

MAC0439 - Laboratório de Bancos de Dados

Aula 10

Consultas Básicas em SQL

14 de setembro de 2016

Prof^a Kelly Rosa Braghetto

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

Exemplo para a aula

- ◆ As consultas SQL serão baseadas no seguinte esquema relacional 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)

Comandos

Select-From-Where

SELECT <lista de atributos>
FROM <lista de tabelas>
WHERE <condição>

Exemplo

- ◆ Usando **Refrigerante(nome, fabricante)**, quais “refris” são feitos por *Cola-Coca* ?

```
SELECT nome
```

```
FROM Refrigerante
```

```
WHERE fabricante = 'Cola-Coca';
```

Resultado da consulta

nome
Fanfa
Kuaif
Sprife
...

A resposta é uma relação com um único atributo, **nome**, e tuplas com o nome de cada refrigerante produzido pela Cola-Coca.

Semântica operacional - visão geral

- ◆ Processamento de uma consulta:
 - ▶ Considere que há uma *variável-tupla* percorrendo cada tupla da relação mencionada na cláusula FROM.
 - ▶ Verifique se a tupla “atual” satisfaz a cláusula WHERE.
 - ▶ Se sim, compute os atributos ou expressões da cláusula SELECT usando os componentes dessa tupla.

Semântica operacional

nome	fabricante
Fanfa	Cola-Cola

Verifica se é
Cola-Coca

Se sim, inclui
t.nome no resultado

A variável-tupla t
percorre todas as
tuplas

```
SELECT nome  
FROM Refrigerante  
WHERE fabricante = 'Cola-Coca';
```

O * em cláusulas SELECT

- ◆ Quando há apenas uma relação na cláusula FROM, um * na cláusula SELECT equivale a “todos os atributos dessa relação”.

- ◆ Exemplo:

Usando Refrigerante(nome, fabricante)

```
SELECT *  
FROM Refrigerante  
WHERE fabricante = 'Cola-Coca';
```

Resultado da consulta

nome	fabricante
Fanfa	Cola-Coca
Kuaif	Cola-Coca
Sprife	Cola-Coca
.

Agora, o resultado possui todos os atributos de Refrigerante.

Renomeando atributos

- ◆ Para modificar os nomes dos atributos no resultado, use “AS <novo nome>” para renomear um atributo.

- ◆ **Exemplo:**

usando **Refrigerante(nome, fabricante)**

```
SELECT nome AS refri, fabricante  
FROM Refrigerante  
WHERE fabricante = 'Cola-Coca'
```

Resultado da consulta

refri	fabricante
Fanfa	Cola-Coca
Kuaif	Cola-Coca
Sprife	Cola-Coca
...	...

Expressões em cláusulas SELECT

◆ Qualquer expressão que faça sentido pode aparecer como um elemento na cláusula SELECT.

◆ **Exemplo:**

Usando `Venda(nome_lanch, nome_refri, preco)`

```
SELECT nome_lanch, nome_refri,  
       preço*30.43 AS preco_em_yen  
FROM Venda;
```

Resultado da consulta

nome_lanch	nome_refri	preco_em_yen
Sujinhos	Fanfa	155
Bar do Zé	Sprife	142
...

Exemplo: Constantes como expressões

Usando `Apreciador(nome_cliente, nome_refri)`:

```
SELECT nome_cliente,  
       'aprecia Fanfa' AS  
descricao  
FROM Appreciador  
WHERE nome_refri = 'Fanfa';
```

Resultado da consulta

cliente	descricao
Sally	aprecia Fanfa
Fred	aprecia Fanfa
...	...

Condições complexas para a cláusula WHERE

- ◆ Operadores booleanos AND, OR, NOT.
- ◆ Comparações =, <>, <, >, <=, >=.
- ◆ E muitos outros operadores que produzem valores booleanos como resultado.

Exemplo: Condição complexa

- ◆ Usando `Venda(nome_lanch, nome_refri, preco)`, encontre o preço cobrado no Sujinhos pela Fanfa:

```
SELECT preco
```

```
FROM Venda
```

```
WHERE nome_lanch = 'Sujinhos'
```

```
      AND nome_refri = 'Fanfa';
```

Padrões

- ◆ Uma condição pode comparar uma *string* com um padrão (~ expressão regular) usando:
 - ▶ <Atributo> **LIKE** <padrão> ou <Atributo> **NOT LIKE** <padrão>
- ◆ *Padrão* é uma *string* contendo caracteres especiais:
 - ▶ '%' = “casa” com qualquer *string*
 - ▶ '_' = “casa” com qualquer (1) caractere

Exemplo: LIKE

◆ Usando

Cliente(nome, endereço, telefone),
encontre os clientes com DDD de São
Paulo:

```
SELECT nome  
FROM Cliente  
WHERE telefone LIKE '(11)%';
```

