

Bases de Dados

T11 - Álgebra Relacional Parte II

Prof. Daniel Faria

Prof. Flávio Martins

Sumário

- Recapitulação Breve
- Álgebra Relacional Parte II
- Exemplos

Recapitulação Breve

Álgebra Relacional

- Linguagem procedimental que consiste em operações algébricas sobre relações, cujo resultado é também uma relação
 - Inclui operações unárias (argumento é uma só relação) e binárias (argumento são duas relações)
- Uma vez que o resultado de cada operação é uma relação, operações de álgebra relacional podem ser combinadas em expressões

Álgebra Relacional

- Operadores básicos

- Seleção: σ
- Projeção: Π
- Produto Cartesiano: \times
- Renomear: ρ
- Diferença: $-$
- União: \cup

Sobre estes operadores podem-se definir outros, e.g.:

- Join: \bowtie
- Interseção: \cap
- Divisão: \div
- ...

Operadores de Álgebra Relacional

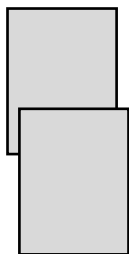
Select: σ

A	B	C	D

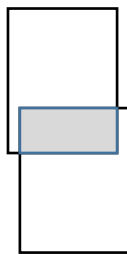
Project: Π

A	B	C	D

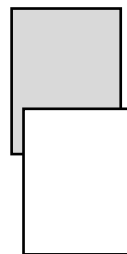
Union: \cup



Intersection: \cap



Difference: $-$



Cartesian Product: \times

a1	b1
a2	b2
a3	b3

b1	c1
b3	c2



a1	b1	b1	c1
a1	b1	b3	c2
a2	b2	b1	c1
a2	b2	b3	c2
a3	b3	b1	c1
a3	b3	b3	c2

(Natural) Join: \bowtie

a1	b1
a2	b2
a3	b3

b1	c1
b3	c2



a1	b1	c1
a3	b3	c2

Outer Join: $\ltimes, \ltimes, \ltimes$

a1	b1
a2	b2
a3	b3

b1	c1
b3	c2



a1	b1	c1
a3	b3	c2
a2	b2	NULL

Division: \div

a1	b1
a1	b2
a2	b1
a3	b2


b1
b2



a1

Aggregation: G or γ

a1
a2
a3
a4



val

Álgebra Relacional Parte II

Atribuição / Assignment

- Operação unária que “guarda” uma relação resultante de uma expressão e lhe atribui um nome
 - Conveniente para guardar resultados de expressões de forma a poder reutilizá-los em expressões mais complexas, como uma série de expressões
- Notação: $r \leftarrow E$
 - onde E é uma expressão algébrica

Atribuição / Assignment

- E.g. quais os nomes dos marinheiros que reservaram um barco vermelho ou verde?

$$r \leftarrow boat \bowtie_{b.bid=r.bid} reserve \bowtie_{r.sid=s.sid} sailor$$

$$\Pi_{sname}(\sigma_{color="red"}(r)) \cup \Pi_{sname}(\sigma_{color="green"}(r))$$

Atribuição / Assignment

- Pode ser usada para expressar remoção de tuplos

$$r \leftarrow s - (E)$$

- Ou adição de tuplos

$$r \leftarrow s \cup (E)$$

- Ou atualização de tuplos, utilizando projeção generalizada (mais adiante)

Divisão / Division

- Operação binária que seleciona tuplos de uma relação que contêm todos os tuplos de uma segunda relação, devolvendo os atributos da primeira relação que não estão na segunda
- Definição: $r \div s = \Pi_{r \cap \neg s}(r) - \Pi_{r \cap \neg s}((s \times \Pi_{r \cap \neg s}(r)) - r)$
 - Só está definida se os atributos de s estão contidos nos atributos de r

T	=	R		÷	S
B		A	B		A
b1		a1	b1		a1
b4		a2	b1		a2
		a3	b1		a3
		a4	b1		
		a1	b2		
		a3	b2		
		a2	b3		
		a3	b3		
		a4	b3		
		a1	b4		
		a2	b4		
		a3	b4		

Divisão / Division

- E.g. quais os nomes dos marinheiros que alugaram todos os barcos

$$\Pi_{sname}(\text{sailor} \bowtie_{r.sid=s.sid} (\Pi_{sid,bid}(\text{reserve}) \div \Pi_{bid}(\text{boat})))$$

