Views

Uma visão (VIEW) é uma tabela que não existe fisicamente, é derivada de uma consulta em uma ou mais tabelas.

CREATE VIEW

O comando CREATE VIEW cria uma visão.

```
CREATE [OR REPLACE] [TEMP|TEMPORARY] [ RECURSIVE ] VIEW
name [(column_name [,...])] AS query
```

Exemplo:

```
CREATE VIEW alunos_sem_curso
            AS SELECT * FROM aluno WHERE curso IS NULL;
SELECT * FROM alunos_sem_curso;
 matricula |      nome       |    rg    | curso | serie | turma
-----------+-----------------+----------+-------+-------+-------
        10 | Fernanda        | 29563735 |       |       |
```

CREATE VIEW

A visão pode ser derivada de mais de uma tabela.

Exemplo:

```
CREATE VIEW carros_vendidos AS SELECT cliente.nome AS
cliente, automovel.modelo, fabricante.nome AS fabricante,
revenda.nome AS revenda FROM
venda,cliente,automovel,fabricante,revenda WHERE
cliente.codigo=venda.cliente AND
automovel.codigo=venda.automovel AND
revenda.codigo=venda.revenda AND
fabricante.codigo=automovel.fabricante;
```

CREATE VIEW

```
SELECT * FROM carros_vendidos;
  cliente   |     modelo       | fabricante |     revenda
------------+------------------+------------+-----------------
 Paulo      | Ford Ka          | Ford       | Paraiso
 Joana      | Gol              | Volkswagen | Alameda
 Jose       | Siena            | Fiat       | Cabana
 Paulo      | Golf             | Volkswagen | Cabana
 Maria      | Corsa Sedan      | Chevrolet  | Santana
 Marcia     | Polo             | Volkswagen | Triangulo
 Maria      | Fiesta           | Ford       | Alameda
 Jose       | Corsa Sedan      | Chevrolet  | Paraiso
 Jose       | Palio            | Fiat       | Santana
 Marcia     | Golf             | Volkswagen | Vale
 Joana      | Polo             | Volkswagen | Paraiso
 Paulo      | Corsa Sedan      | Chevrolet  | Paraiso
 Jose       | Fiesta           | Ford       | Portal
```

CREATE VIEW

Na definição da visão podem ser atribuídos nomes para as colunas, diferentes dos obtidos na consulta.

Exemplo:

```
CREATE VIEW vendas_valor (cliente, automovel, fabricante,
revenda, custo, valor) AS SELECT cliente.nome,
automovel.modelo, fabricante.nome, revenda.nome,
automovel.preco, venda.valor FROM
venda,cliente,automovel,fabricante,revenda WHERE
cliente.codigo=venda.cliente AND
automovel.codigo=venda.automovel AND
revenda.codigo=venda.revenda AND
fabricante.codigo=automovel.fabricante;
```

CREATE VIEW

```
SELECT * FROM vendas_valor;
```

cliente	automovel	fabricante	revenda	custo	valor
Paulo	Ford Ka	Ford	Paraiso	15000.00	17500.00
Joana	Gol	Volkswagen	Alameda	25000.00	28000.00
Jose	Siena	Fiat	Cabana	26000.00	28000.00
Paulo	Golf	Volkswagen	Cabana	39000.00	42000.00
Maria	Corsa Sedan	Chevrolet	Santana	10000.00	11500.00
Marcia	Polo	Volkswagen	Triangulo	27500.00	29500.00
Maria	Fiesta	Ford	Alameda	20000.00	22100.00
Jose	Corsa Sedan	Chevrolet	Paraiso	12500.00	15500.00
Jose	Palio	Fiat	Santana	23000.00	24500.00
Marcia	Golf	Volkswagen	Vale	37000.00	39500.00
Joana	Polo	Volkswagen	Paraiso	29000.00	31000.00
Paulo	Corsa Sedan	Chevrolet	Paraiso	16000.00	17500.00
Jose	Fiesta	Ford	Portal	20000.00	21500.00

CREATE VIEW

A consulta que define os dados da visão pode ter qualquer opção do comando SELECT.

