

Universidade de Trás-os-Montes e Alto Douro

Escola de Ciências e Tecnologias

Departamento de Engenharias

Licenciatura em Engenharia Informática

Base de Dados



Relatório Final do Trabalho Prático

*Docentes: Paulo Nogueira Martins
Daniel Alexandre Moreira Lopes*

*Discentes: Eduardo Gil Macedo Vaz Pinheiro nº68887
José Manuel Gouveia Fernandes nº68849
Miguel Oliveira e Santos nº68916*

Índice

Resumo do Projeto.....	3
1ª Fase	4
Mapeamento do diagrama E-R para o modelo relacional e Normalização do modelo relacional até à 3.ª Forma Normal	4
Criação da base de dados com as tabelas recorrendo ao Microsoft SQL Server Management Studio	6
Diagrama das tabelas criadas na base de dados	7
2ª Fase	8
Perguntas:	8
2.1 Qual o último transporte realizado? [Vaivém (Nome), Satélites (Código), Data]	8
2.2 Quantos transportes já se realizaram por cada Vaivém? [Vaivém (Código e Nome), N_Transportes]	9
2.3 Qual a Empresa que mais pagou por um transporte? [Empresa (Nome), Preço, Data]	9
2.4 Quais as empresas que pagaram por mais de 2 transportes nos últimos 30 dias? Ordene-as alfabeticamente. [Empresas (Nome), N_Transportes]	10
2.5 Quais os 2 armazéns com mais satélites guardados? [Armazém (nome), Quantidade]	10
2.6 Quantos protocolos foram definidos por cada empresa no ano de 2020? [Empresa (nome, telefone)]	11
2.7 Qual o peso total transportado por cada vaivém no ano de 2020? Ordene-os por ordem crescente. [Vaivém (Nome), PesoTotal]	12
Conclusão	16

Resumo do Projeto

No âmbito da unidade curricular Bases de Dados, foi-nos lançado um projeto para ser realizado no contexto de avaliar os conhecimentos adquiridos durante as aulas desta mesma UC.

Este projeto está dividido em 3 fases. Na 1ª Fase, os professores forneceram-nos um diagrama ER, com o objetivo de o passar para o modelo relacional e, consequentemente, normalizá-lo até à terceira forma normal. Com estas duas primeiras etapas da 1ª Fase concluídas, passamos para as duas etapas seguintes, com o auxílio do Microsoft SQL Server Management Studio (SSMS), procedemos então como pedido à criação de uma base de dados baseada no modelo relacional construído previamente. Com esta etapa concluída passamos para a quarta e última etapa que foi a de criar o diagrama da base de dados, procedimento que é feito automaticamente pelo SSMS.

Nesta 2ª Fase do projeto, colocaram-nos o desafio de inserir alguns registos em cada uma das tabelas da base de dados criada na fase anterior com o comando INSERT e, consequentemente, foram-nos lançados alguns desafios com os comandos SELECT, UPDATE e DELETE, onde tivemos de obter dados a partir da base de dados com certas restrições impostas.

A 3ª e última Fase do projeto está relacionada com os stored procedures e triggers, ferramentas disponibilizadas pela linguagem SQL, em que os stored procedures são o equivalente em SQL a subrotinas existentes em outras linguagens de programação enquanto os triggers são um tipo especial de Stored Procedure que é executado automaticamente como parte de uma modificação de dados.

1ª Fase

Nesta primeira fase do trabalho experimental foi-nos proposto um protocolo com um diagrama E-R de uma base de dados. Com este diagrama, iremos proceder ao seu mapeamento para o modelo relacional bem como a sua normalização até à 3ª Forma Normal (3FN). Com o modelo relacional feito, fizemos, recorrendo ao Microsoft SQL Server Management Studio, a implementação do modelo físico da base de dados com as suas respetivas restrições de integridade, dando uso à linguagem utilizada pelo programa, o SQL. Com o modelo físico implementado, procedemos à criação do diagrama desta mesma base de dados.

