

**Curso: Tecnologia em Sistemas para Internet (EaD)**

**Disciplina: Banco de Dados II**

**Professor:** **Angelo Augusto Frozza**

**TRABALHO FINAL BANCO DE DADOS II**

*Jhonn L. S. Gonçalves[[1]](#footnote-0)*

a) Estudo de Caso: Sistema de Gerenciamento de Academia

Objetivo:

O projeto visa desenvolver um sistema de banco de dados para gerenciar, de forma eficiente, as operações de uma academia multifuncional que oferece serviços como musculação, zumba, crossfit e artes marciais. O sistema centralizará informações de alunos, funcionários, aulas, equipamentos e planos, otimizando recursos, automatizando processos e oferecendo relatórios estratégicos para suporte à decisão.

Descrição do Sistema:

O sistema organiza os principais elementos da academia em entidades inter-relacionadas, conforme descrito abaixo:

ALUNO: Representa os membros da academia, com atributos como nome, CPF, telefone, endereço e data de inscrição. Relaciona-se com aulas (frequência) e planos contratados.

EMPREGADO: Representa os funcionários, incluindo instrutores, administradores e zeladores, com dados como nome, CPF, cargo, salário e data de contratação.

- INSTRUTOR: Ministram aulas e possuem qualificações vinculadas à entidade CAPACITAÇÃO.

- ADMINISTRADOR: Gerencia contratos, funcionários e fornecedores.

- ZELADOR: Responsável pela manutenção dos equipamentos e infraestrutura.

PLANO: Representa os contratos oferecidos (mensal, trimestral, etc.) com informações sobre preço, duração e status (ativo/inativo).

AULA: Contém dados sobre atividades como nome, capacidade, duração e horários, permitindo o controle de lotação e frequência.

EQUIPAMENTO: Representa os aparelhos, organizados por departamentos, com controle de uso e manutenção preventiva.

CAPACITAÇÃO: Registra os cursos e certificações dos instrutores.

PAGAMENTO: Controla as transações financeiras relacionadas aos planos dos alunos.

FORNECEDOR: Registra empresas que fornecem equipamentos e serviços.

Principais Relacionamentos:

- Frequentam (ALUNO - AULA): Monitora a participação dos alunos em aulas e a ocupação das mesmas.

- Ministram (INSTRUTOR - AULA): Relaciona as aulas aos instrutores qualificados.

- Possuem (DEPARTAMENTO - EQUIPAMENTO): Relaciona os equipamentos ao setor responsável.

- Matricula (ALUNO - PLANO): Vínculo entre alunos e seus planos contratados.

Relatórios e Consultas Frequentes:

- Gestão de Aulas: Aulas com maior/menor ocupação, frequência por aluno.

- Desempenho dos Funcionários: Carga horária e avaliações de instrutores.

- Controle Financeiro: Receita por tipo de plano, pagamentos pendentes.

- Gestão de Equipamentos: Frequência de uso e planejamento de manutenções.

- Análise de Alunos: Segmentação por faixa etária e padrões de frequência.

- Controle de Acesso: Identificação de horários de pico.

Benefícios do Sistema:

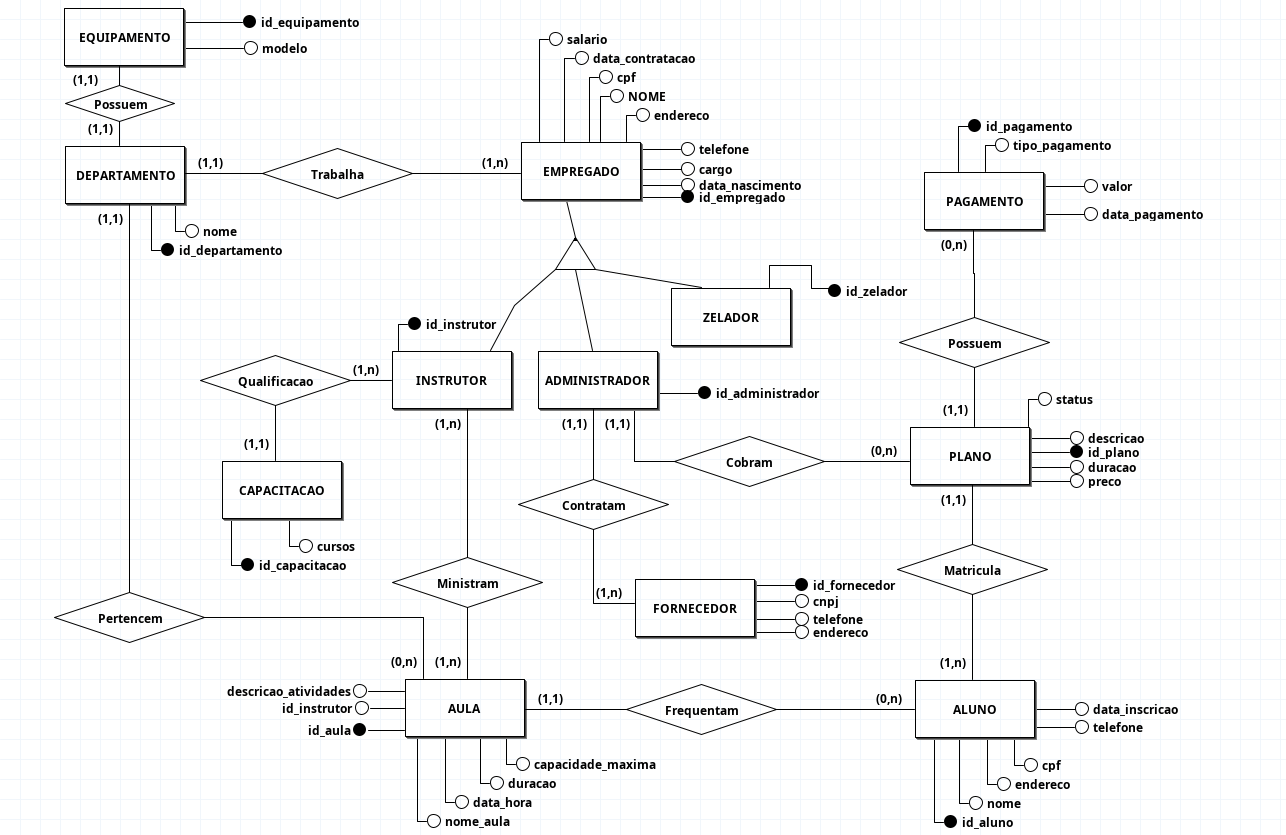
- Otimização de Recursos: Integração de dados e automação de processos.

