Database Modeling & SQL

Aluno: Adriano Moura da Silva

RA: 23814

Criando uma base de dados para que atenda um sistema de uma faculdade.

Para desenvolver um sistema de base de dados que atenda às necessidades de uma faculdade, é necessário considerar diversos aspectos. Vamos explorar as respostas para as perguntas fornecidas e delinear uma estrutura de base de dados adequada para armazenar e manipular as informações de alunos, professores, cursos, matérias, turmas e notas.

1. Principais necessidades do cliente:
Os proprietários de universidades desejam um projeto que:
Armazene informações sobre alunos, professores, cursos, disciplinas e notas. Aulas de controle, incluindo alunos e professores.

Registre as notas dos alunos por disciplina e turma. Obtenha relatórios ou análises do desempenho dos alunos.

2. Quais informações devem ser armazenadas:

- Alunos: dados pessoais e acadêmicos.
- Professores: dados pessoais e disciplinas ministradas.
- Cursos: informações sobre os diversos cursos oferecidos. Disciplinas: cada disciplina pertencente a um determinado curso.
- Turmas: relações entre professores, alunos e disciplinas.
- Notas: qualificações atribuídas a cada aluno de acordo com as disciplinas.

3. Quais dados devem ser armazenados:

- Alunos: nome, matrícula, data de nascimento, contato, curso.
- Professores: nome, matrícula, especialidade, disciplinas que ministram.
- Curso: nome do curso, duração, assuntos relevantes.
- Disciplina: nome, código, carga horária, curso a que pertencem.
- Turmas: identificação da turma, disciplina, professor, alunos matriculados.
- Notas: nota final de cada aluno em cada disciplina.

4. O que será feito com os dados posteriormente:

Consultas sobre o desempenho dos alunos em cada disciplina. Acompanhamento do curso pelo docente.

Controle de matrícula de alunos em turmas especiais.

Geração de notas e relatórios de frequência. Análise dos sujeitos reprovados ou aprovados

5. Tabelas que precisam ser criadas:
Aqui estão as principais tabelas necessárias para armazenar todas as informações:

Tabela de alunos

Tabela dos professores
Tabela de cursos
Tabela de disciplinas
Tabela de turmas
Tabela de notas

6. Atributos de cada tabela

Tabela de alunos

- `id_aluno` (chave primária, inteiro): Identificador único do aluno.
- `nome` (texto): Nome do aluno.
- `matricula` (texto): Número de matrícula.
- `data_nascimento` (data): Data de nascimento.
- `email` (texto): E-mail de contato.

Tabela de professores

- `id_curso` (chave estrangeira, inteiro): Curso ao qual o aluno está matriculado.
- `id_professor` (chave primária, inteiro): Identificador único do professor.
- `nome` (texto): Nome do professor.
- `especialidade` (texto): Área de especialização.
- `matricula` (texto): Número de matrícula do professor.
- `email` (texto): E-mail de contato.

Tabela:cursos

- `id_curso` (chave primária, inteiro): Identificador único do curso.
- `nome_curso` (texto): Nome do curso.
- `duracao` (inteiro): Duração do curso em semestres.

Tabela: disciplinas

- `id_disciplina` (chave primária, inteiro): Identificador único da disciplina.

- `nome_disciplina` (texto): Nome da disciplina.
- `carga_horaria` (inteiro): Carga horária da disciplina.
- `id_curso` (chave estrangeira, inteiro): Curso ao qual a disciplina pertence.

Tabela:Turmas

- `id_turma` (chave primária, inteiro): Identificador único da turma.
- `id_disciplina` (chave estrangeira, inteiro): Disciplina oferecida nesta turma.
- `id_professor` (chave estrangeira, inteiro): Professor responsável pela turma.
- `ano` (inteiro): Ano da turma.
- `semestre` (inteiro): Semestre da turma.

Tabela: Notas

- `id_nota` (chave primária, inteiro): Identificador único da nota.
- `id_aluno` (chave estrangeira, inteiro): Aluno a quem a nota pertence.
- `id_turma` (chave estrangeira, inteiro): Turma na qual a nota foi atribuída.
- `nota_final` (decimal): Nota final do aluno na disciplina.

