SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Curso

TÉCNICO EM DESENVOLVIMENTO

DE SISTEMAS

SQL Views

Ana Julia Sanches de Souza

Sorocaba
Novembro – 2024

SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Ana Julia Sanches de Souza

SQL Views

Aprofundamento a respeito dos
Conceitos, benefícios e
aplicações práticas.
Prof. – Emerson

Sorocaba
Novembro – 2024

INTRODUÇÃO

SQL Views nada mais são do que consultas SQL que foram armazenadas no sistema de banco de dados relacionais. O sistema em questão funciona como uma tabela virtual, que permite uma representação simples de determinados dados, bem como a transformação dos mesmos de uma ou mais tabelas sem que haja a necessidade de duplica-los.

A função das views é justamente realizar o armazenamento e representar os dados sem que haja a necessidade de que eles sejam armazenados em uma tabela física, o que permite assim, que o manuseamento dos dados ocorra de forma mais simples e eficaz.

A importância das views em sistemas de banco de dados relacionais está relacionada com diversos benefícios como a otimização do tempo, já que não há necessidade de escrever diversas instruções (é possível escrever uma vez e armazenar), é possível acessar os dados mais rapidamente além de simplificar e permitir maior organização por parte do usuário.

Com isso, a pesquisa visa esclarecer o que são as SQL Views, sua importância e como elas podem ser utilizadas para possibilitar tais benefícios como a melhor manipulação de dados em banco de dados relacionais. Além disso, será explorado o processo de criação de views e exemplos práticos que ilustram suas aplicações no dia a dia.

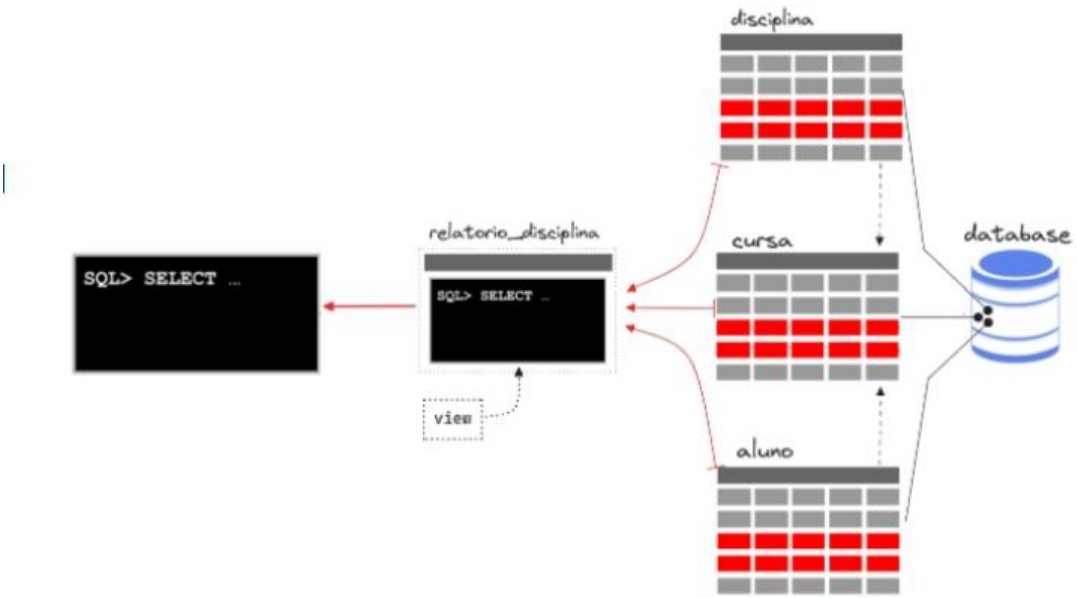
1. Fundamentos Teóricos das SQL Views

1.1. O que são views e como elas funcionam no SQL.

As views (visões), são consultas armazenadas no banco de dados em questão e é popularmente chamada de tabela virtual, já que essas consultas não são armazenadas de forma física no banco de dados, mas sim, a cada execução da view que é realizado uma consulta, esses dados são acessados diretamente em sua tabela de origem. Dessa forma é possível que os dados sejam consultados de maneira simples e eficaz, uma vez que fornece a exibição das informações de forma personalizada e permite a limitação do acesso aos dados presentes nas tabelas reais.

A view é formada pela declaração “SELECT’s” do usuário, o que acaba por retornar uma “visão” específica dos dados dispostos em uma ou mais tabelas do banco de dados.