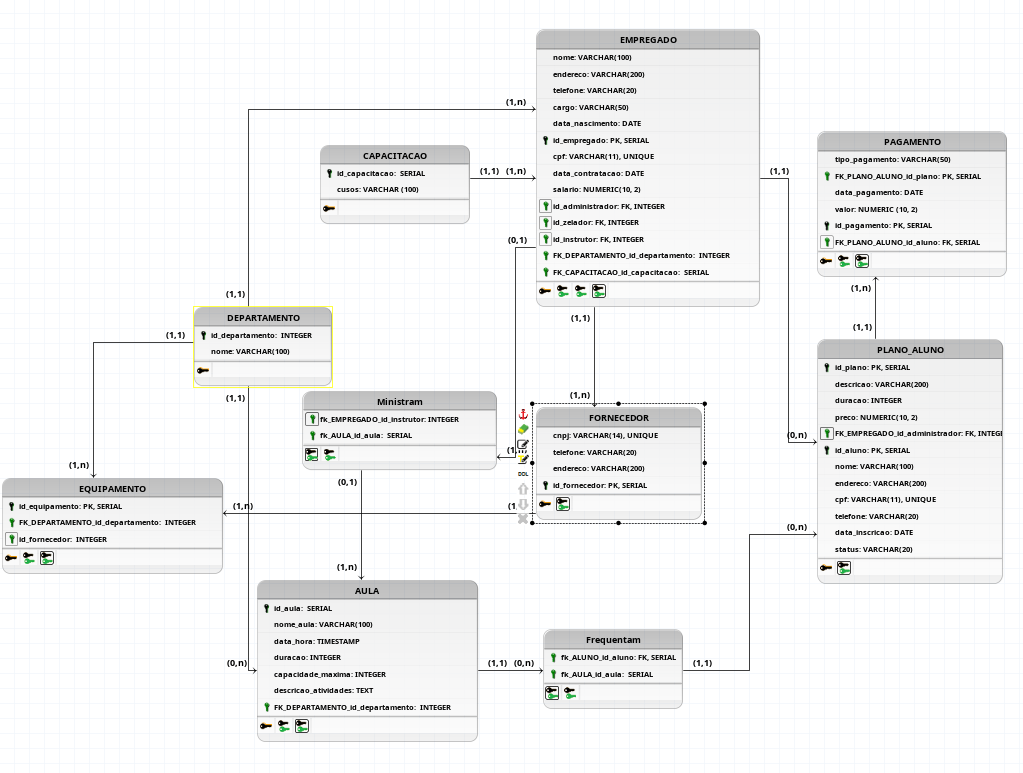
- Gestão Estratégica: Relatórios detalhados para suporte à decisão.

- Melhoria na Experiência do Aluno: Personalização de planos e organização eficiente.

- Sustentabilidade: Monitoramento eficiente de recursos físicos e humanos.

**b) Modelagem Conceitual e Modelagem Lógica**





**c)** Projeto Físico DDL

-- Definição de Domínios

-- Usamos domínios para garantir consistência e reutilização de tipos em várias tabelas.

CREATE DOMAIN dom\_varchar\_100 AS VARCHAR(100); -- Nome de tamanho fixo para campos textuais

CREATE DOMAIN dom\_varchar\_200 AS VARCHAR(200); -- Usado para endereços mais longos

CREATE DOMAIN dom\_varchar\_20 AS VARCHAR(20); -- Para armazenar números de telefone

CREATE DOMAIN dom\_varchar\_11 AS VARCHAR(11); -- Para armazenar CPF

CREATE DOMAIN dom\_varchar\_14 AS VARCHAR(14); -- Para armazenar CNPJ

CREATE DOMAIN dom\_varchar\_50 AS VARCHAR(50); -- Para tipo de pagamento e outros textos curtos

CREATE DOMAIN dom\_numeric\_2\_2 AS NUMERIC(20, 2); -- Para valores monetários com 2 casas decimais

CREATE DOMAIN dom\_date AS DATE; -- Para armazenar datas simples

CREATE DOMAIN dom\_timestamp AS TIMESTAMP; -- Para data e hora

CREATE DOMAIN dom\_status AS VARCHAR(20) CHECK (VALUE IN ('ativo', 'inativo')); -- Controle de status

-- Tabela DEPARTAMENTO

-- Tabelas de referência a entidades simples e com dados não duplicados.