7. Tipos de dados de cada atributo

- `Inteiro`: Para atributos como IDs, duração e ano/semestre.
- Texto (VARCHAR): Para nomes, matrículas, e-mails.
- Decimal: Para armazenar notas.
- Data: Para armazenar datas de nascimento.

8. Relacionamentos a serem criados entre as tabelas

Aluno-Curso: Relacionamento de um para muitos. Um curso pode ter vários alunos, mas um aluno pertence a um único curso.
Curso-Disciplina: Relacionamento de um para muitos. Um curso pode ter várias disciplinas.
Professor-Turma: Relacionamento de um para muitos. Um professor pode ter várias turmas.
Disciplina-Turma: Relacionamento de um para muitos. Uma disciplina pode ter várias turmas.
Turma-Nota: Relacionamento de um para muitos. Uma turma pode ter várias notas.
Aluno-Turma (através da tabela de Notas): Relacionamento de muitos para muitos (um aluno pode estar em várias turmas, e uma turma pode ter vários alunos).

Modelo de Diagrama Entidade-Relacionamento (DER)

O diagrama seria útil para ilustrar a estrutura da base de dados, mostrando os relacionamentos entre as tabelas. Seria algo como:

Aluno (1
) Curso
Curso (1
) Disciplina
Disciplina (1
) Turma
Professor (1
) Turma
Aluno (N
) Turma (com Notas ligando ambos)
Com este esqueleto, a base de dados será capaz de armazenar e gerenciar as informações necessárias para a faculdade.

9 Objetivos de Aprendizagem (Plano de Aprendizagem)

Os objetivos de aprendizagem deste projeto são abrangentes das várias etapas do ciclo de desenvolvimento de bases de dados, a qual é importante para garantir a funcionalidade conforme a proposta da solução.

Análise dos requisitos

Compreender as necessidades dos clientes (faculdade) e identificar quais informações são necessárias para o funcionamento do sistema.

Identificar quais tipos de dados devem, na verdade, ser armazenados, como alunos, professores, turmas, notas etc.

Modelagem Conceitual

Uma representação abstrata da estrutura dos dados e dos relacionamentos entre as entidades (alunos, professores, cursos etc.) a partir das necessidades levantadas.

Definir as entidades e de relacionamentos entre elas, sem preocupação ainda com a implementação técnica.

Modelagem Lógica

Realizar a transposição da modelagem conceitual para um modelo lógico mais próximo da implementação.

Modelagem física

Definir tabelas, atributos e relacionamentos com base no modelo relacional, com foco em chaves primárias, chaves estrangeiras e cardinalidade dos relacionamentos. Modelagem Física, por meio de modelagem lógica usando SQL.

Geração do código que será utilizado para caracterizar o esquema do banco de dados, incluindo a criação de tabelas, definições de tipos de dados e restrições, como chaves estrangeiras e chaves únicas.

