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

Unidade Curricular de Bases de Dados

Ano Lectivo de 2017/2018

A oficina da Porta Aberta

A78218 Tiago Alves

A73909 Francisco Lira

A78296 Sérgio Alves

A70922 Francisco Costa

Janeiro, 2018

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

A oficina da Porta Aberta

A78218 Tiago Alves

A73909 Francisco Lira

A78296 Sérgio Alves

A70922 Francisco Costa

Janeiro, 2018

Resumo

Este relatório aborda a segunda parte do trabalho da Unidade Curricular de Base de Dados, na qual foi sugerida a realização de um trabalho de análise, planeamento, e implementação de um Sistema de Base de Dados não relacional, sendo o modelo de dados definido o Neo4j – Graphs Databases.

O tema é "A Oficina da Porta Aberta", sendo este o resultado da migração dos dados contidos no sistema relacional implementado na primeira parte deste trabalho. Todo o procedimento realizado para esta migração é explicado neste relatório.

Numa primeira fase, é feita uma contextualização do projeto e abordado o caso de estudo. Seguidamente, é apresentado o novo paradigma do SBD não relacional. Neste tópico é explicado o porquê da escolha de um sistema deste gênero, bem como uma descrição sobre o Neo4j.

Após esta fase inicial, passamos para a explicação pormenorizada do processo de migração, onde é apresentado o esquema da base de dados relacional, o esquema em grafos, e, para finalizar, a migração propriamente dita dos dados de MySQL para Neo4j.

Área de Aplicação: Desenho e arquitetura de Sistemas de Bases de Dados no âmbito de uma aplicação responsável pela gestão de uma oficina.

Palavras-Chave: Base de Dados, Modelo não Relacional, SQL, Neo4j, Sistema de gestão de Base de Dados

Índice

Anexos

Índice de Figuras

1. Introdução

1.1. Contextualização

O Sr Lourenço tem uma oficina de mecânica que está em crescimento. Ele abriu esta oficina em 2002, e desde aí tem gerido a oficina de uma maneira que ele acha ser ímpar no mercado e essencial para explicar o sucesso da mesma. Na opinião do Sr. Lourenço, é bom um cliente chegar a uma oficina com o seu veículo e ser direcionado para o mecânico com que lidou das outras vezes que lá foi, e saber que aquele é o melhor mecânico para aquele tipo de veículos. É algo que torna melhor a sua experiência enquanto cliente, uma vez que dá a possibilidade de se desenvolver um certo relacionamento entre mecânico e cliente, e este passa a ser uma mais valia para o negócio.

Até aqui, ele geria isso apenas de memória e às vezes guardava a informação num ficheiro excel, pois o volume de clientes não era de todo aquele que ele gostaria para a sua empresa. Mas recentemente, alguns dos seus novos clientes ao repararem que os clientes são tratados de uma maneira tão díspar nesta oficina deram a conhecer a outros amigos, e estes decidiram vir experimentar. Agora, ele usa muito frequentemente a folha excel, mas também já percebeu que este método não é o melhor.

É então necessário que haja uma melhor maneira para armazenar esta informação, e o Sr. Lourenço lembrou-se, que era bom conseguir também, guardar toda a informação relativa a quem tratou de cada carro e o que lhe fez, e conseguir guardar algumas notas acerca dos carros para quando estes voltarem à oficina. Poderá também medir o rendimento dos funcionários, pois tem guardada a informação do trabalho feito pelos mesmos, e pode deste modo, compensar aqueles que façam um trabalho melhor.

E com isto, o Sr. Lourenço pode passar mais tempo a tratar da sua oficina do que a decidir que mecânico faz cada serviço.

1.2. Fundamentação da implementação da base de dados

Para a realização deste projeto foi-nos proposta a implementação de uma Base de Dados com base num tema à nossa escolha. Assim sendo, ficamos com algumas dúvidas em relação à complexidade do tema que deveríamos escolher. Optamos então por escolher este tema, a gestão dos arranjos feitos por uma oficina automóvel, pois apresenta já alguma complexidade ao nível do sistema de Base de Dados, e é algo diferente dos temas abordados nas aulas, tornando o projeto um pouco mais interessante. Uma ida ao mecânico é uma experiência diferente daquelas que costumamos experimentar, pois, em alguns casos, estamos a pôr nas mãos de alguém que não conhecemos, um recurso valioso do nosso dia-a-dia, o nosso veículo.