CREATE TABLE DEPARTAMENTO (

id\_departamento SERIAL PRIMARY KEY, -- SERIAL para auto incremento

nome dom\_varchar\_100 NOT NULL -- Garantimos que o nome do departamento é obrigatório

);

-- Tabela CAPACITACAO

CREATE TABLE CAPACITACAO (

id\_capacitacao SERIAL PRIMARY KEY, -- Auto incremento para ID

cursos dom\_varchar\_100 NOT NULL -- Campo obrigatório para o nome do curso

);

-- Tabela FORNECEDOR

-- Armazenamento de fornecedores, onde garantimos que o CNPJ seja único e não nulo.

CREATE TABLE FORNECEDOR (

id\_fornecedor SERIAL PRIMARY KEY, -- SERIAL para auto incremento

cnpj dom\_varchar\_14 UNIQUE NOT NULL, -- CNPJ único, não pode ser nulo

endereco dom\_varchar\_200 NOT NULL -- Endereço obrigatório

);

-- Tabela EMPREGADO

-- Tabela com detalhes de funcionários, incluindo referências a outros empregados como administradores, zeladores e instrutores.

CREATE TABLE EMPREGADO (

id\_empregado SERIAL PRIMARY KEY, -- Auto incremento

nome dom\_varchar\_100 NOT NULL, -- Nome obrigatório

endereco dom\_varchar\_200, -- Endereço opcional

telefone dom\_varchar\_20, -- Telefone opcional

cargo dom\_varchar\_100, -- Cargo do empregado

data\_nascimento dom\_date, -- Data de nascimento

cpf dom\_varchar\_11 UNIQUE NOT NULL, -- CPF único e obrigatório

data\_contratacao dom\_date, -- Data de contratação

salario dom\_numeric\_2\_2, -- Salário com 2 casas decimais

id\_administrador INTEGER, -- Referência ao administrador (auto referência)

id\_zelador INTEGER, -- Referência ao zelador (auto referência)

id\_instrutor INTEGER, -- Referência ao instrutor (auto referência)

id\_departamento INTEGER REFERENCES DEPARTAMENTO(id\_departamento) ON DELETE SET NULL, -- Relacionamento com DEPARTAMENTO

id\_capacitacao INTEGER REFERENCES CAPACITACAO(id\_capacitacao) ON DELETE SET NULL, -- Relacionamento com CAPACITACAO

-- Regras de integridade referencial para garantir a consistência dos dados

CONSTRAINT fk\_administrador FOREIGN KEY (id\_administrador) REFERENCES EMPREGADO(id\_empregado) ON DELETE SET NULL,

CONSTRAINT fk\_zelador FOREIGN KEY (id\_zelador) REFERENCES EMPREGADO(id\_empregado) ON DELETE SET NULL,

CONSTRAINT fk\_instrutor FOREIGN KEY (id\_instrutor) REFERENCES EMPREGADO(id\_empregado) ON DELETE SET NULL

);

-- Tabela EQUIPAMENTO

-- Equipamentos são associados a departamentos e fornecedores.

CREATE TABLE EQUIPAMENTO (

id\_equipamento SERIAL PRIMARY KEY, -- Auto incremento para ID

nome dom\_varchar\_100 NOT NULL, -- Nome do equipamento

id\_departamento INTEGER REFERENCES DEPARTAMENTO(id\_departamento) ON DELETE CASCADE, -- Relacionamento com DEPARTAMENTO

id\_fornecedor INTEGER REFERENCES FORNECEDOR(id\_fornecedor) ON DELETE SET NULL -- Relacionamento com FORNECEDOR

);

-- Tabela AULA

-- Detalhes sobre as aulas, incluindo horário, instrutores e capacidade.

CREATE TABLE AULA (

id\_aula SERIAL PRIMARY KEY, -- Auto incremento para ID da aula

nome\_aula dom\_varchar\_100 NOT NULL, -- Nome obrigatório para a aula

data\_hora dom\_timestamp NOT NULL, -- Data e hora obrigatória da aula

duracao INTEGER CHECK (duracao > 0), -- Duracao maior que 0

id\_instrutor INTEGER REFERENCES EMPREGADO(id\_empregado) ON DELETE SET NULL, -- Relacionamento com o instrutor

capacidade\_maxima INTEGER CHECK (capacidade\_maxima > 0), -- Capacidade maior que 0

descricao\_atividades TEXT, -- Descrição das atividades realizadas

id\_departamento INTEGER REFERENCES DEPARTAMENTO(id\_departamento) ON DELETE CASCADE -- Relacionamento com DEPARTAMENTO

);

-- Tabela PLANO\_ALUNO

-- Relacionamento de planos com alunos, com informações detalhadas do aluno.