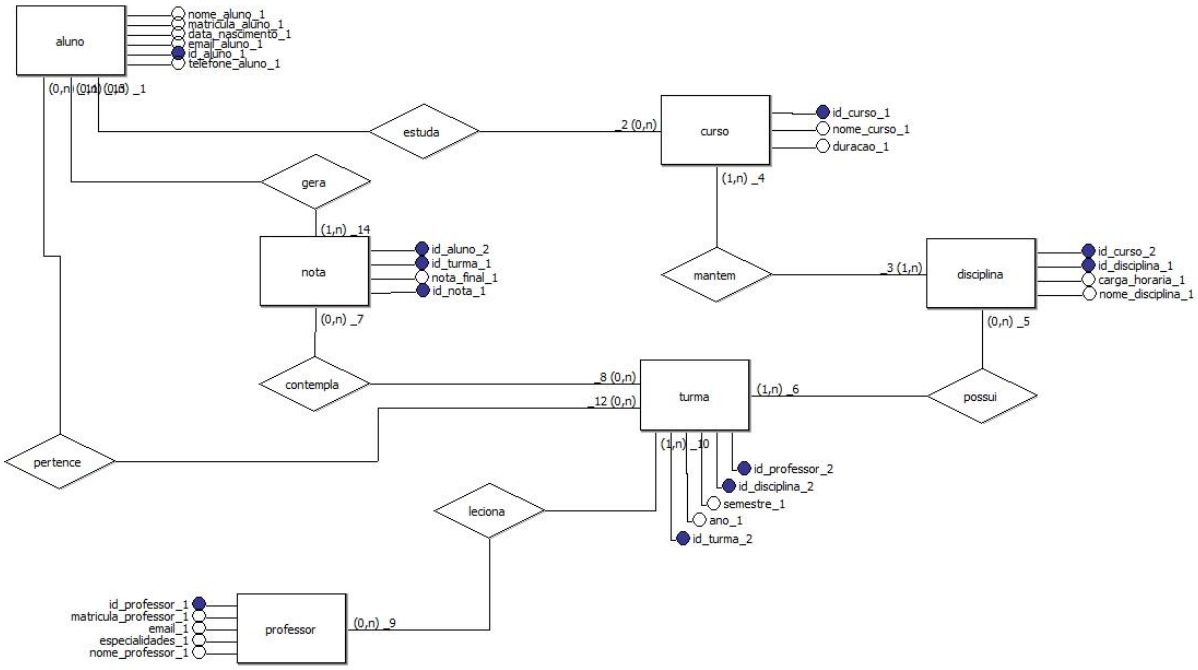
10. Entregável

a) Texto contido na seção "projeto" deste documento

Este texto identifica as exigências para o sistema de armazenamento de