



UNIVERSIDADE
FEDERAL DO CEARÁ
Campus Russas

Disciplina: Fundamentos de Banco de Dados

14. Visões em SQL

Professora: Marília S. Mendes

E-mail: marilia.mendes@ufc.br

Organização da aula

- ▶ **Conceito de visões**
 - ▶ Especificação de visões em SQL
 - ▶ Implementação de visões
 - ▶ Atualização através de visões
-

Conceito de visão em SQL

Visão em SQL NÃO É:

Visão externa (aula 4)

Visão externa pode
incluir várias tabelas



Conceito de visão em SQL

Visão em SQL
Tabela única derivada
de outra **tabela**

tabela base

outra visão



Visão – Tabela virtual

- ▶ Uma visão não existe fisicamente
 - ▶ É considerada uma **tabela virtual**

Visão – Tabela virtual

⇒ **Tabela básica**: tuplas são realmente armazenadas no banco de dados

X

⇒ **Tabela virtual (visão)**: tuplas são derivadas de outras tuplas - não são armazenadas fisicamente

Visão – Tabela virtual

Consequências

- ⇒ Consultas podem ser feitas sem problemas
- ⇒ Atualizações são **limitadas**



Uso de visões

- ➡ Simplificar a especificação de consultas freqüentes
- ➡ Exemplo consulta freqüente: recuperar os nomes dos empregados e os projetos nos quais eles trabalham
 - ➡ Exige 2 junções: EMPREGADO, PROJETO, TRABALHA_EM

Exemplo

```
SELECT PNAME, UNOME, PJNAME, HORAS  
FROM EMPREGADO, PROJETO, TRABALHA_EM  
WHERE SSN=ESSN AND PNO=PNUMERO
```

➡ Se esta tabela fosse definida como uma visão
EMPREGADO_PROJETO, a consulta seria bem mais simples,
sem junções:

```
SELECT * FROM EMPREGADO_PROJETO
```



Uso de Visões

- ⇒ Auxílio aos mecanismos de autorização e segurança
- ⇒ Exemplo: a empresa poderia não querer que os funcionários tivessem acesso aos salários dos empregados
 - ⇒ Solução: visão que contém todos os dados da tabela EMPREGADO, exceto o salário



Organização da aula

- ▶ Conceito de visões
 - ▶ **Especificação de visões em SQL**
 - ▶ Implementação de visões
 - ▶ Atualização através de visões
-

Especificando Visões

➡ Comando CREATE VIEW

➡ Sintaxe:

```
CREATE VIEW <NOME_VISAO>  
AS <CONSULTA>
```



Exemplo

```
CREATE VIEW EMPREGADO_PROJETO  
AS SELECT PNAME, UNOME, PJNAME, HORAS  
FROM EMPREGADO, PROJETO, TRABALHA_EM  
WHERE SSN=ESSN AND PNO=PNUMERO
```

	PNAME	UNOME	PJNAME	HORAS



Exemplo

select * from empregado_projeto

pnome character varying(30)	unome character varying(30)	pjnome character varying(30)	horas double precision
John	Smith	ProdutoX	32.5
John	Smith	ProdutoY	7.5
Ramesh	Narayan	ProdutoZ	40
Joyce	English	ProdutoX	20
Joyce	English	ProdutoY	20
Franklin	Wong	ProdutoY	10
Franklin	Wong	ProdutoZ	10
Franklin	Wong	Automatização	10
Franklin	Wong	Reorganização	10
Alicia	Zelaya	Novos Benefícios	30
Alicia	Zelaya	Automatização	10
Ahmad	Jabbar	Automatização	35
Ahmad	Jabbar	Novos Benefícios	5
Jennifer	Wallace	Novos Benefícios	20
Jennifer	Wallace	Reorganização	15
James	Borg	Reorganização	

Tuplas retornadas pela consulta usada na definição da visão

Renomeando colunas

➡ É possível especificar os nomes das colunas da visão

➡ Sintaxe:

```
CREATE VIEW <NOME_VISAO> (<COLUNAS>)  
AS <CONSULTA>
```



Exemplo

```
CREATE VIEW DEPT_INFO (DEPT_NOME, NO_EMPS, TOTAL_SAL)
AS SELECT DNAME, COUNT(*), SUM(SALARIO)
FROM DEPARTAMENTO, EMPREGADO
WHERE DNUMERO=DNO
GROUP BY DNAME
```

