

Comandos SQL :

Os comandos SQL (Structured Query Language).

Permitem que usuários e aplicativos realizem uma ampla gama de operações, desde a criação de estruturas de banco de dados até a manipulação precisa de dados.

Comandos SQL

Os comandos SQL são categorizados em diferentes grupos.

Comandos de Definição de Dados (DDL)

Comandos de Manipulação de Dados (DML)

Comandos de Consulta de Dados (DQL)

Comandos de Definição de Dados (DDL)

Responsáveis pela criação e modificação da estrutura do banco de dados.

Comandos: CREATE, ALTER, DROP.

Comandos de Definição de Dados (DDL)

CREATE: É usado para fazer novos objetos dentro do seu banco de dados.

A estrutura geral é **CREATE TIPO_DE_OBJETO NOME_DO_OBJETO.**

IF NOT EXISTS: essa clausula verifica se um banco de dados com o nome especificado já existe, caso já exista, o comando é ignorado, evitando erros.

**CREATE TABLE nome_tabela (nome_coluna tipo_de_dado restrição,
nome_coluna tipo_de_dado restrição);**

CREATE: É usado para fazer novos objetos dentro do seu banco de dados.

Exemplos:

```
CREATE DATABASE IF NOT EXISTS `gestao_eventos`;
```

```
CREATE TABLE `evento` ( `idEvento` int(11) NOT NULL, `Nome` varchar(100)  
DEFAULT NULL, `data_inicio` date NOT NULL, `data_fim` date NOT NULL,  
`descricao` text DEFAULT NULL, `Local_id_local` int(11) NOT NULL)  
ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_general_ci;
```

Comandos de Definição de Dados (DDL)

ALTER: Permite modificar a estrutura de objetos existentes, como adicionar ou remover colunas de uma tabela.

ADD COLUMN: A cláusula ADD COLUMN é usada para adicionar uma nova coluna a uma tabela. Você especifica o nome da coluna e seu tipo de dados.

Comandos de Definição de Dados (DDL)

ALTER: Permite modificar a estrutura de objetos existentes, como adicionar ou remover colunas de uma tabela.

Adicionar colunas:

**ALTER TABLE nome_da_tabela ADD nome_da_coluna
tipo_de_dados;**

**ALTER TABLE local ADD COLUMN Telefone_Responsavel
VARCHAR(20);**

Comandos de Definição de Dados (DDL)

ALTER: Permite modificar a estrutura de objetos existentes, como adicionar ou remover colunas de uma tabela.

DROP COLUMN : A cláusula **DROP COLUMN** é usada para remover uma coluna de uma tabela. Você especifica o nome da coluna.

Comandos de Definição de Dados (DDL)

ALTER: Permite modificar a estrutura de objetos existentes, como adicionar ou remover colunas de uma tabela.

Remover uma coluna existente:

ALTER TABLE local DROP COLUMN nome_da_coluna;

ALTER TABLE local DROP COLUMN Telefone_Responsavel;

Comandos de Definição de Dados (DDL)

ALTER: Permite modificar a estrutura de objetos existentes, como adicionar ou remover colunas de uma tabela.

MODIFY COLUMN: A cláusula **MODIFY COLUMN** é usada para modificar o tipo de dado de uma coluna e sua restrição em uma tabela. Você especifica o tipo de dado da coluna e sua nova restrição.

Comandos de Definição de Dados (DDL)

ALTER: Permite modificar a estrutura de objetos existentes, como adicionar ou remover colunas de uma tabela.

Modificar o tipo de dados de uma coluna:

ALTER TABLE local MODIFY COLUMN nome_da_coluna novo_tipo_da_coluna;

ALTER TABLE local MODIFY COLUMN Capacidade INT;

Comandos de Definição de Dados (DDL)

ALTER: Permite modificar a estrutura de objetos existentes, como adicionar ou remover colunas de uma tabela.

