

MAC0439 - Laboratório de Bancos de Dados

Aula 11 **Consultas em SQL**

Comandos Select-From-Where

23 de setembro de 2015

Prof^a Kelly Rosa Braghetto

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

Comandos

Select-From-Where

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

Exemplo para a aula

- ◆ As 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)

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.

“Processamento” de uma consulta sobre uma única relação

- ◆ Começa com a relação na cláusula FROM.
- ◆ Aplica-se a seleção indicada na cláusula WHERE.
- ◆ Aplica-se a projeção indicada pela cláusula SELECT.

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
...	...

Exemplo: Integração de informação

- ◆ Frequentemente, construímos *data warehouses* (bancos de dados “integradores”) a partir de dados de diferentes *fontes*.
- ◆ Suponha que cada lanchonete tenha o seu próprio BD e uma relação *Menu(nome_refri, preco)*.
- ◆ Para “alimentar” *Venda(nome_lanch, nome_refri, preco)*, precisamos consultar cada lanchonete e inserir o seu nome no resultado.

Integração de informação (2)

- ◆ Por exemplo, para a lanchonete Sujinhos, podemos realizar a seguinte consulta:

```
SELECT 'Sujinhos',  
       nome_refri, preco  
FROM Menu;
```

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) caracter

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 caracter especial, é preciso fazer o “*scape*” deles.
- ◆ O SQL nos permite usar qualquer caracter 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'

Valores NULL

- ◆ Tuplas em relações SQL podem ter o NULL como valor para um ou mais de seus atributos.
- ◆ O significado do NULL depende do contexto. Dois casos comuns:
 - ▶ *Valor desconhecido* – ex.: sabemos que o *Sujinhos* tem um endereço, mas não sabemos qual é.
 - ▶ *Não aplicável* – ex.: o valor do atributo *nome_cônjuge* para uma pessoa solteira.

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).

Lógica ternária (ou *trivalente*)

- ◆ Para entender como o **AND**, **OR** e o **NOT** funcionam na lógica ternária, pense que TRUE = 1, FALSE = 0 e UNKNOWN = $\frac{1}{2}$.
- ◆ AND = MIN; OR = MAX, NOT(x) = $1-x$.
- ◆ Exemplo:

$$\begin{aligned} \text{TRUE AND (FALSE OR NOT(UNKNOWN))} &= \\ \text{MIN(1, MAX(0, (1 - } \frac{1}{2} \text{)))} &= \\ \text{MIN(1, MAX(0, } \frac{1}{2} \text{))} &= \text{MIN(1, } \frac{1}{2} \text{)} = \frac{1}{2}. \end{aligned}$$

Um exemplo surpreendente

- ◆ A partir da relação Venda a seguir:

nome_lanch	nome_refri	preco
Sujinhos	Fanfa	NULL

```
SELECT nome_lanch FROM Venda
WHERE preco < 2.00 OR preco >= 2.00;
```

← UNKNOWN → ← UNKNOWN →
← UNKNOWN →

Resultado: nenhuma tupla é selecionada!

Razão: Leis para a lógica binária != Leis para a lógica ternária

- ◆ Algumas leis comuns, como a comutatividade do AND, valem na lógica ternária.
- ◆ Mas outras **não**, e.g., a *lei do meio excluído*: $p \text{ OR NOT } p = \text{TRUE}$.
- ◆ Quando $p = \text{UNKNOWN}$, o lado esquerdo é $\text{MAX}(\frac{1}{2}, (1 - \frac{1}{2})) = \frac{1}{2} \neq 1$.

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;
```

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