
Licenciatura em Engenharia Informática
Bases de Dados
2º Semestre

# Plataforma de Gestão para um Produtor Agrícola

Trabalho elaborado por:
Diego Trovisco 80228
Francisco Silveira 84802

Aveiro, 8 de Junho de 2018

# Índice

# Introdução

Hoje em dia, muitos produtores agrícolas, tanto com negócios pequenos como negócios maiores, ainda fazem a gestão dos seus animais em papel. Esta plataforma tem como objetivo passar esta gestão do papel para o computador e desta forma agilizar e facilitar estes processos.

Esta plataforma tem como objetivo ser usada localmente pelo utilizador. O utilizador pode adicionar os seus animais, bem como as suas informações. Também é possível registar compras e vendas de animais e consultar o histórico de compras e vendas.

Em relação aos seus animais é possível controlar a sua vacinação podendo registar as datas das vacinações e qual as vacinas administradas. Também as análises feitas aos seus animais podem ser registadas na plataforma.

Além disto, o utilizador poderá registar a produção de leite de manhã e de tarde para a sua fábrica ficando registado na base de dados estes dados e mostrando algumas informações adicionais.

Esta plataforma é desenvolvida em SQL (Structured Query Language) e a interface gráfica para o utilizador desenvolvida na linguagem C#.

O desenvolvimento desta plataforma tem como objetivo no futuro ser implementada e usada em ambiente real pelo que algumas funções podem parecer mais adequadas no papel mas não tão práticas na realidade.

# Análise de Requisitos

O sistema foi idealizado para ser usado por um produtor. Desta forma ele pode registar animais, registar informações deles como também a sua vacinação e análises. Também é possível registar vendas e compras e registar a produção de leite.

## Requisitos Iniciais

- Uma Fábrica de Leite é identificada pelo seu nome, e tem com atributos o local onde se encontra e o preço que leva por litro
- O produtor gere a aplicação e é identificado por um número e pelo nif;
- Cada pessoa é identificada pelo seu nif, pelo nome,pelo sexo, localidade, pela data de nascimento, pelo telefone e pelo seu email;
- Na produção de leite é registado a data da colheita, o produtor, a produção matinal e a produção da tarde;
- Cada animal tem o seu nome, a data de nascimento, o seu número de registo, a identificação de cada progenitor tanto masculino como feminino, bem como o estado de vacinação e o estado de análises e o número do produtor;
- A vaca é um animal e tem o seu preço e o tipo;
- O boi é um animal e tem o seu preço associado;
- Cada Animal tem a sua raça;
- O veterinário é uma pessoa e tem um número que o identifica;
- Na parte de vacinação é cada vacina administrada é identificada por um número, pela data, pelo nome e pelo local onde foi administrada;
- O produtor consegue efetuar vendas e compras de Animais;
- Cada vendedor e cada comprador é identificado pelo nif;
- Cada produtor efetua uma venda de um animal e regista a data o montante que ganhou e o destino para qual vai o animal;
- Cada produtor efetua uma compra de um animal e regista a data o montante gasto e o destino para qual vai o animal;
- O produtor venda ou compra animais e regista o destino para o qual vai o animal se vai para criação ou para abate.

## Entidades

- Fábrica de Leite
- Venda de Leite
- Produtor
- Produção de Leite
- Pessoa
- Veterinário
- Vacinação
- Compra
- Comprador

- Venda
- Vendedor
- Animal
- Raça
- Vaca
- Boi

## Identificação das relações entre entidades (cardinalidade)

- Fábrica vende a Produtor(1:N)
- Produtor tem Animais(1:N)
- Produtor tira Produção de Leite(1:1)
- Veterinário administra Vacinação(1:N)
- Animal toma Vacinação(1:N)
- Animal tem Raça(1:N)
- Animal é vendido Venda(1:N)
- Animal é comprado Compra(1:N)
- Comprador faz Compra(1:N)
- Vendedor participa na Venda(1:N)
- Comprador recebe Venda(1:N)