Exemplo:

```
CREATE VIEW cliente_gasto AS SELECT venda.cliente,
cliente.nome, SUM(venda.valor) FROM cliente,venda WHERE
cliente.codigo=venda.cliente GROUP BY
venda.cliente,cliente.nome;
```

```
SELECT * FROM cliente_gasto;
 cliente |    nome     |   sum
---------+-------------+----------
 06      | Marcia      | 69000.00
 02      | Paulo       | 77000.00
 01      | Jose        | 89500.00
 04      | Joana       | 59000.00
 03      | Maria       | 33600.00
```

CREATE VIEW

Visões podem ser definidas com base em outras visões.

Exemplo:

```
CREATE VIEW revenda_lucro AS SELECT revenda,SUM(valor-
custo) FROM vendas_valor GROUP BY revenda;

SELECT * FROM revenda_lucro;
     revenda      |   sum
------------------+---------
 Portal           | 1500.00
 Alameda          | 5100.00
 Santana          | 3000.00
 Paraiso          | 9000.00
 Triangulo        | 2000.00
 Vale             | 2500.00
 Cabana           | 5000.00
```

Views

Na seleção de registros da uma visão pode ser utilizado qualquer opção do comando SELECT.

Exemplo:

```
SELECT fabricante,SUM(valor) FROM vendas_valor WHERE
fabricante!='Ford' GROUP BY fabricante ORDER BY
fabricante;
 fabricante |    sum
------------+-----------
 Chevrolet  |  44500.00
 Fiat       |  52500.00
 Volkswagen | 170000.00
```

DROP VIEW

O comando **DROP VIEW** remove uma visão.

```
DROP VIEW [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]
```

Exemplo:

```
DROP VIEW alunos_sem_curso;
```

Visões Recursivas

A cláusula **RECURSIVE** permite criar visões recursivas, equivalente a criar a visão com **WITH RECURSIVE** na definição da consulta.

Exemplo:

```
CREATE RECURSIVE VIEW organizacao_lista(nivel, nome, vendas, superior, sequencia, total) AS
(
SELECT 1, nome, vendas, superior, CAST(nome AS TEXT ),vendas FROM organizacao WHERE superior IS NULL
UNION
SELECT nivel+1, organizacao.nome, organizacao.vendas, organizacao.superior, sequencia || ' > ' || organizacao.nome,CAST(organizacao.vendas+organizacao_lista.total AS NUMERIC(9,2) ) FROM organizacao, organizacao_lista WHERE organizacao.superior=organizacao_lista.nome
);
```

Visões Recursivas

```
SELECT * FROM organizacao_lista ORDER BY sequencia;
```

nivel	nome	vendas	superior	sequencia	total
1	Andreia	8100.00		Andreia	8100.00
2	Fabio	7800.00	Andreia	Andreia > Fabio	15900.00
3	Raquel	7100.00	Fabio	Andreia > Fabio > Raquel	23000.00
2	Sandra	8400.00	Andreia	Andreia > Sandra	16500.00
3	Ana	6800.00	Sandra	Andreia > Sandra > Ana	23300.00
3	Silvia	8900.00	Sandra	Andreia > Sandra > Silvia	25400.00
1	Claudia	7500.00		Claudia	7500.00
2	Andre	8300.00	Claudia	Claudia > Andre	15800.00
3	Luana	5200.00	Andre	Claudia > Andre > Luana	21000.00
3	Marta	7800.00	Andre	Claudia > Andre > Marta	23600.00
3	Sandro	8100.00	Andre	Claudia > Andre > Sandro	23900.00
4	Luis	7200.00	Sandro	Claudia > Andre > Sandro > Luis	31100.00
4	Marcos	8100.00	Sandro	Claudia > Andre > Sandro > Marcos	32000.00
2	Nanci	8300.00	Claudia	Claudia > Nanci	15800.00
3	Paula	6900.00	Nanci	Claudia > Nanci > Paula	22700.00
2	Pedro	9200.00	Claudia	Claudia > Pedro	16700.00
1	Rodrigo	8900.00		Rodrigo	8900.00

