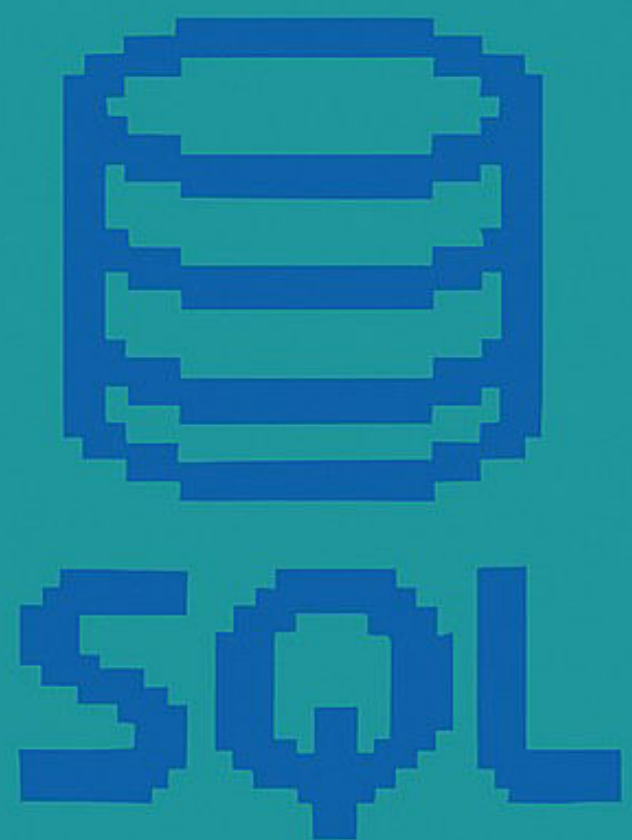
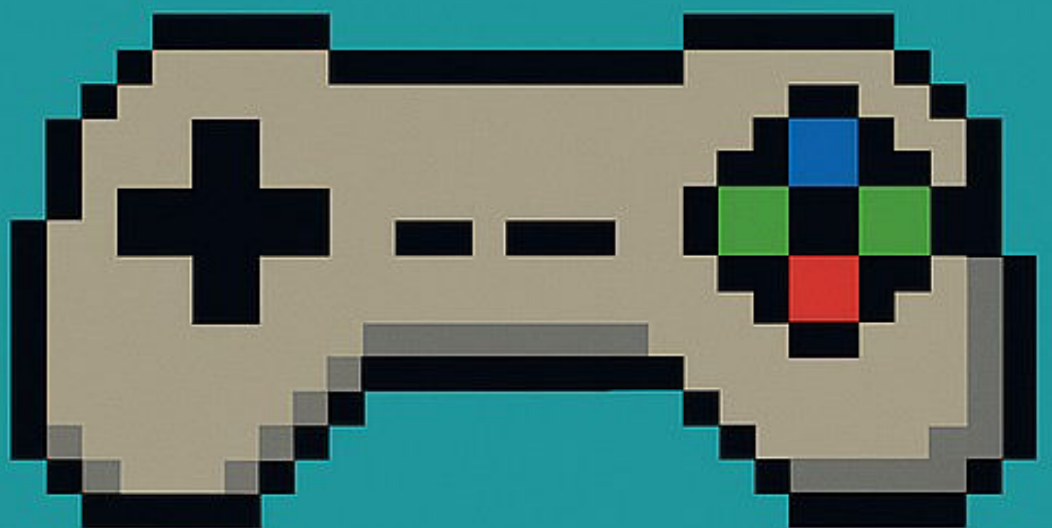
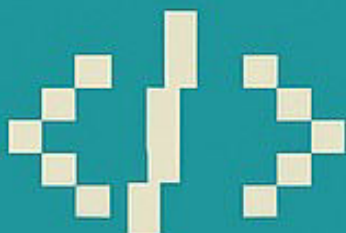
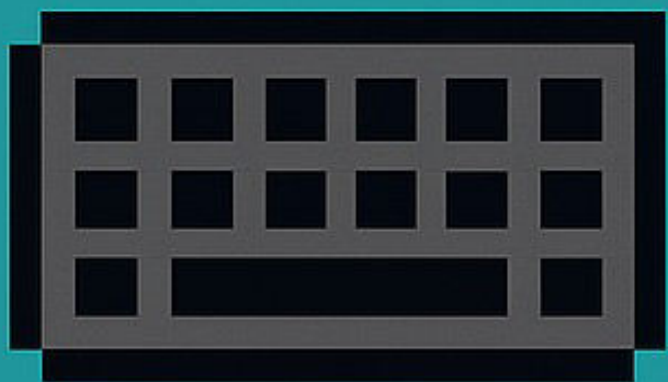
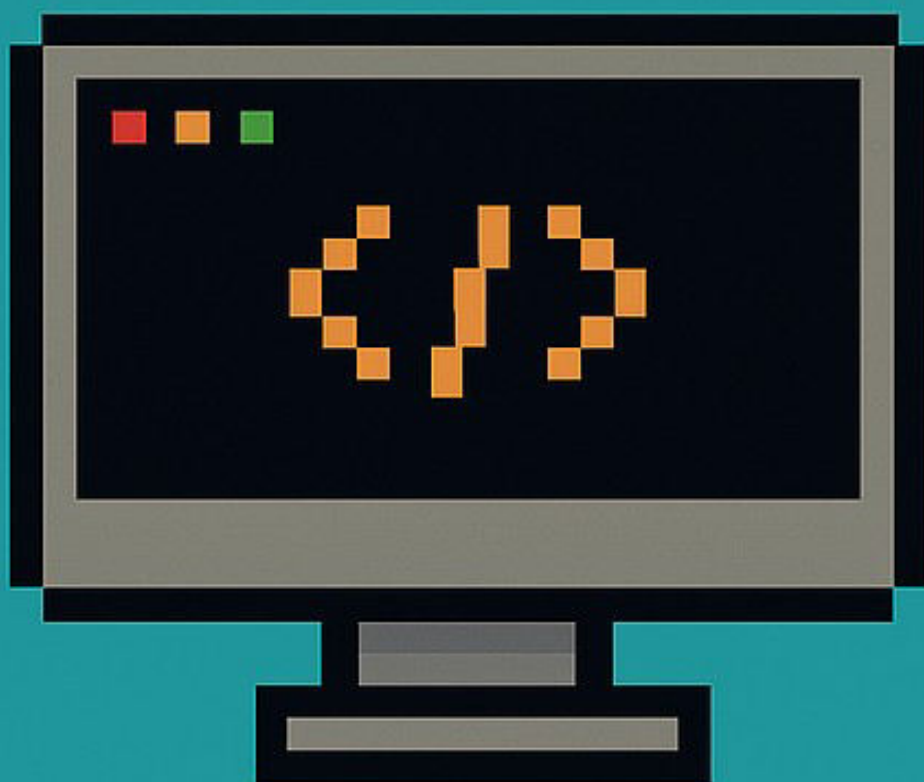