RENAME COLUMN: A cláusula é usada para renomear uma coluna existente em uma tabela. Você especifica o nome da coluna que deseja renomear e o novo nome que deseja atribuir a ela.

Comandos de Definição de Dados (DDL)

ALTER: Permite modificar a estrutura de objetos existentes, como adicionar ou remover colunas de uma tabela.

Renomear uma coluna:

ALTER TABLE local RENAME COLUMN nome_antigo TO nome_novo;

ALTER TABLE local RENAME COLUMN Endereco TO Endereço;

Comandos de Definição de Dados (DDL)

ALTER: Permite modificar a estrutura de objetos existentes, como adicionar ou remover colunas de uma tabela.

Adicionar uma restrição de chave primária:

**ALTER TABLE local ADD PRIMARY KEY
(nome_da_coluna);**

**ALTER TABLE local ADD id INT AUTO_INCREMENT
PRIMARY KEY;**

Comandos de Definição de Dados (DDL)

ALTER: Permite modificar a estrutura de objetos existentes, como adicionar ou remover colunas de uma tabela.

ADD CONSTRAINT: Adiciona uma nova restrição à tabela (chave estrangeira, restrição única e restrição de verificação).

Comandos de Definição de Dados (DDL)

ALTER: Permite modificar a estrutura de objetos existentes, como adicionar ou remover colunas de uma tabela.

Adicionar uma restrição de chave estrangeira:

```
ALTER TABLE local ADD CONSTRAINT nome_da_restricao FOREIGN KEY (nome_da_coluna)
REFERENCES outra_tabela(coluna_referenciada);
```

```
ALTER TABLE `evento` ADD CONSTRAINT `fk_Evento_Local` FOREIGN KEY
(`Local_id_local`) REFERENCES `local` (`id_local`) ON DELETE NO ACTION ON UPDATE NO
ACTION;
```

Comandos de Definição de Dados (DDL)

ALTER: Permite modificar a estrutura de objetos existentes, como adicionar ou remover colunas de uma tabela.

Adicionar uma restrição UNIQUE:

**ALTER TABLE local ADD CONSTRAINT nome_da_restricao
UNIQUE (nome_da_coluna);**

**ALTER TABLE local ADD CONSTRAINT unique_endereco
UNIQUE (Endereço);**

Comandos de Definição de Dados (DDL)

ALTER: Permite modificar a estrutura de objetos existentes, como adicionar ou remover colunas de uma tabela.

Adicionar uma restrição CHECK:

**ALTER TABLE local ADD CONSTRAINT nome_da_restricao
CHECK (condição);**

**ALTER TABLE local ADD CONSTRAINT check_capacidade
CHECK (Capacidade > 0);**

Comandos de Definição de Dados (DDL)

ALTER: Permite modificar a estrutura de objetos existentes, como adicionar ou remover colunas de uma tabela.

RENAME TO : A cláusula é usada para renomear uma tabela existente. Você especifica o nome atual da tabela e o novo nome que deseja atribuir a ela.

Comandos de Definição de Dados (DDL)

ALTER: Permite modificar a estrutura de objetos existentes, como adicionar ou remover colunas de uma tabela.

Renomear a tabela:

ALTER TABLE local RENAME TO novo_nome;

ALTER TABLE local RENAME TO Locais;

Comandos de Definição de Dados (DDL)

DROP: Permite remover a estrutura de objetos existentes, como banco de dados , tabelas e etc.

Remove um banco de dados inteiro e todos os seus objetos:

DROP DATABASE nome_do_banco_de_dados;

Remove uma tabela inteira do banco de dados, incluindo todos os seus dados, índices e restrições.

DROP TABLE nome_da_tabela

Utilizados para inserir, atualizar e excluir dados nas tabelas.

Comandos: INSERT, UPDATE, DELETE.

