UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
DEPARTAMENTO DE CIÊNCIAS EXATAS E NATURAIS — PAU DOS FERROS
DOCENTE: BRUNO BORGES DA SILVA
DISCIPLINA: BANCO DE DADOS - PEX1248

ANTONIO ANDSON DE OLIVEIRA ROCHA
FRANCISCO THIAGO DA SILVA PINHEIRO
LEVI FILGUEIRA CHAGAS
LUIZ FELIPE IZIDRO DA SILVA
SOPHIA HELLEN PIRES DA SILVEIRA

SISTEMA DE CONSULTÓRIO/CLÍNICA

PAU DOS FERROS
18/2025

1 Introdução

O sistema proposto é um Sistema de Consultório/Clínica, desenvolvido para gerenciar de forma eficiente as informações e processos essenciais de um ambiente clínico. Ele abrange o cadastro de pacientes e médicos, o agendamento e gerenciamento de consultas, o registro de atendimento e prontuários, além do controle financeiro por meio de pagamentos associados às consultas.

O objetivo principal deste projeto é projetar um banco de dados relacional capaz de suportar as operações diárias de uma clínica médica. O sistema deve garantir organização, integridade e disponibilidade dos dados, permitindo que as atividades administrativas e assistenciais sejam realizadas com precisão, segurança e agilidade.

1.1 Requisitos Detalhados

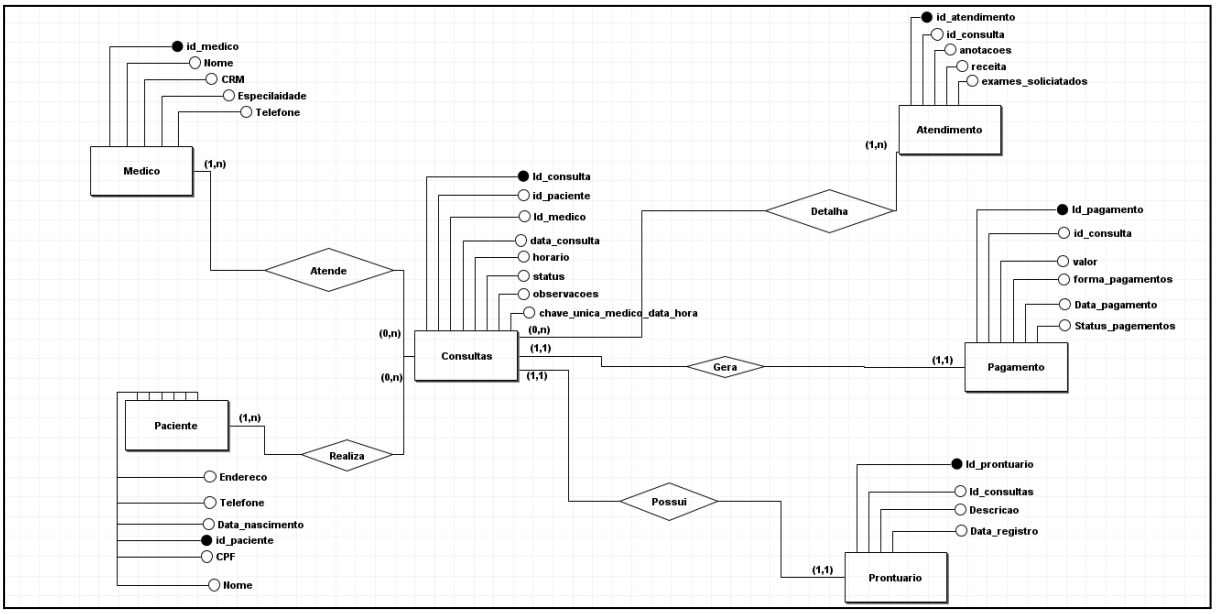
- Requisitos Funcionais
    - Cadastro e Gerenciamento
        - O sistema deve permitir o cadastro de novos pacientes.
        - O sistema deve permitir o cadastro de novos médicos.
        - O sistema deve permitir a atualização, consulta e remoção de pacientes.
        - O sistema deve permitir a atualização, consulta e remoção de médicos.
    - Consultas
        - O sistema deve permitir o agendamento de consultas.
        - O sistema deve permitir alterar o status da consulta (ex.: agendada, confirmada, realizada, cancelada).
        - O sistema deve permitir visualizar consultas por data, paciente ou médico.
        - O sistema deve impedir o agendamento de dois pacientes para o mesmo médico no mesmo dia e horário.
    - Atendimento e Prontuário
        - O sistema deve permitir registrar atendimentos relacionados a uma consulta.
        - O sistema deve permitir registrar um prontuário para cada consulta realizada.
        - O sistema deve permitir visualizar o histórico de atendimentos de um paciente.
        - O sistema deve permitir visualizar o prontuário associado a uma consulta.
    - Pagamentos
        - O sistema deve permitir registrar pagamentos relacionados a consultas.
        - O sistema deve permitir atualizar o status do pagamento (pendente, pago, estornado).
        - O sistema deve permitir consultar valores recebidos por período.
        - O sistema deve permitir consultar pagamentos por paciente.

