Aluno: Armazena os dados dos estudantes.

• Instrutor: Guarda as informações dos professores.

• Curso: Contém os detalhes de cada curso oferecido.

• Aula: Representa cada aula individual que compõe um curso.

• Matrícula: É uma entidade associativa que conecta um `Aluno` a um `Curso`, registrando o status e a data da inscrição.

Atributos e Chaves Primárias (PK):

• Aluno

◦ `cpf` (PK): Identificador único do aluno.

◦ `nome\_completo`: Nome do aluno.

◦ `email`: E-mail único do aluno.

◦ `data\_nascimento`: Data de nascimento.

• Instrutor

◦ `codigo\_instrutor` (PK): Código único gerado pelo sistema.

◦ `nome\_completo`: Nome do instrutor.

◦ `mini\_biografia`: Breve descrição sobre a experiência.

• Curso

◦ `codigo\_curso` (PK): Código único do curso (ex: “BCO-01”).

◦ `titulo`: Título do curso.

◦ `carga\_horaria\_total`: Duração total em horas.

◦ `nivel\_dificuldade`: Iniciante, Intermediário ou Avançado.

• Aula

◦ `numero\_ordem` (PK parcial): Número sequencial da aula dentro do curso.

◦ `titulo`: Título da aula.

◦ `link\_video`: URL para o vídeo da aula.

• Matrícula

◦ `data\_matricula`: Data em que a matrícula foi realizada.

◦ `status\_atual`: Situação do aluno no curso (Cursando, Concluído, Cancelado).

Relacionamentos e Cardinalidade:

• Um `Instrutor` pode ministrar vários `Cursos`, mas um `Curso` é ministrado por apenas um `Instrutor`.

◦ Relacionamento: `Instrutor` (1) — (N) `Curso`

• Um `Curso` é composto por várias `Aulas`, mas uma `Aula` pertence a apenas um `Curso`.

◦ Relacionamento: `Curso` (1) — (N) `Aula`

• Um `Aluno` pode se matricular em vários `Cursos`, e um `Curso` pode ter vários `Alunos` matriculados.

◦ Relacionamento: `Aluno` (N) — (N) `Curso`. Este relacionamento gera a entidade associativa `Matrícula`.

Parte 2: Modelo Lógico (Esquema Relacional)

Agora, convertemos o DER em um esquema de tabelas, definindo as Chaves Primárias (PK) e as Chaves Estrangeiras (FK).

• ALUNO

◦ `cpf` (PK, VARCHAR(11))

◦ `nome\_completo` (VARCHAR(255))

◦ `email` (VARCHAR(255))

◦ `data\_nascimento` (DATE)

• INSTRUTOR

◦ `codigo\_instrutor` (PK, INT ou VARCHAR)

◦ `nome\_completo` (VARCHAR(255))

◦ `mini\_biografia` (TEXT)

• CURSO

◦ `codigo\_curso` (PK, VARCHAR(10))

◦ `titulo` (VARCHAR(255))

◦ `carga\_horaria\_total` (INT)

◦ `nivel\_dificuldade` (VARCHAR(50))

◦ `fk\_instrutor` (FK, referencia INSTRUTOR.codigo\_instrutor)

• AULA

◦ `id\_aula` (PK, INT AUTO\_INCREMENT) - É comum criar uma chave primária simples para a tabela `Aula` para facilitar as referências.

◦ `numero\_ordem` (INT)

◦ `titulo` (VARCHAR(255))

◦ `link\_video` (VARCHAR(255))

◦ `fk\_curso` (FK, referencia CURSO.codigo\_curso)

• MATRICULA

◦ `fk\_aluno` (FK, referencia ALUNO.cpf)

◦ `fk\_curso` (FK, referencia CURSO.codigo\_curso)

◦ `data\_matricula` (DATE)

◦ `status\_atual` (VARCHAR(50))

◦ Chave Primária Composta: (`fk\_aluno`, `fk\_curso`)

Parte 3: Modelo Físico (Script SQL)

Finalmente, este é o código SQL (DDL) para criar as tabelas no banco de dados, com os tipos de dados e as restrições `NOT NULL` e `PRIMARY KEY`.

CREATE TABLE ALUNO (

cpf VARCHAR(11) PRIMARY KEY,

nome\_completo VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL UNIQUE,

data\_nascimento DATE NOT NULL

);

– Tabela para armazenar os dados dos instrutores

CREATE TABLE INSTRUTOR (

codigo\_instrutor INT PRIMARY KEY AUTO\_INCREMENT, – Usando INT com auto-incremento para código gerado pelo sistema

nome\_completo VARCHAR(255) NOT NULL,

mini\_biografia TEXT

);

– Tabela para armazenar os cursos

CREATE TABLE CURSO (

codigo\_curso VARCHAR(10) PRIMARY KEY,

titulo VARCHAR(255) NOT NULL,

carga\_horaria\_total INT NOT NULL,

nivel\_dificuldade VARCHAR(50) NOT NULL,

fk\_instrutor INT NOT NULL,

FOREIGN KEY (fk\_instrutor) REFERENCES INSTRUTOR(codigo\_instrutor)

);

– Tabela para armazenar as aulas de cada curso

CREATE TABLE AULA (

id\_aula INT PRIMARY KEY AUTO\_INCREMENT,

numero\_ordem INT NOT NULL,

titulo VARCHAR(255) NOT NULL,

link\_video VARCHAR(255),

fk\_curso VARCHAR(10) NOT NULL,

FOREIGN KEY (fk\_curso) REFERENCES CURSO(codigo\_curso)

);

– Tabela associativa para as matrículas dos alunos nos cursos

CREATE TABLE MATRICULA (

fk\_aluno VARCHAR(11) NOT NULL,

fk\_curso VARCHAR(10) NOT NULL,

data\_matricula DATE NOT NULL,

status\_atual VARCHAR(50) NOT NULL,

PRIMARY KEY (fk\_aluno, fk\_curso), – Chave primária composta para garantir que um aluno não se matricule duas vezes no mesmo curso

FOREIGN KEY (fk\_aluno) REFERENCES ALUNO(cpf),

FOREIGN KEY (fk\_curso) REFERENCES CURSO(codigo\_curso)

);

É a linguagem padrão para criar, gerenciar, manipular e consultar banco de dados relacionais.

