**SERVIÇO NACIONAL DE APRENDIZAGEM**

**INDUSTRIAL SENAI “GASPAR RICARDO JUNIOR”**

**Curso**

**TÉCNICO EM DESENVOLVIMENTO**

**DE SISTEMAS**

# *1.SQL Views - Conceito, Benefícios e Aplicações Práticas*

Marco Antônio da Costa Silva

Sorocaba
Março – 2024


**SERVIÇO NACIONAL DE APRENDIZAGEM**

**INDUSTRIAL SENAI “GASPAR RICARDO JUNIOR”**

Marco Antônio da Costa Silva


# SQL Views - Conceito, Benefícios e Aplicações Práticas

Pesquisa sobre SQL Views

Profº Emerson Magalhães


Sorocaba

Março – 2024

# Introdução

SQL Views, ou simplesmente "views" (visões, em português), são estruturas criadas para apresentar dados específicos de uma ou mais tabelas em um banco de dados, funcionando como uma "janela" para visualizar essas informações sem modificar os dados originais. As views são valiosas para sistemas de banco de dados relacionais, pois facilitam a criação de consultas personalizadas, restrições de acesso e a reutilização de consultas complexas, ajudando a simplificar o trabalho com grandes volumes de dados. Esta pesquisa visa fornecer um entendimento completo sobre SQL Views, incluindo sua definição, tipos, vantagens, desvantagens e aplicações práticas.

# Fundamentos Teóricos das SQL Views:

## O que São SQL Views e Como Funcionam:

Uma SQL View é um objeto de banco de dados que armazena uma consulta SQL, permitindo visualizar um conjunto específico de dados de uma ou mais tabelas. As views ajudam a organizar dados de maneira lógica e personalizada para os usuários.

## Diferença entre Views e Tabelas Comuns:

A principal diferença entre views e tabelas comuns é que as views não armazenam dados fisicamente; elas apenas mostram dados existentes nas tabelas. Já as tabelas armazenam dados permanentemente no banco.

## Tipos de Views:

**Views Simples**: Baseadas em uma única tabela e sem operações complexas.

**Views Complexas**: Utilizam junções e agregações para combinar e resumir dados de várias tabelas.

**Views Materializadas**: Se o banco de dados suportar, são views que armazenam dados fisicamente para melhorar o desempenho.

# Vantagens e Desvantagens de usar Views:

## Vantagens:

**Simplificação de Consultas**: Views podem simplificar consultas SQL, especialmente as que envolvem junções e agregações.

**Aumento da Segurança**: As views permitem limitar o acesso a certas colunas e linhas, o que pode ser útil para controlar quais dados são visíveis para diferentes usuários.

**Facilita Manutenção**: Views podem ser reutilizadas, facilitando o gerenciamento de consultas frequentemente usadas.

## Desvantagens:

**Impacto no Desempenho**: Views complexas podem tornar as consultas mais lentas, especialmente se envolverem junções e agregações em grandes volumes de dados.

**Limitações para Atualizações**: Nem todas as views permitem operações de atualização, especialmente as complexas.

**Manutenção de Views Materializadas**: As views materializadas precisam de atualização periódica para garantir que os dados estejam sincronizados com as tabelas de origem.

**Processo de Criação de Views no SQL:**

**Exemplos de Views Simples**:

```sql
-- VIEW SIMPLES Cria uma view chamada 'view_clientes_basico' para selecionar apenas id_cliente e nome
CREATE VIEW view_clientes_basico AS
SELECT
    id_cliente,   -- Seleciona a coluna id_cliente
    nome          -- Seleciona a coluna nome
FROM
    clientes;     -- Tabela base: clientes
```

**View de Filtragem**: Seleção de colunas e linhas específicas.

```sql
-- VIEW DE FILTRAGEM Cria uma view chamada 'view_eletronicos_caro' para produtos da categoria "Eletrônicos" com preço acima de 1000
CREATE VIEW view_eletronicos_caro AS
SELECT
    id_produto, -- Seleciona a coluna id_produto
    nome,       -- Seleciona a coluna nome do produto
    preco       -- Seleciona a coluna preco do produto
FROM
    produtos    -- Tabela base: produtos
WHERE
    categoria = 'Eletrônicos' AND -- Filtra a categoria para "Eletrônicos"
    preco > 1000;               -- Filtra o preço para produtos acima de 1000
```

**View de Agregação**: Uso de funções como SUM, AVG, COUNT.

```sql
-- VIEW DE AGREGAÇÃO Cria uma view chamada 'view_agregacao_vendas' para agrupar e calcular total de vendas e média por produto
CREATE VIEW view_agregacao_vendas AS
SELECT
    id_produto,                    -- Seleciona a coluna id_produto para agrupamento
    SUM(valor_total) AS total_vendas, -- Calcula o total de vendas por produto
    AVG(quantidade) AS media_quantidade -- Calcula a quantidade média por produto
FROM
    vendas -- Tabela base: vendas
GROUP BY
    id_produto; -- Agrupa os resultados por produto
```