Exemplo de visão com base nas tabelas "aluno", "cursa" e "disciplina":



.No exemplo acima a tabela virtual (view) está sendo criada a partir de uma consulta, combinando os dados de tabelas diferentes, permitindo assim, que o

usuário veja apenas os dados específicos que deseja consultar, de forma que os dados da tabela original não estão sendo alterados.
A view da imagem acima foi criada para que determinados dados específicos das tabelas “aluno”, “curso” e “disciplina”, fossem dispostos em conjunto de forma simples e que não houvesse a necessidade de serem armazenados de forma física.

1.2. Diferença entre views e tabelas comuns.

Uma tabela comum é um objeto formado por linhas e colunas que armazenam dados que podem ser consultados sempre que quisermos.
Ao utilizar o “Select*from” estamos visualizando a tabela, mas ao continuar realizando outros movimentos no SQL acabamos por perder de vista essa exibição. Deste modo, é necessário uma maneira de armazenar essa consulta que foi realizada, e é ai que utilizamos a view.
Portanto, uma tabela é um objeto que armazena os dados dentro de um banco de dados, enquanto a view é um objeto que permite armazenar e gerar um conjunto de dados a partir da consulta de outras tabelas.
Entre outras diferenças apresentadas, existe o fato de que a tabela é independente e já a view é dependente de uma ou mais tabelas, na tabela é possível adicionar, atualizar e excluir os dados, já a view não apresenta essas possibilidades. Por fim, uma tabela armazena dados e a view armazena uma “quary” (consulta).

1.3. Tipos de Views

Views simples: A views simples é aquela que baseia-se em apenas uma tabela e não apresenta junções ou cálculos complexos, ou seja, ela realiza apenas as funções básicas. Vale ressaltar que ela também não armazena dados fisicamente.

```
CREATE VIEW V_Produtos_Disponiveis AS
SELECT ProdutoID, Nome, Preço
FROM Produtos
WHERE Disponível = 1;
```

No exemplo acima a view “V_Produtos_Disponiveis” exibe apenas os produtos disponíveis na tabela produtos.

Views complexas (com junções e agregações): As views complexas envolvem diversas tabelas e podem conter junções, agregações e subconsultas, o que torna a visualização mais rica em detalhes. Esse tipo de view ´w recomendado para criar relatórios e análises mais complexas.

```
CREATE VIEW V_Vendas_Cliente AS
SELECT C.Nome, SUM(V.Valor) AS TotalVendas
FROM Clientes C
JOIN Vendas V ON C.ClienteID = V.ClienteID
GROUP BY C.Nome;
```

Nesse exemplo a view “V_Vendas_Cliente” exibe a combinação dos dados da das vendas e dos clientes para permitir a visualização do total de vendas realizadas pelo cliente.

Views materializadas (caso o banco de dados suporte): Quando a view está materializada, significa que houve a cópia dos dados da consulta de forma física. Com o armazenamento físico, há uma melhora significativa no desempenho de consultas mais complexas, além de otimizar o desempenho em leituras excessivas.

```
CREATE MATERIALIZED VIEW MV_Vendas_Mensais AS
SELECT MONTH(DataVenda) AS Mês, SUM(Valor) AS Total
FROM Vendas
GROUP BY MONTH(DataVenda);
```

Nesse exemplo a view “M_Vendas_Mensais” armazena o total de vendas realizadas por mês para consultas mais eficientes.

2. Vantagens e Desvantagens de usar view.

2.1. Vantagens

Simplificação de consultas: Quando você precisa realizar consultas complexas, como comparar várias tabelas, pode ser difícil reescrever a mesma consulta toda vez. A view resolve esse problema ao permitir que você defina uma consulta complexa uma única vez, e depois a reutilize sempre que necessário, apenas chamando o nome da view. Assim, a view facilita o uso de consultas repetitivas, evitando a necessidade de reescrevê-las e tornando o processo mais simples e eficiente.

Aumento da segurança: É possível limitar o acesso direto às tabelas, permitindo que os usuários acessem os dados apenas por meio de uma view. Com isso, você pode configurar a view para exibir apenas as informações relevantes para cada usuário. Esse controle reduz o risco de exposição de dados sensíveis, ajudando a proteger a privacidade das informações e oferecendo uma camada adicional de segurança.

Facilita o controle e a manutenção: O uso de views facilita a manutenção, pois, uma vez criada, ela pode ser utilizada em qualquer parte do banco de dados. Dessa forma, se for necessário alterar a lógica da consulta, basta atualizar a definição da view, em vez de modificar todas as consultas que a utilizam.

2.2. Desvantagens

Impactos de desempenho: O nível de complexidade de uma view pode fazer com que o banco de dados precise realizar todas as operações sempre que ela for consultada, o que pode consumir tempo e recursos do sistema, prejudicando a eficiência da execução.

