

MAC0439 - Laboratório de Bancos de Dados

Aula 12 **Consultas em SQL**

Consultas envolvendo múltiplas relações

25 de setembro de 2015
Profa. Kelly Rosa Braghetto

(Adaptação dos slides do prof. Jeffrey Ullman, da *Stanford University*)

Exemplo para a aula

- ◆ Todas as nossas consultas SQL serão baseadas no seguinte esquema de BD:

Refrigerante(nome, fabricante)

Lanchonete(nome, endereco, cnpj)

Cliente(nome, endereco, telefone)

Apreciador(nome_cliente, nome_refri)

Vendedor(nome_lanch, nome_refri, preco)

Frequentador(nome_cliente, nome_lanch)

Consultas envolvendo múltiplas relações

- ◆ Consultas interessantes frequentemente combinam dados de mais de uma relação.
- ◆ Podemos considerar várias relações em uma consulta listando-as na cláusula FROM.
- ◆ Para distinguir atributos de relações diferentes que possuem o mesmo nome: “<relação>.<atributo>” .

Exemplo: junção de duas relações

- ◆ Usando a relação `Apreciador(nome_cliente, nome_refri)` e `Frequentador(nome_cliente, nome_lanch)`, encontre os refrigerantes apreciados por pelo menos uma pessoa que frequenta a lanchonete Sujinhos.

```
SELECT nome_refri
FROM Appreciador, Frequentador
WHERE nome_lanch = 'Sujinhos' AND
Frequentador.nome_cliente =
Appreciador.nome_cliente;
```