↓

sname
Maria

↙

sid
101

↓

sid	bid
101	33
120	30
134	33
101	20
101	30

÷

↓

bid
20
30
33

Sailor			
sid	sname	rating	age
101	Maria	1	35
120	Zé	2	30
134	João	3	60

Boat		
bid	bname	color
20	wave	green
30	Delfin	red
33	wind	blue

Reserve		
sid	bid	day
101	33	01/01/2020
120	30	11/01/2019
134	33	07/03/2019
101	20	11/11/2019
101	30	11/11/2019

Divisão / Division

- E.g. quais os nomes dos marinheiros que alugaram todos os barcos

$$\Pi_{sid, bid}(reserve) \div \Pi_{bid}(boat) =$$

$$\Pi_{sid}(reserve) - \Pi_{sid}((boat \times \Pi_{sid}(reserve)) - \Pi_{sid, bid}(reserve))$$

sid
101
120
134

sid
101

sid
120
134

sid	bid
101	20
101	30
101	33
120	20
120	30
120	33
134	20
134	30
134	33

sid	bid
120	20
120	33
134	20
134	30

sid	bid
101	33
120	30
134	33
101	20
101	30

Todos os marinheiros que não alugaram todos os barcos

Todos os pares marinheiro, barco que não ocorreram

Todos os pares possíveis marinheiro, barco

Boat		
bid	bname	color
20	wave	green
30	Delfin	red
33	wind	blue

Reserve		
sid	bid	day
101	33	01/01/2020
120	30	11/01/2019
134	33	07/03/2019
101	20	11/11/2019
101	30	11/11/2019

Todos os pares reais marinheiro, barco

Extensões à Álgebra Relacional

- Projeção generalizada
- Agregação (aggregation)
 - c/ agrupamento (grouping)
- NULLs
- Junção externa (outer-join)

Projeção Generalizada

- Extensão à projeção onde são permitidas operações algébricas envolvendo atributos

$$\Pi_{F1, \dots, FN}(r)$$

- onde Fi pode ser um atributo de r ou uma expressão algébrica envolvendo um ou mais atributos de r
- E.g. obter o salário dos professores com um aumento de 10%

ID	name	dept	salary
1	John	Chemistry	65000
2	Jack	Physics	95000
3	Jill	Physics	82000
4	Joan	Biology	82000


$$\Pi_{name, salary * 1.1}(\text{professor})$$


name	salary
John	71500
Jack	104500
Jill	90200
Joan	90200

Projeção Generalizada

- E.g. quanto é que cada cliente pode gastar até ao seu limite de crédito?

$$\rho_{result(2 \rightarrow credit_available)} (\prod_{customer_name, limit - credit_balance} (credit_info))$$

<i>credit_info</i>					<i>result</i>	
<i>customer_name</i>	<i>limit</i>	<i>credit_balance</i>			<i>customer_name</i>	<i>credit_available</i>
Curry	2000	1750			Curry	250
Hayes	1500	1500			Jones	5300
Jones	6000	700			Smith	1600
Smith	2000	400			Hayes	0

Projeção Generalizada

- A projeção generalizada pode ser utilizada para representar actualização de relações (update):

$$r \leftarrow \prod_{f1, \dots, fn}(r)$$

- Em que fi é:
 - O nome do atributo, para os atributos que não queremos alterar
 - Um valor ou operação algébrica com atributos, para os atributos que queremos alterar

- Se quisermos actualizar apenas alguns tuplos:

$$r \leftarrow \prod_{f1, \dots, fn}(\sigma_p(r)) \cup (r - \sigma_p(r))$$

Agregação

- Operação unária que devolve um tuplo único que agrega os valores dos tuplos de uma relação
- Notação: $G_{G1, \dots, Gn}^{F1(), \dots, Fm()}(r)$
 - Onde G_i são os atributos a agrupar e $F_i()$ as funções de agregação a aplicar em cada grupo tais que:
 - Todos os tuplos num grupo têm o mesmo valor dos atributos
 - Tuplos de grupos diferentes têm diferentes valores dos atributos
 - Para cada grupo são calculadas as funções de agregação
 - Não se especificando G_i , toda a relação é considerada um grupo

Agregação

- Exemplos de funções de agregação: avg, min, max, sum, count, count-distinct
- E.g.
 - Total dos salários dos funcionários

$G_{\text{sum}(\text{salary})}(\text{works})$

- Número de agências com funcionários

$G_{\text{count-distinct}(\text{branch_name})}(\text{works})$

works

<i>employee_name</i>	<i>branch_name</i>	<i>salary</i>
Adams	Perryridge	1500
Brown	Perryridge	1300
Gopal	Perryridge	5300
Johnson	Downtown	1500
Loreena	Downtown	1300
Peterson	Downtown	2500
Rao	Austin	1500
Sato	Austin	1600



