Views

Em PostgreSQL, uma view é uma representação virtual de uma tabela que é criada usando uma consulta SQL. É basicamente uma consulta armazenada que é definida como um objeto de banco de dados e pode ser tratada como uma tabela física em muitos casos. Uma view é uma tabela virtual que não é armazenada fisicamente em disco, mas que é tratada como se fosse uma tabela real.

Views são como janelas em uma casa: assim como uma janela permite que você veja uma parte específica do exterior de sua casa, uma view permite que você veja uma parte específica dos dados armazenados em uma ou mais tabelas do PostgreSQL. Assim como você pode ter várias janelas diferentes em uma casa para diferentes vistas, você pode ter várias views diferentes em um banco de dados para diferentes perspectivas dos dados.

As views são usadas principalmente para simplificar consultas complexas e reduzir a quantidade de código SQL que precisamos escrever. Elas também podem ser usadas para limitar o acesso a certas informações, permitindo que os usuários vejam apenas as informações relevantes para eles.

As views podem ser criadas usando a seguinte sintaxe:

```
CREATE VIEW nome_da_view AS
SELECT coluna1, coluna2, ...
FROM tabela
WHERE condição
```

Aqui, `nome_da_view` é o nome da view que estamos criando, `coluna1`, `coluna2`, etc. são as colunas da view e `tabela` é a tabela na qual estamos baseando a view. A cláusula `WHERE` é opcional e pode ser usada para filtrar os resultados.

Após criar a view, podemos consultá-la usando a sintaxe básica de SELECT:

```
SELECT * FROM nome_da_view;
```

Também é possível executar consultas mais complexas em cima da view, incluindo joins, subconsultas e funções de agregação.

Algumas características importantes das views em PostgreSQL incluem:

- As views podem ser atualizáveis ou somente leitura, dependendo da definição da consulta subjacente.

As views são armazenadas no banco de dados, permitindo que outros usuários ou aplicativos acessem a mesma definição.

As views podem ser usadas para abstrair a complexidade da consulta subjacente e fornecer uma interface mais simples para os usuários.

As views podem ser usadas para proteger dados sensíveis, permitindo que os usuários vejam apenas as informações relevantes para eles.

Suponha que temos duas tabelas: "clientes" e "pedidos". A tabela "clientes" tem informações sobre os clientes da loja, enquanto a tabela "pedidos" tem informações sobre os pedidos que os clientes fizeram.

```
CREATE TABLE clientes (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    telefone VARCHAR(20)
);

CREATE TABLE pedidos (
    id SERIAL PRIMARY KEY,
    cliente_id INTEGER REFERENCES clientes(id),
    data_pedido DATE NOT NULL,
    total DECIMAL(10, 2) NOT NULL
);
```

Agora, vamos supor que queremos criar uma view que mostre o nome do cliente, o número de pedidos que ele fez e o valor total desses pedidos. Podemos fazer isso com o seguinte código SQL:

```
CREATE VIEW resumo_pedidos AS
    SELECT  c.nome,
            COUNT(p.*) AS num_pedidos,
            SUM(p.total) AS valor_total
        FROM clientes c
            INNER JOIN pedidos p
                ON c.id = p.cliente_id
        GROUP BY c.nome;
```

Aqui, estamos usando uma cláusula INNER JOIN para juntar as tabelas "clientes" e "pedidos" com base no campo "cliente_id". Em seguida, estamos usando uma cláusula GROUP BY para agrupar os resultados pelo nome do cliente. Finalmente, estamos usando a função COUNT para contar o número de pedidos e a função SUM para calcular o valor total dos pedidos para cada cliente.

Agora que temos a view "resumo_pedidos", podemos usá-la para obter informações resumidas sobre os pedidos de cada cliente com o seguinte código SQL:

```
SELECT * FROM resumo_pedidos;
```

Isso irá nos retornar um conjunto de resultados com o nome de cada cliente, o número de pedidos que ele fez e o valor total desses pedidos. Podemos usar essa view sempre que precisarmos dessas informações resumidas, sem precisar escrever a consulta SQL complexa toda vez.

Referências e materiais complementares

https://www.postgresql.org/docs/current/tutorial-views.html