



SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Curso

**TÉCNICO EM DESENVOLVIMENTO
DE SISTEMAS**

**SQL Views - Conceito, Benefícios e
Aplicações Práticas**

Murilo Moreno Figuerôa Vieira

**Sorocaba
Novembro – 2024**



SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Murilo Moreno Figuerôa Vieira

SQL Views - Conceito, Benefícios e Aplicações Práticas

Trabalho documentado sobre as
sql views na disciplina de banco
de dados

Prof. – Emerson Magalhães

Sorocaba
Novembro – 2024

SUMÁRIO

INTRODUÇÃO	4
Definição	4
Importância	4
Simplificação de Consultas.....	4
Segurança.....	4
Manutenção e Flexibilidade	5
Objetivo e Abrangência.....	5
1. FUNDAMENTOS TEÓRICOS DA SQL VIEWS.....	5
1.1. Definição e Funcionamento das Views	5
1.2. Diferença entre Views e Tabelas.....	5
1.3. Tipos de Views.....	6
1.3.1. Simples	6
1.3.2. Complexas.....	6
1.3.3. Materializadas.....	6
2. VANTAGENS E DESVANTAGENS DE USAR VIEWS.....	6
2.1. Vantagens.....	6
2.1.1. Simplificação de Consultas Complexas	6
2.1.2. Segurança e Controle de Acesso	6
2.1.3. Facilidade de Manutenção.....	7
2.2. Desvantagens.....	7
2.2.1. Impacto no Desempenho.....	7
2.2.2. Limitações de Atualização	7
2.2.3. Manutenção de Views Materializadas	7
3. PROCESSO DE CRIAÇÃO DE VIEWS NO SQL.....	7
3.1. Sintaxe básica em SQL	7
3.2. Exemplos práticos de views	8
3.2.1. View de filtragem	8
3.2.2. View de agregação.....	8
3.2.3. View de Junção.....	8
3.2.4. Exemplo de View Complexa.....	9
4. VIEWS ATUALIZÁVEIS E NÃO ATUALIZÁVEIS	9
4.1. Condições para Atualização	9
4.2. Exemplo Prático	10
4.2.1. View Atualizável	10
4.2.2. View Não Atualizável.....	10
5. ESTUDO DE CASO	10
CONCLUSÃO.....	14
Resumo	14
Considerações Finais.....	14
Sugestões de boas práticas.....	15
BIBLIOGRAFIA	15

SQL Views - Conceito, Benefícios e Aplicações Práticas

INTRODUÇÃO

Definição

SQL Views são consultas pré-definidas, salvas no banco de dados, que fornecem uma "visão" específica de dados contidos em uma ou mais tabelas. Quando criamos uma view, estamos criando uma camada de abstração sobre as tabelas base, que pode exibir uma seleção filtrada e, às vezes, resumida dos dados. Essa camada de abstração facilita consultas complexas e permite o uso de dados de forma simplificada, pois o banco de dados executa a consulta original toda vez que a view é acessada.

Importância

Views desempenham um papel importante em sistemas de bancos de dados, pois elas permitem:

Simplificação de Consultas

Permitem simplificar consultas complexas, encapsulando lógica de consulta que pode envolver múltiplas tabelas e operações de agregação. Uma vez criadas, as views podem ser reutilizadas, economizando tempo e garantindo consistência nas consultas.

Segurança

Uma view pode limitar o acesso a certas colunas e linhas, garantindo que os usuários tenham acesso somente aos dados necessários. Dessa forma, é possível ocultar informações confidenciais ou sensíveis, restringindo o acesso apenas ao necessário.

Manutenção e Flexibilidade

A view centraliza a lógica de consulta. Caso o esquema do banco de dados ou a lógica de negócio mude, basta atualizar a view, ao invés de modificar cada consulta individualmente.

Objetivo e Abrangência

A pesquisa cobre desde a criação básica de tabelas e views até exemplos avançados, incluindo as melhores práticas de uso e manutenção de views em um sistema de banco de dados SQL.

