

# Sistemas de Informação

# **Bando de Dados 1**

Prof. Dr. Ronaldo Castro de Oliveira

[ronaldo.co@ufu.br](mailto:ronaldo.co@ufu.br)

FACOM



Visões



# Visão - Definição

---

## ▶ Definição:

- ▶ Visão é uma tabela derivada de outras tabelas.
- ▶ OBS: geralmente é montada dinamicamente, uma tabela virtual, mas pode existir fisicamente, neste caso, chamamos de visão materializada.

## ▶ Objetivos:

- ▶ Disponibilidade: simplificar e centralizar a definição de consultas frequentes, evitando erros e melhorando a produtividade de usuários;
- ▶ • Confidencialidade: restringir acesso somente a projeções ou seleções de tabelas reais;
- ▶ • Integridade: evitar alterações indevidas no BD;



# VISÕES

---

- ▶ Uma visão é uma descrição alternativa para um sub-conjunto dos dados selecionados de uma ou mais tabelas da base de dados (tabela virtual). É possível executar operações de seleção, junção, inserção, atualização e remoção sobre uma visão, como se ela fosse uma tabela regular da base de dados, com apenas poucas restrições:
  - ▶ Existem visões atualizáveis e visões “read-only”
  - ▶ No PostgreSQL elas são *read-only* – ou seja, não é possível inserir, deletar ou atualizar por meio de uma visão
- ▶ Sintaxe

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

---

# Visões – Exemplo

---

- ▶ **CREATE VIEW** worksname  
    **AS SELECT** fname, lname, pname, hours  
    **FROM** employee, works\_on, project  
    **WHERE** ssn=essn **AND** pno=pnumber;
- ▶ **GRANT SELECT ON** worksname **TO** usuario;
- ▶ **SELECT \* FROM** worksname;
- ▶ OBS: a última consulta é mais simples (para o usuário da visão)



# Confidencialidade em Visões - Exemplo

---

- ▶ **CREATE VIEW** empdepto5  
    **AS SELECT** fname, minit, lname, ssn,  
        address, sex, superssn  
    **FROM** employee **WHERE** dno=5;
- ▶ OBS: o usuário da visão terá acesso somente a uma projeção ou seleção dos dados de empregados, neste caso:
  - ▶ projeção não inclui coluna “salary”
  - ▶ seleção restringe aos dados de empregados do departamento 5



# Integridade em Visões - Exemplo

---

- ▶ **CREATE VIEW** worksname  
    **AS SELECT** fname, lname, pname, hours  
    **FROM** employee, works\_on, project  
    **WHERE** ssn=essn **AND** pno=pnumber;
- ▶ **OBS:**
  - ▶ não é viável fazer atualização por meio de uma visão como esta,
  - ▶ então, um usuário com acesso por meio dessa visão não pode atualizar o BD,
  - ▶ diminuindo a chance de alterações indevidas no BD



# Implementação de Visões

---

- ▶ Existem duas formas de um SGBD implementar visões:
  - ▶ **Modificação de consultas** (QM): a visão é criada a cada consulta
    - ▶ **VANTAGEM**: não é necessário mecanismo de atualização para garantia de consistência da visão em relação às tabelas-base
    - ▶ **DESvantAGEM**: desempenho de consultas frequentes é prejudicada





# Implementação de Visões

---

- ▶ **Materialização de Visões (VM)**: a visão é criada na primeira consulta
  - ▶ **VANTAGEM**: consultas frequentes à visão têm bom desempenho
  - ▶ **DESVANTAGEM**: atualizações nas tabelas-base devem ser propagadas para as visões

OBS: são muito utilizadas em aplicações onde os dados podem ficar temporariamente desatualizados, com atualizações periódicas, por exemplo, dados estatísticos, pois mantêm a vantagem de desempenho sem prejuízo na propagação de atualizações.



# Visões Atualizáveis

---

- ▶ Visões são chamadas atualizáveis se permitem aos usuários realizarem alterações nos dados do banco de dados por meio da visão
- ▶ O PostgreSQL não implementa diretamente visões atualizáveis, mas pode-se implementá-la por meio de **gatilhos (visto mais a frente)**, ou seja, ativar um gatilho quando receber um update na visão, atualizando a(s) tabela(s) base(s).
- ▶ UPDATE empdepto5  
SET Iname = 'Watson'  
WHERE ssn='123456789';
- ▶ OBS: para isso a visão deve ser definida como atualizável, o que não é o caso.