informações da instituição de ensino, de acordo com a especificação de requisitos apresentada acima.

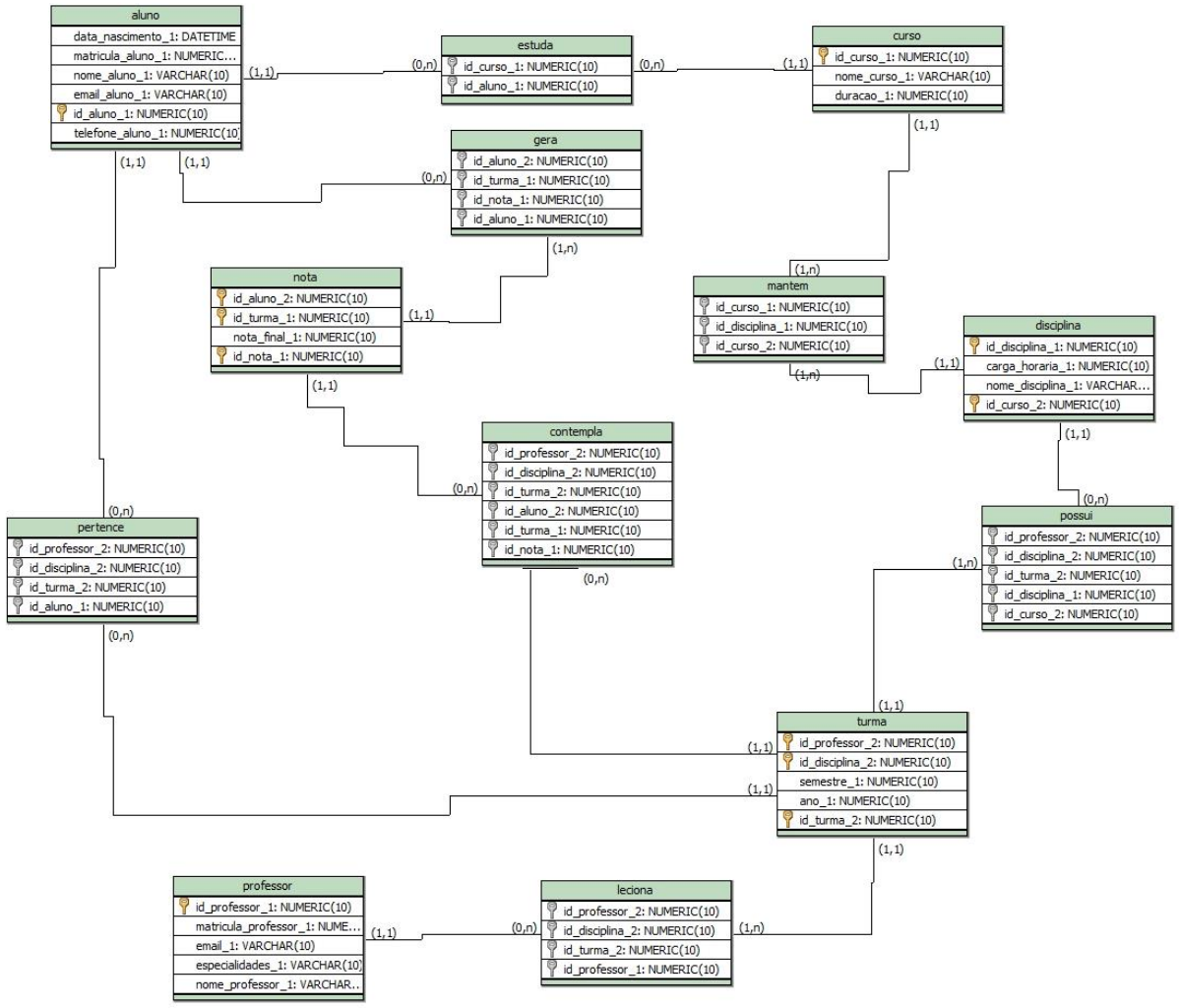
b) Modelo conceitual para a solução do problema