Views

Visões são utilizadas para criar uma camada de abstração entre os aplicativos e usuários e a estrutura como os dados são armazenados no banco, permitindo que haja uma independência lógica entre a forma que os dados são armazenados e a forma como os aplicativos e usuários enxergam os dados.

Alterações na estrutura de armazenamento podem ser mascaradas com visões que mantenham a mesma estrutura original, de modo que os aplicativos e usuários não percebam as alterações.

Views

Visões também podem ser utilizadas como recurso de segurança para dar acesso aos usuários a apenas aos dados da visão, sem que esses usuários possam acessar as tabelas das quais a visão é derivada.

Views

Exemplo:

(como dono do banco aluno)

```
CREATE VIEW aluno_nome AS SELECT matricula,nome FROM aluno;
GRANT SELECT ON aluno_nome TO teste;
```

(como usuário sem permissão de acesso a tabela aluno)

```
SELECT * FROM aluno;
ERROR:  permission denied for relation aluno
SELECT * FROM aluno_nome;
 matricula |      nome
-----------+-----------------
         1 | Ana Lucia
         2 | Luis Claudio
         3 | Marcelo
         4 | Debora
         5 | Fernanda
         6 | Alvaro
         7 | Claudio
         8 | Andrea
         9 | Carla
        10 | Fernanda
```

Atualizações em Visões

Atualizações (INSERT, UPDATE, DELETE) em visões baseadas em apenas uma tabela real podem facilmente ser mapeadas em comandos para atualização da tabela correspondente, desde de que não sejam para atualizar campos referentes a agregações ou outros campos calculados. Mas essas atualizações podem levar a inconsistências como no caso de uma inserção em uma visão que não tenha algum campo não nulo da tabela real.

Mapear atualizações em visões baseadas em mais de uma tabela para comandos para atualização das tabelas correspondentes pode não ser possível ou levar a efeitos colaterais indesejáveis pois, um mesmo registro de uma das tabelas reais pode estar relacionado a mais de um registro da visão.

Devido a dificuldade para mapear as atualizações das visões em atualizações das tabelas reais, o padrão SQL define a atualização de forma restritiva, impondo condições para definir visões que podem ser atualizáveis ou não. Alguns SGBD não implementam atualização em visões, sendo essas então utilizadas apenas para consultas.

Atualizações em Visões

A partir da versão 9.3, o PostgreSQL implementa atualização para visões que sigam as seguintes regras:

- A visão deve apenas uma entrada em sua lista **FROM**, que deve ser uma tabela ou outra visão atualizável.
- A definição da visão não deve conter **WITH**, **DISTINCT**, **GROUP BY**, **HAVING**, **OFFSET** ou **FETCH** no nível superior.
- A definição da visão não deve conter operações de conjunto (**UNION**, **INTERSECT** ou **EXCEPT**) no nível superior.
- Todas as colunas na lista de seleção da visão devem ser simples referências a colunas da relação subjacente. Elas não podem ser expressões, literais ou funções. As colunas do sistema também não podem ser referenciadas.
- Nenhuma coluna da relação subjacente pode aparecer mais de uma vez na lista de seleção da visão.

A alteração ou inserção de registros em uma visão pode gerar registros na tabela original que não se encaixam na definição da visão e consequentemente, não fazem parte da visão.

