

Code & Query

A Jornada SQL do Zero

SQL

Seu Guia Completo para Domínio de Banco de Dados

O COMEÇO DA JORNADA

Code & Query - Augusto Belussi

Entendendo a Linguagem SQL

O que é SQL?

SQL (*Structured Query Language*) é a linguagem usada para se comunicar com bancos de dados. Ela permite que você **peça informações, adicione novos dados, modifique registros existentes e até crie estruturas inteiras dentro do banco.**

Mesmo que pareça técnico, pense em SQL como “frases” que você diz para o banco de dados executar uma ação.



A Importância do SQL no Mundo Real

Por que SQL é tão importante?

Praticamente tudo que usamos no dia a dia depende de informação armazenada:

- Redes sociais usam bancos de dados para guardar perfis e posts.
- Apps de delivery armazenam pedidos, restaurantes e entregadores.
- Bancos registram transações, contas e clientes.

Sem SQL, esses sistemas não teriam como guardar, buscar e organizar essas informações. Aprender SQL significa entender como os dados se movimentam no “coração” de qualquer aplicação moderna.



2

ENTENDENDO O BANCO DE DADOS

O que é um Banco de Dados?

A “Biblioteca de Dados”

Um banco de dados é um sistema criado para **armazenar informações de forma organizada**. Imagine uma grande biblioteca: ao invés de livros, temos **dados**. Cada parte desse sistema serve para facilitar o acesso rápido e seguro às informações..

Estrutura Fundamental

Tabelas

As tabelas são as estruturas principais do banco. Cada tabela armazena informações sobre um assunto específico .Exemplos de tabelas comuns: **clientes, produtos, pedidos**.

Colunas e Tipos de Dados

As colunas definem *o tipo de informação* que será armazenada em cada tabela. Exemplo de colunas de uma tabela de clientes:

- **nome** → texto
- **idade** → número
- **email** → texto
- **data_cadastro** → data e hora

Esses tipos de dados garantem que cada informação seja armazenada corretamente.

Linhas (Registros)

Cada linha é um item real armazenado na tabela. Se a tabela é **clientes**, cada linha representa *um cliente específico*.

3

CONSULTANDO DADOS

SELECT – Buscando Informações

Comando SELECT

O comando **SELECT** é o mais usado no SQL, pois permite consultar informações de uma tabela.

Exemplo:

```
• • •      Code & Query - Augusto Belussi  
SELECT * FROM clientes;
```

O *asterisco* significa “traga todas as colunas”.

Selecionando colunas específicas

```
• • •      Code & Query - Augusto Belussi  
SELECT nome, idade FROM clientes;
```

Assim, você vê apenas as informações de nomes da tabela clientes.

Filtrando com WHERE

Usando WHERE

O **WHERE** permite buscar dados que atendam a uma condição:

Code & Query - Augusto Belussi

```
SELECT * FROM clientes  
WHERE idade > 18;
```

Combinando condições

Você pode combinar condições:

Code & Query - Augusto Belussi

```
WHERE idade > 18 AND cidade = 'São Paulo';
```

Buscando partes de um texto (LIKE)

Usando LIKE

Procura por pedaços de texto:



Code & Query - Augusto Belussi

```
SELECT * FROM produtos  
WHERE nome LIKE '%mouse%';
```

O símbolo % significa “qualquer sequência de caracteres”.

