SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Curso

TÉCNICO EM DESENVOLVIMENTO

DE SISTEMAS

SQL Views

Lucas Duarte Geraldo

Sorocaba

Novembro - 2024

SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Lucas Duarte Geraldo

SQL Views

O que são SQL views e por que
elas são importantes
Prof. – Emerson

Sorocaba
Novembro - 2024

SUMÁRIO

INTRODUÇÃO

SQL Views são consultas salvas que funcionam como um objeto no banco de dados, permitindo que os usuários visualizem dados de uma ou mais tabelas de maneira organizada, simplificada e personalizada. Em vez de acessar diretamente as tabelas, os usuários podem visualizar dados específicos sem alterar as tabelas originais. Essa funcionalidade é crucial para bancos de dados relacionais, pois além de simplificar o acesso, protege informações sensíveis, controlando quais dados cada usuário pode acessar.

Esta pesquisa busca explicar o que são SQL Views e como funcionam, a importância de seu uso, as vantagens e limitações, além de exemplos práticos de sua criação e aplicação. Ao final, um estudo de caso demonstrará suas aplicações práticas e os benefícios de seu uso em um contexto cotidiano.

1. O que são views e como funcionam

Conceitualmente, uma SQL View é uma consulta nomeada armazenada no banco de dados que gera uma “visão” dos dados em tempo real, sem realmente armazená-los, exceto no caso das views materializadas, que mantêm os dados para melhorar a performance. Diferente de uma tabela, que armazena dados de forma permanente, a view representa uma consulta que é executada apenas quando o usuário a acessa, retornando os dados com base na estrutura da consulta. Existem diferentes tipos de views, como as simples, que representam dados de uma única tabela; as complexas, que podem combinar dados de várias tabelas e realizar agregações; e as materializadas, que armazenam os dados de forma persistente, sendo atualizadas periodicamente.

2. Tipos de Views

2.1. Simple View (View Simples)

A Simple View é a forma mais básica de view no Oracle. Ela é criada a partir de uma única tabela e contém apenas uma única consulta SELECT. Essas views são úteis para oferecer uma visão resumida ou segmentada dos dados originais. Vejamos um exemplo prático:

```
CREATE VIEW exemplo_simple_view AS

SELECT coluna1, coluna2

FROM tabela_origem

WHERE coluna3 = 'valor';
```

2.2. Complex View (View Complexa)

A Complex View é uma view que pode ser criada a partir de várias tabelas, usando joins, funções agregadas ou subconsultas. Isso permite que os usuários obtenham resultados consolidados ou personalizados a partir de várias fontes de dados. Essas views são especialmente úteis para simplificar consultas complexas e fornecer uma visão abrangente dos dados.

Vamos ver um exemplo:

```
CREATE VIEW exemplo_complex_view AS

SELECT t1.coluna1, t2.coluna2

FROM tabela1 t1

JOIN tabela2 t2 ON t1.chave = t2.chave

WHERE t1.coluna3 = 'valor';
```

2.3. Materialized View (View Materializada)

As Materialized Views são views que armazenam fisicamente os dados em disco. Isso permite que os resultados da consulta sejam pré-calculados e atualizados periodicamente, reduzindo a carga do servidor e melhorando o

desempenho em consultas repetitivas. Essas views são ideais para consultas com alto consumo de recursos ou que envolvam agregações complexas. Vejamos um exemplo prático:

```
CREATE MATERIALIZED VIEW exemplo_materialized_view

REFRESH COMPLETE ON DEMAND

AS

SELECT coluna1, COUNT(coluna2) AS total

FROM tabela_origem

GROUP BY coluna1;
```

3. Vantagens e desvantagens

3.1. Vantagens

Simplificam consultas complexas, melhorando a legibilidade e a manutenção do código;

Aumentam a segurança dos dados, permitindo controle de acesso personalizado;

Melhoram o desempenho ao pré-calcular resultados em Materialized Views, reduzindo a carga do servidor;

Possibilitam a criação de visões personalizadas para diferentes usuários, garantindo que cada um tenha acesso somente às informações relevantes.