CREATE TABLE PLANO\_ALUNO (

id\_plano SERIAL NOT NULL, -- Auto incremento para ID do plano

id\_aluno SERIAL NOT NULL, -- ID do aluno

descricao dom\_varchar\_200, -- Descrição do plano

duracao INTEGER CHECK (duracao > 0), -- Duracao maior que 0

preco dom\_numeric\_2\_2 CHECK (preco >= 0), -- Preço não pode ser negativo

id\_administrador INTEGER REFERENCES EMPREGADO(id\_empregado) ON DELETE SET NULL, -- Referência ao administrador responsável

nome dom\_varchar\_100 NOT NULL, -- Nome do aluno

endereco dom\_varchar\_200 NOT NULL, -- Endereço do aluno

telefone dom\_varchar\_20, -- Telefone do aluno

cpf dom\_varchar\_11 UNIQUE NOT NULL, -- CPF único do aluno

data\_insercao dom\_date DEFAULT CURRENT\_DATE, -- Data de inserção, padrão para a data atual

status dom\_status NOT NULL, -- Status do plano

PRIMARY KEY (id\_plano, id\_aluno) -- Chave primária composta

);

-- Tabela PAGAMENTO

-- Relacionamento entre planos e pagamentos realizados pelos alunos.

CREATE TABLE PAGAMENTO (

id\_pagamento SERIAL PRIMARY KEY, -- Auto incremento para ID do pagamento

tipo\_pagamento dom\_varchar\_50 NOT NULL, -- Tipo de pagamento (cartão, boleto, etc)

data\_pagamento dom\_date NOT NULL DEFAULT CURRENT\_DATE, -- Data do pagamento

valor dom\_numeric\_2\_2 CHECK (valor >= 0), -- Preço não pode ser negativo

id\_plano INTEGER NOT NULL, -- Referência ao plano do aluno

id\_aluno INTEGER NOT NULL, -- Referência ao aluno

FOREIGN KEY (id\_plano, id\_aluno) REFERENCES PLANO\_ALUNO(id\_plano, id\_aluno) ON DELETE CASCADE -- Chave estrangeira composta

);

-- Tabela Ministram (Associação N:M entre EMPREGADO e AULA)

-- Registro de aulas ministradas pelos instrutores.

CREATE TABLE Ministram (

id\_instrutor INTEGER NOT NULL, -- ID do instrutor

id\_aula INTEGER NOT NULL, -- ID da aula

PRIMARY KEY (id\_instrutor, id\_aula), -- Chave primária composta

FOREIGN KEY (id\_instrutor) REFERENCES EMPREGADO(id\_empregado) ON DELETE CASCADE, -- Relacionamento com EMPREGADO

FOREIGN KEY (id\_aula) REFERENCES AULA(id\_aula) ON DELETE CASCADE -- Relacionamento com AULA

);

-- Tabela Frequentam (Associação N:M entre PLANO\_ALUNO e AULA)

-- Registro de quais alunos frequentam quais aulas.

CREATE TABLE Frequentam (

id\_plano INTEGER NOT NULL, -- ID do plano do aluno

id\_aluno INTEGER NOT NULL, -- ID do aluno

id\_aula INTEGER NOT NULL, -- ID da aula

PRIMARY KEY (id\_plano, id\_aluno, id\_aula), -- Chave primária composta

FOREIGN KEY (id\_plano, id\_aluno) REFERENCES PLANO\_ALUNO(id\_plano, id\_aluno) ON DELETE CASCADE, -- Relacionamento com PLANO\_ALUNO

FOREIGN KEY (id\_aula) REFERENCES AULA(id\_aula) ON DELETE CASCADE -- Relacionamento com AULA

);

-- Índices para melhorar o desempenho nas consultas frequentes

CREATE INDEX idx\_empregado\_cpf ON EMPREGADO(cpf); -- Index para melhorar buscas por CPF de empregado

CREATE INDEX idx\_plano\_aluno\_cpf ON PLANO\_ALUNO(cpf); -- Index

**d)** Scripts SQL para consultas, algumas usando diferentes tipos de JOIN.

-- Consulta 1: Aulas e seus Instrutores por Departamento

SELECT a.nome\_aula, a.data\_hora, e.nome as instrutor, d.nome as departamento

FROM AULA a

INNER JOIN EMPREGADO e ON a.id\_instrutor = e.id\_empregado -- Junta a tabela AULA com EMPREGADO para obter o nome do instrutor

INNER JOIN DEPARTAMENTO d ON a.id\_departamento = d.id\_departamento -- Junta a tabela AULA com DEPARTAMENTO para obter o nome do departamento

ORDER BY a.data\_hora; -- Ordena os resultados pela data e hora da aula

-- Consulta 2: Pagamentos de Alunos Ativos

SELECT pa.nome as aluno, pa.cpf, p.tipo\_pagamento, p.valor, p.data\_pagamento

FROM PLANO\_ALUNO pa

LEFT JOIN PAGAMENTO p ON pa.id\_plano = p.id\_plano AND pa.id\_aluno = p.id\_aluno -- Junta a tabela PLANO\_ALUNO com PAGAMENTO para obter os pagamentos dos alunos

WHERE pa.status = 'ativo' -- Filtra apenas os alunos com plano ativo

ORDER BY p.data\_pagamento DESC; -- Ordena os resultados pela data de pagamento em ordem decrescente

