



Índice

Descrição de contexto	2
Diagrama de classes UML	3
Esquema Relacional	4
Instruções LDD-SQL.....	5
Instruções LDM-SQL	11

DESCRIÇÃO DE CONTEXTO

Nesta segunda entrega continuámos a expansão do nosso projeto relativo ao jardim zoológico, desta vez com o esquema relacional, instruções LDD-SQL para a criação da base de dados e ainda algumas instruções LMD-SQL que permitem preencher a base de dados com informação fictícia.

Todo este trabalho foi relativo ao diagrama de classes UML que sofreu uma atualização, sendo que agora está mais completo com mais alguns atributos e restrições.

```

classDiagram
    class Cliente {
        - QuantidadeDeVisitas int
    }
    class Pessoa {
        - CC int
        - Idade int
        - NIF int
        - Nome string
        - Sexo char
    }
    class Tratador {
        - DataDeInicio string
    }
    class Espécie {
        - NomeCientifico string
        - NomeComum string
    }
    class TipoDeAlimento {
        - NomeTipoDeAlimento string
        - Stock int
    }
    class Animal {
        - DataNascimento string
        - DataObito string
        - Nome string
        - Sexo char
    }
    class Habit {
        - Area double
        - Descricao string
        - NomeHabit string
    }
    class Evento {
        - DataEvento string
        - Duracao double
        - NomeEvento string
    }
    class TipoDeObjeto {
        - Descricao string
        - IDTipoDeObjeto int
    }
    class Quantidade {
        - Quantidade int
    }

    Cliente "1" -- "1..*" Pessoa
    Cliente "1" -- "1..*" Tratador
    Cliente "1" -- "1..*" Espécie
    Cliente "1" -- "1..*" TipoDeAlimento
    Cliente "1" -- "1..*" Animal
    Cliente "1" -- "1..*" Habit
    Cliente "1" -- "1..*" Evento
    Cliente "1" -- "1..*" TipoDeObjeto

    Pessoa "1" -- "1..*" Tratador
    Pessoa "1" -- "1..*" Espécie
    Pessoa "1" -- "1..*" TipoDeAlimento
    Pessoa "1" -- "1..*" Animal
    Pessoa "1" -- "1..*" Habit
    Pessoa "1" -- "1..*" Evento
    Pessoa "1" -- "1..*" TipoDeObjeto

    Tratador "1" -- "1..*" Espécie
    Tratador "1" -- "1..*" TipoDeAlimento
    Tratador "1" -- "1..*" Animal
    Tratador "1" -- "1..*" Habit
    Tratador "1" -- "1..*" Evento
    Tratador "1" -- "1..*" TipoDeObjeto

    Espécie "1" -- "1..*" TipoDeAlimento
    Espécie "1" -- "1..*" Animal
    Espécie "1" -- "1..*" Habit
    Espécie "1" -- "1..*" Evento
    Espécie "1" -- "1..*" TipoDeObjeto

    TipoDeAlimento "1" -- "1..*" Animal
    TipoDeAlimento "1" -- "1..*" Habit
    TipoDeAlimento "1" -- "1..*" Evento
    TipoDeAlimento "1" -- "1..*" TipoDeObjeto

    Animal "1" -- "1..*" Habit
    Animal "1" -- "1..*" Evento
    Animal "1" -- "1..*" TipoDeObjeto

    Habit "1" -- "1..*" Evento
    Habit "1" -- "1..*" TipoDeObjeto

    Evento "1" -- "1..*" TipoDeObjeto

    TipoDeObjeto "1" -- "1..*" Quantidade
  
```

The diagram illustrates the relationships between various entities in a pet shop system. Key classes include Cliente, Pessoa, Tratador, Espécie, Tipo de Alimento, Animal, Habit, Evento, and Tipo de Objeto. Each class has specific attributes and constraints. Relationships are defined with multiplicity and role names, such as 'Possui' (Owns) and 'Participa' (Participates). Constraints are provided for many attributes, such as age ranges for Pessoa and stock levels for Tipo de Alimento.

