

# CURSO TÉCNICO EM INFORMÁTICA



# BANCO DE DADOS



# CURSO TÉCNICO EM INFORMÁTICA

## COMANDOS ESSENCIAIS DE SQL (DDL E DML)

A **DDL (Linguagem de Definição de Dados (Data Definition Language))** é usada para criar, modificar e excluir os objetos do banco de dados, como as tabelas. Os principais comandos são CREATE, ALTER e DROP.

### **CREATE TABLE:** Criando a Estrutura

Este é o ponto de partida. O comando **CREATE TABLE** constrói uma nova tabela no seu banco de dados. Ao criá-la, você precisa definir o nome de cada coluna e o tipo de dado que ela irá armazenar (texto, número, data, etc.).

### A Base Comum (ANSI SQL)

```
CREATE TABLE nome_da_tabela (
    nome_coluna1 TIPO_DE_DADO,
    nome_coluna2 TIPO_DE_DADO,
    ...
    CONSTRAINT nome_restricao PRIMARY KEY (nome_coluna1)
);
```



# CURSO TÉCNICO EM INFORMÁTICA

## COMANDOS ESSENCIAIS DE SQL (DDL E DML)

### Criar o novo usuário

Use a seguinte sintaxe para criar o usuário. Substitua novo\_usuario pelo nome que você quer dar ao usuário e senha\_segura pela senha desejada.

**CREATE USER 'NOVO\_USUARIO'@'%' IDENTIFIED BY 'SENHA\_SEGURA';**

- '**novo\_usuario**': o nome do novo usuário.
- '**localhost**': o local de onde o usuário pode se conectar. Se você quer que o usuário possa se conectar de qualquer lugar, use '%' em vez de 'localhost'.
- **IDENTIFIED BY 'senha\_segura'**: a senha para este usuário.



# CURSO TÉCNICO EM INFORMÁTICA

## Linguagem de Manipulação de Dados (DML) - Comparações

### **DELETE FROM** (Deletar Dados)

Para remover registros, a sintaxe também é a mesma.

#### Exemplo Prático (**DML - DELETE FROM**)

```
DELETE FROM livros WHERE id = 1;
```

- **Comentário:** O comando **DELETE** funciona da mesma forma.



# CURSO TÉCNICO EM INFORMÁTICA

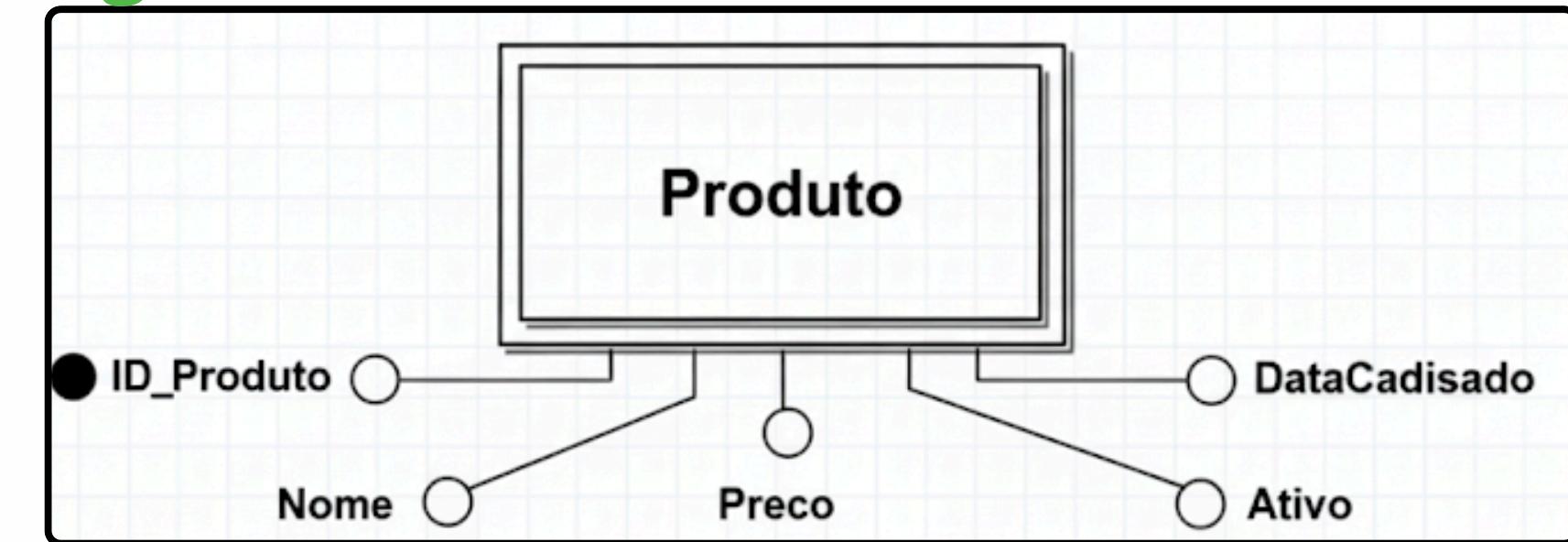
## COMANDOS ESSENCIAIS DE SQL (DDL E DML)