Criando e Inserindo Dados

```
CREATE TABLE clients (client_id INT PRIMARY KEY, name VARCHAR(50), address VARCHAR(100), phone VARCHAR(15), email VARCHAR(50), created_at DATETIME, updated_at DATETIME);

CREATE TABLE orders (order_id INT PRIMARY KEY, client_id INT, quantity INT, unit_price DECIMAL(10, 2), total DECIMAL(10, 2), status VARCHAR(20), created_at DATETIME, updated_at DATETIME);

CREATE TABLE order_items (item_id INT PRIMARY KEY, order_id INT, product_name VARCHAR(50), quantity INT, unit_price DECIMAL(10, 2), total DECIMAL(10, 2), created_at DATETIME, updated_at DATETIME);

CREATE TABLE products (product_id INT PRIMARY KEY, name VARCHAR(50), description TEXT, price DECIMAL(10, 2), stock INT, created_at DATETIME, updated_at DATETIME);

CREATE TABLE users (user_id INT PRIMARY KEY, first_name VARCHAR(50), last_name VARCHAR(50), email VARCHAR(50), password VARCHAR(100), role VARCHAR(20), created_at DATETIME, updated_at DATETIME);

CREATE TABLE roles (role_id INT PRIMARY KEY, name VARCHAR(50), description TEXT, created_at DATETIME, updated_at DATETIME);

CREATE TABLE categories (category_id INT PRIMARY KEY, name VARCHAR(50), description TEXT, created_at DATETIME, updated_at DATETIME);

CREATE TABLE categories_products (category_id INT, product_id INT, created_at DATETIME, updated_at DATETIME);

CREATE TABLE reviews (review_id INT PRIMARY KEY, user_id INT, product_id INT, rating INT, comment TEXT, created_at DATETIME, updated_at DATETIME);

CREATE TABLE payment_methods (method_id INT PRIMARY KEY, name VARCHAR(50), description TEXT, created_at DATETIME, updated_at DATETIME);

CREATE TABLE payments (payment_id INT PRIMARY KEY, order_id INT, method_id INT, amount DECIMAL(10, 2), status VARCHAR(20), created_at DATETIME, updated_at DATETIME);

CREATE TABLE shipping_addresses (address_id INT PRIMARY KEY, user_id INT, address_line1 VARCHAR(100), address_line2 VARCHAR(100), city VARCHAR(50), state VARCHAR(50), zip_code VARCHAR(20), created_at DATETIME, updated_at DATETIME);

CREATE TABLE tracking_numbers (number_id INT PRIMARY KEY, tracking_number VARCHAR(50), status VARCHAR(20), created_at DATETIME, updated_at DATETIME);

CREATE TABLE notifications (notification_id INT PRIMARY KEY, user_id INT, message TEXT, type VARCHAR(50), created_at DATETIME, updated_at DATETIME);

CREATE TABLE logins (login_id INT PRIMARY KEY, user_id INT, date_logged_in DATE, created_at DATETIME, updated_at DATETIME);

CREATE TABLE favorites (favorite_id INT PRIMARY KEY, user_id INT, product_id INT, created_at DATETIME, updated_at DATETIME);

CREATE TABLE cart_items (item_id INT PRIMARY KEY, user_id INT, product_id INT, quantity INT, created_at DATETIME, updated_at DATETIME);

CREATE TABLE cart_sessions (session_id INT PRIMARY KEY, user_id INT, created_at DATETIME, updated_at DATETIME);

CREATE TABLE wishlists (wishlist_id INT PRIMARY KEY, user_id INT, product_id INT, created_at DATETIME, updated_at DATETIME);
```

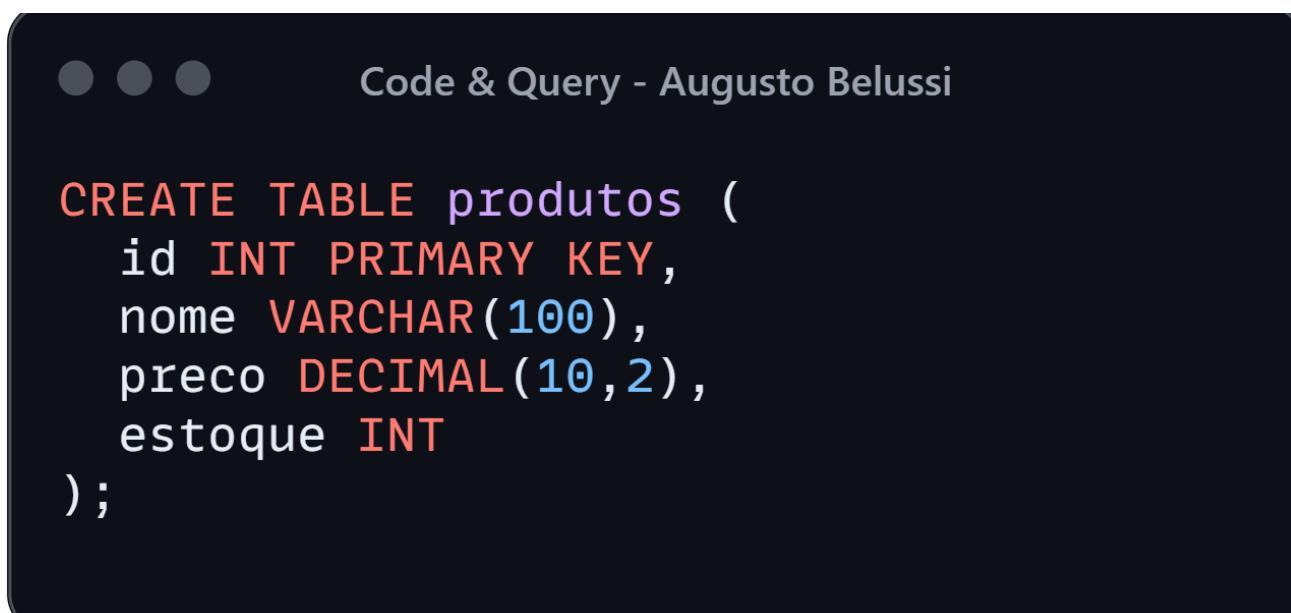
4



Criando Tabelas com CREATE TABLE

Definindo a estrutura

Ao criar uma tabela, você define suas colunas:



Code & Query - Augusto Belussi

```
CREATE TABLE produtos (
    id INT PRIMARY KEY,
    nome VARCHAR(100),
    preco DECIMAL(10,2),
    estoque INT
);
```

