

MAC0439 – Laboratório de Bancos de Dados

# Aula 14

## **Mais SQL**

Álgebra Relacional Estendida  
Agrupamento/Agregação  
Junções

07 de outubro de 2015  
Profa. Kelly Rosa Braghetto

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

# Álgebra Relacional Estendida

$\delta$  = elimina tuplas duplicadas em multiconjuntos.

$\tau$  = ordena tuplas.

$\gamma$  = agrupamento e agregação.

*Junção Externa (Outerjoin)*: evita “tuplas soltas” = tuplas que não se “juntam” a nada.

# Eliminação de duplicações

- ◆  $R1 := \delta(R2)$ .
- ◆ R1 é formada por uma **única cópia** de cada tupla que aparece em R2 (mesmo que a tupla aparece em R2 mais de uma vez).

# Exemplo: eliminação de duplicações

R =

A	B
1	2
3	4
1	2

$\delta(R) =$

A	B
1	2
3	4

# Ordenação

- ◆  $R1 := \tau_L (R2)$ .
  - ◆  $L$  é uma lista de alguns dos atributos de  $R2$ .
- ◆  $R1$  é a lista das tuplas de  $R2$  ordenadas primeiro pelo valor do primeiro atributo de  $L$ , então pelo segundo atributo de  $L$ , e assim por diante.
- ◆  $\tau$  é o único operador de Álgebra Relacional que gera um resultado que não é nem um conjunto, nem um multiconjunto.

# Exemplo: ordenação

$R =$ 

A	B
1	2
3	4
5	2

$$T_{B,A}(R) = [(1,2), (5,2), (3,4)]$$

# Operadores de agregação

- ◆ Operadores de agregação não são operadores da álgebra relacional.
- ◆ Eles se aplicam a colunas inteiras de uma tabela e produzem um único resultado.
- ◆ Os exemplos mais importantes são: **SUM, AVG, COUNT, MIN e MAX.**

# Exemplo: agregação

R =

A	B
1	3
3	4
3	2

SUM(A) = 7

COUNT(A) = 3

MAX(B) = 4

AVG(B) = 3



# Operador de agrupamento

◆  $R1 := \gamma_L (R2).$

$L$  é uma lista de elementos que podem ser:

1. Atributos individuais (agrupadores).
2.  $AGG(A)$ , onde  $AGG$  é um operador de agregação e  $A$  é um atributo.
  - Uma seta e um novo nome de atributo renomeia o componente.

# Aplicando $\gamma_L(R)$

- ◆ Agrupa  $R$  segundo todos atributos agrupadores em  $L$ .
  - ◆ Ou seja: forma um grupo para cada combinação de valor distinta para esses atributos em  $R$ .
- ◆ Dentro de cada grupo, computa  $AGG(A)$ .
- ◆ O resultado tem uma tupla para cada grupo, contendo:
  1. Os atributos agrupadores e
  2. E as agregações de seus grupos.

# Exemplo: agrupamento/agregação

$R = ($

A	B	C
1	2	3
4	5	6
1	2	5

)

Então, calcula a média de C dentro dos dois grupos:

A	B	X
1	2	4
4	5	6

$$Y_{A,B,AVG(C) \rightarrow X}(R) = ??$$

Primeiro, agrupa  $R$  por  $A$  e  $B$  :

A	B	C
1	2	3
1	2	5
4	5	6

# Junção externa

- ◆ Considere a junção  $R \bowtie_c S$ .
- ◆ Uma tupla de  $R$  que não possui uma tupla em  $S$  para realizar a junção é chamada de *solta*.
  - ▶ O mesmo vale para uma tupla de  $S$ .
- ◆ Uma **junção externa** preserva as tuplas soltas, “complementando-as” com NULL.

# Exemplos de junções externas

R =

A	B
1	2
4	5

S =

B	C
2	3
6	7

(1,2) junta com (2,3), mas as duas outras tuplas são “soltas”.

R Junção Externa S =

A	B	C
1	2	3
4	5	NULL
NULL	6	7

# Agora, de volta ao SQL

Cada uma dessas operações  
tem um comando equivalente  
em SQL

# Aggregações

- ◆ **SUM, AVG, COUNT, MIN e MAX** podem ser aplicados a uma coluna na cláusula **SELECT** para produzir a agregação da referida coluna.
- ◆ Além disso, **COUNT(\*)** conta o número de tuplas.

# Exemplo: Agregação

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

```
SELECT AVG(preço)
```

```
FROM Venda
```

```
WHERE nome_refri = 'Fanfa';
```



# Eliminando duplicações em uma agregação

- ◆ Use DISTINCT dentro de uma 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 NULL nunca contribui para uma soma, média ou contagem, e nunca pode ser nem o mínimo, nem o máximo de uma coluna.
- ◆ Mas se não existir valores não nulos em uma coluna, então o resultado da agregação é NULL.
  - ▶ **Exceção:** COUNT de um conjunto vazio é 0.

