

4. Projecto de Bases de Dados

4.1 Introdução aos SGBD - Sistemas de Gestão de Bases de Dados

4.2 Sistemas Relacionais e Linguagem SQL

4.3 Normalização Funcional de Dados para Concepção de BDr

4.4 Outros Modelos de SGBD

4.5 Conclusões e Principais Referências

Sistemas de Gestão de Bases de Dados Relacionais

Um **SGBDR** (em inglês, RDBMS) é um sistema no qual:

- Os dados são armazenados logicamente em **tabelas** (e apenas em tabelas). Fisicamente, podem ser usadas outras estruturas de armazenagem, dependentes do sistema.
- Toda a informação da base de dados é representada de uma única forma, através dos **valores dos dados**.
- Todos os valores dos dados são **atômicos** (escalares).
- Os operadores que interrogam os dados são operadores que geram novas tabelas a partir das existentes, designados **operadores relacionais**.

SQL (Structured Query Language)

Foi inicialmente desenvolvida pela *IBM* nos anos 70

A primeira implementação comercial de **SQL** foi desenvolvida pela Relational Software Inc. (hoje conhecida por *Oracle Corporation*)

Actualmente, a **SQL** é considerada um standard por todos os SGBDR.

A **SQL** é usada para formular **operações relacionais** (operações que definem e manipulam dados na forma relacional) com base no modelo relacional proposto por:

E. F. Codd em "*A Relational Model of Data for Large Shared Data Banks*", *ACM Journal*, 1970.

Álgebra Relacional

A álgebra relacional definida por Codd consiste em oito **operadores**:

- 4 operadores baseados nos operadores “tradicionais” sobre conjuntos:
 - União (*Union*)
 - Intersecção (*Intersect*)
 - Diferença (*Minus*)
 - Produto Cartesiano (*Times*)
- 4 operadores relacionais específicos:
 - Restrição (*Where*)
 - Projecção (*[]*)
 - Junção (*Join*)
 - Divisão (*Divide*)

Para que serve a Álgebra Relacional?

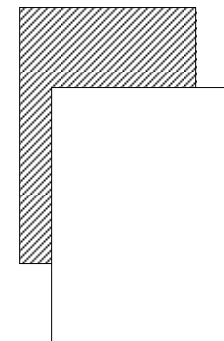
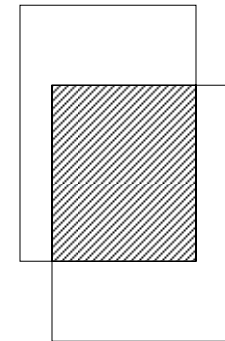
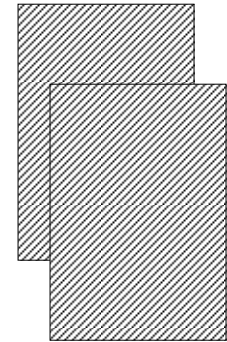
- definir os dados que deverão ser obtidos como resultado de uma **consulta**
- definir os dados que deverão ser **inseridos, alterados** ou **eliminados**
- definir os dados que deverão ser visualizados através de uma **vista**
- definir regras de **segurança**, ou seja, definir os dados sobre os quais alguma autorização é necessária
- definir requisitos de **estabilidade**, ou seja, definir os dados que irão ser a base de operações de controlo de concorrência
- definir regras de **integridade**, ou seja, definir regras específicas que a base de dados deve satisfazer

Álgebra Relacional - operadores

União: retorna uma relação (tabela) que contém todas as tuplas (linhas) de cada uma ou de ambas as relações (tabelas). O operador União é comutativo e associativo.

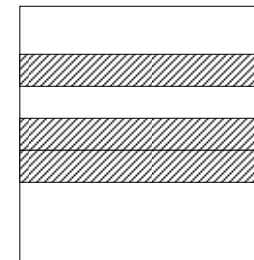
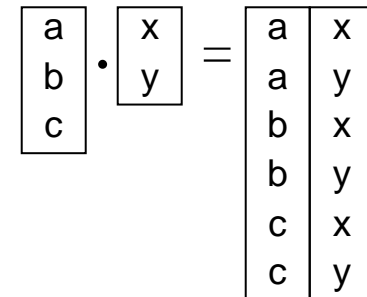
Intersecção: retorna uma tabela que contém todas as linhas comuns a ambas as tabelas. O operador Intersecção é comutativo e associativo.