ESQUEMA RELACIONAL

Pessoa(CC , Idade, NIF, Nome, Sexo)

Cliente(CC, Quantidade de Visitas)

Bilhete(idBilhete, Hora de Entrada, Preço, Tipo, idTrajeto->Trajeto)

ClienteCompraBilhete(CC-> Cliente, idBilhete -> Bilhete)

Trajeto(idTrajeto, Descricao)

Seccao(idSeccao, Area, Descricao)

TrajetoIncluiSeccao(idTrajeto -> Trajeto, idSeccao -> Seccao)

Habitat(NomeHabitat, Area, Descricao, idSeccao -> Seccao)

TipodeObjeto(idTipodeObjeto, Descricao)

HabitatContemTipodeObjeto(NomeHabitat -> Habitat, idTipodeObjeto -> TipodeObjeto, Quantidade)

Animal(NomeAnimal, DataNascimento, DataObito, Sexo, NomeHabitat -> Habitat, NomeCientifico -> Especie)

Evento(NomeEvento, Data, Duracao, Hora de Inicio)

AnimalParticipaEvento(NomeAnimal -> Animal, [NomeEvento, DataEvento] -> Evento)

Especie(NomeCientifico, NomeComum)

TipodeAlimento(NomeTipodeAlimento, Stock)

EspecieComeTipodeAlimento(NomeCientifico -> Especie, NomeTipodeAlimento -> TipodeAlimento)

Tratador(CC, DataInicio)

Especialidade(Area)

TratadorTemEspecialidade(CC -> Tratador, Area -> Especialidade)

INSTRUÇÕES LDD-SQL

```
CREATE TABLE TratadorTemEspecialidade (  
    Tratador      TEXT,  
    Especialidade TEXT,  
    PRIMARY KEY(Tratador,Especialidade),  
    FOREIGN KEY(Tratador) REFERENCES Tratador(CC),  
    FOREIGN KEY(Especialidade) REFERENCES Especialidade(Area) );
```

```
CREATE TABLE Tratador (  
    Pessoa TEXT,  
    DataDeInicio NUMERIC NOT NULL,  
    PRIMARY KEY(Pessoa),  
    FOREIGN KEY(Pessoa) REFERENCES Pessoa(CC) );
```

```
CREATE TABLE TrajetoIncluiSeccao (  
    Trajeto INTEGER,  
    Seccao INTEGER,  
    PRIMARY KEY(Trajeto,Seccao),  
    FOREIGN KEY(Trajeto) REFERENCES Trajeto(IDTrajeto),  
    FOREIGN KEY(Seccao) REFERENCES Seccao(IDSeccao) );
```

```
CREATE TABLE Trajeto (  
    IDTrajeto      INTEGER,  
    Descricao      TEXT,  
    PRIMARY KEY(IDTrajeto),  
    CHECK(IDTrajeto > 0) );
```

```
CREATE TABLE TipoDeObjecto (  
    IDTipoDeObjecto    INTEGER,  
    Descricao    TEXT NOT NULL,  
    PRIMARY KEY(IDTipoDeObjecto),  
    CHECK(IDTipoDeObjecto > 0) );
```

```
CREATE TABLE TipoDeAlimento (  
    NomeTipoDeAlimento TEXT,  
    Stock    INTEGER NOT NULL,  
    PRIMARY KEY(NomeTipoDeAlimento) );
```

```
CREATE TABLE Seccao (  
    IDSeccao    INTEGER,  
    Area    REAL NOT NULL,  
    Descricao    TEXT NOT NULL,  
    PRIMARY KEY(IDSeccao),  
    CHECK(IDSeccao > 0) );
```

```
CREATE TABLE Pessoa (  
    CC    TEXT,  
    Nome TEXT NOT NULL,  
    Idade INTEGER NOT NULL,  
    Sexo TEXT NOT NULL,  
    NIF    INTEGER NOT NULL UNIQUE,  
    PRIMARY KEY(CC)  
    CHECK(Idade > 0 AND NIF > 99999999 AND NIF < 1000000000) );
```

```
CREATE TABLE HabitatContemTipoDeObjecto (  
    Habitat      TEXT,  
    TipoDeObjecto INTEGER,  
    QuantidadeObjecto  INTEGER NOT NULL,  
    PRIMARY KEY(Habitat,TipoDeObjecto),  
    FOREIGN KEY(Habitat) REFERENCES Habitat(NomeHabitat),  
    FOREIGN KEY(TipoDeObjecto) REFERENCES TipoDeObjecto(IDTipoDeObjecto) );
```

```
CREATE TABLE Habitat (  
    NomeHabitat  TEXT,  
    Area  REAL NOT NULL,  
    Descricao  TEXT NOT NULL,  
    Seccao  INTEGER,  
    PRIMARY KEY(NomeHabitat),  
    FOREIGN KEY(Seccao) REFERENCES Seccao(IDSeccao),  
    CHECK(Area > 0) );
```

```
CREATE TABLE Evento (  
    NomeEvento  TEXT,  
    Data  NUMERIC,  
    Duracao  INTEGER NOT NULL,  
    HoraDeInicio  INTEGER NOT NULL,  
    PRIMARY KEY(NomeEvento,Data),  
    CHECK(Duracao > 0) );
```



```
CREATE TABLE EspecieComeTipoDeAlimento (
    Especie TEXT,
    TipoDeAlimento TEXT,
    PRIMARY KEY(Especie,TipoDeAlimento),
    FOREIGN KEY(Especie) REFERENCES Especie(NomeCientifico),
    FOREIGN KEY(TipoDeAlimento) REFERENCES
TipoDeAlimento(NomeTipoDeAlimento) );
```

```
CREATE TABLE Especie (
    NomeCientifico TEXT,
    NomeComum TEXT,
    PRIMARY KEY(NomeCientifico) );
```

```
CREATE TABLE Especialidade (
    Area TEXT,
    PRIMARY KEY(Area) );
```

```
CREATE TABLE Cliente (
    Pessoa TEXT,
    Quantidade_de_visitas INTEGER NOT NULL,
    PRIMARY KEY(Pessoa),
    FOREIGN KEY(Pessoa) REFERENCES Pessoa(CC)
    CHECK(Quantidade_de_visitas >= 0) );
```

```
CREATE TABLE Bilhete (
  IDBilhete      INTEGER,
  Hora_De_Entrada  NUMERIC,
  Preco  REAL NOT NULL,
  Tipo   TEXT NOT NULL,
  Trajeto INTEGER,
  PRIMARY KEY(IDBilhete),
  FOREIGN KEY(Trajeto) REFERENCES Trajeto(IDTrajeto),
  CHECK(Preco >= 0 AND IDBilhete > 0) );
```

```
CREATE TABLE AnimalParticipaEvento (
  Animal TEXT,
  NomeEvento  TEXT,
  DataEvento  NUMERIC,
  PRIMARY KEY(Animal, NomeEvento, DataEvento),
  FOREIGN KEY(Animal) REFERENCES Animal(NomeAnimal),
  FOREIGN KEY(NomeEvento, DataEvento) REFERENCES Evento(NomeEvento,
DataEvento) );
```

```
CREATE TABLE ClienteCompraBilhete (
  Cliente TEXT,
  Bilhete INT,
  PRIMARY KEY(Cliente, Bilhete),
  FOREIGN KEY(Cliente) REFERENCES Cliente(CC),
  FOREIGN KEY(Bilhete) REFERENCES Bilhete(IDBilhete) );
```

```

CREATE TABLE Animal (
    NomeAnimal TEXT,
    DataNascimento NUMERIC,
    DataObito NUMERIC,
    Sexo TEXT NOT NULL,
    Espécie TEXT,
    Habitat TEXT,
    PRIMARY KEY(NomeAnimal),
    FOREIGN KEY(Espécie) REFERENCES Espécie(NomeCientífico),
    FOREIGN KEY(Habitat) REFERENCES Habitat(NomeHabitat) );

CREATE TABLE TratadorCuidaAnimal (
    Tratador TEXT,
    Animal TEXT,
    PRIMARY KEY(Tratador, Animal)
    FOREIGN KEY(Tratador) REFERENCES Tratador(CC),
    FOREIGN KEY(Animal) REFERENCES Animal(NomeAnimal) );

CREATE TRIGGER IncrementBilhete
AFTER INSERT ON ClienteCompraBilhete

UPDATE Cliente SET Quantidade_de_visitas = Quantidade_de_visitas + 1,
[WHERE CC = Cliente];