Ao desenvolver este projeto pretendemos melhorar a experiência de um cliente quando este se dirige a uma oficina automóvel, e facilitar o trabalho do responsável da oficina no momento de distribuir trabalho.

De modo a responder a este problema, criamos uma Base de Dados simples, que preencha todos os requisitos, mas tivemos sempre em conta os problemas de organização atuais de uma oficina automóvel. Assim, tencionamos explicar e dar a entender como funciona uma oficina automóvel no momento em que um cliente traz o seu veículo para reparação ou manutenção.

1.3. Análise da viabilidade do processo

O objetivo deste projeto é a implementação de um Sistema de Base de Dados (SBD), ou, mais concretamente, um Sistema de Base de Dados Relacional (SBDR), que ajude o Sr. Lourenço na gestão e organização dos trabalhos na sua oficina. Contudo, não sabemos ainda se a implementação da mesma é viável, por isso vamos tentar perceber se realmente é necessário e se trará vantagens a implementação da mesma.

Começamos então por tentar perceber qual é o real problema do Sr. Lourenço. Ele tem neste momento demasiados clientes habituais, e gosta que, qualquer carro seja sempre que possível tratado pelo mesmo mecânico que tenha já lidado com o mesmo.

Neste momento o ficheiro excel não é possível de manter pois está uma confusão imensa, e ele tem de migrar toda a informação para outra plataforma. E este é o próximo passo que deve ele deve dar.

Em relação às vantagens que a implementação da base de dados trará à oficina, podemos destacar a rapidez com que ele é capaz de obter informação, relacionada com um funcionário, ou um veículo, ou até mesmo um determinado serviço. Estas consultas tornaram-se bastante mais rápidas, por exemplo, no caso do veículo, basta inserir a matrícula no sistema e a resposta àquilo que interrogamos é quase instantânea. Supondo que o Sr.Lourenço desejaria saber todos os arranjos que foram realizados a um certo veículo: apenas com a matrícula é possível obter essa informação, através das relações estabelecidas entre as diversas entidades. Existe, portanto, muito mais eficácia na resposta a

interrogações por parte do Sr.Lourenço caso o mesmo tenha uma base de dados implementada para a sua oficina.

Como é um SBDR, toda a informação está relacionada e sabemos que é integra, isto é, reduzimos quase para zero a possibilidade de haver informação contraditória, evitando conflitos.

O facto da gestão da informação ser rápida e eficaz, permite-nos evitar, assim, perdas de tempo. Também o facto da informação estar toda no mesmo sítio permite-nos garantir que sabemos sempre onde está a informação e de que informação dispomos.

Assim sendo, podemos concluir que é uma mais valia para a oficina a implementação de uma base de dados. Desta maneira conseguimos facilitar a gestão do trabalho executado e a obtenção de informação relativamente ao mesmo quando precisamos de verificar o que foi feito no passado.

2. Paradigma NoSQL

2.1. Adoção de NoSQL para a Oficina da Porta Aberta

Após a implementação da base de dados relacional a oficina do Sr. Lourenço começou a receber cada vez mais clientes. No entanto, devido ao bom atendimento e serviço único que o Sr. Lourenço conseguiu providenciar aos seus clientes, este foi expandindo o seu negócio, sendo agora dono de várias oficinas por todo o país e mesmo internacionalmente. Devido a esta expansão a base de dados relacional anteriormente pensada já não serve, pois esta não escala propriamente e o SGBD está a ficar lento.
Apesar de um SGBDR ser o mais comum pela forma que mantém a integridade dos dados através do modelo ACID (atomicidade, consistência, isolamento e durabilidade), estes sistemas pecam quando falámos de escalabilidade.

Este ponto é fulcral hoje-em-dia, já que os dados escalam massivamente, logo os SGBDR começam a não fazer sentido, visto que há pontos mais importantes em algumas situações.