### Comparativo Prático: Criando a Tabela **Produtos**

Vamos criar uma tabela Produtos com as seguintes colunas:

- ID\_Produto: Chave primária numérica que se auto-incremente.
- Nome: Texto para o nome do produto.
- Preco: Número decimal para o preço.
- DataCadastro: Data e hora do cadastro, com o valor padrão sendo o momento atual.
- Ativo: Um valor booleano (verdadeiro/falso).

### Diagrama de Entidade-Relacionamento



# CURSO TÉCNICO EM INFORMÁTICA

## Comparativo Prático: Criando a Tabela **Produtos**

### MySQL

O **MySQL** é conhecido por sua simplicidade na criação de chaves primárias com **AUTO\_INCREMENT** e pela necessidade de especificar o **ENGINE**.

```
CREATE TABLE Produtos (
    ID_Produto INT PRIMARY KEY AUTO_INCREMENT,
    Nome VARCHAR(100) NOT NULL,
    Preco DECIMAL(10, 2) NOT NULL,
    DataCadastro DATETIME DEFAULT CURRENT_TIMESTAMP,
    Ativo BOOLEAN DEFAULT TRUE
) ENGINE=InnoDB;
```

- **Auto-Incremento:** AUTO\_INCREMENT é a palavra-chave específica do MySQL.
- **Booleano:** BOOLEAN é um sinônimo para TINYINT(1).
- **Engine:** ENGINE=InnoDB é uma cláusula específica do MySQL para definir o motor de armazenamento (InnoDB é o padrão e o mais recomendado).



## Comparativo Prático: Criando a Tabela **Produtos**

### PostgreSQL

O **PostgreSQL** é conhecido por ser muito aderente aos padrões SQL e por ter um sistema de tipos de dados robusto.

```
CREATE TABLE Produtos (
    ID_Produto SERIAL PRIMARY KEY,
    Nome VARCHAR(100) NOT NULL,
    Preco DECIMAL(10, 2) NOT NULL,
    DataCadastro TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    Ativo BOOLEAN DEFAULT TRUE
);
```

- **Auto-Incremento:** **SERIAL** é um atalho do PostgreSQL que cria um inteiro e uma sequência associada. A forma mais moderna e padrão SQL seria **INT GENERATED BY DEFAULT AS IDENTITY**.
- **Booleano:** Possui um tipo **BOOLEAN** verdadeiro (true/false).
- **Data e Hora:** **TIMESTAMP WITH TIME ZONE** é um tipo de dado mais robusto que armazena a data/hora junto com o fuso horário.