3



16500

Agregação c/ Agrupamento

$\rho_{result(2 \rightarrow \text{sum of salary})}(\text{branch_name } G_{\text{sum(salary)}}(\text{works}))$

works

<i>employee_name</i>	<i>branch_name</i>	<i>salary</i>
Adams	Perryridge	1500
Brown	Perryridge	1300
Gopal	Perryridge	5300
Johnson	Downtown	1500
Loreena	Downtown	1300
Peterson	Downtown	2500
Rao	Austin	1500
Sato	Austin	1600

Lógica da agregação por branch_name

<i>employee_name</i>	<i>branch_name</i>	<i>salary</i>
Rao	Austin	1500
Sato	Austin	1600
Johnson	Downtown	1500
Loreena	Downtown	1300
Peterson	Downtown	2500
Adams	Perryridge	1500
Brown	Perryridge	1300
Gopal	Perryridge	5300

result

<i>branch_name</i>	<i>sum of salary</i>
Austin	3100
Downtown	5300
Perryridge	8100

Agregação c/ Agrupamento

$\rho_{result(2 \rightarrow \text{sum of salary}, 3 \rightarrow \text{max salary})}(\text{branch_name} \text{ } \mathbf{G} \text{ } \text{sum}(\text{salary}), \text{max}(\text{salary}) (\text{works}))$

works

<i>employee_name</i>	<i>branch_name</i>	<i>salary</i>
Adams	Perryridge	1500
Brown	Perryridge	1300
Gopal	Perryridge	5300
Johnson	Downtown	1500
Loreena	Downtown	1300
Peterson	Downtown	2500
Rao	Austin	1500
Sato	Austin	1600



result

<i>branch_name</i>	<i>sum_salary</i>	<i>max_salary</i>
Austin	3100	1600
Downtown	5300	2500
Perryridge	8100	5300

Agregação

- Uma vez que a agregação necessita sempre de renomear o atributo para poder ser referenciado em expressões, é comum a notação abreviada

$G_{\text{sum}(\text{salary}) \rightarrow \text{total}}(\text{works})$

$\text{branch_name } G_{\text{sum}(\text{salary}) \rightarrow \text{sum of salary}, \text{max}(\text{salary}) \rightarrow \text{max salary}}(\text{works})$

NULLs

- NULLs são uma extensão ao modelo relacional, e por conseguinte também à álgebra relacional
- NULLs levam a uma lógica de predicados de três valores: verdadeiro, falso, e desconhecido
 - Comparações ou expressões aritméticas envolvendo NULLs produzem desconhecido (incluindo $\text{NULL} = \text{NULL}$)
 - Operadores booleanos (AND, OR, NOT) envolvendo desconhecido produzem desconhecido, excepto:
 - $\text{false AND unknown} = \text{false}$
 - $\text{true OR unknown} = \text{true}$

NULLs

- NULLs afectam o comportamento de operações algébricas de forma diferente:
 - Seleção & Join:
 - $\text{NULL} = \text{NULL} \Rightarrow \text{unknown}$
 - Apenas predicados verdadeiros são selecionados
 - Projeção, União, Diferença, Interseção & Agregação:
 - $\text{NULL} = \text{NULL} \Rightarrow \text{true}$
 - NULL é tratado como qualquer outro valor da perspectiva de unicidade dos tuplos (e.g. todos os NULL são agrupados numa agregação)

Outer Join

- Operação binária que estende o conceito de Join por forma a evitar perda de informação
 - Os tuplos que não têm correspondente são incluídos no resultado, com NULLs tomando o lugar do correspondente
- Notação:
 - Left Outer Join: $r \bowtie_{\theta} s$
 - Right Outer Join: $r \ltimes_{\theta} s$
 - Full Outer Join: $r \Join_{\theta} s$
 - Tal como no Join, sem θ é considerado um “Natural” Join

Outer Join

- Left Outer Join: $r \bowtie_{\theta} s$
 - Apenas tuplos em r que não têm correspondência em s são retornados, com os atributos únicos de s substituídos por NULLs
- Right Outer Join: $r \ltimes_{\theta} s$
 - Apenas tuplos em s que não têm correspondência em r são retornados, com os atributos únicos de r substituídos por NULLs
- Full Outer Join: $r \Join_{\theta} s$
 - Ambos os anteriores são retornados
- Apenas o Full Outer Join é comutativo

