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

Unidade Curricular de Bases de Dados

Ano Lectivo de 2017/2018

Migração de uma base dados relacional para uma base de dados não relacional

Carlos Pedrosa – a77320
David Sousa – a78938
Francisco Matos – a77688
Manuel Sousa – a78869

Janeiro, 2018

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Migração de uma base dados relacional para uma base de dados não relacional

Carlos Pedrosa – a77320
David Sousa – a78938
Francisco Matos – a77688
Manuel Sousa – a78869

Janeiro, 2018

Resumo

Este projeto tem como principal objetivo a migração de um sistema de gestão de base de dados relacional, produzido anteriormente, para um sistema de gestão de base de dados não relacional, neste caso, orientada a grafos*(NEO4J)*.

Desta forma, procuramos seguir um conjunto de passos, de modo a que, seja possível obter uma base de dados não relacional funcional e viável. Nesse sentido, começamos por identificar os objetivos da base de dados, transcrever, da forma que nos pareceu ser a mais adequada, o modelo lógico para um modelo orientado a grafos e codificar essas transformações em *Cypher*. Por último, desenvolvemos as *queries*(interrogações) implementadas no modelo relacional neste novo modelo.

Área de Aplicação: Desenho e arquitetura de Sistemas de Bases de Dados, migração de uma base de dados relacional para uma base de dados não relacional.
Palavras-Chave: Base de dados relacional e não relacional, *NEO4J, Cypher.*

Índice

Resumo	i
Índice	ii
Índice de Figuras	iii
Índice de Tabelas	iv
1. Introdução	1
1.1. Contextualização de aplicação do sistema	1
1.2. Fundamentação da implementação da base de dados	1
1.3. Motivação e Objetivos	2
1.4. Estrutura do Relatório	2
2. Definição do processo seguido	3
3. Migração da base de dados	4
3.1 Passagem do modelo anterior para o modelo orientado a grafos	4
3.2. Transcrição para a linguagem usada no NEO4J, Cypher	5
4. Migração das interrogações para o modelo não relacional	13
5. Análise comparativa entre SQL e NEO4J	16
6. Conclusões e Trabalho Futuro	17
Referências	18
Lista de Siglas e Acrónimos	19
Anexos	20

Anexos

I. Anexo 1	21

Índice de Figuras

Figura 1 - Modelo lógico obtido anteriormente 4
Figura 2 - Criação da tabela animal no modelo relacional e respetivo povoamento 6
Figura 3 - Transcrição da tabela animal para NEO4J 6
Figura 4 - Exemplo de resultado do animal em NEO4J 6
Figura 5 -Criação da tabela funcionario no modelo relacional e respetivo povoamento 7
Figura 6 – Transcrição da tabela funcionario para NEO4J 7
Figura 7 - Exemplo de resultado do funcionario em NEO4J 7
Figura 8 - Criação da tabela alimento no modelo relacional e respetivo povoamento 8
Figura 9 - Transcrição da tabela alimento para NEO4J 8
Figura 10 - Exemplo de resultado do Alimento em NEO4J 8
Figura 11 - Criação da tabela espaço no modelo relacional e respetivo povoamento 9
Figura 12 - Transcrição da tabela espaço para NEO4J 9
Figura 13 - Exemplo de resultado do Espaço em NEO4J 9
Figura 14 - Criação da tabela contacto no modelo relacional e respetivo povoamento 10
Figura 15 - Transcrição da tabela contacto para NEO4J 10
Figura 16 - Exemplo de resultado do Contacto em NEO4J 10
Figura 17 - Criação das Constraints necessárias 11
Figura 18 - Implementação de um relacionamento em NEO4J 11
Figura 19 - Modelo geral no NEO4J 12
Figura 20 - Interrogação implementada em MySQL 15
Figura 21 - Resultado da interrogação implementada em MySQL 15
Figura 22 - Interrogação implementada em Cypher 15
Figura 23 - Resultado da interrogação implementada em Cypher 15
Figura 24 - Gráfico comparativo entre Cypher e SQL 16

Índice de Tabelas

Tabela 1 - Transformação das Queries em Cypher 14

1. Introdução

No âmbito da unidade curricular de Base de Dados foi-nos proposto a análise, o planeamento e a implementação de um SGBD não relacional, tendo como base o tema escolhido numa primeira fase. Neste caso, o ambiente proposto foi o Neo4j, um motor de base de dados orientada a grafos (Graph Database). Assim, pretendemos demonstrar no presente relatório todos os passos dados no sentido a cumprir o objetivo proposto.

1.1. Contextualização de aplicação do sistema

Filipe, dono atual do zoológico, estava prestes a levar o Zoo para um novo patamar devido à parceria estabelecida com o Zoológico da Maia. O jardim zoológico iria ser expandido e receber mais de 100 espécies novas. Deixaria de ser um Zoo familiar e passaria para um Zoo ao nível dos mais grandiosos do país. Para tal, iriam ser criados novos postos de trabalho, as instalações seriam melhoradas, os espaços aumentados, as espécies iriam crescer, tornando real o sonho dos antepassados que têm passado o Zoológico de geração em geração.

1.2. Fundamentação da implementação da base de dados

Preocupado com o crescimento do Zoo, Filipe, sendo um pouco entendido em informática, percebeu que esse crescimento poderia causar uma quebra no que toca ao desempenho. De facto, o novo dono do Zoo percebeu que a base de dados começava a ficar um pouco obsoleta pelo que as interrogações que tinha pedido anteriormente, demoravam bastante tempo a retornar um resultado. Para o efeito, Sr. Filipe, satisfeito com o trabalho anterior, decidiu contratar novamente os profissionais que implementaram a base de dados e perguntar se haveria algum tipo de solução

para o problema. A resposta foi a migração da base de dados que possuía atualmente, para uma base de dados não relacional.

1.3. Motivação e Objetivos

Uma das motivações do Sr. Filipe recorrer novamente à ajuda de profissionais foi o facto de a sua parceria com o Zoológico da Maia trazer grandes mudanças, não só a nível económico, mas também no que diz respeito à capacidade do Zoo. Assim, o avultado número de espécies e consequente de funcionários requeriam uma base de dados especializada.

O principal objetivo na elaboração deste projeto é o desenvolvimento de um novo sistema de gestão de base de dados que irá permitir administrar de forma adequada os novos recursos do Jardim zoológico, para que assim seja possível:

- Responder de forma rápida e eficiente a todos os propósitos que anteriormente se verificaram.

1.4. Estrutura do Relatório

Depois de termos definidos as motivações que nos levaram à passagem do modelo relacional para o modelo lógico, procuramos agora descrever um pouco os passos que serão dados de seguida. Assim, num primeiro momento analisámos o modelo relacional desenvolvido e procuramos ver a forma como do modelo lógico poderíamos construir os grafos na plataforma *NEO4J.* De seguida, criamos em *Cypher* (linguagem usada por *NEO4J*) o necessário para que a migração ocorra da maneira mais correta. Por último, transformamos as interrogações desenvolvidas em *MySQL* e fizemos uma análise crítica relativamente aos dois tipos de modelos.

