# Sistema de Bancos de Dados para Gestão Escolar

Beatryz Aparecida do Couto Medeiros de Freitas Carneiro
Iaggo Rauta Ramos de Lima
Gabriel Fonseca Amaro

Universidade Federal de Ouro Preto - UFOP
Banco de Dados I
Prof. Marcos Geraldo Braga Emiliano


Universidade Federal
de Ouro Preto

# Introdução

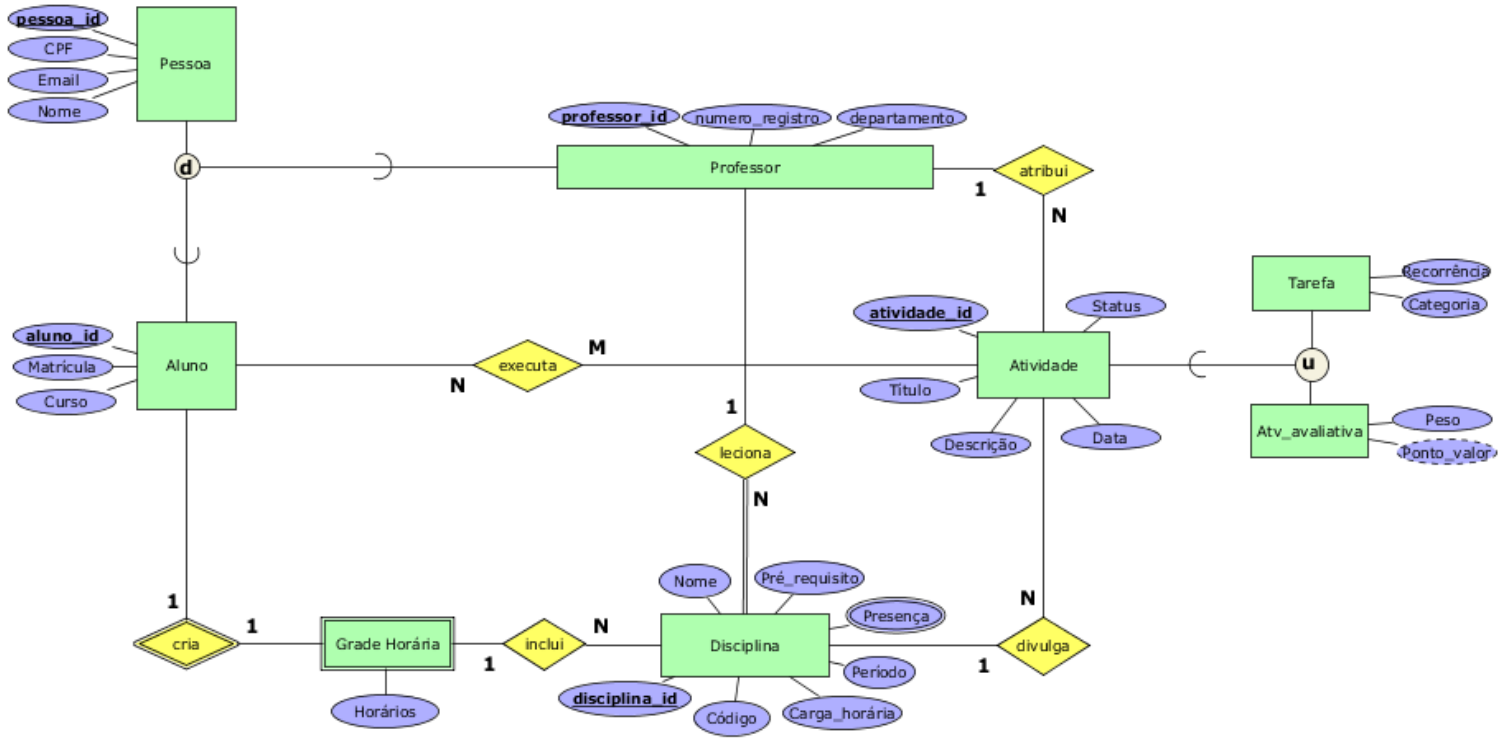
A proposta deste projeto é desenvolver um banco de dados voltado à gestão acadêmica dos alunos da UFOP, com foco na organização e otimização de informações essenciais da vida universitária.

