SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Curso

TÉCNICO EM DESENVOLVIMENTO

DE SISTEMAS

SQL Views - Conceito, Benefícios e Aplicações Práticas

João Alexandre da Silva Pereira

Sorocaba

Março – 2024

SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

João Alexandre da Silva Pereira

SQL Views - Conceito, Benefícios e Aplicações Práticas

Sorocaba
Março – 2024

SUMÁRIO

INTRODUÇÃO 3
1. COMO A UTILILIZAÇÃO DO IOT PODE TORNAR A PRODUÇÃO DAS INDÚSTRIAS MAIS EFICIENTE? 4
1.1. COMO SE COMUNICAM? 6
CONCLUSÃO 7
BIBLIOGRAFIA 8

• INTRODUÇÃO

1 - Definição de SQL Views e sua função em bancos de dados.

Em banco de dados relacionais, uma visão (ou "view"), é uma consulta (instrução SQL) armazenada no banco de dados geralmente chamada de "tabela virtual". Essa tabela virtual é derivada de uma ou mais tabelas já existentes no banco de dados e podem ser consultadas como se fossem uma tabela real. Ao criar uma view, o resultado da consulta não é armazenado fisicamente no banco de dados, a cada execução os dados são acessados diretamente das tabelas de origem. As views geralmente são usadas para simplificar consultas complexas, fornece uma exibição personalizada dos dados ou limitar acesso a dados das tabelas reais.

2 - Importância das views em sistemas de banco de dados relacionais.

As views são muito usadas para ajudar dar entendimento ao projeto lógico do banco de dados. São muitos os motivos e vantagens para utiliza-las nos projetos. a) É possível ter o controle das informações que o usuário terá acesso.

3 - Objetivo e abrangência da pesquisa.

Explicitar o objetivo, que pode ser a análise das principais funcionalidades, vantagens, limitações e as práticas recomendadas para uso de SQL Views em projetos reais de banco de dados.

Abrangência: Cobrir conceitos, funcionalidades, exemplos práticos e um estudo de caso realista.

- # Fundamentos Teóricos das SQL Views

1 - O que são views e como elas funcionam no SQL.

View é um resultado originado de uma consulta pré-definida. Essencialmente é um metadado que mapeia uma query para outra, por isto pode ser considerado como uma tabela virtual. Como o próprio nome diz, ela representa uma visão de dados e não contém dados. Com ela você tem a ilusão que está vendo uma tabela que não existe. Claro que o que você vê nesta tabela existe de outra forma no banco

2 - Diferença entre views e tabelas comuns.

1 - views simples:

Estas são views básicas que consistem em uma consulta direta e, geralmente, não contêm operações complexas, como agregações ou junções de várias tabelas.

```sql
CREATE VIEW cliente_info AS
SELECT nome, telefone, email
FROM clientes;
```

2 - Views Complexas:

Incluem consultas com operações como JOIN, agregações (SUM, COUNT, etc.), e podem envolver várias tabelas.

Útil para consolidar dados de várias tabelas ou realizar cálculos.

```sql
CREATE VIEW vendas_detalhadas AS
SELECT v.id AS venda_id, v.data, c.nome AS cliente, p.nome AS produto, v.quantidade, v.valor_total
FROM vendas v
JOIN clientes c ON v.cliente_id = c.id
JOIN produtos p ON v.produto_id = p.id
WHERE v.data >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH);
```

3 - views materializadas

Tabela Temporária com Atualização Manual: Crie uma tabela para armazenar os dados da consulta e atualize-a manualmente ou periodicamente.

Triggers para Atualização Automática: Use triggers para atualizar a tabela automaticamente ao ocorrerem alterações nos dados originais.