## Obrigatoriedade

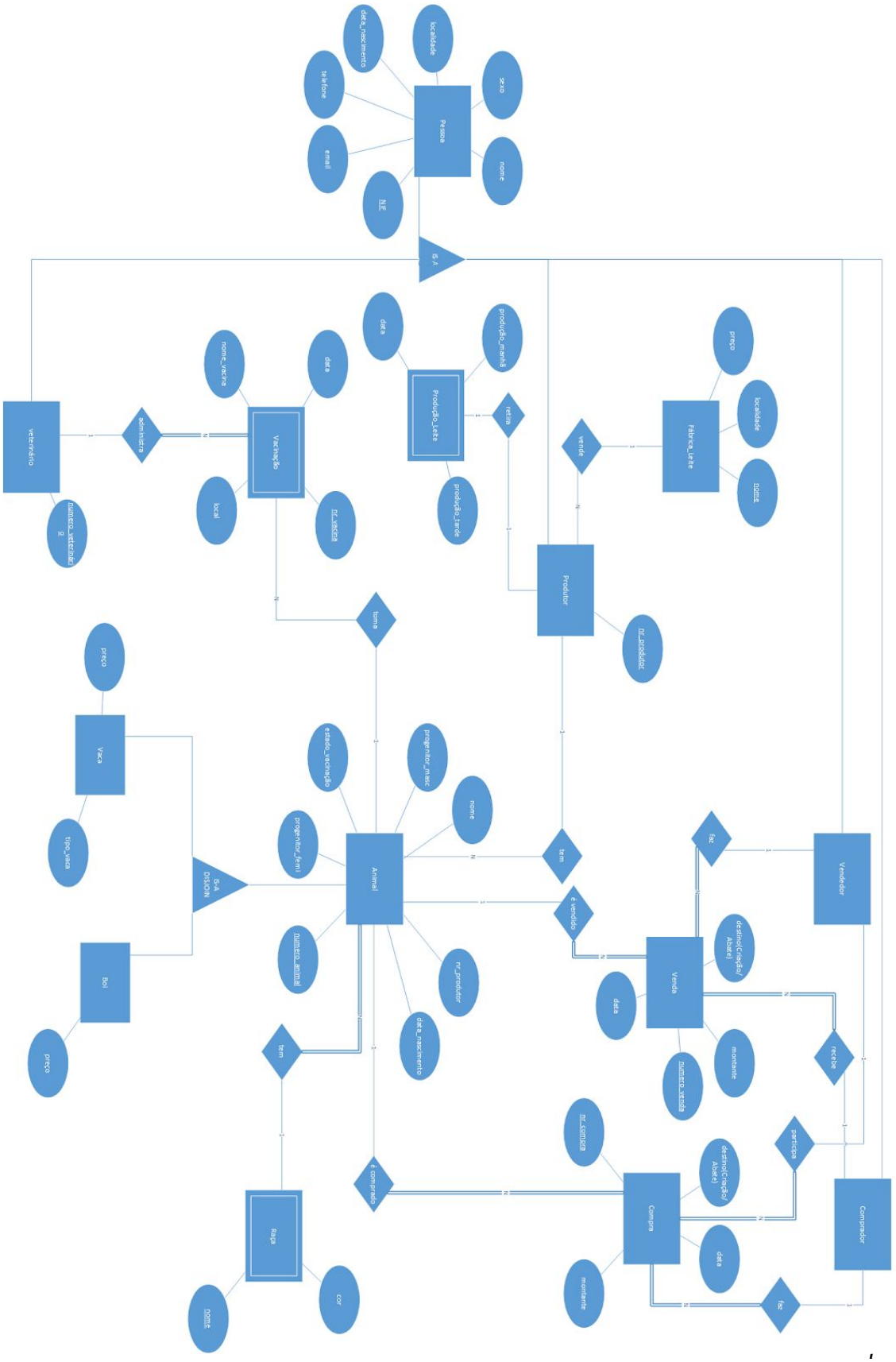
Vendedor faz sempre vendas
Comprador faz sempre vendas
Vendedor participa sempre em compras
Comprador faz sempre compras
Animal é sempre vendido
Animal tem sempre Raça
Veterinário administra sempre Vacinação

## Identificação dos atributos de cada entidade

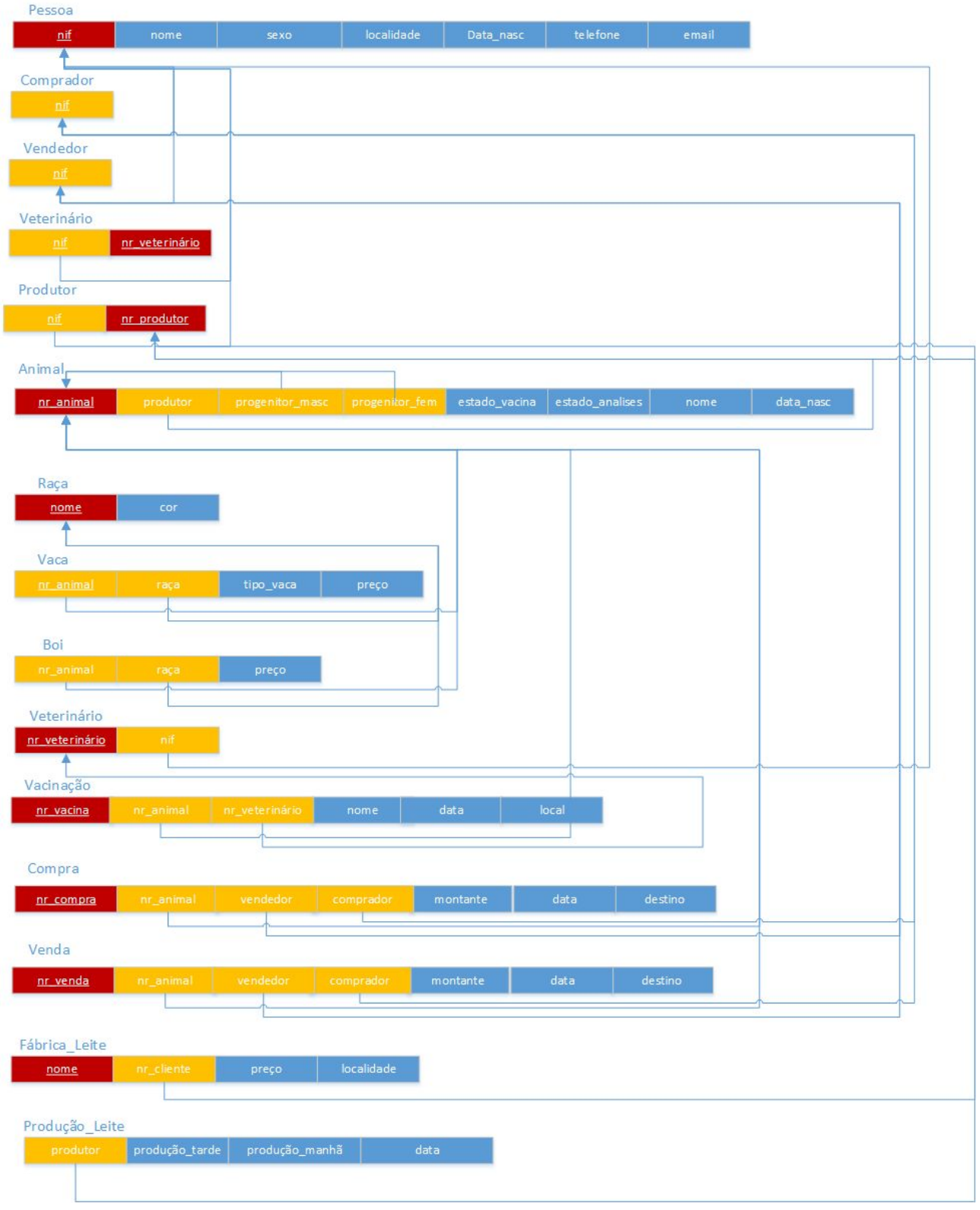
- Fábrica de Leite - nome, preço, localidade
- Venda de Leite - produtor, fabrica
- Produtor - número do produtor, nif
- Produção de Leite - produtor, produção da manhã, produção da tarde, data da produção
- Pessoa - nif, nome, sexo, localidade, data de nascimento, telefone, email
- Veterinário - número de veterinário, nif
- Vacinação - número vacina, número animal, número veterinário, nome, data, local
- Compra - número compra, número animal, vendedor, comprador, montante, data, destino
- Comprador - nif

