



**UAlg FCT**

UNIVERSIDADE DO ALGARVE  
FACULDADE DE CIÊNCIAS E TECNOLOGIA

# Bases de Dados

---

ÁLGEBRA RELACIONAL – EXTENSÕES E  
OPERADORES ADICIONAIS

# Operadores adicionais

---

Operadores que não adicionam capacidade, mas permitem simplificar as queries

- Atribuição
- Natural join
- Divisão

# A Operação de Atribuição

---

Às vezes, é conveniente escrever uma expressão de álgebra relacional atribuindo partes dela a variáveis de relação temporárias.

A operação de atribuição é denotada por  $\leftarrow$  e funciona como a atribuição numa linguagem de programação.

Exemplo: Encontre todos os instrutores do departamento de “Física” ou do departamento de “Música”.

$$\begin{aligned} Física &\leftarrow \sigma_{dept\_name = \text{“ Física ”}}(instrutor) \\ Música &\leftarrow \sigma_{dept\_name = \text{“ Música ”}}(instrutor) \\ Física &\cup Música \end{aligned}$$

Com a operação de atribuição, uma consulta pode ser escrita como um programa sequencial que consiste numa série de atribuições, seguidas por uma expressão cujo valor é exibido como resultado da consulta.

# Operação de junção (*join*) – cf. Slide 10

---

O produto cartesiano

*instrutor X teaches*

associa cada tuplo de *instrutor* com cada tuplo de *teaches*.

- A maioria das linhas resultantes contém informações sobre professores que NÃO ministraram uma disciplina específica.

Para obter apenas os tuplos de “*instrutor X teaches*” que pertencem aos instrutores e às disciplinas que eles ministraram, escrevemos:

$$\sigma_{instrutor.id = teaches.id} (instrutor \times teaches)$$

- Obtemos apenas os tuplos de “*instrutor X teaches*” que pertencem aos instrutores e às disciplinas que eles ministraram.

O resultado desta expressão, é mostrado no próximo slide

# Operação de junção – *join* (continuação) – cf. Slide 11

A tabela correspondente a:

$\sigma_{instructor.id = ensina.id} (instructor \times teaches)$

<i>instructor.ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>	<i>teaches.ID</i>	<i>course_id</i>	<i>sec_id</i>	<i>semester</i>	<i>year</i>
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2018
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2018
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2017
32343	El Said	History	60000	32343	HIS-351	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-101	1	Spring	2018
45565	Katz	Comp. Sci.	75000	45565	CS-319	1	Spring	2018
76766	Crick	Biology	72000	76766	BIO-101	1	Summer	2017
76766	Crick	Biology	72000	76766	BIO-301	1	Summer	2018
83821	Brandt	Comp. Sci.	92000	83821	CS-190	1	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-190	2	Spring	2017
83821	Brandt	Comp. Sci.	92000	83821	CS-319	2	Spring	2018
98345	Kim	Elec. Eng.	80000	98345	EE-181	1	Spring	2017

# Operação de *join* (continuação)

---

A operação **de join** permite combinar uma operação de seleção e uma operação de produto cartesiano numa única operação.

Considere as relações  $r ( R )$  e  $s ( S )$

Seja  $\theta$  (*teta*) um predicado em atributos no esquema R “union” S.

A operação de junção  $r \bowtie_{\theta} s$  é definida como se segue:

$$r \bowtie_{\theta} s = \sigma_{\theta} ( r \times s )$$

Por isso

$$\sigma_{instrutor.id = teaches.id} ( instrutor \times teaches )$$

Pode ser escrito de forma equivalente como (operador **teta join**)

$$instrutor \bowtie_{instrutor.id = teaches.id} teaches$$

# Operação de Natural-Join

---

Notação:  $r \bowtie s$

Sejam  $r$  e  $s$  relações nos esquemas  $R$  e  $S$ , respetivamente.