# SQL92 e Visões Atualizáveis

## Não funciona no PostgreSQL

---

**CREATE VIEW** visao [(coluna [, ...])]

**AS SELECT** ...

[**WITH** [ **CASCADE** | **LOCAL** ] **CHECK OPTION** ]

**WITH** ... **CHECK OPTION**: torna a visão  
"atualizável", controlando atualizações  
somente de dados que pertencem à visão

- ▶ **CASCADE**: propaga atualizações às suas visões derivadas, se houverem.
- ▶ **OBS**: a implementação pode ser por meio de QM ou VM



# Visões Atualizáveis - Exemplo

---

- ▶ **CREATE VIEW** empdepto5  
    **AS SELECT** fname, minit, fname, ssn,  
        address, sex, superssn  
    **FROM** employee **WHERE** dno=5  
    **WITH CHECK OPTION**;
- ▶ **UPDATE** empdepto5  
    **SET** lname = 'Watson'  
    **WHERE** ssn='123456789';
- ▶ OBS: observe que a chave primária faz parte da visão, o que facilita a atualização do banco de dados.



# O Problema de Visões Atualizáveis

---

- ▶ O problema de atualização por meio de visões é a ambiguidade na interpretação do comando.
- ▶ Por exemplo, seja a visão:

```
CREATE VIEW seg(ssn, name, sex) AS  
    ( SELECT ssn, fname, sex FROM employee)  
    UNION  
    ( SELECT essn, name, sex FROM dependent )  
    WITH CHECK OPTION;
```

```
INSERT INTO seg ('123456789', 'Jose', 'M');
```

- ▶ Em qual tabela base será inserida a tupla?



# Restrições para visões atualizáveis

---

- ▶ Em geral, para ser atualizável a visão não deve conter:
  - ▶ 1. junção;
  - ▶ 2. função de agregação;
  - ▶ 3. subconsultas com tabela na cláusula FROM;
  - ▶ 4. cláusula DISTINCT.
- ▶ **OBS: Em geral, para ser atualizável, a visão deve ser derivada de apenas uma tabela base e deve conter a chave primária da tabela**



# Visões no PostgreSQL

---

- ▶ O PostgreSQL implementa visões por meio de **Modificação de Consultas**(QM), portanto as visões não são materializadas
- ▶ As visões no PostgreSQL **não são atualizáveis**
- ▶ O PostgreSQL tem um mecanismo próprio de definir visões atualizáveis e materializadas por meio de “**rules**”, que não serão estudadas neste curso
- ▶ Outro mecanismo de alteração em visões é o uso de **gatilhos**, que serão estudados posteriormente.



# Visões e DDL no PostgreSQL

---

- ▶ **CREATE** [ **OR REPLACE** ]  
    [ **TEMP** | **TEMPORARY** ]  
    **VIEW** nomevisao [ ( nomecoluna [, ...] ) ]  
    **AS** consulta...
- ▶ **OBS:**TEMP indica que a visão será automaticamente removida no término da sessão.
- ▶ **ALTER VIEW** nomevisao **RENAME TO** novonomevisao
- ▶ **DROP VIEW** [ **IF EXISTS** ]  
    nomevisao [, ...]  
    [ **CASCADE** | **RESTRICT** ]
- ▶ **Onde:**
  - ▶ **IF EXISTS:** não retorna erro caso a visão não exista
  - ▶ **CASCADE:** remove automaticamente outras visões que dependem desta
  - ▶ **RESTRICT:** rejeita operação caso existam dependências
  - ▶ No padrão o **DROP** afeta apenas uma visão por vez e não existe a cláusula **IF EXISTS**





# Exemplo

---

## ► Tabela Matrícula

**MATRICULA**(**codigoturma**,**nmat**,**nota**)

**codigoturma** => código da turma da disciplina

**nmat** => número de matrícula do aluno

**nota** => nota do aluno na disciplina



# Comando CREATE VIEW

---

-- cria a view Aprovações

**CREATE VIEW** Aprovacoes **AS**

**SELECT** \*

**FROM** Matricula

**WHERE** nota >= 60;

-- mostrando os alunos aprovados

**SELECT** \* **FROM** Aprovacoes;

-- Tentando inserir dados na base por meio da **VIEW**

**INSERT INTO** Aprovacoes **VALUES** (102, 1234,3);

(PostgreSQL 8.4 não deixa fazer a inserção: “ERRO: não pode inserir em uma visão”. Verificar a possibilidade na versão 9)



# Comando CREATE VIEW

## CREATE RULE

---

- ▶ Inserção por meio de visões: deve-se criar uma “rule” (regra)

-- Criando uma regra para inserção

**CREATE RULE** rap **AS**

**ON INSERT TO** Aprovacoes /\* A regra define o que fazer qdo um ‘insert’ é feito em Aprovações \*/

**DO INSTEAD** -- “faça no lugar”

**INSERT INTO** matricula **VALUES** (

NEW.codigoturma,

NEW.nmat,

NEW.nota);

-- ou seja, quando um ‘insert’ é feito na view Aprovações o que realmente ocorre é um ‘insert’ na tabela Matricula.