- \*\*DQL - Linguagem de Consulta de Dados -\*\* Define o comando utilizado para que possamos consultar (SELECT) os dados armazenados no banco;

- \*\*DML - Linguagem de Manipulação de Dados -\*\* Define os comandos utilizados para manipulação de dados no banco (INSERT, UPDATE e DELETE);

- \*\*DDL - Linguagem de Definição de Dados -\*\* Define os comandos utilizados para criação (CREATE) de tabelas, views, índices, atualização dessas estruturas (ALTER), assim como a remoção (DROP);

- \*\*DCL - Linguagem de Controle de Dados -\*\* Define os comandos utilizados para controlar o acesso aos dados do banco, adicionando (GRANT) e removendo (REVOKE) permissões de acesso;

- \*\*DTL - Linguagem de Transação de Dados -\*\* Define os comandos utilizados para gerenciar as transações executadas no banco de dados, como iniciar (BEGIN) uma transação, confirmá-la (COMMIT) ou desfazê-la (ROLLBACK).

![image.png](attachment:f6c19b7f-5f62-4c4d-afa2-ece219f34e3a:image.png)

DDL- CREATE DATABASE

Principais comandos usados.

create database db\_biblioteca;

**SQL PT2**

16 de setembro de 2025

|  |  |  |
| --- | --- | --- |
| **Comando DDL** | **DESCRIÇÃO** | **SYNTAX** |
|  | Crie uma tabela e suas colunas junto com seu tipo de dados. |  |
|  | Modifique os nomes das colunas e adicione ou exclua uma coluna. |  |
|  | Altere o nome da tabela. |  |
|  | Adicione uma explicação ao código SQL para que outros membros da equipe possam revisar. |  |
|  | Remover dados de uma tabela sem excluir a tabela. |  |
|  | Exclua a tabela com seus dados. |  |

**Comando CREATE no SQL**

Quando preciso criar uma nova tabela, uso o comando , conforme mostrado abaixo:

CREATE TABLE table\_name ( column\_1 datatype, column\_2 datatype, ..., column\_n datatype ); Powered By

No exemplo acima, defino colunas com diferentes tipos de dados. O SQL oferece suporte a vários [**tipos de dados**](https://campus.datacamp.com/pt/courses/functions-for-manipulating-data-in-postgresql/overview-of-common-data-types?ex=4), embora os tipos mais comuns sejam geralmente STRING (texto), INT (números inteiros), FLOAT (números decimais) e DATE (data).

Por exemplo, se eu quisesse criar a tabela , eu usaria  como tal:

CREATE TABLE actor ( actor\_id String(32767), first\_name String(32767), last\_name String(32767), last\_update String(32767) ); Powered By

Onde:

 é o nome da tabela

 é um nome de coluna cujo conteúdo deve estar no formato de texto () e não deve exceder 32767 caracteres.

 é um nome de coluna cujo conteúdo deve estar no formato de texto () e não deve exceder 32767 caracteres.

 é um nome de coluna cujo conteúdo deve estar no formato de texto () e não deve exceder 32767 caracteres.

 é um nome de coluna cujo conteúdo deve estar no formato de texto () e não deve exceder 32767 caracteres.

Depois que eu executar a consulta, a nova tabela ficará visível junto com as outras tabelas, para que eu possa consultá-la a qualquer momento.

atv em sala

create database db\_escola\_a; USE db\_escola\_a; CREATE TABLE tbl\_alunos ( id\_aluno INT , nome VARCHAR(100), data\_nascimento DATE, media\_final DECIMAL(4, 2), primary key (id\_aluno) ); ALTER table tbl\_alunos ADD email VARCHAR(150); drop table tbl\_alunos; drop database db\_escola\_a; create database db\_loja\_virtual\_a; Use db\_loja\_virtual\_a; create table tbl\_produtos ( id\_produto INT, nome VARCHAR(100), preco DECIMAL(10, 2), primary key (id\_produto) ); ALTER table tbl\_produtos ADD estoque int; drop table tbl\_produtos; drop database db\_loja\_virtual\_a; create database db\_rh\_empresa\_a; Use db\_rh\_empresa\_a; create table tbl\_funcionarios ( id\_funcionario INT, nome\_completo VARCHAR(150), data\_admissao DATE, preco decimal(10, 2), primary key (id\_funcionario) ); drop table tbl\_funcionarios; drop database db\_rh\_empresa\_a; create database db\_academia; use db\_academia; create table tbl\_membros ( id\_membro int, nome VARCHAR(100), tipo\_plano VARCHAR(50), primary key (id\_membro) ); ALTER table tbl\_membros ADD data\_inscricao DATE; drop table tbl\_membros; drop database db\_academia