INSTITUTO FEDERAL DA PARAÍBA - IFPB
Unidade Acadêmica de Informação e Comunicação
CST em Sistemas para Internet

Disciplina: Banco de Dados II – 2024.1

Professores: Damires e Thiago

Grupo: Lucas André da Silva Santos Júnior, Thiago Alexandre Augustinho dos Santos.

Roteiro para Projeto de Banco de Dados Relacional (Requisitos)

**** Monte sua equipe: 2 integrantes**

1. **Descreva o escopo da aplicação que precise do banco de dados (seu projeto).**
    a. Descrição geral das regras do negócio e do que se espera da aplicação/banco de dados

        Sistema de Gerenciamento de Academia

        Descrição do negócio e da necessidade Atualmente, a academia "Gym Old School" possui funcionários e vários equipamentos, mas ainda não informatizou os processos de gerenciamento de membros, manutenção de equipamentos. Cada membro tem um perfil que inclui todos os seus detalhes pessoais. O sistema deve ser capaz de atribuir instrutores a cada aula e gerenciar o uso de equipamentos. Também deve registrar os pagamentos efetuados pelos membros.

    b. Requisitos Funcionais da aplicação

        Cadastrar Membro, Manter dados cadastrais dos Instrutores, Registrar Pagamento, Manter dados cadastrais das Aulas e Gerenciar Equipamentos

    c. Requisitos de Dados

        Requisitos de Dados Membro Aula Instrutor Pagamento Agendamento Equipamento

    *** Veja o exemplo de descrição de aplicação mostrado/anexado.*

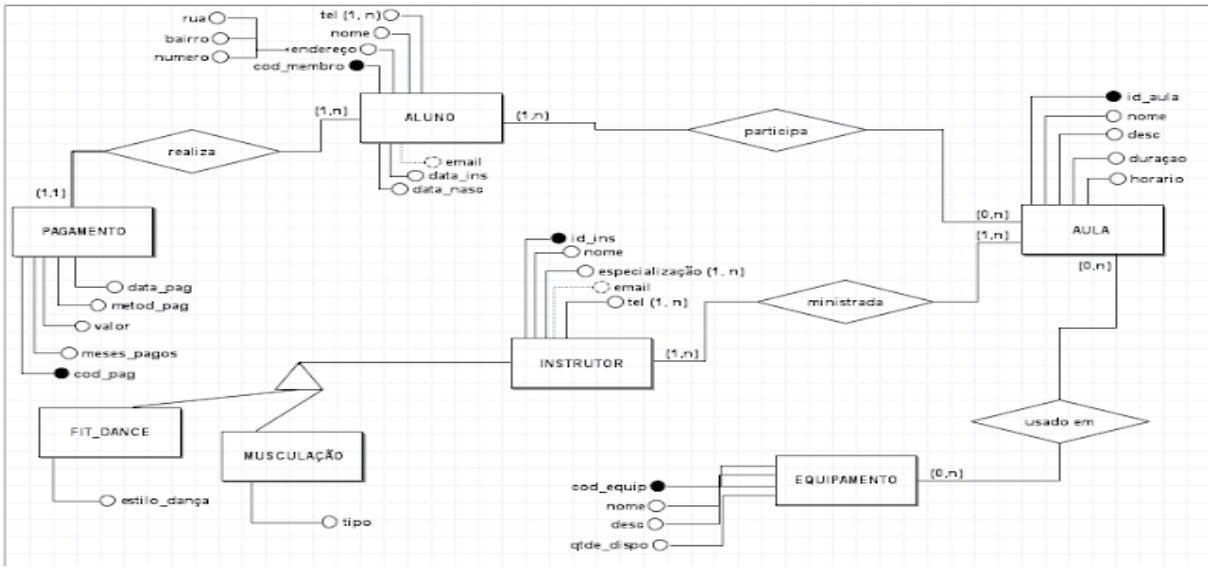
2. **Diagrama Entidade-Relacionamento em nível Conceitual (versão 1)**

Especifique uma versão inicial do seu DER conceitual com as *entidades*, *relacionamentos* e *principais atributos*. As entidades e relacionamentos devem ser especificados conforme requisitos e escopo descritos na seção anterior.

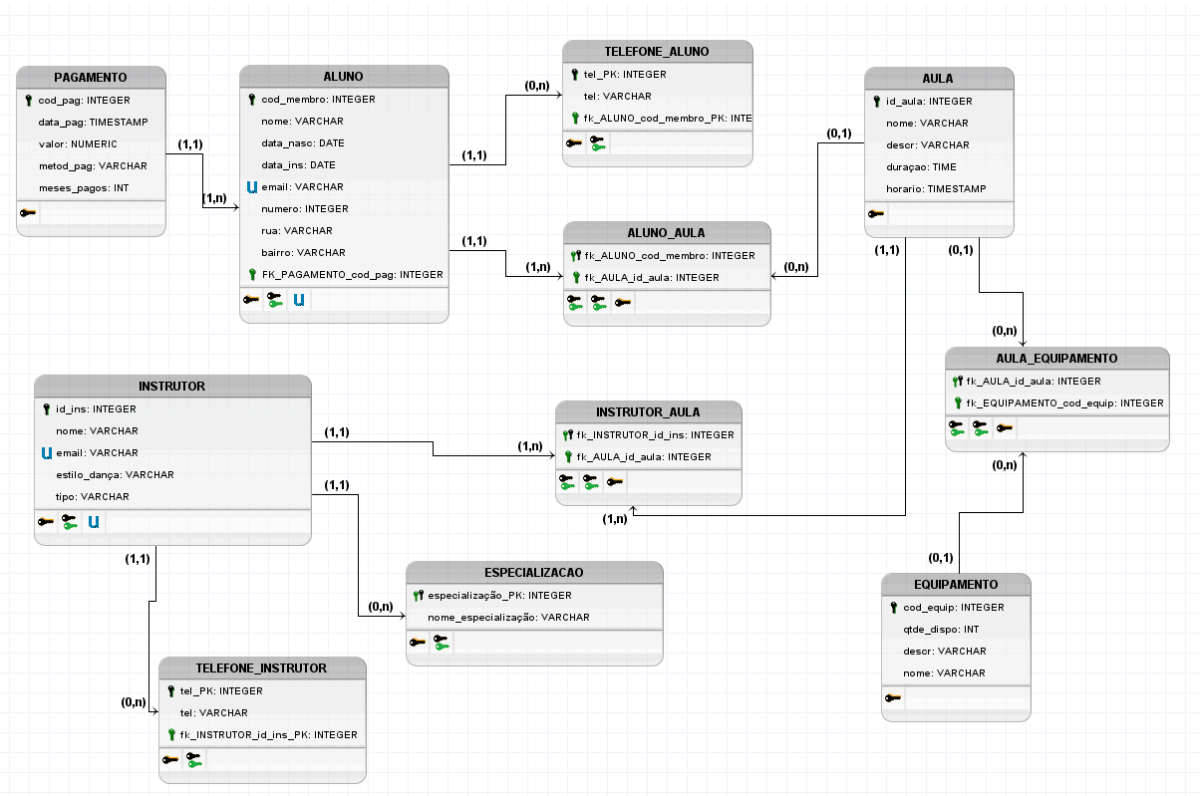
Cada aluno da academia tem dados armazenados, como um código que o identifica, nome, telefones, endereço (rua, bairro e número), opcionalmente o e-mail, data de nascimento e data da inscrição na academia. Cada aluno realiza um pagamento todo mês, e os pagamentos são realizados por vários alunos todos os meses. No pagamento, é armazenado o código para identificação, o código do membro, valor, método de pagamento, meses pagos e a data.

