Escola Superior de Tecnologia e Gestão de Beja

Curso de Engenharia de Informática
(regime nocturno)

Base de Dados II

Relatório do 1º Trabalho

Alunos:
Miguel Bilro Murta Soares nº2863
José Afonso Esteves Janeiro nº2467

Docente:
Artur Lança

2008

Introdução

O armazenamento dos dados na SGBD deve estar estruturado de forma a que a sua manipulação seja o mais eficaz possível do ponto de vista da consulta e actualização dos dados, garantindo assim o maior desempenho possível.

A rapidez com que são efectuadas as transições nas bases de dados depende da estrutura da organização dos dados e do modo como é efectuado o acesso a essas estruturas.

Num sistema informático existem dois tipos de memória:

A memória principal, também conhecida como memória RAM, que por ser constituída por componentes electrónicos, apenas mantém os dados enquanto o sistema (computador) estiver ligado.

A memória secundária, construída com base na tecnologia magnética, que ao contrário da memória principal mantém os dados por períodos de tempo longos.

As principais diferenças destes dois tipos de memória são os seguintes:

- No que se refere á capacidade de armazenamento, devido ao aumento de dados que é habitual numa base de dados, a memória principal na grande maioria dos casos não é suficientemente grande para poder suportar todos os dados de uma base de dados.
- Na memória principal existe o problema da volatilidade, isto é, uma falha de energia significa a perca dos dados da memória central. Por segurança uma base de dados deve ser guardada na memória secundária para evitar um acidente deste tipo.

- O tempo de resposta da memória principal é muito superior ao tempo de resposta da memória secundária.
- Em termos de funcionalidade, a memória principal é essencial, uma vez que os dados não podem ser processados directamente da memória secundária, e têm que ser sempre transferidos para a memória principal o que a torna insubstituível.

Em termos das bases de dados, a memória secundária é utilizada para armazenar a base de dados actualizada e para consulta em tempo real.

Considerando a memória principal como a memória directamente acessível pelo processador, pode ser considerada uma hierarquia de memórias, desde os registos

internos do processador (memória principal) até à memória secundária, em que as memórias de menor capacidade e mais rápidas servem de cache a memórias de maior capacidade e mais lentas. Quando uma solicitação feita pelo processador não pode ser satisfeita pelo nível mais alto da hierarquia de memórias, acedesse ao nível imediatamente mais a baixo de memória e assim sucessivamente, funcionando cada nível da hierarquia de memória como uma cache do nível inferior.

O factor que mais influencia todos os aspectos que definem um nível interno de uma base de dados é o tempo de resposta.

A diferença de velocidade de acesso entre a memória central e a memória secundária vai definir o modo como os dados vão ser organizados na memória secundária e também o modo como vão ser acedidos. O tempo de acesso à memória principal é muito menor que o tempo de acesso à memória secundária, sendo o principal objectivo o mínimo número de acessos à memória secundária.

RAID (Redundant Array of Independent Drivers)

RAID é um meio de criar um subsistema de armazenamento de dados composto por vários discos individuais, com a finalidade de ganhar segurança e desempenho.

As vantagens da utilização de RAIDs são essencialmente:

- Ganho no desempenho no acesso aos dados;
- Redundância em caso de falha num dos discos;
- Uso múltiplo de várias unidades de discos;
- Facilidade na recuperação de conteúdos “perdidos”

No nosso trabalho optámos por aplicar o RAID nível 5 é composto por quatro ou mais discos e os blocos de paridade são distribuídos por todos os discos do array, oferecendo assim um grande desempenho e simultaneamente, tolerância a falhas.

Esta distribuição oferece duas vantagens: vários pedidos de escrita podem ser processados em paralelo uma vez que o engarrafamento de um único checkdisk foi eliminado. Por outro lado, os pedidos de leitura têm um nível mais alto de paralelismo.

Uma vez que os dados são distribuídos por todos os discos, os pedidos de leitura, consequentemente, envolvem todos os discos.

O sistema de RAID nível 5 oferece a melhor performance de todos os níveis de RAID com redundância para pedidos grandes e pequenos de leitura e grandes pedidos de escrita.

Porém, para pequenos pedidos de escrita necessitam de um ciclo de modificação leitura/escrita, o que torna esta situação menos eficiente que o RAID nível 1.

De modo a aumentar o desempenho de leitura, o tamanho de cada segmento em que os dados são divididos, pode ser optimizado para o array que estiver a ser utilizado.

A informação sobre paridade é distribuída por todos os discos. A escrita precisa de paridade actualizada e a leitura pode ser feita em todas as unidades ao mesmo tempo.

