Sistema de Gestão Hospitalar Universitário

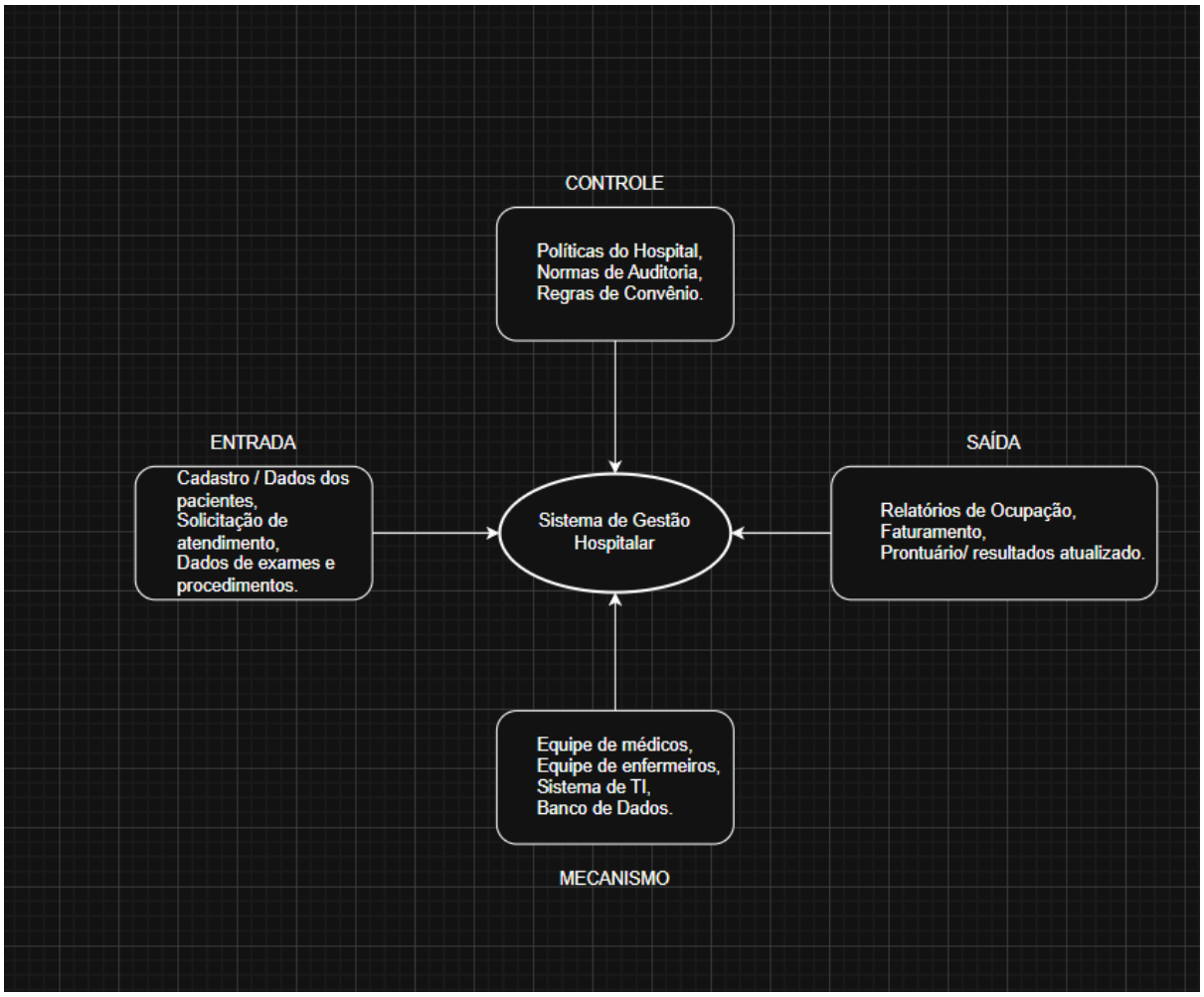
Aluno: Nicolas Melo Nascimento, Pedro Meneses Cruz, Vinicius Vieira da Silva, Pedro Minoru

Disciplina: Banco de Dados

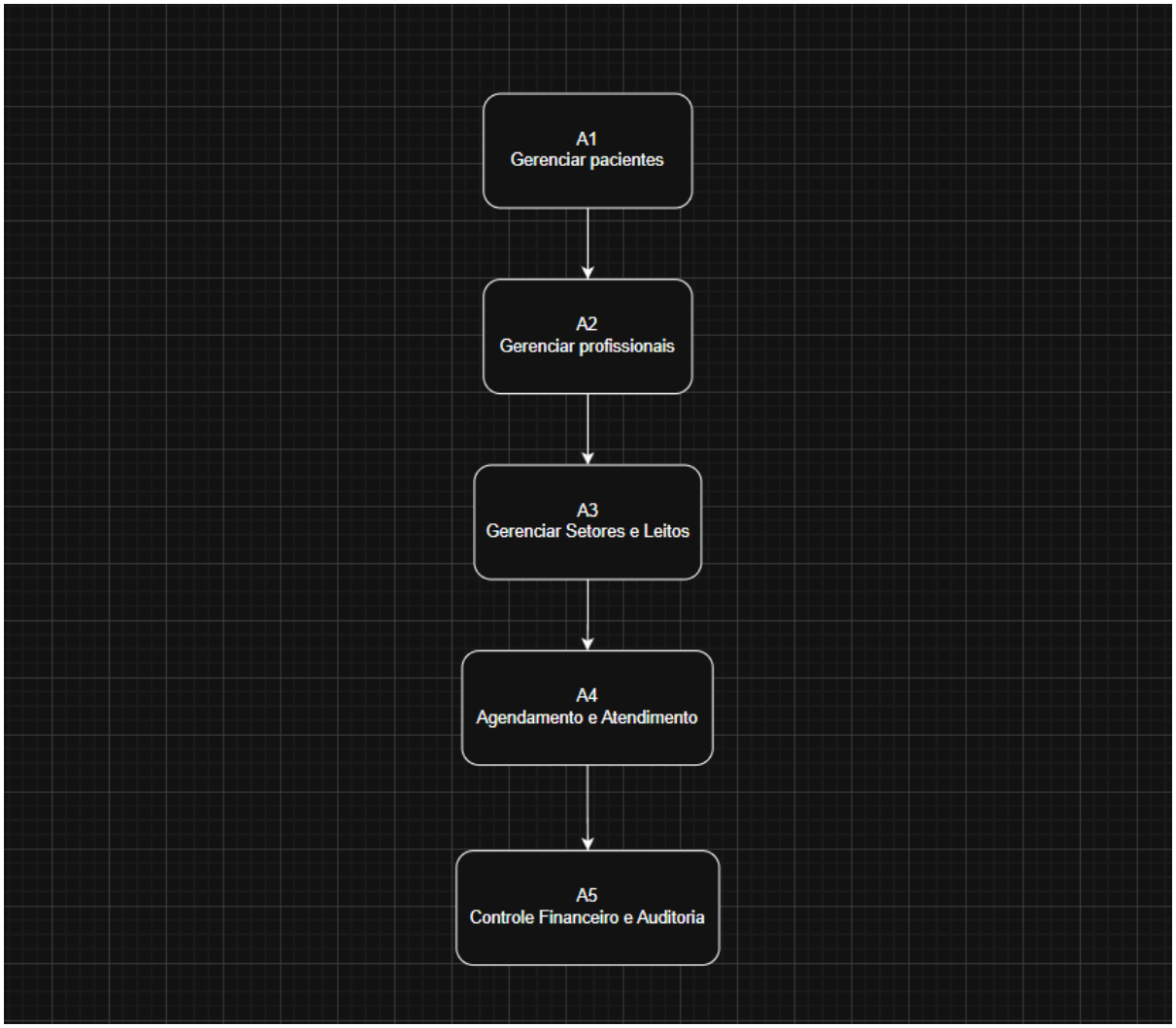
Novembro/2025

Etapa 1 - Modelagem IDEF0

A-0 (feito no draw.io)



A0 (decomposição do A-0) - Feito no draw.io



Modelagem do caso em cima das 5 visões da ISO 10746

1° Visão de Negócio (Objetivo do sistema)

O Hospital Universitário precisa de um sistema capaz de **gerenciar pacientes, profissionais de saúde, setores, leitos e atendimentos**, garantindo organização, segurança das informações e suporte aos processos clínicos, administrativos e financeiros.

O sistema também deve apoiar o ensino e pesquisa, mantendo um **histórico completo dos pacientes**, facilitando o acompanhamento de casos e a geração de relatórios gerenciais para tomada de decisão.

2° Visão Informacional (Quais dados o sistema manipula)

O sistema deve armazenar e manter integridade sobre os seguintes conjuntos de informações:

- **Pacientes:** dados pessoais, CPF, endereço, contatos, plano de saúde ou atendimento particular.
- **Profissionais:** médicos, enfermeiros e técnicos de laboratório, com especialidades e registro profissional.
- **Setores:** emergência, internação, laboratório, ambulatório e cirurgia.
- **Leitos:** identificação, setor, status (disponível/ocupado/liberado).
- **Atendimentos:** motivo da consulta, diagnóstico, exames solicitados, procedimentos realizados.
- **Agendamentos:** data, prioridade, profissional responsável.
- **Financeiro:** valores de procedimentos, convênio utilizado, faturamento.
- **Histórico Médico:** registros consolidados e protegidos por controle de acesso.
- **Auditoria:** alterações e acessos realizados no sistema.