- Venda - número venda, número animal, vendedor, comprador, montante, data, destino
- Vendedor - nif
- Animal - número animal,  produtor, progenitor masculino, progenitor feminino, estado analises, estado de vacinação, nome, data
- Raça - nome cor
- Vaca -  número animal, raça, tipo de vaca, preço
- Boi - número animal, raça, preço

# Diagrama Entidade Relação



# Esquema Relacional



# Normalização

No nosso projeto não foi necessário fazer a normalização do esquema de relação pois desde a nossa primeira implementação que não possuíamos relações de um para um, logo desta forma já tínhamos a base de dados normalizada.

# SQL - Data Definition Language

Criação da tabela pessoa com definição dos nomes das variáveis, o seu tipo e se aceitam ou não argumentos do tipo “NULL”. Tem como primary key o NIF.

```
CREATE TABLE VACAS.PESSOA (
    NIF             INT             NOT NULL,
    NOME            VARCHAR(50)     NOT NULL,
    SEXO            CHAR(1),
    LOCALIDADE      VARCHAR(50),
    DATA_NASCIMENTO DATE,
    TELEFONE        INT             NOT NULL,
    EMAIL           VARCHAR(20)     NOT NULL,
    PRIMARY KEY(NIF)
);
```

Criação da tabela comprador com definição dos nomes das variáveis, o seu tipo e se aceitam ou não argumentos do tipo “NULL”. Tem como primary key o NIF. Possui também uma chave estrangeira(NIF) que referencia a tabela Pessoa.

```
CREATE TABLE VACAS.COMPRADOR (
    NIF             INT             NOT NULL,
    PRIMARY KEY(NIF),
    FOREIGN KEY(NIF) REFERENCES VACAS.PESSOA (NIF)
);
```

Criação da tabela vendedor com definição dos nomes das variáveis, o seu tipo e se aceitam ou não argumentos do tipo “NULL”. Tem como primary key o NIF. Possui também uma chave estrangeira(NIF) que referencia a tabela Pessoa.

```
CREATE TABLE VACAS.VENDEDOR (
    NIF             INT             NOT NULL,
    PRIMARY KEY(NIF),
    FOREIGN KEY(NIF) REFERENCES VACAS.PESSOA (NIF)
);
```

Criação da tabela veterinário com definição dos nomes das variáveis, o seu tipo e se aceitam ou não argumentos do tipo “NULL”. Tem como primary key o número de veterinário. Possui também uma chave estrangeira(NIF) que referencia a tabela Pessoa.

```
CREATE TABLE VACAS.VETERINARIO (
    NR_VETERINARIO  INT             NOT NULL,
    NIF             INT             NOT NULL,
    PRIMARY KEY(NR_VETERINARIO),
    FOREIGN KEY(NIF) REFERENCES VACAS.PESSOA (NIF)
);
```

Criação da tabela produtor com definição dos nomes das variáveis, o seu tipo e se aceitam ou não argumentos do tipo “NULL”. Tem como primary key o número de produtor. Possui também uma chave estrangeira(NIF) que referencia a tabela Pessoa.

```
CREATE TABLE VACAS.PRODUTOR (
    NR_PRODUTOR     INT             NOT NULL,
    NIF             INT             NOT NULL,
    PRIMARY KEY(NR_PRODUTOR),
    FOREIGN KEY(NIF) REFERENCES VACAS.PESSOA (NIF)
);
```