-- Consulta 3: Equipamentos por Departamento e Fornecedor

SELECT d.nome as departamento,

e.nome as equipamento,

f.cnpj as fornecedor\_cnpj

FROM EQUIPAMENTO e

JOIN DEPARTAMENTO d ON e.id\_departamento = d.id\_departamento -- Junta a tabela EQUIPAMENTO com DEPARTAMENTO para obter o nome do departamento

JOIN FORNECEDOR f ON e.id\_fornecedor = f.id\_fornecedor -- Junta a tabela EQUIPAMENTO com FORNECEDOR para obter o CNPJ do fornecedor

ORDER BY d.nome; -- Ordena os resultados pelo nome do departamento

-- Consulta 4: Total de Aulas por Instrutor

SELECT e.nome, e.cpf,

(SELECT COUNT(\*)

FROM MINISTRAM m

WHERE m.id\_instrutor = e.id\_empregado) as total\_aulas -- Subconsulta para contar o total de aulas ministradas por cada instrutor

FROM EMPREGADO e

WHERE e.cargo = 'Instrutor' -- Filtra apenas os empregados com cargo de instrutor

ORDER BY total\_aulas DESC; -- Ordena os resultados pelo total de aulas em ordem decrescente

-- Consulta 5: Estatísticas por Departamento

SELECT d.nome as departamento,

COUNT(DISTINCT a.id\_aula) as total\_aulas, -- Conta o total de aulas distintas por departamento

COUNT(DISTINCT f.id\_aluno) as total\_alunos, -- Conta o total de alunos distintos por departamento

AVG(a.capacidade\_maxima) as media\_capacidade -- Calcula a média da capacidade máxima das aulas por departamento

FROM DEPARTAMENTO d

LEFT JOIN AULA a ON d.id\_departamento = a.id\_departamento -- Junta a tabela DEPARTAMENTO com AULA para obter as aulas por departamento

LEFT JOIN FREQUENTAM f ON a.id\_aula = f.id\_aula -- Junta a tabela AULA com FREQUENTAM para obter os alunos por aula

GROUP BY d.nome -- Agrupa os resultados pelo nome do departamento

ORDER BY total\_alunos DESC; -- Ordena os resultados pelo total de alunos em ordem decrescente

**e)** Scripts SQL para Stored Procedures

-- Stored Procedure 1: Cadastro de Novo Aluno

-- Esta SP realiza todo o processo de matrícula de um novo aluno na academia

CREATE OR REPLACE PROCEDURE sp\_cadastrar\_aluno(

p\_nome VARCHAR(100), -- Nome completo do aluno

p\_cpf VARCHAR(11), -- CPF do aluno (apenas números)

p\_endereco VARCHAR(200), -- Endereço completo

p\_telefone VARCHAR(20), -- Telefone com DDD

p\_tipo\_plano VARCHAR(200), -- Descrição do plano escolhido

p\_duracao INTEGER, -- Duração em dias do plano

p\_preco NUMERIC(20,2), -- Valor do plano

p\_id\_admin INTEGER -- ID do administrador responsável

)

LANGUAGE plpgsql

AS $$

DECLARE

v\_id\_plano INTEGER; -- Variável para armazenar o ID do plano gerado

v\_id\_aluno INTEGER; -- Variável para armazenar o ID do aluno gerado

BEGIN

-- Insere os dados do aluno e do plano

INSERT INTO PLANO\_ALUNO(

nome, cpf, endereco, telefone,

descricao, duracao, preco,

id\_administrador, status

)

VALUES (

p\_nome, p\_cpf, p\_endereco, p\_telefone,

p\_tipo\_plano, p\_duracao, p\_preco,

p\_id\_admin, 'ativo'

)

RETURNING id\_plano, id\_aluno INTO v\_id\_plano, v\_id\_aluno; -- Retorna os IDs gerados

-- Registra o pagamento inicial da matrícula

INSERT INTO PAGAMENTO(

tipo\_pagamento, valor, id\_plano, id\_aluno

)

VALUES (

'Matrícula', p\_preco, v\_id\_plano, v\_id\_aluno

);

COMMIT; -- Confirma a transação

END;

$$;

-- Stored Procedure 2 : Registro de Frequência em Aulas

CREATE OR REPLACE PROCEDURE sp\_registrar\_frequencia(

p\_cpf\_aluno VARCHAR(11), -- CPF do aluno que participará da aula

p\_id\_aula INTEGER -- ID da aula que será frequentada

)

LANGUAGE plpgsql

AS $$

DECLARE

v\_id\_plano INTEGER; -- Armazena o ID do plano do aluno

v\_id\_aluno INTEGER; -- Armazena o ID do aluno

v\_capacidade INTEGER; -- Armazena a capacidade máxima da aula

v\_ocupacao INTEGER; -- Armazena o número atual de alunos na aula

BEGIN

-- Busca os dados do aluno

SELECT id\_plano, id\_aluno

INTO v\_id\_plano, v\_id\_aluno

FROM PLANO\_ALUNO

WHERE cpf = p\_cpf\_aluno AND status = 'ativo';

-- Verifica a capacidade da aula

SELECT capacidade\_maxima,

(SELECT COUNT(\*) FROM FREQUENTAM WHERE id\_aula = p\_id\_aula)

INTO v\_capacidade, v\_ocupacao

FROM AULA

WHERE id\_aula = p\_id\_aula;