Outer Join

- E.g. Left Outer Join: $professor \bowtie_{professor.id=teaches.id} teaches$

ID	name	dept	salary
1	John	Chemistry	65000
2	Jack	Physics	95000
3	Jill	Physics	82000
4	Joan	Biology	82000

ID	course
1	OCH-101
1	BCH-101
3	NPH-315



professor.ID	name	dept	salary	teaches.ID	course
1	John	Chemistry	65000	1	OCH-101
1	John	Chemistry	65000	1	BCH-101
3	Jill	Physics	82000	3	NPH-315
2	Jack	Physics	95000	NULL	NULL
4	Joan	Biology	82000	NULL	NULL

Álgebra Relacional em SQL

Interrogação de Bases de Dados

```
[WITH with_query [, ...]]
```

```
SELECT [ALL | DISTINCT [ON (expression [, ...])]]
```

```
  [* | expression [[AS] output_name] [, ...]]
```

```
  [FROM from_item [, ...]]
```

```
  [WHERE condition]
```

```
  [GROUP BY [ALL | DISTINCT] grouping_element [, ...]]
```

```
  [HAVING condition]
```

```
  [{UNION | INTERSECT | EXCEPT} [ALL | DISTINCT] select]
```

```
  [ORDER BY expression [ASC | DESC | USING operator]
```

```
    [NULLS { FIRST | LAST}] [, ...]]
```

```
  [LIMIT {count | ALL}]
```

Project, Rename, Aggregation

Cartesian Product, Joins

Select

Aggregation w/ Grouping

Select (after Aggregation)

Union, Intersection, Difference

Seleção

Select: σ

A	B	C	D

$\sigma_{balance > 500}(account)$

```
SELECT * FROM account WHERE balance > 500;
```

```
+-----+-----+-----+
| account_number | branch_name | balance |
+-----+-----+-----+
| A-201          | Brighton    | 900.00  |
| A-215          | Mianus      | 700.00  |
| A-217          | Brighton    | 750.00  |
| A-222          | Redwood     | 700.00  |
+-----+-----+-----+
```

Projeção (Generalizada)

Project: Π

A	B	C	D

$\Pi_{\text{account_number}, \text{balance} * 2}(\text{account})$

```
SELECT account_number, balance*2 FROM account;
```

```
+-----+-----+
| account_number | balance*2 |
+-----+-----+
|           A-101 |    1000.00 |
|           A-201 |    1800.00 |
|           A-215 |    1400.00 |
|           A-217 |    1500.00 |
|           A-222 |    1400.00 |
+-----+-----+
```

Unicidade dos tuplos devolvidos requer DISTINCT!

Renomear

$$\rho_{(1 \rightarrow \text{acc_no})}(\Pi_{\text{account_number}, \text{balance}}(\text{account}))$$

```
SELECT account_number AS acc_no, balance FROM account;
```

acc_no	balance
A-101	500.00
A-201	900.00
A-215	700.00
A-217	750.00
A-222	700.00

Rename de tabelas também com AS, mas só é possível quando há nested selects

Produto Cartesiano

Cartesian Product: \times

a1	b1	b1	c1
a2	b2	b3	c2
a3	b3		



a1	b1	b1	c1
a1	b1	b3	c2
a2	b2	b1	c1
a2	b2	b3	c2
a3	b3	b1	c1
a3	b3	b3	c2

account \times *depositor*

```
SELECT * FROM account, depositor;
```

```
+-----+-----+-----+-----+-----+
| account_number | branch_name | balance | customer_name | account_number |
+-----+-----+-----+-----+-----+
| A-101          | Downtown    | 500.00  | Hayes         | A-102          |
| A-102          | Perryridge  | 400.00  | Hayes         | A-102          |
| A-201          | Brighton    | 900.00  | Hayes         | A-102          |
| A-215          | Mianus      | 700.00  | Hayes         | A-102          |
| A-217          | Brighton    | 750.00  | Hayes         | A-102          |
| ...           | ...         | ...     | ...           | ...           |
| A-305          | Round Hill  | 350.00  | Turner        | A-305          |
+-----+-----+-----+-----+-----+
49 rows in set (0.00 sec)
```

Joins

(Natural) Join: ⋈

a1	b1	b1	c1
a2	b2	b3	c2
a3	b3		



a1	b1	c1
a3	b3	c2

Outer Join: ⋈_L, ⋈_R, ⋈_F

a1	b1	b1	c1
a2	b2	b3	c2
a3	b3		



a1	b1	c1
a3	b3	c2
a2	b2	NULL

account ⋈ *depositor*

```
SELECT * FROM account NATURAL JOIN depositor;
```

account ⋈_{a.account_number=d.account_number} *depositor*

```
SELECT * FROM account INNER JOIN depositor USING (account_number);  
SELECT * FROM account a INNER JOIN depositor d ON  
  (a.account_number=d.account_number);
```