Então,  $r \bowtie s$  é uma relação no esquema  $R \cup S$  obtida da seguinte forma:

- Considere cada par de tuplos  $t_r$  de  $r$  e  $t_s$  de  $s$  (i.e.,  $r \times s$ ).
- Se  $t_r$  e  $t_s$  têm o mesmo valor para os atributos em  $R \cap S$ , adicione um tupo  $t$  ao resultado, em que:
  - $t$  tem os valores de  $t_r$  em  $r$
  - $t$  tem os valores de  $t_s$  em  $s$

Exemplo:

$R = (A, B, C, D)$        $S = (E, B, D)$

- Esquema resultante =  $(A, B, C, D, E)$
- $r \bowtie s$  define-se como:

$$\Pi_{r.A, r.B, r.C, r.D, s.E} (\sigma_{r.B = s.B \wedge r.D = s.D} (r \times s))$$

# Operação de Natural-Join: Exemplo

Relações  $r$ ,  $s$ :

$A$	$B$	$C$	$D$
$\alpha$	1	$\alpha$	a
$\beta$	2	$\gamma$	a
$\gamma$	4	$\beta$	b
$\alpha$	1	$\gamma$	a
$\delta$	2	$\beta$	b

$r$

$B$	$D$	$E$
1	a	$\alpha$
3	a	$\beta$
1	a	$\gamma$
2	b	$\delta$
3	b	$\epsilon$

$s$

$r \bowtie s$

$A$	$B$	$C$	$D$	$E$
$\alpha$	1	$\alpha$	a	$\alpha$
$\alpha$	1	$\alpha$	a	$\gamma$
$\alpha$	1	$\gamma$	a	$\alpha$
$\alpha$	1	$\gamma$	a	$\gamma$
$\delta$	2	$\beta$	b	$\delta$



# Operação de Divisão

---

Adequado para queries que têm o pedido: “para todos”.

Notação:

$$r \div s$$

Sejam  $r$  e  $s$  relações nos esquemas  $R$  e  $S$ , respetivamente, onde:

- $R = (A_1, \dots, A_m, B_1, \dots, B_n)$
- $S = (B_1, \dots, B_n)$

O resultado de  $r \div s$  é uma relação no esquema

$$R - S = (A_1, \dots, A_m)$$

$$r \div s = \{ t \mid t \in \Pi_{R-S}(r) \wedge \forall u \in s (tu \in r) \}$$

# Operação de Divisão - Exemplo

Relações  $r$ ,  $s$ :

$A$	$B$
$\alpha$	1
$\alpha$	2
$\alpha$	3
$\beta$	1
$\gamma$	1
$\delta$	1
$\delta$	3
$\delta$	4
$\epsilon$	6
$\epsilon$	1
$\beta$	2

$r$

$B$
1
2

$s$

$r \div s$ :

$A$
$\alpha$
$\beta$

# Operação de Divisão – Exemplo 2

Relações  $r$ ,  $s$ :

$A$	$B$	$C$	$D$	$E$
$\alpha$	$a$	$\alpha$	$a$	$1$
$\alpha$	$a$	$\gamma$	$a$	$1$
$\alpha$	$a$	$\gamma$	$b$	$1$
$\beta$	$a$	$\gamma$	$a$	$1$
$\beta$	$a$	$\gamma$	$b$	$3$
$\gamma$	$a$	$\gamma$	$a$	$1$
$\gamma$	$a$	$\gamma$	$b$	$1$
$\gamma$	$a$	$\beta$	$b$	$1$

$r$

$D$	$E$
$a$	$1$
$b$	$1$

$s$

$r \div s$ :

$A$	$B$	$C$
$\alpha$	$a$	$\gamma$
$\gamma$	$a$	$\gamma$

# Extensão à Álgebra Relacional: Extensões

---

Projeção generalizada

Agregação

Outer Join

# Projeção generalizada

---

Estende o operador de projeção, permitindo a utilização de funções aritméticas na lista de atributos a projetar

$$\Pi_{F_1, F_2, \dots, F_n}(E)$$

$E$  é qualquer expressão de Álgebra Relacional

$F_1, F_2, \dots, F_n$  são expressões aritméticas que usam constantes ou valores de atributos no esquema de  $E$ .

Exemplo:

- $\Pi_{ID, nome, salario*1.1}(instrutor)$

# Agregação

---

**Função de agregação** retorna um **único** valor a partir de um conjunto de valores.

**avg:** média  
**min:** mínimo  
**max:** máximo  
**sum:** soma  
**count:** contagem

**Operação de agregação** em álgebra relacional:

$$G_1, G_2, \dots, G_n \quad \mathcal{g}_{F_1(A_1), F_2(A_2), \dots, F_n(A_n)}(E)$$

- $E$  é uma expressão de álgebra relacional
- $G_1, G_2, \dots, G_n$  é uma lista com os atributos para agrupar (pode ser vazia)
- $F_i$  são funções de agregação
- $A_i$  nomes de atributos no esquema de  $E$