Criação da tabela animal com definição dos nomes das variáveis, o seu tipo e se aceitam ou não argumentos do tipo “NULL”. Tem como primary key o número de animal. Possui também uma chave estrangeira (Produtor) que referencia a tabela animal.

```
CREATE TABLE VACAS.ANIMAL (
    NR_ANIMAL       INT             NOT NULL,
    PRODUTOR        INT             NOT NULL,
    PROGENITOR_MASC INT             NOT NULL,
    PROGENITOR_FEM  INT             NOT NULL,
    ESTADO_VACINA   VARCHAR(20),
    ESTADO_ANÁLISES VARCHAR(20),
    NOME            VARCHAR(20),
    DATA_NASCIMENTO DATE            NOT NULL,
    PRIMARY KEY(NR_ANIMAL),
    FOREIGN KEY (PRODUTOR) REFERENCES VACAS.PRODUTOR(NR_PRODUTOR)
);
```

Referenciar o progenitor masculino e progenitor feminino do animal ao próprio do animal pois são ambos do tipo animal.

```
ALTER TABLE VACAS.ANIMAL ADD FOREIGN KEY (PROGENITOR_MASC) REFERENCES VACAS.ANIMAL(NR_ANIMAL);
ALTER TABLE VACAS.ANIMAL ADD FOREIGN KEY (PROGENITOR_FEM) REFERENCES VACAS.ANIMAL(NR_ANIMAL);
```

Criação da tabela raça com definição dos nomes das variáveis, o seu tipo e se aceitam ou não argumentos do tipo “NULL”. Tem como primary key o nome que é o nome da raça.

```
CREATE TABLE VACAS.RACA (
    NOME            VARCHAR(30)     NOT NULL,
    COR             VARCHAR(10)     NOT NULL,
    PRIMARY KEY(NOME)
);
```

Criação da tabela vaca com definição dos nomes das variáveis, o seu tipo e se aceitam ou não argumentos do tipo “NULL”. Tem como primary key o número de animal. Possui também uma chave estrangeira raça que referencia a tabela raça.

```
CREATE TABLE VACAS.VACA (
    NR_ANIMAL       INT             NOT NULL,
    RACA            VARCHAR(30)     NOT NULL,
    TIPO_VACA       VARCHAR(10)     NOT NULL,
    PRECO           DECIMAL(10,2),
    PRIMARY KEY(NR_ANIMAL),
    FOREIGN KEY (NR_ANIMAL) REFERENCES VACAS.ANIMAL(NR_ANIMAL),
    FOREIGN KEY (RACA) REFERENCES VACAS.RACA(NOME)
);
```

Criação da tabela boi com definição dos nomes das variáveis, o seu tipo e se aceitam ou não argumentos do tipo “NULL”. Tem como primary key o número de animal. Possui também uma chave estrangeira raça que referencia a tabela raça.

```
CREATE TABLE VACAS.BOI (
    NR_ANIMAL       INT             NOT NULL,
    RACA            VARCHAR(30)     NOT NULL,
    PRECO           DECIMAL(10,2),
    PRIMARY KEY(NR_ANIMAL),
    FOREIGN KEY (NR_ANIMAL) REFERENCES VACAS.ANIMAL(NR_ANIMAL),
    FOREIGN KEY (RACA) REFERENCES VACAS.RACA(NOME)
);
```

Criação da tabela vacinação com definição dos nomes das variáveis, o seu tipo e se aceitam ou não argumentos do tipo “NULL”. Tem como primary key o número da vacina. Possui também uma chave estrangeira número animal que referencia a tabela animal, e número de veterinário que referencia a tabela veterinário.

```
CREATE TABLE VACAS.VACINACAO (
    NR_VACINA       INT             NOT NULL,
    NR_ANIMAL       INT             NOT NULL,
    NR_VETERINARIO  INT             NOT NULL,
    NOME            VARCHAR(30)     NOT NULL,
    [DATA]          DATE            NOT NULL,
    [LOCAL]         VARCHAR(20),
    PRIMARY KEY(NR_VACINA),
    FOREIGN KEY (NR_ANIMAL) REFERENCES VACAS.ANIMAL(NR_ANIMAL),
    FOREIGN KEY (NR_VETERINARIO) REFERENCES VACAS.VETERINARIO(NR_VETERINARIO)
);
```

Criação da tabela compra com definição dos nomes das variáveis, o seu tipo e se aceitam ou não argumentos do tipo “NULL”. Tem como primary key o número da compra. Possui também uma chave estrangeira número animal que referencia a tabela animal, comprador que referencia o nif da tabela comprador e vendedor que referencia o nif da tabela vendedor.

```
CREATE TABLE VACAS.COMPRA (
    NR_COMPRA       INT             NOT NULL,
    NR_ANIMAL       INT             NOT NULL,
    VENDEDOR        INT             NOT NULL,
    COMPRADOR       INT,
    MONTANTE        DECIMAL(10,2)   NOT NULL,
    [DATA]          DATE            NOT NULL,
    DESTINO         VARCHAR(10),
    PRIMARY KEY(NR_COMPRA),
    FOREIGN KEY (NR_ANIMAL) REFERENCES VACAS.ANIMAL(NR_ANIMAL),
    FOREIGN KEY (COMPRADOR) REFERENCES VACAS.COMPRADOR(NIF),
    FOREIGN KEY (VENDEDOR) REFERENCES VACAS.VENDEDOR(NIF)
);
```