INSERT: Adiciona novos registros a uma tabela

Inserção Básica;

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

```
INSERT INTO Clientes (Nome, Sobrenome, Email)
VALUES ('João', 'Silva', 'joao.silva@email.com');
```

INSERT: Adiciona novos registros a uma tabela

Inserção Implícita ;

```
INSERT INTO nome_da_tabela  
VALUES (valor1, valor2, valor3, ...);
```

```
INSERT INTO Clientes  
VALUES ('Maria', 'Oliveira', 'maria.oliveira@email.com');
```

INSERT: Adiciona novos registros a uma tabela

Inserção de Valores em Colunas Específicas:

```
INSERT INTO nome_da_tabela (coluna1, coluna3)  
VALUES (valor1, valor3);
```

```
INSERT INTO Clientes (Nome, Email)  
VALUES ('Carlos', 'carlos@email.com');
```

INSERT: Adiciona novos registros a uma tabela

Inserção de Múltiplos Registros :

INSERT INTO nome_da_tabela (coluna1, coluna2, ...)

**VALUES (valor1, valor2, ...),
(valorA, valorB, ...),
(valorX, valorY, ...);**

INSERT INTO Clientes (Nome, Sobrenome)

**VALUES ('Ana', 'Souza'),
('Pedro', 'Santos'),
('Lucas', 'Ferreira');**

Comandos de Definição de Dados (DDL)

ALTER: Permite modificar a estrutura de objetos existentes, como adicionar ou remover colunas de uma tabela.

SELECT : Pode ser considerado tanto um comando quanto uma cláusula, dependendo do contexto. No exemplo a seguir, ele será utilizado para selecionar quais colunas terão seus dados copiados para as tabelas de destino.

Comandos de Definição de Dados (DDL)

ALTER: Permite modificar a estrutura de objetos existentes, como adicionar ou remover colunas de uma tabela.

FROM :A cláusula FROM é utilizada para especificar a(s) tabela(s) ou outras fontes de dados de onde o banco de dados deve recuperar ou manipular informações. Ela desempenha um papel essencial ao estabelecer o contexto de origem dos dados em diversas operações SQL. Em operações que envolvem dados relacionais, como a seleção de dados (com a instrução SELECT), exclusão de dados (através de DELETE) e junção de tabelas (com a cláusula JOIN), a cláusula FROM define a base fundamental de onde os dados serão extraídos ou manipulados.

Comandos de Definição de Dados (DDL)

ALTER: Permite modificar a estrutura de objetos existentes, como adicionar ou remover colunas de uma tabela.

WHERE : A cláusula WHERE é usada para especificar condições que restringem as linhas afetadas por uma operação SQL. Ela permite que você filtre dados em consultas SELECT, mas também é essencial para definir quais linhas serão modificadas em operações UPDATE ou excluídas em operações DELETE.

INSERT: Adiciona novos registros a uma tabela

Inserção de Dados de Outra Tabela :

INSERT INTO tabela_destino (coluna1, coluna2, ...)

SELECT colunaA, colunaB, ...

FROM tabela_origem

WHERE condição;

INSERT: Adiciona novos registros a uma tabela

Inserção de Dados de Outra Tabela :

INSERT INTO Clientes_Backup (Nome, Sobrenome)

SELECT Nome, Sobrenome

FROM Clientes

WHERE Sobrenome LIKE 'S%';

UPDATE : Modifica registros existentes em uma tabela

Atualização Simples:

UPDATE nome_da_tabela

SET coluna1 = valor1, coluna2 = valor2, ...

WHERE condição;

UPDATE Clientes

SET Cidade = 'São Paulo'

WHERE ID_Cliente = 1;

Comandos de Definição de Dados (DDL)

ALTER: Permite modificar a estrutura de objetos existentes, como adicionar ou remover colunas de uma tabela.

SET: A cláusula SET é usada para definir os novos valores das colunas durante uma atualização no banco de dados. Ela é essencial para modificar dados existentes, especificando quais campos serão alterados em comandos como UPDATE.

ALTER: Permite modificar a estrutura de objetos existentes, como adicionar ou remover colunas de uma tabela.

