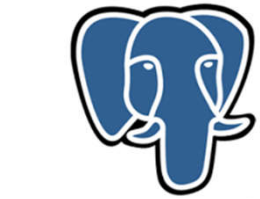
INSTITUTO FEDERAL
Paraíba
Campus João Pessoa

Aula
3

Banco de Dados II

Atualizada em 18/04/2019

Visões



PostgreSQL

Professor:
Dr. Alex Sandro da Cunha Rêgo

alex@ifpb.edu.br

Conteúdo

- Introdução
  - Definição
  - Vantagens x desvantagens
- Aplicabilidade
- Sintaxe de criação
- Consulta à visões
- Restrições de uso
- Comandos DROP e ALTER VIEW
- Visões materializadas

Visão

- **Definição**
  - Uma tabela virtual baseada no *result-set* de uma instrução SQL
    - Os campos da view podem vir de uma ou mais tabelas reais
- **Vantagens**
  - SEGURANÇA: restringir "quais colunas" de uma tabela podem ser acessadas (leitura)
    - Permite esconder dados que certos usuários não precisam visualizar
  - Podem conter valores calculados ou de resumo
  - Representação de consultas SELECT frequentemente utilizadas

Visão

- **Características**
  - Uma vez criadas, as **visões** passam a fazer parte do esquema de banco de dados
  - O usuário do banco de dados deve enxergar uma **visão** como uma tabela
  - Instruções SQL podem mesclar campos de tabelas reais e **visões**
  - Por ter uma instrução SQL própria embutida, a **visão**, na prática, delega a tarefa para sua correspondente instrução SQL desempenhar
  - Não se adiciona ORDER BY na instrução SQL da view, e sim no SELECT que faz o acesso

Visão

- **Sintaxe de criação**

```
CREATE VIEW <nome da visao> AS
( <instrução SELECT> );
```

  - Cenário de contextualização

numEmp	*agencia*	*Valor*
E170	*Tambau*	*1.500,00*
E200	*Cristo*	*2.800,00*

emprestimo

devedor

Nome_cliente	***Num_empres***
Jones	*E170*
Smith	*E230*
Carol	*E155*

Visão

- **Exemplo1:**
  - Um usuário específico do banco de dados precisa ter acesso ao nome da agência e nome de todos os clientes que têm um empréstimo em cada agência. Porém, este usuário não está autorizado a ver informações específicas aos empréstimos dos clientes.

```
CREATE VIEW cliente_emprestimo AS
(SELECT e.agencia, d.nome_cliente
 FROM devedor d JOIN emprestimo e ON
e.numEmp = d.num_empres;
```

Na consulta à visão:

```
SELECT * FROM cliente_emprestimo;
```

Visão

- **Exemplo2:**
  - Crie uma visão no banco de dados **myMail** que forneça automaticamente o nome do usuário e a quantidade de mensagens que o mesmo recebeu

```
CREATE VIEW estatistica_mensagens AS
( SELECT r.destinatario usuario,
  COUNT(*) QuantRecebidas
  FROM si_mensagem m JOIN si_recebedor r
  ON m.idMens = r.idMens
  GROUP BY r.destinatario
);
```

```
SELECT * FROM estatistica_mensagem;
```

OBS: Não se usa **ORDER BY** na view, e sim no **SELECT**

Visão

- **Restrições (PostgreSQL)**
  - **UPDATE e INSERT em uma visão:**
    - ✓ Apenas se a visão foi baseada em uma única tabela, **não teremos problemas.** O PostgreSQL fará a conversão da instrução para a tabela correspondente
    - ✓ A query que defina a view não pode conter uma das seguintes cláusulas: GROUP BY, HAVING, LIMIT, DISTINCT, UNION, INTERSECT e EXCEPT
    - ✓ A lista de seleção não pode conter **funções agregadas** (COUNT, SUM, MIN, MAX e AVG)
    - ✓ Uma ***updatable view*** pode conter colunas atualizáveis ou não. Porém, se você tentar fazer um INSERT ou UPDATE em uma coluna não atualizável, será gerado um erro

Visão

- **Eliminando uma visão**

```
DROP VIEW [IF EXISTS] <nome da visão>;
```

- **Alterando a query de uma visão**

```
CREATE OR REPLACE VIEW <nome da visão>
AS query;
```

- **Alterando o nome de uma visão**

```
ALTER VIEW <nome anterior>
RENAME TO <novo nome>;
```

Visão

- **Perguntas Intrigantes...**
  1. Quando uma tabela originária de *view* é atualizada, os dados da também são atualizados?

Views não são automaticamentes armazenadas em **cache**. Quando um SELECT é emitido a partir de uma **view**, o SGBD executa a query armazenada e devolve o resultado a partir de tabelas. Lembre-se: a **view** é uma entidade lógica

Nota: SGBD's podem implementar regras particulares para melhorar a questão de desempenho.

Visão

- **Perguntas Intrigantes**
  2. As **visões** melhoram a questão do **desempenho** das consultas?

Views atuam apenas como um objeto de banco de dados armazenado, logo, a view NÃO melhorará o desempenho das consultas. A visão é um meio de mascarar ou reorganizar as informações para que apareçam da forma desejada

Visão Materializada

- **Views Materializadas**
  - Uma extensão do conceito de **view** provida pelo PostgreSQL
  - Dados da **view** são armazenados fisicamente
  - Armazena em cache o resultado de uma query complexa e custosa, permitindo que você povoe a view no momento da criação
- **Benefício**
  - Útil nos casos em que se necessita rápido acesso aos dados
  - São usados com frequência em aplicações de *Data Warehouse* e *Business Inteligence*

Visão Materializada

- **Sintaxe**

```
CREATE MATERIALIZED VIEW <nome_visão>
AS query
WITH [NO] DATA
```

  - **WITH DATA**: carrega os dados na view materializada no momento da criação. Caso contrário, é marcada como ilegível e dados não podem ser consultados (até que os carregue)

- **Atualizando dados (REFRESH)**

```
REFRESH MATERIALIZED VIEW <nome_visão>
```

- **Removendo a view**

```
DROP MATERIALIZED VIEW <nome_visão>
```

Referências Bibliográficas

- Creating PostgreSQL Updatable Views: *http://www.postgresqltutorial.com/postgresql-updatable-views/*