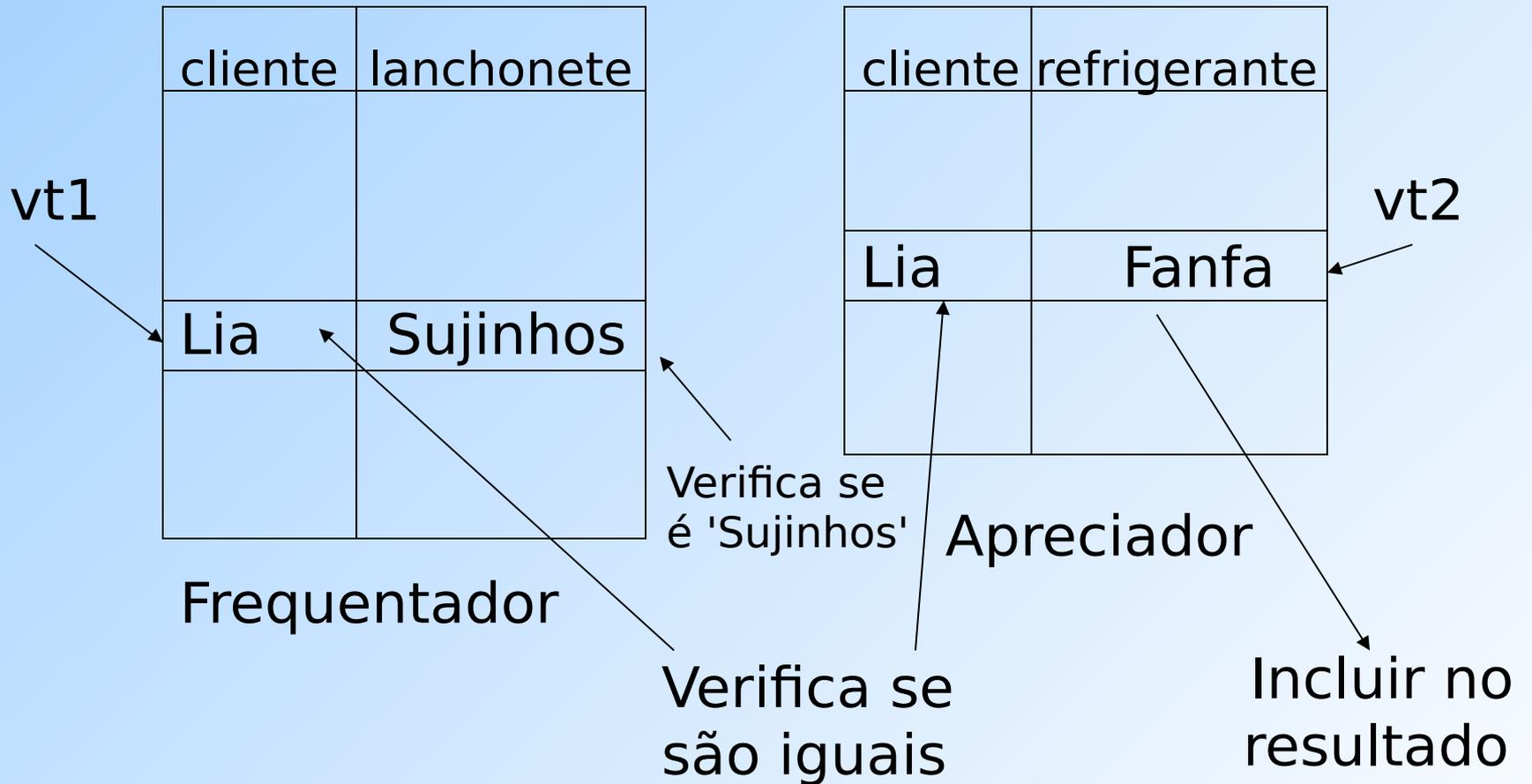
Semântica formal

- ◆ Quase a mesma que a das consultas sobre uma única relação:
 1. Comece com o produto cartesiano de todas as relações da cláusula FROM.
 2. Aplique a condição de seleção da cláusula WHERE.
 3. Projete sobre a lista de atributos e expressões da cláusula SELECT.

Semântica operacional

- ◆ Imagine uma variável-tupla para cada relação na cláusula FROM.
 - ◆ Essas variáveis visitam cada combinação possível de tuplas, uma de cada relação.
- ◆ Se as variáveis-tuplas apontam para tuplas que satisfazem a cláusula WHERE, envie essas tuplas para a cláusula SELECT.

Exemplo



Variáveis-tuplas explícitas

- ◆ Às vezes, uma consulta precisa usar duas cópias de uma mesma relação.
- ◆ Para diferenciar as cópias, acrescenta-se o nome de uma variável-tupla na frente do nome da relação na cláusula FROM.
- ◆ É sempre possível renomear uma relação desta forma (mesmo quando isso não é indispensável para a consulta).

Exemplo: auto-junção

- ◆ A partir de **Refrigerante(nome_refri, fabricante)**, encontre todos os pares de refri feitos por um mesmo fabricante.
 - ▶ Não produza pares como (Fanfa, Fanfa).
 - ▶ Produza pares em ordem alfabética, p.e., (Fanfa, Sprife), mas não (Sprife, Fanfa).

```
SELECT r1.nome, r2.nome
FROM Refrigerante r1, Refrigerante r2
WHERE r1.fabricante = r2.fabricante
      AND r1.nome < r2.nome;
```

Subconsultas

- ◆ Um comando SELECT-FROM-WHERE parentizado (= *subconsulta*) pode ser usado como um valor em diferentes lugares de uma consulta, incluindo nas cláusulas FROM e WHERE.
- ◆ **Exemplo:** no lugar de uma relação na cláusula FROM, nós podemos usar uma subconsulta, e então consultar o seu resultado.
 - ▶ Para isso, faz-se necessário o uso de uma variável-tupla para nomear as tuplas do resultado.

Exemplo: uma subconsulta no FROM

- ◆ Encontre os refrigerantes apreciados por pelo menos uma pessoa que frequenta o Sujinhos.

```
SELECT nome_refri
FROM Apreciador, (SELECT nome_cliente
                  FROM Frequentador WHERE
                  nome_lanch = 'Sujinhos') SC
WHERE Apreciador.nome_cliente =
      SC.nome_cliente;
```

Clientes que frequentam o Sujinhos

Subconsultas que devolvem uma tupla

- ◆ Se uma subconsulta garantidamente produz uma única tupla, então a subconsulta pode ser usada como um valor.
 - ▶ Geralmente, a tupla tem um único componente.
 - ▶ Um erro é gerado em tempo de execução se não há nenhuma tupla no resultado ou se o resultado contém mais do que uma tupla.

Exemplo: subconsulta com tupla única

- ◆ Usando `Venda(nome_lanch, nome_refri, preco)`, encontre as lanchonetes que servem Fanfa pelo mesmo preço que o Sujinhos cobra pela Sprife.
- ◆ A combinação de duas consultas certamente resolve a questão:
 1. Encontre o preço da Sprife no Sujinhos.
 2. Encontre as lanchonetes que vendem Fanfa por esse preço.

