Escola Secundária Avelar Brotero
Curso: Programação de Sistemas Informáticos
Ano Letivo: 2024/2025
Autores: Simão Rodrigues e Miguel Costa

Projeto de Base de Dados: Aluguer de Veículos

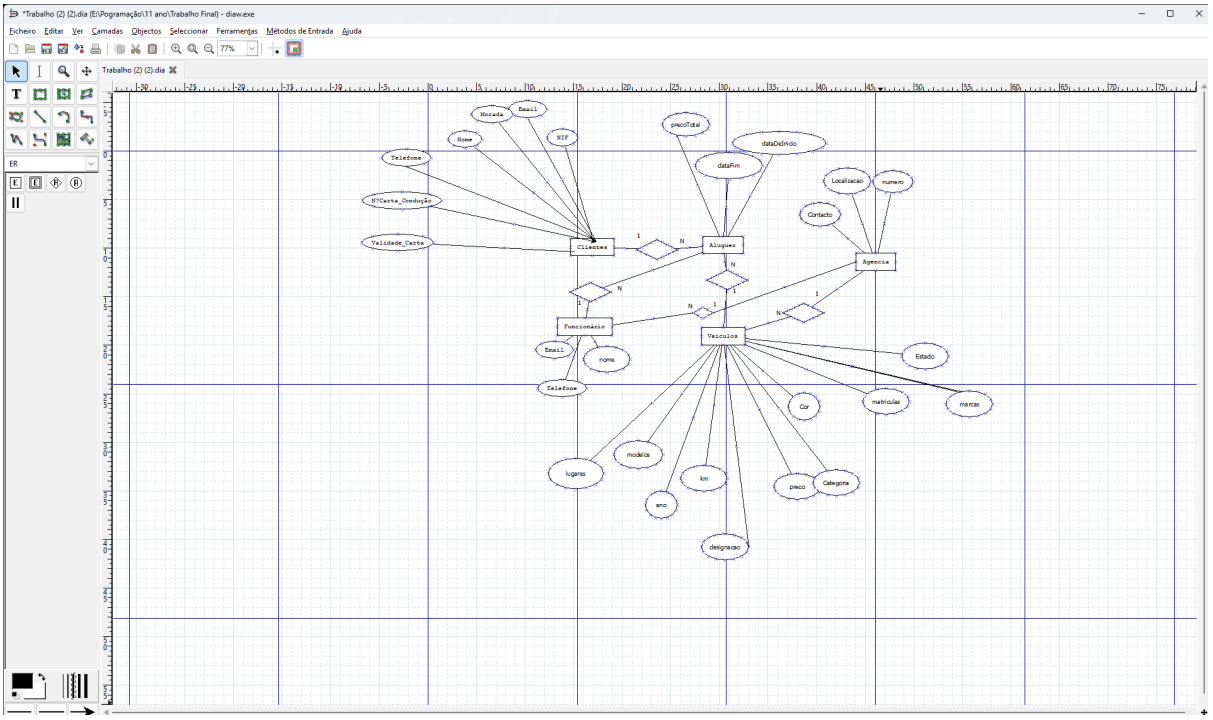
Índice

---

1. Introdução

Este trabalho teve como objetivo o desenvolvimento de uma base de dados relacional para uma empresa fictícia de aluguer de veículos. Com base nos requisitos propostos, foi feita a modelação E-R, a conversão para esquema relacional, a normalização até à 3ª Forma Normal (3FN) e a implementação em SQL. O objetivo principal foi garantir uma estrutura de dados eficiente, sem redundâncias, com integridade referencial e capacidade de expansão futura.

Diagrama ER



Esquema Relacional

CLIENTES(Cliente_ID (PK), Nome, NIF, Morada, Telefone, Email)

AGENCIA(Agencia_ID (PK), Nome, Localizacao, Contacto)

FUNCIONARIOS(Funcionario_ID (PK), Nome, Telefone, Email, Agencia_ID (FK))

VEICULOS(Veiculo_ID (PK), Matricula, Marca, Modelo, Ano, Quilometragem, Cor, PrecoAluguer, Estado, Categoria, Agencia_ID (FK))

ALUGUERES(Aluguer_ID (PK), DataInicio, DataFim, PrecoTotal, Cliente_ID (FK), Veiculo_ID (FK), Funcionario_ID (FK))

NORMALIZAÇAO

✅ Etapa 1 — Verificação da 1ª Forma Normal (1FN)

📘 Definição da 1FN:

Um esquema relacional está na **Primeira Forma Normal (1FN)** se:

- Todos os atributos contêm **valores atómicos** (não listas, nem campos compostos).

