Aula: INNER JOIN em MySQL

1. Introdução

- **Objetivo**: Entender o conceito de **INNER JOIN**, quando e por que usá-lo, e como aplicá-lo em um banco de dados MySQL.
- **Definição**: Um **INNER JOIN** é uma operação que retorna apenas as linhas que possuem correspondências em ambas as tabelas. Ele compara os valores em uma ou mais colunas das tabelas e retorna os registros onde há correspondência.

2. Explicação Teórica: O Que é INNER JOIN?

- O **INNER JOIN** combina as linhas de duas tabelas com base em uma condição de igualdade entre colunas relacionadas. A condição é geralmente baseada em uma chave estrangeira ou algum outro critério lógico.
- **Como funciona?**
    - Imagine que temos duas tabelas. Quando aplicamos um **INNER JOIN**, o MySQL compara uma coluna em ambas as tabelas e retorna apenas as linhas que têm valores correspondentes.
    - **Condições de igualdade**: O **INNER JOIN** mais comum usa o operador de igualdade (`=`), que verifica se os valores em uma coluna da primeira tabela são iguais aos da segunda tabela.

3. Exemplo de INNER JOIN com Tabelas de Exemplo

- Suponha que temos duas tabelas relacionadas: `Clientes` e `Pedidos`. A tabela `Clientes` contém informações sobre os clientes e a tabela `Pedidos` contém informações sobre os pedidos feitos por esses clientes.

Tabela 1: Clientes

id_cliente	nome	email
1	Maria Silva	maria@email.com
2	João Souza	joao@email.com
3	Ana Costa	ana@email.com
4	Pedro Lima	pedro@email.com

```
CREATE TABLE Clientes (
    id_cliente INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100),
    email VARCHAR(100)
);
```

```
INSERT INTO Clientes (nome, email)
VALUES
('Maria Silva', 'maria@email.com'),
('João Souza', 'joao@email.com'),
('Ana Costa', 'ana@email.com'),
('Pedro Lima', 'pedro@email.com');
```

Tabela 2: Pedidos

id_pedido	id_cliente	data_pedido	valor_total
101	1	2023-01-15	500.00
102	2	2023-02-20	300.00
103	1	2023-03-05	200.00
104	4	2023-03-15	400.00

```
CREATE TABLE Pedidos (
    id_pedido INT AUTO_INCREMENT PRIMARY KEY,
    id_cliente INT,
    data_pedido DATE,
    valor_total DECIMAL(10, 2),
    FOREIGN KEY (id_cliente) REFERENCES Clientes(id_cliente)
);
```

```
INSERT INTO Pedidos (id_cliente, data_pedido, valor_total)

VALUES

(1, '2023-01-15', 500.00),

(2, '2023-02-20', 300.00),

(1, '2023-03-05', 200.00),

(4, '2023-03-15', 400.00);
```

4. Aplicando o INNER JOIN

Agora que temos as tabelas, vamos criar uma consulta usando **INNER JOIN** para combinar dados de `Clientes` e `Pedidos`. Queremos ver quais clientes fizeram pedidos e os detalhes dos pedidos.

Consulta INNER JOIN:

sql

Copiar código

```
SELECT Clientes.nome, Pedidos.data_pedido, Pedidos.valor_total
FROM Clientes
INNER JOIN Pedidos ON Clientes.id_cliente = Pedidos.id_cliente;
```

Explicação da Consulta:

- A consulta acima seleciona os campos `nome` da tabela `Clientes`, `data_pedido` e `valor_total` da tabela `Pedidos`.
- O **INNER JOIN** é aplicado na coluna `id_cliente` de ambas as tabelas. Ou seja, estamos procurando por linhas onde o `id_cliente` na tabela `Clientes` corresponde ao `id_cliente` na tabela `Pedidos`.