Uma aula pode ou não usar equipamentos da academia, e os equipamentos podem ou não fazer parte das aulas. É importante que os equipamentos sejam registrados, com dados como código para identificação, nome, descrição e quantidade disponível.

Dos instrutores da academia, são armazenados o código de identificação, o nome, especializações, e-mail (opcional) e telefones. É necessário que os instrutores sejam da área de musculação ou Fit Dance. O instrutor de Fit Dance possui alguns estilos, como por exemplo ritmos que priorizam a queima de gordura e ritmos que servem para pessoas mais frágeis que precisam mexer as articulações. Já o instrutor de musculação pode ter uma especialidade em treino metabólico ou focado em hipertrofia. Um instrutor ministra várias aulas e as aulas são ministradas por, no mínimo, um instrutor.



Modelo lógico :



Implementação do projeto de BDR no SGBD PostgreSQL :

–Criações de tabelas

```
CREATE TABLE ALUNO (
    cod_membro INTEGER PRIMARY KEY,
    nome VARCHAR(100),
    data_nasc DATE CHECK (data_nasc < current_date),
    data_ins DATE,
    email VARCHAR(30) UNIQUE CHECK (email LIKE '%_@__%.__%'),
    numero INTEGER,
    rua VARCHAR(15),
    bairro VARCHAR(15),
    FK_PAGAMENTO_cod_pag INTEGER
);

CREATE TABLE AULA (
    id_aula INTEGER PRIMARY KEY,
    nome VARCHAR(100),
    descr VARCHAR(100),
    duraçao TIME CHECK (duraçao > '00:30:00'),
    horario TIMESTAMP CHECK (EXTRACT(HOUR FROM horario) BETWEEN 6 AND 22)
);

CREATE TABLE INSTRUTOR (
    id_ins INTEGER PRIMARY KEY,
    nome VARCHAR(100),
    email VARCHAR(30) UNIQUE,
    estilo_dança VARCHAR(15),
    tipo VARCHAR(15)

);

CREATE TABLE PAGAMENTO (

        cod_pag INTEGER PRIMARY KEY,
        data_pag TIMESTAMP NOT NULL,
    valor NUMERIC(6,2) NOT NULL CHECK (valor > 60),
    metod_pag VARCHAR(15) NOT NULL,
    meses_pagos INT CHECK (meses_pagos BETWEEN 1 AND 12)
```

```
);

CREATE TABLE EQUIPAMENTO (
    cod_equip INT PRIMARY KEY,
        qtde_dispo INT,
    descr VARCHAR(100),
    nome VARCHAR(100)

);

CREATE TABLE TELEFONE_ALUNO (
    tel_PK INTEGER NOT NULL PRIMARY KEY,
    tel VARCHAR(15),
    fk_ALUNO_cod_membro_PK INTEGER
);

CREATE TABLE ESPECIALIZACAO (
    especialização_PK INTEGER NOT NULL PRIMARY KEY,
    nome_especialização VARCHAR(15)
);

CREATE TABLE TELEFONE_INSTRUTOR (
    tel_PK INTEGER NOT NULL PRIMARY KEY,
    tel VARCHAR(15),
    fk_INSTRUTOR_id_ins_PK INTEGER
);

CREATE TABLE INSTRUTOR_AULA (
    fk_INSTRUTOR_id_ins INTEGER PRIMARY KEY,
    fk_AULA_id_aula INTEGER
);

CREATE TABLE AULA_EQUIPAMENTO (
    fk_AULA_id_aula INTEGER PRIMARY KEY,
    fk_EQUIPAMENTO_cod_equip INTEGER
);

CREATE TABLE ALUNO_AULA (
    fk_ALUNO_cod_membro INTEGER PRIMARY KEY,
    fk_AULA_id_aula INTEGER
);

ALTER TABLE ALUNO ADD CONSTRAINT FK_ALUNO_2
    FOREIGN KEY (FK_PAGAMENTO_cod_pag)
    REFERENCES PAGAMENTO (cod_pag);
```

```
ALTER TABLE TELEFONE_ALUNO ADD CONSTRAINT FK_TELEFONE_ALUNO_2
   FOREIGN KEY (fk_ALUNO_cod_membro_PK)
   REFERENCES ALUNO (cod_membro);

ALTER TABLE ESPECIALIZACAO ADD CONSTRAINT FK_ESPECIALIZACAO_2
   FOREIGN KEY (especialização_PK)
   REFERENCES INSTRUTOR (id_ins);

ALTER TABLE TELEFONE_INSTRUTOR ADD CONSTRAINT
FK_TELEFONE_INSTRUTOR_2
   FOREIGN KEY (fk_INSTRUTOR_id_ins_PK)
   REFERENCES INSTRUTOR (id_ins);

ALTER TABLE INSTRUTOR_AULA ADD CONSTRAINT FK_INSTRUTOR_AULA_1
   FOREIGN KEY (fk_INSTRUTOR_id_ins)
   REFERENCES INSTRUTOR (id_ins);

ALTER TABLE INSTRUTOR_AULA ADD CONSTRAINT FK_INSTRUTOR_AULA_2
   FOREIGN KEY (fk_AULA_id_aula)
   REFERENCES AULA (id_aula);

ALTER TABLE AULA_EQUIPAMENTO ADD CONSTRAINT FK_AULA_EQUIPAMENTO_1
   FOREIGN KEY (fk_AULA_id_aula)
   REFERENCES AULA (id_aula);

ALTER TABLE AULA_EQUIPAMENTO ADD CONSTRAINT FK_AULA_EQUIPAMENTO_2
   FOREIGN KEY (fk_EQUIPAMENTO_cod_equip)
   REFERENCES EQUIPAMENTO (cod_equip);

ALTER TABLE ALUNO_AULA ADD CONSTRAINT FK_ALUNO_AULA_1
   FOREIGN KEY (fk_ALUNO_cod_membro)
   REFERENCES ALUNO (cod_membro);

ALTER TABLE ALUNO_AULA ADD CONSTRAINT FK_ALUNO_AULA_2
   FOREIGN KEY (fk_AULA_id_aula)
   REFERENCES AULA (id_aula);


SELECT * FROM ALUNO;
SELECT * FROM PAGAMENTO;

--


--INSERÇÕES NA TABELA PAGAMENTO