O sistema centraliza dados de alunos, professores, disciplinas, atividades e notas, permitindo maior controle acadêmico e eficiência na administração.

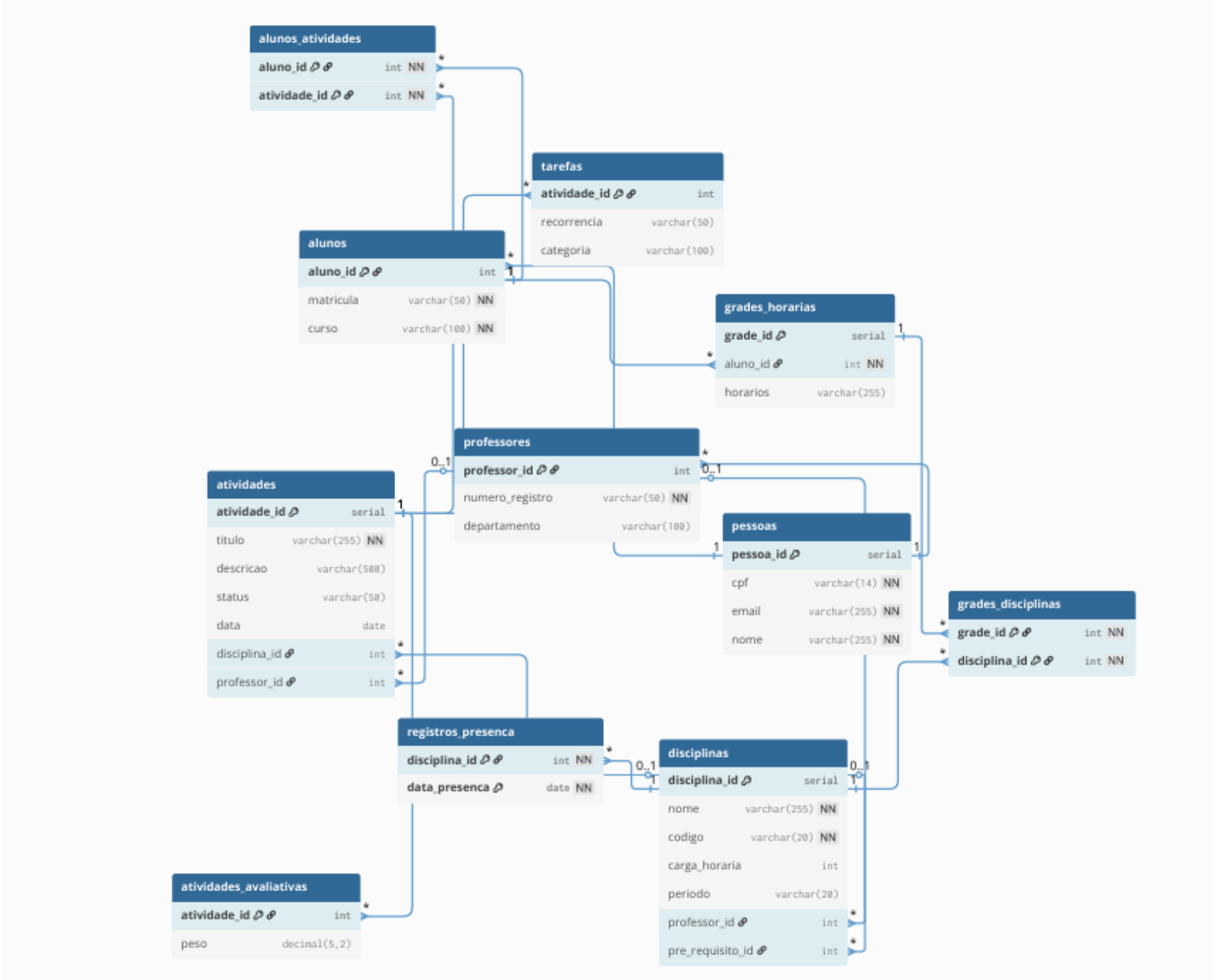
# Objetivos do Sistema

- Centralizar informações acadêmicas em uma única base de dados.
- Garantir integridade e consistência dos dados de alunos, professores e disciplinas.
- Facilitar o acompanhamento de atividades, tarefas e avaliações.
- Proporcionar consultas rápidas sobre desempenho e progresso acadêmico.
- Servir como base para relatórios institucionais.

# Modelo Entidade-Relacionamento



# Diagrama do Modelo Relacional



# Criação do Banco de Dados

```sql
DROP SCHEMA IF EXISTS academico CASCADE;
CREATE SCHEMA academico;

CREATE TABLE academico.pessoas (
    pessoa_id SERIAL NOT NULL,
    cpf VARCHAR(14) NOT NULL,
    email VARCHAR(255) NOT NULL,
    nome VARCHAR(255) NOT NULL,
    CONSTRAINT pk_pessoas PRIMARY KEY (pessoa_id),
    CONSTRAINT un_pessoas_cpf UNIQUE (cpf),
    CONSTRAINT un_pessoas_email UNIQUE (email)
);

CREATE TABLE academico.alunos (
    aluno_id INTEGER NOT NULL,
    matricula VARCHAR(50) NOT NULL,
    curso VARCHAR(100) NOT NULL,
    CONSTRAINT pk_alunos PRIMARY KEY (aluno_id),
    CONSTRAINT fk_alunos_pessoas FOREIGN KEY (aluno_id) REFERENCES academico.pessoas(pessoa_id) ON DELETE CASCADE,
    CONSTRAINT un_alunos_matricula UNIQUE (matricula)