3° Visão Funcional (O que o sistema faz)

O sistema realiza as seguintes funções principais:

- **A1 — Gerenciar Pacientes**
- **A2 — Gerenciar Profissionais**
- **A3 — Gerenciar Setores e Leitos**
- **A4 — Agendamento e Atendimento**
- **A5 — Controle Financeiro e Auditorial**

Essas funções representam o conjunto de processos que sustentam as operações do hospital, do atendimento ao faturamento.

4° Visão Organizacional (Quem usa o sistema e como)

O sistema será utilizado por diferentes perfis dentro do hospital:

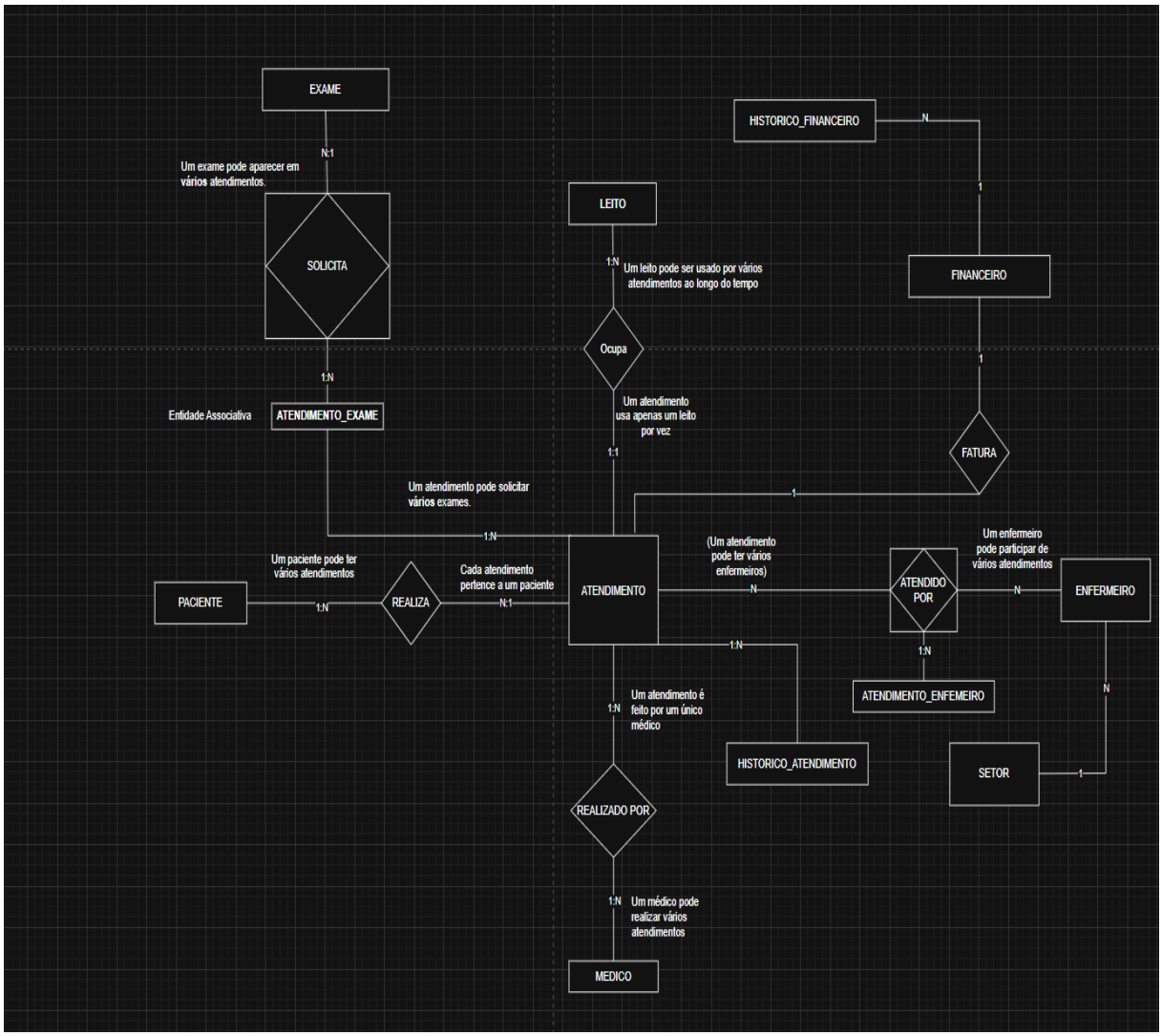
Perfil	Responsabillidades no sistema
Recepção	Cadastro de pacientes e agendamentos
Médicos	Registro de atendimentos, diagnósticos e solicitações de exames
Enfermeiros	Atualização de prontuário e status de leitos
Técnicos de Laboratório	Registro de resultados de exames
Setor Administrativo	Controle financeiro e faturamento
Gestores	Acesso a relatórios e indicadores gerenciais

Cada perfil possui **níveis diferentes de acesso**, garantindo segurança e confidencialidade.

5° Visão Computacional (Como o sistema é implementado tecnicamente)

- O sistema será apoiado por um Banco de Dados Relacional (Oracle).
- A aplicação deve permitir acesso multiusuário com autenticação.
- Deve existir controle de transações para garantir integridade dos dados clínicos e financeiros.
- Serão utilizados triggers e procedures para automatização e consistência (ex: mudança de status do leito).
- Views serão utilizadas para facilitar consultas de médicos e gestores.
- Todo acesso sensível será registrado no módulo de auditoria.

Etapa 2 - MER



- Introdução ao Modelo Entidade-Relacionamento (MER)

O Modelo Entidade-Relacionamento desenvolvido representa de forma conceitual o funcionamento do sistema de gestão de um Hospital Universitário. O objetivo é organizar e estruturar as informações dos principais processos do ambiente hospitalar, permitindo posteriormente a criação de um banco de dados relacional consistente, seguro e funcional.

O MER foi elaborado considerando os requisitos do sistema, garantindo integridade, ausência de redundância e correta representação dos relacionamentos entre as entidades.

Entidades Identificadas

As principais entidades do sistema são:

- **PACIENTE**

  Contém dados pessoais e administrativos do paciente.

- **MÉDICO**

  Registra informações profissionais de cada médico.

- **ENFERMEIRO**

  Armazena os dados dos enfermeiros responsáveis pelo apoio clínico.

- **SETOR**

Representa setores internos do hospital (ex.: emergência, laboratório, enfermaria).

- **LEITO**

Corresponde a cada leito físico do hospital, vinculado a um setor e com status (LIVRE/OCUPADO).

- **ATENDIMENTO**

  Centraliza o atendimento médico de um paciente, vinculando paciente, médico e leito.

- **EXAME**

  Representa exames laboratoriais, radiológicos ou clínicos disponíveis

- **FINANCEIRO**

  Guarda informações de cobrança referentes a um atendimento.

- **HISTORICO_ATENDIMENTO**

  Armazena alterações importantes realizadas no atendimento (diagnóstico ou observações).

- **HISTORICO_FINANCEIRO**

  Registra mudanças nos dados financeiros (valor, forma de pagamento)

Entidades Associativas

Alguns relacionamentos N:N necessitaram de entidades associativas:

- **ATENDIMENTO_EXAME**

  Relaciona quais exames foram solicitados em cada atendimento.
  Justificativa: um atendimento pode solicitar vários exames e um mesmo exame pode aparecer em vários atendimentos → **N:N**.

- **ATENDIMENTO_ENFERMEIRO**

  Registra os enfermeiros que participaram de cada atendimento.
  Justificativa: diversos enfermeiros podem atuar em um mesmo atendimento e um enfermeiro pode participar de vários atendimentos → **N:N**.

Relacionamentos e Cardinalidades

Paciente — Atendimento (1:N)

- Um paciente pode ter diversos atendimentos.
- Cada atendimento pertence a um único paciente.

Médico — Atendimento (1:N)

- Um médico pode realizar vários atendimentos.
- Um atendimento é realizado por apenas um médico.

Enfermeiro — Atendimento (N:N via ATENDIMENTO_ENFERMEIRO)

- Um atendimento pode envolver vários enfermeiros.

- Um enfermeiro pode participar de vários atendimentos.

Setor — Leito (1:N)

- Um setor pode possuir vários leitos.
- Cada leito pertence exclusivamente a um setor.

Leito — Atendimento (1:1 por atendimento)

- Um atendimento usa apenas um leito por vez.
- O mesmo leito pode ser usado em atendimentos diferentes ao longo do tempo (1:N histórico).

Atendimento — Exame (N:N via ATENDIMENTO_EXAME)

- Um atendimento pode solicitar vários exames.
- Um mesmo exame pode ser solicitado em diferentes atendimentos.

Atendimento — Financeiro (1:1)

- Cada atendimento possui exatamente um registro financeiro.
- Financeiro depende totalmente do atendimento.