2. Definição do processo seguido

Num primeiro momento, decidimos identificar os objetivos da base de dados percebendo, assim, o tipo de necessidades que esta terá de responder. Para além disso, identificamos as diferentes entidades envolvidas a partir do modelo lógico obtido anteriormente. A partir destas constatamos os diferentes relacionamentos existentes e, por fim, convertemos as entidades e os relacionamentos em *paths*(caminhos), sendo que, estes são a base do modelo de dados usado. Numa análise às queries desenvolvidas, criamos ainda todos os procedimentos necessários para que essas possam ser respondidas em *NEO4J*. Todo o processo foi realizado de forma manual, isto é, sem o auxílio de ferramenta alguma para que, desta forma, seja mais percetível a implementação da base de dados.

3. Migração da base de dados

3.1 Passagem do modelo anterior para o modelo orientado a grafos

No modelo lógico percebemos que cada entidade da tabela é representada por uma *label*(etiqueta) nos nodos. Para além disso, cada entrada da tabela, no modelo físico, também será um nodo. Colunas de uma tabela, também no modelo físico, são propriedades dos nós. Neste seguimento, removemos as chaves primárias que nos pareceram, de certo modo, técnicas adicionando, no entanto, restrições para as chaves primárias que realmente eram necessárias. Uma vez que, neste modelo não existem as chaves estrangeiras, tivemos em conta, os diversos tipos de relacionamentos e de acordo com estes criamos diferentes relações entre os nodos dos grafos. Nesse sentido, no modelo abaixo representamos o que será aproximadamente implementado numa base de dados orientada a grafos.

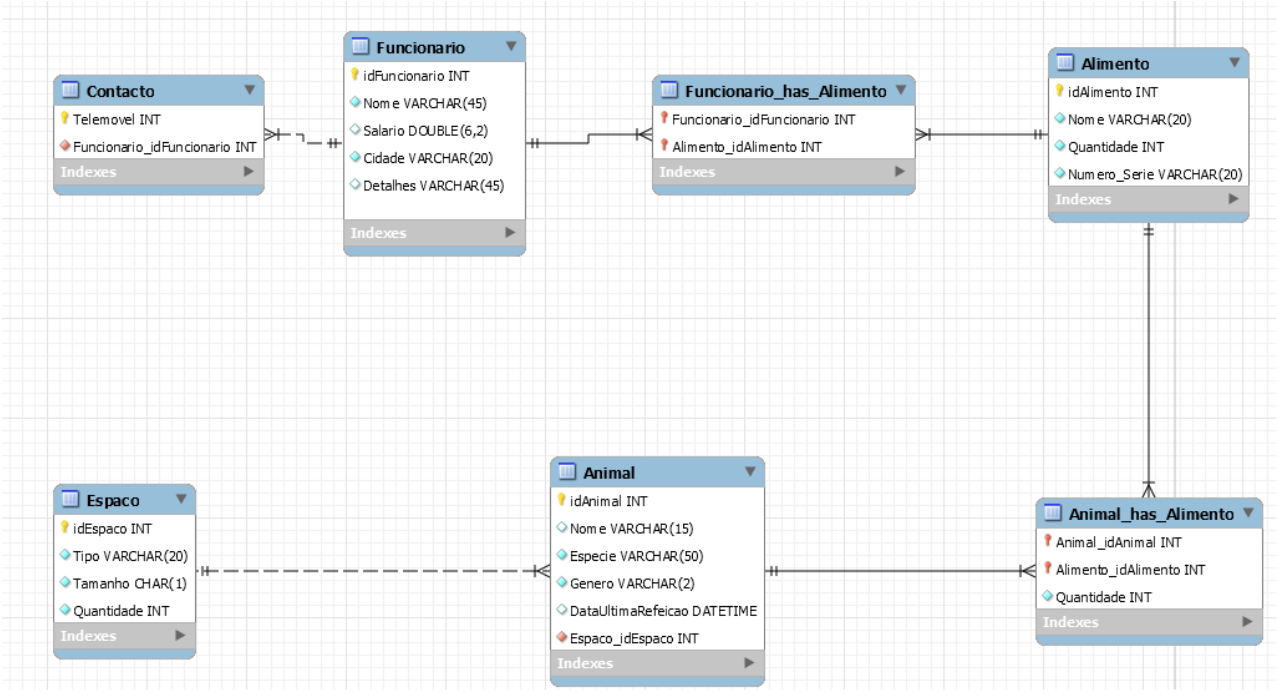
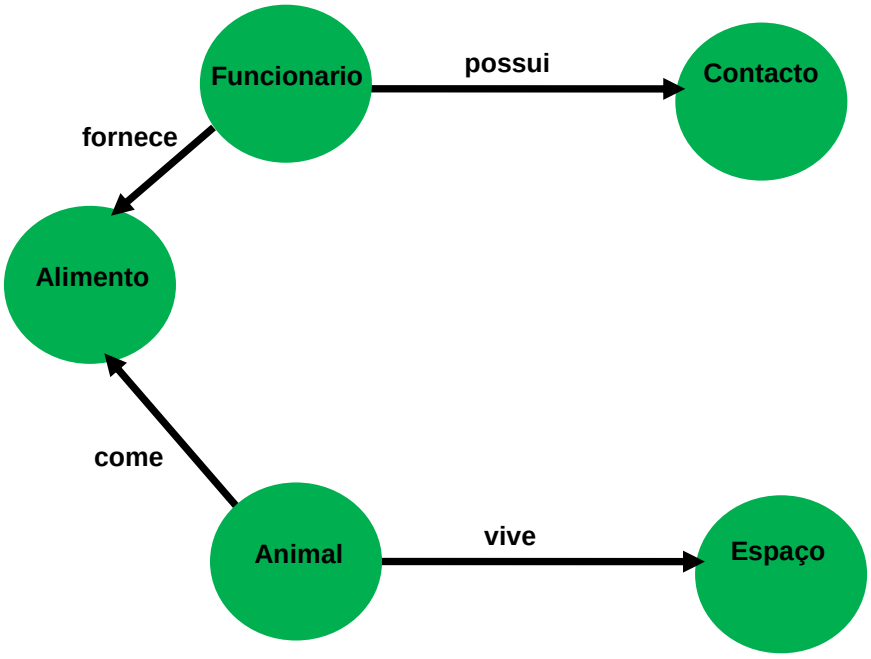