INSERT INTO PAGAMENTO VALUES (1,CURRENT_DATE,300.00,'CREDITO',3);
INSERT INTO PAGAMENTO VALUES (2, CURRENT_DATE, 300.00, 'CREDITO', 3);
```

```
INSERT INTO PAGAMENTO VALUES (3, CURRENT_DATE, 150.00, 'DEBITO', 6);
INSERT INTO PAGAMENTO VALUES (4, CURRENT_DATE, 200.00, 'DINHEIRO', 2);
INSERT INTO PAGAMENTO VALUES (5, CURRENT_DATE, 350.00, 'CREDITO', 4);
INSERT INTO PAGAMENTO VALUES (6, CURRENT_DATE, 400.00, 'DEBITO', 5);
INSERT INTO PAGAMENTO VALUES (7, CURRENT_DATE, 250.00, 'DINHEIRO', 1);
INSERT INTO PAGAMENTO VALUES (8, CURRENT_DATE, 300.00, 'CREDITO', 3);
INSERT INTO PAGAMENTO VALUES (9, CURRENT_DATE, 150.00, 'DEBITO', 6);
INSERT INTO PAGAMENTO VALUES (10, CURRENT_DATE, 200.00, 'DINHEIRO', 2);
INSERT INTO PAGAMENTO VALUES (11, CURRENT_DATE, 350.00, 'CREDITO', 4);


--INSERÇÕES NA TABELA ALUNO

INSERT INTO ALUNO VALUES (1, 'Willie Nelson', '1999-09-14', '2024-09-14',
'Willienelson@gmail.com', 123, 'Francisco Neto', 'Mangabeira', 1);
INSERT INTO ALUNO VALUES (2, 'John Doe', '2000-01-01', '2024-09-14',
'johndoe@example.com', 124, 'Rua A', 'Bairro B', 2);
INSERT INTO ALUNO VALUES (3, 'Jane Smith', '1998-05-23', '2024-09-14',
'janesmith@example.com', 125, 'Rua C', 'Bairro D', 3);
INSERT INTO ALUNO VALUES (4, 'Alice Johnson', '2001-07-12', '2024-09-14',
'alicejohnson@example.com', 126, 'Rua E', 'Bairro F', 4);
INSERT INTO ALUNO VALUES (5, 'Bob Brown', '1997-11-30', '2024-09-14',
'bobbrown@example.com', 127, 'Rua G', 'Bairro H', 5);
INSERT INTO ALUNO VALUES (6, 'Charlie Davis', '1996-03-15', '2024-09-14',
'charliedavis@example.com', 128, 'Rua I', 'Bairro J', 6);
INSERT INTO ALUNO VALUES (7, 'Diana Evans', '1995-08-25', '2024-09-14',
'dianaevans@example.com', 129, 'Rua K', 'Bairro L', 7);
INSERT INTO ALUNO VALUES (8, 'Ethan Harris', '1994-12-05', '2024-09-14',
'ethanharris@example.com', 130, 'Rua M', 'Bairro N', 8);
INSERT INTO ALUNO VALUES (9, 'Fiona Green', '1993-04-18', '2024-09-14',
'fionagreen@example.com', 131, 'Rua O', 'Bairro P', 9);
INSERT INTO ALUNO VALUES (10, 'George White', '1992-10-10', '2024-09-14',
'georgewhite@example.com', 132, 'Rua Q', 'Bairro R', 10);

--INSERÇÕES NA TABELA AULA

INSERT INTO AULA (id_aula, nome, descr, duraçao, horario) VALUES (11, 'Musculação -
Peitoral', 'Treino de musculação focado no peitoral', '01:00:00', '2024-09-15 08:00:00');
INSERT INTO AULA (id_aula, nome, descr, duraçao, horario) VALUES (12, 'Musculação -
Costas', 'Treino de musculação focado nas costas', '01:00:00', '2024-09-15 09:00:00');
INSERT INTO AULA (id_aula, nome, descr, duraçao, horario) VALUES (13, 'Musculação -
Pernas', 'Treino de musculação focado nas pernas', '01:00:00', '2024-09-15 10:00:00');
INSERT INTO AULA (id_aula, nome, descr, duraçao, horario) VALUES (14, 'Musculação -
Ombros', 'Treino de musculação focado nos ombros', '01:00:00', '2024-09-15 11:00:00');
INSERT INTO AULA (id_aula, nome, descr, duraçao, horario) VALUES (15, 'Musculação -
Braços', 'Treino de musculação focado nos braços', '01:00:00', '2024-09-15 12:00:00');
INSERT INTO AULA (id_aula, nome, descr, duraçao, horario) VALUES (16, 'Musculação -
Abdômen', 'Treino de musculação focado no abdômen', '01:00:00', '2024-09-15 13:00:00');
```

```
INSERT INTO AULA (id_aula, nome, descr, duraçao, horario) VALUES (17, 'Musculação - Corpo Inteiro', 'Treino de musculação para o corpo inteiro', '01:30:00', '2024-09-15 14:00:00');
INSERT INTO AULA (id_aula, nome, descr, duraçao, horario) VALUES (18, 'Musculação - Glúteos', 'Treino de musculação focado nos glúteos', '01:00:00', '2024-09-15 15:00:00');
INSERT INTO AULA (id_aula, nome, descr, duraçao, horario) VALUES (19, 'Musculação - Resistência', 'Treino de musculação para resistência muscular', '01:00:00', '2024-09-15 16:00:00');
INSERT INTO AULA (id_aula, nome, descr, duraçao, horario) VALUES (20, 'Musculação - Hipertrofia', 'Treino de musculação para hipertrofia', '01:00:00', '2024-09-15 17:00:00');



SELECT * FROM AULA;

-- INSERÇÕES NA TABELA INSTRUTOR

INSERT INTO INSTRUTOR (id_ins, nome, email, estilo_dança, tipo) VALUES (1, 'Carlos Silva', 'carlos.silva@example.com', 'Salsa', 'Professor');