);

CREATE TABLE academico.professores (
    professor_id INTEGER NOT NULL,
    numero_registro VARCHAR(50) NOT NULL,
    departamento VARCHAR(100),
    CONSTRAINT pk_professores PRIMARY KEY (professor_id),
    CONSTRAINT fk_professores_pessoas FOREIGN KEY (professor_id) REFERENCES academico.pessoas(pessoa_id) ON DELETE CASCADE,
    CONSTRAINT un_professores_registro UNIQUE (numero_registro)
);
```

# Criação do Banco de Dados

```sql
CREATE TABLE academico.grades_horarias (
    grade_id SERIAL NOT NULL,
    aluno_id INTEGER NOT NULL,
    horarios VARCHAR(255),
    CONSTRAINT pk_grades_horarias PRIMARY KEY (grade_id),
    CONSTRAINT fk_grades_alunos FOREIGN KEY (aluno_id) REFERENCES academico.alunos(aluno_id) ON DELETE CASCADE,
    CONSTRAINT un_grades_aluno UNIQUE (aluno_id)
);

CREATE TABLE academico.disciplinas (
    disciplina_id SERIAL NOT NULL,
    nome VARCHAR(255) NOT NULL,
    codigo VARCHAR(20) NOT NULL,
    carga_horaria INTEGER,
    periodo VARCHAR(20),
    professor_id INTEGER,
    pre_requisito_id INTEGER,
    CONSTRAINT pk_disciplinas PRIMARY KEY (disciplina_id),
    CONSTRAINT un_disciplinas_codigo UNIQUE (codigo)
);

CREATE TABLE academico.registros_presenca (
    disciplina_id INTEGER NOT NULL,
    data_presenca DATE NOT NULL,
    CONSTRAINT pk_registros_presenca PRIMARY KEY (disciplina_id, data_presenca)
);
```