- Não há grupos repetidos nem multivalorados.

🔎 Vamos analisar cada entidade:

---

1. `CLIENTES(Cliente_ID, Nome, NIF, Morada, Telefone, Email)`

✅ Todos os atributos são **atómicos**.

🔹 **Cumpre a 1FN**

---

2. `AGENCIA(Agencia_ID, Nome, Localizacao, Contacto)`

✅ Tudo atómico.

🔹 **Cumpre a 1FN**

---

3. `FUNCIONARIOS(Funcionario_ID, Nome, Telefone, Email, Agencia_ID)`

✅ Tudo atómico.

🔹 **Cumpre a 1FN**

---

4. `VEICULOS(...)`

Atributos críticos:

- `Estado`: {"Disponível", "Alugado", "Em manutenção"}
- `Categoria`: {"Citadino", "SUV", etc.}

❌ **Problema identificado**:

- `Estado` e `Categoria` são **valores multivalorados e repetitivos**, o que:
    - **viola a atomicidade**;

- **repete informação textual** em várias linhas.

🔧 **Solução**: Criar entidades separadas:

```
ESTADOS(Estado_ID, Designacao)
CATEGORIAS(Categoria_ID, Designacao)
```

E no `VEICULOS`, substituir:

```
Estado_ID (FK), Categoria_ID (FK)
```

🔹 **Com isto, VEICULOS passa a cumprir a 1FN**

---

5. `ALUGUERES(...)`

✅ Todos os campos são atómicos (datas, preços, FK simples).

🔹 **Cumpre a 1FN**

✅ Etapa 2 — Verificação da 2ª Forma Normal (2FN)

📘 Definição da 2FN:

Um esquema está na **Segunda Forma Normal** se:

- Está na **1FN**
- E **todos os atributos não-chave dependem da totalidade da chave primária** (não pode haver dependência parcial)

🧠 Nota:

- A **dependência parcial** só acontece em **tabelas com chaves compostas**
- Se a tabela tiver **chave primária simples**, ela **cumpre automaticamente a 2FN**

🔍 Análise entidade por entidade:

---

1. `CLIENTES(Cliente_ID (PK), Nome, NIF, Morada, Telefone, Email)`

- Chave primária simples: `Cliente_ID`
- Todos os outros atributos dependem de `Cliente_ID`

✅ **Cumpre 2FN**

---

2. `AGENCIA(Agencia_ID (PK), Nome, Localizacao, Contacto)`

- Chave simples: `Agencia_ID`
- Tudo depende diretamente da PK

✅ **Cumpre 2FN**

---

3. `FUNCIONARIOS(Funcionario_ID (PK), Nome, Telefone, Email, Agencia_ID (FK))`

- PK simples: `Funcionario_ID`

- Tudo depende da PK

✅ **Cumpre 2FN**

---

4. `ESTADOS(Estado_ID (PK), Designacao)`

5. `CATEGORIAS(Categoria_ID (PK), Designacao)`

- Tabelas de referência → PK simples

✅ **Ambas cumprem 2FN**

---

6. `VEICULOS(Veiculo_ID (PK), Matricula, Marca, Modelo, Ano, Quilometragem, Cor, PrecoAluguer, Estado_ID, Categoria_ID, Agencia_ID)`

- Chave simples: `Veiculo_ID`
- Nenhum atributo depende apenas de parte da chave (não há chaves compostas)

✅ **Cumpre 2FN**

---

7. `ALUGUERES(Aluguer_ID (PK), DataInicio, DataFim, PrecoTotal, Cliente_ID, Veiculo_ID, Funcionario_ID)`

- Chave simples: `Aluguer_ID`
- Tudo depende diretamente da chave

✅ **Cumpre 2FN**

✅ Etapa 3 — Verificação da 3ª Forma Normal (3FN)

📘 Definição da 3FN:

Um esquema relacional está na Terceira Forma Normal se:

- **Está na 2FN, e**
- **Todos os atributos não-chave dependem apenas da chave primária, e não de outros atributos não-chave (ou seja, não há dependências transitivas)**

🔍 Verificação entidade por entidade:

---

1. `CLIENTES(Cliente_ID (PK), Nome, NIF, Morada, Telefone, Email)`

- **Nenhum atributo depende de outro que não seja `Cliente_ID`.**
- **`NIF` é único, mas está apenas como atributo (poderia até ser alternativo da PK).**

✅ Cumpre 3FN

---

2. `AGENCIA(Agencia_ID (PK), Nome, Localizacao, Contacto)`

- **Tudo depende diretamente de `Agencia_ID`**
- **`Localizacao` ou `Contacto` não geram dependências transitivas**

✅ Cumpre 3FN

---

3. `FUNCIONARIOS(Funcionario_ID (PK), Nome, Telefone, Email, Agencia_ID (FK))`

- **`Agencia_ID` é uma FK, e não é transitiva (serve para ligação)**
- **Tudo depende diretamente de `Funcionario_ID`**

✅ **Cumpre 3FN**

---

4. `ESTADOS(Estado_ID (PK), Designacao)`

5. `CATEGORIAS(Categoria_ID (PK), Designacao)`

- **Atributos descritivos simples**
- **Sem dependências transitivas**

✅ **Ambas cumprem 3FN**

---

6. `VEICULOS(...)`

- **Todos os atributos (como `Marca`, `Ano`, `PrecoAluguer`) dependem diretamente de `Veiculo_ID`**
- **`Estado_ID` e `Categoria_ID` são FK (já separados como entidades)**

✅ **Cumpre 3FN**

---

7. `ALUGUERES(...)`

- **Nenhum atributo depende de outro não-chave**
- **Tudo está diretamente ligado à PK `Aluguer_ID`**

✅ **Cumpre 3FN**

CLIENTES(Cliente_ID (PK), Nome, NIF, Morada, Telefone, Email)

AGENCIA(Agencia_ID (PK), Nome, Localizacao, Contacto)

FUNCIONARIOS(Funcionario_ID (PK), Nome, Telefone, Email, Agencia_ID (FK))

ESTADOS(Estado_ID (PK), Designacao)

CATEGORIAS(Categoria_ID (PK), Designacao)

VEICULOS(Veiculo_ID (PK), Matricula, Marca, Modelo, Ano, Quilometragem, Cor, PrecoAluguer, Estado_ID (FK), Categoria_ID (FK), Agencia_ID (FK))

ALUGUERES(Aluguer_ID (PK), DataInicio, DataFim, PrecoTotal, Cliente_ID (FK), Veiculo_ID (FK), Funcionario_ID (FK))