INSERT INTO INSTRUTOR (id_ins, nome, email, estilo_dança, tipo) VALUES (2, 'Ana Souza', 'ana.souza@example.com', 'Ballet', 'Instrutora');
INSERT INTO INSTRUTOR (id_ins, nome, email, estilo_dança, tipo) VALUES (3, 'Marcos Pereira', 'marcos.pereira@example.com', 'Hip Hop', 'Professor');
INSERT INTO INSTRUTOR (id_ins, nome, email, estilo_dança, tipo) VALUES (4, 'Julia Costa', 'julia.costa@example.com', 'Jazz', 'Instrutora');
INSERT INTO INSTRUTOR (id_ins, nome, email, estilo_dança, tipo) VALUES (5, 'Pedro Lima', 'pedro.lima@example.com', 'Tango', 'Professor');
INSERT INTO INSTRUTOR (id_ins, nome, email, estilo_dança, tipo) VALUES (6, 'Fernanda Oliveira', 'fernanda.oliveira@example.com', 'Samba', 'Instrutora');
INSERT INTO INSTRUTOR (id_ins, nome, email, estilo_dança, tipo) VALUES (7, 'Rafael Santos', 'rafael.santos@example.com', 'Forró', 'Professor');
INSERT INTO INSTRUTOR (id_ins, nome, email, estilo_dança, tipo) VALUES (8, 'Beatriz Almeida', 'beatriz.almeida@example.com', 'Zouk', 'Instrutora');
INSERT INTO INSTRUTOR (id_ins, nome, email, estilo_dança, tipo) VALUES (9, 'Lucas Rocha', 'lucas.rocha@example.com', 'Bolero', 'Professor');
INSERT INTO INSTRUTOR (id_ins, nome, email, estilo_dança, tipo) VALUES (10, 'Mariana Ribeiro', 'mariana.ribeiro@example.com', 'Flamenco', 'Instrutora');

SELECT * FROM INSTRUTOR;

-- INSERÇÕES NA TABELA EQUIPAMENTO

INSERT INTO EQUIPAMENTO (cod_equip, qtde_dispo, descr, nome) VALUES (1, 10, 'Halteres de 5kg', 'Halteres');
INSERT INTO EQUIPAMENTO (cod_equip, qtde_dispo, descr, nome) VALUES (2, 5, 'Tapetes de Yoga', 'Tapete de Yoga');
INSERT INTO EQUIPAMENTO (cod_equip, qtde_dispo, descr, nome) VALUES (3, 8, 'Bolas de Pilates', 'Bola de Pilates');
```

```
INSERT INTO EQUIPAMENTO (cod_equip, qtde_dispo, descr, nome) VALUES (4, 12, 'Elásticos de Resistência', 'Elástico de Resistência');
INSERT INTO EQUIPAMENTO (cod_equip, qtde_dispo, descr, nome) VALUES (5, 7, 'Kettlebells de 10kg', 'Kettlebell');
INSERT INTO EQUIPAMENTO (cod_equip, qtde_dispo, descr, nome) VALUES (6, 15, 'Colchonetes', 'Colchonete');
INSERT INTO EQUIPAMENTO (cod_equip, qtde_dispo, descr, nome) VALUES (7, 20, 'Caneleiras de Peso', 'Caneleira de Peso');
INSERT INTO EQUIPAMENTO (cod_equip, qtde_dispo, descr, nome) VALUES (8, 6, 'Rolos de Espuma', 'Rolo de Espuma');
INSERT INTO EQUIPAMENTO (cod_equip, qtde_dispo, descr, nome) VALUES (9, 9, 'Barras de Exercício', 'Barra de Exercício');
INSERT INTO EQUIPAMENTO (cod_equip, qtde_dispo, descr, nome) VALUES (10, 4, 'Bicicletas Ergométricas', 'Bicicleta Ergométrica');

SELECT * FROM EQUIPAMENTO;

-- ASSOCIANDO TELEFONE_ALUNO

INSERT INTO TELEFONE_ALUNO (tel_PK, tel, fk_ALUNO_cod_membro_PK) VALUES (1, '1234567890', 1);
INSERT INTO TELEFONE_ALUNO (tel_PK, tel, fk_ALUNO_cod_membro_PK) VALUES (2, '0987654321', 2);
INSERT INTO TELEFONE_ALUNO (tel_PK, tel, fk_ALUNO_cod_membro_PK) VALUES (3, '1122334455', 3);
INSERT INTO TELEFONE_ALUNO (tel_PK, tel, fk_ALUNO_cod_membro_PK) VALUES (4, '2233445566', 4);
INSERT INTO TELEFONE_ALUNO (tel_PK, tel, fk_ALUNO_cod_membro_PK) VALUES (5, '3344556677', 5);
INSERT INTO TELEFONE_ALUNO (tel_PK, tel, fk_ALUNO_cod_membro_PK) VALUES (6, '4455667788', 6);
INSERT INTO TELEFONE_ALUNO (tel_PK, tel, fk_ALUNO_cod_membro_PK) VALUES (7, '5566778899', 7);
INSERT INTO TELEFONE_ALUNO (tel_PK, tel, fk_ALUNO_cod_membro_PK) VALUES (8, '6677889900', 8);
INSERT INTO TELEFONE_ALUNO (tel_PK, tel, fk_ALUNO_cod_membro_PK) VALUES (9, '7788990011', 9);
INSERT INTO TELEFONE_ALUNO (tel_PK, tel, fk_ALUNO_cod_membro_PK) VALUES (10, '8899001122', 10);



SELECT * FROM ALUNO;
SELECT * FROM TELEFONE_ALUNO;

--INSERÇÕES NA TABELA ESPECIALIZACAO
```

```
INSERT INTO ESPECIALIZACAO (especialização_PK, nome_especialização) VALUES (1, 'Salsa');
INSERT INTO ESPECIALIZACAO (especialização_PK, nome_especialização) VALUES (2, 'Ballet');
INSERT INTO ESPECIALIZACAO (especialização_PK, nome_especialização) VALUES (3, 'Hip Hop');
INSERT INTO ESPECIALIZACAO (especialização_PK, nome_especialização) VALUES (4, 'Jazz');
INSERT INTO ESPECIALIZACAO (especialização_PK, nome_especialização) VALUES (5, 'Tango');
INSERT INTO ESPECIALIZACAO (especialização_PK, nome_especialização) VALUES (6, 'Samba');