- **PRIMARY KEY** → identifica cada item de forma única
- **VARCHAR** → texto variável
- **DECIMAL** → números com casas decimais
- **INT** → números inteiros

Inserindo Registros com INSERT INTO

Adicionando novos dados



Code & Query - Augusto Belussi

```
INSERT INTO produtos (id, nome, preco, estoque)  
VALUES (1, 'Teclado Mecânico', 199.90, 50);
```

Inserindo vários valores de uma vez



Code & Query - Augusto Belussi

```
INSERT INTO produtos (id, nome, preco)  
VALUES  
(2, 'Mouse Gamer', 129.00),  
(3, 'Headset', 249.90);
```

Atualizando e Deletando Dados

Code & Query - Augusto Belussi

Atualizando Informações com UPDATE

Modificando registros



Code & Query - Augusto Belussi

```
UPDATE produtos  
SET preco = 179.90  
WHERE id = 1;
```

Sem o WHERE, *todas* as linhas seriam alteradas — por isso, cuidado!

Removendo Dados com DELETE

Excluindo registros



Code & Query - Augusto Belussi

```
DELETE FROM produtos  
WHERE id = 2;
```

Assim como no UPDATE, sempre inclua o WHERE para evitar apagar tudo sem querer.

6

Organizando e Classificando Dados

Ordenação com ORDER BY

Ordenando resultados



Code & Query - Augusto Belussi

```
SELECT nome, preco  
FROM produtos  
ORDER BY preco ASC;
```

- ASC → crescente
- DESC → decrescente

Limitando Resultados com LIMIT

Mostrar poucos resultados



Code & Query - Augusto Belussi

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

Ótimo para visualizar apenas parte dos dados.

Removendo resultados duplicados (DISTINCT)

Evitar repetições



Code & Query - Augusto Belussi

```
SELECT DISTINCT cidade  
FROM clientes;
```

Isso retorna cada cidade apenas uma vez.

Avançando um Pouco

Mais

Junções (JOINS)

Conectando tabelas

JOINS conectam dados de duas tabelas diferentes.

Exemplo de junção simples:



Code & Query - Augusto Belussi

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

Isso permite unir dados relacionados, como pedidos + nome do cliente.

Funções de agregação

Cálculos e resumos

Usadas para cálculos:

- **COUNT()** → conta registros
- **SUM()** → soma valores
- **AVG()** → média
- **MAX()** → maior valor
- **MIN()** → menor valor
- **AS** → definir um pseudônimo (ou alias) para a coluna resultante da operação

Exemplo:



Code & Query - Augusto Belussi

```
SELECT AVG(preco) AS preco_medio  
FROM produtos;
```

Conclusão da Jornada

```
5PM3
02/01/2009 3:05PM
is Ccs )
REJECT x FWDNL Inters + EOM/1)
Tot+ BEEF
9P+BOJ FWI
DWA+ T3
2/0:181913)
CG62 1 0.022WJ G085001.82285:..110
C0000 JH SOS2 theq{ltTSf{o tos001- T
) H6ZL!fctss '1130esr~ 15CLAC:3)
)
;1;
SELECT + FB6RA units WHERE
Ccs)
ORSTATE = 'active'
)
CREATE + FB6RA orders (order
 Sta status ~ 'active')
)
CREATE -1130ce -OSCLAD = (
)
CREATE TABLE orders: 6110003 Genes-21)
(1 cadet-10 INF POSSY M161 SG541
| ( GR511000000 101; DAI1AL - BE14L - E111E31A))
DO11A(10,11)
)
)
#2( «Jeo cltear-als
)
)
User error :
)
)
CREATE TABLE orders (unit: osmtf0181)
( order-10 INT -1130ce-2115. os11(nord)
)
(
(es60TS.18000z: edess (evedo)
oacseme-101 aomed1 $3215-5981x. Eme9f66166
aest16:11)
|
| * 180T-1130ce ekRE + - 1130ce-11. 18 e13
| ) caserio-1101m p1e joet et-1200p [800]
| ) omak21. 11-10(taes- t1eden)
| ) cesentee-1101 00AT see- [8un]
| ) 21\ (test)
)
)
)
SELECT logimpl
)
)
```

A large, stylized number 8 logo, rendered in a light gray color, is centered on a dark blue background. The logo has a thick, three-dimensional appearance with a slight shadow effect.



Revisão Geral

O que você aprendeu

Neste eBook você viu:

- Como funciona um banco de dados
- Estruturas como tabelas, colunas e registros
- Consultas com SELECT
- Filtros com WHERE
- Ordenação e organização dos resultados
- Como criar, inserir, atualizar e apagar dados
- Introdução a JOINs e funções avançadas

Mensagem Final

Encerramento e próximos passos

SQL é uma habilidade essencial e poderosa. Com ela, você domina a linguagem que controla os dados do mundo moderno.

Esse é apenas o primeiro passo — e o próximo capítulo dessa jornada é você quem escreve.

