SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Curso

TÉCNICO EM DESENVOLVIMENTO

DE SISTEMAS

SQL Views - Conceito, Benefícios e Aplicações Práticas

Júlia Garcia de Carvalho

Sorocaba

Novembro – 2024

SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Júlia Garcia de Carvalho

SQL Views - Conceito, Benefícios e Aplicações Práticas

Pesquisa sobre SQL views, seus principais conceitos, benefícios e práticas nos dias

Prof. – Emerson Magalhães

Sorocaba
Novembro – 2024

SUMÁRIO

SQL Views

INTRODUÇÃO

Para começarmos a entender o SQL views temos que saber o básico dele, também conhecido como visões SQL, é a representação virtual dos dados que assemelha a tabelas, MAS não armazenam dados de forma independente. Elas são criadas a partir de consultas SQL, funcionado como uma janela que mostra dados específicos de uma ou mais tabelas, lembre-se que uma qualidade do SQL é que ele pode juntar uma ou mais tabelas e assim relacioná-las. As views são importantes para organizar e simplificar consultas complexas, restringir o acesso de dados sensíveis e reduzir a repetição de código, facilitando o controle sobre os dados acessados por usuários. O principal objetivo dessa pesquisa é explorar o que são as views, como utilizá-las em bancos de dados, e os tipos de operações que possibilitam. Também nessa pesquisa podemos analisar exemplos práticos no MySQL, para utilizar em todas as nossas futuras aulas de quarta do nosso querido professor Emerson Magalhães.

1. Fundamentos Teóricos das SQL Views

O vídeo fornece uma maneira poderosa de ajudá-lo a provar seu argumento. Ao clicar em Vídeo Online, você pode colar o código de inserção do vídeo que deseja adicionar

O que são Views e como elas funcionam no SQL

Como dito anteriormente a View é uma tabela virtual resultante de uma consulta simples no SQL que exibe os dados de uma tabela ou mais. Elas NÃO armazenam os dados diretamente, mas exibem dados SEMPRE atualizados, pois dependem das tabelas de origem.

1.2 Diferença entre Views e tabelas comuns

Ao contrário do que se imagina, existe diferença entre uma simples tabela e uma views, as tabelas simples, são simples, armazenam os dados fisicamente do banco, é a tabela mais tradicional que utilizamos. Já uma views é mais completa pois exibe dados com base em em consultas (nas tabelas simples) e só existem virtualmente, a não ser em casos de views materializadas, mas é outro sentido. Uma se liga com a outra.

1.3 Tipos de views

Ao No outro tópico eu havia dito que existe um caso, as views materializadas, mas agora vou te explicar mais duas desse tipo. Temos as views simples a básica que conhecemos, ela utiliza dados de uma única tabela e incluem seleção básica de colunas ou outros filtros específicos. Logo após temos as views complexas, aí dá uma complicada pois ela inclui junções (JOINS) e funções de agregações, extraindo dados de várias tabelas para exibir uma visão mais detalhada das informações precisas. E por fim as Views Materializadas, elas são armazenadas FISICAMENTE (tipo em um banco de

suporte) e necessita de atualizações periódicas para refletir mudanças nas tabelas originais, oferecendo vantagens de desempenho em consultas extremamente repetitivas.

1.4 Vantagens e desvantagens de usar uma Views

Após apresentar todos esses conceitos, analisamos as vantagens e desvantagens de usar uma views. Vantagens, podemos analisar que a views é muito simples para consultas complexas, isso porque ela ajuda a reduzir a complexidade de consultas que precisam de múltiplos joins e filtros, facilitando a criação de relatórios por exemplo de uma empresa. Outra vantagem é a segurança, pois ela permite ocultar colunas ou linhas específicas, limitando o acesso apenas a dados permitidos para o usuário. A facilidade de manutenção também chama atenção como uma vantagem pois elas centralizam consultas comuns e, ao modificar uma view, todos os relatórios e operações baseados nela são atualizados automaticamente. Analisando agora por outra face é as desvantagens, como o impacto no desempenho, pois as views complexas podem reduzir o desempenho, especialmente se requerem joins em grandes tabelas. Outra desvantagem é as limitações de atualizações, pois nem todas as views permitem atualizações, especialmente as que dependem de múltiplas tabelas ou funções de agregação. E as views materializadas sempre necessitam de atualizações periódicas, aumentando a carga no banco de dados. Causando assim as principais desvantagens.