# CURSO TÉCNICO EM INFORMÁTICA

## Comparativo Prático: Criando a Tabela **Produtos** **SQL Server**

O **SQL Server** tem sua própria sintaxe distinta, especialmente para o auto-incremento e para tipos de dados Unicode.

```
CREATE TABLE Produtos (
    ID_Produto INT PRIMARY KEY IDENTITY(1,1),
    Nome NVARCHAR(100) NOT NULL,
    Preco DECIMAL(10, 2) NOT NULL,
    DataCadastro DATETIME2 DEFAULT GETDATE(),
    Ativo BIT DEFAULT 1
);
```

- **Auto-Incremento:** A sintaxe é IDENTITY(seed, increment), onde (1,1) significa que começa em 1 e incrementa de 1 em 1.
- **Texto Unicode:** NVARCHAR é usado para armazenar textos com caracteres de múltiplos idiomas (Unicode), uma prática recomendada no SQL Server. VARCHAR armazena apenas caracteres non-Unicode.
- **Booleano:** O tipo de dado para verdadeiro/falso é BIT (onde 1 é verdadeiro e 0 é falso).
- **Função de Data:** A função para obter a data e hora atuais é GETDATE() ou SYSDATETIME(). DATETIME2 é o tipo mais preciso



## Linguagem de Definição de Dados (DDL) - Comparações

A **DDL (Data Definition Language)** lida com a estrutura do seu banco de dados. Embora muitos comandos sejam parecidos, as diferenças nos "dialetos" de cada sistema aparecem aqui.

### **ALTER TABLE:** Modificando a Estrutura

Depois de criar uma **tabela**, é comum precisar ajustá-la. O comando **ALTER TABLE** é a ferramenta para isso, mas sua sintaxe varia significativamente.

**Cenário 1:** Adicionar uma nova coluna **Telefone** Neste caso, a sintaxe é felizmente a mesma para os três.

```
-- Sintaxe para MySQL, PostgreSQL e SQL Server
ALTER TABLE Clientes
ADD Telefone VARCHAR(20);
```



# CURSO TÉCNICO EM INFORMÁTICA

## Linguagem de Definição de Dados (DDL) - Comparações

**Cenário 2:** Modificar o tipo de dado da coluna **Nome** para **VARCHAR(150)**

**MySQL:**

```
ALTER TABLE Clientes
MODIFY COLUMN Nome VARCHAR(150);
```



# CURSO TÉCNICO EM INFORMÁTICA

## Linguagem de Definição de Dados (DDL) - Comparações

**Cenário 2:** Modificar o tipo de dado da coluna **Nome** para **VARCHAR(150)**

**PostgreSQL:**

```
ALTER TABLE Clientes  
ALTER COLUMN Nome TYPE VARCHAR(150);
```



## Linguagem de Definição de Dados (DDL) - Comparações

**Cenário 2:** Modificar o tipo de dado da coluna **Nome** para **VARCHAR(150)**

**SQL Server:**

```
ALTER TABLE Clientes
ALTER COLUMN Nome VARCHAR(150);
```

(A sintaxe do SQL Server e PostgreSQL é parecida, mas o PostgreSQL exige a palavra-chave **TYPE**).



## Linguagem de Definição de Dados (DDL) - Comparações

**Cenário 3:** Renomear a coluna **Email** para **Email\_Contato**

**MySQL:**

```
ALTER TABLE Clientes
CHANGE COLUMN Email Email_Contato VARCHAR(100);
```

- Precisa redefinir o tipo da coluna



# CURSO TÉCNICO EM INFORMÁTICA

## Linguagem de Definição de Dados (DDL) - Comparações

**Cenário 3:** Renomear a coluna **Email** para **Email\_Contato**

**PostgreSQL:**

```
ALTER TABLE Clientes
RENAME COLUMN Email TO Email_Contato;
```



# CURSO TÉCNICO EM INFORMÁTICA

## Linguagem de Definição de Dados (DDL) - Comparações

**Cenário 3:** Renomear a coluna **Email** para **Email\_Contato**

**SQL Server:**

```
EXEC sp_rename 'Clientes.Email', 'Email_Contato', 'COLUMN';
```

- Usa um procedimento armazenado (Stored Procedure): sistema ou a aplicação executa um bloco de código SQL que foi salvo previamente dentro do próprio banco de dados.
- **sp\_rename** não é um procedimento que você ou um desenvolvedor da sua equipe criou, mas sim um procedimento armazenado do sistema. Ou seja, é uma ferramenta que já vem "de fábrica" com o próprio banco de dados (neste caso, a sintaxe é típica do Microsoft SQL Server) para realizar tarefas administrativas comuns.



