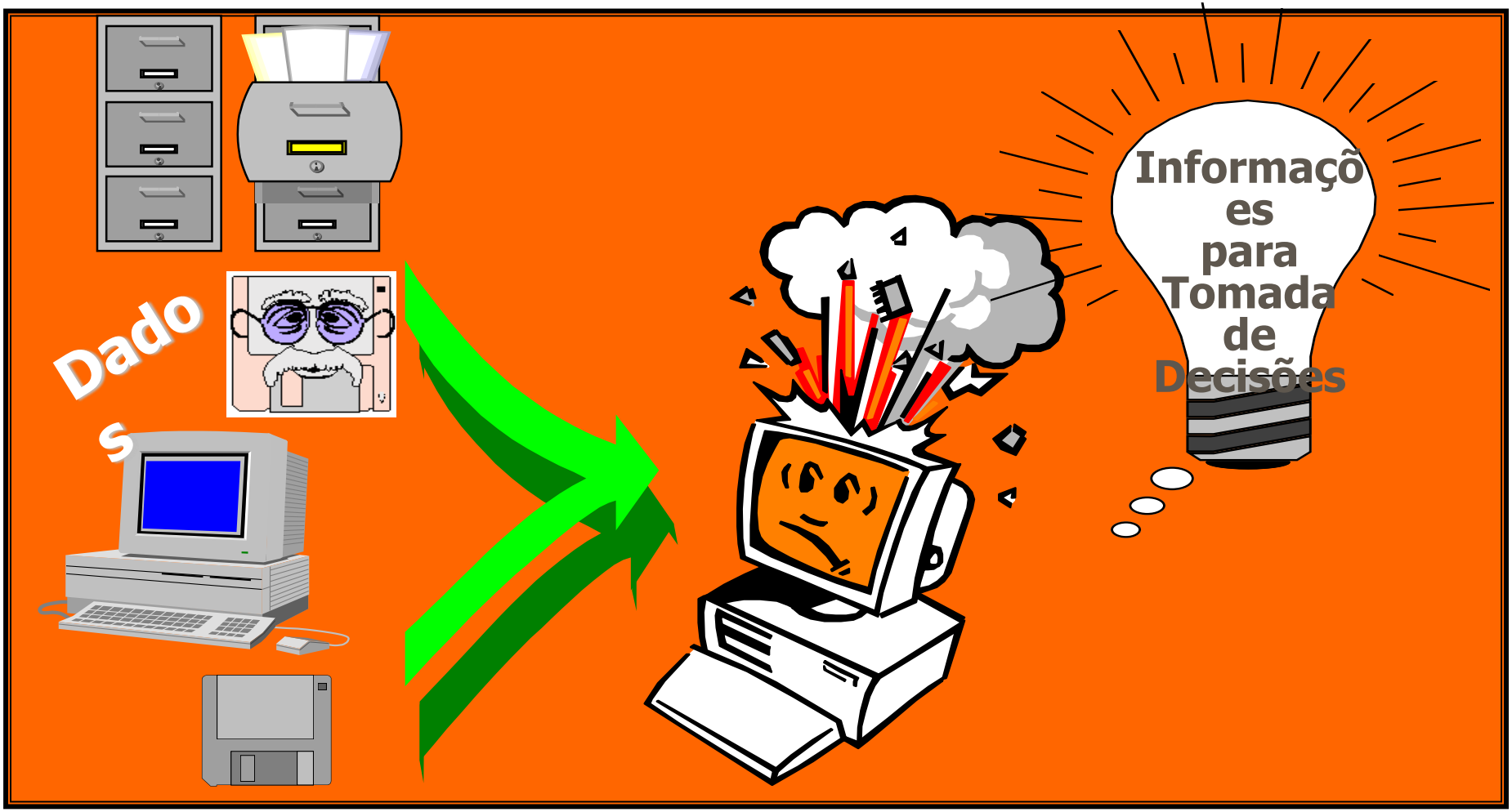
Bancos de Dados



Deborah Ribeiro Carvalho
2019

Álgebra Relacional
Operações básicas

- Básicas
  - Seleção (σ) - seleciona um subconjunto de linhas de uma relação
  - Projeção (π) – apaga colunas desnecessárias de uma relação
  - Produto cartesiano (X) – permite combinar duas relações
  - União (U) - tuplas na relação 1 e na relação 2
  - Diferença (-) – tuplas na relação 1 mas não na relação 2
  - Renomeação (ρ) – renomeia tabela
  - Atribuição (←) – Atribui valores a variaveis
- Derivadas
  - Junção, interseção, divisão

- Produz uma nova relação contendo um subconjunto vertical da relação argumento, sem duplicações

$\pi_{\text{lista\_atributos}}$ (relação argumento)

- lista de atributos
- os atributos são separados por vírgula

- relação
- resultado de alguma operação da álgebra relacional

select nome_professor from PROFESSOR;