O modelo conceitual pode ser expresso por um diagrama de entidade-relação (DER), que descreve as entidades principais e as relações entre elas. Para o projeto da instituição de ensino, temos:



c) Modelo lógico para solução do problema

No **Modelo Lógico**, vamos detalhar as tabelas, com as chaves primárias (PK), estrangeiras (FK) e os tipos de relacionamentos entre elas.



d) Modelo físico (código SQL)

O modelo físico é implementação prática das tabelas SQL, conforme descrito no modelo logico. O código SQL abaixo corresponde ao modelo do projeto.

- Geração de Modelo físico

-- Sql ANSI 2003 - brModelo.

CREATE TABLE aluno (

data_nascimento_1 DATETIME,

matricula_aluno_1 NUMERIC(10),

```
nome_aluno_1 VARCHAR(10),
email_aluno_1 VARCHAR(10),
id_aluno_1 NUMERIC(10) PRIMARY KEY,
telefone_aluno_1 NUMERIC(10)
)

CREATE TABLE curso (
id_curso_1 NUMERIC(10) PRIMARY KEY,
nome_curso_1 VARCHAR(10),
duracao_1 NUMERIC(10)
)

CREATE TABLE disciplina (
id_disciplina_1 NUMERIC(10),
carga_horaria_1 NUMERIC(10),
nome_disciplina_1 VARCHAR(10),
id_curso_2 NUMERIC(10),
PRIMARY KEY(id_disciplina_1,id_curso_2)
)

CREATE TABLE nota (
id_aluno_2 NUMERIC(10),
id_turma_1 NUMERIC(10),
nota_final_1 NUMERIC(10),
id_nota_1 NUMERIC(10),
PRIMARY KEY(id_aluno_2,id_turma_1,id_nota_1)
)

CREATE TABLE professor (
id_professor_1 NUMERIC(10) PRIMARY KEY,
matricula_professor_1 NUMERIC(10),
email_1 VARCHAR(10),
```

```
especialidades_1 VARCHAR(10),
nome_professor_1 VARCHAR(10)
)

CREATE TABLE turma (
id_professor_2 NUMERIC(10),
id_disciplina_2 NUMERIC(10),
semestre_1 NUMERIC(10),
ano_1 NUMERIC(10),
id_turma_2 NUMERIC(10),
PRIMARY KEY(id_professor_2,id_disciplina_2,id_turma_2)
)

CREATE TABLE estuda (
id_curso_1 NUMERIC(10),
id_aluno_1 NUMERIC(10),
FOREIGN KEY(id_curso_1) REFERENCES curso (id_curso_1),
FOREIGN KEY(id_aluno_1) REFERENCES aluno (id_aluno_1)
)

CREATE TABLE mantem (
id_curso_1 NUMERIC(10),
id_disciplina_1 NUMERIC(10),
id_curso_2 NUMERIC(10),
FOREIGN KEY(id_curso_1) REFERENCES curso (id_curso_1),
FOREIGN KEY(id_curso_2,,) REFERENCES disciplina (id_disciplina_1,id_curso_2)
)

CREATE TABLE possui (
id_professor_2 NUMERIC(10),
id_disciplina_2 NUMERIC(10),
id_turma_2 NUMERIC(10),
```

```
id_disciplina_1 NUMERIC(10),

id_curso_2 NUMERIC(10),

FOREIGN KEY(id_turma_2,,,) REFERENCES turma
(id_professor_2,id_disciplina_2,id_turma_2),

FOREIGN KEY(id_curso_2,,) REFERENCES disciplina (id_disciplina_1,id_curso_2)

)


CREATE TABLE contempla (

id_professor_2 NUMERIC(10),

id_disciplina_2 NUMERIC(10),

id_turma_2 NUMERIC(10),

id_aluno_2 NUMERIC(10),

id_turma_1 NUMERIC(10),

id_nota_1 NUMERIC(10),

FOREIGN KEY(id_turma_2,,,) REFERENCES turma
(id_professor_2,id_disciplina_2,id_turma_2),

FOREIGN KEY(id_nota_1,,,) REFERENCES nota (id_aluno_2,id_turma_1,id_nota_1)

)


CREATE TABLE leciona (

id_professor_2 NUMERIC(10),

id_disciplina_2 NUMERIC(10),

id_turma_2 NUMERIC(10),

id_professor_1 NUMERIC(10),

FOREIGN KEY(id_turma_2,,,) REFERENCES turma
(id_professor_2,id_disciplina_2,id_turma_2),

FOREIGN KEY(id_professor_1) REFERENCES professor (id_professor_1)

)


CREATE TABLE pertence (

id_professor_2 NUMERIC(10),

id_disciplina_2 NUMERIC(10),

id_turma_2 NUMERIC(10),
```

```
id_aluno_1 NUMERIC(10),
FOREIGN KEY(id_turma_2,,,) REFERENCES turma
(id_professor_2,id_disciplina_2,id_turma_2),
FOREIGN KEY(id_aluno_1) REFERENCES aluno (id_aluno_1)
)

CREATE TABLE gera (
id_aluno_2 NUMERIC(10),
id_turma_1 NUMERIC(10),
id_nota_1 NUMERIC(10),
id_aluno_1 NUMERIC(10),
FOREIGN KEY(id_nota_1,,,) REFERENCES nota (id_aluno_2,id_turma_1,id_nota_1),
FOREIGN KEY(id_aluno_1) REFERENCES aluno (id_aluno_1)
)
```

e) Link do GITHUB contendo o código SQL

Após criar o código SQL, será necessário subir o arquivo para um repositório no GITHUB e disponibilizar o link n final no PDF final.

O arquivo está disponível em: https://github.com/AdrianoMoura618/TrabalhoBD.git

AUTOAVALIAÇÃO

A autoavaliação deve contemplar a questão:

Se eu fosse o usuário do sistema, o banco de dados seria apropriado para meu uso?

Alguns tópicos para reflexão

Completude: O banco de dados cobre todas as necessidades levantadas? É fácil acessar as informações dos alunos, notas e disciplinas de forma eficaz!

Desempenho: O sistema está adequado em termos de consultas e armazenamento, existem índices criados, onde necessário!

Manutenção: O banco de dados é flexível para modificação caso seja preciso, se novos cursos ou disciplinas forem incluídos, o sistema poderá ser modificado com estrutura necessária!

Segurança: Existem mecanismos para garantir a integridade dos dados, como chaves estrangeiras e restrições adequadas!

Refletir sobre estes itens será útil para verificar se o projeto é adequado ao uso final.

Com esse projeto lidar com os processos diários se tornara mais fácil.