Figura 1 - Modelo lógico obtido anteriormente



3.2. Transcrição para a linguagem usada no *NEO4J*, *Cypher*

Depois de identificarmos quais as entidades e quais os relacionamentos no modelo não relacional, decidimos transcrever a linguagem usada no *MySQL Workbench* para Cypher. Nesse sentido, apresentamos de seguida todos os passos que demos para obter a base de dados não relacional.

```
CREATE TABLE IF NOT EXISTS `ZooDB`.`Animal` (
  `idAnimal` INT NOT NULL,
  `Nome` VARCHAR(15) NULL,
  `Especie` VARCHAR(50) NOT NULL,
  `Genero` VARCHAR(2) NOT NULL,
  `DataUltimaRefeicao` DATETIME NULL,
  `Espaco_idEspaco` INT NOT NULL,
  PRIMARY KEY (`idAnimal`, `Espaco_idEspaco`),
  INDEX `fk_Animal_Espaco1_idx` (`Espaco_idEspaco` ASC),
  CONSTRAINT `fk_Animal_Espaco1`
    FOREIGN KEY (`Espaco_idEspaco`)
    REFERENCES `ZooDB`.`Espaco` (`idEspaco`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```sql
INSERT INTO Animal
    (idAnimal, Nome, Especie, Genero,DataUltimaRefeicao, Espaco_idEspaco)
    VALUES
        (1,'Joao','Panthera leo','M',NULL,1),
        (2,'Oscar','Orcinus orca','M','2017-11-24',2),
        (3,'Ricardo','Panthera pardus','M','2017-11-22',1),
        (4,'Napoleão','Balaenoptera physalus','M',Null,2),
        (5,'Paula','Equus asinus','F',NULL,1),
        (6,'Kika','Macropus rufus','F','2017-11-21',3),
        (7,'Laila','Pan troglodytes','F',NULL,1),
        (8,'Maria','Loxodonta africana','F','2017-11-25',1),
        (9,'Eddy','Ailuropoda melanoleuca','M','2017-11-23',2),
        (10,'Lara','Carcharodon carcharias','F',Null,2),
        (11,'Laila','Apis mellifera','F',NULL,1);
```

Figura 2 - Criação da tabela animal no modelo relacional e respetivo povoamento

```
CREATE (Leão:Animal {nome:'Joao',especie:'Panthera leo', genero:'M',DataUltimaRefeicao:0})
CREATE (Orca:Animal {nome:'Oscar',especie:'Orcinus orca', genero:'M',DataUltimaRefeicao:'2017-11-24 22:36:12'})
CREATE (Leopardo:Animal {nome:'Ricardo',especie:'Panthera Pardus', genero:'M',DataUltimaRefeicao:'2017-11-25 11:21:53'})
CREATE (Baleia:Animal {nome:'Napoleão',especie:'Balaenoptera physalus', genero:'M',DataUltimaRefeicao:'0'})
CREATE (Burro:Animal {nome:'Paula',especie:'Equus asinus', genero:'F',DataUltimaRefeicao:'0'})
CREATE (Kangaroo:Animal {nome:'Kika',especie:'Macropus rufus', genero:'F',DataUltimaRefeicao:'2017-11-23 16:45:21'})
CREATE (Chimpazé:Animal {nome:'Laila',especie:'Pan troglodytes', genero:'F',DataUltimaRefeicao:'2017-11-23 16:45:21'})
CREATE (Elefante:Animal {nome:'Maria',especie:'Loxodonta africana', genero:'F',DataUltimaRefeicao:'2017-11-25 07:43:54'})
CREATE (Panda:Animal {nome:'Eddy',especie:'Ailuropoda melanoleuca', genero:'M',DataUltimaRefeicao:'2017-11-23 12:48:32'})
CREATE (Tubarão:Animal {nome:'Lara',especie:'Carcharodon carcharia', genero:'F',DataUltimaRefeicao:0})
CREATE (Abelha:Animal {nome:'Laila',especie:'Apis mellifera', genero:'F',DataUltimaRefeicao:0})
```

Figura 3 - Transcrição da tabela animal para *NEO4J*

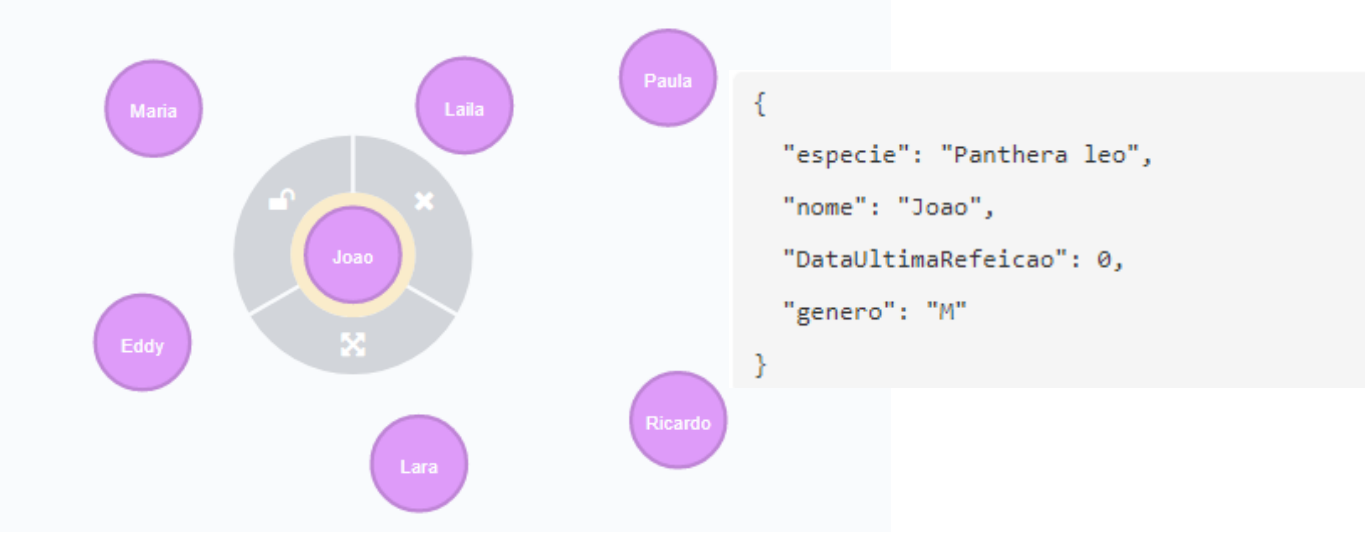


Figura 4 - Exemplo de resultado do animal em *NEO4J*

```
CREATE TABLE IF NOT EXISTS `ZooDB`.`Funcionario` (
  `idFuncionario` INT NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Salario` DOUBLE(6,2) NULL,
  `Cidade` VARCHAR(20) NOT NULL,
  `Detalhes` VARCHAR(45) NULL,
  PRIMARY KEY (`idFuncionario`))
ENGINE = InnoDB;
```

```
INSERT INTO funcionario
    (idfuncionario, Nome, Salario, Cidade, Detalhes)
    VALUES
        (1,'André Marques',250,'Braga','Tenões'),
        (2,'Mariana Fonte',2000,'Porto','Estádio do Dragao'),
        (3,'João Castro',9500,'Braga','Gualtar'),
        (4,'Nuno Gonçalves',700,'Lisboa','Praça do Rato'),
        (5,'Ricardo Martins',1200,'Vila Franca de Xira',Null),
        (6,'Inês Ferreira',9500,'Guimarães','Partes desconhecidas');
```