Ferramentas de ETL: Use ferramentas externas para automatizar a criação e atualização da tabela materializada.

```sql
-- Criar a tabela para armazenar os dados da "view materializada"
CREATE TABLE vendas_ultimo_mes_materializada AS
SELECT v.id, v.data, c.nome AS cliente, p.nome AS produto, v.quantidade, v.valor_total
FROM vendas v
JOIN clientes c ON v.cliente_id = c.id
JOIN produtos p ON v.produto_id = p.id
WHERE v.data >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH);
```

- ## Vantagens das Views:

- Simplificação de consultas complexas: torna consultas complexas mais acessíveis, permitindo que os usuários realizem uma consulta somente para visualização em vez de uma extensa. acessível, permitindo que os usuários realizem uma consulta somente para visualização, em vez de uma consulta extensa.

- Aumentando a segurança: Restrinja o acesso para certas linhas e colunas, salvaguardando protegendo dados sensíveis. confidenciais.

- Facilidade de Controle e Manutenção: Permite consultas frequentes centralizando-as em uma única definição. em uma única definição.

Desvantagens das Views:

- Impacto no desempenho: como as visualizações são recalculados em cada câmara, eles podem reduzir o desempenho.

- Limitações nas operações de atualização : Alguns pontos pontos de vista (particularmente os mais complicados) não nos permitem-nos permitem atualizar os dados diretamente para atualizar os dados diretamente.

- Manutenção de visualizações materializadas: Da mesma forma, uma visão materializada requer atualização manual ou automatizada para manter os dados atualizados, aumentando a carga de manutenção. Atualização manual ou automatizada para manter os dados atualizados, aumentando a carga de manutenção.

- ## Processo de Criação de Views no SQL

As Views são uma forma de armazenar uma consulta SQL em um objeto de banco de dados, permitindo fácil reutilização e simplificação do código. Elas ajudam a criar "visões" específicas dos dados, sem a necessidade de duplicar os dados no banco.

FILTRAGEM:

Seleciona colunas específicas e aplica filtros em linhas.

```sql
-- Exemplo: View que filtra clientes ativos
CREATE VIEW clientes_ativos AS
SELECT id, nome, email
FROM clientes
WHERE status = 'ativo';
```

Agregação:

Usa funções agregadas, como SUM, AVG e COUNT, para obter uma visão resumida dos dados.

```sql
-- Exemplo: View que calcula o total de vendas por mês
CREATE VIEW total_vendas_mensal AS
SELECT DATE_FORMAT(data, '%Y-%m') AS mes,
       SUM(valor_total) AS total_vendas,
       COUNT(id) AS total_transacoes
FROM vendas
GROUP BY mes;
```

Junção:

Combina dados de várias tabelas para fornecer uma visão mais completa e detalhada.

```sql
-- Exemplo: View que combina informações de vendas com dados de clientes
CREATE VIEW vendas_detalhadas AS
SELECT v.id AS venda_id,
       v.data,
       c.nome AS cliente,
       p.nome AS produto,
       iv.quantidade,
       iv.preco_unitario,
       (iv.quantidade * iv.preco_unitario) AS valor_total
FROM vendas v
JOIN clientes c ON v.cliente_id = c.id
JOIN itens_venda iv ON v.id = iv.venda_id
JOIN produtos p ON iv.produto_id = p.id;
```

Views Atualizáveis e Não Atualizáveis

Views Atualizável: permite entrada, exclusão ou atualização direta de dados. Entrada, exclusão ou atualização. Geralmente, ele se com base em uma única tabela sem adições complexas ou adições ou junções.

```sql
CREATE VIEW clientes_info AS
SELECT id, nome, telefone, email
FROM clientes;
```

View nãos atualizáveis: não permite modificação direta de permitir dados. Modificação direta de dados. Isso ocorre quando a visão envolve juntas, agregações ou outras operações complexas. Operações

```sql
CREATE VIEW total_vendas_por_cliente AS
SELECT cliente_id, SUM(valor_total) AS total_gasto
FROM vendas
GROUP BY cliente_id;
```