Descrição do problema

A AgroTejo é uma exploração agrícola com produção de animais de várias espécies (vacas, cavalos, porcos, ovelhas, etc.), na ordem dos milhares.

1. Sobre os vários tipos de animais deve conhecer-se o número do SIA que é um número atribuído pelo Serviço de Identificação Animal, a espécie, o número particular, que é o número atribuído pela AgroTejo, a data de nascimento, a data da morte quando esta ocorrer, o sexo e se está disponível para venda, pois os animais apenas estão disponíveis para venda após os seis meses de idade.

2. Esta exploração agrícola, prima pelo apuramento das raças dos animais que produz, para tal torna-se importante saber quais são os progenitores dos animais que nasceram na AgroTejo.

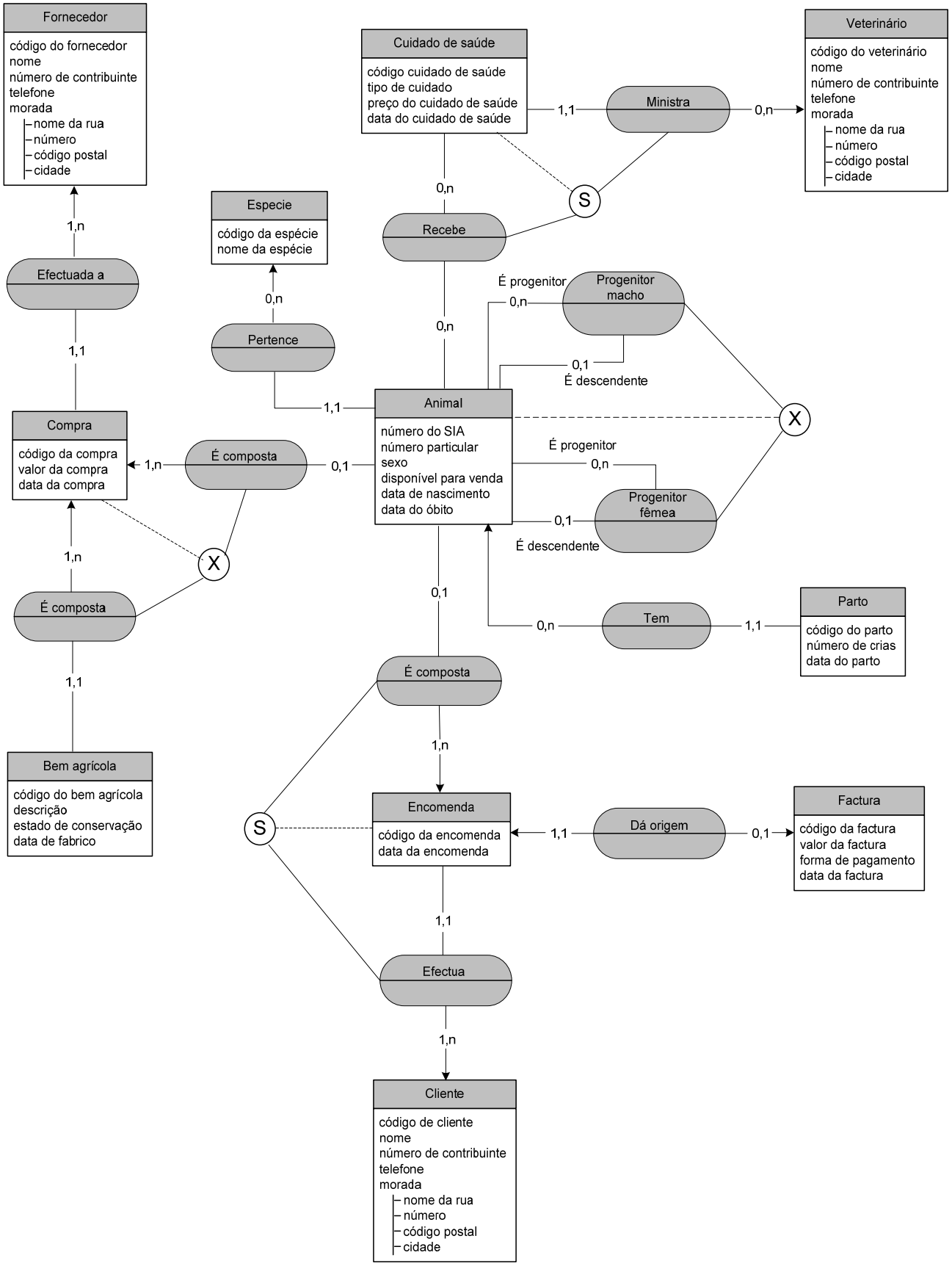
3. Deve conhecer-se a data dos partos das fêmeas e o número de animais que deram à luz nos vários partos, bem como o veterinário que os efectuou.