- Casos de uso
    - Caso de Uso 1 — Agendar Consulta
        - Ator: Recepcionista
        - Pré-condições:
            - Paciente e médico devem estar cadastrados.
            - O horário desejado deve estar livre.
        - Fluxo Principal:
            - O recepcionista seleciona um paciente.
            - O recepcionista seleciona um médico.
            - O recepcionista informa data e horário.
            - O sistema verifica se o médico está disponível naquele horário.
            - O sistema registra a consulta com status “agendada”.
        - Pós condição:
            - Consulta cadastrada e visível na agenda.

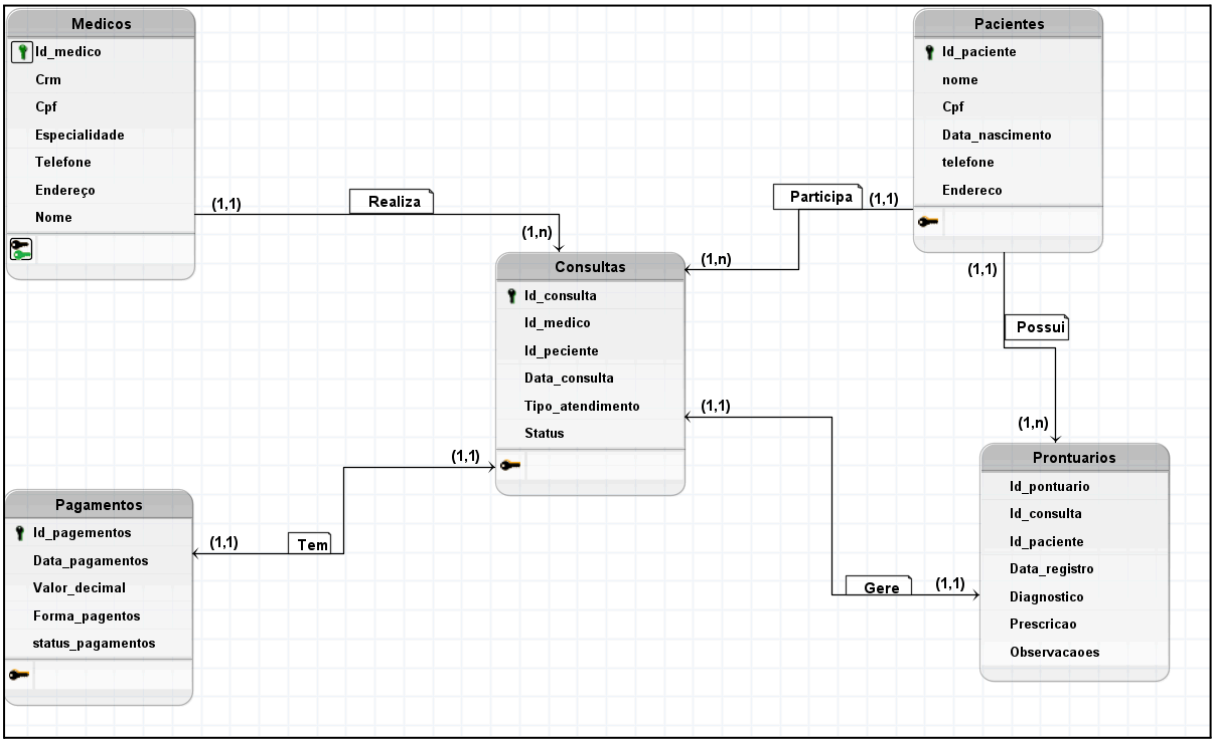
    - Caso de Uso 2 — Registrar Atendimento
        - Ator: Médico
        - Pré-condições:
            - A consulta deve existir e estar com status “realizada”.
        - Fluxo Principal:
            - O médico seleciona uma consulta.
            - O médico insere anotações, exames solicitados e receita.
            - O sistema registra o atendimento vinculado à consulta.
        - Pós condição:
            - - Histórico de atendimento atualizado.