⋈_L: LEFT [OUTER] JOIN

⋈_R: RIGHT [OUTER] JOIN

⋈_F: FULL [OUTER] JOIN

União

$$\Pi_{\text{account_number}} (\sigma_{\text{branch_name}=\text{"Brighton"}}(\text{account})) \cup \Pi_{\text{account_number}} (\sigma_{\text{customer_name}=\text{"Johnson"}}(\text{account}))$$

```
SELECT account_number FROM account WHERE branch_name = 'Brighton'
```

A-201, A-217

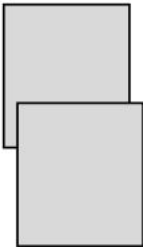
```
UNION
```

```
SELECT account_number FROM depositor WHERE customer_name = 'Johnson';
```

A-101, A-201

```
+-----+
| account_number |
+-----+
| A-201          |
| A-217          |
| A-101          |
+-----+
```

Union



Interseção

$$\Pi_{\text{account_number}}(\sigma_{\text{branch_name}=\text{"Brighton"}}(\text{account})) \cap \Pi_{\text{account_number}}(\sigma_{\text{customer_name}=\text{"Johnson"}}(\text{account}))$$

```
SELECT account_number FROM account WHERE branch_name = 'Brighton'
```

A-201, A-217

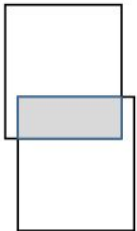
```
INTERSECT
```

```
SELECT account_number FROM depositor WHERE customer_name = 'Johnson';
```

A-101, A-201

```
+-----+
| account_number |
+-----+
| A-201          |
+-----+
```

Intersection



Diferença

$$\Pi_{\text{account_number}}(\sigma_{\text{branch_name}=\text{"Brighton"}}(\text{account})) - \Pi_{\text{account_number}}(\sigma_{\text{customer_name}=\text{"Johnson"}}(\text{account}))$$

```
SELECT account_number FROM account WHERE branch_name = 'Brighton'
```

A-201, A-217

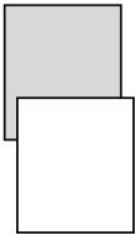
```
EXCEPT
```

```
SELECT account_number FROM depositor WHERE customer_name = 'Johnson';
```

A-101, A-201

```
+-----+
| account_number |
+-----+
| A-217          |
+-----+
```

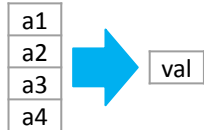
Difference



Agregação

$$\sigma_{total > 500}(\text{branch_name } G_{\text{sum(balance)} \rightarrow \text{total}}(\text{account}))$$

Aggregation: G or γ



```
SELECT branch_name, SUM(balance) AS total FROM account
GROUP BY branch_name HAVING total > 500;
```

```
+-----+-----+
| branch_name | total |
+-----+-----+
| Brighton   | 1650.00 |
| Mianus     | 700.00 |
| Redwood    | 700.00 |
+-----+-----+
```

Divisão

$$\Pi_{sid,bid}(\text{reserve}) \div \Pi_{bid}(\text{boat})$$

Division: \div

a1	b1	b1
a1	b2	b2
a2	b1	
a3	b2	



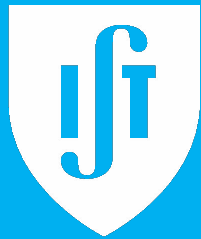
a1

?

- Não implementada por nenhum SGBD SQL
- Podemos computar seguindo a definição:
 - $r \div s = \Pi_{r \cap \neg s}(r) - \Pi_{r \cap \neg s}((s \times \Pi_{r \cap \neg s}(r)) - r)$
- Veremos mais à frente como fazer!

NULLs

- NULLs têm comportamento idêntico em SQL:
 - Seleção & Join:
 - $\text{NULL} = \text{NULL} \Rightarrow \text{unknown}$
 - Apenas predicados verdadeiros são selecionados
 - Projeção, União, Diferença, Interseção & Agregação:
 - $\text{NULL} = \text{NULL} \Rightarrow \text{true}$
 - NULL é tratado como qualquer outro valor da perspectiva de unicidade dos tuplos
 - Unicidade na Projeção em SQL requer DISTINCT



TÉCNICO LISBOA