Mapeamento do diagrama E-R para o modelo relacional e Normalização do modelo relacional até à 3.ª Forma Normal

Armazém (Codigo_Armazem, Nome, Localizacao)

Vaivém (Codigo_Vaivem, Nome, Modulos, Capacidade)

Satélites (Codigo_Satelites, Tipo, Peso, Ano, Modelo)

Empresa (Codigo_Empresa, Telefone, Nome, NIF, Endereco_Localidade, Endereco_Morada, Endereco_CodigoPostal)

Protocolo (Ref_Protocolo, Descricao, Tipo)

Funcionários (ID_Funcionarios, Salario, Nome)

Guardar (Codigo_Armazem, Codigo_Satelites, Codigo_Empresa, Data_Inicio, Data_Fim)

- **Código** referencia Armazém, Satélites, Empresa

Manutenção (ID_Funcionarios, Codigo_Empresa, Codigo_Vaivem, Data, Descricao)

- **ID** referencia Funcionários

- **Código** referencia Empresa, Vaivém

Definir (Codigo_Empresa, Ref_Protocolo, Data, Num_Paginas)

- **Código** referencia Empresa

- **Referência** referencia Protocolo

Transportar (Codigo_Vaivem, Codigo_Satelites, Ref_Protocolo, ID_Transportar, Preco, Data)

-**Código** referencia Vaivém, Satélites

-**Referência** referencia Protocolo

Pagar (Codigo_Empresa, ID_Transportar, Codigo_Vaivem, Codigo_Satelites,
Ref_Protocolo, Data, Preco)

-**Código** referencia Empresa

-**ID_Transportar**, **Codigo_Vaivem**, **Codigo_Satelites**,
Referencia_Protocolo referencia transportar