	DEPT_NOME	NO_EMPS	TOTAL_SAL
►	Administração	3	93000
	Pesquisa	4	133000
	Sede administrativa	1	55000



Consultas



Visões podem ser consultadas, assim como tabelas base



Exemplo 1



Selecionar os nomes dos empregados que trabalham no Projeto "ProdutoX"

	PNAME	UNOME
▶	John	Smith
	Joyce	English



Exemplo 2



Selecionar os departamentos cuja folha de pagamento excede 90000

	DEPT_NOME	TOTAL_SAL
▶	Administração	93000
	Pesquisa	133000



Atualização

➡ SGBD garante que visão está sempre atualizada

- ▬ Qualquer alteração nas tuplas das tabelas base são refletidas na visão



Exemplo

Tabela TRABALHA_EM

🔑 ESN	🔑 PNO	HORAS
123456789	1	32.5
123456789	2	7.5
333445555	2	10
333445555	3	10
333445555	10	10
333445555	20	10
453453453	1	20
453453453	2	20
666884444	3	40
888665555	20	
987654321	10	5
987654321	20	15
987654321	30	20
987987987	10	35
987987987	30	5
999887777	10	10
999887777	30	30

NOVA tupla

Exemplo

Tabela TRABALHA_EM

🔑 ESN	🔑 PNO	HORAS
123456789	1	32.5
123456789	2	7.5
333445555	2	10
333445555	3	10
333445555	10	10
333445555	20	10
453453453	1	20
453453453	2	20
666884444	3	40
888665555	20	
987654321	10	5
987654321	20	15
987654321	30	20
987987987	10	35
987987987	30	5
999887777	10	10
999887777	30	30

NOVA tupla

Visão EMPREGADO_PROJETO

PNAME	UNOME	PJNAME	HORAS
John	Smith	ProdutoX	32.5
Joyce	English	ProdutoX	20
John	Smith	ProdutoY	7.5
Franklin	Wong	ProdutoY	10
Joyce	English	ProdutoY	20
Franklin	Wong	ProdutoZ	10
Ramesh	Narayan	ProdutoZ	40
Franklin	Wong	Automatização	10
Jennifer	Wallace	Automatização	5
Ahmad	Jabbar	Automatização	35
Alicia	Zelaya	Automatização	10
Franklin	Wong	Reorganização	10
James	Borg	Reorganização	
Jennifer	Wallace	Reorganização	15
Jennifer	Wallace	Novos Benefícios	20
Ahmad	Jabbar	Novos Benefícios	5
Alicia	Zelaya	Novos Benefícios	30

Atualização




Consequência:

- visão **não é** montada no instante de sua **definição**, mas sim no momento em que alguma **consulta** é realizada sobre ela
- Em outras palavras:
 - Normalmente, apenas o esquema da visão é armazenado
 - Tuplas são calculadas no momento da consulta
 - Falaremos mais sobre isso ...



Exclusão da visão

 Quando uma visão não é mais necessária

DROP VIEW TRABALHA_EM



Organização da aula

- ▶ Conceito de visões
 - ▶ Especificação de visões em SQL
 - ▶ **Implementação de visões**
 - ▶ Atualização através de visões
-

Implementação de Visões

⇒ Problema complexo, bastante estudado

⇒ Duas abordagens principais:

- ⇒ Modificação da consulta
- ⇒ Materialização da visão



Modificação de coluna

Consulta sobre a visão



Consulta sobre as tabelas base

Exemplo

```
CREATE VIEW EMPREGADO_PROJETO  
AS SELECT PNAME, UNOME, PJNAME, HORAS  
   FROM EMPREGADO, PROJETO, TRABALHA_EM  
  WHERE SSN=ESSN AND PNO=PNUMERO
```

Visão

```
SELECT PNAME, UNOME  
FROM EMPREGADO_PROJETO  
WHERE PJNAME="ProdutoX"
```

Consulta do usuário

```
SELECT PNAME, UNOME  
FROM EMPREGADO, PROJETO, TRABALHA_EM  
WHERE SSN=ESSN AND PNO=PNUMERO  
AND PJNAME="ProdutoX"
```

Consulta
Reescrita

Exemplo

```
CREATE VIEW EMPREGADO_PROJETO  
AS SELECT PNAME, UNOME, PJNAME, HORAS  
FROM EMPREGADO, PROJETO, TRABALHA_EM  
WHERE SSN=ESSN AND PNO=PNUMERO
```