Criação da tabela venda com definição dos nomes das variáveis, o seu tipo e se aceitam ou não argumentos do tipo “NULL”. Tem como primary key o número da venda. Possui também uma chave estrangeira número animal que referencia a tabela animal, comprador que referencia o nif da tabela comprador e vendedor que referencia o nif da tabela vendedor.

```
CREATE TABLE VACAS.VENDA (
    NR_VENDA        INT             NOT NULL,
    NR_ANIMAL       INT             NOT NULL,
    VENDEDOR        INT             NOT NULL,
    COMPRADOR       INT,
    MONTANTE        DECIMAL(10,2)   NOT NULL,
    [DATA]          DATE            NOT NULL,
    DESTINO         VARCHAR(10),
    PRIMARY KEY(NR_VENDA),
    FOREIGN KEY (NR_ANIMAL) REFERENCES VACAS.ANIMAL(NR_ANIMAL),
    FOREIGN KEY (COMPRADOR) REFERENCES VACAS.COMPRADOR(NIF),
    FOREIGN KEY (VENDEDOR) REFERENCES VACAS.VENDEDOR(NIF)
);
```

Criação da tabela fábrica do leite com definição dos nomes das variáveis, o seu tipo e se aceitam ou não argumentos do tipo “NULL”. Tem como primary key o nome da fábrica.

```
CREATE TABLE VACAS.FABRICA_LEITE (
    NOME            VARCHAR(20)     NOT NULL,
    PRECO           DECIMAL(5,2)    NOT NULL,
    LOCALIDADE      VARCHAR(30),
    PRIMARY KEY (NOME)
);
```

Criação da tabela produção do leite com definição dos nomes das variáveis, o seu tipo e se aceitam ou não argumentos do tipo “NULL”. Possui uma foreign key (produtor) que referencia o número de produtor da tabela produtor.

```
CREATE TABLE VACAS.PRODUCAO_LEITE (
    PRODUTOR        INT             NOT NULL,
    PRODUCAO_MANHA  DECIMAL(10,2),
    PRODUCAO_TARDE  DECIMAL(10,2),
    FOREIGN KEY (PRODUTOR) REFERENCES VACAS.PRODUTOR(NR_PRODUTOR)
);
```

Criação da tabela venda de leite com definição dos nomes das variáveis, o seu tipo e se aceitam ou não argumentos do tipo “NULL”. Possui uma foreign key (produtor) que referencia o número de produtor da tabela produtor e fábrica que referencia o nome na tabela fábrica do leite.

```
CREATE TABLE VACAS.VENDA_LEITE (
    PRODUTOR        INT             NOT NULL,
    FABRICA         VARCHAR(20)     NOT NULL,
    FOREIGN KEY (PRODUTOR) REFERENCES VACAS.PRODUTOR(NR_PRODUTOR),
    FOREIGN KEY (FABRICA) REFERENCES VACAS.FABRICA_LEITE(NOME)

);
```

# Stored Procedures

- Este stored procedure serve para fazer a leitura da base de dados da tabela vaca.

```
GO
CREATE PROCEDURE VACAS.VERVACAS
AS
SELECT * FROM VACAS.ANIMAL JOIN VACAS.VACA ON ANIMAL.NR_ANIMAL = vaca.NR_ANIMAL
GO
```

- Este stored procedure serve para fazer a leitura da base de dados da tabela boi.

```
GO
CREATE PROCEDURE VACAS.VERBOIS
AS
SELECT * FROM VACAS.ANIMAL JOIN VACAS.BOI ON ANIMAL.NR_ANIMAL = BOI.NR_ANIMAL
GO
```

- Este stored procedure serve para editar os detalhes da vaca na tabela vaca e na tabela animal.

```
GO
CREATE PROCEDURE VACAS.EDITAR_VACA @nrAnimal INT, @nome VARCHAR(20),
@produtor int, @progenitorMasc int, @progenitorFem int, @preco int,
@raca varchar(30), @dataNasc date, @estadoVacina varchar(20),
@estadoAnalises varchar(20), @tipoVaca varchar(10)
AS
UPDATE VACAS.ANIMAL SET NOME = @nome, PRODUTOR = @produtor,
PROGENITOR_MASC = @progenitorMasc, PROGENITOR_FEM = @progenitorFem,
DATA_NASCIMENTO = @dataNasc, ESTADO_VACINA = @estadoVacina,
ESTADO_ANÁLISES = @estadoAnalises
WHERE VACAS.ANIMAL.NR_ANIMAL = @nrAnimal
UPDATE VACAS.VACA SET RACA = @raca, TIPO_VACA = @tipoVaca, PRECO = @preco
WHERE NR_ANIMAL = @nrAnimal
RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para adicionar à tabela Produção_Leite o registo do leite retirado por dia da parte da manhã e da tarde