Sendo assim, o após uma conversa longa com o Sr. Lourenço decidimos passar para um Sistema de Base de Dados Não Relacional, pois este tipo de sistemas permitem uma maior escalabilidade, e neste momento, isso é o mais importante para o Sr. Lourenço.

2.2. Neo4j

Uma base de dados não relacional baseado em grafos é feita propositadamente para manusear informação muito relacionada. Para além disso, o aumento do volume e conexão dos dados usados hoje em dia representa uma grande oportunidade para este tipo de Base de Dados.
Este tipo de bases de dados têm três principais vantagens:

Performance

Com bases de dados relacionais, as consultas irão chegar a um ponto em que deixam de ser praticáveis, pois à medida que aumentam os relacionamentos e a quantidade de dados a processar as consultas tornam-se mais lentas, ao contrário da performance de bases de dados não relacionais, que mantêm o tempo constante com aumento de informação ano após ano.

Flexibilidade

Com bases de dados não relacionais, a arquitetura de dados acompanha mais o desenvolvimento do negócio, porque a estrutura e o esquema de um modelo adapta-se às mudanças da aplicação e da indústria. Em vez de modelar exaustivamente o futuro e todas as suas possibilidade, podemos adicionar à estrutura existente sem comprometer a funcionalidade atual.

Agilidade

Desenvolver bases de dados não relacionais está diretamente alinhado com a metodologia ágil praticada hoje em dia, práticas de desenvolvimento guiado por testes, permitindo a base de dados não relacional evoluir lado a lado com a aplicação e qualquer mudança nos requisitos do mercado. Bases de dados não relacionais permitem o desenvolvimento sem problemas e sistemas de manutenção agradáveis.

Por todas estas vantagens, o Sr. Lourenço decidiu migrar para neo4j, devido há grande quantidade de dados e relacionamentos entre eles, já que as base de dados baseadas em grafos nos proporcionam uma maior rapidez na procura devido à sua estrutura, flexibilidade e agilidade.