Atendimento — Histórico_Atendimento (1:N)

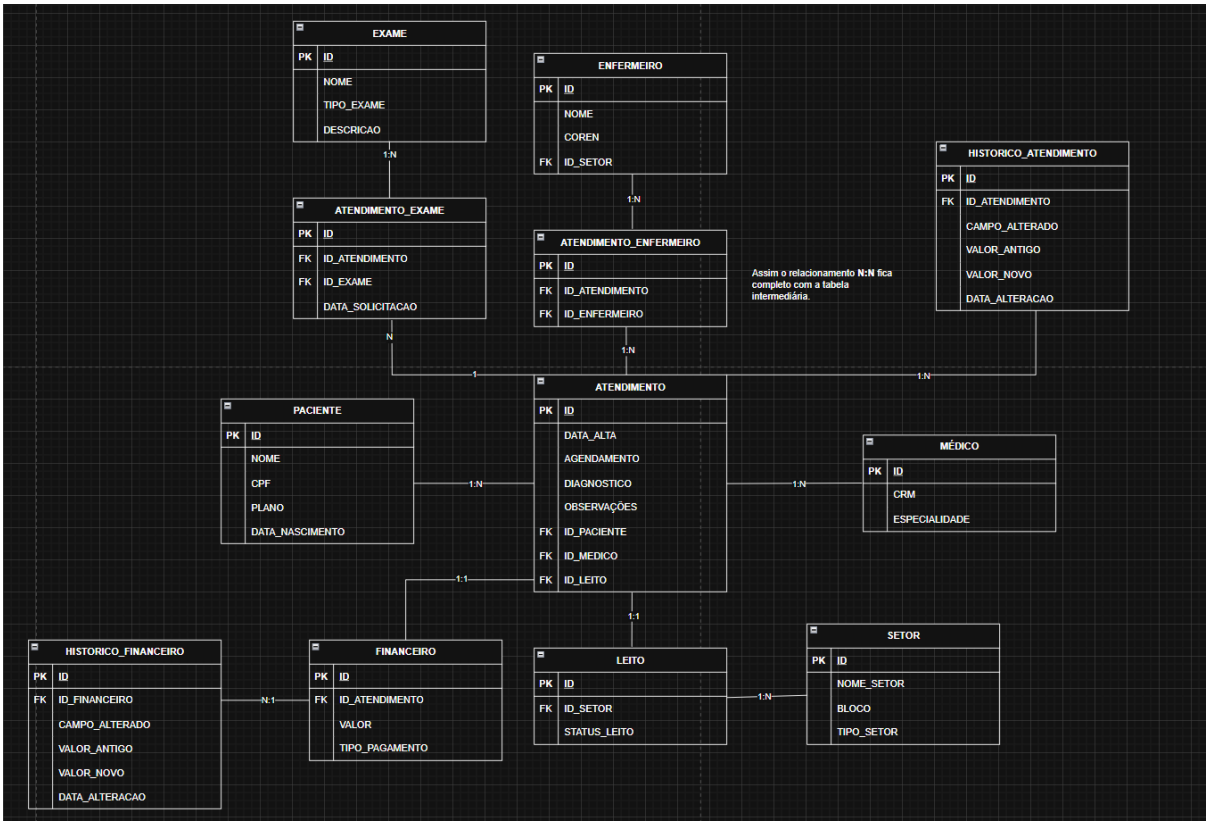
- Um atendimento pode ter várias alterações registradas em seu histórico.

Financeiro — Histórico_Financeiro (1:N)

Cada lançamento financeiro pode sofrer diversas alterações ao longo do tempo.

Justificativas das escolhas do MER

- Normalização: O modelo evita redundância, mantendo cada entidade com sua responsabilidade isolada.
- Integridade clínica: Regras importantes, como ocupação de leito e histórico de alterações, foram modeladas de forma a permitir validação futura via triggers.
- Flexibilidade: A utilização de tabelas associativas permite evoluir facilmente os módulos (ex.: adicionar equipes multidisciplinares, novos tipos de exames).
- Rastreamento: As entidades de histórico foram incluídas para garantir auditoria e rastreabilidade dos dados alterados.
- Modelo fiel à realidade hospitalar, representando pacientes, equipes médicas, exames, ocupação de leitos e movimentações financeiras.



ENTIDADES E SUAS CHAVES

PACIENTE

- **PK:** ID
- **Atributos:** Nome, CPF, Plano, Data_nascimento
- **Justificativa:** Armazena dados essenciais de identificação dos pacientes do hospital.

MÉDICO

- **PK:** ID
- **Atributos:** CRM, Especialidade
- **Justificativa:** Cada atendimento obrigatoriamente possui um médico responsável.

ENFERMEIRO

- **PK:** ID
- **Atributos:** Nome, COREN, ID_Setor (FK)
- **Justificativa:** Enfermeiros pertencem a um setor e podem atuar em vários atendimentos.

SETOR

- **PK:** ID
- **Atributos:** Nome_setor, Bloco, Tipo_setor
- **Justificativa:** Define áreas físicas do hospital, como emergência, laboratório, internação etc.

LEITO

- **PK:** ID
- **Atributos:** Status_leito, ID_Setor (FK)
- **Justificativa:** Cada leito pertence a um setor e pode ser ocupado por vários atendimentos ao longo do tempo.

EXAME

- **PK:** ID
- **Atributos:** Nome, Tipo_exame, Descricao
- **Justificativa:** Tabela com tipos de exames solicitáveis durante atendimentos.

ATENDIMENTO

- **PK:** ID
- **Atributos:** Data_alta, Agendamento, Diagnostico, Observacoes
- **FKs:** ID_paciente, ID_medico, ID_leito
- **Justificativa:** Representa o atendimento principal, reunindo paciente, médico, enfermeiros, exames, financeiro e leito.

FINANCEIRO

- **PK:** ID
- **Atributos:** Valor, Tipo_pagamento
- **FK:** ID_Atendimento
- **Justificativa:** Registra cobrança e dados financeiros de cada atendimento.

HISTORICO_ATENDIMENTO

- **PK:** ID
- **Atributos:** Campo_alterado, Valor_antigo, Valor_novo, Data_alteracao
- **FK:** ID_Atendimento
- **Justificativa:** Registro de auditoria para alterações feitas em um atendimento.

HISTORICO_FINANCEIRO

- **PK:** ID

- **Atributos:** Campo_alterado, Valor_antigo, Valor_novo, Data_alteracao
- **FK:** ID_financeiro
- **Justificativa:** Auditoria das alterações financeiras.

ATENDIMENTO_EXAME (Tabela Associativa)

- **PK:** ID
- **FKs:** ID_Atendimento, ID_Exame
- **Atributos:** Data_solicitacao
- **Justificativa:** Resolve o relacionamento N:N entre Atendimento e Exame.

ATENDIMENTO_ENFERMEIRO (Tabela Associativa)

- **PK:** ID
- **FKs:** ID_Atendimento, ID_Enfermeiro
- **Justificativa:** Resolve o relacionamento N:N entre Atendimento e Enfermeiro.

RELACIONAMENTOS E CARDINALIDADES

PACIENTE 1 — N ATENDIMENTO

- Um paciente pode ter vários atendimentos.
- Cada atendimento pertence a um único paciente.

MÉDICO 1 — N ATENDIMENTO

- Um médico atende vários pacientes.
- Cada atendimento é realizado por apenas um médico.

LEITO 1 — N ATENDIMENTO ao longo do tempo

- Um leito pode ser usado por vários atendimentos (em momentos diferentes).
- Cada atendimento utiliza apenas um leito.

ATENDIMENTO 1 — 1 FINANCEIRO

- Cada atendimento tem apenas um registro financeiro.
- Justificativa: cobrança é única por atendimento.

SETOR 1 — N LEITO

- Um setor possui vários leitos.
- Cada leito pertence a apenas um setor.

SETOR 1 — N ENFERMEIRO

- Um setor pode ter vários enfermeiros.
- Cada enfermeiro trabalha em um setor.

ATENDIMENTO N — N EXAME

- Resolvido por **ATENDIMENTO_EXAME**

ATENDIMENTO N — N ENFERMEIRO

- Resolvido por **ATENDIMENTO_ENFERMEIRO**.

ATENDIMENTO 1 — N HISTORICO_ATENDIMENTO

- Cada atualização do atendimento gera um registro no histórico.

FINANCEIRO 1 — N HISTORICO_FINANCEIRO

- Cada alteração financeira gera um registro de auditoria.

JUSTIFICATIVAS DO MODELO

- O modelo é completamente normalizado, evitando redundância.
- Usa tabelas associativas para resolver relacionamentos N:N.
- Todas as entidades possuem PK simples gerada automaticamente.
- FKs garantem integridade referencial.
- Histórico garante rastreabilidade e auditoria.
- Leito recebe controle de ocupação via trigger + FK.
- Financeiro vincula obrigatoriamente a um atendimento.

Justificativas Complementares do Modelo de Dados (DER)

O modelo de dados desenvolvido para o Sistema de Gestão Hospitalar foi projetado para garantir integridade, consistência e rastreabilidade das informações, atendendo às exigências funcionais e operacionais de um ambiente hospitalar. A seguir, são apresentadas as justificativas adicionais relacionadas à normalização, escolha das chaves primárias e implementação das regras de negócio por meio de triggers.

1. Justificativa da Normalização do Modelo

Durante a construção do modelo lógico e físico, foram aplicadas rigorosamente as três primeiras formas normais, assegurando a eliminação de redundâncias e anomalias:

• 1ª Forma Normal (1FN):
Todos os atributos são atômicos e não possuem valores multivalorados. Cada coluna armazena apenas um dado por registro, garantindo estrutura consistente e clara para consultas e manipulação.

• 2ª Forma Normal (2FN):
Todas as tabelas possuem chaves primárias simples, evitando dependências parciais. Não há tabelas com chaves compostas e, portanto, nenhum atributo não-chave depende de parte da chave primária.

• 3ª Forma Normal (3FN):
Nenhum atributo não-chave depende de outro atributo não-chave. Informações como setor, especialidade, tipo de exame, status de leito e outras foram isoladas em entidades específicas, reduzindo duplicação e facilitando manutenção.