# SQL LEVEL 1

**Sua Primeira Missão no Mundo dos Dados**



# O QUE É UM BANCO DE DADOS?

---

Um banco de dados é um sistema usado para armazenar, organizar e acessar informações de forma rápida e segura. Ele funciona como uma estrutura digital onde você pode guardar dados importantes, como informações de clientes, produtos, pedidos, entre outros. Esses dados ficam salvos em tabelas, que se parecem com planilhas, e podem ser consultados, modificados ou excluídos por meio de linguagens como o SQL. O banco de dados é o cérebro por trás de praticamente qualquer sistema moderno, desde aplicativos de redes sociais até sites de lojas virtuais.

Para entender melhor, imagine um armário de roupas. O armário representa o banco de dados, e dentro dele há várias gavetas. Cada gaveta é como uma tabela, usada para guardar um tipo específico de item: uma gaveta pode ter camisetas, outra calças, outra meias, e assim por diante. Esses itens são os dados.

Quando você quer uma camiseta azul, por exemplo, você vai até a gaveta certa (a tabela) e procura pelo item com essa característica. Da mesma forma, o SQL permite que você procure exatamente pelos dados que precisa, na tabela correta do seu banco de dados.



Armário = Banco  
Gaveta = Tabelas  
Sapatos = Dados

# 01

## **Manipulando Dados com SQL**

# O que são os comandos de manipulação de dados

---

Para entender como programar em SQL, é importante primeiro entender como manipular os dados no mesmo, assim, você entenderá melhor a estrutura de um banco de dados, e como o mesmo funciona.;

Os comandos de manipulação de dados no SQL são: SELECT, INSERT, UPDATE e DELETE. é com eles que você lê, escreve, modifica e apaga informações em um banco de dados.

Imagine que o banco de dados é como uma planilha com várias tabelas, e esses comandos são as ações que você faz nela: ler uma linha, adicionar uma nova, corrigir algo errado ou remover um item que não é mais necessário. Eles são a base do SQL e quase todo sistema que usa banco de dados trabalha com eles nos bastidores.