4. Os animais recebem em massa ou individualmente cuidados de saúde (desparasitação externa, desparasitação interna, vários tipos de vacinas, vários tipos de análises, cirurgias, etc.), sendo importante conhecer o tipo de cuidado prestado, a data em que foi efectuado, o preço e o veterinário que efectuou esse cuidado de saúde.

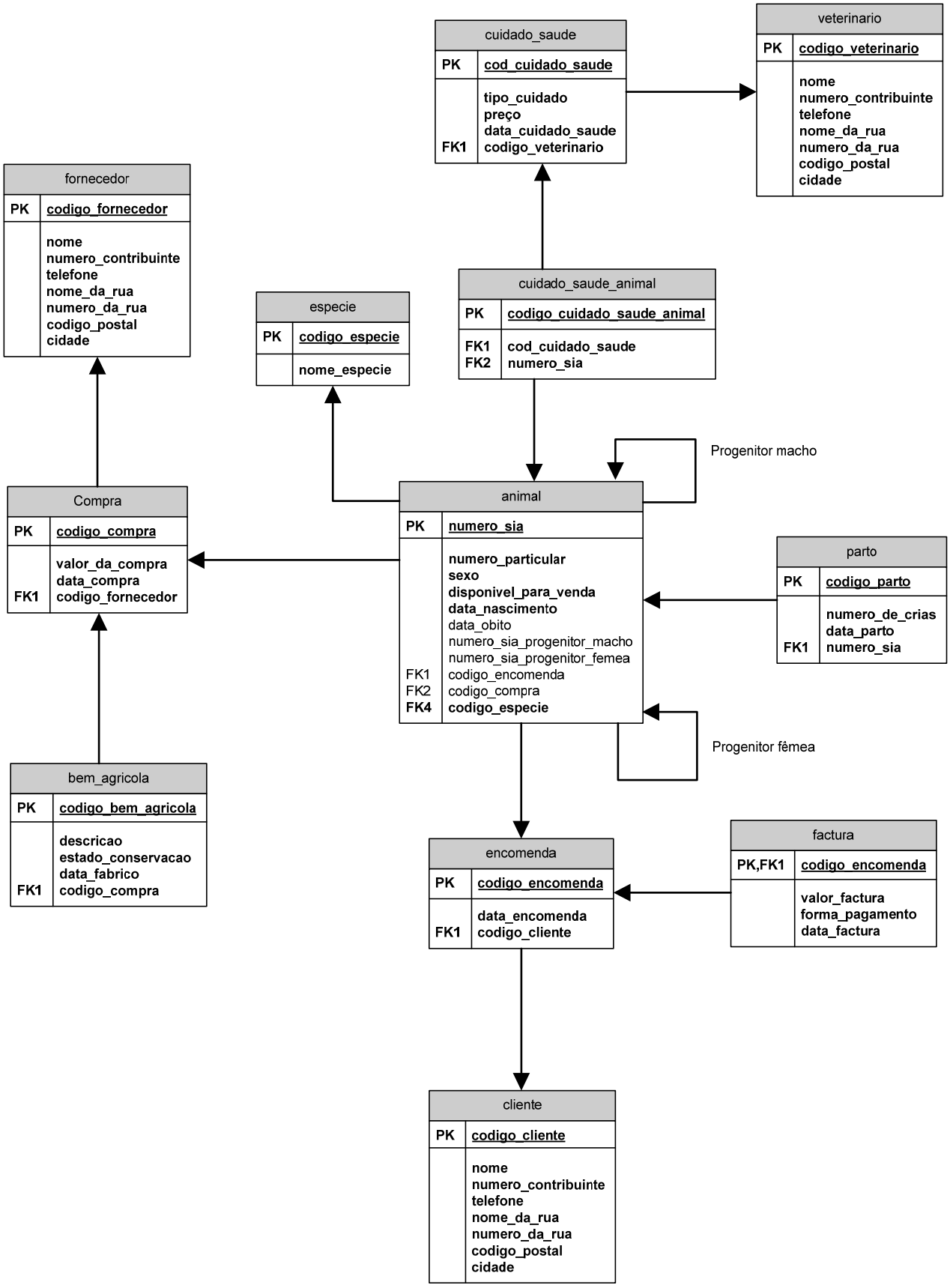
5. A AgroTejo compra animais e recebe encomendas de animais para venda, sendo importante conhecer a data em que efectuou cada compra e a data em que recebeu cada encomenda, o valor da compra ou da venda dos animal, bem como os dados dos clientes que efectuaram as encomendas do ou dos animais e dos vendedores a quem a AgroTejo comprou gado.