```

INSTRUÇÕES LDM-SQL

INSERTS

```
INSERT INTO TipoDeObjecto VALUES (1, "Pedra <5cm");
INSERT INTO TipoDeAlimento VALUES ("Carne", 10);
INSERT INTO Seccao VALUES (1, 10, "Primeira seccao");
INSERT INTO Habitat VALUES ("Savana", 5, "Habitat Savana");
INSERT INTO HabitatContemTipoDeObjecto VALUES ("Savana", 1);
INSERT INTO Trajeto VALUES (1, "");
INSERT INTO TrajetoIncluiSeccao(1, 1);
INSERT INTO Especie VALUES ("Canis lupus", "Lobo");
INSERT INTO EspecieComeTipoDeAlimento VALUES ("Canis lupus","Carne");
INSERT INTO Animal VALUES ("Goncalo", 1990, NULL, "M", "Canis lupus", "Savana");
INSERT INTO Pessoa VALUES ("1234", "Eduardo", 23, "M", 123456789);
INSERT INTO Cliente VALUES ("1234", 0);
INSERT INTO Bilhete VALUES (1, NULL, 1.5, "Especial", 1);
INSERT INTO ClienteCompraBilhete VALUES ("1234", 1);
INSERT INTO Pessoa VALUES ("4321", "Mario", 20, "M", 987654321);
INSERT INTO Tratador VALUES ("4321", "12-10-2015");
INSERT INTO TratadorTemEspecialidade VALUES ("Alimentador");
INSERT INTO TratadorCuidaAnimal VALUES("4321", "Goncalo");
INSERT INTO Evento VALUES ("Primeiro evento", "11-02-2015")
INSERT INTO AnimalParticipaEvento VALUES ("Goncalo", "Primeiro evento", "11-02-2015");
```