Implementação em SQL

```
CREATE DATABASE AluguerVeiculos;
USE AluguerVeiculos;
```

```
CREATE TABLE Clientes (
    Cliente_ID INT PRIMARY KEY AUTO_INCREMENT,
    Nome VARCHAR(100) NOT NULL,
    NIF VARCHAR(9) NOT NULL UNIQUE,
    Morada VARCHAR(200),
    Telefone VARCHAR(15),
    Email VARCHAR(100)
);

CREATE TABLE Agencia (
    Agencia_ID INT PRIMARY KEY AUTO_INCREMENT,
    Nome VARCHAR(100) NOT NULL,
    Localizacao VARCHAR(200),
    Contacto VARCHAR(15)
);

CREATE TABLE Funcionarios (
    Funcionario_ID INT PRIMARY KEY AUTO_INCREMENT,
    Nome VARCHAR(100) NOT NULL,
    Telefone VARCHAR(15),
    Email VARCHAR(100),
    Agencia_ID INT,
    FOREIGN KEY (Agencia_ID) REFERENCES Agencia(Agencia_ID)
);

CREATE TABLE Estados (
    Estado_ID INT PRIMARY KEY AUTO_INCREMENT,
    Designacao VARCHAR(50) NOT NULL
);

CREATE TABLE Categorias (
    Categoria_ID INT PRIMARY KEY AUTO_INCREMENT,
    Designacao VARCHAR(50) NOT NULL
);

CREATE TABLE Veiculos (
    Veiculo_ID INT PRIMARY KEY AUTO_INCREMENT,
    Marca VARCHAR(50),
    Modelo VARCHAR(50),
    Ano INT,
    Cor VARCHAR(30),
    Matricula VARCHAR(15) UNIQUE,
    Km INT,
    Lugares INT,
    Preco DECIMAL(10,2),
    Designacao VARCHAR(100),
    Estado_ID INT,
```

```
    Categoria_ID INT,
    Agencia_ID INT,
    FOREIGN KEY (Estado_ID) REFERENCES Estados(Estado_ID),
    FOREIGN KEY (Categoria_ID) REFERENCES Categorias(Categoria_ID),
    FOREIGN KEY (Agencia_ID) REFERENCES Agencia(Agencia_ID)
);

CREATE TABLE Alugueres (
    Aluguer_ID INT PRIMARY KEY AUTO_INCREMENT,
    Cliente_ID INT,
    Veiculo_ID INT,
    Funcionario_ID INT,
    DataInicio DATE,
    DataFim DATE,
    PrecoTotal DECIMAL(10,2),
    FOREIGN KEY (Cliente_ID) REFERENCES Clientes(Cliente_ID),
    FOREIGN KEY (Veiculo_ID) REFERENCES Veiculos(Veiculo_ID),
    FOREIGN KEY (Funcionario_ID) REFERENCES Funcionarios(Funcionario_ID)
);

INSERT INTO Clientes (Nome, NIF, Morada, Telefone, Email) VALUES
('João Silva', '123456789', 'Rua A, 10', '912345678', 'joao@email.com'),
('Maria Costa', '987654321', 'Rua B, 20', '934567890', 'maria@email.com');

INSERT INTO Agencia (Nome, Localizacao, Contacto) VALUES
('Agencia Centro', 'Lisboa', '211234567'),
('Agencia Norte', 'Porto', '221234567');

INSERT INTO Funcionarios (Nome, Telefone, Email, Agencia_ID) VALUES
('Carlos Mendes', '912222333', 'carlos@email.com', 1),
('Ana Lopes', '913333444', 'ana@email.com', 2);

INSERT INTO Estados (Designacao) VALUES
('Disponível'), ('Alugado'), ('Manutenção');

INSERT INTO Categorias (Designacao) VALUES
('Económico'), ('SUV'), ('Luxo');

INSERT INTO Veiculos (Marca, Modelo, Ano, Cor, Matricula, Km, Lugares, Preco,
Designacao, Estado_ID, Categoria_ID, Agencia_ID) VALUES
('Renault', 'Clio', 2020, 'Branco', 'AA-11-BB', 25000, 5, 35.00, 'Clio Branco', 1, 1, 1),
('BMW', 'X5', 2022, 'Preto', 'CC-22-DD', 10000, 5, 90.00, 'X5 Luxo', 1, 3, 2);

INSERT INTO Alugueres (Cliente_ID, Veiculo_ID, Funcionario_ID, DataInicio, DataFim,
PrecoTotal) VALUES
(1, 1, 1, '2024-06-01', '2024-06-05', 140.00),
(2, 2, 2, '2024-06-10', '2024-06-15', 450.00);
```

📄 Relatório Final – Projeto de Base de Dados: Aluguer de Veículos

1. Introdução

Este trabalho teve como objetivo o desenvolvimento de uma base de dados relacional para uma empresa fictícia de aluguer de veículos. Com base nos requisitos propostos, foi feita a modelação E-R, a conversão para esquema relacional, a normalização até à 3ª Forma Normal (3FN) e a implementação em SQL. O objetivo principal foi garantir uma estrutura de dados eficiente, sem redundâncias, com integridade referencial e capacidade de expansão futura.

---

2. Modelo Entidade-Relacionamento (E-R)

O diagrama E-R construído representa sete entidades principais:

- **Clientes**
- **Agencias**
- **Funcionarios**
- **Veiculos**
- **Alugueres**
- **Estados** (ligado aos veículos)
- **Categorias** (ligado aos veículos)

Cada entidade foi ligada pelas suas relações naturais:

- Um **Funcionário** está associado a uma **Agência**.
- Um **Veículo** pertence a uma **Agência** e está associado a um **Estado** e a uma **Categoria**.
- Um **Aluguer** liga um **Cliente**, um **Veículo** e um **Funcionário responsável**, com as respetivas datas e preço final.