2 Modelagem de Dados

- 2.1 Modelo Conceitual



- 2.2 Modelo Lógico



- 2.3 Modelo Físico (Script SQL)

O script inicia com a criação do banco de dados e, em seguida, define todas as tabelas do sistema (pacientes, medicos, consultas, prontuarios, atendimentos, exames_solicitados e pagamentos) por meio de instruções CREATE TABLE. Cada tabela foi construída com seus respectivos tipos de dados, chaves primárias, chaves estrangeiras e restrições necessárias (NOT NULL, UNIQUE,

DEFAULT). As relações entre as tabelas seguem o Modelo Lógico e garantem integridade referencial, conectando consultas aos pacientes e médicos, atendimentos e prontuários às consultas, exames solicitados aos atendimentos e pagamentos às consultas.

Script referente: creates-tables.sql

- 2.4 Processo de Normalização

1° passo – Estado Inicial (Não Normalizado)

PACIENTE(nome, cpf, data_nascimento, telefone, endereco, email)
MEDICO(nome, crm, especialidade, telefone, email)
CONSULTA(data_consulta, horario, status, observacoes, paciente_nome, medico_nome, prontuario_descricao, atendimento_anotacoes, exames_solicitados, pagamento_valor, pagamento_status)
PRONTUARIO(descricao, data_registro, consulta_dados)
ATENDIMENTO(anotacoes, receita, exames_solicitados, consulta_info)
PAGAMENTO(valor, forma_pagamento, data_pagamento, status_pagamento, consulta_info)

Explicação: Esquema mais plano/não normalizado, onde há uma mistura das informações de pacientes, médicos, consultas, prontuário, atendimento e pagamento, aparecendo juntas em um único registro. Isso gera redundância, inconsistência, repetições de grupos e atributos não atômicos.

2° passo – Separação inicial das entidades + Identificação de PKs

PACIENTE(**paciente_id**, nome, cpf, data_nascimento, telefone, endereco, email)
MEDICO(**medico_id**, nome, crm, especialidade, telefone, email)
CONSULTA(**cod_consulta**, data_consulta, horario, status, observacoes)
PRONTUARIO(**id_prontuario**, descricao, data_registro)
ATENDIMENTO**(id_atendimento**, anotacoes, receita, data_registro)
PAGAMENTO(**id_pagamento**, valor, forma_pagamento, status_pagamento, data_pagamento)

Explicação: Criação de tabelas independentes para cada conceito do sistema e definição das chaves primárias para identificação única. Agora cada grupo lógico vira uma tabela, há a atribuição das chaves primárias (FK) e sem apresentação de relações entre as entidades.

