MSc Sandir Rodrigues Campos

Banco de Dados 2

Aula 7

3”

VIEWS

O que são views?

- Uma view é uma tabela virtual que é baseada em uma consulta SQL. Ela não armazena dados propriamente ditos, mas sim uma consulta que é executada a cada vez que a view é acessada.

Por que usar views?

- Existem várias razões pelas quais você pode querer usar views:
    1. Simplificar consultas complexas: Views podem ser usadas para simplificar consultas complexas, tornando-as mais fáceis de entender e manter.
    2. Proteger dados: Views podem ser usadas para proteger dados, limitando o acesso a certas colunas ou linhas de uma tabela.
    3. Melhorar a performance: Views podem ser usadas para melhorar a performance de consultas, pois elas podem ser otimizadas para usar índices e outras técnicas de otimização.

VIEWS

Como criar uma view?

A sintaxe para criar uma view é a seguinte:

```
CREATE VIEW nome_da_view
AS
SELECT coluna1, coluna2, ...
FROM tabela
[WHERE condicao]
[GROUP BY coluna1, coluna2, ...]
[HAVING condicao];
```

Exemplo simples de view

Suponha que você tenha uma tabela chamada clientes com as seguintes colunas:

- CREATE TABLE clientes (id INT PRIMARY KEY, nome VARCHAR(255), email VARCHAR(255));

Você pode criar uma view que retorne apenas os nomes e emails dos clientes:

- CREATE VIEW clientes_simplificados AS SELECT nome, emailFROM clientes;

Agora você pode consultar a view como se fosse uma tabela:

- SELECT * FROM clientes_simplificados;

Exemplo mais complexo de view

Suponha que você tenha duas tabelas chamadas pedidos e produtos com as seguintes colunas:

- CREATE TABLE pedidos (id INT PRIMARY KEY, cliente_id INT, data_pedido DATE, valor DECIMAL(10, 2));
- CREATE TABLE produtos (id INT PRIMARY KEY, nome VARCHAR(255), preco DECIMAL(10, 2));

Você pode criar uma view que retorne os pedidos com os nomes dos produtos:

```
CREATE VIEW pedidos_com_produtos AS SELECT p.id, p.cliente_id, p.data_pedido, p.valor, pr.nome AS produto FROM pedidos p
JOIN produtos pr ON p.id = pr.id;
```

Agora você pode consultar a view como se fosse uma tabela:

```
SELECT * FROM pedidos_com_produtos;
```

Alterando ou excluindo uma view

Alterar uma view

Sintaxe para alterar uma view é a seguinte:

```
CREATE OR REPLACE VIEW nome_da_view AS SELECT coluna1, coluna2,
...FROM tabela[WHERE condicao][GROUP BY coluna1, coluna2,
...][HAVING condicao];
```

Excluir uma view

A sintaxe para excluir uma view é a seguinte:

```
DROP VIEW nome_da_view;
```

Fazer

- Você pode imaginar que já existem essas tabelas:
  - clientes(id_cliente, nome, email, cidade, gasto_total)
  - pedidos(id_pedido, id_cliente, data_pedido, valor_total)
  - produtos(id_produto, nome_produto, preco)
  - itens_pedido(id_pedido, id_produto, quantidade)

Crie uma view chamada clientes_ricos que exibe apenas o nome, cidade e gasto_total dos clientes com gasto_total acima de R$ 20.000.
(Mímimo de3 clientes que atendam e 3 que não atendam)

Agora vamos ALTERAR essa view para:

- Mostrar também a **coluna email**
- E mudar o filtro para **gasto_total > 30000**

FJ

- Crie uma view chamada clientes_top_vendas que mostre:
  - id_cliente
  - Nome
  - total_gasto (soma dos valor_total dos pedidos)
  - Mas só inclua os clientes que gastaram mais de R$ 50.000 no total.

FJ

- Crie uma view chamada detalhes_vendas que mostre:
  - id_pedido
  - nome_produto
  - Quantidade
  - preco_unitário
  - valor_item (preço x quantidade)

FAZER

- Crie uma view chamada clientes_com_pedidos_medios que mostre:
  - id_cliente
  - Nome
  - media_valor_pedido (média do valor dos pedidos de cada cliente)
  - Mas só mostre clientes com mais de 2 pedidos registrados.