ESTUDO DE CASO :

```sql

-- Tabela de clientes
CREATE TABLE clientes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100),
    email VARCHAR(100),
    telefone VARCHAR(20),
    endereco VARCHAR(255)
);

-- Tabela de produtos
CREATE TABLE produtos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100),
    descricao TEXT,
    preco DECIMAL(10, 2),
    categoria VARCHAR(50)
);

-- Tabela de vendas
CREATE TABLE vendas (
    id INT AUTO_INCREMENT PRIMARY KEY,
    cliente_id INT,
    data DATE,
    total DECIMAL(10, 2),
    FOREIGN KEY (cliente_id) REFERENCES clientes(id)
);

  -- Tabela de itens de cada venda
CREATE TABLE itens_venda (
    id INT AUTO_INCREMENT PRIMARY KEY,
    venda_id INT,
    produto_id INT,
    quantidade INT,
    preco_unitario DECIMAL(10, 2),
    FOREIGN KEY (venda_id) REFERENCES vendas(id),
    FOREIGN KEY (produto_id) REFERENCES produtos(id)
);

  -- Tabela de estoque
CREATE TABLE estoque (
    id INT AUTO_INCREMENT PRIMARY KEY,
    produto_id INT,
    quantidade INT,
    FOREIGN KEY (produto_id) REFERENCES produtos(id)
);


```

```sql

-- Essa view exibe o total de vendas por dia, com base na tabela vendas.
CREATE VIEW relatorio_vendas_diarias AS
SELECT data,
       COUNT(id) AS total_vendas,
       SUM(total) AS receita_total
FROM vendas
GROUP BY data;



-- View de Vendas por Produto
CREATE VIEW vendas_por_produto AS
SELECT p.nome AS produto,
       SUM(iv.quantidade) AS quantidade_total_vendida,
       SUM(iv.quantidade * iv.preco_unitario) AS receita_total
FROM itens_venda iv
JOIN produtos p ON iv.produto_id = p.id
GROUP BY p.id;



-- View de Consulta de Estoque
CREATE VIEW consulta_estoque AS
SELECT p.nome AS produto,
       p.categoria,
       e.quantidade AS estoque_atual,
       CASE
           WHEN e.quantidade < 10 THEN 'Baixo Estoque'
           WHEN e.quantidade BETWEEN 10 AND 50 THEN 'Estoque Médio'
           ELSE 'Estoque Alto'
       END AS status_estoque
FROM produtos p
JOIN estoque e ON p.id = e.produto_id;



-- compras por cliente
CREATE VIEW compras_por_cliente AS
SELECT c.nome AS cliente,
       c.email,
       v.data AS data_compra,
       SUM(iv.quantidade * iv.preco_unitario) AS total_compra
FROM clientes c
JOIN vendas v ON c.id = v.cliente_id
JOIN itens_venda iv ON v.id = iv.venda_id
GROUP BY c.id, v.data;


```

• CONCLUSÃO

Principais Pontos:

Views são consultas SQL armazenadas no banco de dados, oferecendo uma "tabela virtual" que simplifica o acesso aos dados.

São úteis para simplificar consultas, melhorar a segurança e facilitar a manutenção de consultas frequentes.

No entanto, podem impactar o desempenho e limitar atualizações, especialmente em views complexas com junções ou agregações.

Considerações Finais:

Views são essenciais para organizar dados e controlar acessos em projetos de banco de dados, mas devem ser usadas com cautela para evitar problemas de desempenho e atualização.

Práticas Recomendadas:

Priorizar views simples e eficientes.

Evitar complexidade desnecessária, que pode prejudicar o desempenho.

Planejar atualizações de views materializadas adequadamente.

Monitorar o impacto no desempenho.

Utilizar views para controlar acessos a dados sensíveis.

Em resumo, views são uma ferramenta valiosa quando bem utilizadas, ajudando a organizar e otimizar a consulta de dados em sistemas de banco de dados.

BIBLIOGRAFIA

https://sae.unb.br/cae/conteudo/unbfga/lbd/banco2_visoes.html#:~:text=Em%20banco%20de%20dados%20relacionais,se%20fossem%20uma%20tabela%20real

https://pt.stackoverflow.com/questions/35413/o-que-s%C3%A3o-views-em-sql-quais-vantagens-e-desvantagens-em-utilizar#:~:text=View%20%C3%A9%20um%20resultado%20originado,dados%20e%20n%C3%A3o%20cont%C3%A9m%20dados

-

-