## Linguagem de Definição de Dados (DDL) - Comparações

### DROP TABLE: Excluindo a Estrutura

Este comando remove permanentemente uma tabela e todos os seus dados. A sintaxe básica é padronizada, o que facilita o trabalho.

```
-- Sintaxe para MySQL, PostgreSQL e SQL Server
DROP TABLE Clientes;
```

**Dica de boa prática:** Para evitar erros caso a tabela não exista, você pode usar a cláusula **IF EXISTS**, que é suportada pelos três sistemas.

```
-- Sintaxe para MySQL, PostgreSQL e SQL Server
DROP TABLE IF EXISTS Clientes;
```



# CURSO TÉCNICO EM INFORMÁTICA

## Linguagem de Manipulação de Dados (DML) - Comparações

O DML (Linguagem de Manipulação de Dados) lida com a inserção, consulta, atualização e exclusão de dados. A sintaxe base é a mesma, mas algumas funções podem mudar.

### **INSERT INTO** (Inserir Dados)

O comando para adicionar novos registros é universal.

#### Exemplo Prático (**DML - INSERT INTO**)

```
INSERT INTO livros (id, titulo, autor, ano, capa)
VALUES (1, 'Aventuras na Floresta', 'Júlia Prado', 2022, '/uploads/capa1.png');
```



# CURSO TÉCNICO EM INFORMÁTICA

## Linguagem de Manipulação de Dados (DML) - Comparações

### **SELECT** (Consultar Dados)

A consulta de dados também segue um padrão.

#### Exemplo Prático (**DML - SELECT**)

```
SELECT * FROM livros WHERE autor = 'Júlia Prado';
```



# CURSO TÉCNICO EM INFORMÁTICA

## Linguagem de Manipulação de Dados (DML) - Comparações

### **SELECT** (Consultar Dados)

A consulta de dados também segue um padrão.

#### Exemplo Prático (**DML - SELECT**)

```
SELECT * FROM livros WHERE autor = 'Júlia Prado';
```

**Comentário:** A sintaxe é a mesma para todos. A principal diferença entre eles pode aparecer em funções de data e hora ou manipulação de **strings**.

Por exemplo, a forma de concatenar **strings** é diferente:

**PostgreSQL:** Usa o operador || (SELECT autor || '-' || titulo FROM livros).

**MySQL:** Usa a função CONCAT (SELECT CONCAT(autor, '-', titulo) FROM livros).

**SQL Server:** Usa o operador + (SELECT autor + ' - ' + titulo FROM livros).



# CURSO TÉCNICO EM INFORMÁTICA

## Linguagem de Manipulação de Dados (DML) - Comparações

### **UPDATE** (Atualizar Dados)

Para atualizar informações, a sintaxe é a mesma.

#### Exemplo Prático (**DML - UPDATE**)

```
UPDATE livros SET ano = 2023 WHERE id = 1;
```

- Comentário: A sintaxe para **UPDATE** é a mesma nos três.



# CURSO TÉCNICO EM INFORMÁTICA

## Linguagem de Manipulação de Dados (DML) - Comparações

### **DELETE FROM** (Deletar Dados)

Para remover registros, a sintaxe também é a mesma.

#### Exemplo Prático (**DML - DELETE FROM**)

```
DELETE FROM livros WHERE id = 1;
```

- **Comentário:** O comando **DELETE** funciona da mesma forma.