# Criação do Banco de Dados

```sql
CREATE TABLE academico.grades_horarias (
    grade_id SERIAL NOT NULL,
    aluno_id INTEGER NOT NULL,
    horarios VARCHAR(255),
    CONSTRAINT pk_grades_horarias PRIMARY KEY (grade_id),
    CONSTRAINT fk_grades_alunos FOREIGN KEY (aluno_id) REFERENCES academico.alunos(aluno_id) ON DELETE CASCADE,
    CONSTRAINT un_grades_aluno UNIQUE (aluno_id)
);

CREATE TABLE academico.disciplinas (
    disciplina_id SERIAL NOT NULL,
    nome VARCHAR(255) NOT NULL,
    codigo VARCHAR(20) NOT NULL,
    carga_horaria INTEGER,
    periodo VARCHAR(20),
    professor_id INTEGER,
    pre_requisito_id INTEGER,
    CONSTRAINT pk_disciplinas PRIMARY KEY (disciplina_id),
    CONSTRAINT un_disciplinas_codigo UNIQUE (codigo)
);

CREATE TABLE academico.registros_presenca (
    disciplina_id INTEGER NOT NULL,
    data_presenca DATE NOT NULL,
    CONSTRAINT pk_registros_presenca PRIMARY KEY (disciplina_id, data_presenca)
);
```

# Criação do Banco de Dados

```sql
CREATE TABLE academico.atividades (
    atividade_id SERIAL NOT NULL,
    titulo VARCHAR(255) NOT NULL,
    descricao VARCHAR(500),
    status VARCHAR(50),
    data DATE,
    disciplina_id INTEGER,
    professor_id INTEGER,
    CONSTRAINT pk_atividades PRIMARY KEY (atividade_id)
);

CREATE TABLE academico.atividades_avaliativas (
    atividade_id INTEGER NOT NULL,
    peso DECIMAL(5, 2),
    CONSTRAINT pk_atv_avaliativas PRIMARY KEY (atividade_id),
    CONSTRAINT fk_atv_avaliativas_atividades FOREIGN KEY (atividade_id) REFERENCES academico.atividades(atividade_id) ON DELETE CASCADE
);

CREATE TABLE academico.tarefas (
    atividade_id INTEGER NOT NULL,
    recorrencia VARCHAR(50),
    categoria VARCHAR(100),
    CONSTRAINT pk_tarefas PRIMARY KEY (atividade_id),
    CONSTRAINT fk_tarefas_atividades FOREIGN KEY (atividade_id) REFERENCES academico.atividades(atividade_id) ON DELETE CASCADE
);
```

# Criação do Banco de Dados

```sql
CREATE TABLE academico.alunos_atividades (
    aluno_id INTEGER NOT NULL,
    atividade_id INTEGER NOT NULL,
    CONSTRAINT pk_alunos_atividades PRIMARY KEY (aluno_id, atividade_id)
);

CREATE TABLE academico.grades_disciplinas (
    grade_id INTEGER NOT NULL,
    disciplina_id INTEGER NOT NULL,
    CONSTRAINT pk_grades_disciplinas PRIMARY KEY (grade_id, disciplina_id)
);

ALTER TABLE academico.disciplinas ADD CONSTRAINT fk_disciplinas_professores
FOREIGN KEY (professor_id) REFERENCES academico.professores(professor_id) ON DELETE SET NULL;

ALTER TABLE academico.disciplinas ADD CONSTRAINT fk_disciplinas_prerequisito
FOREIGN KEY (pre_requisito_id) REFERENCES academico.disciplinas(disciplina_id) ON DELETE SET NULL;
```