Diferença: retorna uma tabela que contém as linhas da primeira mas não da segunda tabela. O operador Diferença não é comutativo nem associativo.



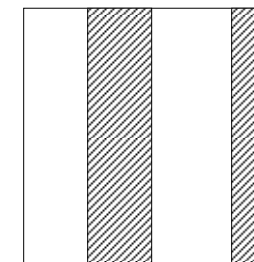
Álgebra Relacional - operadores

Produto Cartesiano: retorna uma tabela em que as linhas são o resultado de todas as possíveis combinações de cada uma das linhas das tabelas originais. As tabelas originais não têm atributos comuns. O produto cartesiano é comutativo e associativo.



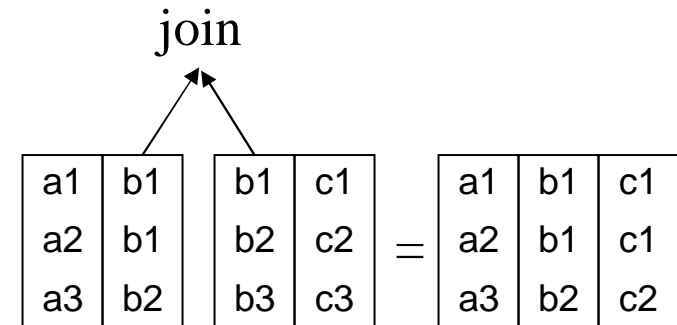
Restrição: retorna uma tabela em que todas as linhas satisfazem uma determinada condição

Projecção: retorna uma tabela com as colunas que restam de uma tabela depois de determinados atributos terem sido eliminados.

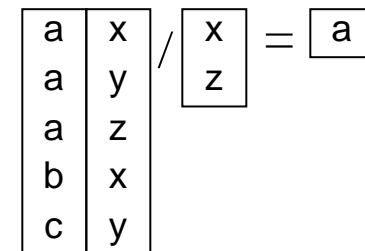


Álgebra Relacional - operadores

Junção (Natural join): retorna uma tabela com todas as possíveis linhas que são uma combinação de duas linhas, uma de cada uma das tabelas originais. As duas linhas que contribuem para uma dada combinação têm um valor comum para o(s) atributo(s) comum(ns) e esse valor aparece apenas uma única vez na linha da nova tabela. O operador Join é comutativo e associativo.



Divisão: Para duas relações (tabelas), uma binária e a outra unária, retorna uma nova tabela com todos os valores de um atributo da relação binária que são iguais (no segundo atributo) a todos os valores da relação unária.



Álgebra Relacional- fecho e compatibilidade

Fecho (Closure):

Da mesma forma que as operações algébricas sobre números retornam números e as operações sobre conjuntos retornam conjuntos, **as operações sobre relações (tabelas) retornam relações (tabelas).**

Compatibilidade:

Para podermos utilizar os operadores relacionais União, Intersecção e Diferença sobre tabelas, estas deverão ser **compatíveis**:

- devem ter o mesmo conjunto de nomes de atributos
- os atributos correspondentes devem estar definidos no mesmo domínio

União - exemplo

A

#cliente	nome	codPostal	Localidade
123	António	4500	Porto
456	Sara	1200	Lisboa

B

#cliente	nome	codPostal	Localidade
123	António	4500	Porto
789	Fernando	3600	Aveiro

União (A **UNION** B)

#cliente	nome	codPostal	Localidade
123	António	4500	Porto
456	Sara	1200	Lisboa
789	Fernando	3600	Aveiro

Intersecção - exemplo

A

#cliente	nome	codPostal	Localidade
123	António	4500	Porto
456	Sara	1200	Lisboa

B

#cliente	nome	codPostal	Localidade
123	António	4500	Porto
789	Fernando	3600	Aveiro