Figura 5 -Criação da tabela funcionario no modelo relacional e respetivo povoamento

```
CREATE (AndreM:Funcionario {idFuncionario: 1, nome: 'André Marques', salario: 250, cidade: 'Braga', detalhes: 'Tenões'})
CREATE (MarianaF:Funcionario {idFuncionario: 2, nome: 'Mariana Fonte', salario: 2000, cidade: 'Porto', detalhes: 'Estádio do Dragão'})
CREATE (FilipeS:Funcionario {idFuncionario: 3, nome: 'Filipe Sousa', salario: 9500, cidade: 'Braga', detalhes: 'Partes desconhecidas'})
CREATE (NunoG:Funcionario {idFuncionario: 4, nome: 'Nuno Gonçalves', salario: 700, cidade: 'Lisboa', detalhes: 'Praça do Rato'})
CREATE (RicardoM:Funcionario {idFuncionario: 5, nome: 'Ricardo Martins', salario: 1200, cidade: 'Vila Franca de Xira', detalhes: ''})
CREATE (InesF:Funcionario {idFuncionario: 6, nome: 'Inês Ferreira', salario: 9500, cidade: 'Guimarães', detalhes: 'Partes desconhecidas'})
```

Figura 6 – Transcrição da tabela funcionario para *NEO4J*

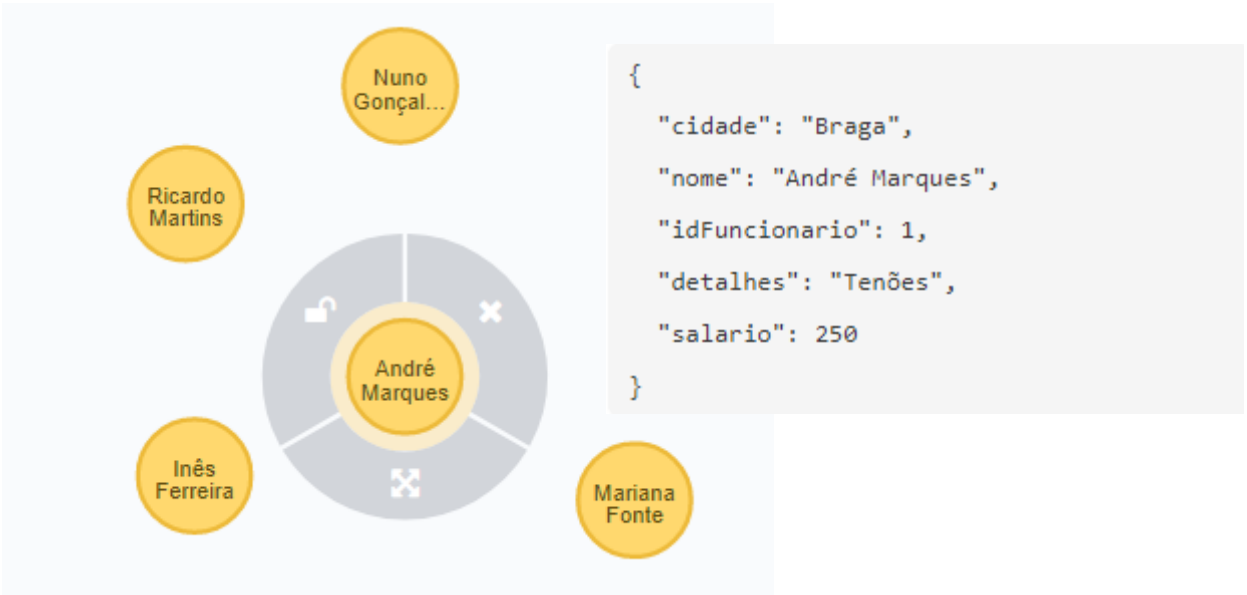


Figura 7 - Exemplo de resultado do funcionario em NEO4J

```sql
CREATE TABLE IF NOT EXISTS `ZooDB`.`Alimento` (
  `idAlimento` INT NOT NULL,
  `Nome` VARCHAR(20) NOT NULL,
  `Quantidade` INT NOT NULL,
  `Numero_Serie` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`idAlimento`))
ENGINE = InnoDB;

INSERT INTO Alimento
    (idAlimento, Nome, Quantidade, Numero_Serie)
    VALUES
        (1,'Frango',10,'4312'),
        (2,'Peixinhos',230,'2042'),
        (3,'Frutos',589,'324'),
        (4,'Bamboo',20,'1324'),
        (5,'Atum',1000,'642');
```

Figura 8 - Criação da tabela alimento no modelo relacional e respetivo povoamento

```
CREATE (Frango:Alimento {nome: 'Frango', quantidade: 10, numero_Serie: '4312'})
CREATE (Peixinhos:Alimento {nome: 'Peixinhos', quantidade: 230, numero_Serie: '2042'})
CREATE (Frutos:Alimento {nome: 'Frutos', quantidade: 589, numero_Serie: '324'})
CREATE (Bamboo:Alimento {nome: 'Bamboo', quantidade: 20, numero_Serie: '1324'})
CREATE (Atum:Alimento {nome: 'Atum', quantidade: 1000, numero_Serie: '624'})
```