Criação da base de dados com as tabelas recorrendo ao Microsoft SQL Server Management Studio

```
USE master

CREATE DATABASE empresa_foguete;
USE empresa_foguete;
GO

CREATE TABLE Armazem(
    cod_armazem INTEGER PRIMARY KEY,
    nome VARCHAR(50) NOT NULL,
    localizacao VARCHAR(50) NOT NULL
);

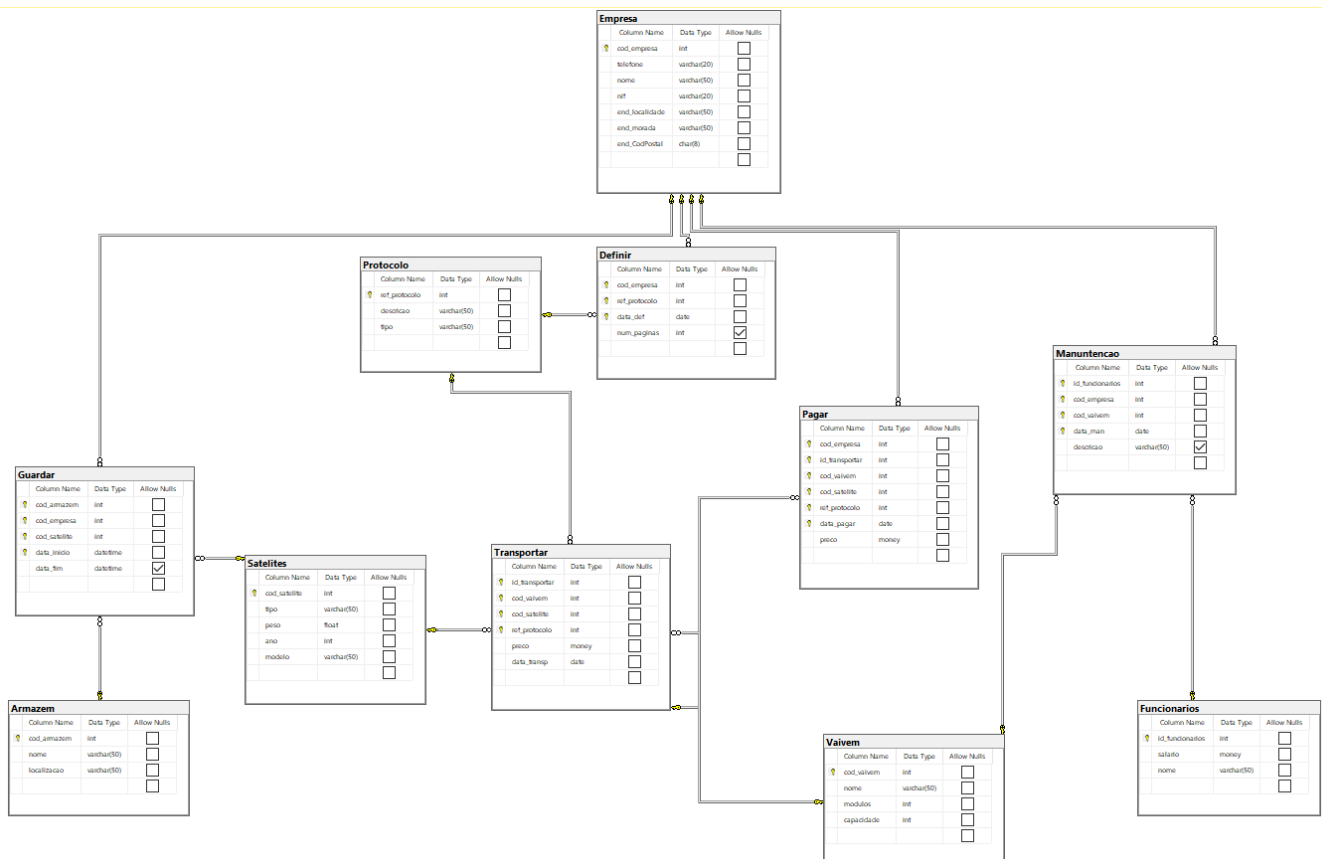
CREATE TABLE Empresa(
    cod_empresa INTEGER PRIMARY KEY,
    telefone VARCHAR(20) NOT NULL,
    nome VARCHAR(50) NOT NULL,
    nif VARCHAR(20) NOT NULL,
    end_localidade VARCHAR(50) NOT NULL,
    end_morada VARCHAR(50) NOT NULL,
    end_CodPostal CHAR(8) NOT NULL,

    CHECK (end_CodPostal LIKE '[0-9][0-9][0-9][0-9]-[0-9][0-9][0-9]')
);
```

Aqui criámos a base de dados usando o comando CREATE DATABASE com o nome que queríamos para esta, certificámo-nos que utilizamos a base de dados criada com o comando USE e utilizamos o CREATE TABLE com as respetivas colunas que queremos inserir na tabela e especificamos o tipo de cada uma juntamente com os atributos que são chaves primárias. Por fim utilizamos o comando CHECK para impor as restrições de cada coluna.

Diagrama das tabelas criadas na base de dados

Nesta parte final foi-nos pedida a criação do diagrama da base de dados, que se apresenta representado na figura seguinte, onde podemos verificar todas as relações entre cada tabela, bem com as suas colunas, chaves primárias, chaves estrangeiras, tipo de atributo e se é permitida a inserção de registos nulos em cada uma das colunas de cada tabela:



2ª Fase

Nesta fase do projeto, submetemos várias entradas em cada uma das tabelas da base de dados criada na fase anterior deste mesmo projeto utilizando o comando **INSERT INTO** [tabela] **VALUES**. Após o passo anterior estar concluído, foram-nos dados vários desafios envolvendo a utilização dos comandos de manipulação de dados como o **SELECT**, o **FROM** etc.