1. FUNDAMENTOS TEÓRICOS DA SQL VIEWS

1.1. Definição e Funcionamento das Views

Uma view funciona como uma "janela" para visualizar dados sem armazená-los fisicamente (exceto em views materializadas). Cada vez que uma view é acessada, o banco de dados executa a consulta subjacente, o que permite que os dados exibidos estejam sempre atualizados. Em sistemas SQL, views são muito úteis para extrair dados específicos de forma ordenada, economizando tempo e esforço em consultas repetitivas e complexas.

1.2. Diferença entre Views e Tabelas

Uma **tabela** é uma estrutura de armazenamento de dados onde as informações são mantidas de forma permanente, exceto se forem excluídas explicitamente.

Uma **view** não armazena dados; ela contém uma consulta SQL que é executada cada vez que a view é chamada. Assim, uma view é basicamente uma consulta nomeada que pode ser reutilizada como uma tabela para simplificar o trabalho com dados.

1.3. Tipos de Views

1.3.1. Simples

Estas views incluem dados de uma única tabela. São usadas para simplificar o acesso a uma tabela específica, limitando colunas ou linhas e evitando a exposição de dados sensíveis.

1.3.2. Complexas

Estas views envolvem junções e agregações, permitindo consolidar informações de várias tabelas e realizar operações como soma, média ou contagem de dados.

1.3.3. Materializadas

Estas views armazenam os dados fisicamente, em vez de executarem uma consulta toda vez que são acessadas. As views materializadas são úteis quando é necessário acessar frequentemente uma view complexa, e onde o desempenho é mais crítico. No entanto, as views materializadas requerem manutenção para manter os dados sincronizados com as tabelas originais.

2. VANTAGENS E DESVANTAGENS DE USAR VIEWS

2.1. Vantagens

2.1.1. Simplificação de Consultas Complexas

As views permitem que consultas complicadas, envolvendo várias tabelas e operações, sejam encapsuladas e utilizadas como tabelas simples. Isso facilita a leitura, o entendimento e a reutilização dessas consultas.

2.1.2. Segurança e Controle de Acesso

As views são uma ótima ferramenta para ocultar informações confidenciais, permitindo o acesso a apenas uma seleção dos dados da tabela base. Isso ajuda a limitar o que o usuário pode ver e manipular.

2.1.3. Facilidade de Manutenção

Views podem centralizar a lógica de consultas complexas que precisam ser usadas em várias partes do sistema. Com isso, ao atualizar uma view, todas as consultas que a utilizam são automaticamente atualizadas.

2.2. Desvantagens

2.2.1. Impacto no Desempenho

Em views complexas, o banco de dados precisa realizar consultas intensas a cada acesso, o que pode prejudicar o desempenho, especialmente se as views forem complexas ou envolverem múltiplas junções e agregações.

2.2.2. Limitações de Atualização

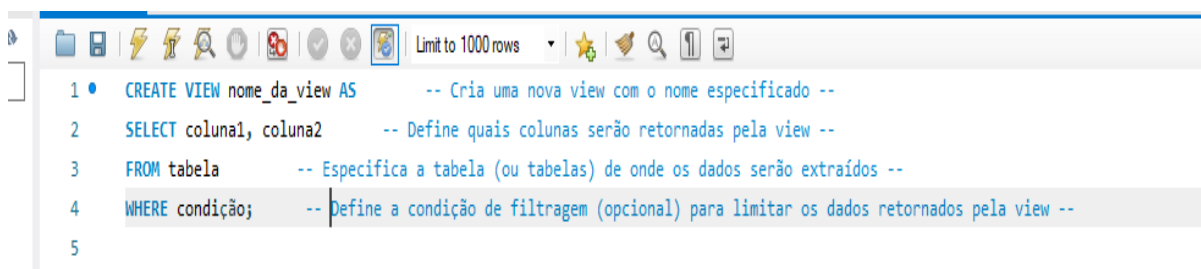
Algumas views não suportam operações de atualização, inserção ou exclusão, especialmente quando envolvem agregações ou junções complexas. Isso limita a interatividade com os dados.