-- Verifica se há vagas disponíveis

IF v\_ocupacao >= v\_capacidade THEN

RAISE EXCEPTION 'Aula está com capacidade máxima atingida';

END IF;

-- Registra a frequência do aluno

INSERT INTO FREQUENTAM(id\_plano, id\_aluno, id\_aula)

VALUES (v\_id\_plano, v\_id\_aluno, p\_id\_aula);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN -- Tratamento de erro quando aluno não é encontrado

RAISE EXCEPTION 'Aluno não encontrado ou plano inativo';

END;

$$;

-- Stored Procedure 3: Relatório de Frequência por Aluno

CREATE OR REPLACE PROCEDURE sp\_relatorio\_frequencia\_aluno(

p\_cpf\_aluno VARCHAR(11) -- CPF do aluno para buscar frequência

)

LANGUAGE plpgsql

AS $$

DECLARE

v\_nome\_aluno VARCHAR(100); -- Armazena nome do aluno

v\_total\_aulas INTEGER; -- Contador de aulas frequentadas

BEGIN

-- Busca nome do aluno

SELECT nome INTO v\_nome\_aluno

FROM PLANO\_ALUNO

WHERE cpf = p\_cpf\_aluno AND status = 'ativo';

-- Conta total de aulas frequentadas

SELECT COUNT(\*) INTO v\_total\_aulas

FROM FREQUENTAM f

JOIN PLANO\_ALUNO pa ON f.id\_plano = pa.id\_plano

WHERE pa.cpf = p\_cpf\_aluno;

-- Exibe o relatório

RAISE NOTICE 'Relatório de Frequência';

RAISE NOTICE 'Aluno: %', v\_nome\_aluno;

RAISE NOTICE 'Total de aulas frequentadas: %', v\_total\_aulas;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE EXCEPTION 'Aluno não encontrado ou inativo';

END;

$$

;

-- Teste da SP1: Cadastrar novo aluno

CALL sp\_cadastrar\_aluno(

'João Silva', -- Nome do aluno

'12345678901', -- CPF do aluno

'Rua Exemplo, 123', -- Endereço do aluno

'48999999999', -- Telefone do aluno

'Plano Mensal', -- Tipo de plano escolhido

30, -- Duração do plano em dias

99.90, -- Preço do plano

1 -- ID do administrador responsável

);

-- Teste da SP2

CALL sp\_registrar\_frequencia('12345678901', 1); -- CPF do aluno e ID da aula

-- Teste da SP3

CALL sp\_relatorio\_frequencia\_aluno('12345678901'); -- CPF do aluno para gerar relatório

**f)** Scripts SQL para Functions

| -- Function 1: Calcular total faturado por período -- Parâmetros de entrada: -- data\_inicio: data inicial do período -- data\_fim: data final do período -- Retorno: valor total faturado no período **CREATE** **OR** **REPLACE** **FUNCTION** fn\_calcular\_faturamento(  data\_inicio DATE, -- Data inicial do período  data\_fim DATE -- Data final do período )  **RETURNS** NUMERIC(20,2) -- Retorna o valor total faturado no período **LANGUAGE** plpgsql **AS** $$ **DECLARE**  total\_faturado NUMERIC(20,2); -- Variável para armazenar o total faturado **BEGIN**  -- Calcula o total faturado no período especificado  **SELECT** **COALESCE**(**SUM**(valor), 0)  **INTO** total\_faturado  **FROM** PAGAMENTO  **WHERE** data\_pagamento **BETWEEN** data\_inicio **AND** data\_fim;    RETURN total\_faturado; -- Retorna o total faturado **END**; $$ ;  -- Function 2: Verificar disponibilidade de vaga em aula -- Parâmetros de entrada: -- p\_id\_aula: ID da aula a ser verificada -- Retorno: boolean (true se há vagas, false se não há) **CREATE** **OR** **REPLACE** **FUNCTION** fn\_verificar\_disponibilidade\_aula(  p\_id\_aula INTEGER -- ID da aula a ser verificada )  **RETURNS** BOOLEAN -- Retorna true se há vagas disponíveis, false caso contrário **LANGUAGE** plpgsql **AS** $$ **DECLARE**  v\_capacidade INTEGER; -- Variável para armazenar a capacidade máxima da aula  v\_ocupacao INTEGER; -- Variável para armazenar o número atual de alunos na aula **BEGIN**  -- Busca capacidade máxima da aula  **SELECT** capacidade\_maxima  **INTO** v\_capacidade  **FROM** AULA  **WHERE** id\_aula = p\_id\_aula;    -- Conta alunos matriculados  **SELECT** **COUNT**(\*)  **INTO** v\_ocupacao  **FROM** FREQUENTAM  **WHERE** id\_aula = p\_id\_aula;    -- Retorna true se há vagas disponíveis  RETURN v\_ocupacao < v\_capacidade; **END**; $$ ;  -- Function 3: Calcular idade do aluno -- Parâmetros de entrada: -- p\_cpf: CPF do aluno -- Retorno: idade em anos **CREATE** **OR** **REPLACE** **FUNCTION** fn\_calcular\_idade\_aluno(  p\_cpf VARCHAR(11) -- CPF do aluno )  **RETURNS** INTEGER -- Retorna a idade em anos **LANGUAGE** plpgsql **AS** $$ **DECLARE**  v\_data\_nascimento DATE; -- Variável para armazenar a data de nascimento do aluno **BEGIN**  -- Busca a data de nascimento do aluno na tabela EMPREGADO  **SELECT** data\_nascimento  **INTO** v\_data\_nascimento  **FROM** EMPREGADO  **WHERE** cpf = p\_cpf;    -- Calcula e retorna a idade do aluno em anos  RETURN EXTRACT(YEAR FROM age(current\_date, v\_data\_nascimento)); **END**; $$ ;  -- Teste Function 1: Calcular faturamento **SELECT** fn\_calcular\_faturamento('2024-01-01', '2024-12-31') **as** faturamento\_anual; -- Calcula o faturamento anual para o ano de 2024  -- Teste Function 2: Verificar disponibilidade **SELECT** fn\_verificar\_disponibilidade\_aula(1) **as** tem\_vaga\_disponivel; -- Verifica se há vagas disponíveis para a aula com ID 1  -- Teste Function 3: Calcular idade **SELECT** fn\_calcular\_idade\_aluno('12345678901') **as** idade\_aluno; -- Calcula a idade do aluno com o CPF '12345678901' |
| --- |