Intersecção (A **INTERSECT** B)

#cliente	nome	codPostal	Localidade
123	António	4500	Porto

Diferença - exemplo

A

#cliente	nome	codPostal	Localidade
123	António	4500	Porto
456	Sara	1200	Lisboa

B

#cliente	nome	codPostal	Localidade
123	António	4500	Porto
789	Fernando	3600	Aveiro

Diferença (A **MINUS** B)

#cliente	nome	codPostal	Localidade
456	Sara	1200	Lisboa

(B **MINUS** A)

#cliente	nome	codPostal	Localidade
789	Fernando	3600	Aveiro

Produto cartesiano - exemplo

A

#cliente	nome
123	António
456	Sara

B

#conta	saldo
9999	100
8888	5000

Produto (A **TIMES** B)

#cliente	nome	#conta	saldo
123	António	9999	100
123	António	8888	5000
456	Sara	9999	100
456	Sara	8888	5000

Restrição - exemplo

A

#produto	nome	cor	stock	#fornecedor
11111	tintaA	amarelo	45	123
22222	tintaB	azul	20	123
33333	tintaA	vermelho	200	456
44444	verniz	dourado	90	789

A **WHERE** #fornecedor = "123"

#produto	nome	cor	stock	#fornecedor
11111	tintaA	amarelo	45	123
22222	tintaB	azul	20	123

A **WHERE** stock < 100 AND #fornecedor = "789"

#produto	nome	cor	stock	#fornecedor
44444	verniz	dourado	90	789

Projecção - exemplo

A

#produto	nome	cor	stock	#fornecedor
11111	tintaA	amarelo	45	123
22222	tintaB	azul	20	123
33333	tintaA	vermelho	200	456
44444	verniz	dourado	90	789

A [nome]

nome
tintaA
tintaB
verniz

A [nome, cor]

nome	cor
tintaA	amarelo
tintaB	azul
tintaA	vermelho
verniz	dourado

(A WHERE stock < 100) [#fornecedor]

#fornecedor
123
789

Junção – exemplo 1 o atributo comum (#fornecedor) é chave estrangeira em A

A

#produto	cor	stock	#fornecedor
11111	amarelo	45	123
22222	azul	20	123
33333	vermelho	200	456
44444	dourado	90	789

B

#fornecedor	nome	Localidade
123	António	Porto
456	Sara	Lisboa
789	João	Aveiro

A JOIN B

#produto	cor	stock	#fornecedor	nome	Localidade
11111	amarelo	45	123	António	Porto
22222	azul	20	123	António	Porto
33333	vermelho	200	456	Sara	Lisboa
44444	dourado	90	789	João	Aveiro

Junção – exemplo 2 o atributo comum (Localidade) não é chave estrangeira

A

#produto	cor	stock	Localidade
11111	amarelo	45	Porto
22222	azul	20	Aveiro
33333	vermelho	200	Lisboa
44444	dourado	90	Aveiro

B

#fornecedor	nome	Localidade
123	António	Porto
456	Sara	Porto
789	João	Aveiro

A JOIN B

#produto	cor	stock	Localidade	#fornecedor	nome
11111	amarelo	45	Porto	123	António
11111	amarelo	45	Porto	456	Sara
22222	azul	20	Aveiro	789	João
44444	dourado	90	Aveiro	789	João

Divisão – exemplo 1

A

#fornecedor	#produto
123	11111
123	22222
123	33333
456	11111
456	22222
789	11111

B

#produto
11111

C

#produto
11111
22222

A DIVIDEBY B

#fornecedor
123
456
789

A DIVIDEBY C

#fornecedor
123
456

Os operadores **RENAME** e Atribuição Relacional

- O operador **RENAME** permite renomear atributos de uma determinada relação. Para uma dada relação, o operador **RENAME** retorna uma cópia dessa relação, na qual cada atributo tem um nome diferente.