6. Para que a AgroTejo possa funcionar da melhor forma, esta possui bens agrícolas, sendo importante conhecer o tipo de bem, a data da aquisição, a data de fabrico, o estado de conservação e a quem foi comprado o bem.

MCD - Modelo Conceptual de dados



MFD - **Modelo Físico de Dados**



Processo de criação da BD

Os ficheiros das bases de dados podem ser agrupados em filegroups, para alocação e fins administrativos.

Existem dois tipos de filegroups, os filegroups primários e os secundários. Os filegroups primários têm extensão .mdf e contêm a informação para a inicialização da base de dados e apontam para outros ficheiros da mesma. Os filegroups secundários têm extensão .ndf e podem ser utilizados para distribuir dados por vários discos, pondo cada ficheiro num disco diferente.

As bases de dados em SQL Server 2005 têm ainda outro tipo de ficheiros, os ficheiros de log, que contêm informação necessária para recuperar todas as transacções da base de dados. A extensão dos ficheiros de log é .ldf.

Assim sendo, na criação da base de dados da AgroTejo, foi criado um filegroup primário, três filegroups secundários e dois ficheiros de log. O filegroup primário tem associado um ficheiro primário que tem como nome lógico *Agro*. O primeiro filegroup secundário chama-se *Animal* e é constituído por três ficheiros com o nome lógico *Agro_Ani1*, *Agro_Ani2* e *Agro_Ani3*. O segundo filegroup secundário chama-se *Encomenda* e é constituído por dois ficheiros com o nome lógico *Agro_Enc1* e *Agro_Enc2*. O terceiro filegroup secundário chama-se *Venda* e é constituído por um ficheiro de nome lógico *Agro_Ven*. Foram ainda criados dois ficheiros log de nome lógico *Agro_log1* e *Agro_log2*.

```
CREATE DATABASE AgroTejo
ON PRIMARY
      (NAME = Agro,
            FILENAME = 'C:\DATABASES\AgroTejo\Agro.MDF',
            SIZE = 150MB,
            FILEGROWTH = 15MB),
FILEGROUP Animal
      (NAME = Agro_Ani1,
            FILENAME = 'C:\DATABASES\AgroTejo\Agro_Ani1.NDF',
            SIZE = 10MB,
            FILEGROWTH = 2MB),
      (NAME = Agro_Ani2,
            FILENAME = 'C:\DATABASES\AgroTejo\Agro_Ani2.NDF',
            SIZE = 10MB,
            FILEGROWTH = 2MB),
```

```
	(NAME = Agro_Ani3,
		FILENAME = 'C:\DATABASES\AgroTejo\Agro_Ani3.NDF',
		SIZE = 10MB,
		FILEGROWTH = 2MB),
FILEGROUP Encomenda
	(NAME = Agro_Enc1,
		FILENAME = 'C:\DATABASES\AgroTejo\Agro_Enc1.NDF',
		SIZE = 5MB,
		FILEGROWTH = 1MB),
	(NAME = Agro_Enc2,
		FILENAME = 'C:\DATABASES\AgroTejo\Agro_Enc2.NDF',
		SIZE = 5MB,
		FILEGROWTH = 2MB),
FILEGROUP Compra
	(NAME = Agro_Comp,
		FILENAME = 'C:\DATABASES\AgroTejo\Agro_Comp.NDF',
		SIZE = 2MB,
		FILEGROWTH = 10%)
LOG ON
	(NAME = Agro_Log1,
		FILENAME = 'C:\DATABASES\AgroTejo\Agro_Log1.LDF',
		SIZE = 3MB,
		FILEGROWTH = 4%,
		MAXSIZE = 10MB),
	(NAME = Agro_Log2,
		FILENAME = 'C:\DATABASES\AgroTejo\Alent_Log2.LDF',
		SIZE = 3MB,
		FILEGROWTH = 4%,
		MAXSIZE = 10MB)
```

Após a criação da base de dados, dos seus filegroups e ficheiros de log, criámos os seguintes user data types:

```
CREATE TYPE cod_postal
FROM Varchar(7) NOT NULL;

CREATE TYPE telef
FROM NUMERIC(9,0) NOT NULL;

CREATE TYPE num_contribuinte
FROM NUMERIC (9,0) NOT NULL;
```