Esse processo de normalização garante que o modelo seja eficiente, evite inconsistências e facilite futuras expansões.

2. Justificativa do Uso de Chaves Primárias Auto-Geradas (IDENTITY)

Todas as entidades do modelo utilizam chaves primárias numéricas geradas automaticamente por meio de sequência interna (IDENTITY). Essa escolha foi realizada por diversos motivos:

- Evita conflitos de valores e elimina necessidade de gerenciar manualmente PKs.
- Independe de atributos externos como CPF, CRM ou COREN, que podem variar, ser alterados ou gerar problemas legais ao serem usados como chave.
- Facilita relacionamentos e criação de FKs entre tabelas.
- Garante maior desempenho em operações de busca e junção (joins).

- Permite padronização em todas as tabelas do esquema.

Essa abordagem melhora a integridade e a escalabilidade do sistema como um todo.

3. Justificativa das Triggers e Regras de Negócio Implementadas

Para garantir integridade lógica e regras específicas do ambiente hospitalar, foram implementadas diversas triggers. Cada uma tem papel fundamental no funcionamento correto do sistema:

• Controle de Leitos

- **TRG_OCUPAR_LEITO:**
  Após um atendimento, atualiza automaticamente o status do leito para "OCUPADO", garantindo que ele não seja erroneamente marcado como livre.
- **TRG_IMPEDIR_LEITO_OCUPADO:**
  Impede que um atendimento utilize um leito que já está ocupado, preservando a integridade da lógica de internação.

• Controle de Datas e Agendamentos

- **TRG_DATA_SOLICITACAO_VALIDA:**
  Impede o registro de solicitações de exames com data no futuro, evitando inconsistências operacionais.

• Sistema de Auditoria do Atendimento

- **TRG_HISTORICO_ATENDIMENTO:**
  Registra automaticamente qualquer alteração feita nos campos Diagnóstico ou Observações, salvando o valor antigo, o novo e a data da modificação.
  Isso garante rastreabilidade, elemento essencial em sistemas de saúde.

• Sistema de Auditoria Financeira

- **TRG_AUDITORIA_FINANCEIRO:**
  Registra alterações nos valores cobrados e nas formas de pagamento, garantindo transparência financeira e permitindo auditoria completa.

• Preservação de Relacionamentos

- **TRG_IMPEDIR_DELETE_PACIENTE:**
  Impede a exclusão de pacientes que possuem atendimentos registrados, evitando perda de histórico, dados críticos e possíveis problemas legais.

Essas triggers reforçam a integridade referencial, automatizam regras de negócio e tornam o sistema mais seguro e confiável.

Conclusão Final

As justificativas apresentadas complementam o diagrama ER (DER), demonstrando que o modelo é **robusto, normalizado, auditável e alinhado às regras práticas de um hospital**. As decisões técnicas tomadas garantem consistência, escalabilidade e confiabilidade para o sistema de banco de dados.

Script SQL completo

-- TABELA: SETOR -- Armazena os setores do hospital (ex.: Emergência, UTI, Pediatria)

CREATE TABLE SETOR (ID_SETOR NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY, -- Identificador único do setor NOME_SETOR VARCHAR2(200) NOT NULL, -- Nome do setor BLOCO VARCHAR2(200), -- Bloco físico do hospital onde o setor está localizado TIPO_SETOR VARCHAR2(200) -- Classificação do setor (Atendimento, Internação etc.));

-- TABELA: PACIENTE -- Contém os dados cadastrais dos pacientes do hospital

CREATE TABLE PACIENTE (ID_PACIENTE NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY, -- Identificador do paciente NOME VARCHAR2(200) NOT NULL, -- Nome completo do paciente CPF NUMBER UNIQUE NOT NULL, -- CPF único para evitar duplicidades PLANO VARCHAR2(200) -- Plano de saúde);

-- TABELA: MEDICO -- Dados dos médicos que atendem no hospital

CREATE TABLE MEDICO (ID_MEDICO NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY, -- Identificador do médico CRM NUMBER UNIQUE NOT NULL, -- Número do CRM (único) ESPECIALIDADE VARCHAR2(200) -- Especialidade médica (ex.: Cardiologia, Clínico Geral));

-- TABELA: ENFERMEIRO -- Registra as informações de cada enfermeiro e seu setor

CREATE TABLE ENFERMEIRO (ID_ENFERMEIRO NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY, -- Identificador do enfermeiro NOME VARCHAR2(200) NOT NULL, COREN NUMBER UNIQUE NOT NULL, ID_SETOR NUMBER, FOREIGN KEY (ID_SETOR) REFERENCES SETOR(ID_SETOR) -- Relacionamento com SETOR);

-- TABELA: LEITO -- Representa os leitos disponíveis dentro de um setor

CREATE TABLE LEITO (ID_LEITO NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY, -- Identificador do leito STATUS_LEITO VARCHAR2(200), -- Situação atual do leito (LIVRE ou OCUPADO) ID_SETOR NUMBER NOT NULL, -- Setor ao qual o leito pertence FOREIGN KEY (ID_SETOR) REFERENCES SETOR(ID_SETOR));

-- TABELA: EXAME -- Catálogo de exames realizados pelo hospital

CREATE TABLE EXAME (ID_EXAME NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY, -- Identificador do exame NOME VARCHAR2(200) NOT NULL, TIPO_EXAME VARCHAR2(200) NOT NULL, -- Tipo (Laboratório, Imagem, etc.) DESCRICAO VARCHAR2(200));

-- TABELA: ATENDIMENTO -- Registro principal de atendimentos realizados no hospital

CREATE TABLE ATENDIMENTO (ID_ATENDIMENTO NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY, AGENDAMENTO VARCHAR2(200), DIAGNOSTICO VARCHAR2(200), OBSERVACOES VARCHAR2(200), -- Observações complementares

```
ID_PACIENTE NUMBER NOT NULL, -- Paciente atendido
ID_MEDICO   NUMBER NOT NULL, -- Médico responsável
ID_LEITO    NUMBER, -- Leito utilizado (se houver)

FOREIGN KEY (ID_PACIENTE) REFERENCES PACIENTE(ID_PACIENTE),
FOREIGN KEY (ID_MEDICO) REFERENCES MEDICO(ID_MEDICO),
FOREIGN KEY (ID_LEITO) REFERENCES LEITO(ID_LEITO)
```

);

-- TABELA ASSOCIATIVA: ATENDIMENTO_EXAME -- Relaciona quais exames foram solicitados em cada atendimento -- Representa uma relação N:N entre ATENDIMENTO e EXAME

CREATE TABLE ATENDIMENTO_EXAME (ID_ATENDIMENTO_EXAME NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY, DATA_SOLICITACAO DATE NOT NULL,

```
ID_ATENDIMENTO NUMBER NOT NULL,
ID_EXAME       NUMBER NOT NULL,

FOREIGN KEY (ID_ATENDIMENTO) REFERENCES ATENDIMENTO(ID_ATENDIMENTO),
FOREIGN KEY (ID_EXAME) REFERENCES EXAME(ID_EXAME)
```

);

-- TABELA ASSOCIATIVA: ATENDIMENTO_ENFERMEIRO -- Registra quais enfermeiros participaram de determinado atendimento -- Relação N:N entre ATENDIMENTO e ENFERMEIRO

CREATE TABLE ATENDIMENTO_ENFERMEIRO (ID_ATENDIMENTO_ENFERMEIRO NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,

```
ID_ATENDIMENTO NUMBER NOT NULL,
ID_ENFERMEIRO  NUMBER NOT NULL,

FOREIGN KEY (ID_ATENDIMENTO) REFERENCES ATENDIMENTO(ID_ATENDIMENTO),
FOREIGN KEY (ID_ENFERMEIRO)  REFERENCES ENFERMEIRO(ID_ENFERMEIRO)
```

);

-- TABELA: FINANCEIRO -- Armazena informações de cobrança relacionadas ao atendimento

CREATE TABLE FINANCEIRO (ID_FINANCEIRO NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY, -- Identificador do registro financeiro ID_ATENDIMENTO NUMBER NOT NULL, VALOR NUMBER(10,2) NOT NULL, -- Valor cobrado TIPO_PAGAMENTO VARCHAR2(200), -- Forma de pagamento (PIX, Cartão, Convênio, etc.)

```
FOREIGN KEY (ID_ATENDIMENTO) REFERENCES ATENDIMENTO(ID_ATENDIMENTO)
```

);

-- TRIGGER: TRG_DATA_SOLICITACAO_VALIDA -- Impede que um exame seja registrado com data futura. -- Valida a integridade temporal dos registros de exames.

