**SISTEMAS DE CONTROLE ACADÊMICO – DOCUMENTO TÉCNICO**

**1. Introdução**

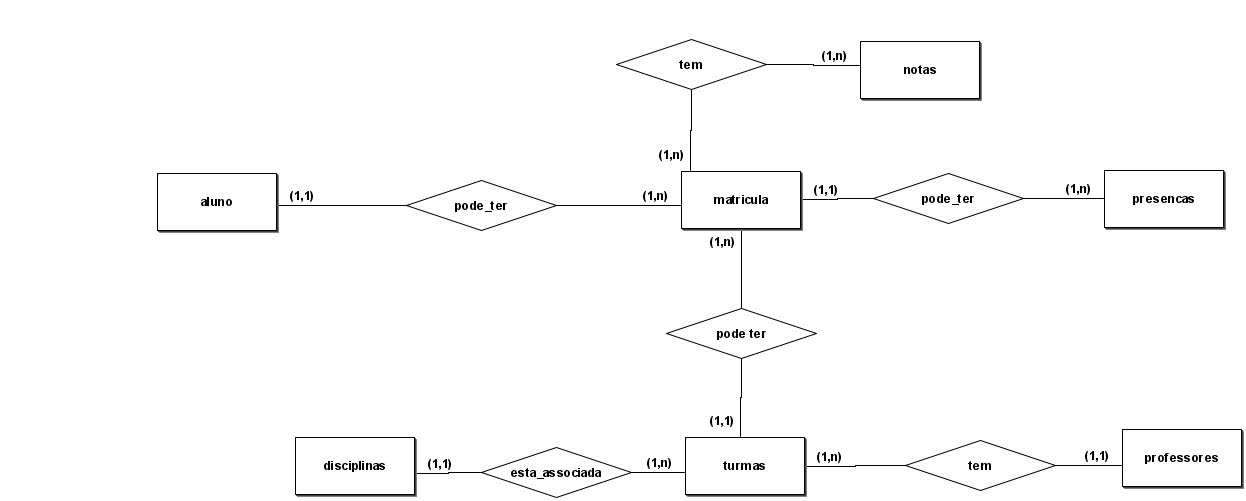
Este documento descreve o projeto de um sistema de controle acadêmico utilizando PostgreSQL. O sistema abrange cadastros de alunos, professores, disciplinas, turmas, matrículas, notas e presenças, além de rotinas avançadas como stored procedures, funções, triggers e otimizações com índices.

**2. Modelagem do Banco de Dados**

**TABELAS CAMPOS PRINCIPAIS RELAÇÕES**

| alunos | id, nome, data\_nascimento, email | Primária: id |
| --- | --- | --- |
| professores | id, nome, email | Primária: id |
| disciplinas | id,nome,carga horaria | Primária: id |
| turmas | id, disciplinas\_id, professores\_id, semestre | FK: disciplinas(id), professores(id) |
| matriculas | id, alunos\_id, turmas\_id, data\_matricula | FK: alunos(id), turmas(id) |
| notas | id, matricula\_id, avaliacao, nota | FK: matriculas(id) |
| presencas | id, matricula\_id, date\_aula, presente | FK: matriculas(id) |

**Diagrama Entidade-Relacionamento (ER)**

****

## 

## **3. Detalhamento das Implementações**

### **3.1 Criação de tabelas**

Definidas com chaves primárias e estrangeiras para garantir integridade referencial.

### **3.2 Procedimentos armazenados (Procedures)**

* adicionar\_aluno: Inserção simples de novo aluno.
* listar\_alunos: Exibe nomes de todos os alunos via cursor.
* cadastrar\_professor: Inserção de professor com email opcional.

### 

### 

### **3.3 Funções (Functions)**

* calcular\_idade: Retorna idade em anos a partir da data de nascimento.
* alunos\_maiores\_de: Retorna alunos cuja idade é maior que o parâmetro.
* dividir\_notas: Divide duas notas, com tratamento de divisão por zero.

### **3.4 Triggers**

* trg\_auditar\_aluno: Registra alterações no nome do aluno em tabela de log.
* trg\_inserir\_presenca: Insere presença padrão (falta) automaticamente ao matricular aluno.
* trg\_bloquear\_email: Bloqueia atualização do email do professor.

### **3.5 Índices**

* Índices simples em colunas nome (alunos), data\_matricula (matriculas) e combinação disciplinas\_id, semestre (turmas).
* Índice específico para busca rápida com ILIKE (pg\_trgm e índice GIN em alunos.nome).

## **4. Consultas e Joins Exemplares**

### **4.1 Aluno + Turma + Disciplina + Professor**

select a.nome as aluno, t.semestre, d.nome as disciplina, p.nome as professor

from alunos a

inner join matriculas m on a.id = m.alunos\_id

inner join turmas t on m.turmas\_id = t.id

inner join disciplinas d on t.disciplinas\_id = d.id

inner join professores p on t.professores\_id = p.id;

### **4.2 Alunos mesmo sem nota lançada (LEFT JOIN)**

select a.nome as aluno, n.avaliacao, n.nota

from alunos a

left join matriculas m on a.id = m.alunos\_id

left join notas n on m.id = n.matricula\_id;

### **4.3 Presenças mesmo sem vínculo com alunos (RIGHT JOIN)**

select a.nome as aluno, pr.date\_aula, pr.presente

from alunos a

right join matriculas m on a.id = m.alunos\_id

right join presencas pr on m.id = pr.matricula\_id;

### 

### 

### 

### 

### **4.4 Média de notas por aluno**

select a.nome as aluno, medias.media\_geral

from alunos a

inner join (

select m.alunos\_id, avg(n.nota) as media\_geral

from matriculas m

join notas n on m.id = n.matricula\_id

group by m.alunos\_id

) as medias on a.id = medias.alunos\_id;

**6. Conclusão**

O modelo implementado oferece uma estrutura robusta para gerenciar dados acadêmicos com integridade e eficiência. O uso combinado de procedimentos armazenados, funções, triggers e índices permite tanto a manutenção automatizada quanto a otimização de consultas. O índice trigram para busca textual melhora bastante a performance em filtros com padrão parcial.