```
CREATE PROCEDURE VACAS.ADD_PRODUCAO_LEITE @produtor INT, @producao_manha INT,
@producao_tarde INT, @data VARCHAR(20)
AS
INSERT INTO VACAS.PRODUCAO_LEITE(PRODUTOR, PRODUCAO_MANHA, PRODUCAO_TARDE,
DATA_PRODUCAO) VALUES (@produtor, @producao_manha, @producao_tarde,@data);
RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para adicionar uma vacina na tabela Vacinação e edita o animal colocado o seu estado de vacina como Vacinado

```
CREATE PROCEDURE VACAS.ADD_VACINACAO @nr_animal int,
@nr_veterinario int, @nome VARCHAR(50), @data varchar(20),@local varchar(50)
AS
DECLARE @index INT
SET @index = (SELECT COUNT(*) FROM VACAS.VACINACAO)
INSERT INTO VACAS.VACINACAO(NR_VACINA, NR_ANIMAL,NR_VETERINARIO,NOME,DATA,LOCAL)
VALUES (@index+1, @nr_animal, @nr_veterinario, @nome, @data, @local);

UPDATE VACAS.ANIMAL SET VACAS.ANIMAL.ESTADO_VACINA = 'Vacinado'
WHERE VACAS.ANIMAL.NR_ANIMAL = @nr_animal

RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para editar a vacina atualizando a tabela vacinação

```
CREATE PROCEDURE VACAS.EDITAR_VACINACAO @nr_vacina INT, @nr_animal int,
@nr_veterinario int, @nome VARCHAR(50), @data varchar(20),@local varchar(50)
AS
UPDATE VACAS.VACINACAO SET NR_VACINA = @nr_vacina, NR_ANIMAL = @nr_animal, NOME = @nome,
DATA = @data , LOCAL = @local
WHERE VACAS.VACINACAO.NR_VACINA = @nr_vacina
RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para remover a vacina removendo a vacina da tabela vacinação e fazendo o update da tabela animal passado o estado de vacina a null

```
CREATE PROCEDURE VACAS.REMOVER_VACINACAO @nr_vacina int, @nr_animal int
AS
BEGIN
DELETE FROM VACINACAO WHERE NR_VACINA = @nr_vacina
UPDATE VACAS.ANIMAL SET ESTADO_VACINA = NULL
WHERE VACAS.ANIMAL.NR_ANIMAL = @nr_animal
END
RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para adicionar um comprador à tabela comprador e à tabela pessoa

```
CREATE PROCEDURE VACAS.ADD_COMPRADOR @nif INT, @nome varchar(50),
@sexo char(1), @localidade VARCHAR(50), @data varchar(20),@telefone int, @email varchar(50)
AS
INSERT INTO VACAS.PESSOA(NIF, NOME, SEXO,LOCALIDADE,DATA_NASCIMENTO,EMAIL)
VALUES (@nif, @nome, @sexo, @localidade, @data, @email);
INSERT INTO VACAS.COMPRADOR(NIF)
VALUES (@nif)
RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para atualizar o comprador na tabela comprador e atualiza a tabela pessoa

```
CREATE PROCEDURE VACAS.EDITAR_COMPRADOR @nif INT, @nome varchar(50),
@sexo char(1), @localidade VARCHAR(50), @data varchar(20),@telefone int, @email varchar(50)
AS
UPDATE VACAS.PESSOA SET NIF = @nif, NOME = @nome, SEXO = @sexo, LOCALIDADE = @localidade, DATA_NASCIMENTO = @data,
TELEFONE = @telefone, EMAIL = @email
WHERE VACAS.PESSOA.NIF = @nif
UPDATE VACAS.COMPRADOR SET NIF = @nif
RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para remover o comprador da tabela comprador e da tabela pessoa

```
CREATE PROCEDURE VACAS.REMOVER_COMPRADOR @nif INT, @nome varchar(50),
@sexo char(1), @localidade VARCHAR(50), @data varchar(20),@telefone int, @email varchar(50)
AS
BEGIN
DELETE FROM PESSOA WHERE NIF = @nif
DELETE FROM COMPRADOR WHERE NIF = @nif
END
RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para adicionar um vendedor à tabela pessoa e à tabela vendedor

```
CREATE PROCEDURE VACAS.ADD_VENDEDOR @nif INT, @nome varchar(50),
@sexo char(1), @localidade VARCHAR(50), @data varchar(20),@telefone int, @email varchar(50)
AS
INSERT INTO VACAS.PESSOA(NIF, NOME, SEXO,LOCALIDADE,DATA_NASCIMENTO,EMAIL)
VALUES (@nif, @nome, @sexo, @localidade, @data, @email);
INSERT INTO VACAS.VENDEDOR(NIF)
VALUES (@nif)
RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para atualizar o vendedor na tabela vendedor e na tabela pessoa