De seguida criámos as tabelas, sendo estas distribuídas pelos diferentes filegroups. Todas as tabelas se encontram na terceira forma normal, ou seja, uma entidade está na 3FN quando estiver na 2FN e nenhum atributo que não seja chave primária depender de outro que também não seja chave.

```
create table veterinario(
      codigo_veterinario smallint identity (1,1) primary key,
      nome varchar(55) not null,
      numero_contribuinte num_contribuinte not null,
      nome_da_rua varchar(40) not null,
      numero_da_rua smallint not null,
      codigo_postal cod_postal not null,
      telefone telef not null,
      cidade varchar(20) not null,
) on Animal


create table cuidado_saude(
      cod_cuidado_saude smallint identity (1,1) primary key,
      tipo_cuidado varchar(30) not null,
      preco money not null,
      data_cuidado_saude datetime not null,
      codigo_veterinario smallint not null,
      foreign key (codigo_veterinario)
            references veterinario (codigo_veterinario)
            on update cascade
            on delete no action
) on Animal



create table fornecedor(
      codigo_fornecedor smallint identity (1,1) primary key,
      nome varchar(55) not null,
      numero_contribuinte num_contribuinte not null,
      telefone telef not null,
      nome_da_rua varchar(40) not null,
      numero_da_rua smallint not null,
      codigo_postal cod_postal not null,
      cidade varchar(20) not null,
) on Compra



create table cliente(
      codigo_cliente smallint identity (1,1) primary key,
      nome varchar(55) not null,
      numero_contribuinte num_contribuinte not null,
      telefone telef not null,
      nome_da_rua varchar(40) not null,
      numero_da_rua smallint not null,
      codigo_postal cod_postal not null,
      cidade varchar(30) not null,
) on Encomenda
```

```
create table compra(
      codigo_compra smallint identity (1,1) primary key,
      valor_da_compra money not null,
      data_compra datetime not null,
      codigo_fornecedor smallint not null,
    foreign key (codigo_fornecedor)
            references fornecedor (codigo_fornecedor)
            on update cascade
            on delete no action
) on Compra



create table animal(
      numero_sia smallint identity (1,1) primary key,
      numero_particular smallint not null,
      sexo varchar(5) not null,
      disponivel_para_venda varchar(3) not null,
      data_nascimento datetime not null,
      data_obito datetime,
      numero_sia_progenitor_macho smallint,
      numero_sia_progenitor_femea smallint,
      codigo_encomenda smallint,
      foreign key (codigo_encomenda)
            references encomenda (codigo_encomenda)
            on update cascade
            on delete set null,
      codigo_compra smallint,
      foreign key (codigo_compra)
            references compra (codigo_compra)
            on update cascade
            on delete set null,
      codigo_especie smallint not null,
      foreign key (codigo_especie)
            references especie (codigo_especie)
            on update cascade
            on delete no action
) on Animal



reate table parto(
      codigo_parto smallint identity (1,1) primary key,
      numero_de_crias smallint not null,
      data_parto datetime not null,
      numero_sia smallint not null,
    foreign key (numero_sia)
            references animal (numero_sia)
            on update cascade
            on delete no action
) on Animal
```

```
create table cuidado_saude_animal(
      codigo_cuidado_saude_animal smallint identity (1,1) primary key,
      cod_cuidado_saude smallint not null,
    foreign key (cod_cuidado_saude)
            references cuidado_saude (cod_cuidado_saude)
            on update cascade
            on delete no action,
      numero_sia smallint not null,
      foreign key (numero_sia)
            references animal (numero_sia)
            on update cascade
            on delete no action
) on Animal



create table bem_agricola(
      codigo_bem_agricola smallint identity (1,1) primary key,
      descricao varchar(40) not null,
      estado_conservacao varchar(30) not null,
      data_fabrico datetime not null,
      codigo_compra smallint not null,
    foreign key (codigo_compra)
            references compra (codigo_compra)
            on update cascade
            on delete no action
) on Compra



create table factura(
      codigo_encomenda smallint identity (1,1) primary key,
      valor_da_factura money not null,
      data_factura datetime not null,
      foreign key (codigo_encomenda)
            references encomenda (codigo_encomenda)
            on update no action
            on delete no action
) on Encomenda



create table encomenda(
      codigo_encomenda smallint identity (1,1) primary key,
      data_encomenda datetime not null,
      codigo_cliente smallint not null,
    foreign key (codigo_cliente)
            references cliente (codigo_cliente)
            on update cascade
            on delete no action
) on Encomenda



create table especie(
      codigo_especie smallint identity (1,1) primary key,
      nome_especie varchar(20) not null,
) on Animal
```