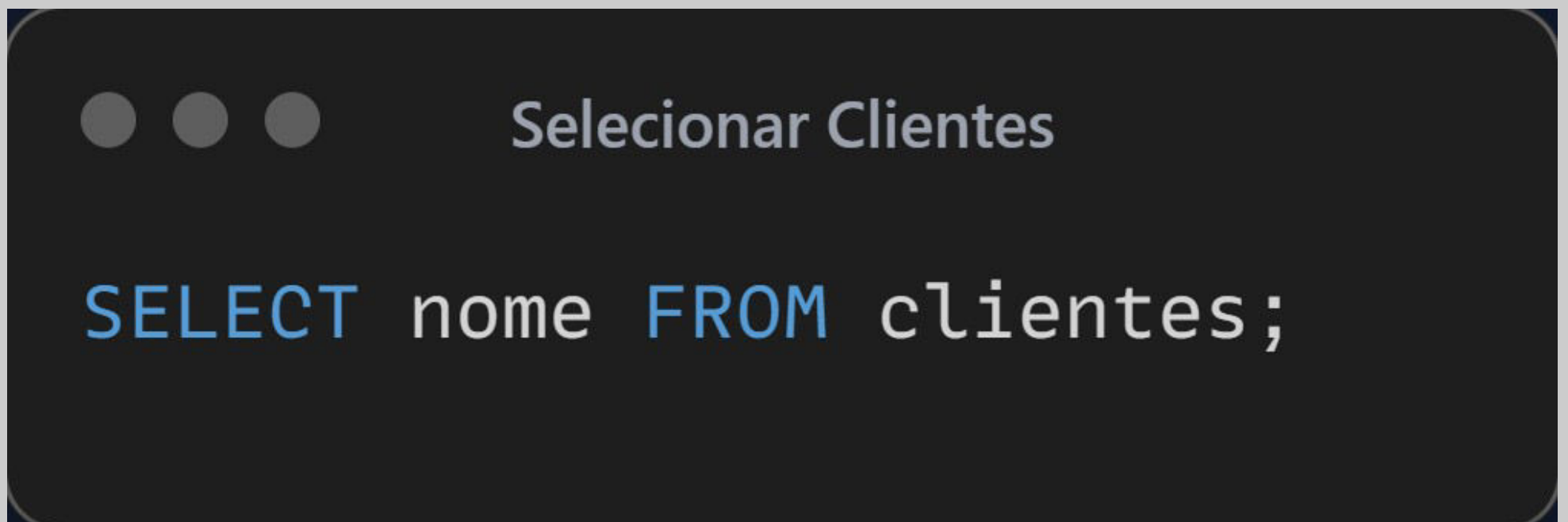


# SELECT — Consultar dados

---

O SELECT é usado para buscar informações que estão no banco. É como perguntar: “quais são os clientes cadastrados?”, ou “quais pedidos foram feitos hoje?”. Ele não muda nada no banco, só lê e mostra os dados.

No exemplo abaixo, estamos selecionando todos os nomes de todos os clientes cadastrados na tabela clientes

A terminal window with a dark background and rounded corners. At the top left, there are three small gray circles representing window control buttons. To their right, the title 'Selecionar Clientes' is written in a light gray font. The main area of the terminal displays the SQL query 'SELECT nome FROM clientes;' in a light blue monospace font. The word 'SELECT' is on the first line, and 'nome FROM clientes;' is on the second line.

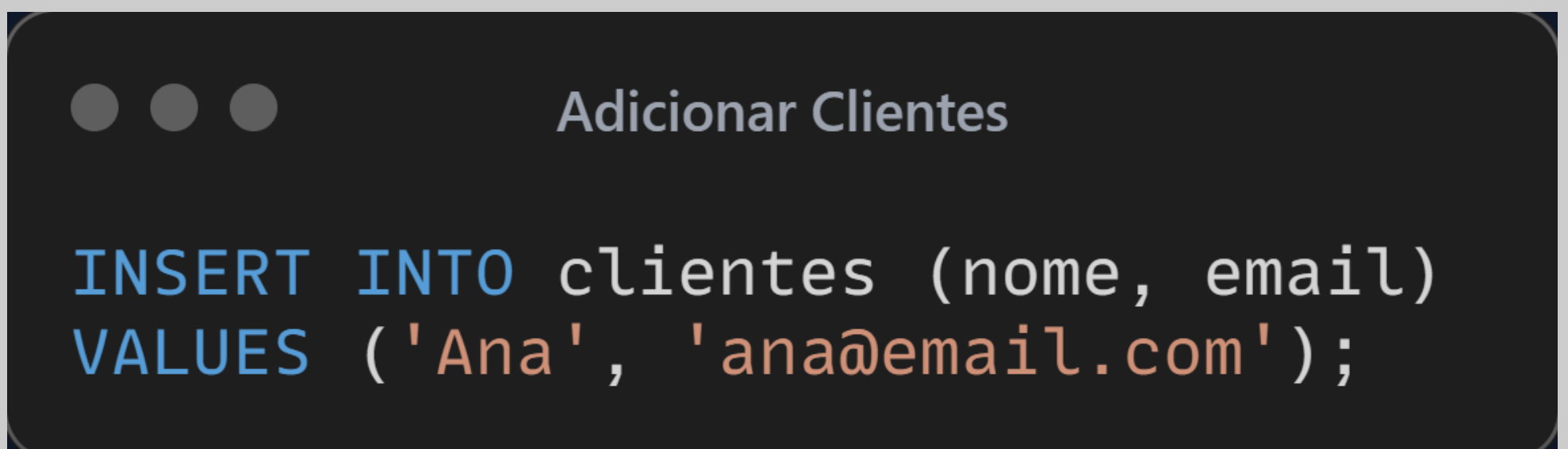
```
Selecionar Clientes  
SELECT nome FROM clientes;
```

# INSERT — Adicionar dados

---

O INSERT serve para inserir novas informações em uma tabela. Imagine que você acabou de cadastrar um novo cliente ou adicionou um novo produto ao sistema — é isso que o INSERT faz: ele grava um novo

No exemplo abaixo, nós estamos adicionando um novo cliente no sistema, adicionando o seu nome e o seu e-mail.



