**SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL**

**SENAI “GASPAR RICARDO JUNIOR”**

**Curso**

**TÉCNICO EM DESENVOLVIMENTO**

**DE SISTEMAS**

# SQL Views - Conceito, Benefícios e Aplicações Práticas

Isabela Etore Lopes

Sorocaba
Novembro – 2024

**SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL**

**SENAI “GASPAR RICARDO JUNIOR”**

Isabela Etore Lopes

# SQL Views - Conceito, Benefícios e Aplicações Práticas

Este trabalho aborda as SQL Views, que são visões virtuais usadas para simplificar consultas e aumentar a segurança em banco de dados.

Prof. – Emerson Magalhães

Sorocaba
Novembro – 2024

# 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, no trabalho contém também um pouco sobre o processo de criação de views e exemplos práticos que ilustram suas aplicações no dia a dia.

# INTRODUÇÃO

As SQL Views, ou simplesmente "views", são muito úteis nos bancos de dados relacionais, especialmente para quem precisa organizar e acessar informações de forma prática. Basicamente, uma view funciona como uma consulta salva, que cria uma “visão” dos dados, sem duplicá-los ou precisar armazená-los de novo. Isso ajuda bastante na hora de acessar informações específicas e também a manter o banco de dados mais seguro, já que dá para limitar o que cada pessoa pode ver

As views se tornaram comuns em áreas como relatórios e análise de dados, porque facilitam a vida de quem precisa fazer consultas mais complexas. Assim, o objetivo deste trabalho é entender melhor o que são essas views, como funcionam, suas vantagens e desvantagens e como podem ser aplicadas. No final, também vamos ver exemplos práticos para visualizar de que forma elas podem ser usadas no dia a dia e como ajudam a deixar o trabalho com dados mais eficiente e seguro.

# 1. FUNDAMENTOS TEÓRICOS DAS SQL VIEWS

## 1.1. O QUE SÃO VIEWS E COMO ELAS FUNCIONAM NO SQL

Os Views são objetos de banco de dados que representam consultas pré-definidas e são tratadas como tabelas virtuais. Elas não armazenam dados por conta própria, mas sim uma definição de consulta que recupera dados de outras tabelas. Ao acessar uma view, o banco de dados executa a consulta associada, retornando dados atualizados em tempo real.

## 1.2. DIFERENÇA ENTRE VIEWS E TABELAS COMUNS

A principal diferença é que, enquanto tabelas armazenam dados, views apenas armazenam uma consulta. As views não ocupam espaço significativo de armazenamento, a menos que sejam materializadas, o que ocorre em alguns sistemas para melhorar o desempenho.

## 1.3. TIPOS DE VIEWS

### 1.3.1. VIEWS SIMPLES

Baseadas em uma única tabela, permitindo operações básicas de seleção e filtragem.

### 1.3.2. VIEWS COMPLEXAS

Combinam dados de várias tabelas e frequentemente usam operações de junções e agregações.

### 1.3.3. VIEWS MATERIALIZADAS

Armazenam uma cópia dos dados gerados pela consulta, atualizada periodicamente.

# 2. VANTAGENS E DESVANTAGENS DE USAR VIEWS

## 2.1. VANTAGENS:

### 2.1.1. SIMPLIFICAÇÃO DE CONSULTAS COMPLEXAS

As views encapsulam consultas que combinam e manipulam dados de várias tabelas, facilitando o uso e o entendimento das consultas.

### 2.1.2. AUMENTO DA SEGURANÇA

Ao ocultar colunas sensíveis, as views ajudam a proteger dados confidenciais.

### 2.1.3. FACILIDADE NA MANUTENÇÃO

Views permitem que consultas complexas sejam centralizadas e reutilizadas por vários usuários.

## 2.2. DESVANTAGENS:

### 2.2.1. IMPACTOS NO DESEMPENHO

Views complexas podem reduzir o desempenho de consultas, especialmente se incluírem junções ou cálculos pesados.

### 2.2.2. LIMITAÇÕES EM ATUALIZAÇÕES

Em muitos casos, views não suportam operações de atualização, como inserção ou exclusão de dados.

### 2.2.3. MANUTENÇÃO DE VIEWS MATERIALIZADAS

Requerem uma atualização periódica para refletir mudanças nos dados subjacentes.

Baseadas em uma única tabela, permitindo operações básicas de seleção e filtragem.

# 3. PROCESSO DE CRIAÇÃO DE VIEWS NO SQL

## 3.1. INSTRUÇÃO CREATE VIEW: SINTAXE E PARÂMETROS

A criação de uma view é feita pela instrução CREATE VIEW, seguida do nome da view e da consulta SQL que ela deve encapsular.

```sql
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Essa instrução cria uma nova view chamada view_name que contém os dados resultantes da consulta SELECT. As views não armazenam dados fisicamente, mas sim a definição da consulta.

## 3.2. EXEMPLOS DE VIEWS

### 3.2.1. VIEW DE FILTRAGEM

Seleciona apenas colunas e linhas relevantes.

```sql
CREATE VIEW ActiveCustomers AS
SELECT customer_id, customer_name
FROM Customers
WHERE status = 'Active';
```

Essa view, chamada ActiveCustomers, seleciona apenas os clientes ativos da tabela Customers, mostrando as colunas customer_id e customer_name. Isso facilita a consulta de clientes ativos sem precisar reescrever a cláusula WHERE repetidamente.

### 3.2.2. VIEW DE AGREGAÇÃO

Permite cálculos como SUM, AVG e COUNT.

