

CURSO TÉCNICO EM INFORMÁTICA



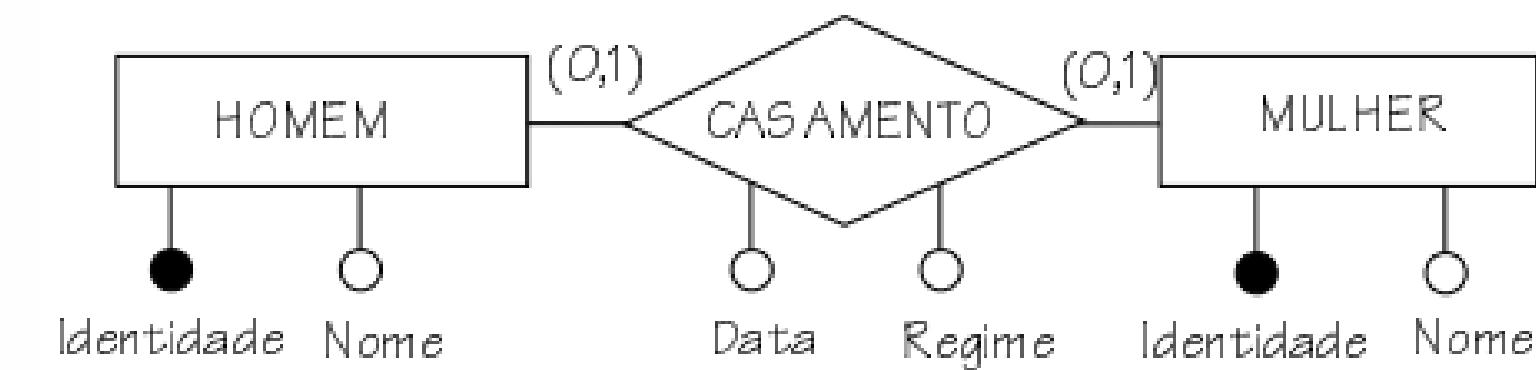
BANCO DE DADOS



CURSO TÉCNICO EM INFORMÁTICA

COMO TRADUZIR UM RELACIONAMENTO 1:1 COM PARTICIPAÇÃO OPCIONAL?

A outra alternativa seria a de gerar uma tabela própria para o relacionamento, conforme o esquema abaixo:



Mulher (IdentM, Nome)

Homem (IdentH, Nome)

Casamento (IdentM, IdentH, Data, Regime)

IdentM referencia Mulher

IdentH referencia Homem

Implementação com Tabela de Relacionamento

A segunda alternativa para implementar um relacionamento 1:1, como no exemplo do **casamento**, é usar uma tabela específica para o relacionamento, a tabela **Casamento**.

Nessa tabela:

- As colunas **IdentH** e **IdentM** são ambas chaves estrangeiras, conectando cada linha de casamento a um Homem e uma Mulher.
- Como o relacionamento é **1:1**, qualquer uma das duas colunas (IdentH ou IdentM) pode ser a chave primária.
- No exemplo, IdentM foi escolhida arbitrariamente como chave primária, e IdentH se torna uma chave alternativa.



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)

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

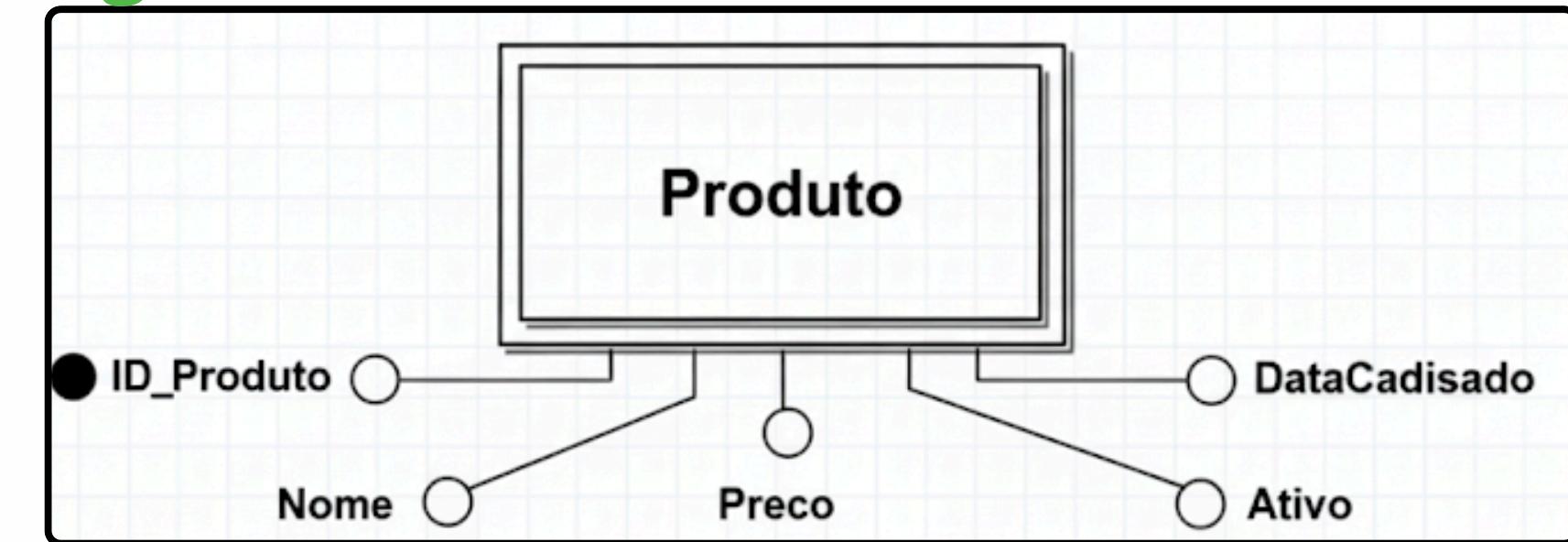
COMANDOS ESSENCIAIS DE SQL (DDL E DML)

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

A DML gerencia os dados dentro das tabelas. Felizmente, os comandos **UPDATE** e **DELETE** são altamente padronizados pela norma ANSI SQL, então a sintaxe é idêntica nos três bancos de dados.

UPDATE: Atualizando Dados

O comando **UPDATE** modifica registros existentes. A regra de ouro é: sempre use a cláusula **WHERE** para especificar exatamente qual(is) linha(s) você quer alterar. Se você esquecer, todos os registros da tabela serão alterados.

