

# Database Modeling & SQL



## Portfólio

**Aluno: Marco Antonio Monteiro Pedro**

<https://github.com/marcosamambaia/ModelagemSQLPortfolio.git>

## **Levantamento de requisitos**

### **Respostas simuladas:**

#### **1:Quais informações sobre os alunos precisam ser armazenadas?**

*Precisamos guardar o nome completo, CPF, matrícula, curso matriculado, telefone, e-mail e o histórico de notas dos alunos.*

#### **2: Quais dados dos professores devem ser cadastrados?**

*O sistema deve registrar o nome, CPF, disciplinas que lecionam, telefone e e-mail dos professores.*

#### **3: Como os cursos e matérias devem ser organizados no sistema?**

*Cada curso tem várias matérias, e cada matéria pode ter diferentes professores ensinando em turmas distintas.*

#### **4: Será necessário armazenar dados sobre as turmas e horários das aulas?**

*Sim, precisamos definir turmas com um código único, além de guardar o horário e dia da semana em que as aulas acontecem.*

#### **5: Os alunos poderão se matricular em mais de um curso simultaneamente?**

*Não, cada aluno pode estar matriculado em apenas um curso por vez.*

#### **6: Como devem ser registradas as notas dos alunos?**

*As notas devem ser armazenadas por matéria, incluindo a nota de cada prova e a média final.*

#### **7: Existem requisitos para controle de presença dos alunos?**

*Sim, queremos um sistema que registre a presença dos alunos, pois a frequência mínima para aprovação é 75% das aulas.*

#### **8: Os professores terão acesso ao sistema para lançar notas e frequência?**

*Sim, cada professor deve ter um login e senha para cadastrar notas e frequência dos alunos em suas matérias.*

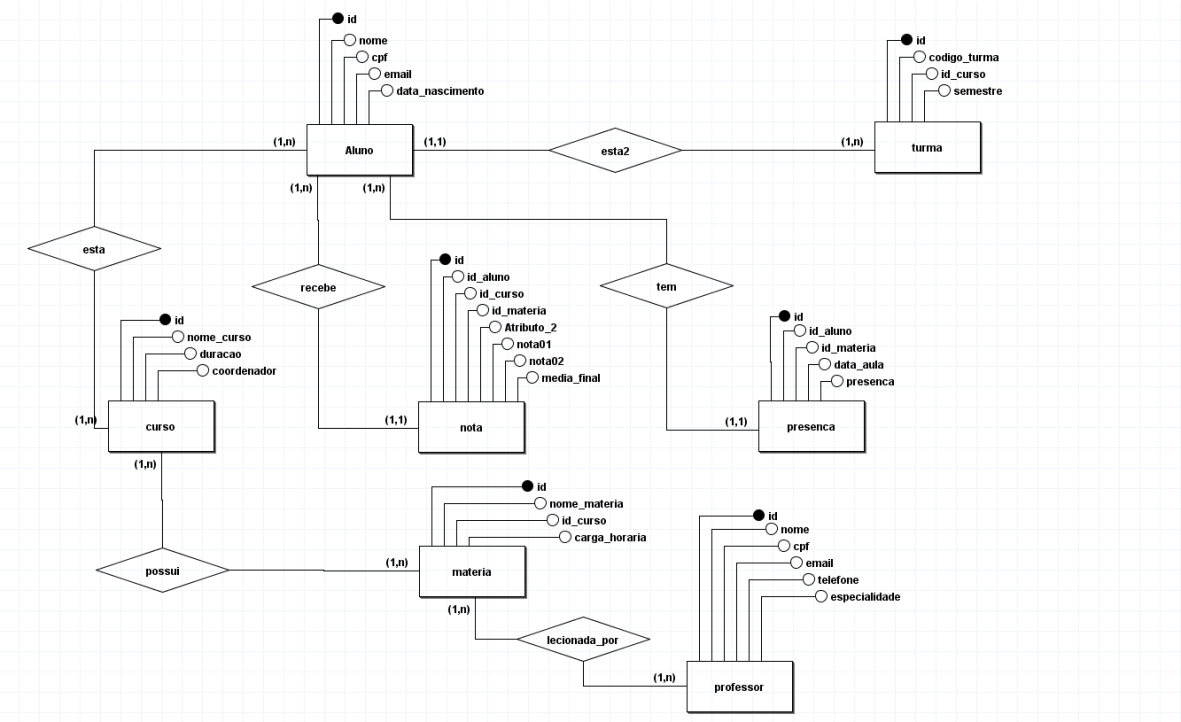
#### **9: Haverá alguma restrição de acesso a determinadas informações dentro do sistema?**

*Sim, os alunos só podem visualizar suas próprias notas, enquanto os professores podem editar as notas e presença de suas turmas.*