2.2.3. Manutenção de Views Materializadas

Em views materializadas, os dados não são atualizados automaticamente. É necessário atualizar a view materializada para que os dados reflitam as mudanças nas tabelas de origem, o que requer uma política de atualização bem definida.

3. PROCESSO DE CRIAÇÃO DE VIEWS NO SQL

3.1. Sintaxe básica em SQL

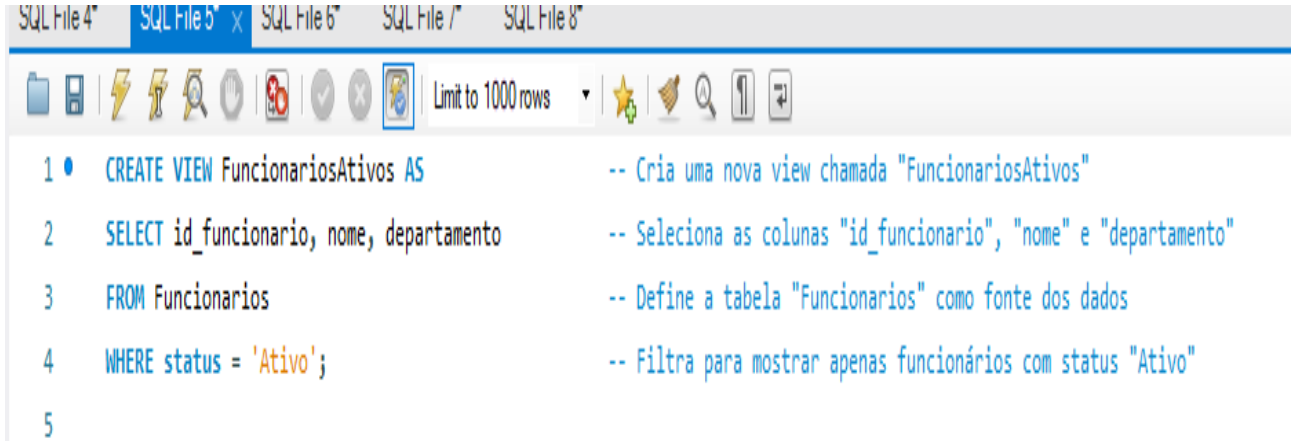
A screenshot of a SQL editor interface. The top toolbar includes icons for file operations, execution, and search. Below the toolbar, a text area contains SQL code for creating a view. The code is as follows:

```
1 CREATE VIEW nome_da_view AS      -- Cria uma nova view com o nome especificado --
2 SELECT coluna1, coluna2          -- Define quais colunas serão retornadas pela view --
3 FROM tabela                      -- Especifica a tabela (ou tabelas) de onde os dados serão extraídos --
4 WHERE condição;                  -- Define a condição de filtragem (opcional) para limitar os dados retornados pela view --
5
```

3.2. Exemplos práticos de views

3.2.1. View de filtragem

Seleção de colunas e linhas específicas de uma tabela.

