

# ***Structured Query Language (SQL)***

## **Parte 5**

### **Consultas com Agrupamentos e Agregações**

Kelly Rosa Braghetto  
[kellyrb@ime.usp.br](mailto:kellyrb@ime.usp.br)

**Departamento de Ciência da Computação**

# Consultas mais avançadas



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(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)**

# Agregações



- As funções

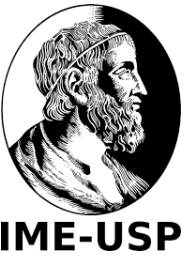
**SUM, AVG, COUNT, MIN e MAX**

podem ser aplicadas a uma coluna na cláusula **SELECT** para produzir a agregação dos valores da referida coluna.

- Além disso, **COUNT(\*)** conta o número de tuplas.

# Exemplo:

## Agregação com AVG



- A partir de `Venda(nome_lanch, nome_refri, preço)`, encontre o preço médio do refri Fanfa:

```
SELECT AVG(preço)
FROM Venda
WHERE nome_refri = 'Fanfa';
```

# Eliminando duplicações em uma agregação



- O DISTINCT pode ser usado para eliminar duplicação de valores antes da agregação.
- **Exemplo:** encontre o número de preços *diferentes* cobrados pela Fanfa:

```
SELECT COUNT(DISTINCT preço)
FROM Venda
WHERE nome_refri = 'Fanfa';
```

# Valores NULL são ignorados na agregação



- Um valor NULL nunca contribui para uma soma, média ou contagem.
- O valor NULL também não será nem o mínimo, nem o máximo de uma coluna se ela possuir outros valores nela.
- Mas se não existirem valores não nulos em uma coluna, então o resultado da agregação é NULL.
  - ▶ **Exceção:** COUNT de um conjunto vazio é 0.

# Exemplo: efeito de NULLs



```
SELECT count(*)  
FROM Venda WHERE  
nome_refri = 'Fanfa';
```

O número de lanchonetes  
que vendem Fanfa.

```
SELECT count(preço)  
FROM Venda WHERE  
nome_refri = 'Fanfa';
```

O número de lanchonetes  
que vendem Fanfa a um  
preço conhecido (ou seja,  
diferente de NULL).



# Agrupamento



- Depois de uma expressão SELECT-FROM-WHERE, podemos adicionar **GROUP BY** e uma lista de atributos.
- A relação resultante do SELECT-FROM-WHERE com GROUP BY é agrupada de acordo com os valores de todos os referidos atributos
  - Quando há agrupamento, as agregações são aplicadas sobre cada grupo.

# Exemplo: agrupamento

- A partir de  
`Venda(nome_lanch, nome_refri, preço)`, encontre  
o preço médio de cada refri:

```
SELECT nome_refri, AVG(preço)
FROM Venda
GROUP BY nome_refri;
```

nome_refri	AVG(preço)
Fanfa	5.33
Cola-coca	5.12
...	...

# Exemplo: agrupamento



- A partir de `Venda(nome_lanch, nome_refri, preço)` e `Frequentador(nome_cliente, nome_lanch)`, encontre, para cada cliente, o preço médio da Fanfa nas lanchonetes que ele frequenta:

```
SELECT nome_cliente, AVG(preço)
FROM Frequentador, Venda
WHERE nome_refri = 'Fanfa' AND
Frequentador.nome_lanch =
      Venda.nome_lanch
GROUP BY nome_cliente;
```

Computa todas as tuplas cliente-lanch-preço para Fanfa.

Depois, as agrupa por cliente.

# Restrição no SELECT: listas com agregação



- ◆ Se um agrupamento é usado, então cada elemento da lista do SELECT precisa ser:
  1. Uma agregação, ou
  2. Um atributo da lista do GROUP BY.

# Exemplo:

## Consulta incorreta



- Alguém pode pensar que é possível encontrar a lanchonete que vende Fanfa mais barato usando:

```
SELECT nome_lanch, MIN(preço)
FROM Venda
WHERE nome_refri = 'Fanfa';
```

- Mas essa consulta **NÃO** é permitida em SQL.

# Exemplo:

Lanchonete que vende Fanfa mais barato

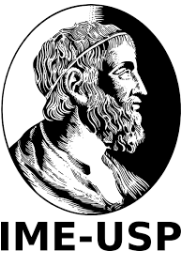


```
SELECT nome_lanch, preço FROM Venda  
WHERE nome_refri = 'Fanfa' AND  
       preço <= ALL (SELECT preço FROM Venda  
                     WHERE nome_refri = 'Fanfa');
```

ou

```
SELECT nome_lanch, preço FROM Venda  
WHERE nome_refri = 'Fanfa' AND  
       preço = (SELECT MIN(preço) FROM Venda  
               WHERE nome_refri = 'Fanfa');
```

# Cláusulas HAVING



- A cláusula **HAVING** <condição> pode aparecer depois da cláusula GROUP BY.
  - Se aparecer, a condição do HAVING é aplicada sobre cada grupo de tuplas.
  - Grupos que não satisfazem a condição são eliminados da resposta da consulta.

# Exemplo: HAVING



- A partir de

`Venda(nome_lanch, nome_refri, preço)`

encontre os nomes dos refri que são vendidos a um preço médio menor do que R\$ 6,00.

```
SELECT nome_refri, AVG(preço)
FROM Venda
GROUP BY nome_refri
HAVING AVG(preço) <= 6
```



# Exemplo: HAVING



- A partir de  
Venda(nome\_lanch, nome\_refri, preço) e  
Refrigerante(nome, fabricante),  
encontre o preço médio dos refri que são servidos  
em pelo menos 3 lanchonetes ou que são fabricados  
pela Cola-Coca.



IME-USP

# Exemplo: HAVING

Preço médio dos refri servidos em pelo menos 3 lanchonetes ou fabricados pela Cola-Coca

```
SELECT nome_refri, AVG(preço)
FROM Venda
GROUP BY nome_refri
HAVING COUNT(nome_lanch) >= 3 OR
nome_refri IN
    (SELECT nome
     FROM Refrigerantes
     WHERE fabricante = 'Cola-Coca');
```

Inclui na resposta o grupo de um refri quando ele é vendido em pelo menos 3 lanchonetes ou quando o fabricante é a Cola-Coca.

Refris fabricados pela Cola-Coca.

# Requisitos para as condições do HAVING



- ◆ Nas condições do HAVING, pode-se ter qualquer tipo de subconsultas;
- ◆ Fora de subconsultas, o HAVING pode referenciar um elemento somente se ele for:
  1. Um atributo agrupador, ou
  2. Uma agregação(essa é a mesma regra usada para cláusulas SELECT com agregação).

# 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