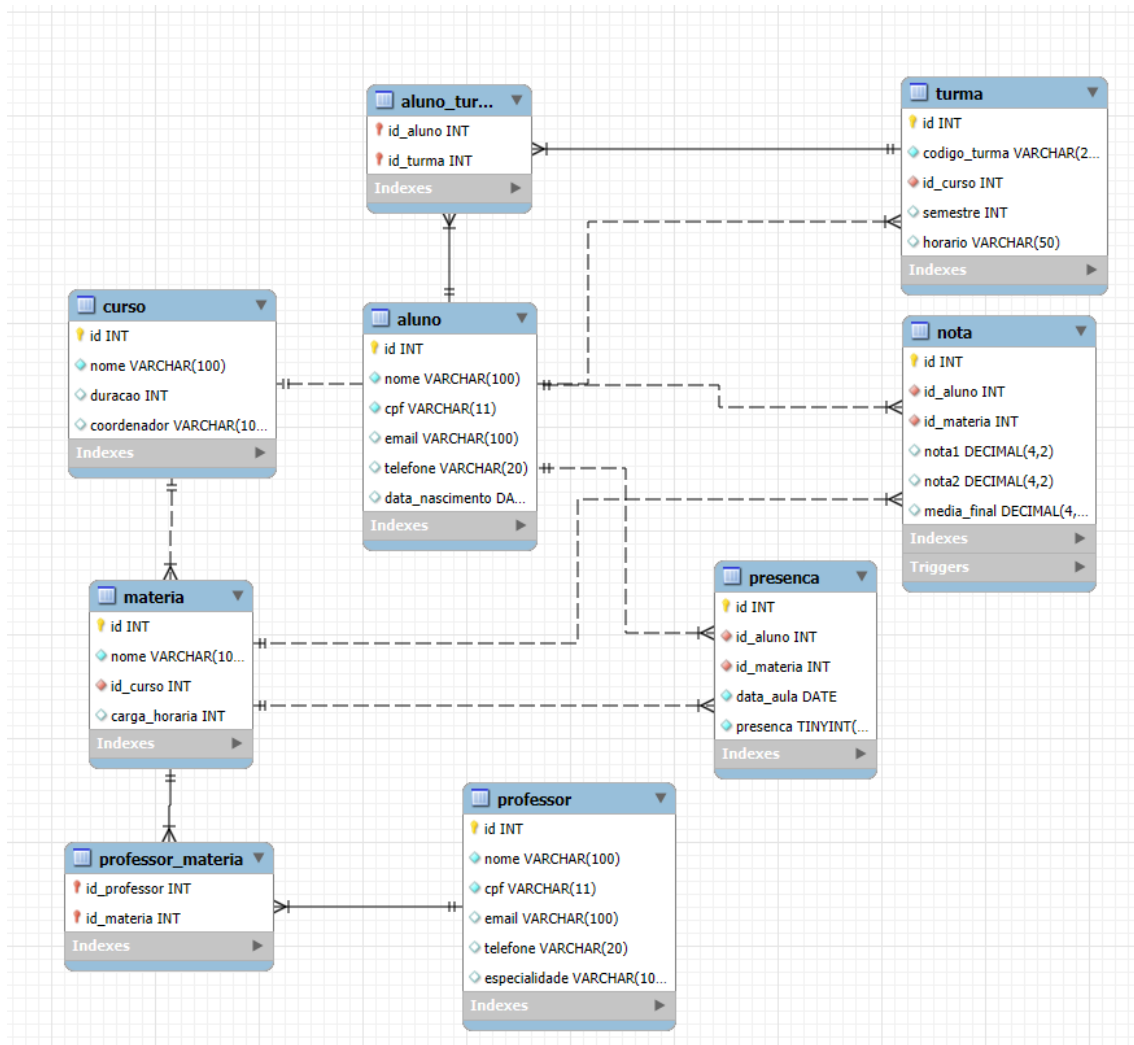
#### **10: O sistema deve gerar relatórios sobre alunos, turmas, notas ou desempenho acadêmico?**

*Sim, queremos relatórios de notas por matéria, desempenho dos alunos por curso e frequência média por turma.*

Modelo Conceitual:



## Modelo Logico:



## Código com testes:

```

/*****
*****

***** Portfolio Modelagem SQL *****/

*****
*****/
```

-- criar o banco de dados da faculdade

```
create database db_portifolio_faculdade;
```

```
use db_portifolio_faculdade;
```

```
-- criar a tabela de alunos
```

```
create table aluno (  
    id int primary key auto_increment not null, -- identificador único para cada aluno  
    nome varchar(100) not null, -- nome completo do aluno  
    cpf varchar(11) unique not null, -- CPF do aluno, deve ser único e ter 11 dígitos  
    email varchar(100), -- e-mail para contato  
    telefone varchar(20), -- telefone para contato  
    data_nascimento date, -- data de nascimento do aluno  
    constraint chk_cpf_aluno check (length(cpf) = 11) -- garante que o CPF tenha  
    exatamente 11 dígitos  
);
```

```
-- criar a tabela de cursos
```

```
create table curso (  
    id int primary key auto_increment not null, -- identificador único para cada curso  
    nome varchar(100) not null, -- nome do curso  
    duracao int, -- duração do curso em semestres  
    coordenador varchar(100) -- nome do coordenador responsável pelo curso  
);
```

```
-- criar a tabela de matérias
```

```
create table materia (  
    id int primary key auto_increment not null, -- identificador único da matéria  
    nome varchar(100) not null, -- nome da matéria  
    id_curso int not null, -- cada matéria pertence a um curso específico  
    carga_horaria int, -- número total de horas da matéria  
    foreign key (id_curso) references curso(id) on delete cascade -- se um curso for  
    removido, suas matérias também serão excluídas  
);
```

-- criar a tabela de professores

```
create table professor (  
    id int primary key auto_increment not null, -- identificador único do professor  
    nome varchar(100) not null, -- nome do professor  
    cpf varchar(11) unique not null, -- CPF do professor, deve ser único e ter 11 dígitos  
    email varchar(100), -- e-mail para contato  
    telefone varchar(20), -- telefone do professor  
    especialidade varchar(100), -- área de especialização do professor  
    constraint chk_cpf_professor check (length(cpf) = 11) -- validação para garantir que o  
    CPF tenha 11 dígitos  
);
```

-- criar a tabela de turmas

```
create table turma (  
    id int primary key auto_increment not null, -- identificador único da turma  
    codigo_turma varchar(20) unique not null, -- código único que identifica a turma  
    id_curso int not null, -- cada turma pertence a um curso específico  
    semestre int, -- semestre da turma (ex: 1º, 2º)  
    horario varchar(50), -- horário da turma  
    foreign key (id_curso) references curso(id) on delete cascade -- se um curso for  
    removido, suas turmas também serão excluídas  
);
```

-- criar a tabela de associação entre aluno e turma

```
create table aluno_turma (  
    id_aluno int not null, -- identificador do aluno  
    id_turma int not null, -- identificador da turma  
    primary key (id_aluno, id_turma), -- chave primária composta pelos dois IDs  
    foreign key (id_aluno) references aluno(id) on delete cascade, -- se um aluno for  
    removido, sua relação com a turma também será apagada  
    foreign key (id_turma) references turma(id) on delete cascade -- se uma turma for  
    removida, seus alunos vinculados também serão excluídos
```

```
);
```

```
-- criar a tabela de associação entre professor e matéria
```

```
create table professor_materia (
```

```
    id_professor int not null, -- identificador do professor, não pode ser nulo
```

```
    id_materia int not null, -- identificador da matéria, não pode ser nulo
```

```
    primary key (id_professor, id_materia), -- chave primária composta pelos dois IDs
```

```
    foreign key (id_professor)
```

```
        references professor(id) on delete cascade, -- se um professor for removido, todas as  
associações dele com matérias serão apagadas
```

```
    foreign key (id_materia)
```

```
        references materia(id) on delete cascade -- se uma matéria for removida, todas as  
associações dela com professores serão excluídas
```

```
);
```

```
-- criar a tabela de notas
```

```
create table nota (
```

```
    id int primary key auto_increment not null, -- identificador único da nota
```

```
    id_aluno int not null, -- identificador do aluno que recebeu a nota
```

```
    id_materia int not null, -- identificador da matéria da nota
```

```
    nota1 decimal(4,2), -- nota da primeira avaliação
```

```
    nota2 decimal(4,2), -- nota da segunda avaliação
```

```
    media_final decimal(4,2), -- média final calculada automaticamente pelo trigger
```

```
    foreign key (id_aluno) references aluno(id) on delete cascade, -- se um aluno for  
removido, suas notas também serão excluídas
```

```
    foreign key (id_materia) references materia(id) on delete cascade -- se uma matéria for  
removida, suas notas também serão apagadas
```

```
);
```

```
-- criar a tabela de presença
```

```
create table presenca (
```

```
    id int primary key auto_increment not null, -- identificador único da presença
```

```
id_aluno int not null, -- identificador do aluno que teve sua presença registrada
id_materia int not null, -- identificador da matéria
data_aula date not null, -- data da aula
presenca boolean not null, -- presença (true = presente, false = ausente)

foreign key (id_aluno) references aluno(id) on delete cascade, -- se um aluno for
removido, seus registros de presença também serão apagados

foreign key (id_materia) references materia(id) on delete cascade -- se uma matéria for
removida, seus registros de presença também serão excluídos
);
```

```
-- criar trigger para calcular a média automaticamente
delimiter $$
```

```
create trigger trg_calcular_media
before insert on nota
for each row
begin
    -- calcula a média antes de inserir no banco de dados
    set new.media_final = (new.nota1 + new.nota2) / 2;
end $$
```

```
delimiter ;
```

```
-- criar procedure para listar todas as matérias de um curso específico
delimiter $$
```

```
create procedure listar_materias_por_curso (in curso_id int)
begin
    -- retorna todas as matérias de um curso informado
    select nome
    from materia
    where id_curso = curso_id;
```



