UniSENAI

Módulo 5: Consultas Avançadas (JOINs e Subconsultas)

Módulo 5: Consultas Avançadas (JOINs e Subconsultas)

Consultas avançadas são essenciais para trabalhar com dados interconectados em um banco de dados relacional. Este módulo explora como combinar e manipular dados de múltiplas tabelas utilizando JOINs, subconsultas e funções de agregação para realizar análises complexas.

---

1. Consultas com JOIN

O que são JOINs e sua Importância

Um JOIN é usado para combinar registros de duas ou mais tabelas com base em uma condição. Ele permite que você trabalhe com dados relacionados de forma integrada, evitando redundância e promovendo a eficiência.

- **Importância dos JOINs:**
  - Facilitam a recuperação de dados relacionados.
  - Reduzem a duplicação de informações, promovendo um design eficiente do banco de dados.
  - São a base para relacionamentos em bancos de dados relacionais.

Tipos de JOINs

1. **INNER JOIN:** Retorna apenas os registros que têm correspondência em ambas as tabelas.
2. **LEFT JOIN:** Retorna todos os registros da tabela à esquerda e as correspondências da tabela à direita.
3. **RIGHT JOIN:** Retorna todos os registros da tabela à direita e as correspondências da tabela à esquerda.
4. **FULL OUTER JOIN:** Retorna todos os registros de ambas as tabelas, com `NULL` para dados sem correspondência (emulado no MySQL).

INNER JOIN

O INNER JOIN combina tabelas e retorna apenas os registros que possuem correspondência em ambas.

Exemplo Prático

- **Tabela `clientes`:**

id_cliente	Nome
1	João Silva
2	Maria Souza

- **Tabela `pedidos`:**
-

id_pedido	id_cliente	valor_total
101	1	150.00
102	2	200.00

- **Consulta**: Recuperar pedidos com os nomes dos clientes.

```sql
SELECT clientes.nome, pedidos.valor_total
FROM clientes
INNER JOIN pedidos ON clientes.id_cliente = pedidos.id_cliente;
```

Resultado:

nome	valor_total
João Silva	150.00
Maria Souza	200.00

LEFT JOIN

O **LEFT JOIN** retorna todos os registros da tabela à esquerda e os registros correspondentes da tabela à direita. Quando não há correspondência, exibe `NULL`.

Exemplo Prático

- **Tabela `clientes`**:

id_cliente	nome
1	João Silva
2	Maria Souza
3	Ana Oliveira

Consulta: Recuperar todos os clientes, mesmo aqueles sem pedidos.

```
SELECT clientes.nome, pedidos.valor_total
FROM clientes
LEFT JOIN pedidos ON clientes.id_cliente = pedidos.id_cliente;
```

Resultado:

nome	valor_total
João Silva	150.00
Maria Souza	200.00
Ana Oliveira	NULL

RIGHT JOIN

O **RIGHT JOIN** retorna todos os registros da tabela à direita e os registros correspondentes da tabela à esquerda. Quando não há correspondência, exibe `NULL`.

Exemplo Prático

Inversão da lógica do **LEFT JOIN**:

```
SELECT clientes.nome, pedidos.valor_total
FROM clientes
RIGHT JOIN pedidos ON clientes.id_cliente = pedidos.id_cliente;
```

FULL OUTER JOIN (emulação no MySQL)

Como o MySQL não possui suporte nativo para FULL OUTER JOIN, ele pode ser emulado com um UNION de `LEFT JOIN` e `RIGHT JOIN`.

Exemplo Prático

```sql
SELECT clientes.nome, pedidos.valor_total
FROM clientes
LEFT JOIN pedidos ON clientes.id_cliente = pedidos.id_cliente
UNION
SELECT clientes.nome, pedidos.valor_total
FROM clientes
RIGHT JOIN pedidos ON clientes.id_cliente = pedidos.id_cliente;
```

2. Subconsultas (Subqueries)

As subconsultas são consultas aninhadas dentro de outra consulta principal. Podem ser usadas para recuperar informações que serão utilizadas pela consulta principal.

Subconsultas Simples

Em SELECT

Consulta: Recuperar o nome do cliente com o maior pedido.

```sql
SELECT nome
FROM clientes
WHERE id_cliente = (
    SELECT id_cliente
    FROM pedidos
    ORDER BY valor_total DESC
    LIMIT 1
);
```

Em INSERT

Adicionar um novo pedido para o cliente com o maior pedido atual:

```
INSERT INTO pedidos (id_cliente, valor_total)
VALUES (
    (SELECT id_cliente FROM pedidos ORDER BY valor_total DESC LIMIT 1),
    500.00
);
```

Subconsultas Correlacionadas

Uma subconsulta correlacionada depende de cada registro da consulta principal para ser executada.

Exemplo Prático

Listar clientes e o valor total de seus pedidos.

```
SELECT nome, (
    SELECT SUM(valor_total)
    FROM pedidos
    WHERE pedidos.id_cliente = clientes.id_cliente
) AS total_pedidos
FROM clientes;
```

3. Funções de Agregação

As funções de agregação processam um conjunto de valores para retornar um único valor estatístico. São amplamente utilizadas em análise de dados.