O diagrama representa corretamente os **tipos de relacionamentos** (1:N e N:1) e as **chaves primárias e estrangeiras**.

---

3. Esquema Relacional

A conversão do diagrama E-R para o modelo relacional seguiu o formato correto. Cada entidade transformou-se numa tabela com uma chave primária (`PK`) e, quando necessário, chaves estrangeiras (`FK`):

- `Clientes(Cliente_ID PK, Nome, NIF, Morada, Telefone, Email)`
- `Agencia(Agencia_ID PK, Nome, Localizacao, Contacto)`
- `Funcionarios(Funcionario_ID PK, Nome, Telefone, Email, Agencia_ID FK)`
- `Estados(Estado_ID PK, Designacao)`
- `Categorias(Categoria_ID PK, Designacao)`
- `Veiculos(Veiculo_ID PK, ..., Estado_ID FK, Categoria_ID FK, Agencia_ID FK)`
- `Alugueres(Aluguer_ID PK, DataInicio, DataFim, PrecoTotal, Cliente_ID FK, Veiculo_ID FK, Funcionario_ID FK)`

Todos os relacionamentos foram corretamente implementados através de `FOREIGN KEY`, garantindo a integridade referencial entre entidades.

---

4. Normalização

Foi realizada a análise das três formas normais:

- **1FN (Primeira Forma Normal):** Todos os atributos são atómicos e não existem listas, campos compostos ou grupos repetidos. O problema dos atributos multivalorados em `Estado` e `Categoria` foi resolvido através da criação das tabelas `Estados` e `Categorias`.
- **2FN (Segunda Forma Normal):** Todas as tabelas têm chave primária simples, portanto não existem dependências parciais. Cada atributo não-chave depende diretamente da chave primária.
- **3FN (Terceira Forma Normal):** Verificou-se que não existem dependências transitivas. Por exemplo, `Marca`, `Modelo`, `Ano` e `Preco` em `Veiculos` dependem diretamente de `Veiculo_ID`, e `Categoria`/`Estado` estão normalizados em tabelas

separadas.

Resultado: **todas as tabelas estão corretamente normalizadas até à 3FN**.

---

5. Implementação SQL

Foi criado um **script SQL completo** que inclui:

- Criação da base de dados: `CREATE DATABASE AluguerVeiculos;`
- Criação de tabelas com restrições de integridade (`PRIMARY KEY`, `FOREIGN KEY`)
- Inserção de dados de exemplo para simular um cenário real
- Os campos obrigatórios estão bem definidos (`NOT NULL` em campos essenciais como nomes e NIF)
- Todas as relações foram validadas com sucesso (ex. associação correta de `Funcionario_ID` na tabela `Alugueres`)

---

6. Validação e Testes

O script SQL foi testado com sucesso. A base de dados cria todas as tabelas sem erros e os dados de exemplo são inseridos corretamente. A integridade referencial está assegurada – por exemplo, não é possível registar um aluguer com um veículo ou funcionário inexistente.

---

7. Potenciais Melhorias Futuras

Embora a base de dados esteja funcional e completa, há sempre oportunidades para melhorias futuras:

- **Tabela de pagamentos:** para registar o estado do pagamento (pago, pendente, modo de pagamento).
- **Histórico de manutenção de veículos:** incluir uma tabela `Manutencoes(Veiculo_ID, Data, Tipo, Custo)` para registar manutenções.
- **Login para funcionários/clientes:** permitir autenticação e controlo de acessos (ideal para uma aplicação web associada).

- **Melhor gestão de disponibilidade:** permitir calcular automaticamente a disponibilidade de veículos com base nas datas de aluguer.

---

8. Conclusão

O projeto cumpriu integralmente os objetivos propostos. Foi feita uma modelação sólida com base na análise do problema, e todas as regras de normalização foram corretamente aplicadas. O resultado é uma base de dados eficiente, funcional e preparada para crescer em futuras versões. Este exercício foi fundamental para aplicar conhecimentos de modelação, SQL e boas práticas de desenvolvimento de bases de dados relacionais.