Perguntas:

1. Insira, pelo menos, 3 registos em cada tabela.

```
INSERT INTO Armazem
VALUES (1,'Armazem Boavista', 'Boavista'),(2,'Armazem Braga', 'Braga'),(3,'Armazem Bragança', 'Bragança');

INSERT INTO Empresa
VALUES (1,'917769690','Empresa Coelho Lta','123456789','Lisboa','Circular Norte','1800-134'),
(2,'921234567','Empresa Ribeiro DJ8 Lta','234567890','Porto','Avenida do Dj8','4000-011'),
(3,'934567890','Empresa Corby Lta','345678910','Amadora','Rua do Corby','1500-173');
```

Nesta primeira pergunta preenchemos as tabelas criadas na base de dados criada na fase anterior do projeto para seguidamente a podermos manipular com os comandos de manipulação de dados, como, por exemplo, o **SELECT**. Nesta figura, demonstramos dois desses mesmos **INSERT**s que utilizamos para cada uma das tabelas.

Em todas as alíneas, utilizamos primeiro o comando **USE** para indicarmos, de que base de dados estamos a selecionar a informação.

2.1 Qual o último transporte realizado? [Vaivém (Nome), Satélites (Código), Data]

```
--2.1
USE empresa_foguete;
SELECT Vaivem.nome AS 'Vaivém (Nome)', Satelites.cod_satelite AS 'Satélites (Código)', Transportar.data_transp AS Data
FROM Vaivem, Satelites, Transportar
WHERE Transportar.data_transp=(SELECT MAX(data_transp) FROM Transportar)
AND Satelites.cod_satelite=Transportar.cod_satelite
AND Vaivem.cod_vaivem=Transportar.cod_vaivem
```

Recorremos ao comando **SELECT** para indicarmos os dados que queremos que sejam mostrados na tabela e o comando **AS** para indicarmos o nome da coluna que estamos a selecionar. Para indicarmos ao programa de que tabelas

estamos a selecionar os dados, utilizamos o comando **FROM** e, finalmente, o comando **WHERE** para indicarmos as restrições dos dados que estamos a tentar encontrar. Por fim, como só queríamos último transporte realizado, ou seja, a data mais recente, utilizamos o **MAX** para o programa saber que apenas queremos que seja mostrada a maior data da tabela.

2.2 Quantos transportes já se realizaram por cada Vaivém? [Vaivém (Código e Nome), N_Transportes]

```
--2.2
USE empresa_foguete;
SELECT Vaivem.cod_vaivem AS 'Vaivém (Código)', Vaivem.nome AS 'Vaivém (Nome)', COUNT(Transportar.id_transportar) AS N_Transportes
FROM Vaivem, Transportar
WHERE Vaivem.cod_vaivem=Transportar.cod_vaivem
GROUP BY Vaivem.nome, Vaivem.cod_vaivem
```

O objetivo nesta pergunta era sabermos quantos transportes já se realizaram por cada Vaivém. Para este fim, utilizamos o comando **SELECT** e o comando **FROM** para sabermos que tipo de dados queremos da base de dados e de que respetivas tabelas, e o comando **WHERE** para sabermos as restrições dos dados que pretendemos encontrar. Utilizamos o comando **GROUP BY** para ordenarmos os resultados da forma que queremos. O comando **COUNT** é utilizado pois o que é pretendido no exercício é saber a contagem de transportes já se realizaram por cada Vaivém.

2.3 Qual a Empresa que mais pagou por um transporte? [Empresa (Nome), Preço, Data]

```
--2.3
USE empresa_foguete;
SELECT Empresa.nome AS 'Empresa (Nome)', Pagar.preco AS Preço, Pagar.data_pagar AS Data
FROM Empresa, Pagar, Transportar
WHERE Pagar.preco=(SELECT MAX(preco) FROM Pagar)
AND Empresa.cod_empresa=Pagar.cod_empresa
AND Pagar.preco=Transportar.preco
```

Recorremos ao comando **SELECT** para indicarmos os dados que queríamos que fossem mostrados na tabela e o comando **AS** para indicarmos o nome da coluna que estamos a selecionar. Para indicarmos ao programa de que tabelas estamos a selecionar os dados, utilizamos o comando **FROM** e, finalmente, o

comando **WHERE** para indicarmos as restrições dos dados que estamos a tentar encontrar. Por fim, como só queríamos a Empresa que mais pagou por um transporte utilizámos o **MAX** para o programa saber que apenas queremos que seja mostrada na tabela onde o preço do pagamento é o mais alto.