end \$\$

delimiter ;

```
/*****
*****

***** Simulação inserindo dados: *****/
*****/
```

-- inserir alunos

insert into aluno (nome, cpf, email, telefone, data\_nascimento)

values

('João Silva', '12345678901', 'joao@email.com', '11999999999', '2000-05-12'),

('Maria Oliveira', '98765432100', 'maria@email.com', '21988888888', '1999-09-23');

-- inserir cursos

insert into curso (nome, duracao, coordenador)

values

('Engenharia de Software', 8, 'Prof. Ricardo Santos'),

('Administração', 8, 'Prof. Ana Souza');

-- inserir matérias

insert into materia (nome, id\_curso, carga\_horaria)

values

('Banco de Dados', 1, 60),

('Gestão Empresarial', 2, 50);

-- inserir professores

insert into professor (nome, cpf, email, telefone, especialidade)

values

```
('Carlos Ferreira', '11223344556', 'carlos@email.com', '11977777777', 'Banco de Dados'),  
(  
'Fernanda Lima', '66778899000', 'fernanda@email.com', '21966666666', 'Gestão  
Empresarial');
```

```
-- inserir turmas
```

```
insert into turma (codigo_turma, id_curso, semestre, horario)
```

```
values
```

```
('ES2025-A', 1, 2, '18h - 21h'),
```

```
('ADM2025-B', 2, 3, '19h - 22h');
```

```
-- associar alunos às turmas
```

```
insert into aluno_turma (id_aluno, id_turma)
```

```
values
```

```
(1, 1), -- João está na turma ES2025-A
```

```
(2, 2); -- Maria está na turma ADM2025-B
```

```
-- associar professores às matérias
```

```
insert into professor_materia (id_professor, id_materia)
```

```
values
```

```
(1, 1), -- Carlos leciona Banco de Dados
```

```
(2, 2); -- Fernanda leciona Gestão Empresarial
```

```
-- inserir notas
```

```
insert into nota (id_aluno, id_materia, nota1, nota2)
```

```
values
```

```
(1, 1, 8.5, 9.0), -- Nota de João em Banco de Dados
```

```
(2, 2, 7.0, 8.2); -- Nota de Maria em Gestão Empresarial
```

```
-- inserir presença
```

```
insert into presenca (id_aluno, id_materia, data_aula, presenca)
```

```
values
```

(1, 1, '2025-06-01', true), -- João esteve presente na aula de Banco de Dados

(2, 2, '2025-06-01', false); -- Maria faltou à aula de Gestão Empresarial

```
/*****  
*****  
***** Simulação Consulta de dados *****/8  
*****  
*****/
```

-- listar todos os alunos

```
select * from aluno;
```

-- listar todas as matérias e seus cursos

```
select materia.nome as materia, curso.nome as curso
```

```
from materia
```

```
join curso on materia.id_curso = curso.id;
```

-- listar professores e matérias que lecionam

```
select professor.nome as professor, materia.nome as materia
```

```
from professor_materia
```

```
join professor on professor_materia.id_professor = professor.id
```

```
join materia on professor_materia.id_materia = materia.id;
```

-- listar alunos com suas respectivas turmas

```
select aluno.nome as aluno, turma.codigo_turma as turma
```

```
from aluno_turma
```

```
join aluno on aluno_turma.id_aluno = aluno.id
```

```
join turma on aluno_turma.id_turma = turma.id;
```

-- listar notas dos alunos

```
select aluno.nome as aluno, materia.nome as materia, nota.nota1, nota.nota2,  
nota.media_final
```

```
from nota
```

```
join aluno on nota.id_aluno = aluno.id
```

```
join materia on nota.id_materia = materia.id;
```

```
-- listar presença dos alunos
```

```
select aluno.nome as aluno, materia.nome as materia, presenca.data_aula,  
presenca.presenca
```

```
from presenca
```

```
join aluno on presenca.id_aluno = aluno.id
```

```
join materia on presenca.id_materia = materia.id;
```

```
drop database db_estudo_de_caso;
```