**g)** Scripts SQL para Views

-- View 1: Visão geral das aulas com detalhes

-- Esta view fornece informações completas sobre as aulas,

-- incluindo instrutor responsável e departamento

CREATE OR REPLACE VIEW vw\_detalhes\_aulas AS

SELECT

a.nome\_aula, -- Nome da aula

a.data\_hora, -- Data e hora da aula

a.duracao, -- Duração da aula

a.capacidade\_maxima, -- Capacidade máxima de alunos na aula

e.nome as instrutor, -- Nome do instrutor responsável

d.nome as departamento, -- Nome do departamento

(SELECT COUNT(\*)

FROM FREQUENTAM f

WHERE f.id\_aula = a.id\_aula) as alunos\_matriculados -- Número de alunos matriculados na aula

FROM AULA a

JOIN EMPREGADO e ON a.id\_instrutor = e.id\_empregado -- Junta a tabela AULA com EMPREGADO para obter o nome do instrutor

JOIN DEPARTAMENTO d ON a.id\_departamento = d.id\_departamento; -- Junta a tabela AULA com DEPARTAMENTO para obter o nome do departamento

-- View 2: Relatório financeiro de alunos

-- Esta view apresenta um resumo financeiro dos alunos,

-- incluindo planos e pagamentos realizados

CREATE OR REPLACE VIEW vw\_financeiro\_alunos AS

SELECT

pa.nome as aluno, -- Nome do aluno

pa.cpf, -- CPF do aluno

pa.descricao as plano, -- Descrição do plano do aluno

pa.preco as valor\_plano, -- Valor do plano do aluno

pa.status, -- Status do plano do aluno

COUNT(p.id\_pagamento) as total\_pagamentos, -- Total de pagamentos realizados pelo aluno

SUM(p.valor) as total\_pago -- Valor total pago pelo aluno

FROM PLANO\_ALUNO pa

LEFT JOIN PAGAMENTO p ON pa.id\_plano = p.id\_plano

AND pa.id\_aluno = p.id\_aluno -- Junta a tabela PLANO\_ALUNO com PAGAMENTO para obter os pagamentos dos alunos

GROUP BY pa.nome, pa.cpf, pa.descricao, pa.preco, pa.status; -- Agrupa os resultados pelos dados do aluno

-- View 3: Status dos equipamentos por departamento

-- Esta view mostra a distribuição de equipamentos

-- entre os departamentos e seus fornecedores

CREATE OR REPLACE VIEW vw\_equipamentos\_departamento AS

SELECT

d.nome as departamento, -- Nome do departamento

e.nome as equipamento, -- Nome do equipamento

f.cnpj as fornecedor, -- CNPJ do fornecedor do equipamento

COUNT(e.id\_equipamento) OVER (PARTITION BY d.id\_departamento) as total\_equipamentos -- Total de equipamentos por departamento

FROM DEPARTAMENTO d

LEFT JOIN EQUIPAMENTO e ON d.id\_departamento = e.id\_departamento -- Junta a tabela DEPARTAMENTO com EQUIPAMENTO para obter os equipamentos por departamento

LEFT JOIN FORNECEDOR f ON e.id\_fornecedor = f.id\_fornecedor; -- Junta a tabela EQUIPAMENTO com FORNECEDOR para obter o CNPJ do fornecedor

-- Consultar detalhes das aulas

SELECT \* FROM vw\_detalhes\_aulas; -- Consulta a view vw\_detalhes\_aulas para obter detalhes das aulas

-- Consultar relatório financeiro

SELECT \* FROM vw\_financeiro\_alunos; -- Consulta a view vw\_financeiro\_alunos para obter o relatório financeiro dos alunos

-- Consultar equipamentos por departamento

SELECT \* FROM vw\_equipamentos\_departamento; -- Consulta a view vw\_equipamentos\_departamento para obter o status dos equipamentos por departamento

