

Banco de Dados I

10 - Visões e Regras

Marcos Roberto Ribeiro



Instituto Federal Minas Gerais - Campus Bambuí

2018

- Visões são objetos de um banco de dados semelhantes a uma tabela, porém não armazenam dados;
- As visões são criadas por meio de consultas sobre outros objetos do banco de dados;
- As aplicações e usuários do banco de dados podem realizar consultas sobre as visões como se fossem tabelas.

Criando Visões

- As visões são criadas através da instrução **CREATE VIEW**;
- Para nossos exemplos vamos considerar o seguinte banco de dados

aluno

id_aluno: INTEGER [PK]
nome_aluno: VARCHAR(50)
cpf: CHAR(11) [AK]
data_nascimento: DATE
media: DOUBLE

professor

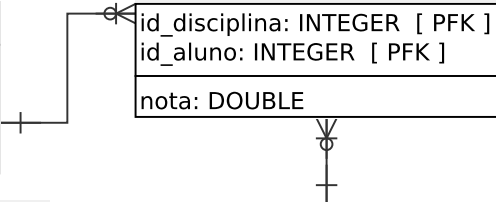
id_professor: INTEGER [PK]
nome_professor: VARCHAR(50)
sobrenome: VARCHAR(50)
area: VARCHAR(20)

matriculado

id_disciplina: INTEGER [PFK]
id_aluno: INTEGER [PFK]
nota: DOUBLE

disciplina

id_disciplina: INTEGER [PK]
nome_disciplina: VARCHAR(30)
carga_horaria: INTEGER
id_professor: INTEGER [FK]



Dados para o Banco de Dados Acadêmico I

Tabela *aluno*

id_aluno	nome_aluno	cpf	data_nascimento	media
1	José	111111111111	1990-01-20	85
2	João		1993-09-10	84.8
3	Maria	333333333333	1989-05-15	66.9
4	Ana	444444444444	1992-04-21	70.5

Dados para o Banco de Dados Acadêmico II

Tabela *professor*

id_professor	nome_professor	sobrenome	area
10	Roberto	Silva	Computação
20	Carlos	Alves	Matemática
30	José	Teodoro	Administração
40	Rodrigo	Martins	Engenharia

Dados para o Banco de Dados Acadêmico III

Tabela *disciplina*

id_disciplina	nome_disciplina	carga_horaria	id_professor
100	Algoritmos	80	10
200	Banco de Dados	80	10
300	Cálculo	60	20
400	Álgebra	60	20
500	Empreendedorismo	40	30
600	Redes	80	

Dados para o Banco de Dados Acadêmico IV

Tabela *matriculado*

id_disciplina	id_aluno	nota
100	1	89.5
200	1	78
300	1	90
400	1	82.5
100	2	88.7
200	2	81
400	2	77.5
500	2	92
100	3	72.5
200	3	52.8
400	3	83.3
100	4	71
200	4	70
300	3	59

Exemplo de Visão

- Vamos criar uma visão, considerando a necessidade de listar os nomes e médias dos alunos cuja média seja superior a 70:

```
CREATE OR REPLACE VIEW aluno70 AS  
SELECT nome_aluno, media  
FROM aluno  
WHERE media >= 70;
```

- Podemos visualizar os dados da visão com a seguinte consulta:

```
SELECT * FROM aluno70;
```

- Como exercício vamos modificar os dados da tabela **aluno** e verificar o que acontece com a visão **aluno70**.

Visão sobre mais de uma tabela

- Agora vamos criar uma visão sobre várias tabela para mostrar a disciplina, o aluno e a nota do aluno na disciplina:

```
CREATE OR REPLACE VIEW aluno_disciplina AS
SELECT d.nome_disciplina AS disciplina,
       a.nome_aluno AS aluno,
       m.nota
FROM aluno AS a,
     disciplina AS d,
     matriculado AS m
WHERE a.id_aluno = m.id_aluno
     AND d.id_disciplina = m.id_disciplina
ORDER BY disciplina,
        aluno;
```

- Modifique os dados das tabelas envolvidas na visão e verifique o que acontece.

- Alguns SGBD possuem *visões materializadas* que permitem a alteração dos dados através da visão, além de armazená-la em disco para melhorar o desempenho das consultas;
- O PostgreSQL não possui visões materializadas nativamente, mas podemos criar *visões editáveis* por meio de seus sistemas de regras (*rules*).
- Uma alteração de dados na visão editável propagará esta mesma alteração para as tabelas que deram origem a visão.

