SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Curso

TÉCNICO EM DESENVOLVIMENTO

DE SISTEMAS

SQL Views

Fernanda de Oliveira Nunes

Sorocaba
Novembro – 2024

SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Fernanda de Oliveira Nunes

SQL Views

O que são as SQL Views, por que elas são importantes, e como podem ser utilizadas.

Prof. – Emerson Magalhães

Sorocaba
Novembro – 2024

HISTÓRICO DE VERSÕES

Data	Versão	Descrição	Autor
11/11/2024	1.0	Pesquisa feita e documentada para envio do orientador Emerson Magalhães	Fernanda de O. Nunes

SUMÁRIO

OBJETIVO

O objetivo desta pesquisa é entender o que são as SQL Views, por que elas são importantes, e como podem ser utilizadas para facilitar o acesso e a manipulação de dados em bancos de dados relacionais. Além disso, será explorado o processo de criação de views e exemplos práticos que ilustram suas aplicações no dia a dia.

INTRODUÇÃO

As SQL Views são consultas armazenadas em bancos de dados relacionais, que permitem a visualização de dados de uma ou mais tabelas sem armazená-los fisicamente. Elas são usadas para simplificar o acesso a informações complexas, melhorar a segurança e a performance de consultas, e garantir consistência ao centralizar regras de negócio.

As views são essenciais em sistemas de banco de dados relacionais, pois ajudam na organização e no controle de dados, permitindo que os usuários acessem apenas partes específicas das tabelas e não a estrutura completa. Além disso, são fundamentais para gerar relatórios dinâmicos e consolidar dados de diferentes fontes.

Este trabalho tem como objetivo explorar a definição, a função e as aplicações das SQL Views, destacando suas vantagens e como contribuem para a eficiência e segurança em bancos de dados relacionais.

1. Fundamentos Teóricos

1.1. O que são views e como funcionam no SQL

Uma view é um objeto que é formado por declarações SELECT, que retornam uma visualização de dados específica de uma ou mais tabelas de um banco de dados. Esses objetos também são conhecidos como virtual tables (tabelas virtuais), por não fazerem parte do esquema físico da base. Uma view pode ser utilizada com um conjunto de tabelas que podem ser agregadas a outros conjuntos através do comando UNION. Também pode aumentar a segurança da base por definir políticas de acesso em nível de tabela e coluna.

Ao criar uma view, é possível filtrar o conteúdo de uma tabela a ser exibida, já que a função da view é essa: filtrar tabelas, servindo para agrupá-las, protegendo certas colunas e simplificando o código de programação.

1.2. Diferença entre views e tabelas comuns

Views e tabelas são objetivos dentro de um banco de dados que é possível criar. Uma **tabela** é um objeto formado por linhas e colunas que armazenam dados que podem ser consultados. Já as **views** no SQL são a possibilidade de armazenar o resultado de um query dentro de uma tabela virtual que pode facilitar a utilização delas posteriormente. Dessa forma não vai precisar ficar refazendo suas análises para obter os mesmos resultados novamente.

FIGURA 1

1.3. Tipos de views

1.3.1. Views Simples

São views que encapsulam uma simples consulta (SELECT) a uma ou mais tabelas, sem a necessidade de junções complexas ou agregações. Elas realizam uma consulta direta a uma única tabela. Não envolvem junções ou operações avançadas de agregação. E, em sua maioria, são usadas para simplificar o acesso aos dados de uma única tabela.

1.3.2. Views complexas

São views que podem envolver operações mais complexas, como junções (JOIN) entre múltiplas tabelas ou funções de agregação (como SUM, AVG, COUNT, etc.). Elas realizam operações como junção de tabelas (INNER JOIN, LEFT JOIN, etc.), filtros (WHERE), agrupamentos (GROUP BY), ordenações (ORDER BY) e agregações. São úteis quando precisamos combinar dados de diferentes fontes ou realizar cálculos baseados em agregações.

1.3.3. Views materializadas

São views que o resultado da consulta é armazenado fisicamente no banco de dados, ao contrário das views simples, que apenas definem a consulta a ser executada quando acessada. As views materializadas armazenam os resultados da consulta e não precisam ser recalculadas a cada acesso. Apresente uma melhoria no desempenho de consultas complexas, já que os dados são pré-calculados e armazenados em disco. Mas elas precisam ser atualizadas periodicamente (de forma manual ou automática) para refletir as alterações nos dados. A capacidade de criar views materializadas depende do sistema de gerenciamento de banco de dados (SGBD) que está sendo utilizado (como Oracle, PostgreSQL, etc.).