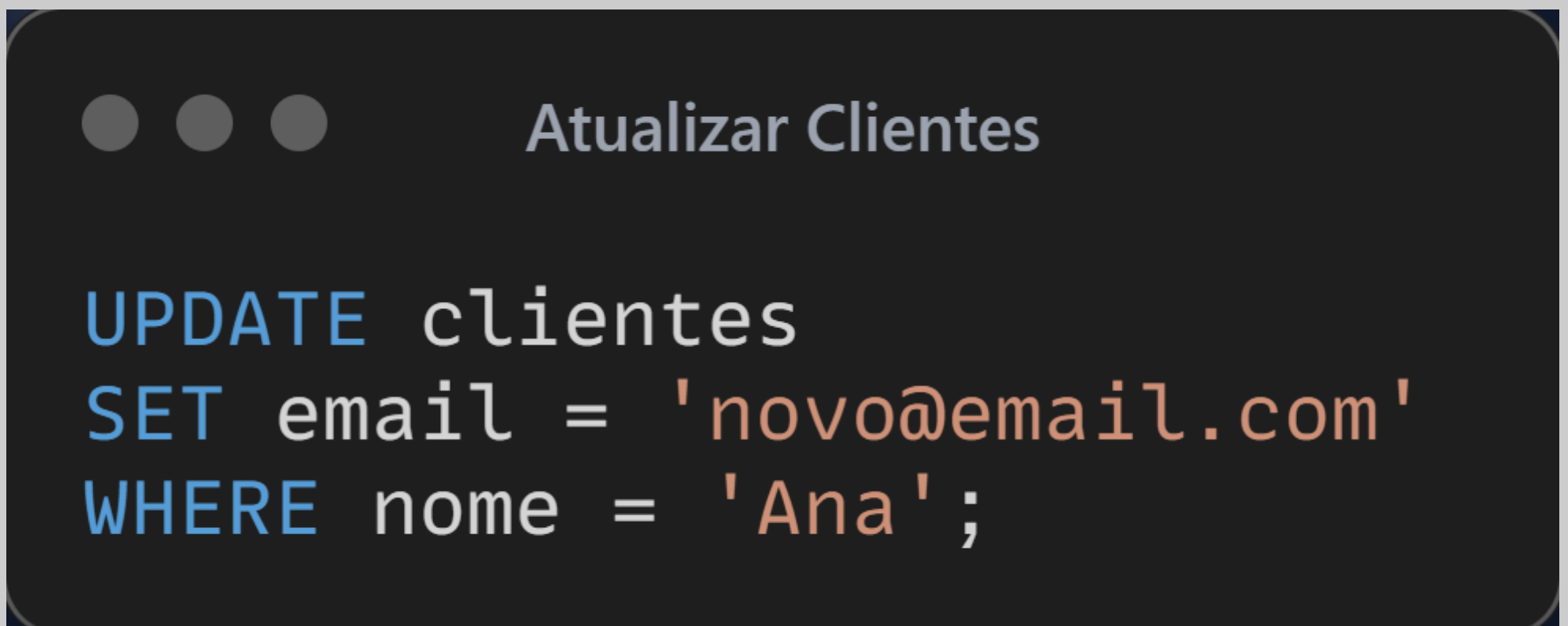
```
INSERT INTO clientes (nome, email)
VALUES ('Ana', 'ana@email.com');
```

# UPDATE — Atualizar dados

---

O UPDATE é usado para modificar informações que já estão no banco. Por exemplo, se o cliente trocou de email ou cidade, você usa o UPDATE para atualizar esse dado, mantendo o restante do cadastro.

No exemplo abaixo, estamos atualizando o e-mail do cliente chamado “Ana”



```
Atualizar Clientes

UPDATE clientes
SET email = 'novo@email.com'
WHERE nome = 'Ana';
```

