SQL com MySQL

Guia Completo para Iniciantes

Do básico ao avançado

O que é SQL?

> **SQL** significa **Structured Query Language**

📊 **Linguagem padrão** para gerenciar bancos de dados relacionais

🔍 **Permite consultar**, inserir, atualizar e deletar dados

💼 **Usada por empresas** em todo o mundo

🎯 **Simples de aprender**, poderosa de usar

O que é MySQL?

> **MySQL** é um Sistema de Gerenciamento de Banco de Dados

🆓 **Open Source** - Gratuito e de código aberto

⚡ **Rápido e confiável** - Alto desempenho

🌐 **Popular** - Usado por Facebook, Twitter, YouTube

💻 **Multiplataforma** - Windows, Linux, macOS

Conceitos Básicos

🗄️ Banco de Dados

Coleção organizada de dados

📝 Registro

Uma linha na tabela

📋 Tabela

Dados em linhas e colunas

🔤 Coluna

Campo de tipo específico

Tipos de Dados Principais

📊 Numéricos

INT - Números inteiros
DECIMAL - Decimais precisos
FLOAT - Decimais aproximados

✏️ Texto

VARCHAR(n) - Texto variável
TEXT - Texto longo
CHAR(n) - Texto fixo

📅 Data e Hora

DATE - Data (AAAA-MM-DD)
DATETIME - Data e hora
TIME - Apenas hora

✅ Outros

BOOLEAN - Verdadeiro/Falso
BLOB - Dados binários

SELECT - Consultando Dados

O comando **SELECT** busca dados de tabelas.

Selecionar todas as colunas:

```
SELECT * FROM clientes;
```

Selecionar colunas específicas:

```
SELECT nome, email, idade FROM clientes;
```

Com alias (apelido):

```
SELECT nome AS cliente, email AS contato
FROM clientes;
```

INSERT - Inserindo Dados

O comando **INSERT** adiciona novos registros.

Inserir um registro:

```
INSERT INTO clientes (nome, email, idade)
VALUES ('João Silva', 'joao@email.com', 30);
```

Inserir múltiplos registros:

```
INSERT INTO clientes (nome, email, idade) VALUES
('Maria Santos', 'maria@email.com', 25),
('Pedro Costa', 'pedro@email.com', 35);
```

> 💡 **Dica:** Sempre especifique as colunas!

UPDATE - Atualizando Dados

O comando **UPDATE** modifica registros existentes.

Atualizar um campo:

```
UPDATE clientes SET email = 'novo@email.com' WHERE id = 1;
```

Atualizar múltiplos campos:

```
UPDATE clientes SET email = 'novo@email.com', idade = 31
WHERE nome = 'João Silva';
```

⚠️ **Sempre use WHERE!** Sem ele, TODOS os registros serão atualizados.

DELETE - Deletando Dados

O comando **DELETE** remove registros de uma tabela.

Deletar um registro:

```
DELETE FROM clientes WHERE id = 5;
```

Deletar com condição:

```
DELETE FROM clientes WHERE idade < 18;
```

> ⚠️ **PERIGO: DELETE sem WHERE**
>
> Sem WHERE, TODOS os registros serão deletados!

WHERE - Filtrando Dados

A cláusula **WHERE** filtra registros.

Comparação:

```
WHERE idade >= 18
```

IN (lista):

```
WHERE cidade IN ('SP', 'RJ')
```

AND e OR:

```
WHERE idade >= 18 AND cidade = 'SP'
```

BETWEEN (intervalo):

```
WHERE idade BETWEEN 18 AND 30
```

LIKE (padrões):

```
WHERE nome LIKE 'João%'
```

IS NULL:

```
WHERE email IS NULL
```

ORDER BY e LIMIT

ORDER BY - Ordenar resultados

Ordem crescente (padrão):

```
SELECT * FROM clientes ORDER BY nome;
```

Ordem decrescente:

```
SELECT * FROM clientes ORDER BY idade DESC;
```

LIMIT - Limitar resultados

Primeiros N registros:

```
SELECT * FROM clientes LIMIT 10;
```

JOINs - Relacionando Tabelas

JOIN combina linhas de duas ou mais tabelas baseado em uma coluna relacionada.

Por que usar JOINs?

- ✓ Evitar duplicação de dados
- ✓ Manter dados organizados
- ✓ Relacionar informações
- ✓ Consultas mais eficientes

Exemplo básico:

```
SELECT clientes.nome,
pedidos.data
FROM clientes
JOIN pedidos
ON clientes.id =
pedidos.cliente_id;
```

Tipos de JOINs

INNER JOIN

Retorna apenas registros com correspondência em ambas tabelas

RIGHT JOIN

Retorna todos da tabela direita + correspondências da esquerda

LEFT JOIN

Retorna todos da tabela esquerda + correspondências da direita

FULL JOIN

Retorna todos os registros quando há correspondência em qualquer tabela

Funções Agregadas

Funções que realizam cálculos em conjunto de valores.

COUNT - Contar registros

```
SELECT COUNT(*) FROM clientes;
```

SUM - Somar valores

```
SELECT SUM(preco) FROM produtos;
```

AVG - Média

```
SELECT AVG(idade) FROM clientes;
```

MAX - Valor máximo

```
SELECT MAX(preco) FROM produtos;
```

MIN - Valor mínimo

```
SELECT MIN(preco) FROM produtos;
```

GROUP BY e HAVING

GROUP BY - Agrupar resultados

Agrupa linhas com valores iguais em colunas específicas.

```
SELECT cidade, COUNT(*) AS total
FROM clientes
GROUP BY cidade;
```

HAVING - Filtrar grupos

Como WHERE, mas para resultados agrupados.

```
SELECT cidade, COUNT(*) AS total
FROM clientes
GROUP BY cidade
HAVING total > 5;
```

Chaves e Relacionamentos

🔑 PRIMARY KEY (Chave Primária)

Identifica UNICAMENTE cada registro

```
id INT PRIMARY KEY AUTO_INCREMENT
```

🔗 FOREIGN KEY (Chave Estrangeira)

Cria relacionamento com outra tabela

```
FOREIGN KEY (cliente_id) REFERENCES clientes(id)
```

Boas Práticas

✅ Use nomes descritivos

✅ Sempre use WHERE

✅ Use índices

✅ Faça backups

❌ Evite SELECT *

❌ Não armazene senhas em texto

❌ Cuidado com SQL Injection

❌ Não ignore erros

Resumo

✅ SQL é a linguagem padrão para bancos de dados

✅ MySQL é um SGBD popular e gratuito

✅ Comandos principais: SELECT, INSERT, UPDATE, DELETE

✅ Use JOINs para relacionar tabelas

✅ Funções agregadas para análise de dados

Próximos passos: Pratique muito! 🚀