3° passo – Criação das relações (FKs) e tabelas associativas

PACIENTE**(paciente_id**, nome, cpf, data_nascimento, telefone, endereco, email)
MEDICO(**medico_id**, nome, crm, especialidade, telefone, email)

CONSULTA(**cod_consulta**, data_consulta, horario, status, observacoes, paciente_id, medico_id)

CONSULTA.paciente_id → PACIENTE.paciente_id

CONSULTA.medico_id → MEDICO.medico_id

PRONTUARIO(**id_prontuario**, consulta_id, descricao, data_registro)

PRONTUARIO.consulta_id → CONSULTA.cod_consulta

ATENDIMENTO(**id_atendimento**, consulta_id, anotacoes, receita, data_registro)

ATENDIMENTO.consulta_id → CONSULTA.cod_consulta

PAGAMENTO(**id_pagamento**, consulta_id, valor, forma_pagamento, status_pagamento, data_pagamento)

PAGAMENTO.consulta_id → CONSULTA.cod_consulta

Explicação: Conectar as tabelas por meios de chaves estrangeiras e criar tabelas intermediárias quando necessário. Com isso, ocorre a organização lógica do banco de dados, removendo totalmente a duplicação de dados e garantindo a integridade.

4° passo – Normalização Formal

- 1° Forma Normal (1FN)

PACIENTE(**paciente_id**, nome, cpf, data_nascimento, telefone, endereco, email)

MEDICO(medico_id, nome, crm, especialidade, telefone, email)

CONSULTA(**cod_consulta**, data_consulta, horario, status, observacoes, paciente_id, medico_id)

CONSULTA.paciente_id → PACIENTE.paciente_id

CONSULTA.medico_id → MEDICO.medico_id

PRONTUARIO(**id_prontuario**, consulta_id, descricao, data_registro)

PRONTUARIO.consulta_id → CONSULTA.cod_consulta

ATENDIMENTO(**id_atendimento**, consulta_id, anotacoes, receita, data_registro)

ATENDIMENTO.consulta_id → CONSULTA.cod_consulta

EXAME_SOLICITADO(**id_exame**, atendimento_id, codigo_exame, descricao_exame)

EXAME_SOLICITADO.atendimento_id → ATENDIMENTO.id_atendimento

PAGAMENTO(**id_pagamento**, consulta_id, valor, forma_pagamento, status_pagamento, data_pagamento)

PAGAMENTO.consulta_id → CONSULTA.cod_consulta

Explicação: A Primeira Forma Normal (1FN) foi aplicada garantindo que todas as tabelas possuíssem apenas atributos atômicos, sem valores múltiplos ou grupos repetidos. No modelo inicial, informações de paciente, médico, consulta, prontuário, atendimento e pagamento estavam misturadas em uma mesma estrutura, com campos que poderiam conter listas, como exames solicitados. Para atender à 1FN, cada conjunto de dados foi separado em sua própria tabela (PACIENTE, MEDICO, CONSULTA, PRONTUARIO, ATENDIMENTO, PAGAMENTO e EXAME_SOLICITADO), cada uma com sua chave primária. Isso eliminou repetições e assegurou que cada campo armazenasse apenas um valor indivisível, tornando o modelo adequado à 1ª Forma Normal.

- 2° Forma Normal (2FN)

Explicação: Sem aplicação. Todas as tabelas do sistema usam chaves primárias simples (ID numérico). Não existem chaves compostas como (paciente_id, consulta_id). Portanto, nenhum atributo depende parcialmente de uma PK composta, pois elas simplesmente não existem.

- 3° Forma Normal (3FN)

Explicação: Sem aplicação. As dependências transitivas foram eliminadas no passo 3: A especialidade do médico não está na tabela CONSULTA; O nome do paciente não está em ATENDIMENTO ou PAGAMENTO; Dados do médico não aparecem em PRONTUÁRIO; Nada depende de outro atributo que não seja a chave primária da tabela. Assim, todas as tabelas já estão adequadamente em 3FN.