Cientes

#cliente	nome	codPostal	Localidade
123	João	4500	Porto
456	Sara	1200	Lisboa

RENAME Cientes nome AS nomeCliente

- O operador de **Atribuição Relacional** (**:=**) permite criar uma cópia de uma relação, com um novo nome

Exemplos: T1 := Cientes WHERE nome = 'João'
 T2 := T1 JOIN Encomendas

Álgebra Relacional – Notação

Dadas duas relações A e B:

União :	A UNION B
Intersecção:	A INTERSECT B
Diferença:	A MINUS B
Produto cartesiano:	A TIMES B
Restrição:	A WHERE condição
Projecção:	A [atributo1 , atributo2 , ... atributok]
Junção:	A JOIN B
Divisão:	A DIVIDEBY B

Outros operadores

Renomeação:	A RENAME atributoX AS atributoY
Atribuição:	A := B

Exercícios

A

#fornecedor	nome	Cidade
123	António	Porto
456	Sara	Lisboa
789	João	Aveiro

B

#produto	cor	stock	Cidade
11111	amarelo	45	Porto
22222	azul	20	Aveiro
33333	vermelho	200	Lisboa

C

#fornecedor	#produto	Qtd
123	11111	200
123	22222	100
123	33333	300
456	11111	500
456	22222	200
789	11111	100

1. Nome dos fornecedores que fornecem o produto cujo #produto="22222"

T1 := A JOIN C

T2 := T1 WHERE #produto = "22222"

T3 := T2 [nome]

((A JOIN C) WHERE #produto = "22222") [nome]

Exercícios

A

#fornecedor	nome	Cidade
123	António	Porto
456	Sara	Lisboa
789	João	Aveiro

B

#produto	cor	stock	Cidade
11111	amarelo	45	Porto
22222	azul	20	Aveiro
33333	vermelho	200	Lisboa

C

#fornecedor	#produto	Qtd
123	11111	200
123	22222	100
123	33333	300
456	11111	500
456	22222	200
789	11111	100

2. Nome dos fornecedores que fornecem pelo menos um produto vermelho

OU

T1 := B WHERE cor = "vermelho"

T2 := T1 [#produto]

T3 := T2 JOIN C

T4 := T3 JOIN A

T5 := T4 [nome]

**(((B WHERE cor = "vermelho") [#produto]
JOIN C) JOIN A) [nome]**

T1 := B WHERE cor = "vermelho"

T2 := T1 JOIN C

T3 := T2 [#fornecedor]

T4 := T3 JOIN A

T5 := T4 [nome]

**(((B WHERE cor = "vermelho") JOIN C)
[#fornecedor] JOIN A) [nome]**

Exercícios

A

#forneced or	nome	Cidade
123	António	Porto
456	Sara	Lisboa
789	João	Aveiro

B

#produto	cor	stock	Localidade
11111	amarelo	45	Porto
22222	azul	20	Aveiro
33333	vermelho	200	Lisboa

C

#fornecedor	#produto	Qtd
123	11111	200
123	22222	100
123	33333	300
456	11111	500
456	22222	200
789	11111	100

- Nome dos fornecedores que fornecem todos os produtos
- Código dos fornecedores que fornecem pelo menos todos os produtos fornecidos pelo fornecedor #fornecedor = "123"
- Nome dos fornecedores que não fornecem o produto com #produto = "22222"

Exercício 3 (resolução)

A

#fornecedor	nome	Cidade
123	António	Porto
456	Sara	Lisboa
789	João	Aveiro

B

#produto	cor	stock	Localidade
11111	amarelo	45	Porto
22222	azul	20	Aveiro
33333	vermelho	200	Lisboa

C

#fornecedor	#produto	Qtd
123	11111	200
123	22222	100
123	33333	300
456	11111	500
456	22222	200
789	11111	100

3. T1 := C [#fornecedor, #produto]

T2 := B [#produto]

T3 := T1 DIVIDEBY T2

T4 := T3 JOIN A

T5 := T4 [nome]

((C [#fornecedor, #produto] DIVIDEBY B [#produto]) JOIN A) [nome]

Exercício 4 (resolução)