2.4 Quais as empresas que pagaram por mais de 2 transportes nos últimos 30 dias? Ordene-as alfabeticamente. [Empresas (Nome), N_Transportes]

```
--2.4 NESTA NAO APARECERA NENHUMA EMPRESA POIS NENHUMA PAGOU POR MAIS DE 2 TRANSPORTES NOS ULTIMOS 30 DIAS
USE empresa_foguete;
SELECT Empresa.nome AS 'Empresas (Nome)', COUNT(Transportar.cod_satelite) AS N_Transportes
FROM Empresa, Transportar, Pagar
WHERE Pagar.cod_empresa=Empresa.cod_empresa
AND Pagar.id_transportar=Transportar.id_transportar
AND Transportar.data_transp> GETDATE()-30
GROUP BY Empresa.nome
HAVING COUNT(DISTINCT Transportar.cod_satelite)>2
ORDER BY Empresa.nome ASC
```

Nesta questão decidimos agrupar todas as empresas que tinham pagado mais de 2 transportes nos últimos 30 dias, que, por sua vez, foram distinguidos com o **HAVING COUNT > 2**, para indicar as empresas com mais de 2 transportes, por ordem ascendente com os comandos **ORDER BY ASC** para obtermos a lista das empresas que pagaram mais de 2 transportes por ordem alfabética.

NOTA: No nosso caso, a tabela não irá apresentar nenhum dado nesta pois nenhuma empresa pagou por mais de 2 transportes nos últimos 30 dias, e para saber este intervalo de tempo usámos **> GETDATE()-30**.

2.5 Quais os 2 armazéns com mais satélites guardados? [Armazém (nome), Quantidade]

```
--2.5
USE empresa_foguete;
SELECT TOP 2 Armazem.nome AS 'Armazém (Nome)', COUNT(Guardar.data_fim) AS Quantidade
FROM Guardar, Satelites, Armazem
WHERE Armazem.cod_armazem=Guardar.cod_armazem
AND Satelites.cod_satelite=Guardar.cod_satelite
GROUP BY Armazem.nome
ORDER BY Quantidade DESC
```

Fizemos uso do **SELECT** para indicarmos os dados que queríamos que fossem mostrados na tabela e o comando **AS** para indicarmos o nome da coluna que estamos a seleccionar. Para indicarmos ao programa de que tabelas estamos a seleccionar os dados, utilizamos o comando **FROM** e, finalmente, o comando **WHERE** para indicarmos as restrições dos dados que estamos a tentar encontrar. Nesta diferente das anteriores recorremos ao **TOP 2** para saber os 2 armazéns com mais satélites guardados e usámos um **ORDER BY** para a quantidade ficar por ordem decrescente melhorando a função **TOP**.

2.6 Quantos protocolos foram definidos por cada empresa no ano de 2020?
[Empresa (nome, telefone)]

```
--2.6
USE empresa_foguete;
SELECT Empresa.nome AS 'Empresa (nome)', Empresa.telefone AS 'Empresa (telefone)', COUNT(Protocolo.ref_protocolo) AS N_Protocolos
FROM Empresa, Definir, Protocolo
WHERE Definir.data_def <= '2020/12/31'
AND Definir.data_def >= '2020/01/01'
AND Definir.cod_empresa=Empresa.cod_empresa
AND Protocolo.ref_protocolo=Definir.ref_protocolo
GROUP BY Empresa.nome, Empresa.telefone
```

