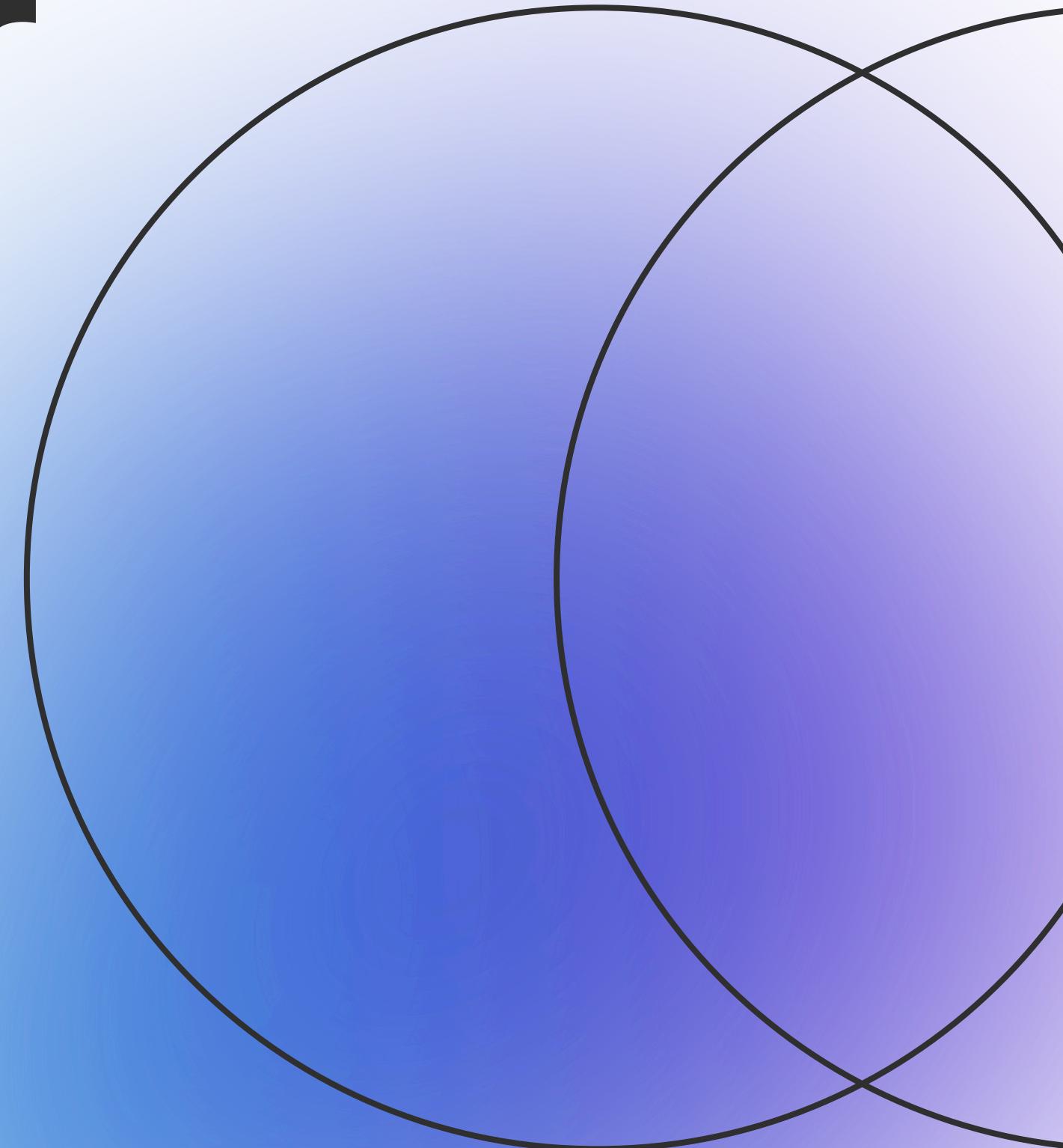


Administrar e implantar banco de dados



Manipulando Dados e Estruturas em SQL



Um guia prático para os comandos essenciais:

- ✓ **INSERT INTO**: Adiciona novos dados.
- ✓ **UPDATE**: Modifica dados existentes.
- ✓ **ALTER TABLE**: Modifica a estrutura de uma tabela.

1. Adicionando Dados com INSERT INTO

O comando **INSERT INTO** é usado para inserir novas linhas (ou registros) em uma tabela.