Exemplo(2): LIKE

◆ Usando

Cliente(nome, endereço, telefone),
encontre os clientes cujo primeiro nome
tem 3 letras:

```
SELECT nome  
FROM Cliente  
WHERE nome LIKE '_____%';
```

Caracteres especiais em expressões com o LIKE

- ◆ Para usar '%' ou o '_' em um padrão sem que eles exerçam a função de caractere especial, é preciso fazer o “*scape*” deles.
- ◆ O SQL nos permite usar qualquer caractere como *scape*.
- ◆ **Exemplo:** padrão que “casa” o valor do atributo *s* com uma *string* iniciada e finalizada por '%'

```
s LIKE 'x%%x%' ESCAPE 'x'
```

Comparação de strings, datas e horários

- ◆ Também podemos usar os operadores $>$, $>=$, $<$ e $<=$ para comparar *strings*, datas e horários
- ◆ Quando comparamos *strings* com o $<$, por exemplo, estamos perguntando se uma *string* precede a outra na ordem lexicográfica
- ◆ Exemplos:
'facada' $<$ 'farpa' e 'bar' $<$ 'barganha'

Comparando NULL com outros valores

- ◆ A lógica das condições em SQL é uma lógica ternária: **TRUE, FALSE, UNKNOWN**.
- ◆ Comparar qualquer valor (incluindo o próprio NULL) com NULL resulta em **UNKNOWN**.
- ◆ Uma tupla é incluída no conjunto resposta de uma consulta se e somente se a cláusula WHERE é **TRUE** (não pode ser FALSE nem UNKNOWN).

Ordenação do resultado de uma consulta

- ◆ É possível ordenar as tuplas da relação resultante de uma consulta por meio da cláusula

ORDER BY <lista de atributos> [ASC | DESC]

- ◆ A ordenação ascendente (ASC) é a padrão

- ◆ Exemplos:

```
SELECT * FROM Cliente ORDER BY  
                                nome, telefone;
```

ou

```
SELECT * FROM Cliente ORDER BY nome DESC;
```

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

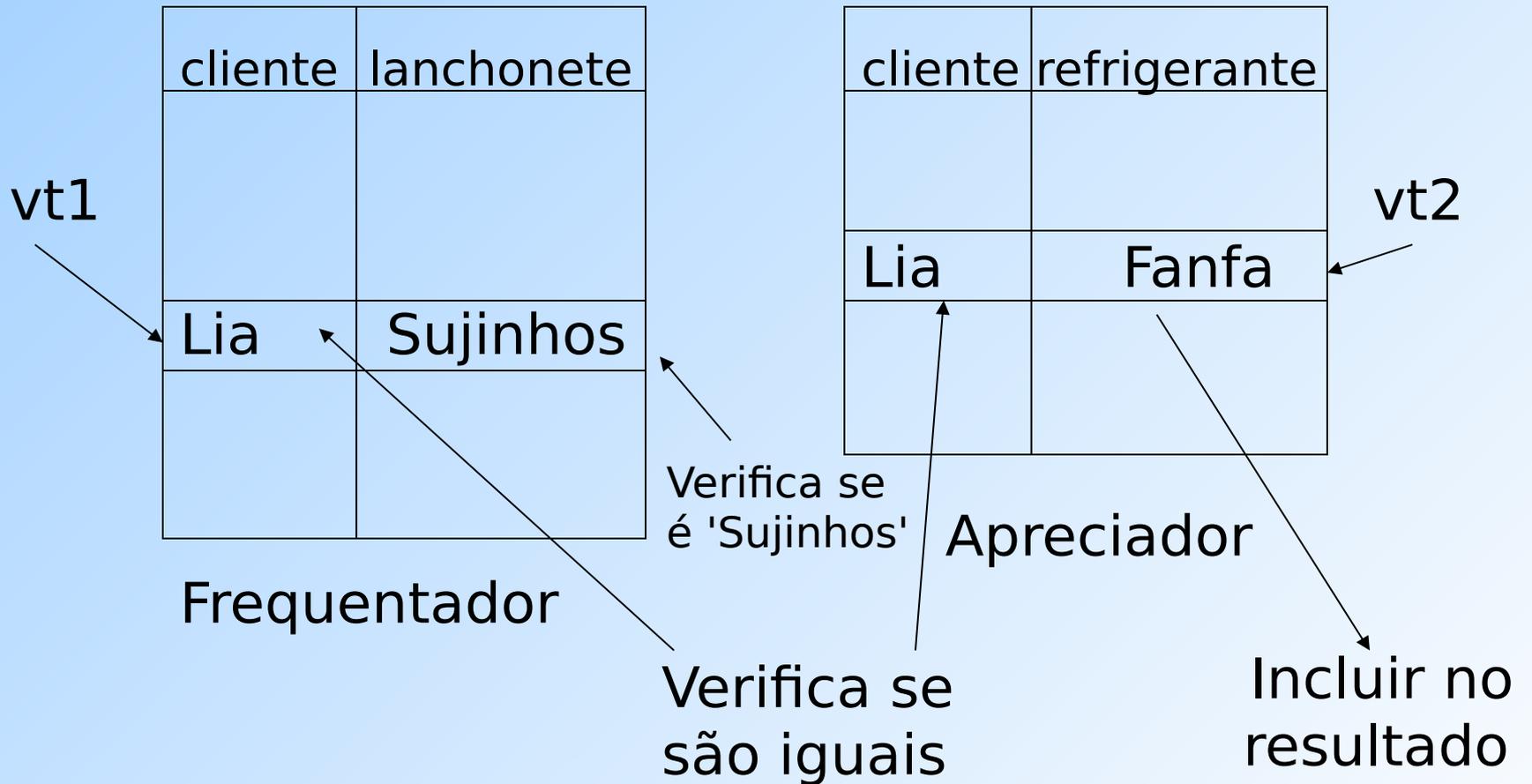
- ◆ Usando a relação `Apreciador(nome_cliente, nome_refri)` e `Frequentador(nome_cliente, nome_lanch)`, encontre os refrigerantes apreciados por pelo menos uma pessoa que frequenta a lanchonete Sujinhos.