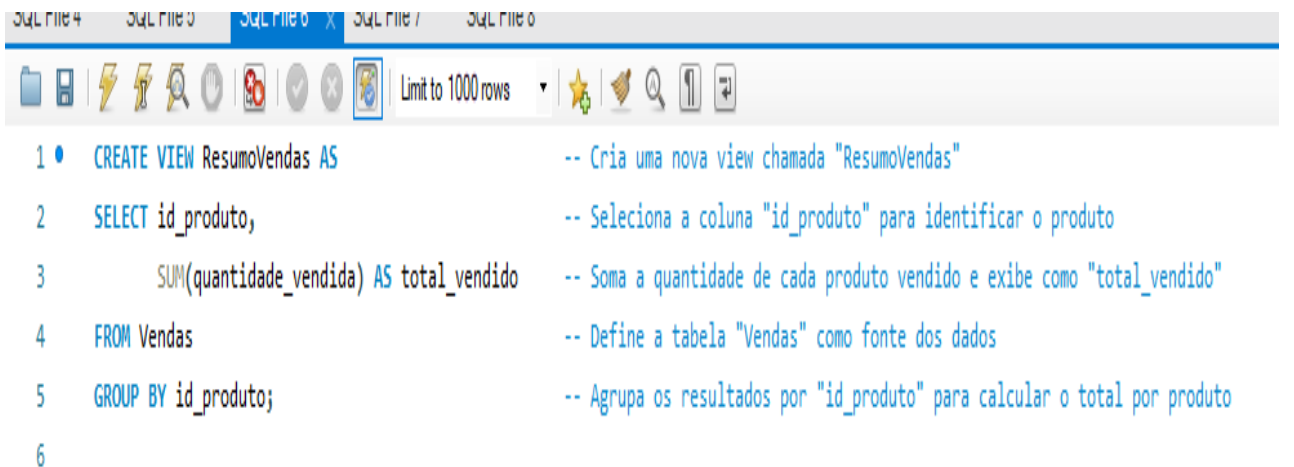


The screenshot shows a SQL editor with a toolbar at the top containing icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The SQL code is as follows:

```
1 • CREATE VIEW FuncionariosAtivos AS -- Cria uma nova view chamada "FuncionariosAtivos"
2   SELECT id_funcionario, nome, departamento -- Seleciona as colunas "id_funcionario", "nome" e "departamento"
3   FROM Funcionarios -- Define a tabela "Funcionarios" como fonte dos dados
4   WHERE status = 'Ativo'; -- Filtra para mostrar apenas funcionários com status "Ativo"
5
```

3.2.2. View de agregação

Exemplo com funções de agregação para sumarizar dados.



The screenshot shows a SQL editor with a toolbar at the top containing icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The SQL code is as follows:

```
1 • CREATE VIEW ResumoVendas AS -- Cria uma nova view chamada "ResumoVendas"
2   SELECT id_produto, -- Seleciona a coluna "id_produto" para identificar o produto
3          SUM(quantidade_vendida) AS total_vendido -- Soma a quantidade de cada produto vendido e exibe como "total_vendido"
4   FROM Vendas -- Define a tabela "Vendas" como fonte dos dados
5   GROUP BY id_produto; -- Agrupa os resultados por "id_produto" para calcular o total por produto
6
```

3.2.3. View de Junção

Combina dados de várias tabelas.


```
1 • CREATE VIEW FuncionarioDepartamento AS -- Cria uma nova view chamada "FuncionarioDepartamento"
2 SELECT f.id_funcionario, -- Seleciona a coluna "id_funcionario" da tabela "Funcionarios"
3        f.nome, -- Seleciona a coluna "nome" da tabela "Funcionarios"
4        d.nome_departamento -- Seleciona a coluna "nome_departamento" da tabela "Departamentos"
5 FROM Funcionarios f -- Define a tabela "Funcionarios" com um alias "f"
6 JOIN Departamentos d ON f.id_departamento = d.id_departamento; -- Realiza uma junção com "Departamentos" onde os IDs de departamento coincidem
-
```

3.2.4. Exemplo de View Complexa

```
1 • CREATE VIEW PedidosAltaValor AS -- Cria uma nova view chamada "PedidosAltaValor"
2 SELECT p.id_pedido, -- Seleciona a coluna "id_pedido" da tabela "Pedidos"
3        p.data_pedido, -- Seleciona a coluna "data_pedido" da tabela "Pedidos"
4        c.nome_cliente, -- Seleciona a coluna "nome_cliente" da tabela "Clientes"
5        SUM(i.quantidade * i.preco_unitario) AS valor_total -- Calcula o valor total do pedido somando (quantidade * preço unitário) de cada item
6 FROM Pedidos p -- Define a tabela "Pedidos" com o alias "p"
7 JOIN Clientes c ON p.id_cliente = c.id_cliente -- Realiza uma junção com "Clientes" onde os IDs de cliente coincidem
8 JOIN ItensPedido i ON p.id_pedido = i.id_pedido -- Realiza uma junção com "ItensPedido" onde os IDs de pedido coincidem
9 GROUP BY p.id_pedido, p.data_pedido, c.nome_cliente -- Agrupa os resultados por pedido, data e nome do cliente
10 HAVING valor_total > 5000; -- Filtra para mostrar apenas pedidos com valor total acima de 5000
11
```