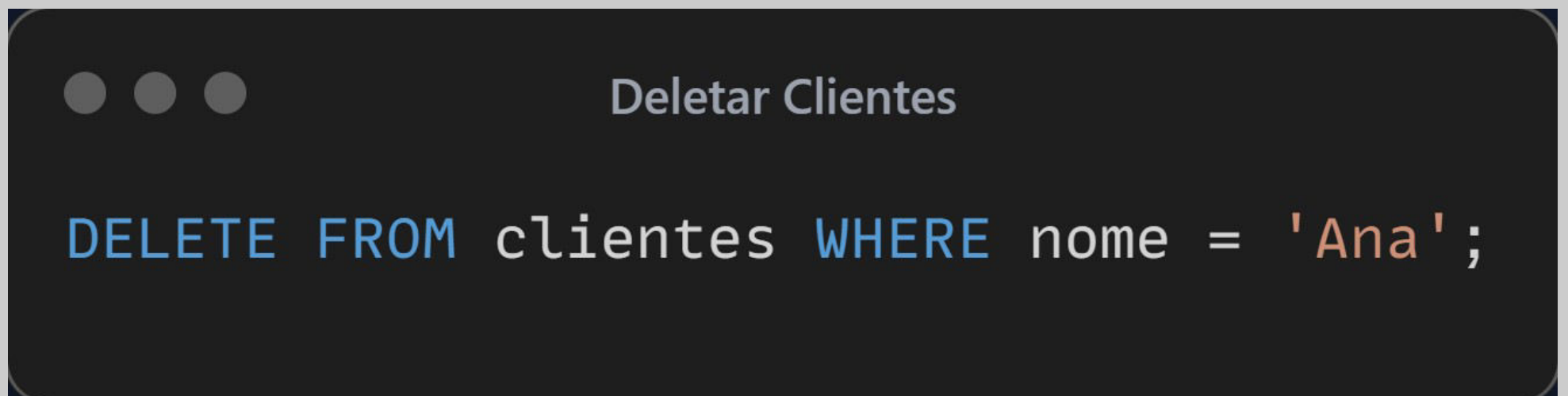


# DELETE — Apagar dados

---

O DELETE é o comando que você usa quando quer remover algo do banco, como apagar um cliente que cancelou a conta ou excluir um pedido com erro. É importante usar com cuidado, pois os dados são realmente eliminados.

No exemplo abaixo, estamos deletando por completo o cliente “Ana”

A terminal window with a dark background and rounded corners. At the top left, there are three small gray circles representing window control buttons. The title bar at the top center reads "Deletar Clientes" in a light gray font. The main area of the terminal displays a SQL command in a monospaced font: "DELETE FROM clientes WHERE nome = 'Ana';". The word "DELETE" is blue, "FROM" is light blue, "clientes" is white, "WHERE" is blue, "nome" is white, "=" is white, "'Ana'" is orange, and ";" is white.

```
DELETE FROM clientes WHERE nome = 'Ana';
```



02

# **Entendendo Tabelas em SQL**

# O que é uma Tabela no SQL

---

Uma tabela no SQL é como uma planilha digital dentro do banco de dados. Ela é usada para guardar dados de forma organizada em linhas e colunas.

Cada tabela representa um tipo de informação — por exemplo, uma tabela chamada clientes pode armazenar nome, email e cidade de várias pessoas.

As colunas representam os campos (ex: nome, idade, email), e cada linha representa um registro individual (ex: um cliente). Para que o banco de dados funcione bem, cada tabela precisa ser bem pensada: com os tipos de dados corretos, nomes claros, e uma estrutura que evite bagunça. Tudo começa com uma tabela bem feita.

Clientes
id
nome
email

# Colunas: Os Campos da Tabela

---

As colunas são como categorias que definem o tipo de dado que cada linha vai armazenar.

Por exemplo, a tabela clientes pode ter colunas como id, nome, email.

Cada coluna precisa de um tipo de dado, como texto (VARCHAR), número (INT) ou data (DATE), para o banco entender o que está sendo armazenado ali.

Abaixo vem um exemplo visual de uma coluna (a parte a direita) no SQL:

<i>Cliente</i>	
<i>id</i>	<i>INT PK</i>
<i>nome</i>	<i>VARCHAR(100)</i>
<i>email</i>	<i>VARCHAR(100)</i>

# Linhas: Os Registros da Tabela

---

Cada linha da tabela representa um registro completo de dados.

Se a tabela for de clientes, cada linha é um cliente diferente, com suas informações preenchidas nas colunas correspondentes.

Ou seja, enquanto as colunas são os tipos de dados, as linhas são os dados em si.

No exemplo abaixo, a parte de cima representa as colunas, e a parte de baixo as linhas do banco de dados

id	nome	email
1	Ana Oliveira	ana@email.com

# Chave Primária: O Identificador Único

---

Toda tabela precisa de um campo que funcione como identificador único — isso se chama chave primária (ou primary key).

Ela garante que cada linha seja única, como um número de identificação.

Por exemplo, o id do cliente pode ser a chave primária, assim como demonstrado nos capítulos anteriores.

Nenhum cliente pode ter o mesmo id que outro, pois caso isso ocorra, não teríamos como identificá-los individualmente