```
CREATE VIEW SalesReport AS
SELECT sale_id, sale_date, customer_name, total_amount
FROM Sales
JOIN Customers ON Sales.customer_id = Customers.customer_id;
```

A view SalesSummary cria um resumo das vendas por departamento. Utiliza a função SUM para calcular o total de vendas por departamento, agrupando os resultados por department. Isso ajuda a obter uma visão rápida do desempenho de vendas em cada departamento.

### 3.2.3. VIEW DE JUNÇÃO

Combina dados de múltiplas tabelas para apresentar uma visão integrada.

```
CREATE VIEW EmployeeDetails AS
SELECT e.employee_id, e.employee_name, d.department_name
FROM Employees e
JOIN Departments d ON e.department_id = d.department_id;
```

A view EmployeeDetails combina dados das tabelas Employees e Departments. Ela mostra o employee_id e employee_name da tabela Employees e o department_name da tabela Departments, facilitando a consulta de detalhes dos empregados junto com os nomes dos departamentos.

### 3.2.4. EXEMPLO DE CRIAÇÃO DE UMA VIEW COMPLEXA

```
CREATE VIEW DetailedSalesReport AS
SELECT s.sale_id, s.sale_date, c.customer_name, p.product_name, s.quantity,
FROM Sales s
JOIN Customers c ON s.customer_id = c.customer_id
JOIN Products p ON s.product_id = p.product_id;
```

A view DetailedSalesReport combina dados de três tabelas: Sales, Customers e Products. Ela mostra sale_id, sale_date da tabela Sales, customer_name da tabela Customers, e product_name da tabela Products, além de quantity e total_amount da tabela Sales. As junções são feitas com base nos IDs dos clientes (customer_id) e dos produtos (product_id). Essa view é útil para obter um relatório detalhado das vendas, incluindo informações sobre clientes e produtos.

# 4. VIEWS ATUALIZÁVEIS E NÃO ATUALIZÁVEIS

## 4.1. EXPLICAÇÃO SOBRE A POSSIBILIDADE DE ATUALIZAR DADOS DIRETAMENTE EM VIEWS

Algumas views permitem que os dados sejam atualizados diretamente, desde que certas condições sejam atendidas, como a ausência de junções complexas.

## 4.2. CONDIÇÕES PARA UMA VIEW SER ATUALIZÁVEL

Uma view é atualizável se:

- Não contiver funções agregadas.
- Não tiver junções complexas.
- For baseada em uma única tabela com chave primária.

```
CREATE VIEW SimpleEmployeeView AS
SELECT employee_id, employee_name
FROM Employees;
```

Esta view, SimpleEmployeeView, é baseada na tabela Employees e mostra employee_id e employee_name. Como é simples e baseada em uma única tabela, permite atualizações diretas.

## 4.3. EXEMPLO DE VIEW NÃO ATUALIZÁVEL

```
CREATE VIEW NonUpdatableView AS
SELECT e.employee_id, e.employee_name, d.department_name
FROM Employees e
JOIN Departments d ON e.department_id = d.department_id;
```

A view NonUpdatableView combina dados das tabelas Employees e Departments. Devido à junção entre múltiplas tabelas, esta view não permite atualizações diretas nos dados. A view NonUpdatableView combina dados das tabelas Employees e Departments. Devido à junção entre múltiplas tabelas, esta view não permite atualizações diretas nos dados.

# 5. ESTUDO DE CASO

## 5.1. CRIAÇÃO DE UM BANCO DE DADOS FICTÍCIO (LOJA DE E-COMMERCE)

Desenvolve-se um banco de dados para uma loja online, com tabelas como clientes, produtos, vendas e estoque.

## 5.2. EXEMPLO DE VIEW

### 5.2.1. VIEW DE RELATÓRIO DE VENDAS

```
CREATE VIEW SalesReport AS
SELECT sale_id, sale_date, customer_name, total_amount
FROM Sales
JOIN Customers ON Sales.customer_id = Customers.customer_id;
```

A view SalesReport combina dados das tabelas Sales e Customers. Ela mostra sale_id, sale_date, customer_name e total_amount, facilitando a geração de relatórios de vendas.A view SalesReport combina dados das tabelas Sales e Customers. Ela mostra sale_id, sale_date, customer_name e total_amount, facilitando a geração de relatórios de vendas.

## 5.3. DISCUSSÃO SOBRE AS VANTAGENS DAS VIEWS

No contexto da loja de e-commerce, as views facilitam a consulta e análise de dados, oferecendo relatórios e consultas filtradas, e garantem segurança ao permitir acesso seletivo.

# CONCLUSÃO

As SQL Views são ferramentas fundamentais para otimizar e organizar o acesso a dados em bancos de dados relacionais. Ao encapsular consultas complexas, elas simplificam o acesso e aumentam a segurança da informação, ao mesmo tempo em que facilitam a manutenção de consultas complexas e o controle de acesso.

No entanto, como qualquer tecnologia, apresentam limitações, como restrições em operações de atualização e possíveis impactos no desempenho em consultas complexas.

Em projetos de banco de dados, é recomendável o uso cuidadoso e estratégico das views, priorizando o equilíbrio entre segurança, praticidade e desempenho. Dessa forma, é possível maximizar os benefícios dessa tecnologia para o gerenciamento eficaz de dados.

# BIBLIOGRAFIA


- ELMASRI, R.; NAVATHE, S. B. Sistemas de Banco de Dados. 6ª ed. Pearson, 2015.
- DATE, C. J. Introdução a Sistemas de Bancos de Dados. 8ª ed. Campus, 2004.
- SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. Sistema de Banco de Dados. 6ª ed. McGraw-Hill, 2011.
- Artigo: "The Power of SQL Views" Oracle Documentation
- Artigo: "Using Views to Simplify Complex Queries" IBM Knowledge Center