4. VIEWS ATUALIZÁVEIS E NÃO ATUALIZÁVEIS

Algumas views permitem operações INSERT, UPDATE e DELETE, enquanto outras são apenas de leitura.

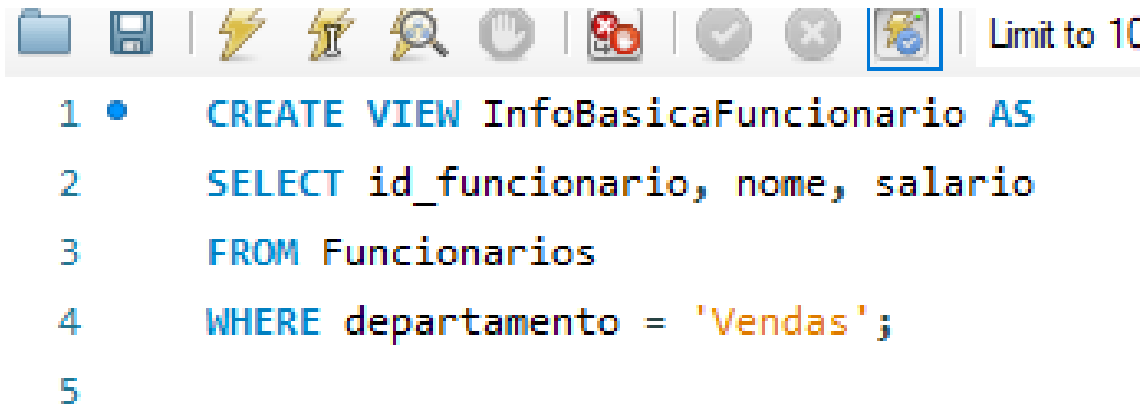
4.1. Condições para Atualização

A view deve referenciar uma única tabela;

Não deve conter funções de agregação nem a cláusula DISTINCT.

4.2. Exemplo Prático

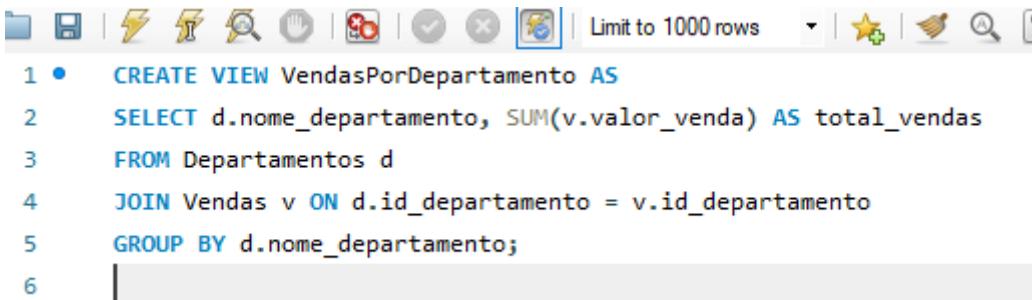
4.2.1.View Atualizável



```
1 • CREATE VIEW InfoBasicaFuncionario AS
2   SELECT id_funcionario, nome, salario
3   FROM Funcionarios
4   WHERE departamento = 'Vendas';
5
```

Esta view permite atualização se os dados modificados atenderem aos critérios do WHERE.

4.2.2.View Não Atualizável



```
1 • CREATE VIEW VendasPorDepartamento AS
2   SELECT d.nome_departamento, SUM(v.valor_venda) AS total_vendas
3   FROM Departamentos d
4   JOIN Vendas v ON d.id_departamento = v.id_departamento
5   GROUP BY d.nome_departamento;
6
```

Esta view não permite atualizações, pois utiliza uma função de agregação (SUM).