# CURSO TÉCNICO EM INFORMÁTICA

## Linguagem de Manipulação de Dados (DML) - Comparações

### **DELETE FROM** (Deletar Dados)

Para remover registros, a sintaxe também é a mesma.

#### Exemplo Prático (**DML - DELETE FROM**)

```
DELETE FROM livros WHERE id = 1;
```

- **Comentário:** O comando **DELETE** funciona da mesma forma.



# CURSO TÉCNICO EM INFORMÁTICA

## Estrutura Básica das Consultas em MySQL

O **MySQL** é um dos sistemas de gerenciamento de bancos de dados relacionais mais utilizados no mundo. Ele utiliza a linguagem **SQL (Structured Query Language)** para realizar operações como consultar, inserir, atualizar e excluir dados.

Entre essas operações, a **consulta de dados** é uma das mais importantes, pois permite extrair informações específicas das tabelas de um banco de dados.

Vamos entender como uma consulta básica é estruturada.

### 1. A instrução **SELECT**

A consulta em SQL começa sempre com o comando **SELECT**, que indica quais colunas queremos visualizar.

```
SELECT nome, idade FROM alunos;
```



# CURSO TÉCNICO EM INFORMÁTICA

## Estrutura Básica das Consultas em MySQL (DML)

### 1. A instrução **SELECT**

A consulta em SQL começa sempre com o comando **SELECT**, que indica quais colunas queremos visualizar.

```
SELECT nome, idade FROM alunos;
```

Explicação:

- **SELECT nome, idade** → indica as colunas que serão exibidas;
- **FROM alunos** → indica de qual tabela os dados serão buscados.

Se você quiser exibir todas as colunas, pode usar o asterisco (\*):

```
SELECT * FROM alunos;
```



# CURSO TÉCNICO EM INFORMÁTICA

## Estrutura Básica das Consultas em MySQL (DML)

2. Filtrando resultados com **WHERE**:

O **WHERE** é usado para filtrar os registros que atendem a uma condição específica.

```
SELECT nome, idade FROM alunos  
WHERE idade >= 18;
```

Explicação:

Somente os alunos com 18 anos ou mais serão exibidos.

Outros exemplos de condições:

```
SELECT nome, cidade  
FROM alunos  
WHERE cidade = 'São Paulo';
```



# CURSO TÉCNICO EM INFORMÁTICA

## Estrutura Básica das Consultas em MySQL (DML)

2. Filtrando resultados com **WHERE**:

**Exemplo 2 — Filtro numérico (maior que)**

```
SELECT nome, nota
FROM alunos
WHERE nota > 7;
```

**Explicação:**

- Mostra o nome e a nota de todos os alunos com nota maior que 7.
- O operador **>** significa “**maior que**”.



## Estrutura Básica das Consultas em MySQL (DML)

2. Filtrando resultados com **WHERE**:

### Exemplo 3 — Filtro por padrão de texto (LIKE)

```
SELECT nome  
FROM alunos  
WHERE nome LIKE 'A%';
```

#### Explicação:

- Mostra apenas os alunos cujos nomes começam com a letra “A”.
- O operador **LIKE** é usado para comparações parciais de texto.
- O caractere **%** significa “qualquer sequência de caracteres”.



# CURSO TÉCNICO EM INFORMÁTICA

## Estrutura Básica das Consultas em MySQL (DML)

Resumo dos operadores usados no WHERE

Operador	Significado	Exemplo
=	Igual a	cidade = 'São Paulo'
!= OU <>	Diferente de	curso != 'Matemática'
>	Maior que	nota > 7
<	Menor que	idade < 18
>=	Maior ou igual a	nota >= 6
<=	Menor ou igual a	nota <= 10
LIKE	Compara textos parciais	nome LIKE 'A%'
IN	Pertence a um conjunto	cidade IN ('São Paulo', 'Curitiba')
BETWEEN	Intervalo de valores	nota BETWEEN 7 AND 9

