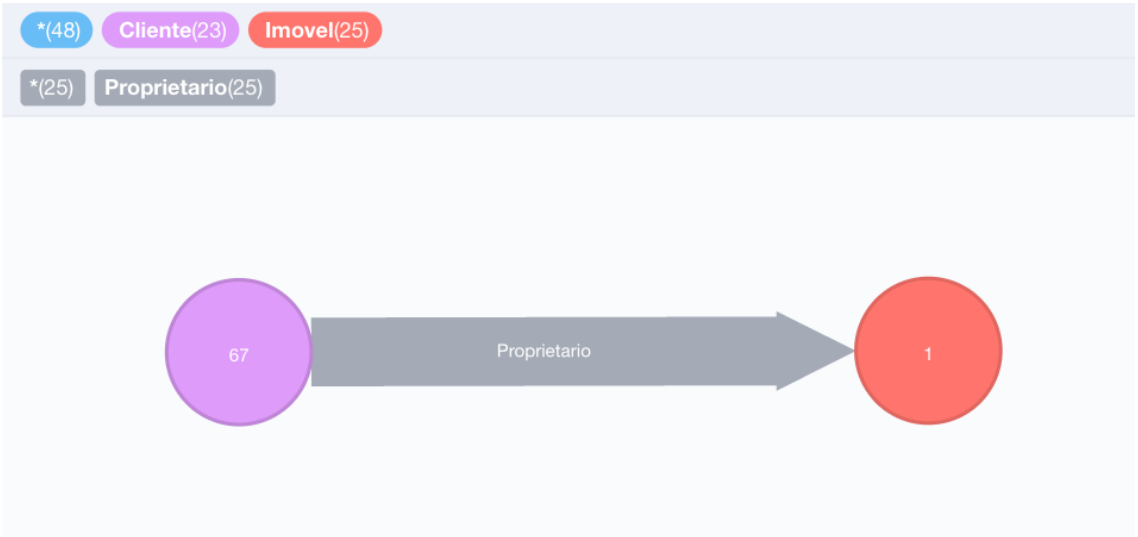

Figura 7 Relação Proprietário

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///Transacao.csv" As row
Match (funcionario:Funcionario{id:row.idFuncionario})
Match (transacao:Transacao {id:row.idTransacao})
MERGE (transacao)<-[:Realiza]-(funcionario);
```

Figura 8 Relação Realiza

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///Imovel.csv" As row
Match (imovel:Imovel{id:row.idImovel})
Match (funcionario:Funcionario {id:row.idFuncionario})
MERGE (funcionario)-[:Tem]->(imovel);
```