Visão

```
SELECT PNAME, UNOME  
FROM EMPREGADO_PROJETO  
WHERE PJNAME="ProdutoX"
```

Consulta do usuário

```
SELECT PNAME, UNOME  
FROM EMPREGADO, PROJETO, TRABALHA_EM  
WHERE SSN=ESSN AND PNO=PNUMERO  
AND PJNAME="ProdutoX"
```

Consulta
Reescrita

Exemplo

```
CREATE VIEW EMPREGADO_PROJETO  
AS SELECT PNAME, UNOME, PJNAME, HORAS  
FROM EMPREGADO, PROJETO, TRABALHA_EM  
WHERE SSN=ESSN AND PNO=PNUMERO
```

Visão

```
SELECT PNAME, UNOME  
FROM EMPREGADO_PROJETO  
WHERE PJNAME="ProdutoX"
```

Consulta do usuário

```
SELECT PNAME, UNOME  
FROM EMPREGADO, PROJETO, TRABALHA_EM  
WHERE SSN=ESSN AND PNO=PNUMERO  
AND PJNAME="ProdutoX"
```

Consulta
Reescrita

Exemplo

```
CREATE VIEW EMPREGADO_PROJETO  
AS SELECT PNAME, UNOME, PJNAME, HORAS  
FROM EMPREGADO, PROJETO, TRABALHA_EM  
WHERE SSN=ESSN AND PNO=PNUMERO
```

Visão

```
SELECT PNAME, UNOME  
FROM EMPREGADO_PROJETO  
WHERE PJNAME="ProdutoX"
```

Consulta do usuário

```
SELECT PNAME, UNOME  
FROM EMPREGADO, PROJETO, TRABALHA_EM  
WHERE SSN=ESSN AND PNO=PNUMERO  
AND PJNAME="ProdutoX"
```

Consulta
Reescrita

Modificação da consulta



Desvantagens:

- Ineficiente para visões definidas via consultas complexas
- Exemplo: visão que envolve muitas junções
 - Junções têm que ser realizadas a cada vez que a visão é consultada



Materialização de visão

- ➡ Visão é mantida fisicamente, em uma tabela temporária
 - ➡ Não há mais necessidade de composição de consultas
- ➡ Problema: manter a visão atualizada
 - ➡ Atualização incremental

Atualização incremental

- ➡ Técnicas que auxiliam na manutenção da visão
 - Dizem quais tuplas precisam ser inseridas, alteradas e excluídas da visão, de acordo com alterações feitas nas tabelas base
- ➡ Normalmente visões são mantidas materializadas apenas pelo tempo necessário para a realização de um conjunto de consultas.
- ➡ Depois são descartadas, e reconstruídas quando necessário



Organização da aula

- ▶ Conceito de visões
 - ▶ Especificação de visões em SQL
 - ▶ Implementação de visões
 - ▶ **Atualização através de visões**
-

Atualização de visões





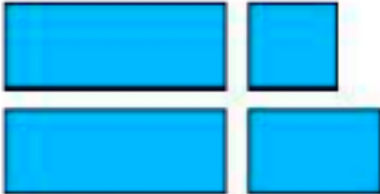
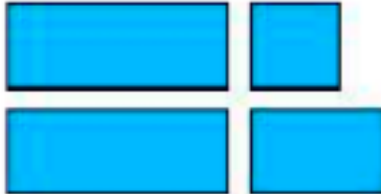
Alterar a visão:

- Inserir, modificar ou excluir tuplas da visão
- Atualizações têm que ser refletidas nas tabelas base



Atualização de visões



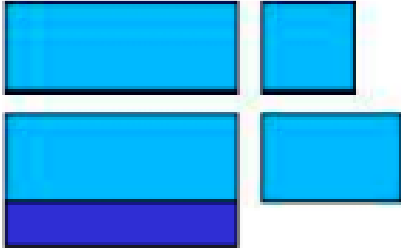
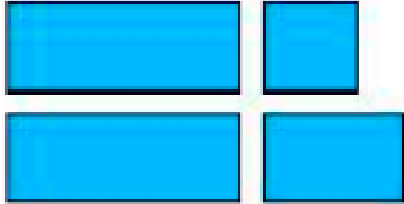
➡ Problemas opostos:

Visões		
Tabelas Base		



Atualização de visões

➡ Problemas opostos:

Visões		
Tabelas Base		

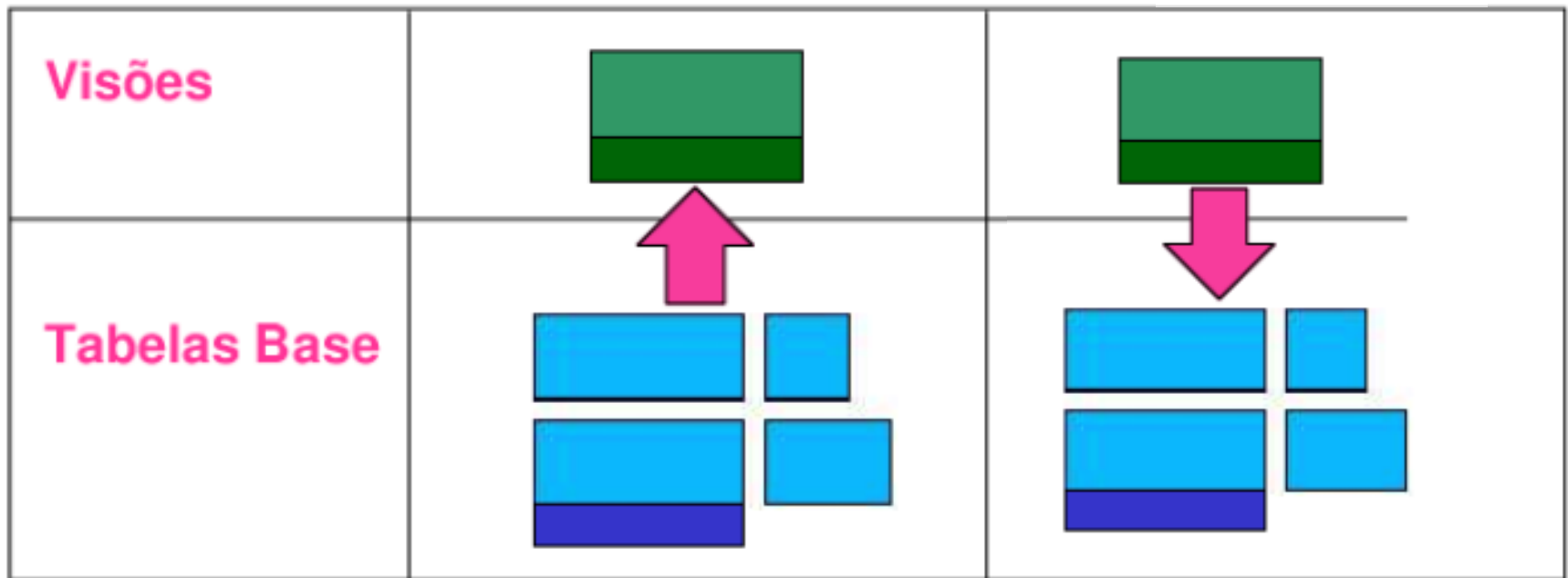


Atualização de visões

➡ Problemas opostos:

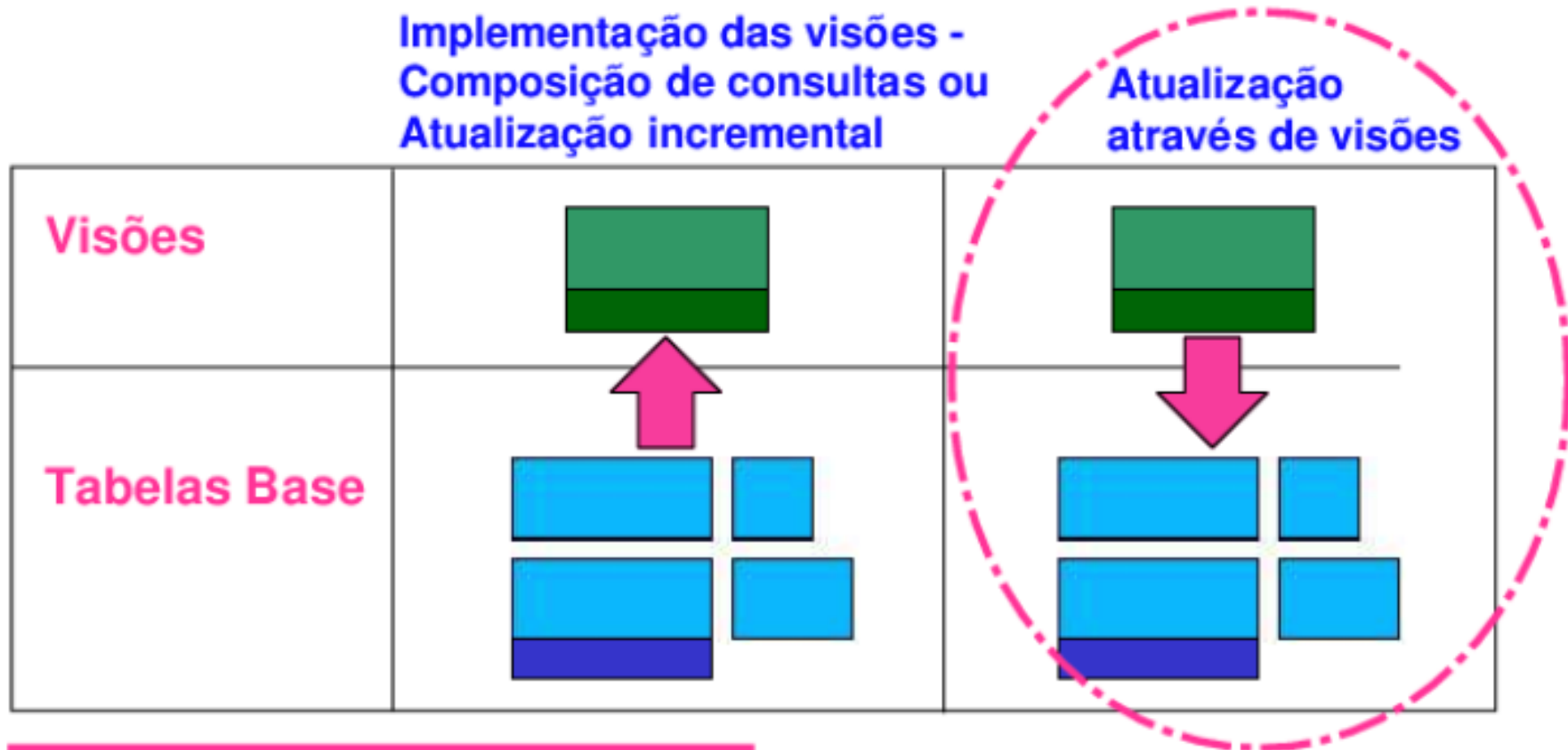
Implementação das visões -
Composição de consultas ou
Atualização incremental

Atualização
através de visões



Atualização de visões

➡ Problemas opostos:



**Problema bem mais complexo:
pode haver ambiguidade**

Exemplo

➡ Atualização na visão EMPREGADO_PROJETO

```
UPDATE EMPREGADO_PROJETO  
SET PJNOME="ProdutoY"  
WHERE UNOME="Smith" AND PNOME="John" AND  
      PJNOME="ProdutoX"
```

➡ Este comando pode ser mapeado para as tabelas base de diversas formas

Possível mapeamento (A)

```
UPDATE EMPREGADO_PROJETO  
SET PJNOME="ProdutoY"  
WHERE UNOME="Smith" AND PNOME="John" AND  
    PJNOME="ProdutoX"
```

```
UPDATE TRABALHA_EM  
SET PNO= (SELECT PNUMERO FROM PROJETO  
          WHERE PJNOME="ProdutoY")  
WHERE ESSN IN (SELECT SSN FROM EMPREGADO  
               WHERE UNOME="Smith" AND PNOME="John")  
AND PNO = (SELECT PNUMERO FROM PROJETO  
           WHERE PJNOME="ProdutoX")
```



Outro possível mapeamento (B)

```
UPDATE EMPREGADO_PROJETO  
SET PJNOME="ProdutoY"  
WHERE UNOME="Smith" AND PNOME="John" AND  
    PJNOME="ProdutoX"
```

```
UPDATE PROJETO  
SET PJNOME="ProdutoY"  
WHERE PJNOME="ProdutoX"
```



Existem outros

➡ Qual deles é o desejado?

➡ Provavelmente não é o mapeamento B, que troca o nome do Projeto ProdutoX para ProdutoY

➡ Como decidir?

➡ Tomar esta decisão automaticamente é uma tarefa complexa



Algumas atualizações podem não fazer sentido

```
UPDATE DEPT_INFO  
SET TOTAL_SAL=100000  
WHERE DNOME="Pesquisa"
```

➡ TOTAL_SAL é um campo calculado

➡ Diversas atualizações na base de dados podem resultar em
TOTAL_SAL = 100000

Então, quando é possível atualizar a visão?