3. Processo de migração

3.1. Modelo Lógico

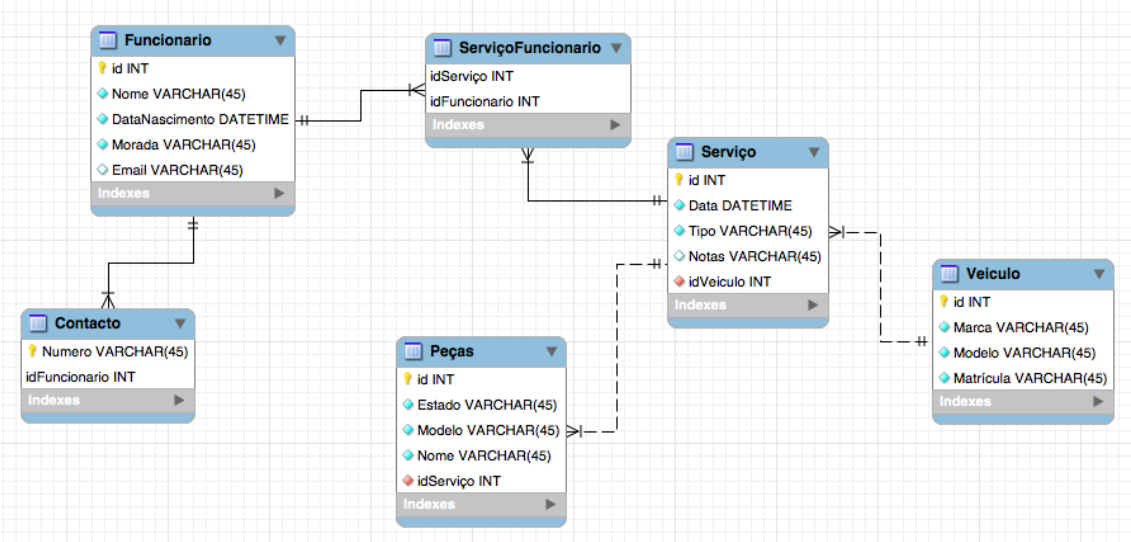

Figura 1 - Modelo Lógico

3.2. Esquema em Grafos

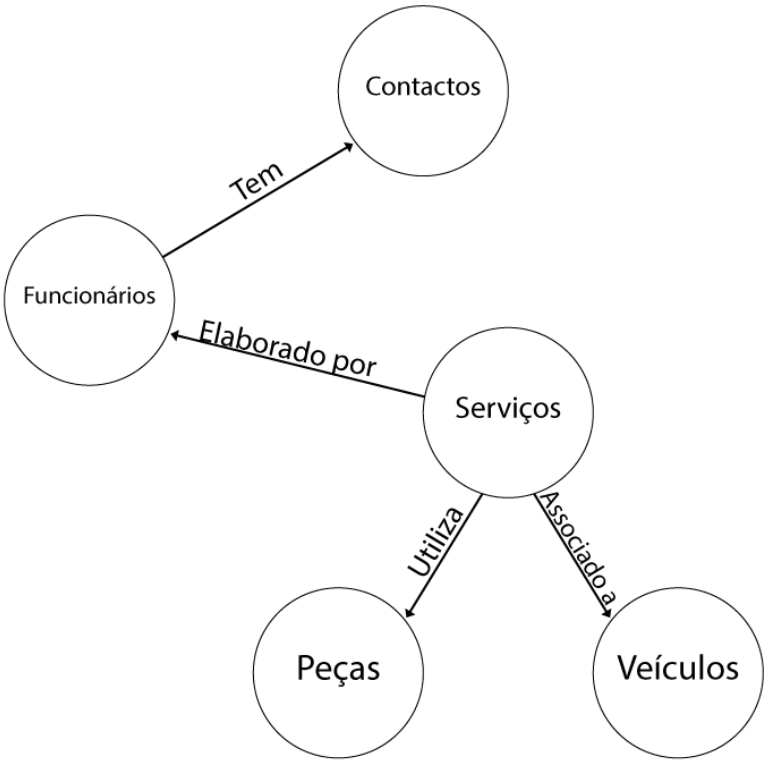

Figura 2 - Esquema em Grafos

3.3. Migração de MySQL para Neo4j

Para o processo de migração de uma base de dados relacional para uma não relacional, existem diversas opções de importação de dados. A opção tomada foi a implementação de uma aplicação que importa os dados de uma base de dados em MySQL, e guarda a informação em Neo4j. A aplicação foi codificada em Java, linguagem de programação em que os administradores da Oficina da Porta Aberta se sentiam mais entrosados.

Durante a fase de recolha de informação da base de dados, foi necessário ter em conta o esquema delineado anteriormente. Na base de dados em Neo4j, comparativamente com MySQL, existem caraterísticas fundamentais que importam realçar, tais como:

- Um tuplo é um nodo;

- Um nome de uma tabela é uma “etiqueta”(label name);
- Não há nulos(nulls). Os valores a nulo, no modelo relacional, simplesmente não existem num grafo;
- A descrição das relações é mais detalhada, visto que a relação não é feita através de chaves estrangeiras;
- Maior normalização;
- Tabelas de m:n são uma relação entre nodos;
- Tabelas que representam atributos multi-valor são um nodo;

Dadas estas caraterísticas, e aplicando-as ao modelo relacional, deu origem a 5 tipos de nodos – Funcionário, Contacto, Serviço, Veículo e Peça - relacionados entre si.

3.4. Esquema em Neo4j

Os dados precisam de ser armazenados de maneira a fazer corresponder todas as necessidades que eram anteriormente tidas no modelo relacional, uma vez que o propósito da oficina se mantém.

Tendo em conta a necessidade inicial requerida pelo Sr. Lourenço, os nodos “Funcionário” são fundamentais para o sistema, logo é nestes que guardamos a informação de todos os funcionários. Foi este mesmo motivo que deu origem aos nodos “Veículo”.

São também necessários nodos para representar os Serviços realizados na oficina, as peças utilizadas, e também os contatos dos funcionários. Começando pelos serviços, é bastante percetível a necessidade de nodos para representar e guardar toda a informação de cada um dos serviços efetuados pelos funcionários da oficina. Em relação às peças, também é preciso um nodo para estes, dado que um serviço pode utilizar várias peças, tornando-se necessário guardar toda a informação de cada peça em nodos.

