1. Introdução

O presente relatório apresenta o desenvolvimento de um projecto prático de desenho e implementação de uma base de dados relacional da Clínica Médica Viva Mais. O objetivo é aplicar os conceitos de modelagem, normalização, SQL e PL/SQL, além de simular transações e demonstrar mecanismos de integridade e concorrência.

2. Importância da Base de Dados para a Clínica

Uma base de dados relacional permite:
- O armazenamento estruturado e consistente dos dados;
- A redução da redundância e inconsistência;
- O apoio à tomada de decisão administrativa;
- A agilidade no acesso ao histórico clínico e operacional.

3. Modelo Relacional

Chaves Primárias (PK)

paciente(id_paciente); **medico**(id_medico); **consulta**(id_consulta); **medicamento**(id_medicamento); **prescricao**(id_prescricao); **fatura**(id_fatura).

Chaves Estrangeiras (FK)

consulta.id_paciente → **paciente**.id_paciente; **consulta**.id_medico → **medico**.id_medico; **prescricao**.id_consulta → **consulta**.id_consulta; **prescricao**.id_medicamento → **medicamento**.id_medicamento; **fatura**.id_consulta → **consulta**.id_consulta.

As entidades identificadas e os respetivos atributos são:

Paciente

- **id_paciente** (PK, INT) (obrigatório)

- **nome** (VARCHAR 100, NOT NULL) (obrigatório)
- **data_nascimento** (DATE, NOT NULL) (obrigatório)
- **contacto** (VARCHAR 20) (opcional)
- **endereco** (VARCHAR 150) (opcional)

Médico

- **id_medico** (PK, INT) (obrigatório)
- **nome** (VARCHAR 100, NOT NULL) (obrigatório)
- **especialidade** (VARCHAR 50, NOT NULL) (obrigatório)
- **contacto** (VARCHAR 20) (opcional)

Consulta

- **id_consulta** (PK, INT) (obrigatório)
- **data_consulta** (DATE, NOT NULL) (obrigatório)
- **hora_consulta** (TIME, NOT NULL) (obrigatório)
- **diagnostico** (TEXT) (opcional)
- **id_paciente (FK → Paciente)** (obrigatório)
- **id_medico (FK → Médico)** (obrigatório)

Medicamento

- **id_medicamento** (PK, INT) (obrigatório)
- **nome** (VARCHAR 100, NOT NULL) (obrigatório)
- **estoque** (INT, NOT NULL) (obrigatório, >=0)
- **validade** (DATE) (opcional)

Prescrição

- **id_prescricao** (PK, INT) (obrigatório)

- **quantidade** (INT, NOT NULL) (obrigatório, >0)
- **posologia** (VARCHAR 100) (opcional)
- **id_consulta (FK → Consulta)** (obrigatório)
- **id_medicamento (FK → Medicamento)** (obrigatório)

Fatura

- **id_fatura** (PK, INT) (obrigatório)
- **valor** (DECIMAL(10,2), NOT NULL) (obrigatório, >=0)
- **data_pagamento** (DATE) (opcional)
- **id_consulta (FK → Consulta, UNIQUE)** (obrigatório)

4. Relacionamentos e Cardinalidades

a. **Paciente – Consulta**

Um **Paciente** pode ter **0..N Consultas** por outro lado, cada **Consulta** pertence a **1 Paciente → (1:N).**

b. **Médico – Consulta**

Um **Médico** pode atender **0..N Consultas** por outro lado, cada **Consulta** é realizada por **1 Médico→ (1:N).**

c. **Consulta – Prescrição**

Uma **Consulta** pode gerar **0..N Prescrições** por outro lado, cada **Prescrição** refere-se a **1 Consulta→ (1:N).**

d. **Medicamento – Prescrição**

Um **Medicamento** pode estar em **0..N Prescrições** por outro lado, cada **Prescrição** utiliza **1 Medicamento→ (1:N)**

e. **Consulta – Fatura**

Uma **Consulta** pode ter **0 ou 1 Fatura** por outro lado, cada **Fatura** pertence a **1 Consulta→ (1:1 opcional).**

5. Regras de Obrigatoriedade

- **Consulta**: só existe se tiver **Paciente** e **Médico** associados (FK obrigatórias).
- **Prescrição**: só existe se tiver **Consulta** e **Medicamento** (FK obrigatórias).
- **Fatura**: é opcional para uma **Consulta** (uma consulta pode não ser faturada).
- **Atributos opcionais**: contacto (Paciente e Médico), endereco (Paciente), validade (Medicamento), diagnostico (Consulta), posologia (Prescrição), data_pagamento (Fatura).

6. Diagrama Entidade-Relacionamento (DER)

O diagrama encontra-se incluído no anexo deste relatório.

7. **SQL (DDL e DML)**

Foram definidos comandos SQL para criação de tabelas, inserção de dados fictícios, consultas, views e índices. Exemplo de criação da tabela Paciente:

```
CREATE TABLE Paciente (
   id_paciente INT PRIMARY KEY,
   nome VARCHAR(100) NOT NULL,
   data_nascimento DATE NOT NULL,
   contacto VARCHAR(20),
   endereco VARCHAR(150)
);
```

8. **Transações e Concorrência**

As transações permitem garantir a consistência dos dados. Exemplo:

```
START TRANSACTION;
UPDATE Medicamento SET estoque = estoque - 2 WHERE id_medicamento = 1;
INSERT INTO Fatura (id_fatura, id_consulta, valor, data_pagamento)
VALUES (101, 55, 2500.00, '2025-09-19');
COMMIT;
```

Caso ocorra um erro, deve-se utilizar ROLLBACK para reverter.

Situação de concorrência: dois atendentes tentando registrar prescrições para o mesmo medicamento simultaneamente. A solução é o uso de bloqueios (LOCK) ou níveis de isolamento de transações.

9. **PL/SQL**

Exemplos de objetos PL/SQL implementados:
- **Procedure**s: Inserção, actualização e exclusão de médicos, pacientes e medicamentos.
Triggers: Impedir exclusão de Pacientes e Médicos com consultas marcadas.

10. **Conclusão**

O projeto de base de dados para a Clínica Médica Viva Mais demonstra a aplicação prática dos conceitos da disciplina, cobrindo desde a modelagem até a implementação com SQL e PL/SQL. O sistema proposto contribui para a gestão eficiente da informação clínica e administrativa, assegurando integridade, consistência e disponibilidade dos dados.