1.5 Processo de Criação de Views no SQL

Para iniciar o comando **CREATE VIEW** é bem simples

```
1 •   CREATE VIEW nome_da_view_exemplo_empresaA AS
2     SELECT colunas
3     FROM tabela
4     WHERE condição;
```

nome_da_view_exemplo_empresaA: o nome da view a ser criada.

SELECT: consulta que define o conteúdo da view, podendo incluir junções, agregações, filtragens, etc.

FROM tabela: a tabela ou tabelas da qual você está extraindo os dados.

WHERE condição: uma condição opcional para filtrar os dados.

Lembra que eu expliquei sobre as Views simples, então segue um exemplo dele:

```
-- Views SIMPLES
CREATE VIEW clientes_sp AS
SELECT nome, endereco, cidade, estado
FROM clientes
WHERE cidade = 'São Paulo';
```

Aqui, a View **clientes_sp** seleciona apenas os clientes que moram na cidade de São Paulo.

Agora utilizaremos uma View de agregação

Uso de Funções como **SUM**, **AVG**, **COUNT** em uma View

```
CREATE VIEW vendas_produtos AS
SELECT produto_id,
       COUNT(*) AS total_vendas,
       SUM(valor_total) AS total_vendas_valor,
       AVG(valor_total) AS media_valor_venda
FROM vendas
GROUP BY produto_id;
```

Neste caso, a view **vendas_produtos** agrupa os dados por produto_id e calcula o total de vendas, a soma total das vendas e a média do valor das vendas para cada produto.

View de Junção: Combinar Dados de Várias Tabelas

Agora, vamos criar uma view que combina informações de vendas e clientes, mostrando o nome do cliente, o produto comprado e o valor total da venda.

```
CREATE VIEW vendas_clientes AS
SELECT c.nome AS cliente, v.produto_id, v.valor_total
FROM vendas v
JOIN clientes c ON v.cliente_id = c.id;
```

A view vendas_clientes faz uma junção entre as tabelas vendas e clientes, unindo os dados pela chave cliente_id da tabela vendas e o id da tabela clientes.

Exemplo de Criação de uma View Complexa

Uma view complexa pode envolver múltiplas junções, agregações e filtragens. Vamos criar uma view que mostra as vendas totais por mês e por cliente, considerando as vendas realizadas entre janeiro e junho.

```
CREATE VIEW vendas_mensais_clientes AS
SELECT c.nome AS cliente,
       YEAR(v.data_venda) AS ano,
       MONTH(v.data_venda) AS mes,
       SUM(v.valor_total) AS total_vendas
FROM vendas v
JOIN clientes c ON v.cliente_id = c.id
WHERE v.data_venda BETWEEN '2024-01-01' AND '2024-06-30'
GROUP BY c.nome, YEAR(v.data_venda), MONTH(v.data_venda)
ORDER BY cliente, ano, mes;
```

Essa view fornece o total de vendas de cada cliente para os primeiros seis meses de 2024, agrupando por nome do cliente e mês de venda.

Views Atualizáveis e Não Atualizáveis

Explicação sobre a Possibilidade de Atualizar Dados Diretamente em Views

As views podem ser atualizáveis ou não atualizáveis, dependendo da complexidade da consulta que define a view. Views atualizáveis permitem que você execute operações como INSERT, UPDATE ou DELETE diretamente nelas, enquanto views não atualizáveis não permitem essas operações.

Condições para uma View ser Atualizável:

- A view deve ser baseada em uma única tabela.
- A consulta não pode envolver funções agregadas (SUM, COUNT, AVG, etc.).
- A view não pode conter DISTINCT, GROUP BY, HAVING ou UNION.
- A view deve refletir as colunas de forma simples e direta, sem operações complexas que modifiquem os dados.