# Criação do Banco de Dados

```
ALTER TABLE academico.registros_presenca ADD CONSTRAINT fk_presenca_disciplina
FOREIGN KEY (disciplina_id) REFERENCES academico.disciplinas(disciplina_id) ON DELETE CASCADE;

ALTER TABLE academico.atividades ADD CONSTRAINT fk_atividades_disciplinas
FOREIGN KEY (disciplina_id) REFERENCES academico.disciplinas(disciplina_id) ON DELETE CASCADE;

ALTER TABLE academico.atividades ADD CONSTRAINT fk_atividades_professores
FOREIGN KEY (professor_id) REFERENCES academico.professores(professor_id) ON DELETE SET NULL;

ALTER TABLE academico.alunos_atividades ADD CONSTRAINT fk_aa_aluno
FOREIGN KEY (aluno_id) REFERENCES academico.alunos(aluno_id) ON DELETE CASCADE;
ALTER TABLE academico.alunos_atividades ADD CONSTRAINT fk_aa_atividade
FOREIGN KEY (atividade_id) REFERENCES academico.atividades(atividade_id) ON DELETE CASCADE;

ALTER TABLE academico.grades_disciplinas ADD CONSTRAINT fk_gd_grade
FOREIGN KEY (grade_id) REFERENCES academico.grades_horarias(grade_id) ON DELETE CASCADE;

ALTER TABLE academico.grades_disciplinas ADD CONSTRAINT fk_gd_disciplina
FOREIGN KEY (disciplina_id) REFERENCES academico.disciplinas(disciplina_id) ON DELETE CASCADE;
```

# Entidades Principais

- **Pessoa**: informações básicas como nome, e-mail e CPF.
- **Aluno**: matrícula, curso, grade horária.
- **Professor**: registro, departamento e disciplinas lecionadas.
- **Disciplina**: código, nome, carga horária, pré-requisitos, período.
- **Atividade**: título, descrição, data, status.
- **Tarefa**: categoria, recorrência.
- **Atividade Avaliativa**: peso e pontuação atribuída.
- **Grade Horária**: horários de disciplinas e atividades.

# Relacionamentos

- **Pessoa** é generalizada em **Aluno** e **Professor**.
- **Professor** leciona **Disciplina**.
- **Professor** atribui **Atividade**.
- **Aluno** executa **Atividade**.
- **Aluno** cria sua **Grade Horária**.
- **Atividade** pode se especializar em **Tarefa** ou **Atividade Avaliativa**.
- **Disciplina** é divulgada por meio de **Atividades**.

# Cardinalidades

- Um professor pode lecionar várias disciplinas (1:N).
- Uma disciplina pertence a apenas um professor (N:1).
- Um aluno pode executar várias atividades, e uma atividade pode ser executada por vários alunos (N:M).
- Cada aluno possui apenas uma grade horária (1:1).
- Uma atividade pode se desdobrar em múltiplas tarefas ou avaliações (1:N).

# Cenários de Uso

- O aluno acessa sua grade horária, visualiza disciplinas e atividades atribuídas.
- O professor cadastra disciplinas e atribui atividades avaliativas.
- O sistema calcula a pontuação do aluno com base nas atividades concluídas.
- Consultas permitem verificar progresso, disciplinas pendentes e desempenho.

# Possíveis Consultas

- Listar todos os alunos com suas matrículas e cursos.
- Listar todas as disciplinas com seus professores responsáveis.
- Consultar todas as atividades atribuídas a um aluno específico.
- Consultar apenas as tarefas pendentes de um aluno.
- Consultar atividades avaliativas de uma disciplina e seus pesos.
- Listar disciplinas de um aluno pela grade horária.
- Gerar relatório de presença por disciplina.
- Calcular pontuação total de um aluno em cada disciplina.
- Ver disciplinas que exigem pré-requisito.