Como nos anteriores usamos do **SELECT** para indicarmos os dados que queríamos que fossem mostrados na tabela e o comando **AS** para indicarmos o nome da coluna que estamos a seleccionar. Para indicarmos ao programa de que tabelas estamos a seleccionar os dados, utilizamos o comando **FROM**. que estamos a tentar encontrar. Usamos um **COUNT** para fazer a conta de quantos protocolos foram definidos para cada empresa. Com o comando **WHERE** e **AND**, impusemos as condições de que as empresas tinham de ter os protocolos definidos no ano de 2020, ou seja, a data fosse maior ou igual a 1 de janeiro de 2020 e menor ou igual a 31 de dezembro de 2020, ou seja, durante o pedido ano de 2020. No final agrupamos os dados com o comando **GROUP BY** com o nome da empresa e o seu número de telefone.

2.7 Qual o peso total transportado por cada vaivém no ano de 2020? Ordene-os por ordem crescente. [Vaivém (Nome), PesoTotal]

```
--2.7
USE empresa_foguete;
SELECT Vaivem.nome AS NomeVaivem, SUM(Vaivem.capacidade + Satelites.peso) AS PesoTotal
FROM Vaivem, Transportar, Satelites
WHERE Vaivem.cod_vaivem=Transportar.cod_vaivem
AND Satelites.cod_satelite=Transportar.cod_satelite
AND Transportar.data_transp <= '2020/12/31'
AND Transportar.data_transp >= '2020/01/01'
GROUP BY Vaivem.nome
ORDER BY PesoTotal ASC
```

Como nos anteriores usamos do **SELECT** para indicarmos os dados que queríamos que fossem mostrados na tabela e o comando **AS** para indicarmos o nome da coluna que estamos a selecionar. Para indicarmos ao programa de que tabelas estamos a selecionar os dados, utilizamos o comando **FROM**. que estamos a tentar encontrar. Usamos um **SUM** para somar todos os pesos para sabermos o peso total sendo este o peso do vaivém mais o peso dos satélites por este transportados. Com o comando **WHERE** e **AND**, impusemos as condições de que os transportes tinham de ter sido feitos no ano de 2020, ou seja, a data fosse maior ou igual a 1 de janeiro de 2020 e menor ou igual a 31 de dezembro de 2020, ou seja, durante o pedido ano de 2020. No final agrupamos os dados com o comando **GROUP BY** com o nome do vaivém e para que o peso total tivesse por ordem crescente usamos o comando **ORDER BY**.

3ª Fase


Nesta 3ª e última Fase, foram nos propostos 3 exercícios que envolvem stored procedures e triggers, que consistem na criação de “funções” para manipulação de dados.

Pergunta 1

Para esta pergunta foi nos lançado o desafio de criar um stored procedure que, dado o NIF de uma empresa verifica se algum dos satélites da empresa foi guardado num armazém. Em caso afirmativo o procedimento deve retornar o tempo médio de armazenamento e retornar ‘-1’ caso contrário.

```
DROP PROCEDURE guardarSatelite
CREATE PROCEDURE guardarSatelite (@nif VARCHAR(20))
AS
BEGIN
    DECLARE @idempresa INTEGER
    SELECT @idempresa = cod_empresa FROM Empresa WHERE @nif = nif
    IF @idempresa > 0
    begin
        SELECT AVG ( datediff(day,G.data_inicio,G.data_fim)) AS 'tempo médio'
        FROM Empresa E, Guardar G
        WHERE E.nif LIKE @nif AND E.cod_empresa = G.cod_empresa
    end
    else
        SELECT '-1' AS ERROR
    END

EXEC guardarSatelite 345678910
SELECT * FROM Guardar
SELECT * FROM Empresa
```



	tempo médio
1	231

Para criarmos uma variável onde o valor de retorno seja armazenado usamos DECLARE e depois verificamos se existia uma empresa através do seu ID. Depois para calcular o tempo médio de armazenamento fizemos o SELECT AVG e alinhamos as tabelas Empresa e Guardar. Caso isto não se verificasse iria retornar ‘-1’.

Neste procedure retornou-nos o tempo médio que os satélites da empresa com o NIF “345678910” teve guardado em que pra isso usamos o EXECUTE, para executarmos o procedure.