Limitações: Views complexas que envolvem diversas tabelas podem apresentar problemas em suas atualizações, o que faz com que o processo ocorra de forma mais lenta.

Manutenção de views materializadas (atualização periódica dos dados): A atualização periódica dos dados exige um planejamento e gerenciamento cuidadosos para garantir que os dados sejam atualizados de maneira adequada e eficiente. Além disso, o processo de atualização constante consome recursos, o que pode impactar o desempenho do sistema e tornar as operações de leitura e escrita mais lentas.

3. Processo de Criação de Views no SQL

3.1. Create View: sintaxe e parâmetros.

Existem quatro “comandos” que são de extrema importância para a criação de views no SQL: O comando “CREATE VIEW” é utilizado para criar uma view no banco de dados. O comando “SELECT” é utilizado para selecionar as colunas desejadas, o “FROM” direciona a tabela onde os dados serão selecionados e o “WHERE” é uma condição para filtrar os dados.

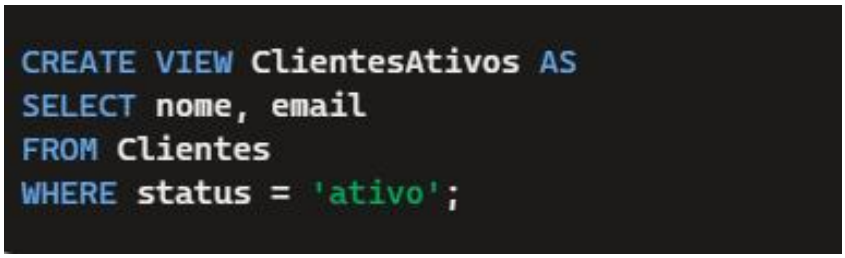
Exemplo:

```sql
-- creating a view
CREATE VIEW cust_england AS
SELECT customer_id, customer_name
FROM customers WHERE country="England";

-- retrieving data from the view
SELECT * FROM cust_england;
```

3.2. Views de filtragem: colunas e linhas específicas.

Exemplo:



```sql
CREATE VIEW ClientesAtivos AS
SELECT nome, email
FROM Clientes
WHERE status = 'ativo';
```

No exemplo acima a view possui o nome “ClientesAtivos”. O comando “Select’ indica que as colunas “nome” e “email” foram selecionas da tabela “Clientes” (“FROM Clientes”). Após isso a view filtra os clientes e seleciona apenas os com status “ativo”;

3.3. View de Agregação: uso de funções como SUM, AVG E COUNT.

```
CREATE VIEW vendas_agregadas AS
SELECT
    vendedor_id,
    COUNT(*) AS total_vendas,          -- Conta o número total de vendas
    SUM(valor_venda) AS total_valor,   -- Soma o valor total das vendas
    AVG(valor_venda) AS valor_medio    -- Calcula o valor médio das vendas
FROM
    vendas
GROUP BY
    vendedor_id;
```

O exemplo acima exemplifica o uso de tais comandos. O comando SUM é utilizado para calcular a soma total dos valores da coluna, o comando AVG que é utilizado para calcular a média dos valores da coluna, enquanto o comando COUNT é utilizado para contar o id dos pedidos.

3.4. View de Junção: Combinação de dados de diversas tabelas para gerar uma nova view.

```
CREATE VIEW PedidosPorCliente AS
SELECT c.nome, p.pedido_id, p.data_pedido, p.valor_pedido
FROM Clientes c
JOIN Pedidos p ON c.cliente_id = p.cliente_id;
```

No exemplo acima a view com o nome PedidosPorCliente é criada. Em seguida há a seleção das colunas c.nome, p.pedido_id, p.data_pedido e p.valor_pedido que pertencem a tabela Clientes (c).

O comando “JOIN” realiza a junção das tabelas Clientes e Pedidos com base campo cliente_id e o ON específica a condição que relaciona as tabelas.

4. VIEWS: Atualizáveis e não atualizáveis.

Views atualizáveis podem receber declarações de atualização, como UPDATE e DELETE, que irão alterar as tabelas base, entretanto essas não podem ser views de agregação, por exemplo. Isto porque views atualizáveis são obrigatoriamente baseadas em uma única tabela, para que sua relação entre as colunas da view e das colunas da tabela base seja uma relação direta.

Exemplo:

```
CREATE VIEW V_Funcionarios AS
SELECT id, nome, salario
FROM Funcionarios;
```

Esta é uma view simples sem agregações.

```
UPDATE V_Funcionarios
SET salario = 5000
WHERE id = 1;
```

Este é um UPDATE que pode ser feito nela.

```
CREATE VIEW V_Agregada AS
SELECT departamento, AVG(salario) AS salario_medio
FROM Funcionarios
GROUP BY departamento;
```

Já esta é uma view com agregações, ou seja, não atualizável.

5. Estudo de caso.

Banco de Dados do Sistema de RH.

```
CREATE DATABASE SistemaRH;
USE SistemaRH;

CREATE TABLE Cargos (
    cargo_id INT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(100) NOT NULL
);

CREATE TABLE Funcionarios (
    funcionario_id INT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(100) NOT NULL,
    sobrenome VARCHAR(100) NOT NULL,
    data_nascimento DATE,
    email VARCHAR(100) UNIQUE,
    telefone VARCHAR(20),
    cargo_id INT,
    FOREIGN KEY (cargo_id) REFERENCES Cargos(cargo_id)
);

CREATE TABLE Salarios (
    salario_id INT PRIMARY KEY AUTO_INCREMENT,
    funcionario_id INT,
    salario DECIMAL(10, 2) NOT NULL,
    data_inicio DATE NOT NULL,
    data_fim DATE,
    FOREIGN KEY (funcionario_id) REFERENCES Funcionarios(funcionario_id)
);

-- Inserir cargos
INSERT INTO Cargos (nome)
```

```
 VALUES
('Secretário'),
('Analista de RH'),
('Contador');

-- Inserir funcionários
INSERT INTO Funcionarios (nome, sobrenome, data_nascimento, telefone, cargo_id)
VALUES
('João', 'Silva', '1985-01-15', '1111-1111', 1),
('Maria', 'Oliveira', '1990-05-25', '2222-2222', 2),
('Ana', 'Souza', '1982-03-30', '3333-3333', 3);

-- Inserir salários
INSERT INTO Salarios (funcionario_id, salario, data_inicio)
VALUES
(1, 5000.00, '2023-01-01'),
(2, 4500.00, '2023-01-01'),
(3, 5500.00, '2023-01-01');

-- View da folha de pagamento
CREATE VIEW Detalhes_Funcionarios_Simples AS
SELECT
    f.nome,
    f.sobrenome,
    c.nome AS cargo,
    s.salario
FROM
    Funcionarios f
JOIN
    Cargos c ON f.cargo_id = c.cargo_id
JOIN
    Salarios s ON f.funcionario_id = s.funcionario_id
WHERE
```

```
    s.data_fim IS NULL;
```

Esta view não é atualizável, pois é uma view que agrega as tabelas Funcionários, Cargos e Salários e não há como estabelecer uma relação direta entre a colunas da view e as colunas de uma tabela base.
Como folha de pagamento, não é necessário que a view seja atualizável, pois consta mais como um relatório do que algum tipo de documento que precise ser atualizado. Portanto, uma folha de pagamento só tem necessidade de mudanças fundamentais, que podem ser realizadas diretamente nas tabelas.

CONCLUSÃO

Em conclusão, as SQL Views são utilizadas para otimizar o trabalho com bancos de dados relacionais, oferecendo uma maneira eficiente de representar dados de forma simples e sem a necessidade de duplicação física. Elas não apenas facilitam a consulta e o acesso a informações, mas também promovem a organização e o gerenciamento eficaz dos dados, permitindo maior agilidade e clareza no processo de manipulação das informações. Através da criação e utilização das views, os usuários podem reduzir a complexidade das operações e melhorar o desempenho geral do sistema, tornando-o mais eficiente e fácil de ser mantido. Assim, entender e aplicar as SQL Views se revela uma prática essencial para profissionais que buscam otimizar a gestão de dados em bancos de dados relacionais.

BIBLIOGRAFIA

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

https://www.devmedia.com.br/introducao-a-views/1614

https://king.host/wiki/artigo/views-mysql/#:~:text=O%20que%20%C3%A9%20uma%20view,de%20um%20banco%20de%20dados.

https://www.hashtagtreinamentos.com/diferenca-tabelas-e-views-no-sql

https://youtu.be/MOXHM8tdiYU?si=-L9MLd-wTXB5prWN

https://dev.to/mizevski/views-sequences-e-synonyms-um-tutorial-pratico-379d

https://comoprogramarjava.com.br/views-em-sql/

https://ramosdainformatica.com.br/views-em-sql-vantagens-e-desvantagens/

https://kb.elipse.com.br/linguagem-sql-capitulo-7-views/

https://pt.stackoverflow.com/questions/35413/o-que-s%C3%A3o-views-em-sql-quais-vantagens-e-desvantagens-em-utilizar