Aula: LEFT JOIN e RIGHT JOIN em MySQL

1. Introdução

- **Objetivo**: Entender o funcionamento dos operadores **LEFT JOIN** e **RIGHT JOIN** em consultas SQL, suas diferenças em relação ao **INNER JOIN**, e quando usá-los.
- **Definição**:
    - O **LEFT JOIN** retorna todas as linhas da tabela à esquerda, junto com as correspondentes da tabela à direita. Se não houver correspondência, ele ainda retorna a linha da tabela da esquerda, com valores `NULL` para as colunas da tabela direita.
    - O **RIGHT JOIN** é o oposto: retorna todas as linhas da tabela à direita, junto com as correspondentes da tabela à esquerda. Se não houver correspondência, a tabela da direita é retornada com valores `NULL` para as colunas da esquerda.

---

2. Explicação Teórica: LEFT JOIN

- **LEFT JOIN** (também chamado de **LEFT OUTER JOIN**) retorna **todas** as linhas da tabela à esquerda da junção, independentemente de haver uma correspondência com a tabela à direita. Quando não há correspondência, o resultado mostra `NULL` nas colunas da tabela à direita.
- **Exemplo Conceitual**: Imagine que você tem uma lista de clientes, e nem todos os clientes fizeram pedidos. Se você usar **LEFT JOIN**, obterá todos os clientes, mesmo aqueles que não têm pedidos registrados.

3. Exemplo de LEFT JOIN

Vamos utilizar as mesmas tabelas que foram apresentadas na aula anterior:

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

Comando de Criação da Tabela `Clientes` com `AUTO_INCREMENT`

sql

Copiar código

```
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

sql

Copiar código

```
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) );
```

sql
Copiar código

```
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);
```

Consulta LEFT JOIN:

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

Explicação da Consulta:

- Esta consulta usa **LEFT JOIN** para retornar todos os clientes, independentemente de terem feito um pedido.
- As colunas de `Pedidos` serão preenchidas com `NULL` quando não houver correspondência 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
Ana Costa	NULL	NULL
Pedro Lima	2023-03-15	400.00

- Note que **Ana Costa**, que não fez nenhum pedido, ainda aparece no resultado, mas com valores `NULL` para `data_pedido` e `valor_total`.

---

4. Explicação Teórica: RIGHT JOIN

- **RIGHT JOIN** (ou **RIGHT OUTER JOIN**) retorna **todas** as linhas da tabela à direita da junção, independentemente de haver uma correspondência com a tabela à esquerda. Quando não há correspondência, o resultado mostra `NULL` nas colunas da tabela à esquerda.
- **Exemplo Conceitual**: Imagine que você tem uma lista de pedidos e alguns pedidos não têm clientes registrados (o que pode acontecer em casos de inconsistências de dados). Se você usar **RIGHT JOIN**, obterá todos os pedidos, mesmo que não haja correspondência com clientes.

5. Exemplo de RIGHT JOIN

Usaremos as mesmas tabelas:

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

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

Consulta RIGHT JOIN:

sql
Copiar código

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

Explicação da Consulta:

- A consulta usa **RIGHT JOIN** para retornar todos os pedidos, independentemente de haver um cliente correspondente.
- Se não houver correspondência com a tabela `Clientes`, os valores de `Clientes` serão `NULL`.

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

- Nesse caso, o **RIGHT JOIN** e o **INNER JOIN** retornam o mesmo resultado, pois todos os pedidos têm clientes correspondentes. Entretanto, em um cenário onde existem pedidos sem clientes correspondentes, os valores da tabela `Clientes` apareceriam como `NULL`.

---

6. Comparação Entre LEFT JOIN e RIGHT JOIN

- **LEFT JOIN**: Retorna **todas as linhas da tabela da esquerda**, com as correspondentes da direita. Se não houver correspondência, preenche os valores da direita com `NULL`.
- **RIGHT JOIN**: Retorna **todas as linhas da tabela da direita**, com as correspondentes da esquerda. Se não houver correspondência, preenche os valores da esquerda com `NULL`.

- **Semelhança**: Ambos retornam as correspondências e as linhas da tabela "não principal" com `NULL` quando não há dados correspondentes.
- **Diferença principal**: A direção de qual tabela é "obrigatória" no retorno.

---

7. Exemplo Comparativo Prático

Vamos supor que a tabela `Pedidos` tenha um registro de pedido que não tenha um cliente associado.

Tabela `Pedidos` Atualizada:

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
105	NULL	2023-04-01	150.00

Consulta LEFT JOIN:
sql
Copiar código
```
SELECT Clientes.nome, Pedidos.data_pedido, Pedidos.valor_total
FROM Clientes
LEFT JOIN Pedidos ON Clientes.id_cliente = Pedidos.id_cliente;
```

Resultado do LEFT JOIN:

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
Ana Costa	NULL	NULL
Pedro Lima	2023-03-15	400.00

Consulta RIGHT JOIN:

sql
Copiar código

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

Resultado do RIGHT JOIN:

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
NULL	2023-04-01	150.00

- O **LEFT JOIN** não retorna o pedido sem cliente, enquanto o **RIGHT JOIN** inclui o pedido sem cliente (com `NULL` para o nome do cliente).

---

8. Conclusão

- **LEFT JOIN** e **RIGHT JOIN** são úteis quando queremos garantir que todas as linhas de uma tabela (esquerda ou direita) estejam presentes no resultado, independentemente de haver correspondência.
- Use **LEFT JOIN** quando a tabela da esquerda for mais importante (ou quando você quer garantir que todos os dados da esquerda apareçam).
- Use **RIGHT JOIN** quando a tabela da direita for a mais importante.