Figura 9 - Transcrição da tabela alimento para *NEO4J*

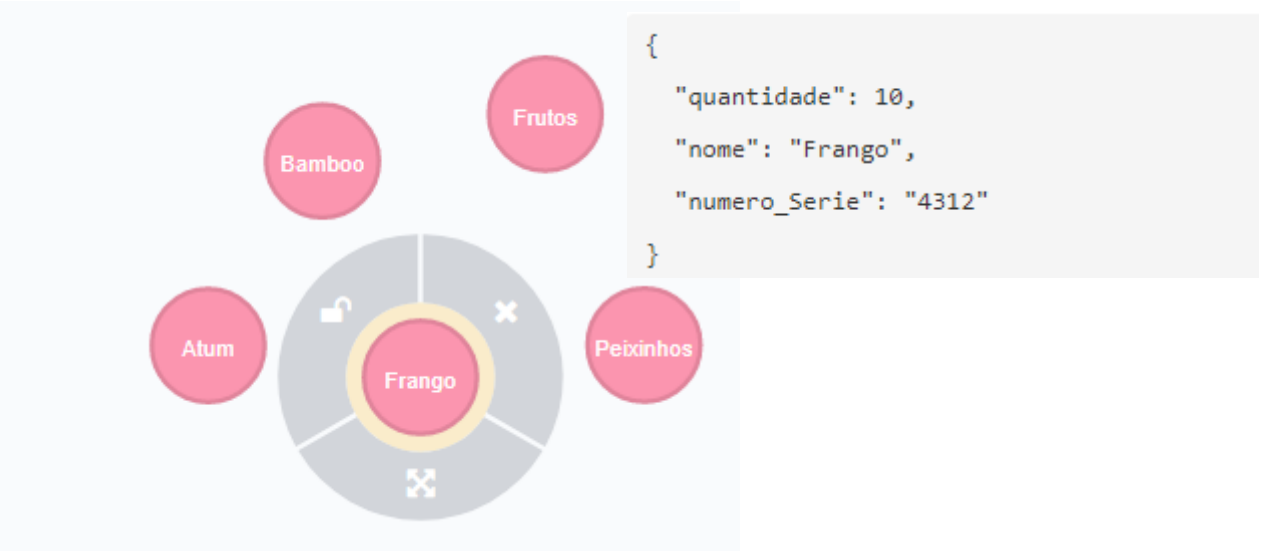


Figura 10 - Exemplo de resultado do Alimento em NEO4J

```sql
CREATE TABLE IF NOT EXISTS `ZooDB`.`Espaco` (
  `idEspaco` INT NOT NULL,
  `Tipo` VARCHAR(20) NOT NULL,
  `Tamanho` VARCHAR(10) NOT NULL,
  `Quantidade` INT NOT NULL,
  PRIMARY KEY (`idEspaco`))
ENGINE = InnoDB;
```

```sql
INSERT INTO Espaco
    (idEspaco, Tipo, Tamanho, Quantidade)
    VALUES
        (1,'Jaula','Grande',9),
        (2,'Aquario','Pequeno',20),
        (3, 'Jaula', 'Medio',4);
```

Figura 11 - Criação da tabela espaço no modelo relacional e respetivo povoamento

```
CREATE (Jaula:Espaco {tipo:'Jaula',tamanho:'G',quantidade:9,idEspaco:'1'})
CREATE (Aquario:Espaco {tipo:'Aquario',tamanho:'P',quantidade:20,idEspaco:'2'})
CREATE (Jaula2:Espaco {tipo:'Jaula',tamanho:'M',quantidade:4,idEspaco:'3'})
```

Figura 12 - Transcrição da tabela espaço para *NEO4J*

Figura 13 - Exemplo de resultado do Espaço em *NEO4J*

```
CREATE TABLE IF NOT EXISTS `ZooDB`.`Contacto` (
  `Telemovel` INT NOT NULL,
  `Funcionario_idFuncionario` INT NOT NULL,
  PRIMARY KEY (`Telemovel`),
  CONSTRAINT `fk_Contacto_Funcionario`
    FOREIGN KEY (`Funcionario_idFuncionario`)
    REFERENCES `ZooDB`.`Funcionario` (`idFuncionario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
INSERT INTO contacto
    (Telemovel, funcionario_idfuncionario)
    VALUES
        (911121121,1),
        (962345678,2),
        (913323321,3),
        (934442454,4),
        (916231524,1),
        (932136555,5),
        (917646664,6);
```

Figura 14 - Criação da tabela contacto no modelo relacional e respetivo povoamento

```
CREATE (Andre1:Contacto {telemovel:911121121})
CREATE (Mariana1:Contacto {telemovel:962345678})
CREATE (Filipe1:Contacto {telemovel:913323321})
CREATE (Nuno1:Contacto {telemovel:934442454})
CREATE (Andre2:Contacto {telemovel:916231524})
CREATE (Ricardo1:Contacto {telemovel:932136555})
CREATE (Ines1:Contacto {telemovel:917646664})
```

Figura 15 - Transcrição da tabela contacto para *NEO4J*

Figura 16 - Exemplo de resultado do Contacto em *NEO4J*

De forma a manter a integridade da base de dados, decidimos criar as restrições necessárias.

```
CREATE CONSTRAINT ON (e:Espaco) ASSERT e.idEspaco IS UNIQUE;
CREATE CONSTRAINT ON (f:Funcionario) ASSERT f.idFuncionario IS UNIQUE;
CREATE CONSTRAINT ON (a:alimento) ASSERT a.nome IS UNIQUE;
```

Figura 17 - Criação das *Constraints* necessárias

Depois de construida e povoados os dados em *NEO4J* passamos à implementação dos diferentes relacionamentos entre as entidades. Assim, um relacionamento N:N no caso do MySQL leva à criação de uma nova tabela, no caso do *Cypher* as *JoinTables* são guardadas nas relacões. Assim, definimos de seguida algumas das relações que implemetamos.

```
CREATE
  (AndreM)-[:Fornece]->(Frango)
  (MarianaF)-[:Fornece]->(Frango)
  (FilipeS)-[:Fornece]->(Peixinhos)
  (FilipeS)-[:Fornece]->(Frutos)
  (NunoG)-[:Fornece]->(Bamboo)
  (RicardoM)-[:Fornece]->(Atum)
  (RicardoM)-[:Fornece]->(Frango)
  (InesF)-[:Fornece]->(Atum)
```