id	nome	email
1	Ana Oliveira	ana@email.com

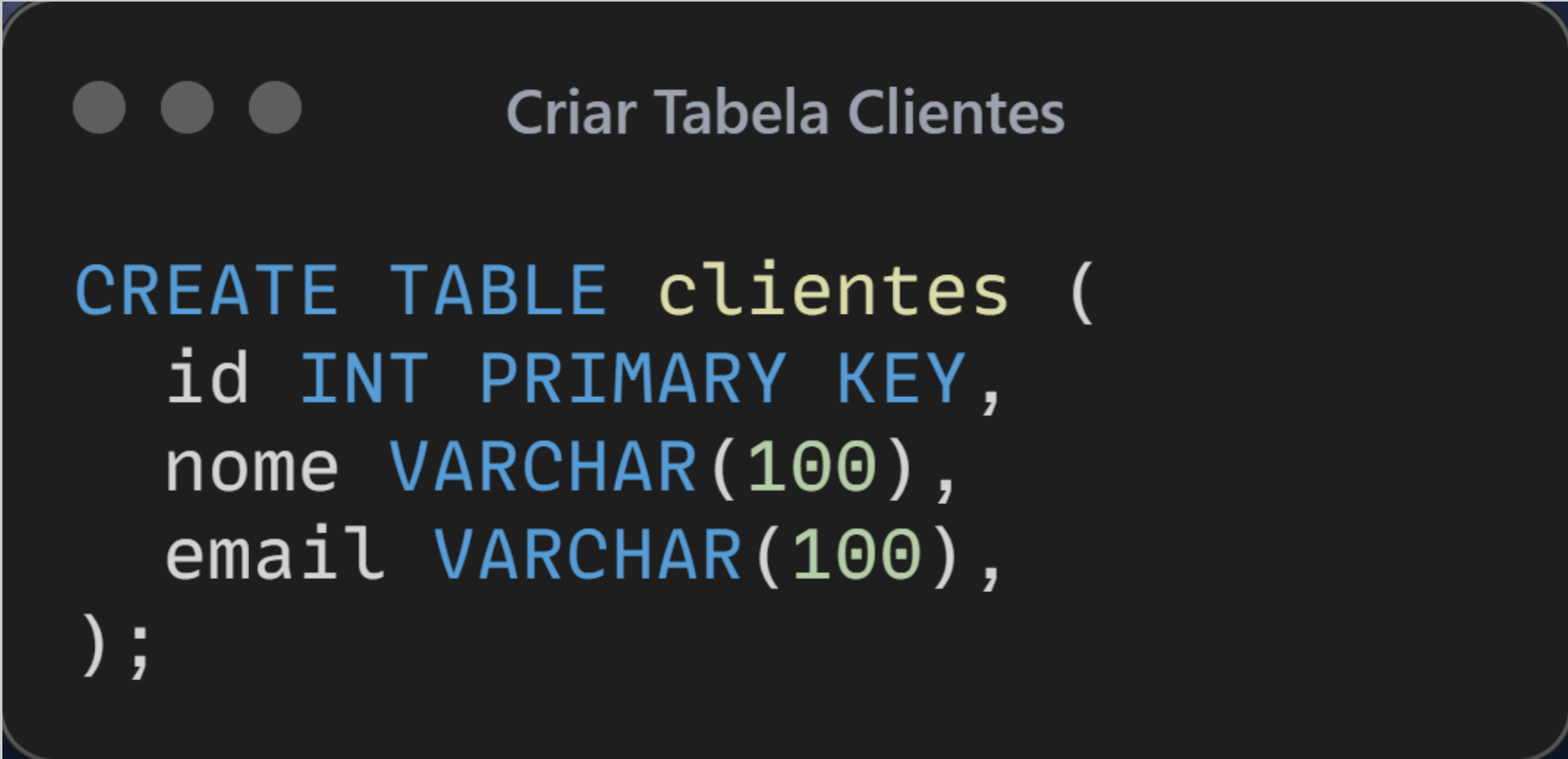
# CREATE TABLE: Criando uma Tabela no SQL

---

Para criar uma tabela no SQL, usamos o comando CREATE TABLE, onde definimos o nome da tabela, as colunas, seus tipos e a chave primária.

Para criar uma tabela, podemos utilizar as representações visuais mostradas nos capítulos anteriores. A mais complexa, no caso a que mostrava as colunas de INT, PK E VARCHAR, é chamada de MER (Modelo Entidade Relacionamento), que é o modelo utilizado para mostrar os códigos em sql.

Segue um exemplo de como criar uma tabela em SQL, baseada nos exemplos anteriores:



```
CREATE TABLE clientes (  
  id INT PRIMARY KEY,  
  nome VARCHAR(100),  
  email VARCHAR(100),  
);
```