O Sistema de Regras do PostgreSQL

- O sistema de regras do PostgreSQL permite ao SGBD a alteração de um comando SQL de forma automática;
- O formato das instruções para criar regras é o seguinte:

```
CREATE [ OR REPLACE ] RULE nome AS ON (INSERT | UPDATE |  
↪ DELETE | SELECT)  
    TO (tabela | visão) [ WHERE condição ]  
    DO [ ALSO | INSTEAD ] { NOTHING | comando | ( comando ;  
↪ comando ... ) }
```

- A instrução **CREATE RULE** cria uma regra de reescrita sobre uma tabela ou visão;
- A condição em **WHERE** pode ser qualquer expressão condicional SQL;
- A palavra-chave **INSTEAD** substitui o comando original pelo novo comando, já palavra-chave **ALSO** executa o comando original e também o novo comando.

Visões e Regras

- No PostgreSQL uma visão é o mesmo que uma tabela com uma regra de rescrita no **SELECT**
- Isto significa que a visão **aluno2** pode ser criada de duas maneiras:

```
CREATE OR REPLACE VIEW aluno2 AS  
SELECT nome_aluno FROM aluno;
```

```
CREATE TABLE aluno2 (nome_aluno VARCHAR(50));  
CREATE RULE _RETURN AS  
ON SELECT TO aluno2 DO INSTEAD (  
    SELECT nome_aluno FROM aluno;  
);
```

- Na segunda instrução SQL o **SELECT** em **aluno2** é substituído por um **SELECT** em **aluno**.

Criando uma Visão Editável

- Vamos criar uma visão simples sobre a tabela aluno e torná-la editável:

```
CREATE OR REPLACE VIEW aluno_editavel AS
SELECT id_aluno, nome_aluno
FROM aluno;
```

```
CREATE RULE aluno_editavel_insert AS
ON INSERT TO aluno_editavel DO INSTEAD (
    INSERT INTO aluno(nome_aluno)
    VALUES (NEW.nome_aluno);
);
```

- Nas regras podemos usar a palavras-chave **NEW** para representar o dado a ser inserido pelo **INSERT** e também para representar o dado modificado em um **UPDATE**;
- Por que estamos inserindo apenas o **nome_aluno** na regra? O que aconteceria se fosse inserido o **id_aluno** também?

Criando uma Visão Editável (alteração e remoção)

```
CREATE RULE aluno_editavel_update AS
ON UPDATE TO aluno_editavel DO INSTEAD (
    UPDATE aluno
    SET id_aluno = NEW.id_aluno, nome_aluno = NEW.nome_aluno
    WHERE id_aluno = OLD.id_aluno;
);
```

```
CREATE RULE aluno_editavel_delete AS
ON DELETE TO aluno_editavel DO INSTEAD (
    DELETE FROM aluno
    WHERE id_aluno = OLD.id_aluno;
);
```

- A palavra-chave **OLD** representa o dado antes da exclusão ou modificação.

Execício I

- Criar as regras necessárias para que a média dos alunos seja atualizada automaticamente.

Execício II

```
CREATE OR REPLACE RULE matriculado_insert AS
ON INSERT TO matriculado DO ALSO (
    UPDATE aluno AS a
    SET media = am.media
    FROM (
        SELECT id_aluno,
               AVG(nota) AS media
        FROM matriculado AS m
        GROUP BY id_aluno) AS am
    WHERE a.id_aluno = am.id_aluno
        AND a.id_aluno = NEW.id_aluno;
);
```


Execício III

```
CREATE OR REPLACE RULE matriculado_update AS
ON UPDATE TO matriculado DO ALSO (
    UPDATE aluno AS a
    SET media = am.media
    FROM (
        SELECT id_aluno,
               AVG(nota) AS media
        FROM matriculado
        GROUP BY id_aluno) AS am
    WHERE a.id_aluno = am.id_aluno
    AND a.id_aluno IN (NEW.id_aluno, OLD.id_aluno);
);
```


Execício IV


```
CREATE OR REPLACE RULE matriculado_delete AS
ON DELETE TO matriculado DO ALSO (
    UPDATE aluno AS a
    SET media = am.media
    FROM (
        SELECT id_aluno,
               AVG(nota) AS media
        FROM matriculado
        GROUP BY id_aluno) AS am
    WHERE a.id_aluno = am.id_aluno
    AND a.id_aluno = OLD.id_aluno;
);
```

Execício V

- O que deveria ser feito para atualizar as médias de todos os alunos? Isto é interessante do ponto de vista do desempenho?

 (2012).
Postgresql documentation.

 Elmasri, R. and Navathe, S. B. (2011).
Sistemas de banco de dados.
Pearson Addison Wesley, São Paulo, 6 edition.

 Ramakrishnan, R. and Gehrke, J. (2008).
Sistemas de gerenciamento de banco de dados.
McGrawHill, São Paulo, 3 edition.