INNER JOIN : A cláusula INNER JOIN é usada para combinar registros de duas ou mais tabelas com base em uma condição de correspondência. Ela retorna apenas as linhas em que há correspondência entre as tabelas, excluindo registros sem correspondência. É essencial para realizar junções entre dados relacionados.

UPDATE : Modifica registros existentes em uma tabela

Atualização com Base em Outra Tabela:

UPDATE tabela1

SET tabela1.coluna = tabela2.coluna

FROM tabela1

INNER JOIN tabela2 ON tabela1.chave_primaria = tabela2.chave_estrangeira

WHERE condição;

UPDATE : Modifica registros existentes em uma tabela

Atualização com Base em Outra Tabela:

UPDATE Pedidos

SET Pedidos.NomeCliente = Clientes.Nome

FROM Pedidos

INNER JOIN Clientes ON Pedidos.ID_Cliente = Clientes.ID_Cliente;

UPDATE : Modifica registros existentes em uma tabela

Atualização Condicional:

UPDATE nome_da_tabela

SET coluna = novo_valor

WHERE condição;

UPDATE Produtos

SET Preço = Preço * 1.10

WHERE Categoria = 'Eletrônicos';

UPDATE : Modifica registros existentes em uma tabela

Atualização de Múltiplas Colunas:

UPDATE nome_da_tabela

SET coluna1 = valor1, coluna2 = valor2, ...

WHERE condição;

UPDATE Funcionários

SET Salário = Salário * 1.05, Cargo = 'Gerente'

WHERE Departamento = 'Vendas';

DELETE :Exclusão de múltiplos registros

DELETE Básico :

```
DELETE FROM nome_da_tabela  
WHERE condição;
```

```
DELETE FROM Clientes  
WHERE Cidade = 'São Paulo';
```

DELETE :Exclusão de múltiplos registros

DELETE Sem WHERE :

DELETE FROM nome_da_tabela;

DELETE FROM Clientes;

Utilizada para consultar e recuperar dados de tabelas.

Comando: SELECT.

Para uma melhor compreensão do comando de Consulta de Dados (DQL), vamos organizá-lo em quatro categorias principais:

Operadores Relacionais, Condicionais, Lógicos e Combinação de Operadores. Além disso, incluiremos exemplos de como realizar a ordenação dos resultados com o comando ORDER BY, assim como exemplos de consultas sem ordenação.

Ordenação:

ASC (Crescente): Ordena os dados de forma crescente, do menor para o maior (por padrão, se não especificado):

SELECT Nome, Capacidade

FROM local

ORDER BY Capacidade ASC;

Ordenação:

DESC (Decrescente): Ordena os dados de forma decrescente, do maior para o menor:

SELECT Nome, Capacidade

FROM local

ORDER BY Capacidade DESC;

Ordenação:

Ordenação por múltiplas colunas: Permite ordenar os dados por mais de uma coluna. O SQL ordena primeiro pela primeira coluna e, em caso de empate, usa a segunda coluna, e assim por diante:

SELECT Nome, Capacidade, Endereco

FROM local

ORDER BY Capacidade DESC, Nome ASC;

Operadores Relacionais:

= (Igual a): Compara se dois valores são iguais.

> (Maior que): Compara se um valor é maior que outro.

< (Menor que): Compara se um valor é menor que outro.

>= (Maior ou igual a): Compara se um valor é maior ou igual a outro.

<= (Menor ou igual a): Compara se um valor é menor ou igual a outro.

!= (Diferente de): Compara se dois valores são diferentes.

Operadores Relacionais:

= (Igual a): Compara se dois valores são iguais:

SELECT *

FROM tabela

WHERE coluna = dado_da_coluna;

SELECT *

FROM evento

WHERE Local_id_local = 2 ORDER BY `data_inicio` DESC;

Operadores Relacionais:

> (Maior que): Compara se um valor é maior que outro:

SELECT *

FROM tabela

WHERE coluna > 'dato_da_coluna_com_data_ou_valor';

SELECT *

FROM evento

WHERE data_inicio > '2025-06-01';

Operadores Relacionais:

>= (Maior ou igual a): Compara se um valor é maior ou igual a outro:

SELECT *

FROM tabela

WHERE coluna >= 'dato_da_coluna_com_data_ou_valor';

SELECT *

FROM evento

WHERE data_inicio >= '2025-06-01';

Operadores Relacionais:

< (Menor que): Compara se um valor é menor que outro.