**View de Junção**: Combina dados de múltiplas tabelas.

```sql
    -- VIEW DE JUNÇÃO Cria uma view chamada 'view_pedidos_clientes' para combinar dados das tabelas clientes e pedidos
CREATE VIEW view_pedidos_clientes AS
SELECT
    clientes.id_cliente,       -- Seleciona a coluna id_cliente da tabela clientes
    clientes.nome_cliente,     -- Seleciona o nome do cliente
    pedidos.id_pedido,         -- Seleciona a coluna id_pedido da tabela pedidos
    pedidos.data_pedido,       -- Seleciona a data do pedido
    pedidos.valor_total        -- Seleciona o valor total do pedido
FROM
    clientes                   -- Tabela base: clientes
JOIN
    pedidos ON clientes.id_cliente = pedidos.id_cliente; -- Realiza a junção baseada no id_cliente
```

**Exemplo de View Complexa**:

```sql
    -- VIEw COMPLEXA Cria uma view complexa chamada 'view_media_salario_departamento' para calcular a média salarial e listar os funcionários
CREATE VIEW view_media_salario_departamento AS
SELECT
    d.nome_departamento,       -- Seleciona o nome do departamento
    f.nome AS nome_funcionario, -- Seleciona o nome do funcionário
    f.salario,                 -- Seleciona o salário do funcionário
    AVG(f.salario) OVER(PARTITION BY f.departamento) AS media_salario_departamento -- Calcula a média salarial por departamento
FROM
    funcionarios f             -- Tabela base: funcionarios (usando alias 'f')
JOIN
    departamentos d ON f.departamento = d.id_departamento; -- Junta funcionários e departamentos pelo id do departamento
```

Uma view complexa poderia combinar informações de produtos, vendas e clientes, fornecendo uma visão detalhada do desempenho de vendas de cada produto.

### Views Atualizáveis e Não Atualizáveis:

**Explicação**: Views atualizáveis permitem modificar dados diretamente através delas, enquanto as não atualizáveis não permitem essa interação.

**Condições para Atualização**: Para que uma view seja atualizável, geralmente deve ser baseada em uma única tabela, sem agregações ou junções complexas.

**Exemplos**: Uma view baseada em uma única tabela com seleção de colunas pode ser atualizável, enquanto uma view com agregações e junções normalmente não será.

## Estudo de Caso:

### Banco de Dados Fictício

Vamos considerar um banco de dados de uma loja de e-commerce, contendo tabelas como Clientes, Produtos, Vendas e Estoque.

### Exemplos de Views Criadas para o Banco de Dados

**View de Relatório de Vendas**: Mostra o total de vendas por produto e

cliente.

```sql
-- Cria uma view chamada 'view_relatorio_vendas' para exibir dados de pedidos e clientes
CREATE VIEW view_relatorio_vendas AS
SELECT
    p.id_pedido,                -- ID do pedido
    c.nome_cliente,             -- Nome do cliente que fez o pedido
    p.data_pedido,              -- Data do pedido
    p.valor_total               -- Valor total do pedido
FROM
    pedidos p                   -- Tabela pedidos, com alias 'p'
JOIN
    clientes c ON p.id_cliente = c.id_cliente; -- Junta com clientes para obter o nome do cliente
```

**View de Estoque**: Lista apenas produtos com estoque baixo.

```sql
-- Cria uma view chamada 'view_estoque_baixo' para exibir produtos com estoque baixo
CREATE VIEW view_estoque_baixo AS
SELECT
    id_produto,                 -- ID do produto
    nome_produto,               -- Nome do produto
    estoque                     -- Quantidade em estoque
FROM
    produtos                    -- Tabela produtos
WHERE
    estoque < 10;               -- Exibe apenas produtos com menos de 10 unidades em estoque
```

**View de Folha de Pagamento (para RH)**: Mostra dados específicos para cálculos de salários e benefícios.

```sql
-- Cria uma view chamada 'view_gastos_fornecedores' para exibir gastos por fornecedor
CREATE VIEW view_gastos_fornecedores AS
SELECT
    f.nome_fornecedor,          -- Nome do fornecedor
    SUM(ip.quantidade * ip.preco_unitario) AS total_gasto -- Calcula o total gasto
FROM
    fornecedores f              -- Tabela fornecedores
JOIN
    produtos p ON f.id_fornecedor = p.id_produto  -- Assume-se que cada produto possui um fornecedor
JOIN
    itens_pedido ip ON p.id_produto = ip.id_produto -- Junta com itens_pedido para acessar a quantidade e preço
GROUP BY
    f.nome_fornecedor;          -- Agrupa o total gasto por fornecedor
```

**Vantagens das Views:**

Views como essas permitem simplificar consultas frequentes e aumentar a segurança, permitindo que diferentes setores acessem apenas os dados necessários para suas operações.

## Conclusão:

Em virtude dos fatos, vimos que as SQL Views são ferramentas importantes para simplificar consultas, melhorar a segurança e tornar a manipulação de dados mais eficiente. Views são essenciais para projetos de banco de dados que envolvem muitos dados e usuários com diferentes permissões de acesso. É importante definir views com cuidado para evitar problemas de desempenho e limitar o acesso a dados confidenciais.

## Referências:


- Documentação SQL e materiais de estudo sobre bancos de dados

- Artigos sobre SQL Views e segurança em bancos de dados