Atualizações em Visões

Exemplo:

```
SELECT * FROM aluno;
 matricula |      nome        |    rg    | curso | serie | turma
-----------+------------------+----------+-------+-------+-------
         1 | Ana Lucia        | 20143531 | 0001  | 1     | A
         2 | Luis Claudio     | 22336362 | 0001  | 1     | A
         3 | Marcelo          | 25343256 | 0001  | 1     | A
         4 | Debora           | 20356328 | 0001  | 1     | B
         5 | Fernanda         | 26344325 | 0001  | 1     | B
         6 | Alvaro           | 21764527 | 0001  | 1     | B
         7 | Claudio          | 23336368 | 0002  | 1     | A
         8 | Andrea           | 28456474 | 0002  | 1     | A
         9 | Carla            | 23636731 | 0002  | 2     | A
        10 | Fernanda         | 29563735 |       |       |
SELECT * FROM alunos_sem_curso ;
 matricula |      nome        |    rg    | curso | serie | turma
-----------+------------------+----------+-------+-------+-------
        10 | Fernanda         | 29563735 |       |       |
```

Atualizações em Visões

```
UPDATE alunos_sem_curso SET nome='Aline';

SELECT * FROM aluno;
 matricula |      nome        |    rg    | curso | serie | turma
-----------+------------------+----------+-------+-------+-------
         1 | Ana Lucia        | 20143531 | 0001  | 1     | A
         2 | Luis Claudio     | 22336362 | 0001  | 1     | A
         3 | Marcelo          | 25343256 | 0001  | 1     | A
         4 | Debora           | 20356328 | 0001  | 1     | B
         5 | Fernanda         | 26344325 | 0001  | 1     | B
         6 | Alvaro           | 21764527 | 0001  | 1     | B
         7 | Claudio          | 23336368 | 0002  | 1     | A
         8 | Andrea           | 28456474 | 0002  | 1     | A
         9 | Carla            | 23636731 | 0002  | 2     | A
        10 | Aline            | 29563735 |       |       |

SELECT * FROM alunos_sem_curso ;
 matricula |      nome        |    rg    | curso | serie | turma
-----------+------------------+----------+-------+-------+-------
        10 | Aline            | 29563735 |       |       |
```

Atualizações em Visões

```
UPDATE alunos_sem_curso SET curso='0003';

SELECT * FROM aluno;
 matricula |      nome       |    rg    | curso | serie | turma
-----------+-----------------+----------+-------+-------+-------
         1 | Ana Lucia       | 20143531 | 0001  | 1     | A
         2 | Luis Claudio    | 22336362 | 0001  | 1     | A
         3 | Marcelo         | 25343256 | 0001  | 1     | A
         4 | Debora          | 20356328 | 0001  | 1     | B
         5 | Fernanda        | 26344325 | 0001  | 1     | B
         6 | Alvaro          | 21764527 | 0001  | 1     | B
         7 | Claudio         | 23336368 | 0002  | 1     | A
         8 | Andrea          | 28456474 | 0002  | 1     | A
         9 | Carla           | 23636731 | 0002  | 2     | A
        10 | Aline           | 29563735 | 0003  |       |

SELECT * FROM alunos_sem_curso ;
 matricula | nome | rg | curso | serie | turma
-----------+------+----+-------+-------+-------
(0 registro)
```

Atualizações em Visões

Para simular atualização em visões que não atendam às condições de visões atualizáveis, é possível utilizar gatilhos ou o sistema de regras do PostgreSQL, onde deve ser definido pelo próprio criador do gatilho ou da regra, como mapear a atualização da visão em atualização(ões) na(s) tabela(s) correspondente(s).

Visões Materializadas

Ao definir uma visão, fica armazenado no banco de dados, a consulta que define a visão, e ao executar umaconsultar sobre a visão, essa consulta é executada novamente para gerar os dados que serão utilizados na consulta sobre a visão. Dessa forma, a consulta sobre a visão sempre será executada sobre dados atualizados, porém, exige que a consulta de definição da visão seja sempre reavaliada.