INSERT INTO ESPECIALIZACAO (especialização_PK, nome_especialização) VALUES (7, 'Forró');
INSERT INTO ESPECIALIZACAO (especialização_PK, nome_especialização) VALUES (8, 'Zouk');
INSERT INTO ESPECIALIZACAO (especialização_PK, nome_especialização) VALUES (9, 'Bolero');
INSERT INTO ESPECIALIZACAO (especialização_PK, nome_especialização) VALUES (10, 'Flamenco');

SELECT * FROM ESPECIALIZACAO;

--ASSOCIANDO TELEFONE_INSTRUTOR

INSERT INTO TELEFONE_INSTRUTOR (tel_PK, tel, fk_INSTRUTOR_id_ins_PK) VALUES (1, '1234567890', 1);
INSERT INTO TELEFONE_INSTRUTOR (tel_PK, tel, fk_INSTRUTOR_id_ins_PK) VALUES (2, '0987654321', 2);
INSERT INTO TELEFONE_INSTRUTOR (tel_PK, tel, fk_INSTRUTOR_id_ins_PK) VALUES (3, '1122334455', 3);
INSERT INTO TELEFONE_INSTRUTOR (tel_PK, tel, fk_INSTRUTOR_id_ins_PK) VALUES (4, '2233445566', 4);
INSERT INTO TELEFONE_INSTRUTOR (tel_PK, tel, fk_INSTRUTOR_id_ins_PK) VALUES (5, '3344556677', 5);
INSERT INTO TELEFONE_INSTRUTOR (tel_PK, tel, fk_INSTRUTOR_id_ins_PK) VALUES (6, '4455667788', 6);
INSERT INTO TELEFONE_INSTRUTOR (tel_PK, tel, fk_INSTRUTOR_id_ins_PK) VALUES (7, '5566778899', 7);
INSERT INTO TELEFONE_INSTRUTOR (tel_PK, tel, fk_INSTRUTOR_id_ins_PK) VALUES (8, '6677889900', 8);
INSERT INTO TELEFONE_INSTRUTOR (tel_PK, tel, fk_INSTRUTOR_id_ins_PK) VALUES (9, '7788990011', 9);
INSERT INTO TELEFONE_INSTRUTOR (tel_PK, tel, fk_INSTRUTOR_id_ins_PK) VALUES (10, '8899001122', 10);

SELECT * FROM TELEFONE_INSTRUTOR;
```

```
--ASSOCIANDO INSTRUTOR_AULA

SELECT * FROM AULA;
INSERT INTO INSTRUTOR_AULA (fk_INSTRUTOR_id_ins, fk_AULA_id_aula) VALUES (1,
11);
INSERT INTO INSTRUTOR_AULA (fk_INSTRUTOR_id_ins, fk_AULA_id_aula) VALUES (2,
12);
INSERT INTO INSTRUTOR_AULA (fk_INSTRUTOR_id_ins, fk_AULA_id_aula) VALUES (3,
13);
INSERT INTO INSTRUTOR_AULA (fk_INSTRUTOR_id_ins, fk_AULA_id_aula) VALUES (4,
14);
INSERT INTO INSTRUTOR_AULA (fk_INSTRUTOR_id_ins, fk_AULA_id_aula) VALUES (5,
15);
INSERT INTO INSTRUTOR_AULA (fk_INSTRUTOR_id_ins, fk_AULA_id_aula) VALUES (6,
16);
INSERT INTO INSTRUTOR_AULA (fk_INSTRUTOR_id_ins, fk_AULA_id_aula) VALUES (7,
17);
INSERT INTO INSTRUTOR_AULA (fk_INSTRUTOR_id_ins, fk_AULA_id_aula) VALUES (8,
18);
INSERT INTO INSTRUTOR_AULA (fk_INSTRUTOR_id_ins, fk_AULA_id_aula) VALUES (9,
19);
INSERT INTO INSTRUTOR_AULA (fk_INSTRUTOR_id_ins, fk_AULA_id_aula) VALUES
(10, 20);

SELECT * FROM INSTRUTOR_AULA;

SELECT * FROM EQUIPAMENTO;

--ASSOCIANDO AULA_EQUIPAMENTO

INSERT INTO AULA_EQUIPAMENTO (fk_AULA_id_aula, fk_EQUIPAMENTO_cod_equip)
VALUES (11, 1);
INSERT INTO AULA_EQUIPAMENTO (fk_AULA_id_aula, fk_EQUIPAMENTO_cod_equip)
VALUES (12, 2);
INSERT INTO AULA_EQUIPAMENTO (fk_AULA_id_aula, fk_EQUIPAMENTO_cod_equip)
VALUES (13, 3);
INSERT INTO AULA_EQUIPAMENTO (fk_AULA_id_aula, fk_EQUIPAMENTO_cod_equip)
VALUES (14, 4);
INSERT INTO AULA_EQUIPAMENTO (fk_AULA_id_aula, fk_EQUIPAMENTO_cod_equip)
VALUES (15, 5);
INSERT INTO AULA_EQUIPAMENTO (fk_AULA_id_aula, fk_EQUIPAMENTO_cod_equip)
VALUES (16, 6);
INSERT INTO AULA_EQUIPAMENTO (fk_AULA_id_aula, fk_EQUIPAMENTO_cod_equip)
VALUES (17, 7);
INSERT INTO AULA_EQUIPAMENTO (fk_AULA_id_aula, fk_EQUIPAMENTO_cod_equip)
VALUES (18, 8);
INSERT INTO AULA_EQUIPAMENTO (fk_AULA_id_aula, fk_EQUIPAMENTO_cod_equip)
VALUES (19, 9);
```

```
INSERT INTO AULA_EQUIPAMENTO (fk_AULA_id_aula, fk_EQUIPAMENTO_cod_equip)
VALUES (20, 10);



SELECT * FROM ALUNO;
SELECT * FROM AULA;
SELECT * FROM EQUIPAMENTO;
SELECT * FROM AULA_EQUIPAMENTO;
SELECT * FROM ALUNO_AULA;


--ASSOCIANDO ALUNO A AULA

INSERT INTO ALUNO_AULA (fk_ALUNO_cod_membro, fk_AULA_id_aula) VALUES (1,
11);
INSERT INTO ALUNO_AULA (fk_ALUNO_cod_membro, fk_AULA_id_aula) VALUES (2,
12);
INSERT INTO ALUNO_AULA (fk_ALUNO_cod_membro, fk_AULA_id_aula) VALUES (3,
13);
INSERT INTO ALUNO_AULA (fk_ALUNO_cod_membro, fk_AULA_id_aula) VALUES (4,
14);
INSERT INTO ALUNO_AULA (fk_ALUNO_cod_membro, fk_AULA_id_aula) VALUES (5,
15);
INSERT INTO ALUNO_AULA (fk_ALUNO_cod_membro, fk_AULA_id_aula) VALUES (6,
16);
INSERT INTO ALUNO_AULA (fk_ALUNO_cod_membro, fk_AULA_id_aula) VALUES (7,
17);
INSERT INTO ALUNO_AULA (fk_ALUNO_cod_membro, fk_AULA_id_aula) VALUES (8,
18);
INSERT INTO ALUNO_AULA (fk_ALUNO_cod_membro, fk_AULA_id_aula) VALUES (9,
19);
INSERT INTO ALUNO_AULA (fk_ALUNO_cod_membro, fk_AULA_id_aula) VALUES (10,
20);


/*
• 1 consulta com uma tabela usando operadores básicos de filtro (e.g., IN,
between, is null, etc).
*/

SELECT NOME, DATA_NASC
FROM ALUNO
WHERE DATA_NASC BETWEEN '2000-01-01' AND '2024-01-01';

/*
• 3 consultas com inner JOIN na cláusula FROM (pode ser self join, caso o
```

```
domínio indique esse uso).
*/

--INNER JOIN  ---> Tabelas envolvidas: AULA (A),AULA_EQUIPAMENTO
(E),EQUIPAMENTO (Q),ALUNO_AULA (K),ALUNO (N).
--RETORNA NOME DA AULA, NOME DO ALUNO E A DESCRIÇÃO DO EQUIPAMENTO

SELECT
  A.nome AS Nome_Aula,
  N.nome AS Nome_Aluno,
  Q.descr AS Descricao_Equipamento
FROM
   AULA A
JOIN
   AULA_EQUIPAMENTO E ON A.id_aula = E.fk_AULA_id_aula
JOIN
   EQUIPAMENTO Q ON Q.cod_equip = E.fk_EQUIPAMENTO_cod_equip
JOIN
   ALUNO_AULA K ON K.fk_AULA_id_aula = A.id_aula
JOIN
   ALUNO N ON N.cod_membro = K.fk_ALUNO_cod_membro;


---Inner Join--tabelas envolvidas: ALUNO (A),ALUNO_AULA (AA), AULA (B)
--Consulta para obter o nome dos alunos,o numero, o email e a descrição da aula
SELECT A.nome AS aluno_nome, A.email AS aluno_email, A.numero AS aluno_numero,
      B.nome AS aula_nome, B.descr AS aula_descricao
FROM
         ALUNO A
INNER JOIN
         ALUNO_AULA AA ON A.cod_membro = AA.fk_ALUNO_cod_membro
INNER JOIN
         AULA B ON AA.fk_AULA_id_aula = B.id_aula;

--inner join-- tabelas envolvidas: ALUNO(A),TELEFONE_ALUNO(F)
--CONSULTA PARA OBTER NOMES DE ALUNOS E TELEFONES
SELECT A.NOME, F.TEL
FROM TELEFONE_ALUNO F JOIN ALUNO A
ON F.FK_ALUNO_COD_MEMBRO_PK = A.COD_MEMBRO;

--• 1 consulta com left/right/full outer join na cláusula FROM


--tabelas envolvidas: ALUNO_AULA (AA),AULA (B),ALUNO (A).
--Consulta para obter o nome dos alunos,o numero, o email e a descrição da aula right outer
join e left join

SELECT A.nome AS aluno_nome, A.email AS aluno_email, A.numero AS aluno_numero,
```

```
B.nome AS aula_nome, B.descr AS aula_descricao
FROM ALUNO_AULA AA
RIGHT OUTER JOIN AULA B ON AA.fk_AULA_id_aula = B.id_aula
LEFT JOIN ALUNO A ON AA.fk_ALUNO_cod_membro = A.cod_membro;




--• 2 consultas usando Group By (e possivelmente o having)

--consulta 1-numero de alunos por aula agrupados por aula_nome
--tabelas envolvidas:AULA (A),ALUNO_AULA (AA)
SELECT A.nome AS aula_nome, COUNT(DISTINCT AA.fk_ALUNO_cod_membro) AS
num_alunos
FROM AULA A
LEFT JOIN ALUNO_AULA AA ON A.id_aula = AA.fk_AULA_id_aula
GROUP BY A.nome;

--consulta 2- total de aulas dos alunos, com a condição de que o total de aulas seja maior
que 0
--agrupado pelo nomes do alunos
--tabelas envolvidas: ALUNO (A),ALUNO_AULA (AA)
SELECT A.nome AS aluno_nome, COUNT(AA.fk_AULA_id_aula) AS total_aulas
FROM ALUNO A
INNER JOIN ALUNO_AULA AA ON A.cod_membro = AA.fk_ALUNO_cod_membro
GROUP BY A.nome
HAVING COUNT(AA.fk_AULA_id_aula)>0;




--• 1 consulta usando alguma operação de conjunto (union, except ou
--intersect)
--Retorna uma lista de todos os alunos e os instrutores, combinando duas consultas em
uma única exibição.
SELECT nome AS pessoa, 'Aluno' AS tipo
FROM ALUNO
UNION
SELECT nome AS pessoa, 'Instrutor' AS tipo
FROM INSTRUTOR;


--• 2 consultas que usem subqueries.

--1--Listando as aulas que um aluno,nesse caso, Bob Brown, frequentaa.
--tabelas envolvidas:ALUNO (A), ALUNO_AULA (AA),AULA (B)
```

```
SELECT A.nome AS nome_aluno, B.nome AS aula_nome, B.descr AS aula_descricao
FROM AULA B, ALUNO A -- Join implicito(lista multiplas tabelas após o from)
WHERE A.nome = 'Bob Brown' AND B.id_aula IN (SELECT AA.fk_AULA_id_aula
                FROM ALUNO_AULA AA
                WHERE AA.fk_ALUNO_cod_membro = A.cod_membro);


--2--Equipamentos Utilizados por Alunos Específicos
--tableas utilizadas:
Equipamento(E),AULA_EQUIPAMENTO(AE),ALUNO_AULA(AA),ALUNO(A)
SELECT E.nome AS nome_equipamento
FROM EQUIPAMENTO E
WHERE E.cod_equip IN (
   SELECT AE.fk_EQUIPAMENTO_cod_equip
   FROM AULA_EQUIPAMENTO AE
   WHERE AE.fk_AULA_id_aula IN (
      SELECT AA.fk_AULA_id_aula
      FROM ALUNO_AULA AA
      INNER JOIN ALUNO A ON AA.fk_ALUNO_cod_membro = A.cod_membro
      WHERE A.nome = 'Bob Brown'
   )
);