# Possíveis Consultas

```sql
-- 1. Listar todos os alunos com suas matrículas e cursos
SELECT a.aluno_id, p.nome, a.matricula, a.curso
FROM academico.alunos a
JOIN academico.pessoas p ON p.pessoa_id = a.aluno_id;

-- 2. Listar todas as disciplinas com seus professores responsáveis
SELECT d.disciplina_id, d.nome AS disciplina, pr.nome AS professor
FROM academico.disciplinas d
LEFT JOIN academico.professores pf ON pf.professor_id = d.professor_id
LEFT JOIN academico.pessoas pr ON pr.pessoa_id = pf.professor_id;

-- 3. Consultar todas as atividades atribuídas a um aluno específico
SELECT p.nome AS aluno, a.titulo, a.data, a.status
FROM academico.alunos_atividades aa
JOIN academico.alunos al ON al.aluno_id = aa.aluno_id
JOIN academico.pessoas p ON p.pessoa_id = al.aluno_id
JOIN academico.atividades a ON a.atividade_id = aa.atividade_id
WHERE aa.aluno_id = 1;
```

# Possíveis Consultas

```sql
-- 4. Consultar apenas as tarefas pendentes de um aluno
SELECT t.atividade_id, a.titulo, a.data, t.categoria
FROM academico.alunos_atividades aa
JOIN academico.atividades a ON a.atividade_id = aa.atividade_id
JOIN academico.tarefas t ON t.atividade_id = a.atividade_id
WHERE aa.aluno_id = 1 AND COALESCE(a.status,'') <> 'concluida';

-- 5. Consultar atividades avaliativas de uma disciplina e seus pesos
SELECT d.nome AS disciplina, a.titulo, av.peso
FROM academico.atividades_avaliativas av
JOIN academico.atividades a ON a.atividade_id = av.atividade_id
JOIN academico.disciplinas d ON d.disciplina_id = a.disciplina_id
WHERE d.codigo = 'BD101';

-- 6. Listar disciplinas de um aluno pela grade horária
SELECT p.nome AS aluno, d.nome AS disciplina, d.periodo
FROM academico.grades_horarias g
JOIN academico.grades_disciplinas gd ON gd.grade_id = g.grade_id
JOIN academico.disciplinas d ON d.disciplina_id = gd.disciplina_id
JOIN academico.alunos al ON al.aluno_id = g.aluno_id
JOIN academico.pessoas p ON p.pessoa_id = al.aluno_id
WHERE g.aluno_id = 1;
```

# Possíveis Consultas

```sql
-- 7. Relatório de presença por disciplina
SELECT d.nome AS disciplina, r.data_presenca
FROM academico.registros_presenca r
JOIN academico.disciplinas d ON d.disciplina_id = r.disciplina_id
ORDER BY d.nome, r.data_presenca;

-- 8. Calcular pontuação total de um aluno em cada disciplina
SELECT p.nome AS aluno, d.nome AS disciplina,
       SUM(av.peso) AS pontos_totais
FROM academico.alunos_atividades aa
JOIN academico.atividades a ON a.atividade_id = aa.atividade_id
JOIN academico.atividades_avaliativas av ON av.atividade_id = a.atividade_id
JOIN academico.disciplinas d ON d.disciplina_id = a.disciplina_id
JOIN academico.pessoas p ON p.pessoa_id = aa.aluno_id
WHERE aa.aluno_id = 1
GROUP BY p.nome, d.nome;

-- 9. Ver disciplinas que exigem pré-requisito
SELECT d.nome AS disciplina, dp.nome AS pre_requisito
FROM academico.disciplinas d
JOIN academico.disciplinas dp ON dp.disciplina_id = d.pre_requisito_id;
```

# Conclusão

O sistema desenvolvido busca centralizar e organizar as principais informações da vida acadêmica, tornando a gestão mais simples e eficiente para alunos e professores.

Além disso, a modelagem proposta garante integridade dos dados, facilita consultas importantes do dia a dia e abre espaço para futuras melhorias, como acompanhamento de notas finais, relatórios detalhados e integração com outros sistemas institucionais.

# Dúvidas?