SELECT *

FROM tabela

WHERE coluna < 'dato_da_coluna_com_data_ou_valor';

SELECT *

FROM evento

WHERE data_inicio < '2025-06-01';

Operadores Relacionais:

<= (Menor ou igual a): Compara se um valor é menor ou igual a outro.

SELECT *

FROM tabela

WHERE coluna <= ' dado_da_coluna_com_data_ou_valor';

SELECT *

FROM local

WHERE Capacidade <= 100;

Operadores Relacionais:

!= (Diferente de): Compara se dois valores são diferentes.

SELECT *

FROM tabela

WHERE coluna != ' dado_da_coluna' ;

SELECT *

FROM inscricao

WHERE Inscricaocol != 'Confirmado';

Operadores Condicionais:

BETWEEN: Seleciona valores dentro de um intervalo especificado.

IN: Seleciona valores que correspondem a qualquer valor em uma lista especificada.

LIKE: Seleciona valores que correspondem a um padrão especificado.

IS NULL: Seleciona valores nulos.

IS NOT NULL: Seleciona valores não nulos.

Operadores Condicionais:

BETWEEN: Seleciona valores dentro de um intervalo especificado.

```
SELECT *
```

```
FROM evento
```

```
WHERE data_inicio BETWEEN '2025-06-01' AND '2025-10-01';
```

```
SELECT *
```

```
FROM evento
```

```
WHERE data_inicio >= '2025-06-01' AND data_inicio <= '2025-10-01';
```

Operadores Condicionais:

BETWEEN: Seleciona valores dentro de um intervalo especificado.

SELECT *

FROM evento

**WHERE data_inicio BETWEEN '2025-06-01' AND
'2025-10-01' OR data_inicio > '2025-12-01';**

Operadores Condicionais:

NOT BETWEEN: Valores diferentes do dentro de um intervalo especificado.

SELECT *

FROM evento

**WHERE data_inicio NOT BETWEEN '2025-06-01'
AND '2025-10-01';**

Operadores Condicionais:

IN: Seleciona valores que correspondem a qualquer valor em uma lista especificada:

SELECT *

FROM tabela

WHERE coluna IN ('valor1', 'valor2', 'valor3');

SELECT *

FROM participante

WHERE nome IN ('Maria', 'Carlos', 'Ana');

Operadores Condicionais:

IN: Seleciona valores que correspondem a qualquer valor em uma lista especificada:

SELECT *

FROM evento

WHERE data_inicio IN ('2025-06-01', '2025-07-15', '2025-08-20');

Operadores Condicionais:

IN: Seleciona valores que correspondem a qualquer valor em uma lista especificada:

```
SELECT *  
FROM inscricao  
WHERE Evento_idEvento IN (2, 4, 5)  
AND Participante_idParticipante IN (1, 3, 5);
```

Operadores Condicionais:

IN: Seleciona valores que correspondem a qualquer valor em uma lista especificada:

```
SELECT *
FROM inscricao
WHERE Evento_idEvento IN (2, 4, 5)
OR Participante_idParticipante IN (1, 3, 5)
OR Inscricaocol = 'Confirmado';
```

Operadores Condicionais:

LIKE: Seleciona valores que correspondem a um padrão:

SELECT *

FROM tabela

WHERE coluna LIKE 'começa_com%';

SELECT *

FROM participante

WHERE Nome LIKE 'Maria%';

Operadores Condicionais:

LIKE: Seleciona valores que correspondem a um padrão:

SELECT *

FROM tabela

WHERE coluna LIKE '%contém%';

SELECT *

FROM evento

WHERE Nome LIKE '%Maria%';

Operadores Condicionais:

IS NULL: Seleciona valores nulos:

```
SELECT *  
FROM evento  
WHERE descricao IS NULL;
```

Operadores Condicionais:

IS NOT NULL: Seleciona valores não nulos:

```
SELECT *  
FROM inscricao  
WHERE Inscricaocol IS NOT NULL;
```

Operadores Lógicos :

AND: Retorna verdadeiro se ambas as condições forem verdadeiras.