03

# **Relacionamento Entre Tabelas**

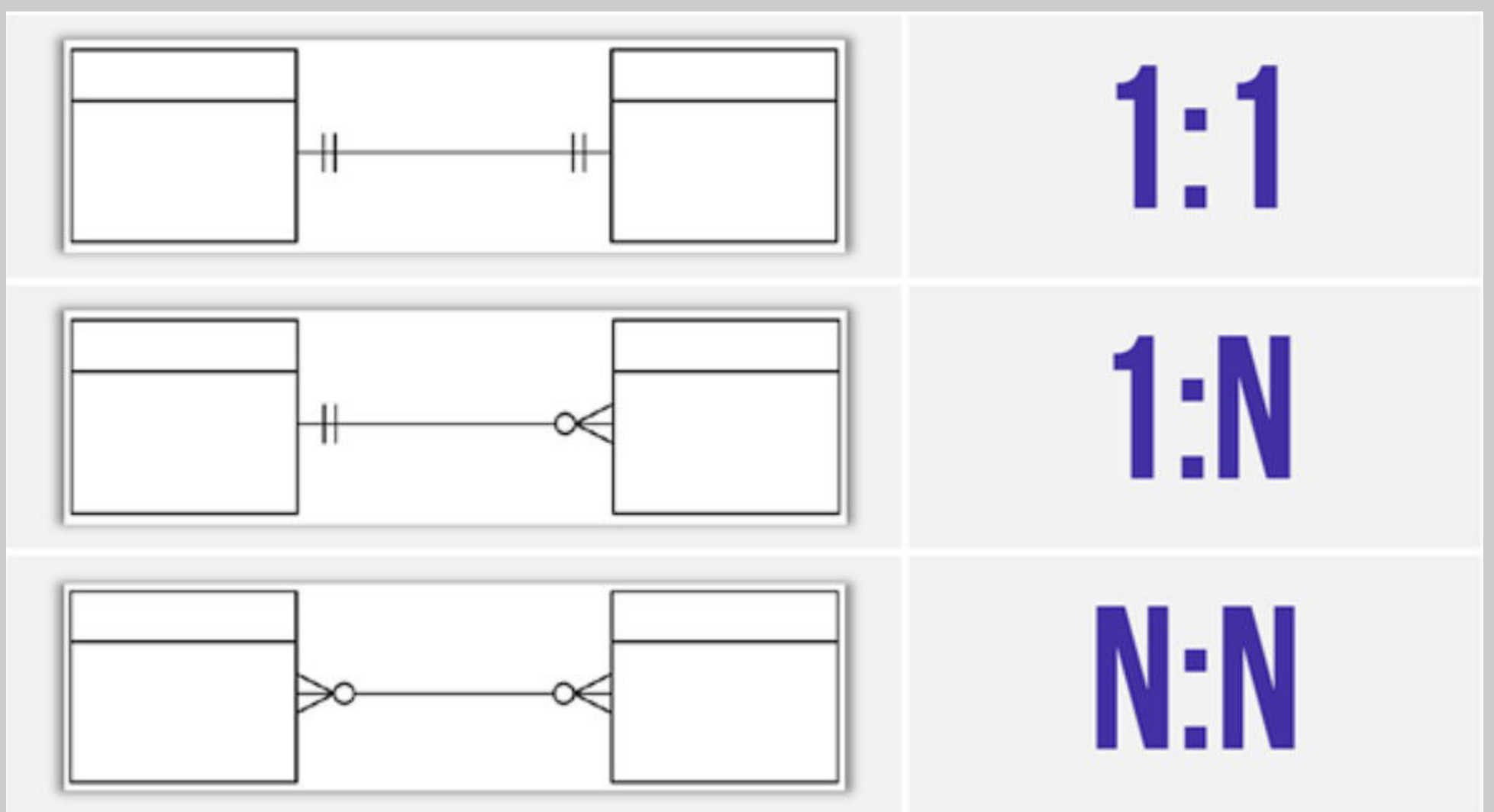


# O que são conexões entre tabelas?

Em bancos de dados relacionais, os dados geralmente estão divididos em várias tabelas, cada uma com um tipo específico de informação (ex: clientes, pedidos, produtos).

Para que essas tabelas "conversem entre si", usamos os relacionamentos. Eles servem para ligar dados que estão separados, garantindo uma estrutura organizada, sem repetições desnecessárias

Existem 3 tipos de relacionamento entre tabelas, todos sendo usados de maneiras diferentes. São eles os relacionamentos 1-1, 1-n e n-n.



# Relacionamento 1 para 1 (1:1)

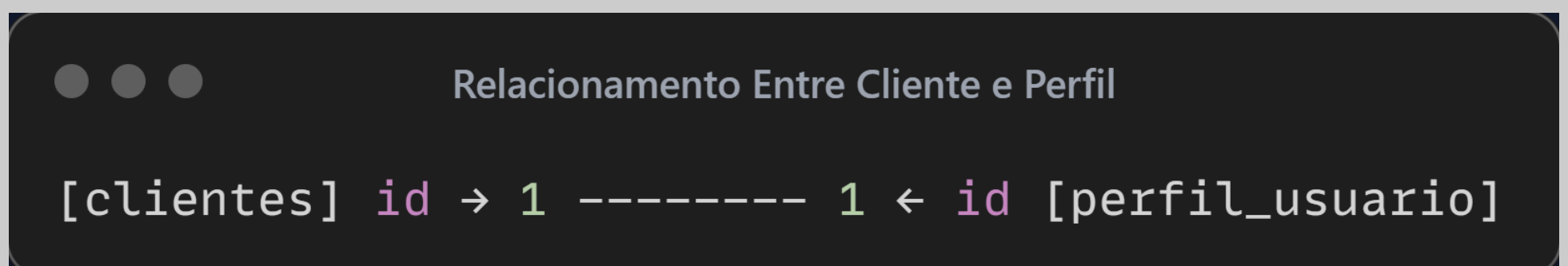


Um relacionamento 1:1 acontece quando um registro em uma tabela se relaciona com exatamente um registro em outra tabela, e vice versa.

Geralmente, esse tipo de relação é usado quando queremos separar informações complementares, mas que ainda mantêm uma ligação direta e exclusiva.

Um exemplo disso seria o perfil de um cliente em um site e um cliente físico. Um cliente físico apenas pode ter um perfil no site, e vice versa

Abaixo vem um exemplo do que eu falei. No caso, o que ele quer dizer aqui é que um cliente só pode receber um id do perfil, e vice versa.