CREATE OR REPLACE TRIGGER TRG_DATA_SOLICITACAO_VALIDA BEFORE INSERT OR UPDATE ON ATENDIMENTO_EXAME FOR EACH ROW BEGIN IF :NEW.DATA_SOLICITACAO > SYSDATE THEN RAISE_APPLICATION_ERROR(-20001, 'DATA_SOLICITACAO Não pode ser no futuro.'); END IF; END; /

-- Inserindo setor

INSERT INTO SETOR (NOME_SETOR, BLOCO, TIPO_SETOR) VALUES ('Emergencia', 'A', 'Atendimento');

```
SELECT ID_SETOR FROM SETOR;

-- Inserindo paciente

INSERT INTO PACIENTE (NOME, CPF, PLANO) VALUES ('Joao da Silva', 12345678900,
'Unimed');

SELECT ID_PACIENTE FROM PACIENTE;

-- Inserindo médico

INSERT INTO MEDICO (CRM, ESPECIALIDADE) VALUES (12345, 'Clinico Geral');

SELECT ID_MEDICO FROM MEDICO;

-- Inserindo leito

INSERT INTO LEITO (STATUS_LEITO, ID_SETOR) VALUES ('LIVRE', 1);

SELECT ID_LEITO FROM LEITO;

-- Inserindo atendimento

INSERT INTO ATENDIMENTO ( AGENDAMENTO, DIAGNOSTICO, OBSERVACOES,
ID_PACIENTE, ID_MEDICO, ID_LEITO ) VALUES ( 'Consulta Geral', 'Dor de cabeca', 'Paciente
estavel', 1, 1, 1 );

SELECT ID_ATENDIMENTO FROM ATENDIMENTO;

-- Inserindo exame

INSERT INTO EXAME (NOME, TIPO_EXAME, DESCRICAO) VALUES ('Exame de Sangue',
'Laboratorio', 'Hemograma Completo');

SELECT ID_EXAME FROM EXAME;

INSERT INTO ATENDIMENTO_EXAME ( DATA_SOLICITACAO, ID_ATENDIMENTO, ID_EXAME )
VALUES ( TO_DATE('30/12/2099', 'DD/MM/YYYY'), -- DATA FUTURA PROPOSITAL 1, 1 );

INSERT INTO ATENDIMENTO_EXAME ( DATA_SOLICITACAO, ID_ATENDIMENTO, ID_EXAME )
VALUES ( SYSDATE, 1, 1 );

ALTER SESSION SET NLS_DATE_FORMAT = 'DD/MM/YYYY HH24:MI:SS';

SELECT ID_ATENDIMENTO_EXAME, DATA_SOLICITACAO FROM ATENDIMENTO_EXAME;
```

-- TRIGGER: TRG_OCUPAR_LEITO -- Atualiza automaticamente o status do leito para OCUPADO -- sempre que um atendimento for registrado com ID_LEITO.

CREATE OR REPLACE TRIGGER TRG_OCUPAR_LEITO AFTER INSERT ON ATENDIMENTO FOR EACH ROW BEGIN IF :NEW.ID_LEITO IS NOT NULL THEN UPDATE LEITO SET STATUS_LEITO = 'OCUPADO' WHERE ID_LEITO = :NEW.ID_LEITO; END IF; END; /

INSERT INTO ATENDIMENTO (AGENDAMENTO, DIAGNOSTICO, OBSERVACOES, ID_PACIENTE, ID_MEDICO, ID_LEITO) VALUES ('Retorno', 'Dor', 'Teste', 1, 1, 1);

SELECT STATUS_LEITO FROM LEITO WHERE ID_LEITO = 1;

INSERT INTO ATENDIMENTO (AGENDAMENTO, DIAGNOSTICO, OBSERVACOES, ID_PACIENTE, ID_MEDICO, ID_LEITO) VALUES ('Primeiro Teste', 'Ok', 'Ok', 1, 1, 1);

INSERT INTO ATENDIMENTO (AGENDAMENTO, DIAGNOSTICO, OBSERVACOES, ID_PACIENTE, ID_MEDICO, ID_LEITO) VALUES ('Segundo Teste', 'Ok', 'Ok', 1, 1, 1);

-- TRIGGER: TRG_IMPEDIR_DELETE_PACIENTE -- Impede exclusão de um paciente se houver atendimentos -- vinculados a ele.

CREATE OR REPLACE TRIGGER TRG_IMPEDIR_DELETE_PACIENTE BEFORE DELETE ON PACIENTE FOR EACH ROW DECLARE v_contador NUMBER; BEGIN

```
-- Verificar se existe atendimento associado ao paciente
SELECT COUNT(*)
INTO v_contador
FROM ATENDIMENTO
WHERE ID_PACIENTE = :OLD.ID_PACIENTE;

-- Se houver atendimentos, impedir a exclusão
IF v_contador > 0 THEN
RAISE_APPLICATION_ERROR(
    -20003,
    'Não é possível excluir este PACIENTE: existem atendimentos
vinculados.'
);
```

```
END IF;
```

END; /

DELETE FROM PACIENTE WHERE ID_PACIENTE = 1;

ALTER TABLE ATENDIMENTO ADD DATA_ALTA DATE;

-- TRIGGER: TRG_REGISTRAR_DATA_ALTA -- Registra automaticamente a DATA_ALTA quando a observação -- do atendimento contiver a palavra "ALTA".

CREATE OR REPLACE TRIGGER TRG_REGISTRAR_DATA_ALTA BEFORE UPDATE ON ATENDIMENTO FOR EACH ROW BEGIN -- Se houver alteração indicando ALTA IF UPPER(:NEW.OBSERVACOES) LIKE '%ALTA%' AND :OLD.DATA_ALTA IS NULL THEN

```
-- Preencher automaticamente a data da alta
:NEW.DATA_ALTA := SYSDATE;
END IF;
```

END; / -- Teste SELECT ID_ATENDIMENTO, DATA_ALTA FROM ATENDIMENTO WHERE ID_ATENDIMENTO = 1;

UPDATE ATENDIMENTO SET OBSERVACOES = 'Paciente recebeu alta' WHERE ID_ATENDIMENTO = 1;

-- TRIGGER: TRG_IMPEDIR_LEITO_OCUPADO -- Impede que um atendimento utilize ou altere para um -- leito que já esteja ocupado.

CREATE OR REPLACE TRIGGER TRG_IMPEDIR_LEITO_OCUPADO BEFORE INSERT OR UPDATE OF ID_LEITO ON ATENDIMENTO FOR EACH ROW DECLARE v_status VARCHAR2(200); BEGIN -- Só verifica quando o ID_LEITO for alterado IF :NEW.ID_LEITO IS NOT NULL AND :NEW.ID_LEITO != :OLD.ID_LEITO THEN

```
   -- Buscar status atual do leito
    SELECT STATUS_LEITO
```

```
    INTO v_status
    FROM LEITO
    WHERE ID_LEITO = :NEW.ID_LEITO;

    -- Se estiver ocupado, impedir troca
    IF v_status = 'OCUPADO' THEN
        RAISE_APPLICATION_ERROR(
            -20002,
            'Este leito já está OCUPADO.'
        );
    END IF;

END IF;


END; /
```

UPDATE ATENDIMENTO SET OBSERVACOES = 'Paciente recebeu alta' WHERE ID_ATENDIMENTO = 1;

SELECT ID_ATENDIMENTO, DATA_ALTA FROM ATENDIMENTO WHERE ID_ATENDIMENTO = 1;

CREATE TABLE HISTORICO_ATENDIMENTO (-- Guarda mudanças feitas no atendimento. ID_HISTORICO NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY, ID_ATENDIMENTO NUMBER NOT NULL, CAMPO_ALTERADO VARCHAR2(200), VALOR_ANTIGO VARCHAR2(200), VALOR_NOVO VARCHAR2(200), DATA_ALTERACAO DATE DEFAULT SYSDATE,

```
FOREIGN KEY (ID_ATENDIMENTO) REFERENCES ATENDIMENTO(ID_ATENDIMENTO)


);
```

-- TRIGGER: TRG_HISTORICO_ATENDIMENTO -- Registra automaticamente as alterações feitas nos campos -- DIAGNOSTICO e OBSERVACOES do atendimento.

CREATE OR REPLACE TRIGGER TRG_HISTORICO_ATENDIMENTO AFTER UPDATE ON ATENDIMENTO FOR EACH ROW BEGIN -- Se o diagnóstico mudou IF :OLD.DIAGNOSTICO != :NEW.DIAGNOSTICO THEN INSERT INTO HISTORICO_ATENDIMENTO (ID_ATENDIMENTO,

CAMPO_ALTERADO, VALOR_ANTIGO, VALOR_NOVO) VALUES (:OLD.ID_ATENDIMENTO, 'DIAGNOSTICO', :OLD.DIAGNOSTICO, :NEW.DIAGNOSTICO); END IF;

```
-- Se as observações mudaram
IF :OLD.OBSERVACOES != :NEW.OBSERVACOES THEN
    INSERT INTO HISTORICO_ATENDIMENTO (
        ID_ATENDIMENTO, CAMPO_ALTERADO, VALOR_ANTIGO, VALOR_NOVO
    ) VALUES (
        :OLD.ID_ATENDIMENTO,
        'OBSERVACOES',
        :OLD.OBSERVACOES,
        :NEW.OBSERVACOES
    );
END IF;
```

END; /

UPDATE ATENDIMENTO SET DIAGNOSTICO = 'Gripe forte' WHERE ID_ATENDIMENTO = 1;

SELECT ID_ATENDIMENTO, CAMPO_ALTERADO, VALOR_ANTIGO, VALOR_NOVO, DATA_ALTERACAO FROM HISTORICO_ATENDIMENTO;

CREATE TABLE HISTORICO_FINANCEIRO (ID_HISTORICO NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY, ID_FINANCEIRO NUMBER NOT NULL, CAMPO_ALTERADO VARCHAR2(200), VALOR_ANTIGO VARCHAR2(200), VALOR_NOVO VARCHAR2(200), DATA_ALTERACAO DATE DEFAULT SYSDATE,

```
FOREIGN KEY (ID_FINANCEIRO) REFERENCES FINANCEIRO(ID_FINANCEIRO)
```

);

-- TRIGGER: TRG_AUDITORIA_FINANCEIRO -- Objetivo: Registrar alterações nos campos VALOR e -- TIPO_PAGAMENTO da tabela FINANCEIRO, criando um histórico -- de auditoria para rastreabilidade.