# Agregação— Exemplo

---

Relação  $r$ :

$A$	$B$	$C$
$\alpha$	$\alpha$	7
$\alpha$	$\beta$	7
$\beta$	$\beta$	3
$\beta$	$\beta$	10

$g_{\text{sum}(C)}(r)$

$sum-C$
27

# Agregação— Exemplo

Relação conta

<i>agencia</i>	<i>Num-conta</i>	<i>saldo</i>
Faro	A-102	400
Faro	A-201	900
Loulé	A-217	750
Loulé	A-215	750
Albufeira	A-222	700

Total de saldo por agência

*agencia*  $\rho_{sum(saldo)}$  (*conta*)

<i>agencia</i>	<i>saldo</i>
Faro	1300
Loulé	1500
Albufeira	700



# Aggregate Functions (Cont.)

---

O resultado de uma função de agregação não tem nome

- Podemos usar o operador *renomear* para lhe atribuir um nome
- Por simplificação, permitimos renomear as colunas diretamente nas funções de agregação

*agencia* ***g*** *sum(saldo) as soma-saldo (conta)*

# Outer Join

---

Extensão ao operador de join para evitar a perda de informação.

Calcula o *join* e adiciona os tuplos de uma relação que não têm correspondência com os tuplos da outra relação.

Usa valores *null*:

- *null* significa que o valor é desconhecido
- As comparações com *null* são (genericamente) **falsas** por definição.

# Outer Join – Exemplo

---

## Relação empréstimo

<i>emprest-num</i>	<i>agencia-nome</i>	<i>valor</i>
L-170	Faro	3000
L-230	Loulé	4000
L-260	Albufeira	1700

## Relação titular-empréstimo

<i>cliente-nome</i>	<i>emprest-num</i>
António	L-170
Maria	L-230
Ana	L-155

# Outer Join – Exemplo

## Inner Join

*emprestimo* ⋈ *titular-emprestimo*

<i>emprest-num</i>	<i>agencia-nome</i>	<i>valor</i>	<i>cliente-nome</i>
L-170	Faro	3000	António
L-230	Loulé	4000	Maria

## Left Outer Join

*emprestimo* ⋈<sub>L</sub> *titular-emprestimo*

<i>emprest-num</i>	<i>agencia-nome</i>	<i>valor</i>	<i>cliente-nome</i>
L-170	Faro	3000	António
L-230	Loulé	4000	Maria
L-260	Albufeira	1700	<i>null</i>

# Outer Join – Exemplo

## Right Outer Join

*emprestimo* ⋈<sub>r</sub> *titular-emprestimo*

<i>emprest-num</i>	<i>agencia-nome</i>	<i>valor</i>	<i>cliente-nome</i>
L-170	Faro	3000	António
L-230	Loulé	4000	Maria
L-155	<i>null</i>	<i>null</i>	Ana

## Full Outer Join

*emprestimo* ⋈<sub>f</sub> *titular-emprestimo*

<i>emprest-num</i>	<i>agencia-nome</i>	<i>valor</i>	<i>cliente-nome</i>
L-170	Faro	3000	Maria
L-230	Loulé	4000	Smith
L-260	Albufeira	1700	<i>null</i>
L-155	<i>null</i>	<i>null</i>	Ana

# Valores Null

---

Os tuplos podem assumir valores *null* para alguns dos seus atributos  
*null* significa que o valor é desconhecido

O resultado de uma expressão aritmética envolvendo *null* é *null*

As funções de agregação ignoram os valores *null*

Para eliminação de duplicados e criação de grupos, o valor *null* é tratado como qualquer outro valor, isto é, dois valores *null* são considerados como o mesmo valor.

# Valores Null

---

Comparações com o valor *null* retornam o valor especial *unknown*

- Se fosse *false*, então *not (A < 5)*  
não seria equivalente a *A >= 5*

Operadores lógicos com *unknown*:

- OR: (*unknown or true*) = *true*  
(*unknown or false*) = *unknown*  
(*unknown or unknown*) = *unknown*
- AND: (*true and unknown*) = *unknown*  
(*false and unknown*) = *false*  
(*unknown and unknown*) = *unknown*
- NOT: (**not** *unknown*) = *unknown*

Se resultado de um predicado no operador de seleção for *unknown*, é tratado como *false*