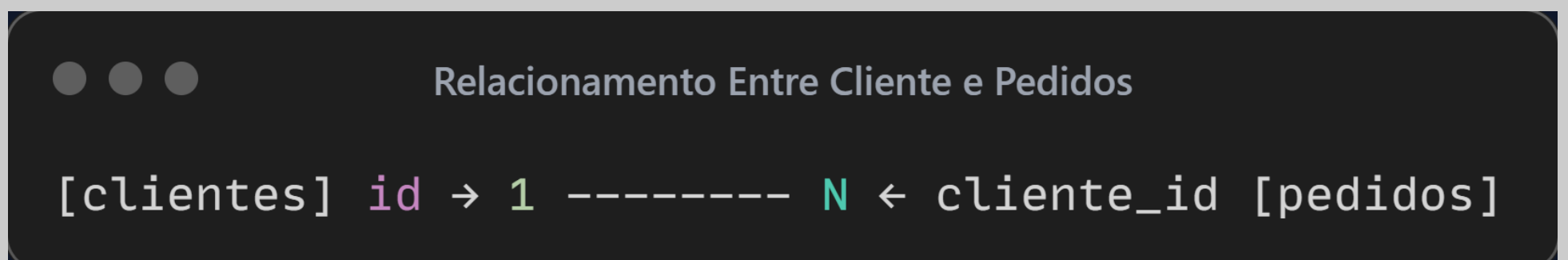
# Relacionamento 1 para muitos (1:n)

---

Esse é o tipo mais comum, por isso, se estiver com problemas aprendendo estes relacionamentos, recomendo que aprenda com esse, vendo exemplos de código e de MER's, pois ele também é o mais fácil de lembrar.

Significa que um registro em uma tabela pode estar ligado a vários registros em outra tabela.

Exemplo: um cliente pode realizar vários pedidos, mas um pedido só pode pertencer a vários clientes.



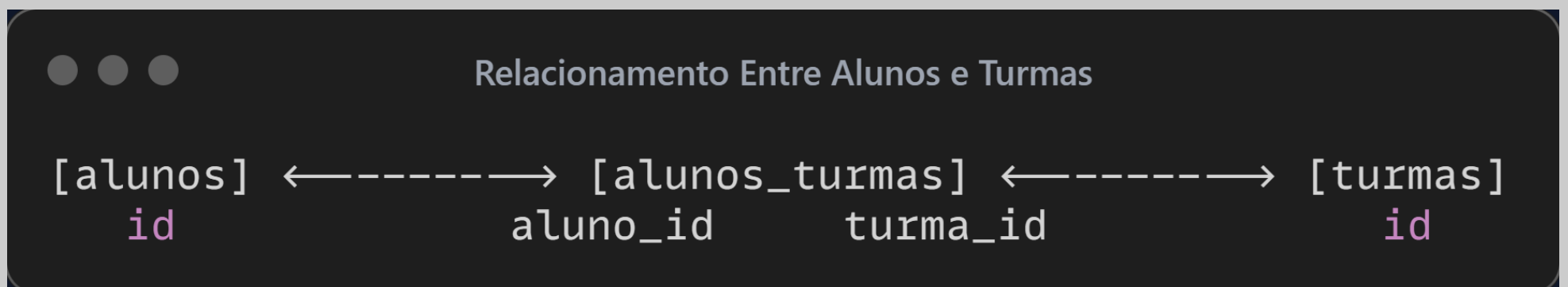
# Relacionamento Muitos para Muitos (N:N)

Ocorre quando vários registros de uma tabela podem se relacionar com vários registros de outra.

Para representar isso, criamos uma tabela intermediária que faz a ponte entre as duas:

Exemplo: (para esse exemplo, vamos nos desviar dos exemplos de clientes, pois será mais fácil de compreender este exemplo)

Um aluno pode estar em várias turmas, e cada turma tem vários alunos.



# Chave Estrangeira (Foreign Key)

---

Como ultima matéria deste ebook, gostaria de explicar para você sobre chaves estrangeiras.

Uma chave estrangeira é um campo que liga uma tabela a outra.

Ela serve para indicar que um dado em uma tabela está relacionado a outro em uma tabela diferente.

No exemplo anterior, de relacionamento 1-n, usamos uma chave estrangeira na tabela pedidos apontando para o id da tabela clientes. Assim, sabemos que aquele pedido é daquele cliente.

No exemplo de relacionamento n-n, nós temos duas chaves estrangeiras na tabelas “alunos\_turmas”, pois ela recebe informação das duas tabelas, servindo como uma intermediaria.

Se segue um exemplo de relacionamento 1-n com chave estrangeira

## Tabela Clientes com Chave Estrangeira

```
CREATE TABLE clientes (  
  id INT PRIMARY KEY,  
  nome VARCHAR(100),  
  email VARCHAR(100),  
  FOREIGN KEY (cliente_id) REFERENCES clientes(id)  
);
```

# Agradecimentos

---

Este ebook foi gerado com auxilio de IA, e diagramado e corrigido por um ser humano. O passo a passo se encontra no meu github.

Este conteúdo foi gerado com fins didáticos, caso você seja vendido este ebook, você foi roubado. Busque meios de recuperar este dinheiro imediatamente.

O ebook não se aprofunda tanto assim em sql, sendo que eu poderia explicar sobre JOINS, mas está é a intenção: Ter um conteúdo mais raso para que as pessoas entrando agora neste meio tenham um bom guia de por onde começar.



[Meu Github](#)