- ➡ A princípio, apenas quando houver uma única tradução possível
 - Exemplo: quando a visão é definida sobre uma tabela base apenas, e existe um mapeamento 1:1 entre tuplas da visão e tuplas da tabela base

Então, quando é possível atualizar a visão?

➡ Quando existe mais de uma tradução possível, certos cuidados são necessários

- ➡ Pesquisadores vêm estudando o problema e propondo soluções
- ➡ Métodos que escolhem as atualizações mais prováveis
- ➡ Métodos que deixam a decisão a cargo do usuário



Em geral

- ➡ Visões com junções de várias tabelas base não são atualizáveis
- ➡ Visões de uma única tabela são atualizáveis se
 - ➡ a chave primária estiver presente
 - ➡ os campos especificados com NOT NULL e que não tiverem valores *default* estiverem presentes
- ➡ Visões que contenham agregações não são atualizáveis

View x Materialized View

- ▶ **Visão Materializada** é uma view, só que neste caso, o que é armazenado não é a consulta e sim o resultado dela.
- ▶ **MATERIALIZED VIEW** é uma tabela real no banco de dados que é atualizada SEMPRE que ocorrer uma atualização em alguma tabela usada pela sua consulta. Por este motivo, no momento em que o usuário faz uma consulta nesta visão materializada o resultado será mais rápido que se ela não fosse materializada.

A diferença entre elas pode se descrita desta forma:

- a view realiza a consulta no momento que o usuário faz uma consulta nela;
- e
- a *materialized view* realiza a consulta no momento em que uma das tabelas consultadas é atualizada.

Quando usar *VIEW* ou *MATERIALIZED VIEW*?

- ▶ A decisão se a sua *view* deve ser simples ou materializada é tomada com base no tipo de utilização das tabelas usadas pela consulta da *view*.

Você consulta mais na *view* do que altera os dados das tabelas? Os dados do seu banco de dados são alterados com frequência?

você deve usar uma *visão materializada* quando **o desempenho das buscas na *view* é mais importante que o desempenho da escrita nas tabelas que ela utiliza**. Mas se uma tabela utilizada pela *view* tem muita alteração de dados, talvez seja mais interessante que a *view* não seja materializada.

visões materializadas

```
CREATE MATERIALIZED VIEW Funcionarias_emp  
AS SELECT pnome, unome FROM empregado  
where sexo='F' ORDER BY pnome;
```

```
REFRESH MATERIALIZED VIEW nome_view;
```

```
DROP MATERIALIZED VIEW nome_view;
```

```
SELECT * FROM PG_MATVIEWS;
```

```
--Saber quantas visões você tem
```

```
ALTER MATERIALIZED VIEW Funcionarias  
RENAME TO func;
```

Exercício 1

- ➡ Criar uma visão sobre a tabela EMPREGADO, que contenha apenas o SSN, UNOME e PNAME
- ➡ Inserir tuplas na visão e checar o resultado, consultando a visão e a tabela base EMPREGADO
- ➡ A inserção foi mapeada para a tabela base?



Exercício 2

- 1) Crie uma visão que mostre os funcionários (pnome, unome, salario) do departamento Pesquisa.
 - 2) Insira uma tupla nesta visão e verifique o resultado.
 - 3) O que aconteceu? Justifique.
-

Condições para uma visão ser 'atualizável'

- ▶ A cláusula **from** tem apenas uma relação do banco de dados;
 - ▶ A cláusula **select** contém apenas nomes de atributos da relação e não possui quaisquer expressões, agregações ou especificação **distinct**;
 - ▶ Qualquer atributo não listado na cláusula **select** pode ser definido como null; ou seja, ele não tem uma restrição **not null** e não faz parte de uma chave primária;
 - ▶ A consulta não tem uma cláusula **group by** ou **having**.
-

Bibliografia Utilizada nesta aula

- ▶ ELMASRI, R.; NAVATHE, S. B. Sistemas de banco de dados. 6 ed. Pearson/Addison-Wesley, 2011. ISBN: 9788579360855
- ▶ Silberschatz, A., Korth, H., Sudarshan, S. Sistema de Banco de Dados. 5ª Edição, Editora Campus, 2006.
- ▶ Ramakrishnan, R. Sistemas de Gerenciamento de Banco de Dados, 3ª Edição, McGraw-Hill, 2008.

Slides parcialmente adaptados das
Professoras: Marta Mattoso (UFRJ) e Vanessa Braganholo (UFF)