A

#fornecedor	nome	Cidade
123	António	Porto
456	Sara	Lisboa
789	João	Aveiro

B

#produto	cor	stock	Localidade
11111	amarelo	45	Porto
22222	azul	20	Aveiro
33333	vermelho	200	Lisboa

C

#fornecedor	#produto	Qtd
123	11111	200
123	22222	100
123	33333	300
456	11111	500
456	22222	200
789	11111	100

4. T1 := C [#fornecedor, #produto]
 T2 := C WHERE #fornecedor = "123"
 T3 := T2 [#produto]
 T4 := T1 DIVIDEBY T3
 T5 := T2 [#fornecedor]
 T6 := T4 MINUS T5

//retira o fornecedor 123

**(C [#fornecedor, #produto] DIVIDEBY ((C WHERE #fornecedor = "123") [#produto]))
 MINUS T2[#fornecedor]) [#fornecedor]**

Exercício 5 (resolução)

A

#fornecedor	nome	Cidade
123	António	Porto
456	Sara	Lisboa
789	João	Aveiro

B

#produto	cor	stock	Localidade
11111	amarelo	45	Porto
22222	azul	20	Aveiro
33333	vermelho	200	Lisboa

C

#fornecedor	#produto	Qtd
123	11111	200
123	22222	100
123	33333	300
456	11111	500
456	22222	200
789	11111	100

5. T1 := A [#fornecedor]

T2 := C WHERE #produto = "22222"

T3 := T2 [#fornecedor] //fornecedores de 22222

T4 = T1 MINUS T3 //fornecedores excepto os que fornecem 22222

T5 := T4 JOIN A

T6 := T5 [nome]

(A [#fornecedor] MINUS (C WHERE #produto = "22222") [#fornecedor] JOIN A) [nome]

Álgebra Relacional – Outros operadores

- Operadores não primitivos
- O operador JOIN revisitado
 - θ - JOIN, EQUIJOIN
 - OUTER JOIN (LEFT, RIGHT and FULL)
- O operador EXTENDS
- O operador SUMMARIZE
- Fecho transitivo

Os Operadores Primitivos

Os operadores **Intersecção**, **Junção** e **Divisão** **não são primitivos**, ou seja, podem ser definidos à custa dos restantes cinco operadores.

- A Intersecção (**INTERSECT**) pode ser definida usando a Diferença
- A Junção (**JOIN**) pode ser definida usando a Projecção, a Restrição e o Produto Cartesiano.
- A Divisão (**DIVIDEBY**) pode ser definida usando a Projecção, Produto Cartesiano e a Diferença

O conjunto dos operadores União, Diferença, Restrição, Projecção e Produto Cartesiano formam o **conjunto de operadores primitivos**.

O operador **Intersecção** revisitado

A Intersecção (**INTERSECT**) pode ser definida usando a Diferença

$$A \text{ INTERSECT } B \equiv A \text{ MINUS } (A \text{ MINUS } B)$$

ou

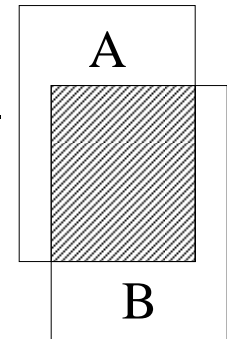
$$A \text{ INTERSECT } B \equiv B \text{ MINUS } (B \text{ MINUS } A)$$

ou ainda (preferível? Porquê?)

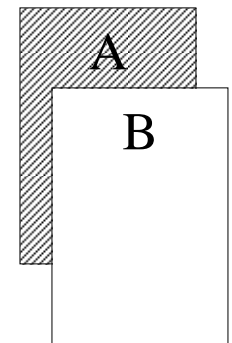
$$(A \text{ MINUS } (A \text{ MINUS } B)) \text{ UNION } (B \text{ MINUS } (B \text{ MINUS } A))$$

Nota: Neste caso, como A e B devem ser compatíveis,
 $A \text{ INTERSECT } B = A \text{ JOIN } B$

A INTERSECT B



A MINUS B



O operador **Junção** revisitado

O operador **JOIN** pode ser definido usando a Projecção, a Restrição e o Produto Cartesiano.

