



AS33C

Banco de Dados 2

Professor: Eduardo Cotrin Teixeira



cotrin@utfpr.edu.br



SQL – Visões

- **Tabela** = relação armazenada
 - São definidas com o comando CREATE TABLE e existem fisicamente no BD.
 - Não mudam a menos que seja aplicado sobre elas algum comando SQL de modificação de dados.
- **Visão** = relação virtual
 - Visões são relações que não são armazenadas fisicamente.
 - São definidas como consultas.
 - Podem ser consultadas como se existissem fisicamente.

- **Por que Visões ?**
 - Por segurança (para esconder dados de alguns usuários).
 - Para tornar algumas consultas mais fáceis ou naturais de serem expressas.
 - Para modularidade.
- Aplicações de bancos de dados reais usam MUITAS visões.



SQL – Visões

■ Sintaxe:

CREATE VIEW <nome-visão> **AS** <definição-visão>;

ou

CREATE VIEW <nome-visão>(<nome-coluna>;<nome-coluna>...)
AS <definição-visão>;

- "definição-visão" é uma consulta.
- Sempre que uma consulta utiliza uma visão, é como se "definição-visão" estivesse sendo executada naquele momento para execução da consulta.



SQL – Visões

■ Exemplo - BD Star Trek (Peças):

Nova regra de negócio: PROJETOS CURTOS = projetos com duração abaixo de 5 meses e custo abaixo de \$ 30.000,00.

Visão - Número, Duração e Custo dos *projetos curtos*:

```
CREATE VIEW ProjetosCurtos AS
SELECT PNro, PDuracao, PCusto
FROM Projeto
WHERE PDuracao < 5 AND PCusto < 30000;
```

PNro	PDuracao	PCusto
P3	2	26700
P4	3	21200
P5	1	17000



SQL – Visões

- A visão *ProjetosCurtos* pode ser usada na seguinte consulta:
Mostre o número e a quantidade das peças fornecidas aos projetos curtos:

```
SELECT PeNro, Sum(Quant)
FROM ProjetosCurtos, Fornece_para
WHERE ProjetosCurtos.PNro = Fornece_para.PNro
GROUP BY PeNro;
```

PeNro	Sum(Quant)
PE1	5
PE2	1
PE3	2
PE4	5



SQL – Visões

- A consulta anterior pode ser "traduzida" como:

```
SELECT PeNro, Sum(Quant)
FROM (SELECT PNro, PDuracao, Pcusto FROM Projeto WHERE
      Pduracao < 5 AND PCusto < 30000) ProjetosCurtos,
      Fornece_para
WHERE ProjetosCurtos.PNro = Fornece_para.PNro
GROUP BY PeNro;
```

PeNro	Sum(Quant)
PE1	5
PE2	1
PE3	2
PE4	5



SQL – Visões

- É possível também renomear os campos resultantes em uma visão (pelo padrão SQL deve haver a mesma quantidade de nomes e de campos resultantes):

```
CREATE VIEW QuantPecasProjetosCurtos (Peca, Total) AS
SELECT PeNro, Sum(Quant)
FROM ProjetosCurtos, Fornece_para
WHERE ProjetosCurtos.PNro = Fornece_para.PNro
GROUP BY PeNro;
```

* O mesmo efeito pode ser obtido com uso de **AS** no SELECT.

* No PostgreSQL, não é obrigatório definir todos os nomes, que são aplicados da esquerda para a direita (p. ex., se houver só um nome, será aplicado ao primeiro campo).

Peca	Total
PE1	5
PE2	1
PE3	2
PE4	5



SQL – Visões

- Como vimos na visão anterior, podemos ainda criar uma visão sobre outra visão:

```
CREATE VIEW FornecedoresProjetosCurtos AS  
SELECT DISTINCT Fnome, FCidade, FCateg  
FROM ProjetosCurtos, Fornece_para, Fornecedor  
WHERE ProjetosCurtos.PNro = Fornece_para.PNro  
AND Fornece_para.FNro = Fornecedor.FNro;
```

FNome	FCidade	FCateg
C&M	São Paulo	D
Kirurgic	Campinas	A
Piloto's	Piracicaba	A
Equipament	São Carlos	C

■ Modificações em visões:

- Como são "consultas", **não é possível alterar a "estrutura"** das visões como é possível nas tabelas (ALTER TABLE).
- Uma **visão** é considerada **atualizável** somente se foi definida por meio da seleção simples (sem DISTINCT) de **atributos de uma única tabela ou visão atualizável**.
- A modificação nos dados da visão é possível (mas nunca desejável!) se for possível traduzir a modificação da visão em uma modificação equivalente em uma tabela de base.



SQL – Visões

■ Exemplos de modificação:

■ Nova Visão:

```
CREATE VIEW NomesProjetosCurtos AS  
SELECT PNro, PNome, PDuracao  
FROM Projeto
```

PNro	PNome	PDuracao
P3	Alfa	2
P4	Sea	3
P5	Paraíso	1

```
WHERE PDuracao < 5 AND PCusto < 30000;
```

■ O comando de inserção:

```
INSERT INTO NomesProjetosCurtos VALUES ('P6', 'Beta', 4);
```

■ É transformado em:

```
INSERT INTO Projeto(PNro, PNome, PDuracao, PCusto)  
VALUES ('P6', 'Beta', 4, NULL);
```

E não aparece na visão !! (PCusto não é menor que 30000)



SQL – Visões

- Para testar:

```
UPDATE Projeto SET PCusto = 28500 WHERE PNro = 'P6' ;
```

*** Agora P6 aparece na visão !**

- O comando de remoção:

```
DELETE FROM NomesProjetosCurtos WHERE PDuracao > 3 ;
```

- É transformado em:

```
DELETE FROM Projeto WHERE PDuracao > 3  
AND PDuracao < 5 AND PCusto < 30000 ;
```

Portanto, operações de dados diretamente em visões, mesmo que possíveis, não são desejáveis!

■ Remoção

- Sintaxe: **DROP VIEW <nome-visão>**
- A remoção da visão remove sua definição, sem afetar as tabelas que ela utiliza.
- Já o comando **DROP TABLE** torna as visões que utilizam a tabela apagada inutilizáveis (PostgreSQL não permite apagar a tabela se há visões dependentes, só é possível com uso de **CASCADE**, que apaga também as visões).



SQL – Visões

- Exemplo Final - Visão para mostrar Fornece_para com nomes no lugar dos números:

Fornece_para_Nomes

PeNome	FNome	PNome	Quant
Cinto	Equipament	Sea	5
Volante	C & M	Pegasus	1
Lanterna	Kirurgic	Sea	2
Limpador	Piloto's	Paraíso	3
Painel	Plastec	Detroit	1
Volante	C & M	Alfa	1
Limpador	Kirurgic	Paraíso	2

Fornece_para

PeNro	FNro	PNro	Quant
PE1	F5	P4	5
PE2	F2	P2	1
PE3	F3	P4	2
PE4	F4	P5	3
PE5	F1	P1	1
PE2	F2	P3	1
PE4	F3	P5	2



SQL – Visões

- Exemplo Final - Visão para mostrar Fornece_para com nomes no lugar dos números:

```
CREATE VIEW Fornece_para_Nomes AS
SELECT PeNome, FNome, PNome, Quant
FROM Peca, Fornecedor, Projeto, Fornece_para
WHERE Peca.PeNro = Fornece_para.PeNro
      AND Fornecedor.FNro = Fornece_para.FNro
      AND Projeto.PNro = Fornece_para.PNro;
```