Imagine uma tabela como uma planilha. O **INSERT** é a ação de preencher uma nova linha com informações.

INSERT INTO: Sintaxe Básica

A forma mais comum de usar o comando é especificando as colunas e os valores correspondentes.

```
◀          ▶
INSERT INTO nome_da_tabela (coluna1, coluna2, coluna3)
VALUES (valor1, valor2, valor3);
```

- ✓ O número de colunas deve ser igual ao número de valores.
- ✓ Os tipos de dados dos valores devem ser compatíveis com os tipos das colunas.

INSERT INTO: Exemplo Prático

Vamos adicionar um novo funcionário à nossa tabela `Funcionarios`.

```
-- Tabela `Funcionarios` (id, nome, cargo, salario)  
INSERT INTO Funcionarios (id, nome, cargo, salario)  
VALUES (1, 'Ana Silva', 'Desenvolvedora', 7500.00);
```

Isso cria um novo registro para Ana Silva na tabela.

2. Modificando Dados com UPDATE

O comando **UPDATE** é utilizado para modificar os registros que já existem em uma tabela.



Você pode alterar o valor de uma ou mais colunas para uma ou várias linhas de uma vez.

UPDATE: Sintaxe e a Cláusula WHERE

A sintaxe básica define qual tabela e colunas atualizar, e **ONDE** aplicar a mudança.

```
UPDATE nome_da_tabela  
SET coluna1 = novo_valor1, coluna2 = novo_valor2  
WHERE condicao;
```



CUIDADO MÁXIMO!

Se você esquecer a cláusula **WHERE**, o **UPDATE** será aplicado a

TODAS AS LINHAS

da tabela. Essa ação é geralmente irreversível.

UPDATE: Exemplo Prático

Vamos promover a funcionária Ana Silva para "Desenvolvedora Sênior" e ajustar seu salário.



```
UPDATE Funcionarios  
SET cargo = 'Desenvolvedora Sênior', salario = 9200.00  
WHERE id = 1;
```

A cláusula **WHERE id = 1** garante que apenas o registro da Ana seja modificado.

3. Modificando a Estrutura com ALTER TABLE

Enquanto **INSERT** e **UPDATE** manipulam os **dados**, o **ALTER TABLE** modifica a **estrutura** da tabela.

É um comando de DDL (Linguagem de Definição de Dados).

Ações comuns:

- ✓ Adicionar uma coluna (**ADD COLUMN**).
- ✓ Remover uma coluna (**DROP COLUMN**).
- ✓ Modificar o tipo de uma coluna (**MODIFY COLUMN**).

ALTER TABLE: Adicionando uma Coluna

Para adicionar uma nova coluna à tabela, usamos a cláusula **ADD COLUMN**.

```
ALTER TABLE nome_da_tabela  
ADD COLUMN nome_da_nova_coluna tipo_de_dado;
```



Exemplo: Adicionar uma data de contratação para os funcionários.

```
ALTER TABLE Funcionarios  
ADD COLUMN data_contratacao DATE;
```

ALTER TABLE: Removendo uma Coluna

Para remover uma coluna existente, usamos **DROP COLUMN**.

Atenção!

Esta ação remove a coluna e todos os dados contidos nela de forma permanente.



```
ALTER TABLE nome_da_tabela  
DROP COLUMN nome_da_coluna;
```



Exemplo: Remover uma coluna `telefone_ramal` que não é mais usada.

```
ALTER TABLE Funcionarios  
DROP COLUMN telefone_ramal;
```

ALTER TABLE: Modificando uma Coluna

Para alterar o tipo de dado de uma coluna, usamos **MODIFY COLUMN** ou **ALTER COLUMN** (a sintaxe pode variar entre sistemas de banco de dados).

```
-- Sintaxe para MySQL
ALTER TABLE nome_da_tabela
MODIFY COLUMN nome_da_coluna novo_tipo_de_dado;
```

Exemplo: Aumentar o tamanho do campo `cargo`.

```
ALTER TABLE Funcionarios
MODIFY COLUMN cargo VARCHAR(100);
```

Resumo Geral

- ✓ Use **INSERT INTO** para **criar** novos registros.
- ✓ Use **UPDATE ... WHERE** para **modificar** registros existentes com segurança.
- ✓ Use **ALTER TABLE** para **mudar a estrutura** da tabela (adicionar, remover ou alterar colunas).



A principal diferença está no alvo da operação:

Dados (Linhas) vs Estrutura (Colunas)