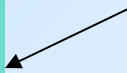
# Exemplo: efeito de NULLs

```
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 e qualquer agregação é aplicada somente dentro de 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	2.33
...	...

# 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 de 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.



# Cláusulas HAVING

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

# 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.

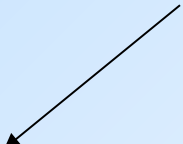
# Solução

```
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');
```

Grupos de refri com pelo menos 3 lanchonetes não nulas e também grupos em que o fabricante é a Cola-Coca.



Refris fabricados pela Cola-Coca.



# Requisitos para as condições do HAVING

- ◆ Vale qualquer coisa dentro de uma subconsulta.
- ◆ 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).

# Expressões de Junção (JOIN)

- ◆ SQL possui várias versões de junções.
- ◆ Os operadores de junção (JOIN) só foram introduzidos no padrão SQL2.
- ◆ Padrões anteriores da SQL não possuíam operadores explícitos para a junção.
- ◆ Mas é sempre possível obter o mesmo efeito deles por meio de uma consulta do tipo SELECT-FROM-WHERE.
- ◆ As expressões JOIN podem ser consultas “por si só” ou podem ser usadas no lugar de relações em uma cláusula FROM.

# Produto Cartesiano

- ◆ É o tipo de junção mais simples:

R CROSS JOIN S;

- ◆ As relações envolvidas no produto também podem ser subconsultas parentizadas (isso vale para todos os tipos de JOIN).
- ◆ O produto cartesiano sozinho raramente é útil.

# Junção Natural

- ◆ Forma da junção natural: **R NATURAL JOIN S**
- ◆ A condição de junção é a igualdade sobre os pares de atributos das duas relações que possuem o mesmo nome.
- ◆ **Exemplo:**

```
    Apreciador NATURAL JOIN Venda;
```

- ◆ Equivale a:

```
SELECT A.nome_cliente, V.*  
FROM Apreciador A, Venda V  
WHERE A.nome_refri = Venda.nome_refri
```

# Junção Teta

◆ **R JOIN S ON <condição>**

◆ **Exemplo:** usando **Cliente(nome, endereço)** e **Frequentador(nome\_cliente, nome\_lanch)**:

```
Cliente JOIN Frequentador ON  
    nome = nome_cliente;
```

nos dá todas quádruplas  $(c, e, c, l)$  tais que cliente  $c$  mora no endereço  $e$  e frequenta a lanchonete  $l$ .



# Junção Externa (Outerjoin)

◆ **R OUTER JOIN S** é o núcleo de uma expressão de junção externa. Ela pode ser modificada por três cláusulas opcionais:

1. **NATURAL** antes de OUTER

2. **ON** <condição> depois de JOIN

3. **LEFT**, **RIGHT**, ou **FULL** antes de OUTER

- ◆ LEFT = inclui apenas as tuplas soltas de R.
- ◆ RIGHT = inclui apenas as tuplas soltas de S.
- ◆ FULL = inclui as tuplas soltas de ambas; esta é a opção padrão.

Apenas uma  
entre essas duas

# Exemplo: junção externa

- ◆ Exemplo: usando `Cliente(nome, endereço)` e `Frequentador(nome_cliente, nome_lanch)`

```
Cliente LEFT OUTER JOIN Frequentador ON  
nome = nome_cliente;
```

nos dá uma lista de tuplas, onde cada tupla associa os dados de um cliente ao um nome de lanchonete que ele frequenta. Se um cliente não frequenta nenhuma lanchonete, seus dados aparecerão na lista associados ao valor NULL (para o nome de lanchonete).

## Exemplo:

select \* from R left outer join S on (B=D and C=E);

Exemplo:  $R \bowtie_{B=D, C=E} S$

A	B	C
1	2	3
4	5	6
7	8	9

Relação R

D	E	F
2	3	10
2	3	11
6	7	12

Relação S

A	B	C	D	E	F
1	2	3	2	3	10
1	2	3	2	3	11
4	5	6	NULL	NULL	NULL
7	8	9	NULL	NULL	NULL

Resultado de  $R \bowtie_{B=D, C=E} S$

## Exemplo:

select \* from R right outer join S on (B=D and C=E);

Exemplo:  $R \bowtie_{B=D, C=E} S$

A	B	C
1	2	3
4	5	6
7	8	9

Relação R

D	E	F
2	3	10
2	3	11
6	7	12

Relação S

A	B	C	D	E	F
1	2	3	2	3	10
1	2	3	2	3	11
NULL	NULL	NULL	6	7	12

Resultado de  $R \bowtie_{B=D, C=E} S$

## Exemplo:

select \* from R full outer join S on (B=D and C=E);

Exemplo:  $R \bowtie_{B=D, C=E} S$

A	B	C
1	2	3
4	5	6
7	8	9

Relação R

D	E	F
2	3	10
2	3	11
6	7	12

Relação S

A	B	C	D	E	F
1	2	3	2	3	10
1	2	3	2	3	11
4	5	6	NULL	NULL	NULL
7	8	9	NULL	NULL	NULL
NULL	NULL	NULL	6	7	12

Resultado de  $R \bowtie_{B=D, C=E} S$

## Exemplo:

select \* from R natural left outer join S;

A	B	C
1	2	3
4	5	6
7	8	9

Relação *R*

B	C	D
2	3	10
2	3	11
6	7	12

Relação *S*

A	B	C	D
1	2	3	10
1	2	3	11
4	5	6	NULL
7	8	9	NULL

Resultado da junção natural externa à esquerda

## Exemplo:

select \* from R natural right outer join S;

A	B	C
1	2	3
4	5	6
7	8	9

Relação *R*

B	C	D
2	3	10
2	3	11
6	7	12

Relação *S*

A	B	C	D
1	2	3	10
1	2	3	11
NULL	6	7	12

Resultado da junção natural externa à direita

## Exemplo:

select \* from R natural full outer join S;

A	B	C
1	2	3
4	5	6
7	8	9

Relação *R*

B	C	D
2	3	10
2	3	11
6	7	12

Relação *S*

A	B	C	D
1	2	3	10
1	2	3	11
4	5	6	NULL
7	8	9	NULL
NULL	6	7	12

Resultado da junção natural externa completa



# 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 5