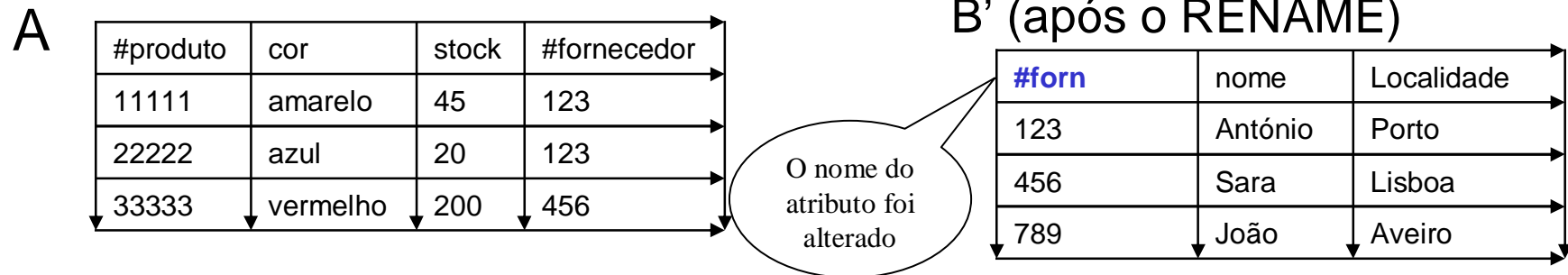
Sejam

- R uma relação com atributos $A_1, \dots, A_n, B_1, \dots, B_k$
- S uma relação com atributos $A_1, \dots, A_n, C_1, \dots, C_m$

• R **JOIN** S tem atributos $A_1, \dots, A_n, B_1, \dots, B_k, C_1, \dots, C_m$

$R \text{ JOIN } S \equiv (((A \text{ TIMES } (B \text{ RENAME } A_1 \text{ AS } A'_1, \dots, A_n \text{ AS } A'_n)) \text{ WHERE } A_1 = A'_1 \text{ AND } A_2 = A'_2 \text{ AND } A_n = A'_n) [A_1 \dots A_n, B_1, \dots, B_k, C_1, \dots, C_m])$

Exemplo – Junção



A TIMES (B RENAME #fornecedor AS #forn)

#produto	cor	stock	#fornecedor	#forn	nome	Localidade
11111	amarelo	45	123	123	António	Porto
11111	amarelo	45	123	456	Sara	Lisboa
11111	amarelo	45	123	789	João	Aveiro
22222	azul	20	123	123	António	Porto
22222	azul	20	123	456	Sara	Lisboa
22222	azul	20	123	789	João	Aveiro
33333	vermelho	200	456	123	António	Porto
33333	vermelho	200	456	456	Sara	Lisboa
33333	vermelho	200	456	789	João	Aveiro

Exemplo – Junção (cont.)

(A TIMES (B RENAME #fornecedor AS #forn)) WHERE #fornecedor = #forn

#produto	cor	stock	#fornecedor	#forn	nome	Localidade
11111	amarelo	45	123	123	António	Porto
22222	azul	20	123	123	António	Porto
33333	vermelho	200	456	456	Sara	Lisboa

**((A TIMES (B RENAME #fornecedor AS #forn))
WHERE #fornecedor = #forn)[#produto, cor, stock,#fornecedor,nome, Localidade]**

#produto	cor	stock	#fornecedor	nome	Localidade
11111	amarelo	45	123	António	Porto
22222	azul	20	123	António	Porto
33333	vermelho	200	456	Sara	Lisboa

≡ **A JOIN B**

O operador **Divisão** revisitado

A Divisão (**DIVIDEBY**) pode ser definida usando a Projecção, Produto Cartesiano e a Diferença

Seja

R uma relação com atributos $A_1, \dots, A_n, B_1, \dots, B_k$

S uma relação com atributos B_1, \dots, B_k

R DIVIDEBY S \equiv

$R [A_1, \dots, A_n] \text{ MINUS } (((R[A_1, \dots, A_n] \text{ TIMES } S) \text{ MINUS } R) [A_1, \dots, A_n])$