3 Dicionário de Dados

Tabela: Pacientes						
Nome da Tabela	Nome da coluna	Tipo de Dado	Tamanho	Restrições	Chave (PK/FK)	Descrição/Propósito
paciente	paciente_id	SERIAL	N/A	PRIMARY KEY	PK	Identificador único do paciente.
paciente	nome	VARCHAR	150	NOT NULL	N/A	Nome completo do paciente.
paciente	cpf	CHAR	11	NOT NULL UNIQUE	N/A	CPF do paciente sem formatação.
paciente	data_nascime	DATE	N/A	NOT NULL	N/A	Data de nascimento do

	nto					cliente.
paciente	telefone	VARCHAR	20	NOT NULL	N/A	Telefone de contato.
paciente	endereco	VARCHAR	255	NOT NULL	N/A	Endereço completo do paciente.
paciente	email	VARCHAR	100	NOT NULL	N/A	E-mail do paciente.
paciente	data_criacao	DATE	N/A	NOT NULL DEFAULT CURRENT_DATE	N/A	Data em que o registro foi criado.

Tabela: Médicos						
Nome da Tabela	Nome da coluna	Tipo de Dado	Tamanho	Restrições	Chave (PK/FK)	Descrição/Propósito
medicos	medico_id	SERIAL	N/A	PRIMARY KEY	PK	Identificador único do médico.
medicos	nome	VARCHAR	150	NOT NULL	N/A	Nome completo do médico.
medicos	crm	VARCHAR	30	NOT NULL UNIQUE	N/A	Número de registro CRM.
medicos	especialidade	VARCHAR	100	NOT NULL	N/A	Especialidade médica.
medicos	telefone	VARCHAR	20	NOT NULL	N/A	Telefone de contato do médico.
medicos	email	VARCHAR	100	NOT NULL	N/A	E-mail do médico.
medicos	ativo	BOOLEAN	N/A	NOT NULL DEFAULT TRUE	N/A	Indica se o médico está ativo na clínica.
medicos	data_criacao	DATE	N/A	NOT NULL DEFAULT CURRENT_DATE	N/A	Data em que o médico foi cadastrado.

Tabela: Consultas						
Nome da Tabela	Nome da coluna	Tipo de Dado	Tamanho	Restrições	Chave (PK/FK)	Descrição/Propósito
consultas	consulta_id	SERIAL	N/A	PRIMARY KEY	PK	Identificador único da consulta.
consultas	paciente_id	INTEGER	N/A	NOT NULL REFERENCES pacientes(paciente_id)	FK	Paciente vinculado à consulta.
consultas	medico_id	INTEGER	N/A	NOT NULL REFERENCES medicos(medico_id)	FK	Médico responsável pela consulta.
consultas	data_consulta	DATE	N/A	NOT NULL	N/A	Data agendada da consulta.
consultas	horario	TIME	N/A	NOT NULL	N/A	Horário da consulta.

consultas	status	VARCHAR	20	NOT NULL DEFAULT 'agendada'	N/A	Status da consulta (agendada, confirmada…)
consultas	observacoes	TEXT	N/A	—	N/A	Observações gerais
consultas	data_agendamento	DATE	N/A	NOT NULL DEFAULT CURRENT_DATE	N/A	Data em que o agendamento foi registrado.

Tabela: Prontuários						
Nome da Tabela	Nome da coluna	Tipo de Dado	Tamanho	Restrições	Chave (PK/FK)	Descrição/Propósito
prontuarios	prontuario_id	SERIAL	N/A	PRIMARY KEY	PK	Identificador único do prontuário.
prontuarios	consulta_id	INTEGER	N/A	NOT NULL REFERENCES consultas(consulta_id)	FK	Consulta relacionada
prontuarios	descricao	TEXT	N/A	—	N/A	Conteúdo clínico do prontuário.
prontuarios	data_registro	DATE	N/A	NOT NULL DEFAULT CURRENT_DATE	N/A	Data de criação do prontuário.