```
CREATE PROCEDURE VACAS.EDITAR_VENDEDOR @nif INT, @nome varchar(50),
@sexo char(1), @localidade VARCHAR(50), @data varchar(20),@telefone int, @email varchar(50)
AS
UPDATE VACAS.PESSOA SET NIF = @nif, NOME = @nome, SEXO = @sexo, LOCALIDADE = @localidade, DATA_NASCIMENTO = @data,
TELEFONE = @telefone, EMAIL = @email
WHERE VACAS.PESSOA.NIF = @nif
UPDATE VACAS.VENDEDOR SET NIF = @nif
RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para remover o vendedor da tabela vendedor e da tabela pessoa

```
CREATE PROCEDURE VACAS.REMOVER_VENDEDOR @nif INT, @nome varchar(50),
@sexo char(1), @localidade VARCHAR(50), @data varchar(20),@telefone int, @email varchar(50)
AS
BEGIN
DELETE FROM PESSOA WHERE NIF = @nif
DELETE FROM VENDEDOR WHERE NIF = @nif
END
RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para adicionar uma venda à tabela venda e apaga o animal da tabela animal

```
CREATE PROCEDURE VACAS.ADD_COMPRA @nr_venda int, @nr_animal int,
@vendedor int, @comprador int, @montante int, @data varchar(20), @destino varchar(50)
AS
INSERT INTO VACAS.VENDA(NR_VENDA,NR_ANIMAL,VENDEDOR,COMPRADOR,MONTANTE,DATA,DESTINO)
VALUES (@nr_venda, @nr_animal, @vendedor, @comprador, @montante, @data,@destino);

DELETE FROM ANIMAL WHERE NR_ANIMAL = @nr_animal

RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para atualizar a venda fazendo o update da tabela venda, da tabela animal e da tabela comprador

```
CREATE PROCEDURE VACAS.EDITAR_VENDA  @nr_compra int, @nr_animal int,
@comprador int, @montante int, @data varchar(20), @destino varchar(50)
AS
UPDATE VACAS.VENDA SET NR_VENDA = @nr_compra, NR_ANIMAL = @nr_animal, COMPRADOR = @comprador,
MONTANTE = @montante, DATA = @data, DESTINO = @destino
WHERE VACAS.VENDA.NR_VENDA = @nr_compra
UPDATE VACAS.ANIMAL SET NR_ANIMAL = @nr_animal
UPDATE VACAS.COMPRADOR SET NIF = @comprador

RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para remover uma venda da tabela venda

```
CREATE PROCEDURE VACAS.REMOVER_VENDA @nr_venda int
AS
BEGIN
DELETE FROM VENDA WHERE NR_VENDA = @nr_venda
END
RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para adicionar uma compra à tabela compra adicionando o animal à tabela animal

```
CREATE PROCEDURE VACAS.ADD_COMPRA @nr_compra int, @nr_animal int,
@vendedor int, @comprador int, @montante int, @data varchar(20), @destino varchar(50)
AS
INSERT INTO VACAS.COMPRA(NR_COMPRA,NR_ANIMAL,VENDEDOR,COMPRADOR,MONTANTE,DATA,DESTINO)
VALUES (@nr_compra, @nr_animal, @vendedor, @comprador, @montante, @data,@destino);

INSERT INTO VACAS.ANIMAL(NR_ANIMAL)
VALUES (@nr_animal)
RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para atualizar uma compra atualizando a tabela compra, a tabela animal e a tabela comprador e a tabela vendedor

```
CREATE PROCEDURE VACAS.EDITAR_COMPRA @nr_compra int, @nr_animal int,
@vendedor int, @comprador int, @montante int, @data varchar(20), @destino varchar(50)
AS
UPDATE VACAS.COMPRA SET NR_COMPRA = @nr_compra, NR_ANIMAL = @nr_animal, VENDEDOR = @vendedor,
COMPRADOR = @comprador, MONTANTE = @montante, DATA = @data, DESTINO = @destino
WHERE VACAS.COMPRA.NR_COMPRA = @nr_compra
UPDATE VACAS.ANIMAL SET NR_ANIMAL = @nr_animal
UPDATE VACAS.COMPRADOR SET NIF = @comprador
UPDATE VACAS.VENDEDOR SET NIF = @vendedor

RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para remover uma  compra da tabela compra

```
CREATE PROCEDURE VACAS.REMOVER_COMPRA @nr_compra int, @nr_animal int,
@vendedor int, @comprador int, @montante int, @data varchar(20), @destino varchar(50)
AS
BEGIN
DELETE FROM COMPRA WHERE NR_COMPRA = @nr_compra
END
RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para ver a tabela da Vacinação

```
CREATE PROCEDURE VACAS.VER_VACINACAO
AS
SELECT * FROM VACAS.VACINACAO JOIN VACAS.ANIMAL ON VACINACAO.NR_ANIMAL = ANIMAL.NR_ANIMAL
RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para ver a tabela da Venda

```
CREATE PROCEDURE VACAS.VER_VENDAS
AS
SELECT * FROM VACAS.VENDA
RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para ver a tabela da Compra