2. Vantagens

As views permitem criar consultas complexas usando várias tabelas e, em seguida, acessá-las como se fossem uma única tabela. Isso facilita o uso de

consultas prontas e reduz a necessidade de escrever consultas complexas repetidamente.

Elas permitem restringir o acesso aos dados, expondo apenas informações específicas aos usuários autorizados. Isso pode ser útil quando é preciso restringir certas colunas sensíveis ou definir regras de acesso específicas para garantir a segurança dos dados.

3. Desvantagens

O uso de views pode ter um impacto no desempenho das consultas. Como as views são consultas executadas toda vez que são acessadas, consultas complexas ou views mal otimizadas podem levar a uma **deterioração no desempenho**, especialmente quando combinadas com outras consultas.

Quando ocorrem mudanças nas estruturas das tabelas base utilizadas pelas views, as views podem precisar ser atualizadas para refletir essas mudanças. Se houver muitas views dependentes, a manutenção pode se tornar complicada e propensa a erros.

Porém, é importante ressaltar que, apesar das desvantagens mencionadas, as views são uma ferramenta útil em SQL, fornecendo flexibilidade e acesso simplificado aos dados.

4. Processo de Criação de Views

4.1. Instrução Create View

```
CREATE VIEW Produtos AS
SELECT IdProduto AS Código,
    Nome AS Produto,
    Fabricante,
    Quantidade,
    VlUnitario AS [ValorUnitario],
    Tipo
FROM Produtos
```

4.2. Exemplos de views simples

4.2.1. View de filtragem

A view de filtragem é uma consulta simples que aplica filtros (WHERE) para retornar um subconjunto de dados. É útil quando você deseja criar uma visualização com apenas um conjunto específico de registros.

```
CREATE VIEW usuarios_ativos AS
SELECT id, nome, email
FROM usuarios
WHERE status = 'ativo';
```

4.2.2. View de agregação

A view de agregação envolve a aplicação de funções de agregação (como SUM, AVG, COUNT, etc.) em um conjunto de dados. Ela pode agrupar os dados (usando GROUP BY) e resumir informações.

```
CREATE VIEW vendas_resumo AS
SELECT produto_id, COUNT(*) AS total_vendas, SUM(valor) AS total_valor_vendido
FROM vendas
GROUP BY produto_id;
```

4.2.3. View de junção

A view de junção combina dados de duas ou mais tabelas relacionadas, usando operações de junção (JOIN). Essa view pode ser útil para acessar dados distribuídos em várias tabelas de forma consolidada.

```
CREATE VIEW produto_venda AS
SELECT p.id, p.nome, v.quantidade, v.preco
FROM produtos p
JOIN vendas v ON p.id = v.produto_id;
```

4.2.4. View complexa

Uma view complexa pode combinar múltiplas operações em uma única consulta, como junções, filtros, agregações e ordenações. Ela é considerada "complexa" porque envolve mais de uma operação ou tabela e pode ser bastante detalhada.

```
CREATE VIEW vendas_produto_cliente AS
SELECT
    p.nome AS produto_nome,
    c.nome AS cliente_nome,
    SUM(v.quantidade) AS total_quantidade_vendida,
    AVG(v.preco) AS preco_medio
FROM vendas v
JOIN produtos p ON v.produto_id = p.id
JOIN clientes c ON v.cliente_id = c.id
WHERE v.data_venda >= '2024-01-01'
GROUP BY p.nome, c.nome;
```

5. Views Atualizáveis e Não Atualizáveis

5.1. Atualizar dados diretamente em views

O comando ALTER VIEW é utilizado para atualizar uma view, após ela já ter sido criada e necessitar de alterações.

```
ALTER VIEW Produtos AS
SELECT IdProduto AS Código,
        Nome AS Produto,
        Fabricante,
        Quantidade,
        VlUnitario AS [ValorUnitario],
        Tipo
FROM Produtos
```

```
WHERE VlUnitario > 499.00
```

5.2. Condições para que uma view seja atualizável

Uma coluna de uma view é atualizável quando todas as regras a seguir são verdadeiras:

- A visualização é deleitável;
- A coluna resolve para uma coluna de uma tabela e a opção READ ONLY não é especificado.
- Todas as colunas correspondentes dos operandos de um UNION ALL têm exatamente correspondência de tipos de dados e correspondência de valores padrão se a seleção completa da visualização incluir um UNION ALL.