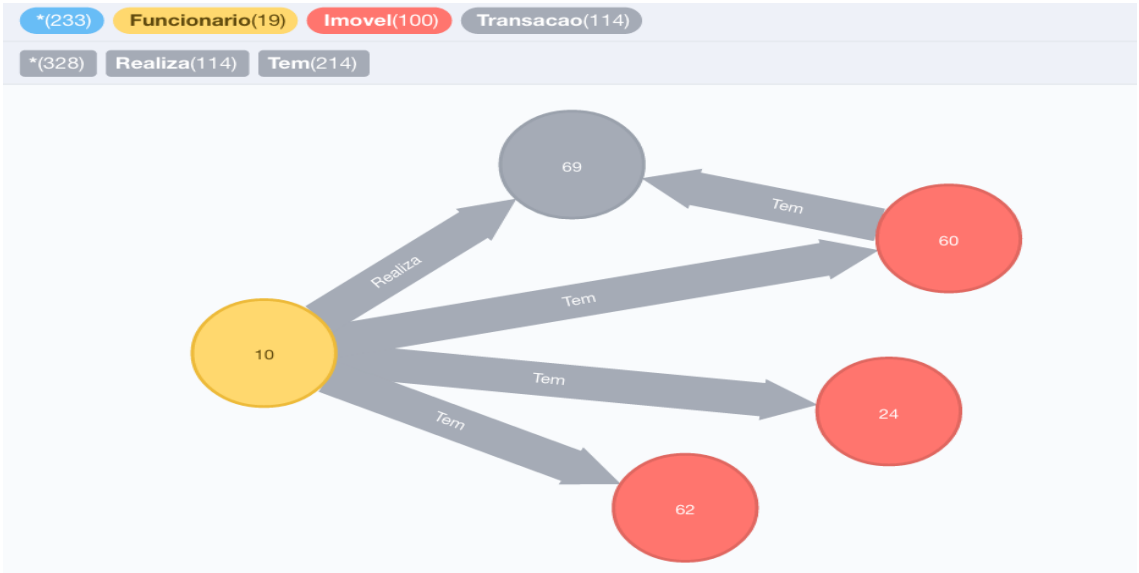

Figura 9 Relação Tem

Resultado final

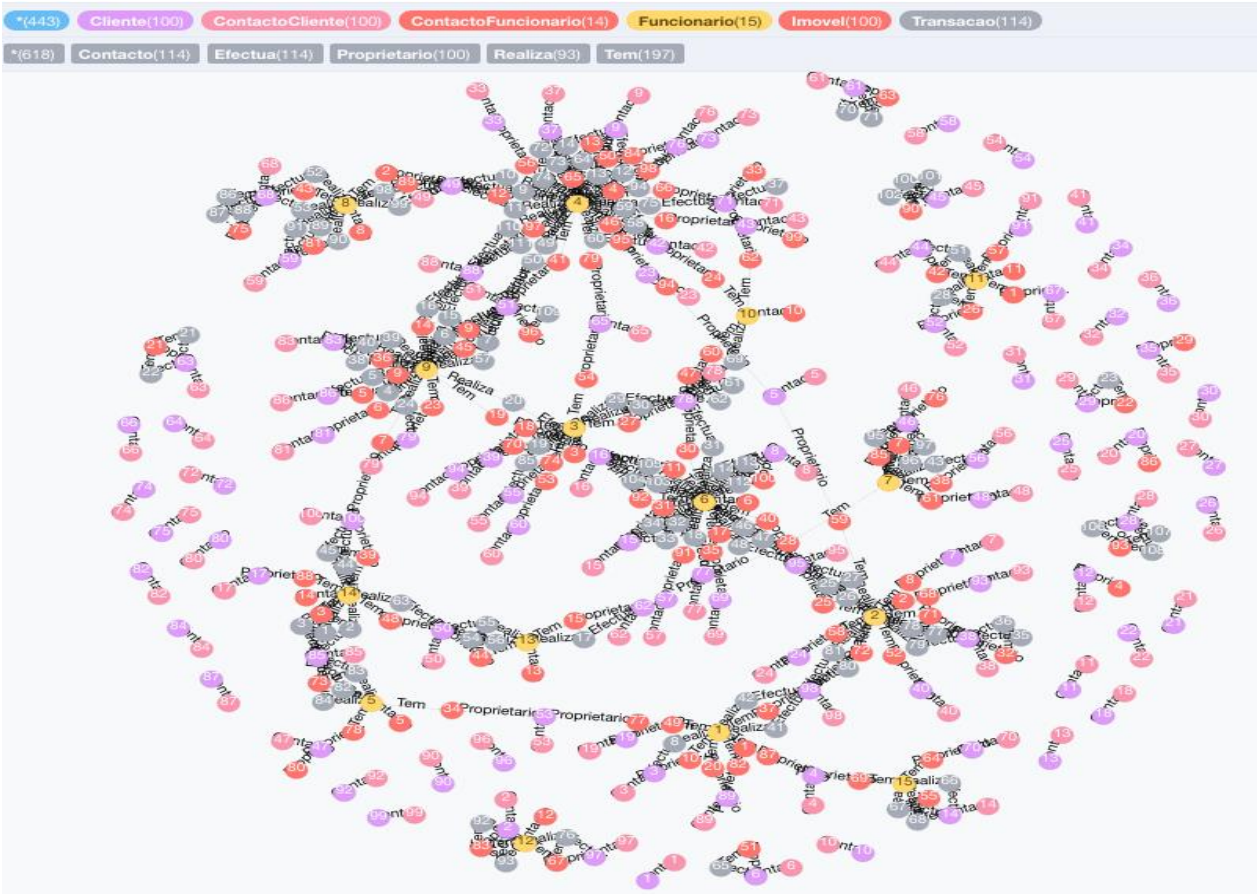

Figura 10 Base de Dados em Neo4J

1.2. Queries

Queries equivalentes em Neo4J.