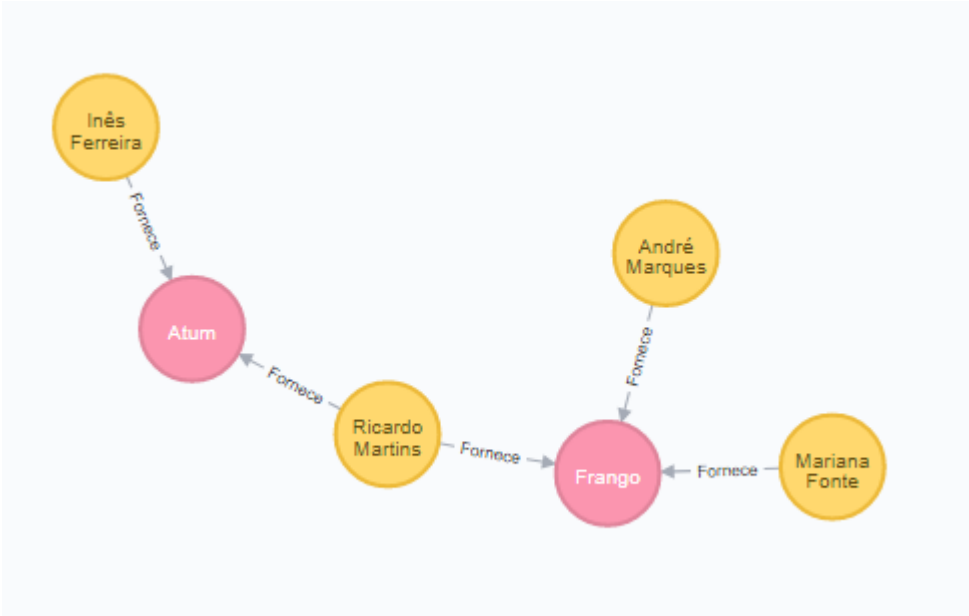


Figura 18 - Implementação de um relacionamento em *NEO4J*

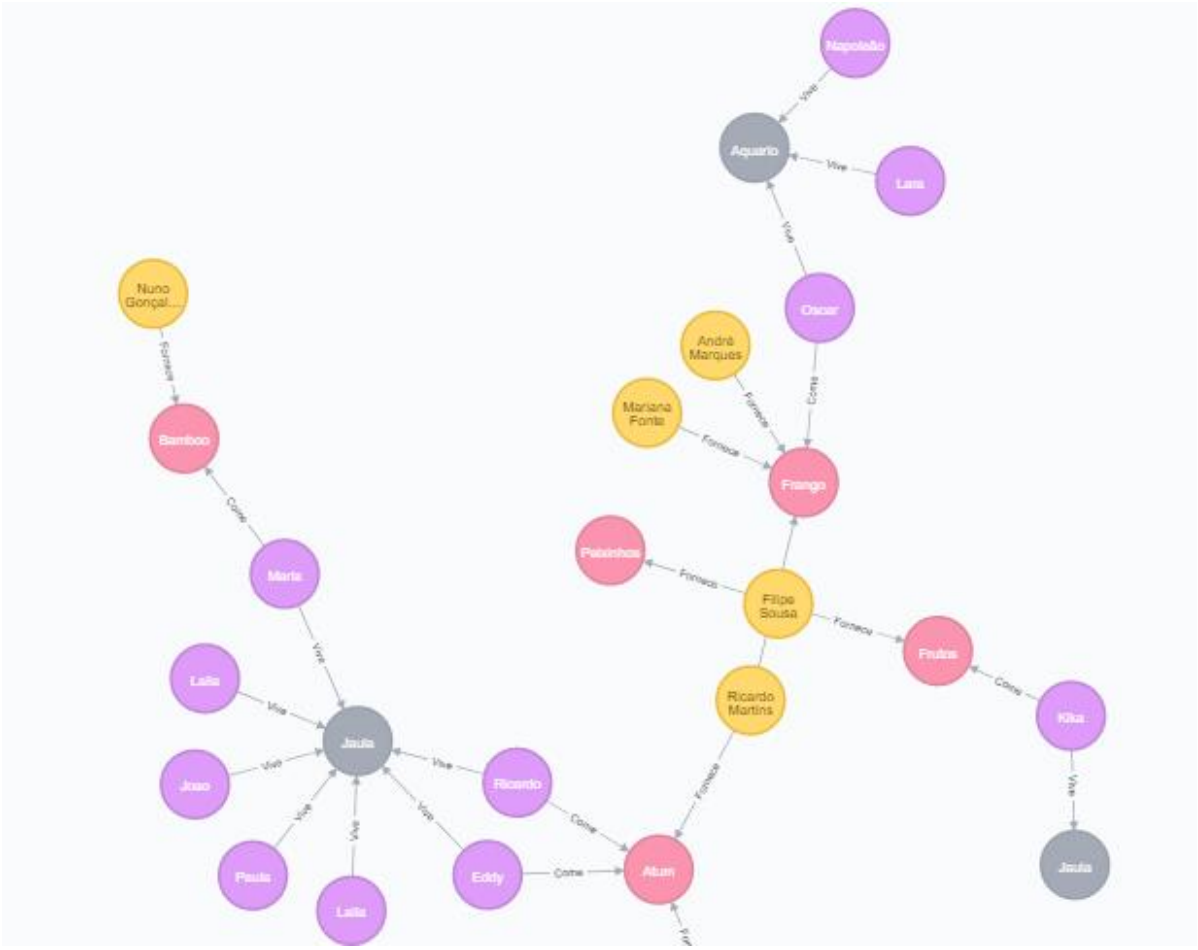


Figura 19 - Modelo geral no *NEO4J*

4. Migração das interrogações para o modelo não relacional

Analisando as interrogações desenvolvidas em *SQL* decidimos transformá-las em *Cypher* por forma a que a base de dados no modelo não relacional pudesse responder de maneira semelhante ao modelo relacional. Nesse sentido, exemplificamos a passagem das diferentes queries de *MySQL* para *NEO4J* através da seguinte tabela.

SQL	Cypher
SELECT * FROM Alimento WHERE quantidade <= 100;	MATCH (a:Alimento) WHERE a.quantidade >= 100 RETURN a;
SELECT idAnimal, nome FROM Animal INNER JOIN Espaco ON animal.Espaco_idEspaco = Espaco.idEspaco WHERE Espaco.tipo = 'Jaula';	MATCH (an:Animal)-[Vive]-(:Espaco {tipo:'Jaula'}) RETURN an;
SELECT idFuncionário, Nome, Cidade, Salario FROM funcionário WHERE Cidade = 'Braga' ORDER BY Salario;	MATCH (func:Funcionario) WHERE func.cidade = 'Braga' RETURN func ORDER BY func.salario ;
SELECT idEspaco, tipo FROM Espaco WHERE tipo='jaula' AND Quantidade >= 3;	MATCH (es:Espaco) WHERE es.tipo = 'Jaula' AND es.quantidade >= 3 RETURN es;
SELECT idAnimal, Nome FROM Animal INNER JOIN animal_has_alimento ON animal.idAnimal = Animal_idAnimal WHERE animal_has_alimento.Quantidade >5;	MATCH (an:Animal)-[c:Come]-(:Alimento) WHERE c.quantidade>5 RETURN an;
SELECT idFuncionario, Funcionario.Nome FROM Alimento INNER JOIN Funcionario_has_Alimento ON Alimento.idAlimento = Alimento_idAlimento INNER JOIN Funcionario ON Funcionario.idFuncionario = Funcionario_idFuncionario WHERE Alimento.Nome = 'Frango';	MATCH (f:Funcionario)-[Fornece]-(al:Alimento {nome:'Frango'}) RETURN f,al;

Tabela 1 - Transformação das *Queries* em *Cypher*

A título exemplificativo demonstramos, de seguida, o resultado produzido pela mesma interrogação em SQL e posteriormente em Cypher.

```
SELECT idFuncionario, funcionario.Nome
    FROM Alimento
        INNER JOIN Funcionario_has_alimento
            ON ALimento.idAlimento = Alimento_idAlimento
                INNER JOIN Funcionario
                    ON Funcionario.idFuncionario = Funcionario_idFuncionario
                        WHERE alimento.Nome = 'Frango';
```

Figura 20 - Interrogação implementada em *MySQL*

	idFuncionario	Nome
	1	André Marques
	2	Mariana Fonte
	5	Ricardo Martins

Figura 21 - Resultado da interrogação implementada em *MySQL*

```
MATCH (f:Funcionario)-[Fornece]-(al:Alimento {nome:'Frango'}) RETURN f,al;
```

Figura 22 - Interrogação implementada em *Cypher*

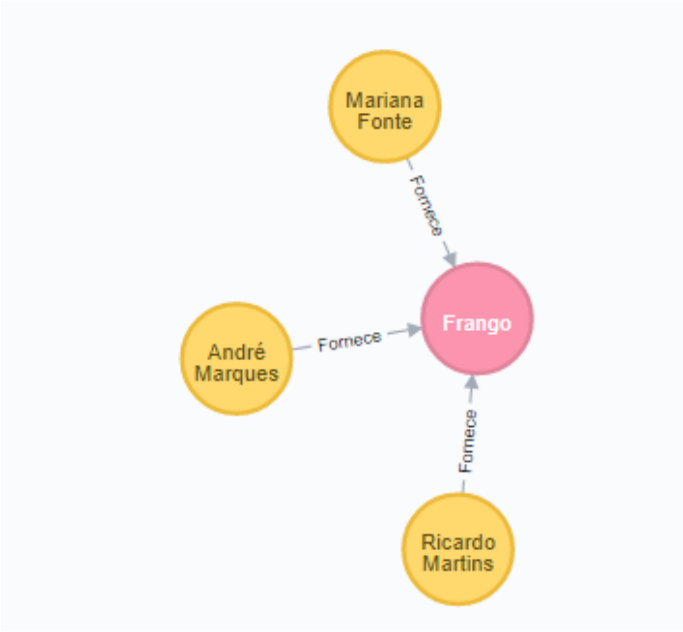


Figura 23 - Resultado da interrogação implementada em *Cypher*

5. Análise comparativa entre *SQL* e *NEO4J*

De acordo com o gráfico apresentado, verificamos que mesmo sendo uma base de dados de pequenas dimensões a diferença nos tempos é bastante acentuada.
Para além disso, verificamos na base de dados anterior o tamanho de armazenamento era de aproximadamente 2119 Bytes, sendo que, no caso apresentado em Cypher o tamanho da base de dados não relacional é na ordem dos 352.84 KiB[1]. Uma diferença colossal no que toca à comparação dos dois modelos de base de dados.

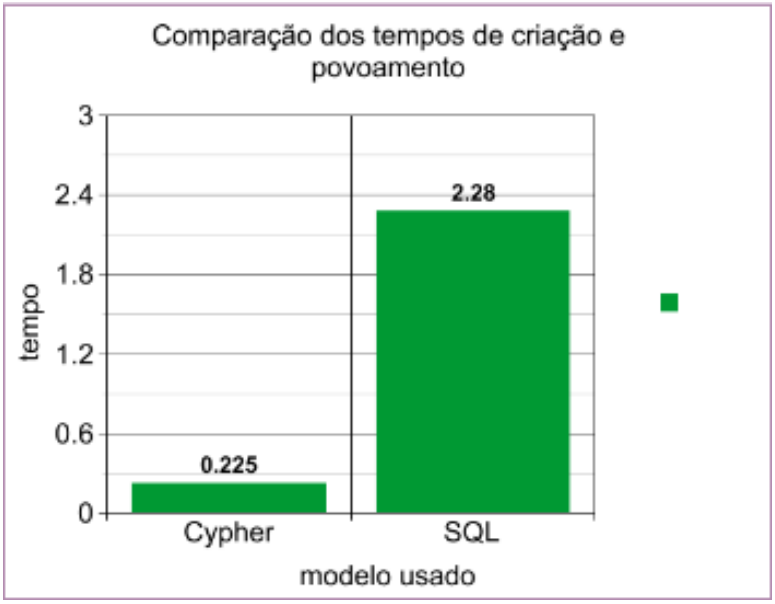


Figura 24 - Gráfico comparativo entre *Cypher e SQL*

[1] Um Kibibyte é igual a 1024 bytes

6. Conclusões e Trabalho Futuro

A migração de uma base de dados relacional para uma base de dados não relacional foi um processo que aconteceu naturalmente, sem grandes complexidades, no entanto, é necessário referir alguns aspetos que se sucederam nesta transição.

De facto, apesar do enquadramento para a utilização de uma base de dados relacional, não é possível demonstrar o verdadeiro potencial deste tipo de bases de dados já que estas exigem quantidades de dados bastante extensas. Contudo, com o exemplo por nós apresentado, foi-nos possível verificar que comparativamente ao modelo relacional esta base de dados permite um melhor desempenho relativamente às interrogações apresentadas. Para além disso, os resultados obtidos são apresentados de forma mais gráfica facilitando a compreensão por parte dos diferentes utilizadores. De salientar que futuramente com esta migração da base de dados o aumento populacional do Zoo poderá ser adicionado mais facilmente devido à flexibilidade do modelo não relacional.

Em suma, revelou-se um projeto bastante interessante pois permitiu entender um pouco mais sobre bases de dados não relacionais e quais são as suas vantagens e desvantagens.

Referências

- https://neo4j.com/blog/rdbms-neo4j-etl-tool/ acedido no dia 12/01/2018 pelas 17h;
- https://neo4j.com/blog/data-migration-between-mysql-and-neo4j/ acedido no dia 12/01/2018 pelas 15h23m.

Lista de Siglas e Acrónimos

BD	Base de Dados
SGBD	Sistema de Gestão de Base de Dados
SQL	*Structured Query Language*

Anexos

I. Anexo 1

CREATE (Leão:Animal {nome: 'Joao', especie: 'Panthera leo', genero: 'M', DataUltimaRefeicao: 0})
CREATE (Orca:Animal {nome: 'Oscar', especie: 'Orcinus orca', genero: 'M', DataUltimaRefeicao: '2017-11-24 22:36:12'})
CREATE (Leopardo:Animal {nome: 'Ricardo', especie: 'Panthera Pardus', genero: 'M', DataUltimaRefeicao: '2017-11-25 11:21:53'})
CREATE (Baleia:Animal {nome: 'Napoleão', especie: 'Balaenoptera physalus', genero: 'M', DataUltimaRefeicao: '0'})
CREATE (Burro:Animal {nome: 'Paula', especie: 'Equus asinus', genero: 'F', DataUltimaRefeicao: '0'})
CREATE (Kangaroo:Animal {nome: 'Kika', especie: 'Macropus rufus', genero: 'F', DataUltimaRefeicao: '2017-11-23 16:45:21'})
CREATE (Chimpazé:Animal {nome: 'Laila', especie: 'Pan troglodytes', genero: 'F', DataUltimaRefeicao: '2017-11-23 16:45:21'})
CREATE (Elefante:Animal {nome: 'Maria', especie: 'Loxodonta africana', genero: 'F', DataUltimaRefeicao: '2017-11-25 07:43:54'})
CREATE (Panda:Animal {nome: 'Eddy', especie: 'Ailuropoda melanoleuca', genero: 'M', DataUltimaRefeicao: '2017-11-23 12:48:32'})
CREATE (Tubarão:Animal {nome: 'Lara', especie: 'Carcharodon carcharia', genero: 'F', DataUltimaRefeicao: 0})
CREATE (Abelha:Animal {nome: 'Laila', especie: 'Apis mellifera', genero: 'F', DataUltimaRefeicao: 0})
CREATE (Jacare:Animal {nome: 'Tomé', especie: 'Caiman crocodilus', genero: 'M', DataUltimaRefeicao: '2017-11-24 21:28:52'})
CREATE (Girafa:Animal {nome: 'Susana', especie: 'Giraffa camelopardalis', genero:'F', DataUltimaRefeicao: 0})
CREATE (Lontra:Animal {nome: 'Jorge', especie:'Lutra longicaudis', genero:'M', DataUltimaRefeicao: '2017-11-25 14:13:51'})