5.3. Exemplo prático de uma view atualizável

Uma view é considerada atualizável quando você pode fazer operações de inserção, atualização ou exclusão diretamente sobre ela, e essas operações se refletem nas tabelas subjacentes (base de dados). Para que uma view seja atualizável, ela geralmente deve ser baseada em uma única tabela e não deve envolver funções agregadas, joins complexos, ou subconsultas que dificultam a rastreabilidade da tabela original.

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

CREATE TABLE pedidos (
    id_pedido INT PRIMARY KEY,
    id_cliente INT,
    valor DECIMAL(10,2),
    data_pedido DATE,
    FOREIGN KEY (id_cliente) REFERENCES clientes(id_cliente)
);
```

A criação de uma view simples que mostra os clientes e os pedidos deles.

```
CREATE VIEW clientes_pedidos AS
SELECT c.id_cliente, c.nome, p.id_pedido, p.valor, p.data_pedido
FROM clientes c
JOIN pedidos p ON c.id_cliente = p.id_cliente;
```

Explicação: essa view pode ser atualizável, pois é baseada em uma única tabela clientes (com uma junção simples de pedidos) e não tem operações que a tornem complexa demais. Ou seja, é possível fizer um UPDATE, INSERT ou DELETE na view, isso será refletido diretamente nas tabelas subjacentes.

5.4. Exemplo prático de uma view não atualizável

Uma view é considerada não atualizável quando ela envolve operações complexas como agregações, junções de múltiplas tabelas, ou quando ela não possui uma correspondência direta com as tabelas subjacentes. Isso ocorre porque o banco de dados não consegue determinar como refletir alterações na view para as tabelas originais.

```
CREATE VIEW resumo_pedidos AS
SELECT c.id_cliente, c.nome, SUM(p.valor) AS total_gasto
FROM clientes c
JOIN pedidos p ON c.id_cliente = p.id_cliente
GROUP BY c.id_cliente, c.nome;
```

Explicação: a view resumo_pedidos mostra o total gasto por cada cliente. No entanto, como ela envolve uma agregação (SUM(p.valor)) e não reflete diretamente os dados individuais nas tabelas, ela não é atualizável.

Tentativa de atualização na view:

```
UPDATE resumo_pedidos
SET total_gasto = 500.00
WHERE id_cliente = 1;
```

Esse comando falharia porque o banco de dados não saberia como atualizar o total_gasto diretamente nas tabelas clientes e pedidos, já que o valor

de total_gasto é um somatório, e não existe um campo diretamente editável para ele.

6. Estudo do caso

6.1. Criação de um bando de dados fictício

```
-- Tabela de clientes
CREATE TABLE clientes (
id_cliente INT PRIMARY KEY,
nome VARCHAR(100),
 email VARCHAR(100),
endereco VARCHAR(200),
telefone VARCHAR(20)
);
-- Tabela de produtos
CREATE TABLE produtos (
id_produto INT PRIMARY KEY,
nome VARCHAR(100),
descricao TEXT,
preco DECIMAL(10, 2),
quantidade_em_estoque INT
);
-- Tabela de pedidos
CREATE TABLE pedidos (
id_pedido INT PRIMARY KEY,
id_cliente INT,
data_pedido DATE,
status VARCHAR(50),
FOREIGN KEY (id_cliente) REFERENCES clientes(id_cliente)
);
 -- Tabela de itens do pedido
CREATE TABLE itens_pedido (
id_item INT PRIMARY KEY,
id_pedido INT,
```

id_produto INT,

quantidade INT,

preco_unitario DECIMAL(10, 2),

FOREIGN KEY (id_pedido) REFERENCES pedidos(id_pedido),

FOREIGN KEY (id_produto) REFERENCES produtos(id_produto));

-- Tabela de pagamentos

CREATE TABLE pagamentos (

id_pagamento INT PRIMARY KEY,

id_pedido INT, valor DECIMAL(10, 2),

data_pagamento DATE,

metodo_pagamento VARCHAR(50),

FOREIGN KEY (id_pedido) REFERENCES pedidos(id_pedido)

);

-- Inserindo clientes

INSERT INTO clientes (id_cliente, nome, email, endereco, telefone) VALUES (1, 'João Silva', 'joao@exemplo.com', 'Rua A, 123', '123456789'),

(2, 'Maria Oliveira', 'maria@exemplo.com', 'Rua B, 456', '987654321');

-- Inserindo produtos

INSERT INTO produtos (id_produto, nome, descricao, preco, quantidade_em_estoque) VALUES

(1, 'Smartphone XYZ', 'Smartphone com 128GB de armazenamento', 1000.00, 50),

(2, 'Camiseta Estampada', 'Camiseta de algodão, tamanho M', 50.00, 200),

(3, 'Fone de Ouvido Bluetooth', 'Fone de ouvido com cancelamento de ruído', 250.00, 30);

-- Inserindo pedidos

INSERT INTO pedidos (id_pedido, id_cliente, data_pedido, status) VALUES

(1, 1, '2024-11-01', 'Pendente'),

(2, 2, '2024-11-02', 'Enviado');

-- Inserindo itens nos pedidos

INSERT INTO itens_pedido (id_item, id_pedido, id_produto, quantidade, preco_unitario) VALUES

(1, 1, 1, 1, 1000.00),

(2, 1, 2, 2, 50.00),

(3, 2, 3, 1, 250.00);

-- Inserindo pagamentos

INSERT INTO pagamentos (id_pagamento, id_pedido, valor, data_pagamento, metodo_pagamento) VALUES

(1, 1, 1100.00, '2024-11-01', 'Cartão de Crédito'),

(2, 2, 250.00, '2024-11-02', 'Boleto');

Exemplos de Views Criadas para Relatórios

- View para Relatório de Vendas (Total de Vendas por Período). Essa view vai calcular o total de vendas realizadas, agrupadas por período (mensal), e por produto.