Exemplo de uma View Atualizável

```
CREATE VIEW clientes_atualizavel AS
SELECT id, nome, email
FROM clientes;
```

Neste exemplo, a view clientes_atualizavel é baseada em uma única tabela (clientes), e permite que você atualize, insira ou delete registros diretamente.

Exemplo de uma View Não Atualizável

```sql
CREATE VIEW vendas_totais_produto AS
SELECT produto_id,
       COUNT(*) AS total_vendas,
       SUM(valor_total) AS total_vendas_valor
FROM vendas
GROUP BY produto id;
```

Essa view não é atualizável porque ela utiliza funções agregadas (COUNT, SUM) e agrupa os resultados. Como a view não mapeia diretamente uma única linha de dados em uma tabela, não é possível realizar operações de atualização diretamente nela.

Estudo de Caso: Sistema de E-commerce

Vamos imaginar um banco de dados para uma loja de e-commerce. O banco de dados contém as seguintes tabelas:

- clientes: informações sobre os clientes (id, nome, email, endereço, etc.).
- produtos: informações sobre os produtos (id, nome, preço, estoque).
- vendas: informações sobre as vendas (id, data_venda, cliente_id, produto_id, quantidade, valor_total).
- estoque: controle do estoque (produto_id, quantidade_disponivel).

Exemplos de Views Criadas para Este Banco de Dados:

1. **View para Relatório de Vendas por Cliente**

```sql
CREATE VIEW relatorio_vendas_cliente AS
SELECT c.nome AS cliente,
       COUNT(v.id) AS total_vendas,
       SUM(v.valor_total) AS valor_total_vendas
FROM vendas v
JOIN clientes c ON v.cliente_id = c.id
GROUP BY c.nome;
```

ssa view fornece o total de vendas e o valor total das compras de cada cliente.

2. **View para Consulta de Estoque**

```
CREATE VIEW consulta_estoque AS
SELECT p.nome AS produto, e.quantidade_disponivel
FROM estoque e
JOIN produtos p ON e.produto_id = p.id;
```

Aqui, a view exibe o nome do produto junto com a quantidade disponível em estoque.

3. **View para Folha de Pagamento de Funcionários (se aplicável em um sistema de e-commerce com funcionários)**

```
CREATE VIEW folha_pagamento AS
SELECT f.nome AS funcionario, f.salario_base, f.bonus, (f.salario_base + f.bonus) AS salario_total
FROM funcionarios f;
```

Discussão sobre as Vantagens das Views no Cenário de E-commerce

- Abstração de Consultas Complexas: As views permitem abstrair a complexidade das consultas, o que facilita a manutenção do banco de dados. Usuários e desenvolvedores podem simplesmente consultar as views ao invés de escrever SQL complexo repetidamente.
- Segurança: Elas ajudam a controlar o acesso a dados sensíveis. Por exemplo, podemos criar uma view que exibe apenas dados específicos de vendas sem expor diretamente a tabela completa de vendas.
- Performance: Com views otimizadas, consultas complexas podem ser reutilizadas de forma eficiente, melhorando a performance em consultas repetidas.
- Facilidade de Relatórios: Views podem ser usados para criar relatórios prontos, como vendas mensais ou status do estoque, o que facilita a geração de relatórios no dia a dia.

Conclusão

Portanto podemos concluir que o SQL Views são "tabelas virtuais" que simplificam consultas complexas e aumentam a segurança ao limitar o acesso a dados. Elas são úteis para manutenção e organização em sistemas de banco de dados relacionais, com algumas limitações em desempenho e atualização.
Usar views pode melhorar significativamente o acesso a dados e reduzir o risco de erros em consultas repetitivas, mas é importante monitorar seu impacto no desempenho e suas limitações de atualização. Algumas práticas recomendadas: utilize a views para organizar e simplificar consultas complexas. Evite criar views excessivamente complexas que dificultam a atualização. Monitore o desempenho, especialmente em views que serão usadas com frequência. Elas são úteis para manutenção e organização em sistemas de banco de dados relacionais, com algumas limitações em desempenho e atualização.

Por fim, podemos usar views que podem melhorar significativamente o acesso a dados e reduzir o risco de erros em consultas repetitivas, mas é importante monitorar seu impacto no desempenho e suas limitações de atualização.