CREATE OR REPLACE TRIGGER TRG_AUDITORIA_FINANCEIRO AFTER UPDATE ON FINANCEIRO FOR EACH ROW BEGIN -- Caso o VALOR seja alterado IF :OLD.VALOR != :NEW.VALOR THEN INSERT INTO HISTORICO_FINANCEIRO (ID_FINANCEIRO,

CAMPO_ALTERADO, VALOR_ANTIGO, VALOR_NOVO) VALUES (:OLD.ID_FINANCEIRO, 'VALOR', TO_CHAR(:OLD.VALOR), TO_CHAR(:NEW.VALOR)); END IF;

```
-- Caso o TIPO_PAGAMENTO seja alterado
IF :OLD.TIPO_PAGAMENTO != :NEW.TIPO_PAGAMENTO THEN
    INSERT INTO HISTORICO_FINANCEIRO (
        ID_FINANCEIRO, CAMPO_ALTERADO, VALOR_ANTIGO, VALOR_NOVO
    ) VALUES (
        :OLD.ID_FINANCEIRO,
        'TIPO_PAGAMENTO',
        :OLD.TIPO_PAGAMENTO,
        :NEW.TIPO_PAGAMENTO
    );
END IF;
```

END; /

-- Inserindo registro financeiro inicial para testes INSERT INTO FINANCEIRO (ID_ATENDIMENTO, VALOR, TIPO_PAGAMENTO) VALUES (1, 200.00, 'PIX');

-- Consultando o ID gerado SELECT ID_FINANCEIRO FROM FINANCEIRO;

-- PROCEDURE: CADASTRAR_PACIENTE -- Objetivo: Cadastrar um novo paciente garantindo que o CPF -- não esteja duplicado. Caso exista, é lançada uma exceção.

CREATE OR REPLACE PROCEDURE CADASTRAR_PACIENTE (p_nome IN VARCHAR2, p_cpf IN NUMBER, p_plano IN VARCHAR2) AS v_count NUMBER; BEGIN -- Verificando duplicidade do CPF SELECT COUNT(*) INTO v_count FROM PACIENTE WHERE CPF = p_cpf;

```
IF v_count > 0 THEN
    RAISE_APPLICATION_ERROR(-20010, 'CPF já cadastrado.');
END IF;
```

-- Inserção do novo paciente INSERT INTO PACIENTE (NOME, CPF, PLANO) VALUES (p_nome, p_cpf, p_plano);

COMMIT;

EXCEPTION WHEN OTHERS THEN -- Em caso de erro, desfaz a transação ROLLBACK; RAISE; END; /

EXEC CADASTRAR_PACIENTE('Maria de Souza', 55555555555, 'Unimed');

-- PROCEDURE: AGENDAR_ATENDIMENTO -- Objetivo: Registrar um atendimento garantindo validações: -- 1) Paciente existe -- 2) Médico existe -- 3) Leito (se informado) está livre -- 4) Insere o atendimento -- Contém controle de transação (COMMIT/ROLLBACK)

CREATE OR REPLACE PROCEDURE AGENDAR_ATENDIMENTO (p_agendamento IN VARCHAR2, p_diagnostico IN VARCHAR2, p_observacoes IN VARCHAR2, p_id_paciente IN NUMBER, p_id_medico IN NUMBER, p_id_leito IN NUMBER DEFAULT NULL) AS v_exists NUMBER; v_status VARCHAR2(200); BEGIN -- Validar paciente SELECT COUNT(*) INTO v_exists FROM PACIENTE WHERE ID_PACIENTE = p_id_paciente; IF v_exists = 0 THEN RAISE_APPLICATION_ERROR(-20011, 'Paciente inexistente.'); END IF;

```
-- Validar médico
SELECT COUNT(*) INTO v_exists FROM MEDICO WHERE ID_MEDICO =
p_id_medico;
IF v_exists = 0 THEN
    RAISE_APPLICATION_ERROR(-20012, 'Médico inexistente.');
END IF;

-- Validar leito
IF p_id_leito IS NOT NULL THEN
    SELECT STATUS_LEITO INTO v_status
    FROM LEITO
    WHERE ID_LEITO = p_id_leito
    FOR UPDATE; -- Garante que ninguém edite o leito ao mesmo tempo

    IF v_status = 'OCUPADO' THEN
        RAISE_APPLICATION_ERROR(-20013, 'Leito já está ocupado.');
    END IF;
END IF;
```

-- Inserção do atendimento INSERT INTO ATENDIMENTO (AGENDAMENTO, DIAGNOSTICO, OBSERVACOES, ID_PACIENTE, ID_MEDICO, ID_LEITO) VALUES (p_agendamento, p_diagnostico, p_observacoes, p_id_paciente, p_id_medico, p_id_leito);

```
COMMIT;
```

EXCEPTION WHEN OTHERS THEN -- Desfaz a transação em caso de erro ROLLBACK; RAISE; END; /

BEGIN CADASTRAR_PACIENTE('Maria de Souza', 55555555555, 'Unimed'); END; /

-- VIEW: VW_RESUMO_ATENDIMENTO -- Objetivo: Exibir um resumo completo dos atendimentos, -- agrupando informações de várias tabelas em uma única visão.

CREATE OR REPLACE VIEW VW_RESUMO_ATENDIMENTO AS SELECT A.ID_ATENDIMENTO, P.NOME AS NOME_PACIENTE, -- Paciente relacionado M.CRM AS CRM_MEDICO, -- CRM do médico responsável M.ESPECIALIDADE, A.DIAGNOSTICO, A.OBSERVACOES, A.AGENDAMENTO, A.DATA_ALTA, -- Data de alta (se houver) L.ID_LEITO, L.STATUS_LEITO FROM ATENDIMENTO A JOIN PACIENTE P ON A.ID_PACIENTE = P.ID_PACIENTE JOIN MEDICO M ON A.ID_MEDICO = M.ID_MEDICO LEFT JOIN LEITO L ON A.ID_LEITO = L.ID_LEITO;

SELECT ID_ATENDIMENTO, NOME_PACIENTE, CRM_MEDICO FROM VW_RESUMO_ATENDIMENTO;

-- VIEW: VW_OCUPACAO_LEITOS -- Objetivo: Mostrar o status dos leitos e seus respectivos -- setores, facilitando o controle da ocupação hospitalar. CREATE OR REPLACE VIEW VW_OCUPACAO_LEITOS AS SELECT S.NOME_SETOR, S.BLOCO, L.ID_LEITO, L.STATUS_LEITO FROM LEITO L JOIN SETOR S ON L.ID_SETOR = S.ID_SETOR;

SELECT NOME_SETOR, BLOCO, ID_LEITO, STATUS_LEITO FROM VW_OCUPACAO_LEITOS;

-- VIEW: VW_HISTORICO_ATENDIMENTO -- Objetivo: Mostrar os registros de auditoria de alterações -- ocorridas nos atendimentos.

CREATE OR REPLACE VIEW VW_HISTORICO_ATENDIMENTO AS SELECT H.ID_HISTORICO, H.ID_ATENDIMENTO, H.CAMPO_ALTERADO, H.VALOR_ANTIGO, H.VALOR_NOVO, H.DATA_ALTERACAO FROM HISTORICO_ATENDIMENTO H;

SELECT ID_HISTORICO, ID_ATENDIMENTO, CAMPO_ALTERADO, VALOR_ANTIGO, VALOR_NOVO, DATA_ALTERACAO FROM VW_HISTORICO_ATENDIMENTO WHERE ID_ATENDIMENTO = 1;

-- VIEW: VW_FINANCEIRO_RESUMO -- Objetivo: Mostrar os registros financeiros junto com -- informações do atendimento, paciente e setor. CREATE OR REPLACE VIEW VW_FINANCEIRO_RESUMO AS SELECT F.ID_FINANCEIRO, F.ID_ATENDIMENTO, P.NOME AS NOME_PACIENTE, M.CRM AS CRM_MEDICO, F.VALOR, F.TIPO_PAGAMENTO, A.DIAGNOSTICO, L.ID_LEITO, S.NOME_SETOR AS SETOR FROM FINANCEIRO F JOIN ATENDIMENTO A ON F.ID_ATENDIMENTO = A.ID_ATENDIMENTO JOIN PACIENTE P ON A.ID_PACIENTE = P.ID_PACIENTE JOIN MEDICO M ON A.ID_MEDICO = M.ID_MEDICO LEFT JOIN LEITO L ON A.ID_LEITO = L.ID_LEITO LEFT JOIN SETOR S ON L.ID_SETOR = S.ID_SETOR;

SELECT ID_ATENDIMENTO, NOME_PACIENTE, VALOR, TIPO_PAGAMENTO, SETOR FROM VW_FINANCEIRO_RESUMO;

-- Adicionando a data de nascimento ao cadastro ALTER TABLE PACIENTE ADD DATA_NASCIMENTO DATE;

-- Inserindo dado de teste UPDATE PACIENTE SET DATA_NASCIMENTO = TO_DATE('12/05/2000', 'DD/MM/YYYY') WHERE ID_PACIENTE = 1;

-- FUNCTION: FN_CALCULA_IDADE -- Objetivo: Calcular a idade de um paciente com base na -- data de nascimento. CREATE OR REPLACE FUNCTION FN_CALCULA_IDADE (p_id_paciente NUMBER) RETURN NUMBER AS v_data_nasc DATE; v_idade NUMBER; BEGIN -- Buscar a data de nascimento SELECT DATA_NASCIMENTO INTO v_data_nasc FROM PACIENTE WHERE ID_PACIENTE = p_id_paciente;

```
-- Calcular a idade
v_idade := TRUNC(MONTHS_BETWEEN(SYSDATE, v_data_nasc) / 12);

RETURN v_idade;
```

EXCEPTION WHEN NO_DATA_FOUND THEN RAISE_APPLICATION_ERROR(-20020, 'Paciente não encontrado.'); END; /

SELECT FN_CALCULA_IDADE(1) FROM dual;

-- FUNCTION: FN_TOTAL_EXAMES_ATENDIMENTO -- Objetivo: Somar o valor total dos exames solicitados -- para um atendimento específico. CREATE OR REPLACE FUNCTION FN_TOTAL_EXAMES_ATENDIMENTO (p_id_atendimento NUMBER) RETURN NUMBER AS v_total NUMBER := 0; BEGIN SELECT SUM(E.VALOR) INTO v_total FROM

ATENDIMENTO_EXAME AE JOIN EXAME E ON AE.ID_EXAME = E.ID_EXAME WHERE AE.ID_ATENDIMENTO = p_id_atendimento;

```
RETURN NVL(v_total, 0); -- Retorna 0 se for NULL


END; /
```

-- FUNCTION: FN_LEITO_LIVRE -- Objetivo: Verificar se um leito está disponível. CREATE OR REPLACE FUNCTION FN_LEITO_LIVRE (p_id_leito NUMBER) RETURN NUMBER AS v_status VARCHAR2(200); BEGIN SELECT STATUS_LEITO INTO v_status FROM LEITO WHERE ID_LEITO = p_id_leito;

```
IF v_status = 'LIVRE' THEN
    RETURN 1;
ELSE
    RETURN 0;
END IF;


EXCEPTION WHEN NO_DATA_FOUND THEN -- Leito inexistente também retorna 0 RETURN 0; END; /

-- PACKAGE: PKG_HOSPITAL (SPECIFICATION) -- Objetivo: Centralizar funções e procedimentos relacionados -- à gestão hospitalar. CREATE OR REPLACE PACKAGE PKG_HOSPITAL AS

-- Funções
FUNCTION FN_CALCULA_IDADE (p_id_paciente NUMBER) RETURN NUMBER;
FUNCTION FN_LEITO_LIVRE (p_id_leito NUMBER) RETURN NUMBER;

-- Procedures
PROCEDURE CADASTRAR_PACIENTE (
    p_nome  IN  VARCHAR2,
    p_cpf   IN  NUMBER,
    p_plano IN  VARCHAR2,
    p_data_nasc IN DATE
);

PROCEDURE AGENDAR_ATENDIMENTO (
    p_agendamento   IN VARCHAR2,
    p_diagnostico   IN VARCHAR2,
    p_observacoes   IN VARCHAR2,
    p_id_paciente   IN NUMBER,
```

```
    p_id_medico     IN NUMBER,
    p_id_leito      IN NUMBER DEFAULT NULL
);

PROCEDURE LIBERAR_LEITO (p_id_leito NUMBER);

PROCEDURE DAR_ALTA (p_id_atendimento NUMBER);


END PKG_HOSPITAL; /
```

-- IMPLEMENTAÇÃO DO PACKAGE BODY PKG_HOSPITAL -- Contém funções e procedures utilizadas no sistema hospitalar CREATE OR REPLACE PACKAGE BODY PKG_HOSPITAL AS

```
-- FUNÇÃO 1: idade do paciente

FUNCTION FN_CALCULA_IDADE (p_id_paciente NUMBER)
RETURN NUMBER AS
    v_data DATE;
BEGIN
    SELECT DATA_NASCIMENTO INTO v_data
    FROM PACIENTE
    WHERE ID_PACIENTE = p_id_paciente;

    -- Calcula idade em anos
    RETURN TRUNC(MONTHS_BETWEEN(SYSDATE, v_data) / 12);
END FN_CALCULA_IDADE;

-- FUNÇÃO 2: leito livre

FUNCTION FN_LEITO_LIVRE (p_id_leito NUMBER)
RETURN NUMBER AS
    v_status VARCHAR2(200);
BEGIN
    SELECT STATUS_LEITO INTO v_status
    FROM LEITO
    WHERE ID_LEITO = p_id_leito;

    IF v_status = 'LIVRE' THEN
        RETURN 1;
    ELSE
        RETURN 0;
    END IF;
```

```
END FN_LEITO_LIVRE;

-- PROCEDURE: cadastrar paciente
PROCEDURE CADASTRAR_PACIENTE (
    p_nome  IN  VARCHAR2,
    p_cpf   IN  NUMBER,
    p_plano IN  VARCHAR2,
    p_data_nasc IN DATE
) AS
    v_count NUMBER;
BEGIN
    -- Verifica se já existe CPF cadastrado
    SELECT COUNT(*) INTO v_count FROM PACIENTE WHERE CPF = p_cpf;
    IF v_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20010, 'CPF já cadastrado.');
    END IF;

    -- Insere novo paciente no sistema
    INSERT INTO PACIENTE (NOME, CPF, PLANO, DATA_NASCIMENTO)
    VALUES (p_nome, p_cpf, p_plano, p_data_nasc);

    COMMIT;
END CADASTRAR_PACIENTE;

-- PROCEDURE: agendar atendimento
-- Cria um novo atendimento e verifica se o leito está disponível
PROCEDURE AGENDAR_ATENDIMENTO (
    p_agendamento   IN VARCHAR2,
    p_diagnostico   IN VARCHAR2,
    p_observacoes   IN VARCHAR2,
    p_id_paciente   IN NUMBER,
    p_id_medico     IN NUMBER,
    p_id_leito      IN NUMBER DEFAULT NULL
) AS
    v_status VARCHAR2(200);
BEGIN
    -- Caso exista leito, verificar se está disponível
    IF p_id_leito IS NOT NULL THEN
        SELECT STATUS_LEITO INTO v_status
        FROM LEITO
        WHERE ID_LEITO = p_id_leito
        FOR UPDATE;
```

```
        IF v_status = 'OCUPADO' THEN
            RAISE_APPLICATION_ERROR(-20015, 'Leito ocupado.');
        END IF;
    END IF;

    INSERT INTO ATENDIMENTO (
        AGENDAMENTO, DIAGNOSTICO, OBSERVACOES,
        ID_PACIENTE, ID_MEDICO, ID_LEITO
    ) VALUES (
        p_agendamento, p_diagnostico, p_observacoes,
        p_id_paciente, p_id_medico, p_id_leito
    );

    COMMIT;
END AGENDAR_ATENDIMENTO;

-- PROCEDURE: LIBERAR_LEITO
-- Atualiza o status do leito para "LIVRE"-- PROCEDURE: liberar
leito
PROCEDURE LIBERAR_LEITO (p_id_leito NUMBER) AS
BEGIN
    UPDATE LEITO
    SET STATUS_LEITO = 'LIVRE'
    WHERE ID_LEITO = p_id_leito;

    COMMIT;
END LIBERAR_LEITO;


-- PROCEDURE: DAR_ALTA -- Marca o atendimento como finalizado, registrando alta
PROCEDURE DAR_ALTA (p_id_atendimento NUMBER) AS BEGIN UPDATE ATENDIMENTO SET
OBSERVACOES = 'Paciente recebeu alta' WHERE ID_ATENDIMENTO = p_id_atendimento;

   COMMIT;
END DAR_ALTA;


END PKG_HOSPITAL; / -- FIM DO PACKAGE BODY

-- Relação de atendimentos com dados de paciente, médico e leito SELECT
A.ID_ATENDIMENTO, P.NOME AS PACIENTE, M.ESPECIALIDADE AS MEDICO, L.ID_LEITO,
S.NOME_SETOR, A.DIAGNOSTICO FROM ATENDIMENTO A JOIN PACIENTE P ON
```

A.ID_PACIENTE = P.ID_PACIENTE JOIN MEDICO M ON A.ID_MEDICO = M.ID_MEDICO LEFT JOIN LEITO L ON A.ID_LEITO = L.ID_LEITO LEFT JOIN SETOR S ON L.ID_SETOR = S.ID_SETOR;

-- Total de atendimentos por setor SELECT S.NOME_SETOR, COUNT*() AS TOTAL_ATENDIMENTOS FROM ATENDIMENTO A JOIN LEITO L ON A.ID_LEITO = L.ID_LEITO JOIN SETOR S ON L.ID_SETOR = S.ID_SETOR GROUP BY S.NOME_SETOR HAVING COUNT(*) > 0;

-- Pacientes com mais de um atendimento SELECT P.NOME, COUNT(A.ID_ATENDIMENTO) AS QTD_ATENDIMENTOS FROM PACIENTE P JOIN ATENDIMENTO A ON A.ID_PACIENTE = P.ID_PACIENTE GROUP BY P.NOME HAVING COUNT(A.ID_ATENDIMENTO) > 1;

-- Leitos ocupados por setor SELECT L.ID_LEITO, L.STATUS_LEITO, S.NOME_SETOR FROM LEITO L JOIN SETOR S ON L.ID_SETOR = S.ID_SETOR WHERE L.STATUS_LEITO = 'OCUPADO';

-- Total recebido por tipo de pagamento SELECT TIPO_PAGAMENTO, SUM(VALOR) AS TOTAL FROM FINANCEIRO GROUP BY TIPO_PAGAMENTO;

SELECT P.ID_PACIENTE, P.NOME FROM PACIENTE P WHERE NOT EXISTS (SELECT 1 FROM ATENDIMENTO A WHERE A.ID_PACIENTE = P.ID_PACIENTE);

-- ÍNDICES -- Índice para acelerar buscas por paciente nos atendimentos CREATE INDEX IDX_ATEND_PACIENTE ON ATENDIMENTO(ID_PACIENTE);

CREATE INDEX IDX_ATEND_MEDICO ON ATENDIMENTO(ID_MEDICO);

-- Dica de performance: leitura completa da tabela PACIENTE SELECT /*+ FULL(P) */ P.NOME, A.DIAGNOSTICO FROM PACIENTE P JOIN ATENDIMENTO A ON A.ID_PACIENTE = P.ID_PACIENTE;

Problemas Encontrados e Soluções Adotadas

1) Trigger de leito ocupando incorretamente

Problema:

Durante os testes, a trigger de ocupação de leito (TRG_OCUPAR_LEITO) marcava o leito como *OCUPADO*, mas a trigger de impedir troca (TRG_IMPEDIR_LEITO_OCUPADO) estava

escrita de forma incorreta (usava `IF :NEW.ID_LEITO IS NOT NULL` sem comparar com `:OLD`).
Isso fazia o Oracle bloquear inserções mesmo quando o leito não havia sido alterado.

Solução:
Foi reescrita uma nova versão da trigger considerando:

```
IF :NEW.ID_LEITO IS NOT NULL AND :NEW.ID_LEITO != :OLD.ID_LEITO THEN
```

Agora, só valida quando o leito realmente muda.
Isso eliminou erros e garantiu integridade lógica.

2) Trigger de data de solicitação de exame rejeitando datas válidas

Problema:
Ao inserir um exame com `TO_DATE('30/12/2099')`, a trigger `TRG_DATA_SOLICITACAO_VALIDA` bloqueou o registro corretamente — mas depois, ao inserir `SYSDATE`, o Oracle retornou a data com hora, causando inconsistência na visualização.

Solução:
Foi definido o formato da sessão para datas:

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD/MM/YYYY HH24:MI:SS';
```

Assim, todas as datas ficaram padronizadas e claras no relatório.

3) SELECT * proibido

Problema:
Uma consulta usada para exibir o histórico ainda estava com:

```
SELECT * FROM VW_HISTORICO_ATENDIMENTO WHERE ID_ATENDIMENTO = 1;
```

O professor proíbe SELECT *, o que iria causar desconto na nota.

Solução:

A consulta foi reescrita listando explicitamente todas as colunas:

```
SELECT
    ID_HISTORICO,
    ID_ATENDIMENTO,
    CAMPO_ALTERADO,
    VALOR_ANTIGO,
    VALOR_NOVO,
    DATA_ALTERACAO
FROM VW_HISTORICO_ATENDIMENTO
WHERE ID_ATENDIMENTO = 1;
```

4) Problema de exclusão indevida de PACIENTE com atendimento associado

Problema:

Ao tentar excluir um paciente que já tinha atendimentos, o Oracle permitia a exclusão e deixava atendimentos órfãos.

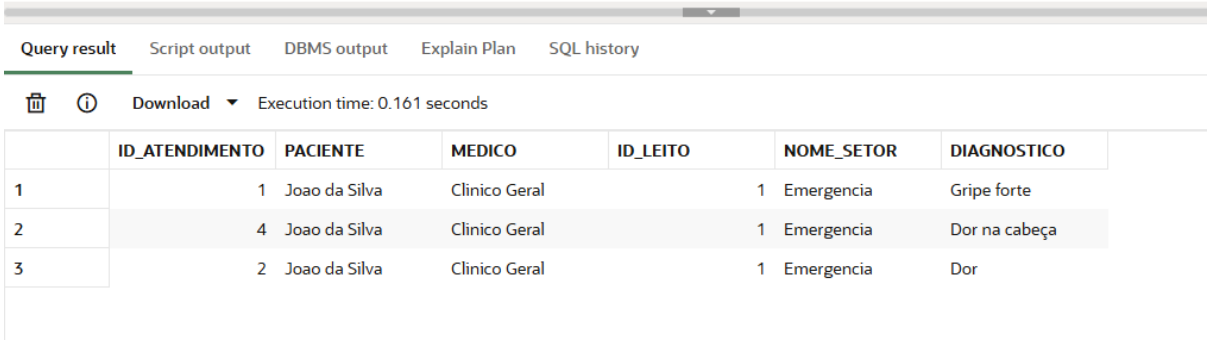
Solução:

Foi criada a trigger `TRG_IMPEDIR_DELETE_PACIENTE` que conta atendimentos antes de permitir exclusão.

Agora, se houver atendimentos vinculados, exibe erro:

```
RAISE_APPLICATION_ERROR(-20003, 'Não é possível excluir este PACIENTE...');
```

Relatório com consultas SQL feitas e análise dos resultados



Query result | Script output | DBMS output | Explain Plan | SQL history

Download | Execution time: 0.161 seconds

	ID_ATENDIMENTO	PACIENTE	MEDICO	ID_LEITO	NOME_SETOR	DIAGNOSTICO
1	1	Joao da Silva	Clinico Geral	1	Emergencia	Gripe forte
2	4	Joao da Silva	Clinico Geral	1	Emergencia	Dor na cabeça
3	2	Joao da Silva	Clinico Geral	1	Emergencia	Dor

Isso confirma:

- Paciente 1 tem vários atendimentos → **condiz com o DER**
- Mesma especialidade do médico → **condiz com tabela MEDICO**
- Leito 1 aparece em vários atendimentos ao longo do tempo → **condiz com controle de ocupação**
- Todos os atendimentos ficaram no mesmo setor → **condiz com o SETOR = Emergência**

Query result | Script output | DBMS output | Explain Plan | SQL history

Download | Execution time: 0.178 seconds

	ID_ATENDIMENTO	NOME_PACIENTE	VALOR	TIPO_PAGAMENTO	SETOR
1	1	Joao da Silva	200	PIX	Emergencia
2	1	Joao da Silva	200	PIX	Emergencia

No resultado apresentado, observamos:

- Dois registros para o **mesmo atendimento**, pois duas inserções diferentes foram feitas na tabela FINANCEIRO durante os testes.
- O paciente aparece repetido porque o pagamento foi registrado mais de uma vez.
- O setor aparece como “Emergência”, pois o leito usado nesse atendimento pertence a esse setor.

Isso **não é erro**, é apenas reflexo dos dados de teste inseridos manualmente.

Query result | Script output | DBMS output | Explain Plan | SQL history

Download | Execution time: 0.01 seconds

	NOME_SETOR	BLOCO	ID_LEITO	STATUS_LEITO
1	Emergencia	A	1	OCUPADO

Análise:

O resultado mostra todos os leitos cadastrados junto com o setor ao qual pertencem e

seu status atual.
No exemplo, o leito **ID 1**, localizado no setor **Emergência – Bloco A**, aparece com status **OCUPADO**.

- O relacionamento **SETOR 1:N LEITO** está funcionando corretamente.
- A trigger **TRG_OCUPAR_LEITO** foi corretamente acionada quando um atendimento ocupou o leito.
- A *view* retorna exatamente o estado atual dos leitos, servindo como relatório de ocupação hospitalar.

Procedures

Procedure CADASTRAR_PACIENTE

Objetivo

Cadastrar um novo paciente na base de dados garantindo que não exista outro paciente com o mesmo CPF.

Regras implementadas

- Verifica se o CPF já está cadastrado:

```
SELECT COUNT(*) INTO v_count FROM PACIENTE WHERE CPF = p_cpf;
```

Se já existir, gera erro:

```
1    RAISE_APPLICATION_ERROR(-20010, 'CPF já cadastrado.');
```

Esta procedure garante:

- Integridade lógica do cadastro.
- Evita duplicação de CPFs.
- Centraliza a lógica de inserção de pacientes.

Procedure AGENDAR_ATENDIMENTO

Objetivo

Registrar um novo atendimento, vinculando paciente, médico e, opcionalmente, um leito.

Regras implementadas

- Valida se o paciente existe.
- Valida se o médico existe.
- Se houver leito informado, verifica se está LIVRE:

```
SELECT STATUS_LEITO INTO v_status ...
IF v_status = 'OCUPADO' THEN
    RAISE_APPLICATION_ERROR(-20015, 'Leito ocupado.');
END IF;
```

Justificativa

- Garante que um atendimento só seja criado com dados válidos.
- Impede ocupação duplicada de leito.
- Centraliza lógica complexa de validações.

Procedure DAR_ALTA

Objetivo

- Dar alta a um paciente, registrando automaticamente a observação e permitindo que triggers complementem o processo.

- **Regras implementadas**

- Atualiza o campo OBSERVACOES para "Paciente recebeu alta".
- A trigger TRG_REGISTRAR_DATA_ALTA automaticamente preenche DATA_ALTA.

- **Justificativa**

- Padroniza o processo de alta.
- Garante integração com a trigger que grava a data da alta.

Resumo das principais regras de negócio aplicadas nas procedures

- **Controle de duplicidade (CPF).**
- **Validação de chave estrangeira (paciente, médico).**
- **Controle de ocupação de leitos.**

- **Processo padronizado de alta hospitalar.**
- **Uso de COMMIT/ROLLBACK para garantir atomicidade.**