**h)** Scripts SQL para Triggers

-- Criar tabela de log simplificada

-- Esta tabela armazena as alterações de valores dos planos dos alunos

CREATE TABLE log\_alteracao\_valores (

id\_log SERIAL PRIMARY KEY, -- Identificador único do log

data\_alteracao TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, -- Data e hora da alteração

id\_plano INTEGER, -- ID do plano alterado

id\_aluno INTEGER, -- ID do aluno cujo plano foi alterado

valor\_anterior dom\_numeric\_2\_2, -- Valor anterior do plano

valor\_novo dom\_numeric\_2\_2, -- Novo valor do plano

usuario\_alteracao VARCHAR(50) -- Usuário que realizou a alteração

);

-- Nova função para a trigger

-- Esta função insere um registro na tabela de log quando o valor de um plano é alterado

CREATE OR REPLACE FUNCTION tf\_registrar\_alteracao\_valor()

RETURNS TRIGGER AS $$

BEGIN

IF OLD.preco IS DISTINCT FROM NEW.preco THEN -- Verifica se o valor do plano foi alterado

INSERT INTO log\_alteracao\_valores (

id\_plano,

id\_aluno,

valor\_anterior,

valor\_novo,

usuario\_alteracao

)

VALUES (

OLD.id\_plano, -- ID do plano antes da alteração

OLD.id\_aluno, -- ID do aluno antes da alteração

OLD.preco, -- Valor anterior do plano

NEW.preco, -- Novo valor do plano

current\_user -- Usuário que realizou a alteração

);

END IF;

RETURN NEW;

END;

$$

LANGUAGE plpgsql;

-- Nova trigger

-- Esta trigger chama a função tf\_registrar\_alteracao\_valor após uma atualização na tabela PLANO\_ALUNO

CREATE TRIGGER tg\_registrar\_alteracao\_valor

AFTER UPDATE ON PLANO\_ALUNO

FOR EACH ROW

EXECUTE FUNCTION tf\_registrar\_alteracao\_valor();

-- Verificar preço atual do plano

-- Consulta para verificar o preço atual do plano de um aluno específico

SELECT id\_plano, id\_aluno, preco, status

FROM PLANO\_ALUNO

WHERE id\_plano = 1 AND id\_aluno = 1;

-- Verificar se a tabela de log está vazia

-- Consulta para verificar se a tabela de log está vazia

SELECT \* FROM log\_alteracao\_valores;

-- Atualizar o preço de um plano existente

-- Atualiza o preço de um plano específico para testar a trigger

UPDATE PLANO\_ALUNO

SET preco = 150.00

WHERE id\_plano = 1 AND id\_aluno = 1;

-- Verificar se a alteração foi registrada no log

-- Consulta para verificar se a alteração foi registrada na tabela de log

SELECT \* FROM log\_alteracao\_valores

ORDER BY data\_alteracao DESC;

**i)** Definição de Usuários e permissões

-- Criar usuário administrador com todas as permissões

CREATE ROLE admin\_academia WITH

LOGIN -- Permite login

PASSWORD 'adminIFC' -- Define a senha do usuário

CREATEDB -- Permite criar bancos de dados

CREATEROLE; -- Permite criar novos roles

-- Conceder todas as permissões nas tabelas

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO admin\_academia; -- Concede todas as permissões em todas as tabelas

GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO admin\_academia; -- Concede todas as permissões em todas as sequências

-- Conceder permissão de execução em todas as funções

GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA public TO admin\_academia; -- Concede permissão de execução em todas as funções

-- Criar usuário instrutor com permissões limitadas

CREATE ROLE instrutor\_academia WITH

LOGIN -- Permite login

PASSWORD 'instrutorIFC' -- Define a senha do usuário

NOINHERIT; -- Não herda permissões de outros roles

-- Permissões específicas para instrutores

GRANT SELECT ON TABLE EMPREGADO TO instrutor\_academia; -- Permite selecionar dados da tabela EMPREGADO

GRANT SELECT, INSERT ON TABLE FREQUENTAM TO instrutor\_academia; -- Permite selecionar e inserir dados na tabela FREQUENTAM

GRANT SELECT ON TABLE PLANO\_ALUNO TO instrutor\_academia; -- Permite selecionar dados da tabela PLANO\_ALUNO

GRANT SELECT, UPDATE ON TABLE AULA TO instrutor\_academia; -- Permite selecionar e atualizar dados na tabela AULA

GRANT SELECT ON TABLE DEPARTAMENTO TO instrutor\_academia; -- Permite selecionar dados da tabela DEPARTAMENTO

GRANT SELECT ON TABLE EQUIPAMENTO TO instrutor\_academia; -- Permite selecionar dados da tabela EQUIPAMENTO

-- Revogar acesso a informações sensíveis do instrutor

REVOKE SELECT (cpf, salario, data\_contratacao) ON TABLE EMPREGADO FROM instrutor\_academia; -- Revoga permissão de selecionar campos sensíveis da tabela EMPREGADO

REVOKE SELECT, INSERT, DELETE ON TABLE EMPREGADO FROM instrutor\_academia; -- Revoga permissão de selecionar, inserir e deletar dados na tabela EMPREGADO

REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA public FROM instrutor\_academia; -- Revoga todas as permissões em todas as tabelas

1. Aluno da Turma 2024, nº 2024001375, [jlsgo.dev@gmail.com](mailto:jlsgo.dev@gmail.com)  **1** [↑](#footnote-ref-0)