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

Semântica 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;
```

União, Intersecção e Diferença

- ◆ União, intersecção e diferença de relações são expressas nas seguintes formas, todas envolvendo subconsultas:
 - ▶ (<subconsulta>) **UNION** (<subconsulta>)
 - ▶ (<subconsulta>) **INTERSECT** (<subconsulta>)
 - ▶ (<subconsulta>) **EXCEPT** (<subconsulta>)

Exemplo: Intersecção

◆ Usando

Apreciador(nome_cliente, nome_refri),
Venda(nome_lanch, nome_refri, preco) e
Frequentador(nome_cliente, nome_lanch),
encontre os clientes e refri tais que:

1. O cliente aprecia o refri e
2. O cliente frequenta pelo menos uma lanchonete que vende o refri

“Truque”:
a subconsulta é
uma tabela
armazenada.

Solução

```
(SELECT * FROM Appreciador)
```

```
INTERSECT
```

```
(SELECT nome_cliente, nome_refri  
FROM Venda, Frequentador  
WHERE Frequentador.nome_lanch  
= Venda.nome_lanch
```

```
);
```

Refris vendidos nas
lanchonetes que o
cliente frequenta.

Semântica de multiconjunto

- ◆ Os comandos SELECT-FROM-WHERE usam **semântica de multiconjunto**
 - ▶ A resposta deles pode conter tuplas repetidas
- ◆ Já para as operações de união, intersecção e diferença, o padrão é a **semântica de conjunto**
 - ▶ As **duplicações de tuplas são eliminadas quando a operação é aplicada**

Motivação: eficiência

- ◆ É muito caro eliminar duplicações de uma relação.
- ◆ A operação de projeção considera somente uma tupla por vez (não requer a ordenação das tuplas) – por isso a consulta feita com o comando `SELECT-FROM-WHERE` fica mais eficiente quando não é preciso remover duplicações.
- ◆ Para intersecções e diferenças, é mais eficiente ordenar as relações antes.
 - ◆ Nesse caso, as duplicações já podem ser facilmente eliminadas.

Controlando a eliminação de duplicações

- ◆ Para forçar que o resultado seja um multiconjunto (ou seja, que as duplicações não sejam eliminadas) use a cláusula **ALL**, como em:

(<subconsulta>) **UNION ALL** (<subconsulta>)

(<subconsulta>) **INTERSECT ALL** (<subconsulta>)

(<subconsulta>) **EXCEPT ALL** (<subconsulta>)

Exemplo: ALL

- ◆ Usando as relações `Frequentador(nome_cliente, nome_lanch)` e `Apreciador(nome_cliente, nome_refri)`:

```
(SELECT nome_cliente FROM Frequentador)
```

```
EXCEPT ALL
```

```
(SELECT nome_cliente FROM Appreciador);
```

- ◆ Lista o nome de cada cliente que frequenta um número de lanchonetes que é maior que o número de refri que ele gosta (e cada nome aparece tantas vezes quanto for a diferença entre essas quantidades).

Controlando a eliminação de duplicações

- ◆ É possível forçar que o resultado de uma consulta seja um conjunto (= sem repetições) usando a cláusula **DISTINCT**:

```
SELECT DISTINCT atrib1, atrib2 ...
```

Exemplo: DISTINCT

- ◆ A partir de

`Venda(nome_lanch, nome_refri, preco)`,
encontre todos os diferentes preços
cobrados por refrigerantes:

```
SELECT DISTINCT preco
```

```
FROM Venda;
```

- ◆ Sem o DISTINCT, cada preço poderia ser listado tantas vezes quanto o número de pares (nome_lanch, nome_refri) associados a esse preço na tabela.

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.
Capítulo 3