Exemplo - Divisão

A

#fornecedor	#produto
123	11111
123	22222
123	33333
456	11111
456	22222
789	11111

C

#produto
11111
22222

A[#fornecedor] TIMES C

#fornecedor	#produto
123	11111
123	22222
456	11111
456	22222
789	11111
789	22222

(A[#fornecedor] TIMES C) MINUS A

#fornecedor	#produto
789	22222

A [#fornecedor] MINUS (((A[#fornecedor] TIMES C) MINUS A) [#fornecedor])

#fornecedor
123
456

≡ A **DIVIDEBY** C

O operador θ -JOIN

O operador θ - JOIN é utilizado quando se pretende juntar duas tabelas usando uma condição diferente da igualdade entre os atributos comuns.

θ é uma operação binária do conjunto $\{<, <=, =, >=, >\}$

O operador θ - JOIN pode ser definido como

$(R \text{ TIMES } S) \text{ WHERE } X \theta Y$

onde R e S não têm atributos comuns, X é um atributo de R e Y é um atributo de S.

Se θ é a operação "=", o operador θ - JOIN chama-se **equijoin**.

Questão: qual a relação com a Junção Natural (JOIN) ?

Exemplo: θ -JOIN

A

#produto	cor	stock	Localidade
11111	amarelo	45	Porto
22222	azul	20	Aveiro
33333	vermelho	200	Lisboa

B

#fornecedor	nome	Cidade
123	António	Porto
456	Sara	Lisboa
789	João	Aveiro

Se $\theta = '>'$, a expressão $A \text{ TIMES } B \text{ WHERE } \text{Localidade} > \text{Cidade}$

#produto	cor	stock	Localidade	#fornecedor	nome	Cidade
11111	amarelo	45	Porto	456	Sara	Lisboa
11111	amarelo	45	Porto	789	João	Aveiro
33333	vermelho	200	Lisboa	789	João	Aveiro

temos as combinações de produtos e fornecedores onde a Localidade é maior (lexicograficamente) do que a Cidade.

OUTER JOIN

- Ao utilizar os operadores JOIN e θ - JOIN pode perder-se informação.
- Os operadores **OUTER JOIN** permitem preservar informação dos operandos.

R LEFT OUTER JOIN S (\equiv R UNION (R JOIN S))

preserva a informação do operando esquerdo. Na tabela resultado, os registos de R para os quais não existe correspondente em S mantêm-se, tomando os atributos de S correspondentes o valor nulo.

R RIGHT OUTER JOIN S (\equiv S UNION (R JOIN S))

preserva a informação do operando direito.

R FULL OUTER JOIN S (\equiv R UNION S UNION (R JOIN S))

preserva a informação de R e de S

OUTER JOIN - Exemplos

A

#produto	cor	stock	Localidade
11111	amarelo	45	Porto
22222	azul	20	Aveiro
33333	vermelho	200	Lisboa

B

Localidade	CodPost
Porto	4000
Aveiro	3900
Coimbra	1000

A **LEFT** OUTER JOIN B

#produto	cor	stock	Localidade	CodPost
11111	amarelo	45	Porto	4000
22222	azul	20	Aveiro	3900
33333	vermelho	200	Lisboa	

A **RIGHT** OUTER JOIN B

#produto	cor	stock	Localidade	CodPost
11111	amarelo	45	Porto	4000
22222	azul	20	Aveiro	3900
			Coimbra	1000

A **FULL** OUTER JOIN B

#produto	cor	stock	Localidade	CodPost
11111	amarelo	45	Porto	4000
22222	azul	20	Aveiro	3900
33333	vermelho	200	Lisboa	
			Coimbra	1000

O operador EXTEND

EXTEND A ADD exp AS Z

para uma dada relação *A*, cria uma nova relação com um cabeçalho igual ao da relação original mas com um atributo adicional, cujos valores são obtidos através do cálculo de uma expressão escalar.