```
CREATE PROCEDURE VACAS.VER_COMPRAS
AS
SELECT * FROM VACAS.COMPRA
RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para ver a tabela da Produção de Leite, a tabela Venda_Leite e a tabela Fabrica_Leite

```
CREATE PROCEDURE VACAS.VER_PRODUCAO_LEITE
AS
SELECT VACAS.PRODUCAO_LEITE.PRODUTOR, VACAS.PRODUCAO_LEITE.PRODUCAO_MANHA,
 VACAS.PRODUCAO_LEITE.PRODUCAO_TARDE, VACAS.PRODUCAO_LEITE.DATA_PRODUCAO,
VACAS.FABRICA_LEITE.NOME, VACAS.FABRICA_LEITE.PRECO
FROM VACAS.PRODUCAO_LEITE
join VACAS.VENDA_LEITE on VACAS.PRODUCAO_LEITE.PRODUTOR = VACAS.VENDA_LEITE.PRODUTOR
inner join VACAS.FABRICA_LEITE on VACAS.VENDA_LEITE.FABRICA = VACAS.FABRICA_LEITE.NOME
RETURN @@ROWCOUNT
GO
```

- Este stored procedure serve para ver a tabela da Produção de Leite, a tabela Venda_Leite e a tabela Fabrica_Leite

```
CREATE PROCEDURE VACAS.VER_COMPRADOR
AS
SELECT VACAS.PESSOA.NIF, VACAS.PESSOA.NOME, VACAS.PESSOA.SEXO, VACAS.PESSOA.LOCALIDADE,
 VACAS.PESSOA.DATA_NASCIMENTO, VACAS.PESSOA.TELEFONE, VACAS.PESSOA.EMAIL
 FROM VACAS.PESSOA JOIN VACAS.COMPRADOR ON VACAS.PESSOA.NIF = VACAS.COMPRADOR.NIF
RETURN @@ROWCOUNT
GO
```

# Triggers

- Verifica se o número do animal existe na tabela Animal, caso se verifique dá erro.
- Verifica também se o número do progenitor tanto masculino como feminino existe, caso não, dá erro.

```
CREATE TRIGGER add_animal ON VACAS.ANIMAL
INSTEAD OF INSERT
AS

BEGIN
    DECLARE @ANIMAL_ID INT
    SELECT @ANIMAL_ID = NR_ANIMAL FROM inserted
    IF EXISTS(SELECT VACAS.ANIMAL.NR_ANIMAL FROM VACAS.ANIMAL WHERE VACAS.ANIMAL.NR_ANIMAL = @ANIMAL_ID)
            RAISERROR('ERRO NR ANIMAL JA EXISTE', 16,1)
END

BEGIN
    DECLARE @ANIMAL_PROG_MASC INT
    SELECT @ANIMAL_PROG_MASC = PROGENITOR_MASC FROM inserted
    IF NOT EXISTS(SELECT VACAS.ANIMAL.NR_ANIMAL FROM VACAS.ANIMAL WHERE VACAS.ANIMAL.PROGENITOR_MASC = @ANIMAL_PROG_MASC)
            RAISERROR('ERRO NR ANIMAL NAO EXISTE', 16,1)
END
BEGIN
    DECLARE @ANIMAL_PROG_FEM INT
    SELECT @ANIMAL_PROG_FEM = PROGENITOR_FEM FROM inserted
    IF NOT EXISTS(SELECT VACAS.ANIMAL.NR_ANIMAL FROM VACAS.ANIMAL WHERE VACAS.ANIMAL.PROGENITOR_FEM = @ANIMAL_PROG_FEM)
            RAISERROR('ERRO NR ANIMAL NAO EXISTE', 16,1)
END
GO
```

# User Defined Functions

- Esta função vai buscar os dados todos inseridos na base de dados relativos à produção da manhã e da tarde do produtor e multiplica pelo preço da leite da fábrica respetiva, calculando assim o montante recebido pelo produtor nas produções inseridas na base de dados.

```
CREATE FUNCTION VACAS.RENDIMENTO_LEITE() RETURNS DECIMAL(10,2)
AS
BEGIN
    DECLARE @total decimal(10,2)
    SELECT @total = SUM(PRODUCAO_MANHA*PRECO + PRODUCAO_TARDE*PRECO)
    FROM VACAS.PRODUCAO_LEITE JOIN VACAS.VENDA_LEITE ON PRODUCAO_LEITE.PRODUTOR=
    VENDA_LEITE.PRODUTOR JOIN VACAS.FABRICA_LEITE ON FABRICA = NOME
    return @total
END
```

# Conclusão

Neste trabalho desenvolvemos uma plataforma para ajudar os produtores agrícolas a gerir o seu negócio. Como é normal em todos os projetos, encontrámos algumas dificuldades no desenvolvimento desta plataforma mas conseguimos dar a volta da melhor forma.

Com esta plataforma o utilizador tem mais controlo sobre os seus animais pois consegue ter uma vista geral de quantos animais possui, e ao escolher um animal tem uma perspetiva simples das informações deste. Também a nível de saúde, que é bastante importante, consegue ter controlo sobre a vacinação dos seus animais. As análises ao sangue também são importantes e também é possível verificar quais os animais que já as fizeram e os que ainda não.

Para um produtor agrícola com um negócio pequeno, talvez esta plataforma não seja uma grande ajuda mas para negócios maiores será uma ferramenta necessária e poupará tempo além de agilizar processos antes feitos em papel.

Não foram utilizados índices pois não se achou necessário neste contexto o uso dos mesmos.

Com este trabalho, consolidámos a importância de uma base de dados num sistema e aprendemos a implementar numa pequena escala uma base de dados operacional com uma interface simples para o utilizador.