Tabela: Atendimentos						
Nome da Tabela	Nome da coluna	Tipo de Dado	Tamanho	Restrições	Chave (PK/FK)	Descrição/Propósito
atendimento	atendimento_id	SERIAL	N/A	PRIMARY KEY	PK	Identificador único do atendimento.
atendimento	consulta_id	INTEGER	N/A	NOT NULL REFERENCES consultas(consulta_id)	FK	Consulta relacionada
atendimento	anotacoes	TEXT	N/A	N/A	N/A	Anotações feitas pelo médico.
atendimento	receita	TEXT	N/A	N/A	N/A	Receita médica emitida.
atendimento	exames_solicitados	TEXT	N/A	N/A	N/A	Exames solicitados durante o atendimento.
atendimento	data_registro	DATE	N/A	NOT NULL DEFAULT CURRENT_DATE	N/A	Data em que o atendimento foi resgistrado.

Tabela: Exames Solicitados						
Nome da Tabela	Nome da coluna	Tipo de Dado	Tamanho	Restrições	Chave (PK/FK)	Descrição/Propósito
exames_solic	exame_id	SERIAL	N/A	PRIMARY KEY	PK	Identificador único do

itados						exame solicitado.
exames_solicitados	atendimento_id	INTEGER	N/A	NOT NULL REFERENCES consultas(atendimento_id)	FK	Relaciona o exame ao atendimento.
exames_solicitados	codigo_exame	VARCHAR	50	NOT NULL	N/A	Código, sigla ou identificação do exame.
exames_solicitados	descricao_exame	VARCHAR	255	NOT NULL	N/A	Nome/descrição do exame solicitado.
exames_solicitados	data_registro	DATE	N/A	NOT NULL DEFAULT CURRENT_DATE	N/A	Data em que o exame foi registrado.

Tabela: Pagamentos						
Nome da Tabela	Nome da coluna	Tipo de Dado	Tamanho	Restrições	Chave (PK/FK)	Descrição/Propósito
pagamentos	pagamento_id	SERIAL	N/A	PRIMARY KEY	PK	Identificador único do pagamento.
pagamentos	consulta_id	INTEGER	N/A	NOT NULL REFERENCES consultas(consulta_id)	FK	Consulta paga.
pagamentos	valor	DECIMAL	10,2	NOT NULL	N/A	Valor cobrado pela consulta.
pagamentos	forma_pagamento	VARCHAR	50	NOT NULL	N/A	Forma de pagamento (PIX, Cartão…)
pagamentos	status_pagamento	VARCHAR	20	NOT NULL, DEFAULT ‘pendente’	N/A	Situação do pagamento.
pagamentos	data_pagamento	DATE	N/A	NOT NULL DEFAULT CURRENT_DATE	N/A	Data em que o pagamento foi realizado.

4 Implementação e Recursos Avançados

- 4.1 Operações Básicas (SQL)

Foram criados comandos INSERT, UPDATE, DELETE e SELECT, permitindo preencher o banco de dados, atualizar registros, remover informações incorretas e realizar consultas relevantes.Essas operações foram utilizadas também para validar o correto funcionamento das chaves estrangeiras, constraints e demais regras de integridade definidas no banco de dados.

Script referente: inserts-data.sql.

- 4.2 Views

As views foram criadas separadamente, no mesmo arquivo **creates-tables.sql**, porém **fora da parte de criação das tabelas**, de modo que não interferem no modelo físico. As views facilitam consultas comuns, eliminando a necessidade de escrever SQL complexo repetidamente.

As views implementadas foram:

- **vw_consultas_detalhadas** – Mostra informações consolidadas de consultas, pacientes e médicos.
- **vw_historico_atendimentos** – Exibe o histórico de atendimentos de cada paciente.
- **vw_pagamentos_recebidos** – Lista apenas os pagamentos já marcados como "Pago".
- **vw_agenda_medico** – Apresenta a agenda completa dos médicos.
- **vw_exames_por_atendimento** – Lista exames vinculados a um atendimento.
- **vw_exames_por_paciente** – Apresenta exames realizados por um paciente ao longo do tempo.