A

#produto	cor	Stock	Localidade	PreçoUnit
11111	amarelo	45	Porto	5
22222	azul	20	Aveiro	10
33333	vermelho	200	Lisboa	1

EXTEND A ADD Stock * PreçoUnit AS valorStock

#produto	cor	Stock	Localidade	PreçoUnit	valorStock
11111	amarelo	45	Porto	5	225
22222	azul	20	Aveiro	10	200
33333	vermelho	200	Lisboa	1	200

O operador EXTEND - exemplo

A

#forn	nome	Cidade
123	António	Porto
456	Sara	Lisboa
789	João	Aveiro

B

#produto	cor	stock	Localidade
11111	amarelo	45	Porto
22222	azul	20	Aveiro
33333	vermelho	200	Lisboa

C

#forn	#produto	Qtd
123	11111	200
123	22222	100
123	33333	300
456	11111	500
456	22222	200
789	11111	100

**EXTEND A ADD COUNT ((C RENAME #forn AS #forn1)
WHERE #forn1= #forn) AS NumEnc**

A sub-expressão ((C RENAME #forn AS #forn1) WHERE #forn1= #forn) dá-nos as encomendas de cada fornecedor para cada fornecedor que possui encomendas.

Esta expressão pode ser substituída por **MATCHING C**.

A função agregada **COUNT** devolve o número de encomendas de cada fornecedor.

EXTEND A ADD COUNT (MATCHING C) AS NumEnc

#fornecedor	nome	Cidade	NumEnc
123	António	Porto	3
456	Sara	Lisboa	2
789	João	Aveiro	1

O operador SUMMARIZE

SUMMARIZE A **BY** (A1, A2, ..., An) **ADD** agg-exp **AS** Z

- A_1, \dots, A_n são atributos distintos de A . O resultado da expressão é uma relação composta pelos atributos $\{A_1, \dots, A_n, Z\}$
- os valores dos dados são todos os registos x onde x é uma projecção de A sobre A_1, \dots, A_n com um novo atributo Z .
- O valor de Z é calculado através da expressão $agg-exp$ calculada sobre todos os atributos de A que têm os mesmos valores para A_1, \dots, A_n que o registo x .
- O conjunto de atributos A_1, \dots, A_n não pode incluir Z e $agg-exp$ não deve referir-se a Z .

O operador SUMMARIZE - exemplo

C

#forn	#produto	Qtd
123	11111	200
123	22222	100
123	33333	300
456	11111	500
456	22222	200
789	11111	100

SUMMARIZE C BY (#produto) ADD SUM (qtd) AS qtdTotal

#produto	qtdTotal
11111	800
22222	300
33333	300

Quantidade total encomendada de cada produto

O operador SUMMARIZE - exemplo

B

#produto	cor	stock	Localidade
11111	amarelo	45	Porto
22222	azul	20	Aveiro
33333	vermelho	200	Lisboa

C

#forn	#produto	Qtd
123	11111	200
123	22222	100
123	33333	300
456	11111	500
456	22222	200
789	11111	100

SUMMARIZE (B JOIN C) BY (Localidade) ADD COUNT AS nEnc

B JOIN C

#forn	#produto	Qtd	cor	stock	Localidade
123	11111	200	amarelo	45	Porto
456	11111	500	amarelo	45	Porto
789	11111	100	amarelo	45	Porto
123	22222	100	azul	20	Aveiro
456	22222	200	azul	20	Aveiro
123	33333	300	vermelho	200	Lisboa

Localidade	nEnc
Porto	3
Aveiro	2
Lisboa	1

Fecho Transitivo

A Álgebra Relacional **não permite** calcular expressões que envolvam relações recursivas entre registos da mesma tabela - **Fecho transitivo**

Exemplos

- todos os subprodutos de um produto
- todos os subordinados de um trabalhador
- todos os descendentes de uma pessoa

#produto	#subproduto
11111	22222
22222	33333
33333	44444

trabalhador	subordinado
joão	pedro
pedro	antónio
antónio	rui
pedro	miguel

pessoa	filho
joão	ana
joão	maria
ana	francisco
maria	tomás