INSTRUÇÕES LDM-SQL

SELECTS

1.Pergunta

Listagem de trajetos, com a respectiva descrição, valor faturado e contagem de bilhetes vendidos por ordem decrescente de valor de vendas e hora de entrada inferior ou igual a 12.

Se o nome do trajeto for nulo, colocar como indefinido.

1.Resposta

Select Trajeto , Coalesce(Descricao, 'Indefinido') as "Descricao" , Sum(Preco) as "Valor Total" , count(*) as "Número de bilhetes"

From Bilhete Left Join Trajeto On Bilhete.Trajeto = Trajeto.idtrajeto

Where Hora_De_Entrada <= 12

Group By Trajeto , Descricao

Having Sum(preco) > 10

Order By [Valor Total] desc

O left join foi utilizado para que os bilhetes sejam contados mesmo que o trajeto tenha sido apagado. (na prática isto nunca acontece porque existe uma relação entre as duas tabelas, mas resolvemos deixar para efeitos académicos)

2.Pergunta

Quantos animais de cada espécie existem em cada habitat?

2.Resposta

Select h.NomeHabitat , h.Descricao , a.especie , Count(*) As "Contagem"

From Habitat as h Join Animal as a On h.NomeHabitat = a.Habitat

Group By h.NomeHabitat , h.Descricao , a.especie

3.Pergunta

Qual é o número de animais por espécie presente em habitats cuja descrição começa por S

3.Resposta

Select especie , Count(*) As "Contagem"

From Animal

Where habitat In (Select NomeHabitat From Habitat Where Descricao Like 'S%')

Group By especie

4.Pergunta

Listagem de todos os tratadores com mais de 30 anos com a especialidade de nutricionista

4.Resposta

```
SELECT p.CC ,p.nome ,p.Idade
FROM tratador AS t
JOIN Pessoa AS p ON t.pessoa = p.cc
WHERE pessoa IN
(
    SELECT tratador
    FROM TratadorTemEspecialidade
    WHERE especialidade = 'Nutricionista'
)
AND p.Idade >= 30
```

5.Pergunta

Alimentos cujo stock se encontra abaixo de 100 unidades.

5.Resposta

```
Select *
From TipoDeAlimento
Where stock < 100
```

6.Pergunta

Listagem de clientes do sexo feminino que visitaram o zoo pelo menos 3 vezes.

6.Resposta

```
Select p.CC ,p.nome ,p.Idade, p.sexo
From cliente as c
JOIN Pessoa as p on c.pessoa = p.cc
Where QuantidadeDeVisitas >= 3
```

7.Pergunta

Quantos bilhetes de cada tipo foram comprados e qual o preço médio, mínimo e máximo?

7.Resposta

```
Select tipo, count(*) as count, avg(preco) as precoAvg, max(preco) as precoMax, min(preco) as precoMin
```

```
From bilhete
```

```
Group by tipo
```

8.Pergunta

Area total das seccoes de cada trajeto?

8.Resposta

```
SELECT ts.trajeto ,t.descricao, Sum(s.area) as Area  
FROM TrajetoIncluiSeccao AS ts  
JOIN seccao AS s ON ts.seccao = s.idseccao  
JOIN trajeto AS t ON ts.trajeto = t.idtrajeto  
GROUP BY ts.trajeto
```

9.Pergunta

Animais dos trajetos Selva e Tundra

9.Resposta

```
Select * from habitat
```

```
Where Descricao = 'Selva'
```

```
UNION
```

```
Select * from habitat
```

```
Where Descricao = 'Tundra'
```

10.Pergunta

Listagem de todos os tipos de objecto

10.Resposta

Select *

From TipoDeObjecto

11.Pergunta

Contagem de bilhetes, contagem de secções e preço médio.

11.Resposta

Select count(distinct idbilhete) nbil, count(distinct ts.seccao) nse , avg(preco) medpr

From Bilhete b

Join TrajetoIncluiSeccao ts On b.Trajeto = ts.Trajeto

Join Habitat h On h.Seccao = ts.seccao