Script referente: creates-tables.sql.

- 4.3 Gatilhos (Triggers)

Foram desenvolvidos dois gatilhos que implementam regras de negócio automáticas:

Trigger 1 – trg_validar_horario_medico

- Quando é acionado: BEFORE INSERT ou UPDATE em *consultas*
- Função: Impede que um médico seja agendado para duas consultas no mesmo dia e horário
- Objetivo: Garantir integridade da agenda e evitar conflitos

Trigger 2 – trg_atualizar_status_consulta

- Quando é acionado: AFTER UPDATE em *pagamentos*
- Função: Atualiza automaticamente o status da consulta para "realizada" quando o pagamento é marcado como "Pago"
- Objetivo: Sincronizar informações financeiras e clínicas

Script referente: *triggers.sql.*

- 4.4 Funções

O arquivo **functions.sql** contém duas funções armazenadas projetadas para consultas avançadas no banco:

Função 1 – fn_total_pago_paciente(id_paciente)

- Calcula o total de valores pagos por um paciente
- Pode ser utilizada em relatórios financeiros

Função 2 – fn_consultas_realizadas_medico(id_medico)

- Retorna a quantidade de consultas concluídas por um médico
- Auxiliar em relatórios de produtividade

Script referente: functions.sql.

5 Plano de Testes

O plano de testes foi elaborado para validar o correto funcionamento do banco de dados do Sistema de Consultório/Clínica, garantindo que as operações de inserção, atualização, remoção, listagem e os recursos avançados (views, triggers e funções) atuem conforme o esperado. Os testes foram executados utilizando comandos SQL individuais, aplicados sobre o banco já criado pelos scripts *creates-tables.sql*, *inserts-data.sql*, *triggers.sql* e *functions.sql*.

- 5.1 Testes de Inserção
  - Cenário 1 – Inserir paciente válido

```
INSERT INTO pacientes (nome, cpf, data_nascimento, telefone, endereco, email)
VALUES ('João Silva', '12345678901', '1990-05-10', '84999998888', 'Rua A, 100', 'joao@email.com');
```

Resultado esperado:
Registro inserido com sucesso. Novo paciente aparece nas consultas seguintes.

  - Cenário 2 – Inserir médico válido

```
INSERT INTO medicos (nome, crm, especialidade, telefone, email)
VALUES ('Dra. Maria Andrade', 'CRM12345', 'Cardiologia', '84988887777', 'maria@clinic.com');
```

Resultado esperado:
Médico inserido normalmente.

  - Cenário 3 – Inserção inválida (CPF duplicado)

```
INSERT INTO pacientes (nome, cpf, data_nascimento, telefone, endereco, email)
VALUES ('Pedro Souza', '12345678901', '1985-07-20', '84977776666', 'Rua B, 200', 'pedro@email.com');
```

Resultado esperado:
Erro de violação de constraint UNIQUE em cpf.

- Cenário 4 – Inserir consulta válida

INSERT INTO consultas (paciente_id, medico_id, data_consulta, horario)
VALUES (1, 1, '2025-03-01', '09:00');

Resultado esperado:
Consulta registrada com status padrão "agendada".

- 5.2 Testes de Remoção
  - Cenário 1 – Remover paciente existente

DELETE FROM pacientes WHERE paciente_id = 1;

Resultado esperado:
Se houver consultas relacionadas, a remoção deve ser bloqueada (FK). Se não houver vínculos, a remoção será permitida.

- Cenário 2 – Remoção inexistente

DELETE FROM medicos WHERE medico_id = 999;

Resultado esperado:
Nenhuma linha afetada.