select * from equipamento;




--b. Visões--
--visão que permita inserção
-- visão que permita inserção de novos alunos
CREATE OR REPLACE VIEW vw_aluno_insercao AS
SELECT cod_membro, nome, email, numero
FROM ALUNO;
--inserindo na view
INSERT INTO vw_aluno_insercao (cod_membro, nome, email, numero)
VALUES (123, 'thiago', 'thiago@example.com', '99887766');
--• 2 visões robustas (e.g., com vários joins) com justificativa semântica, de acordo com os
--requisitos da aplicação.

--Combina dados pessoais dos alunos, informações de pagamento e aulas frequentadas.

-- Facilita a gestão e análise de dados dos alunos para relatórios e decisões.

--tabelas utilizadas: ALUNO(A),PAGAMENTO(P),ALUNO_AULA(AA),AULA(AU)
CREATE VIEW V_ALUNO_DETALHES AS
SELECT
   A.cod_membro,
```

```
    A.nome AS nome_aluno,
    A.data_nasc,
    A.email,
    A.numero,
    A.rua,
    A.bairro,
    P.valor AS valor_pagamento,
    P.data_pag,
    P.metod_pag,
    AU.nome AS nome_aula,
    AU.descr AS descricao_aula,
    AU.horario
FROM
    ALUNO A
JOIN
    PAGAMENTO P ON A.FK_PAGAMENTO_cod_pag = P.cod_pag
JOIN
    ALUNO_AULA AA ON A.cod_membro = AA.fk_ALUNO_cod_membro
JOIN
    AULA AU ON AA.fk_AULA_id_aula = AU.id_aula;

SELECT * FROM V_ALUNO_DETALHES;

SELECT * FROM AULA;



/*
Combina informações sobre aulas, instrutores e equipamentos.

Facilita a coordenação e planejamento das aulas, garantindo
a disponibilidade de recursos e a alocação correta dos instrutores.
*/

--TABELAS UTILIZADAS: AULA (AU),INSTRUTOR_AULA(IA),INSTRUTOR
(I),AULA_EQUIPAMENTO(AE),EQUIPAMENTO(E)
CREATE VIEW V_AULA_INSTRUTOR_EQUIPAMENTO AS
SELECT
    AU.id_aula,
    AU.nome AS nome_aula,
    AU.descr AS descricao_aula,
    AU.duraçao,
    AU.horario,
    I.nome AS nome_instrutor,
    I.email AS email_instrutor,
    I.estilo_dança,
    E.nome AS nome_equipamento,
    E.descr AS descricao_equipamento,
```

```
   E.qtde_dispo
FROM
   AULA AU
JOIN
   INSTRUTOR_AULA IA ON AU.id_aula = IA.fk_AULA_id_aula
JOIN
   INSTRUTOR I ON IA.fk_INSTRUTOR_id_ins = I.id_ins
JOIN
   AULA_EQUIPAMENTO AE ON AU.id_aula = AE.fk_AULA_id_aula
JOIN
   EQUIPAMENTO E ON AE.fk_EQUIPAMENTO_cod_equip = E.cod_equip;

SELECT * FROM V_AULA_INSTRUTOR_EQUIPAMENTO;

SELECT * FROM AULA;
SELECT * FROM INSTRUTOR;
SELECT * FROM INSTRUTOR_AULA;
SELECT * FROM AULA_EQUIPAMENTO;
SELECT * FROM EQUIPAMENTO;


--c.CRIAÇÃO DE INDICES
--Esse índice ajudará a acelerar todas as consultas que fazem JOIN entre a tabela ALUNO e outras tabelas, como ALUNO_AULA.
CREATE INDEX idx_aluno_cod_membro ON ALUNO(cod_membro);

--Melhora o desempenho das consultas relacionadas a aulas que os alunos estão frequentando.
CREATE INDEX idx_aluno_aula_fk_aula ON ALUNO_AULA(fk_AULA_id_aula);

--Acelera a busca por nome de aluno nas consultas que filtram alunos específicos.
CREATE INDEX idx_aluno_nome ON ALUNO(nome);



--d.Reescrita de consultas
--1.consulta Nome do Aluno e nome da aula reescrita de forma mais eficiente,foram alterados:
--Uso de JOIN explícito para melhorar a legibilidade e o desempenho da consulta
--Remoção da subconsulta, pois foi utilizado o join diretamente
--A nova versão faz a busca de uma forma mais direta e clara, tornando-a mais eficiente.
--NOVA VERSÃO
SELECT A.nome AS nome_aluno, B.nome AS aula_nome, B.descr AS aula_descricao
FROM ALUNO A
JOIN ALUNO_AULA AA ON A.cod_membro = AA.fk_ALUNO_cod_membro
JOIN AULA B ON AA.fk_AULA_id_aula = B.id_aula
WHERE A.nome = 'John Doe';
```

```
SELECT * FROM ALUNO;
--d.reescrita de consultas
--2.Total de Aulas dos Alunos
--Remoção do HAVING: A condição HAVING era desnecessária, já que a junção
--com ALUNO_AULA só retorna alunos que estão associados a pelo menos uma aula. Isso
reduz a complexidade da consulta.
--Eficiência: A eliminação do HAVING reduz o processamento necessário durante a
execução da consulta
--NOVA VERSÃO
SELECT A.nome AS aluno_nome, COUNT(AA.fk_AULA_id_aula) AS total_aulas
FROM ALUNO A
INNER JOIN ALUNO_AULA AA ON A.cod_membro = AA.fk_ALUNO_cod_membro
GROUP BY A.nome;



--e. Funções e procedures armazenadas

-- funcao para contar total de pagamentos(utilizando SUM)

CREATE OR REPLACE FUNCTION total_valor_pagamentos()
RETURNS NUMERIC AS $$