```
CREATE VIEW relatorio_vendas AS
SELECT p.id_produto,
p.nome AS produto,
SUM(ip.quantidade) AS quantidade_vendida,
SUM(ip.quantidade * ip.preco_unitario) AS total_vendas
FROM itens_pedido ip JOIN produtos p ON ip.id_produto = p.id_produto
JOIN pedidos pd ON ip.id_pedido = pd.id_pedido
WHERE pd.status = 'Enviado'
GROUP BY p.id_produto, p.nome;
```

- View para Consulta de Estoque

Esta view exibe o estoque disponível de cada produto, incluindo a quantidade atual disponível.

```
CREATE VIEW estoque_produtos AS
SELECT id_produto, nome, quantidade_em_estoque
FROM produtos
```

- View para Folha de Pagamento de Pedidos

Esta view gera um relatório de pagamentos realizados, mostrando os pedidos pagos, o valor pago e o método de pagamento.

```
CREATE VIEW vw_folha_pagamento AS
SELECT p.id_pedido, c.nome AS cliente, pg.valor, pg.metodo_pagamento, pg.data_pagamento
FROM pagamentos pg JOIN pedidos p ON pg.id_pedido = p.id_pedido
```

```
JOIN clientes c ON p.id_cliente = c.id_cliente;
```

CONCLUSÃO

Este trabalho abordou as SQL Views, destacando sua definição, função e importância em bancos de dados relacionais. As views são ferramentas essenciais para simplificar o acesso a dados complexos, melhorar a segurança e otimizar o desempenho de consultas, além de garantir a consistência e organização das informações. Elas ajudam a abstrair a complexidade do modelo de dados, promovem controle de acesso e facilitam a criação de relatórios dinâmicos.

Embora as views sejam poderosas, é fundamental utilizá-las com cautela, especialmente em projetos de grande escala, para evitar problemas de desempenho.

Práticas recomendadas incluem: manter as views simples, usar nomeação clara, monitorar o desempenho, garantir a segurança com controle de acesso e documentar adequadamente as views. Essas práticas asseguram o uso eficiente e seguro das views em sistemas SQL.

BIBLIOGRAFIA

King Host. https://king.host/wiki/artigo/views-mysql/#:~:text=Uma%20view%20%C3%A9%20um%20objeto,do%20esquema%20f%C3%ADsico%20da%20base.

Dev Media. https://www.devmedia.com.br/conceitos-e-criacao-de-views-no-sql-server/22390

Hashtag Treinamentos. https://www.hashtagtreinamentos.com/diferenca-tabelas-e-views-no-sql?gad_source=1&gclid=Cj0KCQiAlsy5BhDeARIsABRc6ZuB9gk8AGXvGpdxNGQBcm2-Oqpdbm8wql5NUROiWqapxybKtnNhQ5kaAnPqEALw_wcB

Stack Overflow. https://pt.stackoverflow.com/questions/35413/o-que-s%C3%A3o-views-em-sql-quais-vantagens-e-desvantagens-em-utilizar

Medium. https://medium.com/@flaviagaia/criando-e-utilizando-views-no-sql-simplificando-consultas-e-melhorando-a-produtividade-eae5144f036b

Microsoft Learn. https://learn.microsoft.com/en-us/sql/relational-databases/views/views?view=sql-server-ver16

LISTA DE FIGURAS