Relativamente aos contatos, e dadas as caraterísticas referidas no ponto anterior(3.3 Migração de dados de MySQL para Neo4j), ou seja, como em MySQL é uma tabela que deriva do facto de “contatos” ser um atributo multi-valorado, em Neo4j transforma-se num nodo, que será, posteriormente, relacionado com o respetivo funcionário.

Quanto à tabela “ServiçoFuncionário”, visto que é uma tabela que deriva da relação de m:n entre Funcionário-Serviço, não se torna num nodo, mas sim numa relação entre os funcionários e o respetivo serviço.

4. Conclusão e apreciação crítica

Após a conclusão do projeto é nos agora possível fazer uma apreciação crítica. Começando pelas desvantagens, embora esta migração tenha sido proposta com o intuito de melhorar a conceção da Base de Dados, é possível apontar vários pontos negativos que com ela surgiram.

• Inexistência de restrições

Uma base de dados em neo4j, não apresenta um esquema rígido como uma base de dados em MySQL, onde é necessário definir previamente um esquema e estrutura, para a construção da base de dados. Para além disso, possui as restrições de integridade, de entidade, de domínio e ainda gerais, que tornam o esquema mais rígido. Pelo contrário, o Neo4j não apresenta qualquer restrição, possuindo desta forma um esquema extremamente flexível, o que pode provocar a inconsistência da base de dados. É por isso mesmo, necessário, um trabalho adicional por parte do programador na parte da aplicação para garantir que qualquer inserção ou modificação feita na base de dados irá manter esta consistente.

• Apresentação de Dados

Uma base de dados MySQL apresenta os dados através duma tabela, onde todos os dados são relacionados e organizados, logo são de fácil consulta. Numa base de dados baseada em grafos os dados são apresentados através de grafos, o que pode não ser uma forma organizada de apresentar os dados, principalmente quando há muitos relacionamentos entre eles.

No entanto, como é óbvio, esta migração também trouxe vantagens, que de certa forma compensam os pontos negativos anteriormente referidos.

• Desempenho das queries

Visto que a queries pedidas pelo Sr. Lourenço apresentam vários relacionamentos, conforme os dados fossem aumentando, pior o desempenho do SGBD no processamento das queries. A migração para neo4j veio melhorar esta parte, já que os grafos permitem um melhor processo de relacionamentos pela sua estrutura natural.

• Alterações no esquema da BD

No modelo relacional a base de dados apresenta uma estrutura muito rígida, e por isso, pouco flexível, o que originava muito tempo perdido pela equipa de gestão da BD, por qualquer razão, uma mudança na estrutura da BD fosse necessária. O mesmo não se verifica com a utilização do neo4j. Nesta situação na base de dados sem esquema, logo sem as características rígidas que uma BD do modelo anterior apresenta, o processo de modificação do esquema implica muito menos tempo gasto pela equipa.

Em suma, cada sistema possui as suas vantagens e desvantagens, logo cada sistema de gestão de base de dados deve ser utilizado em certas situações. Um SGBDR é melhor para base de dados de pequena/média dimensão, pois a integridade e consistência de dados é garantida, no entanto, quando falamos de Base de Dados de grandes dimensões, os sistemas NoSQL começam a fazer mais sentido, já que a performance destes sistemas é muito melhor para grandes quantidades de dados.

Referências

1. Tutorial: Import Data Into Neo4j.[online] Disponivel em: <https://neo4j.com/developer/guide-importing-data-and-etl/>

2. Write CSV Files in Java. [vídeo] Disponível em: <https://www.youtube.com/watch?v=myC5owRpMpQ>

3. Export data from mysql db to csv file using java. [online] Disponível em: < https://www.daniweb.com/programming/software-development/threads/493294/export-data-from-mysql-db-to-csv-file-using-java>

1. Anexos

1.1. Migração da Base de Dados

1.1.1 Dados para csv

```
import java.io.FileWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class export {
  public static void main(String[] args) {

        try {
        Class.forName("com.mysql.jdbc.Driver");
        } catch (Exception e) {
        e.printStackTrace();
        }

        String filename = "funcionarios.csv";
        try {
        String URL = "localhost";
        String SCHEMA = "oficina";
        String USERNAME = "root";
        String PASSWORD = "140697";
        FileWriter fw = new FileWriter(filename);
        Connection conn =
DriverManager.getConnection("jdbc:mysql://"+URL+"/"+SCHEMA+"?auto
Reconnect=true&useSSL=false",USERNAME,PASSWORD);
        String query = "select * from Funcionario";
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(query);
        fw.append("id,Nome,DataNascimento,Morada,Email");
        fw.append('\n');
```

```
while (rs.next()) {
fw.append(String.valueOf(rs.getInt(1)));
fw.append(',');
fw.append(rs.getString(2));
fw.append(',');
fw.append(rs.getString(3));
fw.append(',');
fw.append(rs.getString(4));
fw.append(',');
fw.append(rs.getString(5));
fw.append('\n');
}
fw.flush();
fw.close();
System.out.println("CSV File is created successfully.");


filename = "serviços.csv";
fw = new FileWriter(filename);
query = "select * from Serviço";
stmt = conn.createStatement();
rs = stmt.executeQuery(query);
fw.append("id,Data,Tipo,Notas,idVeiculo");
fw.append('\n');
while(rs.next()){
fw.append(String.valueOf(rs.getInt(1)));
fw.append(',');
fw.append(rs.getString(2));
fw.append(',');
fw.append(rs.getString(3));
fw.append(',');
if(rs.getString(4)!=null){
String s = rs.getString(4).replaceAll(",", " e");
fw.append(s);
}
else{
fw.append(rs.getString(4));
}
fw.append(',');
fw.append(String.valueOf(rs.getInt(5)));
fw.append('\n');
}
fw.flush();
fw.close();
System.out.println("CSV File is created successfully.");

filename="peças.csv";
```

```
fw = new FileWriter(filename);
query = "select * from Peça";
stmt = conn.createStatement();
rs = stmt.executeQuery(query);
fw.append("id,Estado,Modelo,Nome,idServiço");
fw.append('\n');
while(rs.next()){
fw.append(String.valueOf(rs.getInt(1)));
fw.append(',');
fw.append(rs.getString(2));
fw.append(',');
fw.append(rs.getString(3));
fw.append(',');
fw.append(rs.getString(4));
fw.append(',');
fw.append(String.valueOf(rs.getInt(5)));
fw.append('\n');
}
fw.flush();
fw.close();
System.out.println("CSV File is created successfully");

filename = "veiculos.csv";
fw = new FileWriter(filename);
query = "select * from Veiculo";
stmt = conn.createStatement();
rs = stmt.executeQuery(query);
fw.append("id,Marca,Modelo,Matricula");
fw.append('\n');
while(rs.next()){
fw.append(String.valueOf(rs.getInt(1)));
fw.append(',');
fw.append(rs.getString(2));
fw.append(',');
fw.append(rs.getString(3));
fw.append(',');
fw.append(rs.getString(4));
fw.append('\n');
}
fw.flush();
fw.close();
System.out.println("CSV File is created successfully");

filename = "contactos.csv";
fw = new FileWriter(filename);
query = "select * from Contacto";
```

```
        stmt = conn.createStatement();
        rs = stmt.executeQuery(query);
        fw.append("Numero,idFuncionario");
        fw.append('\n');
        while(rs.next()){
        fw.append(String.valueOf(rs.getInt(1)));
        fw.append(',');
        fw.append(String.valueOf(rs.getInt(2)));
        fw.append('\n');
        }
        fw.flush();
        fw.close();
        System.out.println("CSV File is created successfully");

        filename = "sfunc.csv";
        fw = new FileWriter(filename);
        query = "select * from ServiçoFuncionario";
        stmt = conn.createStatement();
        rs = stmt.executeQuery(query);
        fw.append("idServiço,idFuncionario");
        fw.append('\n');
        while(rs.next()){
        fw.append(String.valueOf(rs.getInt(1)));
        fw.append(',');
        fw.append(String.valueOf(rs.getInt(2)));
        fw.append('\n');
        }
        fw.flush();
        fw.close();
        conn.close();
        System.out.println("CSV File is created successfully");
        } catch (Exception e) {
        e.printStackTrace();
        }
  }
}
```