OR: Retorna verdadeiro se pelo menos uma das condições for verdadeira.

NOT: Inverte o resultado de uma condição.

Operadores Lógicos :

AND: Retorna verdadeiro se ambas as condições forem verdadeiras:

SELECT *

FROM local

WHERE Capacidade > 50

AND Endereco LIKE '%Avenida%';

Operadores Lógicos :

AND: Retorna verdadeiro se ambas as condições forem verdadeiras:

SELECT *

FROM inscricao

WHERE Inscricaocol = 'Confirmado'

AND data_inscricao < '2025-06-01 00:00:00';

Operadores Lógicos :

AND: Retorna verdadeiro se ambas as condições forem verdadeiras:

```
SELECT *
FROM evento
WHERE data_inicio > '2025-06-01'
AND Local_id_local IN (SELECT id_local FROM local WHERE
Capacidade > 100);
```

Operadores Lógicos :

OR: Retorna verdadeiro se pelo menos uma das condições for verdadeira:

```
SELECT *  
FROM evento  
WHERE Local_id_local = 2  
OR Local_id_local = 5;
```

Operadores Lógicos :

OR: Retorna verdadeiro se pelo menos uma das condições for verdadeira:

SELECT *

FROM participante

WHERE Nome = 'Carlos Souza'

OR Email = 'julia.costa@email.com';

Operadores Lógicos :

OR: Retorna verdadeiro se pelo menos uma das condições for verdadeira:

```
SELECT *  
FROM evento  
WHERE data_inicio BETWEEN '2025-05-01' AND '2025-05-31'  
OR data_inicio BETWEEN '2025-09-01' AND '2025-09-30';
```

Operadores Lógicos :

OR: Retorna verdadeiro se pelo menos uma das condições for verdadeira:

```
SELECT *
```

```
FROM inscricao
```

```
WHERE Participante_idParticipante IN (
```

```
    SELECT idParticipante FROM participante WHERE Nome = 'Maria Oliveira'
```

```
)
```

```
OR Participante_idParticipante IN (
```

```
    SELECT idParticipante FROM participante WHERE Nome = 'Carlos Souza'
```

```
);
```

Operadores Lógicos :

NOT: Inverte o resultado de uma condição:

SELECT *

FROM evento

WHERE NOT Local_id_local = 2;

Operadores Lógicos :

NOT: Inverte o resultado de uma condição:

```
SELECT *
```

```
FROM participante
```

```
WHERE NOT Email = 'julia.costa@email.com';
```

Operadores Lógicos :

NOT: Inverte o resultado de uma condição:

```
SELECT *
```

```
FROM evento
```

```
WHERE NOT (data_inicio BETWEEN '2025-08-01' AND '2025-08-31');
```

Referências Bibliográficas

MySQL. MySQL Documentation: Data Types. Disponível em:
<https://dev.mysql.com/doc/refman/8.0/en/data-types.html>.

Referências Bibliográficas

Elmasri, R.; Navathe, S. B. Sistemas de Banco de Dados. 7. ed.
São Paulo: Pearson, 2019.

NIELD, Thomas. Introdução à Linguagem SQL: Abordagem Prática Para Iniciantes. 1. ed. São Paulo: Novatec Editora, 26 abr. 2016.

KLINE, Kevin E.; KLINE, Daniel. SQL: O Guia Essencial - Manual de Referência Profissional. 3. ed. Rio de Janeiro: Alta Books, 15 set. 2010.