Resultado da Consulta:

nome	data_pedido	valor_total
Maria Silva	2023-01-15	500.00
Maria Silva	2023-03-05	200.00

João Souza	2023-02-20	300.00
Pedro Lima	2023-03-15	400.00

- O resultado mostra **somente** os clientes que têm pedidos correspondentes na tabela `Pedidos`. Observe que o cliente **Ana Costa** (id_cliente 3) não aparece no resultado porque ela não fez nenhum pedido.

5. Discussão Teórica do INNER JOIN

- **Por que usar INNER JOIN?**
  - O **INNER JOIN** é útil quando você quer trabalhar apenas com dados que têm correspondências em ambas as tabelas. Em termos práticos, isso é muito comum em sistemas de banco de dados relacionais, como quando você precisa de todas as transações de clientes que realmente compraram algo.
- **O que acontece com dados que não têm correspondência?**
  - No nosso exemplo, os clientes que não têm pedidos (como **Ana Costa**) foram excluídos do resultado. O **INNER JOIN** remove automaticamente qualquer linha da primeira ou segunda tabela que não tenha uma correspondência.

6. INNER JOIN com Múltiplas Tabelas

- Você também pode aplicar **INNER JOIN** com mais de duas tabelas. Suponha que temos uma terceira tabela chamada `Produtos`, que contém os produtos comprados em cada pedido.

Tabela 3: Produtos

id_produto	id_pedido	nome_produto	preco
201	101	Notebook	500.00
202	102	Teclado	100.00
203	102	Mouse	50.00
204	103	Impressora	200.00
205	104	Monitor	400.00

```
CREATE TABLE Produtos (

    id_produto INT AUTO_INCREMENT PRIMARY KEY,
```

```
    id_pedido INT,
    nome_produto VARCHAR(100),
    preco DECIMAL(10, 2),
    FOREIGN KEY (id_pedido) REFERENCES Pedidos(id_pedido)
);

INSERT INTO Produtos (id_pedido, nome_produto, preco) VALUES (101,
'Notebook', 500.00), (102, 'Teclado', 100.00), (102, 'Mouse',
50.00), (103, 'Impressora', 200.00), (104, 'Monitor', 400.00);
```

- Agora queremos ver quais produtos foram comprados por cada cliente, utilizando **INNER JOIN** em três tabelas: `Clientes`, `Pedidos` e `Produtos`.

Consulta INNER JOIN com Múltiplas Tabelas:

sql

Copiar código

```
SELECT Clientes.nome, Pedidos.data_pedido, Produtos.nome_produto,
Produtos.preco
FROM Clientes
INNER JOIN Pedidos ON Clientes.id_cliente = Pedidos.id_cliente
INNER JOIN Produtos ON Pedidos.id_pedido = Produtos.id_pedido;
```

Resultado da Consulta:

nome	data_pedido	nome_produto	preco
Maria Silva	2023-01-15	Notebook	500.00
João Souza	2023-02-20	Teclado	100.00
João Souza	2023-02-20	Mouse	50.00
Maria Silva	2023-03-05	Impressora	200.00
Pedro Lima	2023-03-15	Monitor	400.00

- Essa consulta combina as três tabelas e mostra quais produtos cada cliente comprou e em qual pedido.

7. Conclusão

- O **INNER JOIN** é uma operação poderosa em bancos de dados relacionais que permite combinar dados de várias tabelas de forma eficiente, retornando apenas as correspondências.
- Use **INNER JOIN** quando quiser trabalhar apenas com registros que têm correspondências em ambas as tabelas.
- **Vantagens**: Simplicidade e eficiência em relacionar tabelas.
- **Limitação**: Ele não retorna dados de uma tabela se não houver correspondência na outra.

8. Exercícios

1. Usando as tabelas `Clientes` e `Pedidos`, crie uma consulta que mostre os nomes dos clientes e o número total de pedidos que cada cliente fez.
    - **Dica**: Use `COUNT` e `GROUP BY`.
2. Adicione a tabela `Produtos` à consulta do exercício anterior e mostre o número total de produtos comprados por cada cliente.

4o