Funções Comuns

1. **COUNT:** Conta o número de registros.

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

2. SUM: Soma os valores de uma coluna.

```
SELECT SUM(valor_total) AS total_vendas FROM pedidos;
```

3. AVG: Calcula a média dos valores

```
SELECT AVG(valor_total) AS media_pedidos FROM pedidos;
```

4. MAX: Retorna o maior valor.

```
SELECT MAX(valor_total) AS maior_pedido FROM pedidos;
```

5. MIN: Retorna o menor valor

```
SELECT MIN(valor_total) AS menor_pedido FROM pedidos;
```

Uso com `GROUP BY` e `HAVING`

O `GROUP BY` organiza os registros em grupos com base em uma ou mais colunas. O `HAVING` filtra os grupos com base em condições agregadas.

Exemplo Prático

Consulta: Total de vendas por cliente, exibindo apenas clientes com mais de 200 em vendas.

```
SELECT clientes.nome, SUM(pedidos.valor_total) AS total_vendas
FROM clientes
JOIN pedidos ON clientes.id_cliente = pedidos.id_cliente
GROUP BY clientes.nome
HAVING total_vendas > 200;
```

Resultado:

nome	total_vendas
Maria Souza	200.00

Vamos Praticar

Questões Práticas (10 questões)

1. INNER JOIN (1 ponto)
Crie uma consulta que liste o nome dos clientes e os valores dos pedidos realizados, exibindo apenas aqueles que possuem pedidos registrados.

2. LEFT JOIN (1 ponto)
Escreva uma consulta que liste todos os clientes e os valores de seus pedidos. Caso o cliente não tenha realizado pedidos, exiba `NULL` na coluna do valor total.

3. RIGHT JOIN (1 ponto)
Crie uma consulta para listar todos os pedidos e os nomes dos clientes que os realizaram. Se o pedido não estiver associado a um cliente, exiba `NULL` na coluna do nome do cliente.

4. FULL OUTER JOIN (1 ponto)
Emule um FULL OUTER JOIN utilizando `UNION`, para exibir todos os clientes e pedidos, mesmo que não possuam correspondência.

5. Subconsulta Simples em SELECT (1 ponto)
Escreva uma consulta que recupere o nome do cliente que realizou o pedido de maior valor.

6. Subconsulta Simples em INSERT (1 ponto)
Adicione um novo pedido para o cliente que possui o pedido de maior valor. Utilize uma subconsulta para encontrar o cliente.

7. Subconsulta Correlacionada (1 ponto)
Liste os nomes dos clientes e o total de valores de seus pedidos utilizando uma subconsulta correlacionada.

8. Função de Agregação COUNT (1 ponto)
Crie uma consulta que conte o número total de pedidos realizados.

9. Função de Agregação com GROUP BY (1 ponto)
Liste o total de vendas por cliente (nome e total vendido).

10. Função de Agregação com GROUP BY e HAVING (1 ponto)
Liste os clientes que realizaram vendas superiores a 200, exibindo o nome do cliente e o total vendido.

Questões Teóricas (10 questões)

1. (1 ponto) Explique o que é um **INNER JOIN** e cite um exemplo prático onde ele seria utilizado.

2. (1 ponto) Qual a diferença entre **LEFT JOIN** e **RIGHT JOIN**? Cite um exemplo prático para cada um.

3. (1 ponto) Explique como o **FULL OUTER JOIN** pode ser emulado no MySQL. Por que ele não é nativo?

4. (1 ponto) O que são **subconsultas simples**? Dê um exemplo onde uma subconsulta seria mais eficiente do que um `JOIN`.

5. (1 ponto) Qual a diferença entre uma **subconsulta simples** e uma **subconsulta correlacionada**?

6. (1 ponto) Explique o conceito de **funções de agregação** e liste as mais utilizadas no MySQL.

7. (1 ponto) Qual a função do **GROUP BY** em uma consulta SQL? Em quais casos ele deve ser utilizado?

8. (1 ponto) O que é a cláusula **HAVING**? Como ela difere da cláusula **WHERE**?

9. (1 ponto) Cite dois cenários práticos onde você utilizaria a combinação de **GROUP BY** e funções de agregação.

10. (1 ponto) Explique a importância dos **JOINs** em bancos de dados relacionais. Como eles contribuem para evitar redundâncias e promover eficiência?

Instruções para a Entrega das Atividades

1. **Elaboração e Envio do Arquivo**
   - Responda todas as questões de forma clara e objetiva.
   - Gere um arquivo no formato **.PDF** contendo as respostas de cada questão.
   - Envie o arquivo para os e-mails dos professores responsáveis.
2. **Validação da Atividade**
   - Após o envio do arquivo, procure o(s) professor(es) para realizar a validação da atividade.
   - **Não inicie a próxima atividade sem antes validar a anterior com o professor.**
3. **Forma de Validação**
   - **Explicação Verbal**: Explique cada resposta verbalmente ao(s) professor(es).
   - **Perguntas e Respostas**: Esteja preparado para responder aos questionamentos do(s) professor(es) sobre o conteúdo das respostas.
   - **Orientação**: Receba orientações sobre a apresentação do(s) tema(s).