DECLARE
   total_valor NUMERIC(10,2);
BEGIN
   SELECT SUM(valor) INTO total_valor
   FROM PAGAMENTO;

   RETURN total_valor;
END;
$$ LANGUAGE plpgsql;

SELECT  total_valor_pagamentos();


--Outras 2 funções com justificativa semântica, conforme os requisitos da aplicação


-- funcao 1-- contar o total de alunos


CREATE OR REPLACE FUNCTION contar_total_alunos()
RETURNS INTEGER AS $$
DECLARE
   total_alunos INTEGER;
BEGIN
   SELECT COUNT(*) INTO total_alunos
   FROM ALUNO;
```

```
    RETURN total_alunos;
END;
$$ LANGUAGE plpgsql;

SELECT contar_total_alunos();


-- Função 2 -- contar o número de alunos em uma aula específica

CREATE OR REPLACE FUNCTION contar_alunos_aula(id_aula INTEGER)
RETURNS INTEGER AS $$
DECLARE
    total_alunos INTEGER;
BEGIN
    SELECT COUNT(*) INTO total_alunos
    FROM ALUNO_AULA
    WHERE fk_AULA_id_aula = contar_alunos_aula.id_aula;

    RETURN total_alunos;
END;
$$ LANGUAGE plpgsql;




select contar_alunos_aula(10);

-- PROCEDURE PARA INSERIR USUARIOS



CREATE OR REPLACE PROCEDURE atualizar_email_aluno(
    p_cod_membro INTEGER,
    p_novo_email VARCHAR
)
LANGUAGE plpgsql
AS $$
BEGIN
    BEGIN
        -- Tenta atualizar o email do aluno
        UPDATE ALUNO
        SET email = p_novo_email
        WHERE cod_membro = p_cod_membro;

        -- Verifica se a atualização foi bem-sucedida
        IF NOT FOUND THEN
            RAISE EXCEPTION 'Aluno com cod_membro % não encontrado', p_cod_membro;
```

```
    END IF;
  EXCEPTION
    WHEN OTHERS THEN
      -- Retorna uma mensagem de erro personalizada
      RAISE EXCEPTION 'Erro ao atualizar email: %', SQLERRM;
  END;
END;
$$;


CALL atualizar_email_aluno(6, 'novo2.email@example.com');

CALL atualizar_email_aluno(125, 'novo.email@example.com');

SELECT * FROM ALUNO;
SELECT * FROM PAGAMENTO;
/*
• f.3 diferentes triggers com justificativa semântica, de acordo com os requisitos da
aplicação.
*/

--TRIGGER 1- ADICIONA A TABELA ALUNO_OFF OS ALUNOS QUE FORAM
EXCLUIDOS DA TABELA

CREATE TABLE ALUNO_OFF (
        COD_MEMBRO INTEGER PRIMARY KEY,
        NOME VARCHAR (30),
        DATA_DESLIGAMENTO DATE
);
CREATE OR REPLACE FUNCTION ATT_DESLIGAMENTO () RETURNS TRIGGER
AS $$
BEGIN
        INSERT INTO ALUNO_OFF VALUES (OLD.COD_MEMBRO,OLD.NOME,
CURRENT_DATE);
        RETURN OLD;
END;
$$ LANGUAGE PLPGSQL;

CREATE TRIGGER REMOVE_ALUNO
AFTER DELETE ON ALUNO
FOR EACH ROW
EXECUTE FUNCTION ATT_DESLIGAMENTO ();




DELETE FROM ALUNO
WHERE COD_MEMBRO = 5;
```

```
SELECT * FROM ALUNO_OFF;



-- TRIGGER 2-Trigger para verificar se a data de nascimento é valida


CREATE TRIGGER verificar_data_nascimento_aluno
BEFORE INSERT OR UPDATE ON ALUNO
FOR EACH ROW
EXECUTE FUNCTION VERIFICA_NASC ();

CREATE OR REPLACE FUNCTION VERIFICA_NASC () RETURNS TRIGGER
AS $$
BEGIN
   IF NEW.data_nasc >= CURRENT_DATE THEN
      RAISE EXCEPTION 'A data de nascimento não pode ser no futuro.';
   END IF;
         RETURN NEW;
END;
$$ LANGUAGE PLPGSQL;


INSERT INTO ALUNO VALUES (55, 'Willie Nelson', '2030-09-14', '2024-09-14',
'Willienelson@gmail.com', 123, 'Francisco Neto', 'Mangabeira', 1);


--TRIGGER 3

CREATE TABLE HISTORICO_ALUNO (
   id_hist SERIAL PRIMARY KEY,
   cod_membro INTEGER,
   nome VARCHAR(100),
   data_nasc DATE,
   data_ins DATE,
   email VARCHAR(30),
   numero INTEGER,
   rua VARCHAR(15),
   bairro VARCHAR(15),
   FK_PAGAMENTO_cod_pag INTEGER,
   data_alteracao TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
--O gatilho serve para manter um histórico das atualizações feitas na tabela ALUNO.
CREATE OR REPLACE FUNCTION  ATUALIZA_INFO() RETURNS TRIGGER
AS $$
BEGIN
   INSERT INTO HISTORICO_ALUNO (
```

```
        cod_membro, nome, data_nasc, data_ins, email, numero, rua, bairro,
FK_PAGAMENTO_cod_pag
    ) VALUES (
        NEW.cod_membro, NEW.nome, NEW.data_nasc, NEW.data_ins, NEW.email,
NEW.numero, NEW.rua, NEW.bairro, NEW.FK_PAGAMENTO_cod_pag
    );
            RETURN NEW;
END;
$$ LANGUAGE PLPGSQL;

CREATE TRIGGER trg_aluno_update
AFTER UPDATE ON ALUNO
FOR EACH ROW
EXECUTE PROCEDURE ATUALIZA_INFO();



SELECT * FROM HISTORICO_ALUNO;

UPDATE ALUNO
SET EMAIL = 'WILLIE@GMAIL.COM'
WHERE COD_MEMBRO = 1;
```