1. **Mostrar os funcionários responsáveis pelas transações**

MySQL

```
SELECT F.idFuncionario as CodigoFuncionario, F.Nome
      FROM Transacao as T
            INNER JOIN Funcionario as F
                  ON T.idFuncionario = F.idFuncionario
        WHERE T.idTransacao = 1;
```

Neo4J

```
Match (t:Transacao)<-[:Realiza]-(f:Funcionario)
WHERE t.id = '1'
Return f.id as CodigoFuncionario ,f.nome as Nome;
```

2. **Mostrar o numero de imoveis que estão à venda**

MySQL

```
Select Count(idImovel)
      From Imovel
    WHERE Condicao = 'Venda';
```

Neo4J

```
MATCH (i:Imovel{Condicao:"Venda"})
RETURN count(i) AS contaPorVenda
```

3. **Mostrar o numero de imóveis disponíveis para alugar**

MySQL

```
Select Count(idImovel)
      From Imovel
    WHERE Condicao = 'Aluguer';
```

Neo4J

```
MATCH (i:Imovel{Condicao:"Aluguer"})
RETURN count(i) AS contaPorAlugar
```

4. **Mostrar o numero de imoveis marcados como vendidos**

MySQL

```
Select Count(idImovel)
      From Imovel
    WHERE Condicao = 'Vendido';
```

Neo4J

```
MATCH (i:Imovel{Condicao:"Vendido"})
RETURN count(i) AS contaVendidos
```

5. **Mostrar o numero de imoveis alugados**

MySQL

```
Select Count(idImovel)
      From Imovel
    WHERE Condicao = 'Alugado';
```

Neo4J

```
MATCH (i:Imovel{Condicao:"Alugado"})
RETURN count(i) AS contaAlugados
```

6. **Mostrar o valor do imóvel com o id=1**

MySQL

```
Select Valor
      From Imovel
    WHERE idImovel = 1;
```

Neo4J

```
MATCH (i:Imovel)
WHERE i.id = '1'
RETURN i.Valor;
```

7. **Mostrar os imóveis, em Braga, com valor superior ou igual a 50000**

MySQL

```
Select idImovel as Codigo , Morada , Valor
      From Imovel
    Where Morada like ('%Braga') and valor >= 50000;
```

Neo4J

```
MATCH (i:Imovel)
WHERE i.Morada =~ ".*Braga" AND TOINT(i.Valor) >=50000
RETURN i.id,i.Morada,i.Valor;
```

1.3. Análise

Usando a abordagem NoSQL, a implementação da nossa base de dados foi mais rápida e com menor esforço. Embora não consiga garantir a mesma consistência da abordagem relacional, sacrificando-a, permite à base de dados uma tolerância maior em termos de alterações.

Numa abordagem NoSQL, a base de dados do caso em questão poderia sofrer mudanças facilmente, como por exemplo, se fosse necessário adicionar outro atributo à entidade imóvel, apenas seria necessário alterar a estrutura de cada imóvel, enquanto que no modelo relacional esta alteração levaria á criação de uma nova de base dados e o seu devido povoamento.

A quando da criação das entidades em NoSQL adquirimos liberdade, podemos ter entidades com o mesmo nome, mas com atributos diferentes (o que também pode ter os seus problemas, por exemplo, um imóvel teria de manter sempre os mesmos atributos)

contrariamente ao que ocorreu na abordagem relacional, em que é necessário usar uma estrutura previamente bem pensada e que permaneça inalterada.

Em termos de *queries,* torna-se muito mais rápido fazer uma pesquisa de informação, numa base de dados não relacional baseada em grafos, uma vez que apenas se procura a informação nos nós que contêm certos relacionamentos e pode-se ignorar todos os outros. Ao contrário das bases de dados relacionais, onde se tem de percorrer toda a tabela para procurar a informação.

2. Conclusão

É incontornável falar do teorema CAP (Consistency, Availability, Partition tolerance). O Neo4J torna a implementação mais fácil e mais rápida, assim como as operações sobre a informação guardada, mas abdica da chamada consistência, o que significa que podemos não ter sempre a informação mais recente.

Olhando para o tema do nosso trabalho, facilmente percebemos que embora o modelo NoSQL nos trazia mais comodismos na implementação, não seria uma boa escolha devido ao facto de abdicar da consistência em prol da disponibilidade e da tolerância ao particionamento, porque queremos ter sempre o estado mais recente dos imóveis.

Concluímos então que o Neo4J tornou a implementação mais fácil em relação ao MySQL e também significativamente mais rápida. Porém, não seria a melhor escolha.

Referências

https://neo4j.com/

https://neo4j.com/news/introducao-aos-bancos-de-dados-nosql/

https://neo4j.com/developer/guide-importing-data-and-etl/

https://rstudio-pubs-static.s3.amazonaws.com/175345_f97da0204f2540f1b9d679ae09cd3407.html#load_data_into_neo4j

Lista de Siglas e Acrónimos

BD Base de Dados