5. ESTUDO DE CASO

```
CREATE DATABASE LojaOnline; -- Cria a base de dados chamada "LojaOnline"
```

```
USE LojaOnline; -- Define "LojaOnline" como a base de dados ativa para futuras operações
```

```
CREATE TABLE Clientes (
    id_cliente INT PRIMARY KEY, -- Identificador único do cliente
    nome VARCHAR(100) NOT NULL, -- Nome do cliente (obrigatório)
```

```

    email VARCHAR(100) NOT NULL UNIQUE,      -- E-mail do cliente, único e
obrigatório
    telefone VARCHAR(15),                    -- Telefone do cliente (opcional)
    data_registro DATE                        -- Data de registro do cliente, será
preenchida manualmente
);

```

```

CREATE TABLE Produtos (
    id_produto INT PRIMARY KEY,              -- Identificador único do produto
    nome_produto VARCHAR(100) NOT NULL,      -- Nome do produto
(obrigatório)
    preco DECIMAL(10, 2) NOT NULL,          -- Preço do produto com duas
casas decimais
    quantidade_estoque INT DEFAULT 0        -- Quantidade em estoque, com
valor padrão 0
);

```

```

CREATE TABLE Pedidos (
    id_pedido INT PRIMARY KEY,               -- Identificador único do pedido
    id_cliente INT,                         -- Referência ao cliente que fez o pedido
    data_pedido DATE,                      -- Data do pedido
    status VARCHAR(20),                    -- Status do pedido (Ex: 'Concluído',
'Pendente')
    valor_total DECIMAL(10, 2),            -- Valor total do pedido
    FOREIGN KEY (id_cliente) REFERENCES Clientes(id_cliente) -- Chave
estrangeira para "Clientes"
);

```

```

CREATE TABLE ItensPedido (
    id_item INT PRIMARY KEY,                -- Identificador único do item do
pedido

```

```

    id_pedido INT,                -- Identificador do pedido (chave
    estrangeira para "Pedidos")
    id_produto INT,              -- Identificador do produto (chave
    estrangeira para "Produtos")
    quantidade INT NOT NULL,     -- Quantidade do produto no pedido
    preco_unitario DECIMAL(10, 2), -- Preço unitário do produto no
    pedido
    FOREIGN KEY (id_pedido) REFERENCES Pedidos(id_pedido), -- Chave
    estrangeira para "Pedidos"
    FOREIGN KEY (id_produto) REFERENCES Produtos(id_produto) -- Chave
    estrangeira para "Produtos"
);

```

```

CREATE TABLE Vendas (
    id_venda INT PRIMARY KEY,      -- Identificador único da venda
    id_produto INT,              -- Identificador do produto (chave
    estrangeira para "Produtos")
    quantidade_vendida INT NOT NULL, -- Quantidade do produto
    vendido
    data_venda DATE,             -- Data da venda
    valor_venda DECIMAL(10, 2),   -- Valor total da venda
    FOREIGN KEY (id_produto) REFERENCES Produtos(id_produto) -- Chave
    estrangeira para "Produtos"
);

```

```

CREATE VIEW RelatorioVendasMensais AS
SELECT EXTRACT(MONTH FROM data_venda) AS mes, -- Extrai o mês
da data de venda

```

```

        SUM(valor_venda) AS total_vendas          -- Calcula o valor total das
vendas no mês
FROM Vendas
GROUP BY EXTRACT(MONTH FROM data_venda);          -- Agrupa os
resultados por mês

```

```

CREATE VIEW ProdutosBaixoEstoque AS
SELECT id_produto,                                -- Identificador do produto
       nome_produto,                              -- Nome do produto
       quantidade_estoque                         -- Quantidade disponível em estoque
FROM Produtos
WHERE quantidade_estoque < 10;                    -- Filtra produtos com estoque
menor que 10 unidades