Solução com consulta + subconsulta

```
SELECT nome_lanch  
FROM Venda  
WHERE nome_refri = 'Fanfa' AND  
preco = (SELECT preco
```

```
FROM Venda  
WHERE nome_lanch = 'Sujinhos'  
AND nome_refri = 'Sprife');
```

Preço da
Sprife no
Sujinhos



O operador IN

- ◆ A expressão

`<tupla> IN (<subconsulta>)`

é verdadeira se e somente se a tupla é membro da relação produzida pela subconsulta.

- ◆ Oposto:

`<tupla> NOT IN (<subconsulta>).`

- ◆ Expressões com IN podem aparecer na cláusula WHERE.

Exemplo: IN

- ◆ Usando `Refrigerante(nome, fabricante)` e `Apreciador(nome_cliente, nome_refri)`, encontre o nome e o fabricante de cada refri que o Fred gosta.

```
SELECT *
```

```
FROM Refrigerante
```

```
WHERE nome IN (SELECT nome_refri  
FROM Appreciador  
WHERE nome_cliente = 'Fred');
```

Conjunto de
refris que o
Fred gosta



Estas consultas são equivalentes?

```
SELECT a
FROM R, S
WHERE R.b = S.b;
```

a	b
1	2
3	4

R

b	c
2	5
2	6

S

```
SELECT a
FROM R
WHERE b IN (SELECT b FROM S);
```

IN é um predicado sobre as tuplas de R

```
SELECT a
FROM R
WHERE b IN (SELECT b FROM S);
```

Laço sobre as tuplas de S

Laço sobre as tuplas de R

a	b
1	2
3	4

R

b	c
2	5
2	6

S

(1,2) satisfaz
a condição;
1 aparece uma
vez no resultado.

Esta consulta “pareia” tuplas de R e S

```
SELECT a  
FROM R, S  
WHERE R.b = S.b;
```

Laço duplo sobre
as tuplas de R e S

a	b
1	2
3	4

R

b	c
2	5
2	6

S

(1,2) com (2,5) e
(1,2) com (2,6) -
ambos pares
satisfazem a
condição; 1 é
incluído na
resposta 2 vezes!

O operador EXISTS

- ◆ A expressão EXISTS(<subconsulta>) é verdadeira se e somente se o resultado da subconsulta não é vazio.
- ◆ **Exemplo:** A partir de **Refrigerante(nome, fabricante)**, encontre os refris que são os únicos fabricados por seus fabricantes.

Exemplo: EXISTS

```
SELECT nome  
FROM Refrigerante r1  
WHERE NOT EXISTS (
```

Observe a regra do escopo:
fabricante se refere à
relação na cláusula FROM
mais próxima que possua
o atributo.

Cjto de refris
com o mesmo
fabricante de
r1, mas que
não é o mesmo refri.

```
SELECT *  
FROM Refrigerante  
WHERE fabricante =  
r1.fabricante AND  
nome <> r1.nome);
```

Operador de
"diferente"
da SQL

O operador ANY

- ◆ $x = \text{ANY}(\langle \text{subconsulta} \rangle)$ é uma condição booleana que é verdadeira sse x é igual a pelo menos uma tupla no resultado do subconjunto.
 - ◆ “=” pode ser substituído por qualquer operador de comparação.
- ◆ **Exemplo:** $x \geq \text{ANY}(\langle \text{subconsulta} \rangle)$ significa que x não é sozinha a menor tupla produzida pela subconsulta.
 - ◆ Observe que as tuplas resultantes na subconsulta precisam possuir um único componente.

O operador ALL

- ◆ $x \langle \rangle \text{ALL}(\langle \text{subconsulta} \rangle)$ é verdadeira sse para toda tupla t no resultado da subconsulta, x não é igual a t .
 - ◆ Ou seja, x não está no resultado da subconsulta.
- ◆ “ $\langle \rangle$ ” pode ser substituído por qualquer operador de comparação.
- ◆ **Exemplo:** $x \geq \text{ALL}(\langle \text{subconsulta} \rangle)$ significa que não há no resultado da subconsulta tupla maior do que x .

Exemplo: ALL

- ◆ A partir de Venda(nome_lanch, nome_refri, preco), encontre o(s) refri(s) vendidos pelo maior preço.

```
SELECT nome_refri
FROM Venda
WHERE preco >= ALL(
    SELECT preco
    FROM Venda);
```

preco de Venda mais "externo" não pode ser menor do que qualquer outro preco.

Referências Bibliográficas

- ◆ *Database Systems - The Complete Book*, Garcia-Molina, Ullman e Widom. 2002.
Capítulo 6
- ◆ *Sistemas de Bancos de Dados (6ª edição)*, Elmasri e Navathe. 2010.
Capítulos 4 e 5