Para acelerar a execução da consulta sobre a visão, podemos utilizar visões materializadas, onde além da consulta de definição da visão, também é permanentemente armazenado no banco de dados o resultado dessa consulta. Dessa forma, ao executar uma consulta sobre a visão, são utilizados os dados já armazenados no banco de dados sem a necessidade de reavaliar a consulta de definição da visão. A execução da consulta sobre a visão materializada é mais rápida do que sobre a visão comum, porém, pode ser executada sobre dados desatualizados se alguma tabela envolvida na definição da visão foi alterada após o armazenamento dos dados da visão materializada.

CREATE MATERIALIZED VIEW

O comando CREATE MATERIALIZED VIEW cria uma visão materializada.

```
CREATE MATERIALIZED VIEW table_name [ (column_name
[, ...] ) ] AS query [ WITH [ NO ] DATA ]
```

Exemplo:

```
CREATE VIEW fabricante_total(fabricante,total) AS SELECT
fabricante.nome,SUM(valor) FROM venda, automovel,
fabricante WHERE automovel.codigo=venda.automovel AND
fabricante.codigo=automovel.fabricante GROUP BY nome;
```

```
CREATE MATERIALIZED VIEW
fabricante_materializado(fabricante,total) AS SELECT
fabricante.nome,SUM(valor) FROM venda, automovel,
fabricante WHERE automovel.codigo=venda.automovel AND
fabricante.codigo=automovel.fabricante GROUP BY nome;
```

CREATE MATERIALIZED VIEW

```
SELECT * FROM fabricante_total;
 fabricante |   total
------------+-----------
 Fiat       |  52500.00
 Volkswagen | 170000.00
 Chevrolet  |  44500.00
 Ford       |  61100.00

SELECT * FROM fabricante_materializado;
 fabricante |   total
------------+-----------
 Fiat       |  52500.00
 Volkswagen | 170000.00
 Chevrolet  |  44500.00
 Ford       |  61100.00
```

CREATE MATERIALIZED VIEW

```
DELETE FROM venda WHERE ( SELECT fabricante.codigo FROM
fabricante,automovel WHERE
fabricante.codigo=automovel.fabricante AND
automovel.codigo=venda.automovel)='01';
```

```
SELECT * FROM fabricante_total;
 fabricante |  total
------------+----------
 Fiat       | 52500.00
 Chevrolet  | 44500.00
 Ford       | 61100.00
```

```
SELECT * FROM fabricante_materializado;
 fabricante |   total
------------+-----------
 Fiat       |  52500.00
 Volkswagen | 170000.00
 Chevrolet  |  44500.00
 Ford       |  61100.00
```

REFRESH MATERIALIZED VIEW

O comando REFRESH MATERIALIZED VIEW atualiza os dados de uma visão materializada.

```
REFRESH MATERIALIZED VIEW name [ WITH [ NO ] DATA ]
```

Exemplo:

```
REFRESH MATERIALIZED VIEW fabricante_materializado;
SELECT * FROM fabricante_materializado;
 fabricante |  total
------------+----------
 Fiat       | 52500.00
 Chevrolet  | 44500.00
 Ford       | 61100.00
```

Exercícios

Crie as visões:

a) tecnologia_1a(matricula,nome,rg) com os dados dos alunos da sala 1A do curso 0001 (Tecnologia da Informacao)

b) funcionarios_financeiro(funcionario,funcao,salario) com o nome, descrição da função e salário dos funcionarios da diretoria financeira

c) funcao_salario(funcao,salario) com a descrição da função e total de salários dos funcionários da função

d) produto_movimento(produto,entrada,saida) com a descrição, total de entradas e de saidas de todos os produtos

e) curso_alunos(curso,alunos) com o nome do curso e total de alunos do curso

f) fabricante_cidade(fabricante,cidade,lucro) com o nome do fabricante, nome da cidade e o lucro das vendas dos automoveis do fabricante na cidade