3.2. Desvantagens

A view acaba escondendo uma complexidade da query, podendo assim

enganar o desenvolvedor quanto odesempenho necessário para acessar as informações;

Cria uma camada extra, com isso são mais objetos para administrar,

considerando um aumento de complexidade;

Pode limitar até demais o acesso do usuário, impedindo certas tarefas.

4. Processo de criação de views no SQL

O processo de criação de views no SQL permite a criação de "visões" virtuais de dados armazenados em tabelas. Essas views são essencialmente consultas salvas que podem ser reutilizadas para simplificar o acesso a informações complexas ou para proteger dados sensíveis exibindo apenas o necessário.

A instrução básica para criar uma view em SQL é:

```sql
CREATE VIEW clientes_ativos AS
SELECT nome, email, cidade
FROM clientes
WHERE status = 'ativo';
```

4.1. Exemplos de views simples

4.1.1. View de Filtragem

Uma view de filtragem é usada para selecionar colunas e linhas específicas de uma tabela. Por exemplo, para criar uma view com apenas os clientes ativos:

```sql
CREATE VIEW clientes_ativos AS
SELECT nome, email, cidade
FROM clientes
WHERE status = 'ativo';
```

4.1.2. View de Agregação

Essa view usa funções agregadas como SUM, AVG, e COUNT para sumarizar dados. Por exemplo, para calcular a média dos salários dos funcionários por departamento:

```sql
CREATE VIEW media_salarial_departamento AS
SELECT departamento, AVG(salario) AS media_salarial
FROM funcionarios
GROUP BY departamento;
```

4.1.3. View de junção

Uma view de junção combina dados de várias tabelas. Por exemplo, para exibir pedidos com informações do cliente:

```sql
CREATE VIEW pedidos_com_clientes AS
SELECT p.id_pedido, c.nome AS cliente, p.data_pedido, p.valor_total
FROM pedidos p
JOIN clientes c ON p.id_cliente = c.id_cliente;
```

4.2. View complexa

Uma view complexa pode combinar junções, agregações e filtros. Por exemplo, para uma view que exiba a quantidade total de pedidos e o valor médio desses pedidos por cliente:

```sql
CREATE VIEW resumo_pedidos_clientes AS
SELECT c.nome AS cliente, COUNT(p.id_pedido) AS total_pedidos, AVG(p.valor_total) AS media_valor_pedido
FROM clientes c
JOIN pedidos p ON c.id_cliente = p.id_cliente
GROUP BY c.nome
HAVING COUNT(p.id_pedido) > 1;
```

4.3. Views atualizáveis e não atualizáveis

Algumas views permitem a atualização direta dos dados subjacentes, enquanto outras não. Uma view é **atualizável** se seguir certas condições, como:

A view se basear em uma única tabela.

Não conter funções agregadas ou DISTINCT.

Não incluir cláusulas GROUP BY ou HAVING.

Não conter expressões de junção complexas.

```sql
CREATE VIEW clientes_sp AS
SELECT id_cliente, nome, cidade
FROM clientes
WHERE cidade = 'São Paulo';
```

```sql
CREATE VIEW media_pedidos AS
SELECT id_cliente, AVG(valor_total) AS media_valor
FROM pedidos
GROUP BY id_cliente;
```

5. Exemplo prático

Foi criada um banco de dados para uma loja de roupas na internet. Utilizando de 3 tabelas e após uma demonstração das views possíveis para esse caso.

5.1. Tabelas

```sql
CREATE TABLE clientes (
    id_cliente INT PRIMARY KEY,
    nome VARCHAR(100),
    email VARCHAR(100),
    cidade VARCHAR(50),
    status VARCHAR(10) -- 'ativo' ou 'inativo'
);
```

```sql
CREATE TABLE produtos (
    id_produto INT PRIMARY KEY,
    nome_produto VARCHAR(100),
    preco DECIMAL(10, 2),
    estoque INT
);
```

```sql
CREATE TABLE pedidos (
    id_pedido INT PRIMARY KEY,
    id_cliente INT,
    id_produto INT,
    data_pedido DATE,
    quantidade INT,
    valor_total DECIMAL(10, 2),
    FOREIGN KEY (id_cliente) REFERENCES clientes(id_cliente),
    FOREIGN KEY (id_produto) REFERENCES produtos(id_produto)
);
```

