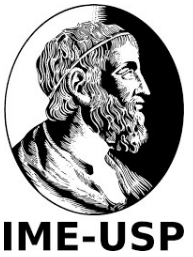
Instituto de Matemática e Estatística
Universidade de São Paulo



Structured Query Language (SQL)
Parte 2

Consultas Simples

Kelly Rosa Braghetto
kellyrb@ime.usp.br

Departamento de Ciência da Computação

Consultas Simples em SQL
Referência

Este material é uma adaptação dos *slides*

do prof. Jeffrey Ullman,

da *Stanford University*

http://infolab.stanford.edu/~ullman/dscb/gslides.html

Exemplo para a aula

- As consultas SQL serão baseadas no seguinte esquema relacional de BD:

Refrigerante(<u>nome</u>, fabricante)
Lanchonete(<u>nome</u>, endereco, cnpj)
Cliente(<u>nome</u>, endereco, telefone)
Apreciador(<u>nome_cliente</u>, <u>nome_refri</u>)
Vendedor(<u>nome_lanch</u>, <u>nome_refri</u>, preco)
Frequentador(<u>nome_cliente</u>, <u>nome_lanch</u>)

Comandos Select-From-Where

SELECT <lista de atributos ou expressões>

FROM <lista de relações>

WHERE <condição>

Exemplo

IME-USP

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

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

Resultado da consulta

IME-USP

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

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 t.fabricante 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

IME-USP

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

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


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

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 as preço_em_real,
       preço*19.21 AS preço_em_iene
FROM Venda;
```

Resultado da consulta


```
SELECT nome_lanch, nome_refri,
       preço as preço_em_real,
       preço*19.21 AS preco_em_iene
FROM Venda;
```

nome_lanch	nome_refri	preço_em_real	preco_em_iene
Sujinhos	Fanfa	6	115.26
Bar do Zé	Sprife	5	96.05
...	...	... ...	

Exemplo:
Constantes como expressões

Usando Apreciador(nome_cliente, nome_refri):

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

Resultado da consulta

IME-USP

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

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:

% (porcentagem) – "casa" com qualquer *string*

_ (*underline*) – "casa" com **um** caractere qualquer

Exemplo: LIKE

IME-USP

- Usando

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

  Exemplos de *strings* de telefone em Cliente:

```
'(11) 91234-5678'
'(15) 5432-5432'
'(11) 3333-4567'
```

Exemplo: LIKE

- Usando

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

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

Exemplo: LIKE


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

```
'(11) 91234-5678' LIKE '(11)%' → TRUE
'(15) 5432-5432' LIKE '(11)%' → FALSE
'(11) 3333-4567' LIKE '(11)%' → TRUE
'11 3333-4567' LIKE '(11)%' → FALSE
'(11)33334567' LIKE '(11)%' → TRUE
```

Exemplo (2): LIKE

IME-USP

- Usando

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

  Exemplos de nomes em Cliente:

```
'Joe Singer'
'Anne Lee Jones'
'Tom'
```

Exemplo (2): LIKE

- Usando

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

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

Exemplo (2): LIKE

IME-USP

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

```
'Joe Singer' LIKE '___ %' → TRUE
'Anne Lee Jones' LIKE '___ %' → FALSE
'Tom' LIKE '___ %' → FALSE
```

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 "escape" deles
- Com a cláusula ESCAPE, a SQL nos permite usar qualquer caractere como escape
- **Exemplo:** selecionar tuplas em que o valor para *s* é uma *string* iniciada e finalizada por '%'

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

- `'%Olá mundo!%' LIKE 'x%%x%' ESCAPE 'x'` → `TRUE`
- `'Olá mundo!' LIKE 'x%%x%' ESCAPE 'x'` → `FALSE`
- `'%Olá mundo!' LIKE 'x%%x%' ESCAPE 'x'` → `FALSE`

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'

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;
```

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;
```

Qual é o resultado dessa consulta?

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**
  - Se a cláusula é FALSE ou UNKNOWN, então a tupla **NÃO** entre na resposta

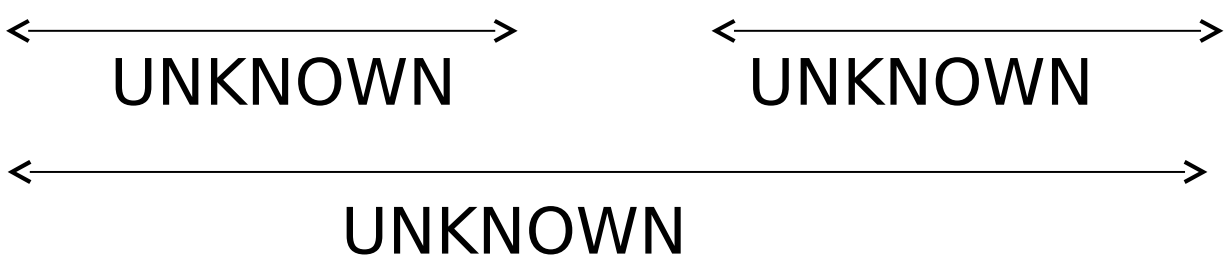
Um exemplo surpreendente

IME-USP

- 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;
```



Resultado: nenhuma tupla é selecionada!

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 = ½
  - AND = MIN , OR = MAX
  - NOT(*x*) = 1-*x*

- **Exemplo:**

TRUE AND (FALSE OR NOT(UNKNOWN)) =

MIN(1, MAX(0, (1 - ½))) =

MIN(1, MAX(0, ½)) = MIN(1, ½) = ½ = UNKNOWN

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 valem**
  - Exemplo: *lei do terceiro excluído*

    p OR NOT *p* = TRUE

  - Quando *p* = UNKNOWN, o lado esquerdo é

    MAX(½, (1 – ½)) = ½ <> 1

Resumo

- SELECT – FROM – WHERE
- Renomeamento de atributos (com AS)
- Expressões na cláusula SELECT
- Operadores lógicos e de comparação
- Operador LIKE
- Cláusula ORDER BY
- Valor UNKNOWN e a lógica ternária da SQL

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