```

```

CREATE VIEW ClientesAltoGasto AS
SELECT c.id_cliente,                              -- Identificador do cliente
       c.nome,                                    -- Nome do cliente
       SUM(p.valor_total) AS gasto_total          -- Soma o valor dos pedidos do
cliente
FROM Clientes c
JOIN Pedidos p ON c.id_cliente = p.id_cliente    -- Realiza uma junção com
"Pedidos" para obter os valores
WHERE p.status = 'Concluído'                     -- Considera apenas pedidos
concluídos
GROUP BY c.id_cliente, c.nome                    -- Agrupa por cliente
HAVING SUM(p.valor_total) > 10000;                -- Exibe clientes com gasto
acima de 10.000

```

```

CREATE VIEW DetalhesPedidoClienteProduto AS
SELECT p.id_pedido,                -- Identificador do pedido
       c.nome AS nome_cliente,    -- Nome do cliente
       pr.nome_produto,           -- Nome do produto
       i.quantidade,              -- Quantidade do produto no pedido
       i.preco_unitario,          -- Preço unitário do produto no pedido
       i.quantidade * i.preco_unitario AS valor_total_item -- Calcula o valor total
do item
FROM Pedidos p
JOIN Clientes c ON p.id_cliente = c.id_cliente -- Junção com "Clientes" para
obter o nome do cliente
JOIN ItensPedido i ON p.id_pedido = i.id_pedido -- Junção com
"ItensPedido" para obter os itens do pedido
JOIN Produtos pr ON i.id_produto = pr.id_produto; -- Junção com "Produtos"
para obter o nome e preço do produto

```

CONCLUSÃO

Resumo

SQL Views são poderosas ferramentas para simplificar consultas complexas, melhorar a segurança de dados e promover boas práticas de modularidade em bancos de dados relacionais. Elas permitem criar abstrações sobre as tabelas subjacentes, facilitando a manutenção e a reutilização de consultas comuns, além de isolar a lógica de negócios do acesso direto aos dados.

Considerações Finais

As views são altamente recomendadas para projetos que buscam centralizar lógicas de consulta e melhorar a organização e segurança de um sistema de banco de dados. Além disso, elas são úteis em cenários de acesso controlado, onde diferentes usuários ou sistemas têm visões limitadas dos dados. No entanto, é importante avaliar o impacto de desempenho, especialmente em sistemas de alta carga, uma vez que views complexas podem afetar a eficiência do banco de dados, caso não sejam projetadas adequadamente.

Sugestões de boas práticas

Utilize views para consultas repetitivas e comuns:

Quando várias partes de um sistema necessitam da mesma lógica de consulta, as views podem centralizar essa lógica, evitando duplicação e facilitando a manutenção.

Implemente views para melhorar a segurança de dados:

Views podem ser usadas para restringir o acesso a colunas ou linhas específicas de uma tabela, permitindo que diferentes níveis de usuários vejam apenas os dados que têm permissão para acessar.

Evite views excessivamente complexas em ambientes de alta demanda:

Embora views sejam convenientes, o uso de várias camadas de views ou views com consultas complexas (como joins ou subconsultas aninhadas) pode degradar o desempenho, especialmente em cenários com grande volume de dados ou alta concorrência.

BIBLIOGRAFIA

MySQL: TRABALHO COM VIEWS. In: **DevMedia.** Disponível em:<
<https://www.devmedia.com.br/mysql-trabalhando-com-views/8724>>. Acesso em:
12 nov. 2024.

MySQL Views. In: **W3Schools.** Disponível em:<
https://www.w3schools.com/mysql/mysql_view.asp >. Acesso em: 12 nov. 2024.

Trabalho com views no MySQL. In: **KingHost.** Disponível em:<
<https://king.host/wiki/artigo/views-mysql/>>. Acesso em: 12 nov. 2024.

MySQL View. In: **javatpoint.** Disponível em:<
<https://www.javatpoint.com/mysql-view>>. Acesso em: 12 nov. 2024.