Pergunta 2

Nesta segunda questão foi nos lançado o desafio, como na questão anterior, de criar um stored procedure mas que dado um mês apresente uma tabela com os descontos de cada empresa nesse mês.

```
DROP PROCEDURE Bonus
CREATE PROCEDURE Bonus @mes NUMERIC(2,0), @ano NUMERIC(4,0)
AS
BEGIN
    SELECT E.nome AS 'nome da empresa', sum(T.preco) * 0.05 AS 'bônus de transporte'
    FROM Empresa E, Pagar P, Transportar T
    WHERE E.cod_empresa = P.cod_empresa AND MONTH(T.data_transp) = @mes
        AND YEAR(T.data_transp) = @ano AND P.id_transportar = T.id_transportar
    GROUP BY E.cod_empresa, E.nome
END

EXEC Bonus 06, 2020
SELECT * FROM Empresa
SELECT * FROM Pagar
SELECT * FROM Transportar
```

100 %

Results Messages

	nome da empresa	bônus de transporte
1	Empresa Coelho Lta	100.000000

Para isso, criamos o procedure com o comando CREATE PROCEDURE e passamos como parâmetros o mês, com o nome @mes e o ano com o nome @ano. Este procedure vai dar uso aos comandos SELECT e FROM para ir buscar os dados necessários às tabelas indicadas, com o WHERE vamos indicar as restrições necessárias e com o MONTH () e YEAR () vamos retirar o mês e o ano indicado passados como parâmetros.

Para terminar vamos agrupar os dados com o comando GROUP BY e utilizar o EXECUTE para testar o procedure com o mês e o ano pretendidos.

Pergunta 3

Nesta última pergunta, foi-nos pedido que criássemos um trigger que, se após se inserir uma manutenção esta não tiver descrição, insira automaticamente na descrição: 'descrição omissa'.

```
CREATE TRIGGER Descricao
ON Manutencao
AFTER INSERT
AS
BEGIN
    IF EXISTS (SELECT NULL FROM INSERTED WHERE descricao IS NULL)
    BEGIN
        UPDATE Manutencao
        SET descricao = 'descrição omissa'
        WHERE descricao IS NULL;
    END;
END;

DROP TRIGGER Descricao
INSERT INTO Manutencao VALUES (2,1,1,'2020/07/14',NULL)
SELECT * FROM Manutencao
```

100 %

Results Messages

	id_funcionarios	cod_empresa	cod_vaivem	data_man	descricao
1	1	1	1	2020-07-14	descrição omissa
2	2	1	1	2020-07-14	descrição omissa
3	2	3	3	2018-11-20	A manutenção deve ser feita dentro de 20 dias
4	3	2	2	2019-09-08	A manutenção deve ser feita dentro de 30 dias

Para a resolução desta última alínea criamos o trigger com o comando CREATE TRIGGER na tabela onde queremos inserir o comando e fizemos a condição 'IF' para verificar se existe descrição e se esta conter a palavra 'NULL' a tabela irá ser atualizada para descrição omissa. Usamos o comando INSERT para inserir valores na tabela e verificar se o trigger realmente funciona.

Conclusão

Com a resolução deste trabalho e do protocolo apresentado pelos professores conseguimos não só adquirir conhecimento sobre o funcionamento de um sistema de base de dados, mas também sobre todos os processos que envolvem a gestão do mesmo. Ao mesmo tempo permitiu-nos refletir sobre os problemas de armazenamento e gestão de dados que nos foram aparecendo. Ao longo do trabalho fomos explicando como é feita esta gestão através de exemplos e dos respetivos resultados obtidos. Os objetivos do nosso projeto consistiram em passar de um modelo de Entidade-Relacionamento para um Modelo Relacional, resolver vários problemas de pesquisa utilizando queries SQL e colocarem prática os conceitos de procedimentos e triggers, tendo estes objetivos sido cumpridos.