1.1.2 Criação dos nodos

```
//Create Contacto
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///contactos.csv" AS row
CREATE (:Contacto {Numero: row.Numero});

// Create funcionarios
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///funcionarios.csv" AS row
CREATE (:Funcionario {id: row.id, Nome: row.Nome, DataNascimento:
row.DataNascimento, Morada: row.Morada, Email: row.Email});

// Create peças
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///peças.csv" AS row
CREATE (:Peça {id: row.id, Estado: row.Estado, Modelo: row.Modelo,
Nome: row.Nome});

// Create serviços
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///serviços.csv" AS row
CREATE (:Serviço {id: row.id, Data: row.Data, Tipo: row.Tipo, Notas:
row.Notas, idVeiculo: row.idVeiculo});

// Create veiculos
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///veiculos.csv" AS row
CREATE (:Veiculo {id:row.id,  Marca: row.Marca, Modelo: row.Modelo,
Matricula: row.Matricula});
```

1.1.3 Criação de relacionamentos

```
//relação funcionario-contacto
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///contactos.csv" AS row
MATCH (c:Contacto {Numero: row.Numero})
MATCH (f:Funcionario {id: row.idFuncionario})
MERGE (f)-[:TEM]->(c);

//relação serviço-Funcionario
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///sfunc.csv" AS row
MATCH (s:Serviço {id: row.idServiço})
MATCH (f:Funcionario {id: row.idFuncionario})
```

```
MERGE (s)-[fu:ELABORADO_POR]->(f);

//relação serviço-peça
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///peças.csv" AS row
MATCH (p:Peça {id: row.id})
MATCH (s:Serviço {id: row.idServiço})
MERGE (s)-[:UTILIZA]->(p);

//relação serviço-veiculo
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///serviços.csv" AS row
MATCH (s:Serviço {id: row.id})
MATCH (v:Veiculo {id: row.idVeiculo})
MERGE (v)-[:ASSOCIADO_A]->(s);
```

1.2. Queries em chyper (neo4j)

```
//Funcionario de Serviço
MATCH(f:Funcionario)<-[:ELABORADO_POR]-(:Serviço {Tipo: "Mudança
de oleo"})
RETURN DISTINCT f.Nome;

//Funcionario de Veiculo
MATCH(f:Funcionario)<-[:ELABORADO_POR]-(:Serviço)<-
[:ASSOCIADO_A]-(:Veiculo {Matricula: "SHH8039K"})
RETURN DISTINCT f.Nome;

//Mais Trabalhadores
MATCH(f:Funcionario)<-[:ELABORADO_POR]-(:Serviço)
RETURN f.Nome, count(*) AS cnt
ORDER BY cnt DESC;

//Numero de peças novas
MATCH(p:Peça {Estado:"novo"})
RETURN p.Nome, count(*) AS cnt
ORDER BY cnt DESC;

//Numero de peças usadas
MATCH(p:Peça {Estado:"usado"})
RETURN p.Nome, count(*) AS cnt
ORDER BY cnt DESC;

//Total Serviços de Funcionarios
MATCH(f:Funcionario)<-[:ELABORADO_POR]-(:Serviço)
```

```
RETURN f.Nome, count(*)
ORDER BY f.Nome;

//Peças usadas em Veiculos
MATCH(p:Peça)<-[:UTILIZA]-(s:Serviço)<-[:ASSOCIADO_A]-(v:Veiculo)
RETURN v.Marca, v.Modelo, p.Nome
ORDER BY v.Marca;

//procura Funcionario
MATCH(c:Contacto)<-[:TEM]-(f:Funcionario)<-[:ELABORADO_POR]-
(s:Serviço)<-[:ASSOCIADO_A]-(v:Veiculo)
WHERE v.Matricula = "CYEA2I65" OR s.Tipo = "Limpeza dos filtros do
ar"
RETURN DISTINCT f.Nome, c.Numero;

//Serviços de Funcionarios
MATCH(f:Funcionario)<-[:ELABORADO_POR]-(s:Serviço)
RETURN f.Nome, s.Tipo;

//Veiculos com mais serviços
MATCH(:Serviço)<-[:ASSOCIADO_A]-(v:Veiculo)
RETURN v.Marca, count(*) as ct
order by ct desc;
```