-- Apagando a rule “rap”:

-- **DROP RULE** rap **ON** aprovacoes

---



# Variável NEW

---

- ▶ Carrega as informações que foram passadas ao comando INSERT INTO

- ▶ Comando que ‘dispara’ a regra:

-- Tentando inserir dados na base por meio da VIEW

**INSERT INTO** Aprovacoes **VALUES** (102, 1234, 3);

-- Criando uma regra para inserção

**CREATE RULE** rap **AS**

**ON INSERT TO** Aprovacoes

**DO INSTEAD** -- “faça no lugar”

**INSERT INTO** matricula **VALUES** (

NEW.codigoturma,

NEW.nmat,

NEW.nota);



# Sintaxe para REGRA

---

```
CREATE [ OR REPLACE ] RULE nome AS ON evento  
  TO tabela [ WHERE condição ]  
  DO [ ALSO | INSTEAD ] { NOTHING | comando | ( comando ; comando ... ) }
```

Observe que mais de um comando pode ser executado no DO INSTEAD, desde que coloque parênteses



# Comando CREATE VIEW

## CREATE RULE

---

- ▶ Inserção por meio de visões: deve-se criar uma “rule” (regra)

-- Criando uma regra para inserção

**CREATE RULE** rap **AS**

**ON INSERT TO** Aprovacoes /\* A regra define o que fazer qdo um ‘insert’ é feito em Aprovações \*/

**DO INSTEAD** -- “faça no lugar”

**INSERT INTO** matricula **VALUES** (

NEW.codigoturma,

NEW.nmat,

NEW.nota);

-- ou seja, quando um ‘insert’ é feito na view Aprovações o que realmente ocorre é um ‘insert’ na tabela Matricula.

-- Apagando a rule “rap”:

-- **DROP RULE** rap **ON** aprovacoes

---



# Comando CREATE VIEW

## CREATE RULE

---

**INSERT INTO** Aprovacoes **VALUES** (102, 1234,3);

-- observe que a nota inserida é  $\leq 60$ , violando a condição usada na criação da VIEW. Para evitar essa violação deve-se usar “WITH CHECK OPTION”

-- cria a view aprovações

**CREATE VIEW** Aprovacoes  
**AS SELECT \*** **FROM** Matricula  
**WHERE** nota  $\geq 60$

**WITH CHECK OPTION;** -- Não implementado no postgresql 8.4 – verificar nas novas versões

**Exercício:** Verificar se (102, 1234,3) foi gravado na tabela matrícula e se aparece em aprovações.

---




# Comando CREATE VIEW

---

-- VIEW com as médias de cada aluno

```
CREATE VIEW Medias (NumeroMatricula, Media) AS
SELECT NMat, Avg(nota)
FROM Matricula
GROUP BY NMat;
```



Ao usar funções de agregação, deve ser informado o nome da coluna para essa função

-- Selecionando a maior média

```
SELECT NumeroMatricula, media
FROM Medias
WHERE media = (SELECT Max(media)
               FROM Medias);
```





# Visões Exemplo

---

- ▶ a) Visão 'managers' contendo nome do departamento, nome do gerente e o salário do gerente para todos os departamentos do BD;

```
CREATE OR REPLACE VIEW managers AS
```

```
SELECT dname, fname, ||' '||minit||' '||lname AS manager, salary
```

```
FROM employee, department
```

```
WHERE mgrssn=ssn;
```



# Visões Exemplo

---

- ▶ b) Visão 'researches' contendo nome do empregado, salário do empregado e nome de seu supervisor, para todos os empregados do departamento 'Research';

```
CREATE OR REPLACE VIEW researches AS
```

```
SELECT e.fname || ' ' || e.minit || ' ' || e.lname AS Employee,  
       s.fname || ' ' || s.minit || ' ' || s.lname AS Supervisor,  
       s.salary AS EmpSalary
```

```
FROM employee e, employee s, department d
```

```
WHERE d.dname='Research' AND d.dnumber=e.dno
```

```
AND s.ssn=e.superssn;
```



# Visões Exemplo

---

- ▶ c) Visão 'psummaryI' contendo para cada projeto seu nome, nome do departamento que o controla, número de empregados que trabalham no projeto e total de horas trabalhadas no projeto, por semana;

```
CREATE OR REPLACE VIEW psummaryI AS
SELECT pname AS project, dname AS department,
COUNT(DISTINCT essn) AS NumberOfEmp,
SUM(hours) AS SumOfHours
FROM project, works_on, department
WHERE pno=pnumber and dnum=dnumber
GROUP BY pname, dname;
```



# Visões Exemplo

---

- ▶ d) Visão 'psummary2' contendo para cada projeto onde trabalham mais de doze empregados, o nome do projeto, nome do departamento que o controla, número de empregados que trabalham no projeto e o total de horas trabalhadas no projeto

```
CREATE OR REPLACE VIEW psummary2
AS SELECT pname AS project, dname AS department
COUNT(DISTINCT essn) AS NumOfEmp,
SUM(hours) AS SumOfHours
FROM project, works_on, department
WHERE pno=pnumber and dnum=dnumber
GROUP BY pname, dname
HAVING COUNT(DISTINCT essn)>12;
```

---



**OBRIGADO A TODOS**

**DÚVIDAS**