Exercícios com views

Exercício 1: Criar uma view que retorne os pedidos com os nomes dos produtos e os nomes dos clientes

Crie as seguintes tabelas:

- CREATE TABLE clientes (id INT PRIMARY KEY, nome VARCHAR(255), email VARCHAR(255));
- CREATE TABLE pedidos (id INT PRIMARY KEY, cliente_id INT, data_pedido DATE, valor DECIMAL(10, 2), FOREIGN KEY (cliente_id) REFERENCES clientes(id));
- CREATE TABLE produtos (id INT PRIMARY KEY, nome VARCHAR(255), preco DECIMAL(10, 2));
- CREATE TABLE itens_pedidos (id INT PRIMARY KEY, pedido_id INT, produto_id INT, quantidade INT, FOREIGN KEY (pedido_id) REFERENCES pedidos(id), FOREIGN KEY (produto_id) REFERENCES produtos(id));

Insira alguns dados:

- INSERT INTO clientes (id, nome, email) VALUES (1, 'João', 'joao@example.com'), (2, 'Maria', 'maria@example.com'), (3, 'Pedro', 'pedro@example.com');
- INSERT INTO pedidos (id, cliente_id, data_pedido, valor) VALUES (1, 1, '2022-01-01', 100.00), (2, 1, '2022-01-15', 200.00), (3, 2, '2022-02-01', 50.00);
- INSERT INTO produtos (id, nome, preco) VALUES (1, 'Produto 1', 10.00), (2, 'Produto 2', 20.00), (3, 'Produto 3', 30.00);
- INSERT INTO itens_pedidos (id, pedido_id, produto_id, quantidade) VALUES (1, 1, 1, 2), (2, 1, 2, 1), (3, 2, 3, 3), (4, 3, 1, 1);

Crie uma view que retorne os pedidos com os nomes dos produtos e os nomes dos clientes:

Exercícios com views

Exercício 2: Criar uma view que retorne os produtos com os nomes dos fornecedores e os preços médios

Crie as seguintes tabelas:

- CREATE TABLE fornecedores (id INT PRIMARY KEY, nome VARCHAR(255), email VARCHAR(255));
- CREATE TABLE produtos (id INT PRIMARY KEY, nome VARCHAR(255), preco DECIMAL(10, 2), fornecedor_id INT, FOREIGN KEY (fornecedor_id) REFERENCES fornecedores(id));

Insira alguns dados:

- INSERT INTO fornecedores (id, nome, email) VALUES (1, 'Fornecedor 1', 'fornecedor1@example.com'), (2, 'Fornecedor 2', 'fornecedor2@example.com'), (3, 'Fornecedor 3', 'fornecedor3@example.com’);
- INSERT INTO produtos (id, nome, preco, fornecedor_id) VALUES (1, 'Produto 1', 10.00, 1), (2, 'Produto 2', 20.00, 1), (3, 'Produto 3', 30.00, 2), (4, 'Produto 4', 40.00, 3), (5, 'Produto 5', 50.00, 3);

Crie uma view que retorne os produtos com os nomes dos fornecedores e os preços médios:

Exercícios com views

Exercício 3: Criar uma view que retorne os pedidos com os nomes dos clientes e os valores totais, agrupados por mês

Crie as seguintes tabelas:

- CREATE TABLE clientes (id INT PRIMARY KEY, nome VARCHAR(255), email VARCHAR(255));
- CREATE TABLE pedidos (id INT PRIMARY KEY, cliente_id INT, data_pedido DATE, valor DECIMAL(10, 2), FOREIGN KEY (cliente_id) REFERENCES clientes(id));

Insira alguns dados:

- INSERT INTO clientes (id, nome, email) VALUES (1, 'João', 'joao@example.com'), (2, 'Maria', 'maria@example.com'), (3, 'Pedro', 'pedro@example.com’);
- INSERT INTO pedidos (id, cliente_id, data_pedido, valor) VALUES (1, 1, '2022-01-01', 100.00), (2, 1, '2022-01-15', 200.00), (3, 2, '2022-02-01', 50.00), (4, 3, '2022-03-01', 150.00), (5, 1, '2022-03-15', 250.00);

Crie uma view que retorne os pedidos com os nomes dos clientes e os valores totais, agrupados por mês:

Exercícios com views

Exercício 4: Criar uma view que retorne os produtos com os nomes dos fornecedores e os preços médios, considerando apenas os produtos com preço maior que 50

Crie as seguintes tabelas:

- CREATE TABLE fornecedores (id INT PRIMARY KEY, nome VARCHAR(255), email VARCHAR(255));
- CREATE TABLE produtos (id INT PRIMARY KEY, nome VARCHAR(255), preco DECIMAL(10, 2), fornecedor_id INT, FOREIGN KEY (fornecedor_id) REFERENCES fornecedores(id));

Insira alguns dados:

- INSERT INTO fornecedores (id, nome, email) VALUES (1, 'Fornecedor 1', 'fornecedor1@example.com'), (2, 'Fornecedor 2', 'fornecedor2@example.com'), (3, 'Fornecedor 3', 'fornecedor3@example.com’);
- INSERT INTO produtos (id, nome, preco, fornecedor_id) VALUES (1, 'Produto 1', 10.00, 1), (2, 'Produto 2', 20.00, 1), (3, 'Produto 3', 60.00, 2), (4, 'Produto 4', 40.00, 3), (5, 'Produto 5', 70.00, 3);

Crie uma view que retorne os produtos com os nomes dos fornecedores e os preços médios, considerando apenas os produtos com preço maior que 50: