SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Curso

TÉCNICO EM DESENVOLVIMENTO

DE SISTEMAS

SQL views

Victoria de Mattos Ferreira

Sorocaba
Novembro – 2024

SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL

SENAI “GASPAR RICARDO JUNIOR”

Victoria de Mattos Ferreira

SQL views

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.
Prof. – Emerson

Sorocaba
Novembro – 2024

HISTÓRICO DE VERSÕES

Data	Versão	Descrição	Autor
11/11/2024	1.0	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.	Victoria de Mattos Ferreira

SUMÁRIO

INTRODUÇÃO

De acordo com Alves (2015), uma view é uma consulta armazenada no banco de dados, podendo ser consultada como se fosse uma tabela. Uma de suas principais funções é controlar a segurança do banco de dados, geralmente ela é criada com campos que determinado perfil de usuário pode acessar, concedendo acesso apenas a essa view e não a tabelas diretamente.

A view também é utilizada para apresentar informações mais organizadas para o usuário, ela já ficaria armazenada no próprio banco e o usuário não precisaria elabora uma consulta complexa.

Figura 1 – Exemplo de tabela em um banco de dados

FUNCIONARIO				
CPF	Nome	E-mail	Salário	ID_DEPARTAMENTO
123.456.789-10	João	joao@exemplo.com	R$ 4.500,00	1
111.222.333-44	Gustavo	gustavo@exemplo.com	R$ 2.500,00	1
222.333.444-55	Pedro	pedro@exemplo.com	R$ 3.500,00	2

DEPARTAMENTO	
ID_DEPARTAMENTO	NOME
1	Financeiro
2	Recursos Humanos

Considerando que um usuário precisa de uma lista atualizada dos

funcionários e seus departamentos. Em questões de segurança, não pode ser fornecido mais nenhumas informações como CPF, e-mail e salário dos funcionários.

A melhor forma é criar uma view onde essas informações não são apresentadas e fornece ao usuário acesso apenas a esta view, ou seja, o usuário só terá acesso ao nome e ao departamento.

V_FUNCIONARIO_DEPARTAMENTO	
FUNCIONARIO	DEPARTAMENTO
Gustavo	Financeiro
João	Financeiro
Pedro	Recursos Humanos

A consulta dessa view poderia ser: SELECT F.NOME AS FUNCIONARIO, D.NOME AS DEPERTAMENTO FROM FUNCIONARIO F INNER JOIN DEPARTAMENTO D ON D.ID_DEPARTAMENTO = F.ID_DEPARTAMENTO A view realiza uma consulta (query) em tempo de execução. Em uma view simples essa consulta que é armazenada. Essa consulta pode ter condições próprias para restringir os dados que serão visualizados pelos usuários, tanto horizontal (colunas que serão apresentadas) quanto vertical (linhas que serão apresentadas).

Query: é uma consulta SQL que acessa ou modifica os dados no banco de dados. Quando você cria uma view, você está armazenando uma query que será executada sempre que a view for acessada.

1. Fundamentos Teóricos das SQL Views

View: é uma consulta armazenada no banco de dados que pode ser acessada como se fosse uma tabela. A view pode ser usada para simplificar a consulta de dados e também para controlar a segurança, expondo apenas uma parte dos dados de acordo com a necessidade do usuário.

As tabelas armazenam dados reais no storage e podem ser consultadas e manipuladas usando SQL comando ou DataFrame APIs, suportando operações como insert, update, delete e merge. A view é uma tabela virtual definida por uma consulta SQL. O site view não armazena dados por si só.

1.1. Tipos de views

1. Views Simples

Uma view simples é aquela que contém uma única consulta (query) SQL sem realizar junções, agregações ou sub consultas complexas. Em outras palavras, ela simplesmente seleciona e exibe dados de uma ou mais tabelas, mas sem fazer manipulações complicadas nos dados.

```
CREATE VIEW v_funcionarios AS
SELECT NOME FROM FUNCIONARIOS;
```

2. Views Complexas (com Junções e Agregações)

As views complexas são aquelas que envolvem junções (joins) entre várias tabelas, agregações (como SUM(), AVG(), COUNT()) ou até subconsultas (queries dentro de queries). Essas views são usadas quando você precisa combinar dados de várias tabelas ou fazer cálculos para apresentar um conjunto de informações mais complexo.

```
CREATE VIEW v_funcionarios_departamento AS
SELECT D.NOME AS DEPARTAMENTO, COUNT(F.ID_FUNCIONARIO) AS NUM_FUNCIONARIOS
FROM DEPARTAMENTOS D
INNER JOIN FUNCIONARIOS F ON D.ID_DEPARTAMENTO = F.ID_DEPARTAMENTO
GROUP BY D.NOME;
```

3. Views Materializadas

Uma view materializada (ou materialized view) é um tipo especial de view que, ao contrário das views simples e complexas, armazena fisicamente os resultados da consulta em disco, ao invés de ser apenas uma consulta dinâmica executada toda vez que é acessada. Ou seja, ela cria uma cópia dos dados consultados e, sempre que você consultar a view, estará acessando esses dados armazenados, não sendo necessário executar a consulta SQL toda vez.

```
CREATE MATERIALIZED VIEW v_funcionarios_departamento AS
```

```
SELECT D.NOME AS DEPARTAMENTO, COUNT(F.ID_FUNCIONARIO)
AS NUM_FUNCIONARIOS
FROM DEPARTAMENTOS D
INNER JOIN FUNCIONARIOS F ON D.ID_DEPARTAMENTO =
F.ID_DEPARTAMENTO
GROUP BY D.NOME;
```

.

1.2. Vantagens e Desvantagens

As vantagens de se usar views são:

Economizar tempo com retrabalho;

Ex.: Você não precisar escrever aquela instrução enorme. Escreva uma vez e armazene!

Velocidade de acesso às informações;

Ex.: Uma vez compilada, o seu recordset (conjunto de dados) é armazenado em uma tabela temporária (virtual).

Mascarar complexidade do banco de dados;

Ex.: As views isolam do usuário a complexidade do banco de dados. Nomes de domínios podem ser referenciados com literais e outros recursos. Isso proporciona aos desenvolvedores a capacidade de alterar a estrutura sem afetar a interação do usuário com o banco de dados.

Simplifica o gerenciamento de permissão de usuários;

Ex.: Em vez de conceder permissão para que os usuários contem tabelas base, os proprietários de bancos de dados podem conceder permissões para que os usuários consultem dados somente através de views. Isso também protege as alterações na estrutura das tabelas base subjacentes. Os usuários não serão interrompidos durante uma visualização de dados.

Organizar dados a serem exportados para outros aplicativos;

Ex.: Você pode criar uma view baseada em uma consulta complexa, que associe até 32 tabelas e depois exportar dados para outro aplicativo para análise adicional. Pode ser gerado um arquivo de DUMP* automaticamente.

DESVANTAGENS:

- **Impactos de Desempenho**: As views podem causar problemas de desempenho, especialmente quando as consultas envolvem junções complexas ou agregações em grandes volumes de dados. O acesso a dados em tempo real pode ser mais lento.
- **Limitações para Atualizações**: Views complexas (com junções ou agregações) geralmente não são atualizáveis diretamente, o que pode exigir abordagens alternativas para operações de inserção, atualização e exclusão.
- **Manutenção de Views Materializadas**: As views materializadas requerem atualização periódica para refletir mudanças nas tabelas subjacentes. Isso pode consumir recursos de processamento e armazenamento, e o processo de atualização pode ser custoso.

2. Processo de Criação de Views no SQL

View simples:

```sql
CREATE VIEW v_funcionarios AS
SELECT NOME FROM FUNCIONARIOS;
```

View complexa:

```sql
CREATE VIEW v_funcionarios_departamento AS
SELECT D.NOME AS DEPARTAMENTO, COUNT(F.ID_FUNCIONARIO) AS NUM_FUNCIONARIOS
FROM DEPARTAMENTOS D
INNER JOIN FUNCIONARIOS F ON D.ID_DEPARTAMENTO = F.ID_DEPARTAMENTO
GROUP BY D.NOME;
```

View materializada:

```sql
CREATE MATERIALIZED VIEW v_funcionarios_departamento AS
SELECT D.NOME AS DEPARTAMENTO, COUNT(F.ID_FUNCIONARIO) AS NUM_FUNCIONARIOS
FROM DEPARTAMENTOS D
INNER JOIN FUNCIONARIOS F ON D.ID_DEPARTAMENTO = F.ID_DEPARTAMENTO
GROUP BY D.NOME;
```

3. Views atualizáveis e não atualizáveis

3.1. Atualizáveis

Uma view atualizável é uma view através da qual você pode modificar os dados nas tabelas subjacentes. Ou seja, você pode inserir, atualizar ou deletar registros diretamente na view, e essas mudanças refletirão nas tabelas de origem.

```sql
-- Tabela FUNCIONARIOS
CREATE TABLE FUNCIONARIOS (
    ID_FUNCIONARIO INT PRIMARY KEY,
    NOME VARCHAR(100),
    SALARIO DECIMAL(10, 2),
    DEPARTAMENTO_ID INT
);

-- View atualizável
CREATE VIEW v_funcionarios AS
SELECT ID_FUNCIONARIO, NOME, SALARIO
FROM FUNCIONARIOS;
```

3.2. Não atualizáveis

Uma view não atualizável é aquela onde você não pode realizar operações de modificação de dados (como INSERT, UPDATE, DELETE) diretamente através da view. Isso ocorre quando a estrutura da view é complexa demais para que o banco de dados entenda como refletir essas operações nas tabelas subjacentes.

```sql
CREATE TABLE FUNCIONARIOS (
    ID_FUNCIONARIO INT PRIMARY KEY,
    NOME VARCHAR(100),
    SALARIO DECIMAL(10, 2),
    DEPARTAMENTO_ID INT
);

-- Tabela DEPARTAMENTOS
CREATE TABLE DEPARTAMENTOS (
    ID_DEPARTAMENTO INT PRIMARY KEY,
    NOME VARCHAR(100)
);

-- View não atualizável com junção e agregação
CREATE VIEW v_departamento_salario AS
SELECT D.NOME AS DEPARTAMENTO, SUM(F.SALARIO) AS SALARIO_TOTAL
FROM DEPARTAMENTOS D
JOIN FUNCIONARIOS F ON F.DEPARTAMENTO_ID = D.ID_DEPARTAMENTO
GROUP BY D.NOME;
```

Estudo de caso

```sql
-- Tabela Clientes
CREATE TABLE clientes (
    cliente_id INT PRIMARY KEY,
    nome VARCHAR(100),
    email VARCHAR(100),
    telefone VARCHAR(20),
    endereco VARCHAR(200)
);

-- Tabela Produtos
CREATE TABLE produtos (
    produto_id INT PRIMARY KEY,
    nome VARCHAR(100),
    preco DECIMAL(10, 2),
    descricao TEXT
);

-- Tabela Vendas
CREATE TABLE vendas (
    venda_id INT PRIMARY KEY,
    cliente_id INT,
    data_venda DATE,
    total DECIMAL(10, 2),
    FOREIGN KEY (cliente_id) REFERENCES clientes(cliente_id)
```

```sql
CREATE TABLE itens_venda (
    item_venda_id INT PRIMARY KEY,
    venda_id INT,
    produto_id INT,
    quantidade INT,
    preco DECIMAL(10, 2),
    FOREIGN KEY (venda_id) REFERENCES vendas(venda_id),
    FOREIGN KEY (produto_id) REFERENCES produtos(produto_id)
);

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

-- Tabela Pagamentos
CREATE TABLE pagamentos (
    pagamento_id INT PRIMARY KEY,
    venda_id INT,
    metodo_pagamento VARCHAR(50),
    valor DECIMAL(10, 2),
    data_pagamento DATE,
    FOREIGN KEY (venda_id) REFERENCES vendas(venda_id)
);

```

```sql
CREATE VIEW relatorio_vendas_periodo AS
SELECT
    v.data_venda,
    c.nome AS cliente_nome,
    SUM(iv.quantidade * iv.preco) AS total_venda
FROM
    vendas v
JOIN clientes c ON v.cliente_id = c.cliente_id
JOIN itens_venda iv ON v.venda_id = iv.venda_id
GROUP BY
    v.data_venda, c.nome
ORDER BY
    v.data_venda DESC;

```

Vantagens:

- Facilita a consulta do estoque disponível, levando em consideração as vendas realizadas.
- Pode ser útil para o time de compras ou logística planejar a reposição de produtos.

CONCLUSÃO

Apesar dessas desvantagens, as views continuam sendo uma ferramenta poderosa para simplificar consultas, melhorar a segurança e abstrair complexidade. Contudo, é essencial avaliar o impacto no desempenho, a necessidade de atualizações e o tipo de dados com os quais você está trabalhando ao decidir se vai usar uma view simples, complexa ou materializada.

BIBLIOGRAFIA

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

https://ric.cps.sp.gov.br/bitstream/123456789/9920/1/bancodedados_2020_1_anabeatrizsantos_osbeneficiosdautilizacaodeviews.pdf

https://www.ibm.com/docs/pt-br/db2/11.1?topic=views-updatable

https://www.devmedia.com.br/mysql-trabalhando-com-views/8724