CREATE (Camelo:Animal {nome: 'Areias', especie:'Struthio camelus', genero:'M', DataUltimaRefeicao: '2017-11-18 09:03:32'})

CREATE (AndreM:Funcionario {idFuncionario: 1, nome: 'André Marques', salario: 250, cidade: 'Braga', detalhes: 'Tenões'})
CREATE (MarianaF:Funcionario {idFuncionario: 2, nome: 'Mariana Fonte', salario: 2000, cidade: 'Porto', detalhes: 'Estádio do Dragão'})
CREATE (FilipeS:Funcionario {idFuncionario: 3, nome: 'Filipe Sousa', salario: 9500, cidade: 'Braga', detalhes: 'Partes desconhecidas'})
CREATE (NunoG:Funcionario {idFuncionario: 4, nome: 'Nuno Gonçalves', salario: 700, cidade: 'Lisboa', detalhes: 'Praça do Rato'})
CREATE (RicardoM:Funcionario {idFuncionario: 5, nome: 'Ricardo Martins', salario: 1200, cidade: 'Vila Franca de Xira', detalhes: ''})
CREATE (InesF:Funcionario {idFuncionario: 6, nome: 'Inês Ferreira', salario: 9500, cidade: 'Guimarães', detalhes: 'Partes desconhecidas'})

CREATE (Frango:Alimento {nome: 'Frango', quantidade: 10, numero_Serie: '4312'})
CREATE (Peixinhos:Alimento {nome: 'Peixinhos', quantidade: 230, numero_Serie: '2042'})
CREATE (Frutos:Alimento {nome: 'Frutos', quantidade: 589, numero_Serie: '324'})
CREATE (Bamboo:Alimento {nome: 'Bamboo', quantidade: 20, numero_Serie: '1324'})
CREATE (Atum:Alimento {nome: 'Atum', quantidade: 1000, numero_Serie: '624'})
CREATE (Palha:Alimento {nome: 'Palha', quantidade: 878, numero_Serie: '2138'})

CREATE (Jaula:Espaco {tipo: 'Jaula', tamanho: 'G', quantidade: 9, idEspaco: '1'})
CREATE (Aquario:Espaco {tipo:'Aquario', tamanho: 'P', quantidade: 20, idEspaco: '2'})
CREATE (Jaula2:Espaco {tipo: 'Jaula', tamanho: 'M', quantidade: 4, idEspaco: '3'})
CREATE (Ilha:Espaco {tipo: 'Ilha', tamanho: 'M', quantidade: 10, idEspaco: '4'})

CREATE (Andre1:Contacto {telemovel: 911121121})
CREATE (Mariana1:Contacto {telemovel: 962345678})
CREATE (Filipe1:Contacto {telemovel: 913323321})
CREATE (Nuno1:Contacto {telemovel: 934442454})
CREATE (Andre2:Contacto {telemovel: 916231524})
CREATE (Ricardo1:Contacto {telemovel: 932136555})
CREATE (Ines1:Contacto {telemovel: 917646664})

```
CREATE
  (Orca)-[:Come {quantidade: 4}]->(Frango),
  (Leopardo)-[:Come {quantidade: 26}]->(Atum),
  (Kangaroo)-[:Come {quantidade: 589}]->(Frutos),
  (Elefante)-[:Come {quantidade: 17}]->(Bamboo),
  (Panda)-[:Come {quantidade: 13}]->(Atum),
  (Jacare)-[:Come {quantidade: 21}]->(Atum),
  (Lontra)-[:Come {quantidade: 10}]->(Frutos),
  (Camelo)-[:Come {quantidade: 56}]->(Palha)

CREATE
  (AndreM)-[:Possui]->(Andre1),
  (MarianaF)-[:Possui]->(Mariana1),
  (FilipeS)-[:Possui]->(Filipe1),
  (NunoG)-[:Possui]->(Nuno1),
  (AndreM)-[:Possui]->(Andre2),
  (RicardoM)-[:Possui]->(Ricardo1),
  (InesF)-[:Possui]->(Ines1)

CREATE
  (Leão)-[:Vive]->(Jaula),
  (Orca)-[:Vive]->(Aquario),
  (Leopardo)-[:Vive]->(Jaula),
  (Baleia)-[:Vive]->(Aquario),
  (Burro)-[:Vive]->(Jaula),
  (Kangaroo)-[:Vive]->(Jaula2),
  (Chimpazé)-[:Vive]->(Jaula),
  (Elefante)-[:Vive]->(Jaula),
  (Panda)-[:Vive]->(Jaula),
  (Tubarão)-[:Vive]->(Aquario),
  (Abelha)-[:Vive]->(Jaula),
  (Jacaré)-[:Vive]->(Ilha),
  (Girafa)-[:Vive]->(Jaula),
  (Lontra)-[:Vive]->(Ilha),
  (Camelo)-[:Vive]->(Jaula2)

CREATE
```

```
(AndreM)-[:Fornece]->(Frango),
(MarianaF)-[:Fornece]->(Frango),
(FilipeS)-[:Fornece]->(Peixinhos),
(FilipeS)-[:Fornece]->(Frutos),
(NunoG)-[:Fornece]->(Bamboo),
(NunoG)-[:Fornece]->(Palha),
(RicardoM)-[:Fornece]->(Atum),
(RicardoM)-[:Fornece]->(Frango),
(InesF)-[:Fornece]->(Atum)

//CONSTRAINTS PARA ALGUNS ATRIBUTOS
CREATE CONSTRAINT ON (e:Espaco) ASSERT e.idEspaco IS UNIQUE;
CREATE CONSTRAINT ON (f:Funcionario) ASSERT f.idFuncionario IS UNIQUE;
CREATE CONSTRAINT ON (a:Alimento) ASSERT a.nome IS UNIQUE;

CREATE INDEX ON :Animal(nome)
```