- 5.3 Testes de Listagem (SELECT)
  - Cenário 1 – Listar todos os pacientes

SELECT * FROM pacientes;

Resultado esperado:
Retorna todos os pacientes com seus atributos.

- Cenário 2 – Listar consultas por médico

SELECT * FROM consultas WHERE medico_id = 1;

Resultado esperado:
Aparecem apenas consultas do médico informado.

- Cenário 3 – Listar atendimentos com JOIN

SELECT p.nome, c.data_consulta, a.anotacoes

FROM atendimentos a

JOIN consultas c ON a.consulta_id = c.consulta_id

JOIN pacientes p ON c.paciente_id = p.paciente_id;

Resultado esperado:

Tabela combinada com paciente + consulta + atendimento.

- 5.4 Testes de Atualização (UPDATE)
    - Cenário 1 – Atualizar status da consulta

UPDATE consultas

SET status = 'confirmada'

WHERE consulta_id = 1;

Resultado esperado:

Status alterado corretamente.

    - Cenário 2 – Atualizar pagamento

UPDATE pagamentos

SET status_pagamento = 'Pago'

WHERE pagamento_id = 1;

Resultado esperado:

O gatilho trg_atualizar_status_consulta deve alterar automaticamente o status da consulta para “realizada”.

- 5.5 Testes de Views
    - Cenário 1 – Testar view vw_consultas_detalhadas

SELECT * FROM vw_consultas_detalhadas;

Resultado esperado:

Exibe consulta + nome do paciente + nome do médico.

    - Cenário 2 – Testar vw_pagamentos_recebidos

SELECT * FROM vw_pagamentos_recebidos;

Resultado esperado:

Mostra apenas pagamentos com status “Pago”.

5.6 Testes de Gatilhos (Triggers)

- Trigger 1 – Impedir conflito de horário (trg_validar_horario_medico)

Teste: criar duas consultas no mesmo horário para o mesmo médico

```
INSERT INTO consultas (paciente_id, medico_id, data_consulta, horario)
VALUES (2, 1, '2025-03-01', '09:00');
```

Resultado esperado:

Gatilho bloqueia a operação com erro: “Médico já possui consulta neste horário”.

- Trigger 2 – Atualização automática de status da consulta

Passo 1 – Criar consulta e pagamento pendente

```
INSERT INTO pagamentos (consulta_id, valor, forma_pagamento)
VALUES (1, 250.00, 'PIX');
```

Passo 2 – Atualizar pagamento

```
UPDATE pagamentos
SET status_pagamento = 'Pago'
WHERE pagamento_id = 1;
```

Resultado esperado:

Consulta 1 deve ter status = 'realizada' automaticamente.

- 5.7 Testes de Funções
  - Função 1 – fn_total_pago_paciente

```
SELECT fn_total_pago_paciente(1);
```

Resultado esperado:

Retorna a soma de todos os pagamentos marcados como “Pago” do paciente informado.

- Função 2 – fn_consultas_realizadas_medico

```
SELECT fn_consultas_realizadas_medico(1);
```

Resultado esperado:

Quantidade total de consultas com status “realizada” executadas pelo médico.

6 Conclusão

O desenvolvimento do sistema de consultório/clínica permitiu uma aplicação de forma prática os principais conceitos de modelagem e implementação de Banco de Dados, desde a definição dos requisitos até a criação do modelo físico e validação por meio de testes. Ao longo do projeto, foi possível compreender a importância da normalização, da integridade referencial e do uso de recursos avançados como views, gatilhos e funções para garantir organização, segurança e uma eficiência no tratamento das informações. Entre os maiores desafios, destacam-se a definição precisa das relações entre as entidades, a prevenção de conflitos de agendamento e a criação de mecanismos automáticos que mantivessem a consistência dos dados. Apesar das dificuldades, o projeto proporcionou um aprendizado significativo, consolidando habilidades técnicas e reforçando a importância de um banco de dados bem estruturado para o funcionamento adequado de sistemas reais na área da saúde.