5.2. Views possíveis

5.2.1. View de Filtragem: Clientes Ativos

Essa view mostra apenas os clientes que estão com o status ativo. Isso é útil para o time de marketing, que deseja enviar campanhas para clientes ativos.

```sql
CREATE VIEW clientes_ativos AS
SELECT id_cliente, nome, email, cidade
FROM clientes
WHERE status = 'ativo';
```

Vantagem: Facilita o acesso rápido aos dados de clientes ativos, reduzindo a necessidade de filtrar manualmente em cada consulta. Garante que campanhas e comunicações sejam direcionadas apenas aos clientes em atividade.

5.2.2. View de agregação: Estoque de produtos

Essa view calcula o valor total de estoque de cada produto, multiplicando o preço pelo estoque disponível.

```
CREATE VIEW valor_estoque_produtos AS
SELECT id_produto, nome_produto, preco, estoque, (preco * estoque) AS valor_estoque
FROM produtos;
```

Vantagem: Fornece uma visão clara e direta do valor total do estoque por produto, facilitando a análise de inventário e auxiliando na tomada de decisões de compras e reposições.

5.2.3. View de Junção: Pedidos com Informações de Clientes

Essa view une dados de pedidos e clientes, fornecendo uma visão completa de cada pedido, incluindo informações sobre o cliente.

```
CREATE VIEW pedidos_detalhados AS
SELECT p.id_pedido, c.nome AS cliente, p.data_pedido, p.quantidade, p.valor_total
FROM pedidos p
JOIN clientes c ON p.id_cliente = c.id_cliente;
```

Vantagem: Permite que a equipe de atendimento acesse rapidamente detalhes dos pedidos e identifique o cliente associado, o que agiliza o suporte ao cliente e reduz a complexidade de consultas com múltiplas junções.

5.2.4. View Complexa: Resumo de Vendas por Cliente

Essa view usa junções e agregações para exibir o número total de pedidos e o valor médio de cada pedido por cliente.

```sql
CREATE VIEW resumo_vendas_clientes AS
SELECT c.nome AS cliente, COUNT(p.id_pedido) AS total_pedidos, AVG(p.valor_total) AS media_valor_pedido
FROM clientes c
JOIN pedidos p ON c.id_cliente = p.id_cliente
GROUP BY c.nome;
```

Vantagem: Auxilia a equipe de vendas e marketing a identificar os clientes mais engajados, baseando-se no total de pedidos, e a calcular o valor médio das compras, ajudando no planejamento de estratégias de retenção e recompensas.

CONCLUSÃO

Uma view em SQL é uma tabela virtual que exibe dados de outras tabelas, permitindo simplificar consultas complexas, reforçar a segurança e melhorar a organização do banco de dados. Elas não armazenam dados diretamente, mas geram resultados em tempo real, ajudando a ocultar detalhes de implementação e a restringir o acesso a informações sensíveis.

No entanto, views complexas podem impactar o desempenho, especialmente em grandes volumes de dados, e nem sempre permitem operações de manipulação (inserção, atualização, exclusão). Para usar views de forma eficaz, é recomendável nomeá-las de forma clara, documentar seu propósito, monitorar seu impacto no desempenho, e evitar o uso excessivo de views aninhadas. Views materializadas podem melhorar a performance em leituras frequentes, mas precisam ser atualizadas periodicamente. Essas práticas ajudam a garantir que views adicionem valor ao projeto sem comprometer a eficiência do sistema.

REFERÊNCIAS

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

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

https://ric.cps.sp.gov.br/bitstream/123456789/9920/1/bancodedados_2020_1_anabeatrizsantos_osbeneficiosdautilizacaodeviews.pdf

https://www.profissionaloracle.com.br/2023/07/29/guia-das-views-no-banco-de-dados-oracle-tipos-funcionalidades-e-exemplos-praticos/