- Seleciona tuplas da relação argumento que satisfaçam à condição de seleção

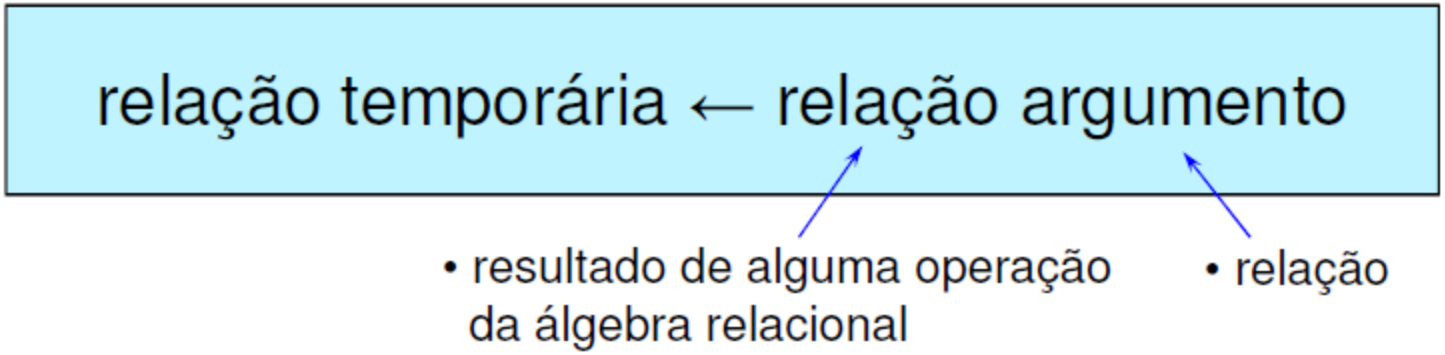
$$\sigma_{\text{condição\_seleção}} (\text{ relação argumento})$$

- pode envolver operadores de comparação ($=, <, \leq, >, \geq, \neq$)
- pode combinar condições usando-se $\wedge, \vee, \neg$

- relação
- resultado de alguma operação da álgebra relacional

.... Where nome_professor = ' joao das coves';

– associa uma relação argumento a uma relação temporária
– permite o uso da relação temporária em expressões subseqüentes



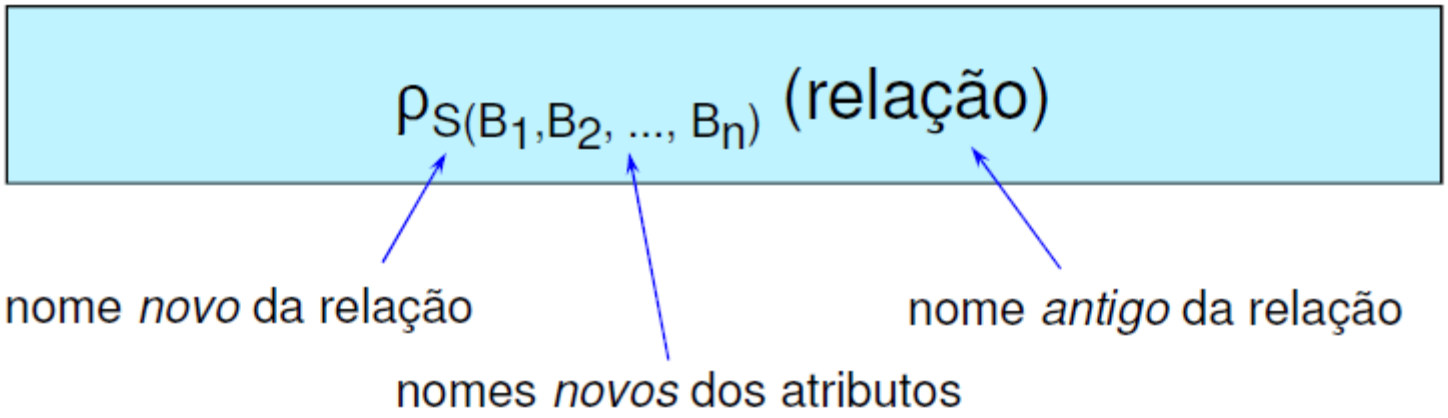
```
select a.nome_aluno, p.nome_professor
 from
     (select ad.codigo_aluno, d.codigo_professor, count(*) as qtas
      from alunos_disciplinas ad
      inner join disciplinas d on d.codigo_disciplina = ad.codigo_disc
      group by ad.codigo_aluno, d.codigo_professor) as consulta
 inner join ALUNOS a on a.codigo_aluno = consulta.codigo_aluno
 inner join PROFESSOR p on p.codigo_professor = consulta.codigo_professor
 where consulta.qtas > 1;
```

- Liste o número e o nome de todos os clientes que possuam saldo inferior a R$ 200,00 e que morem na Rua X.

$$\pi_{nro\_cli,\ nome\_cli}\ (\sigma_{saldo\_dev < 200,00\ \wedge\ end\_cli = \text{“Rua X”}}\ (cliente))$$

- Usando atribuição
  - $temp \leftarrow \sigma_{saldo\_dev < 200,00\ \wedge\ end\_cli = \text{“Rua X”}}\ (cliente)$
  - $\pi_{nro\_cli,\ nome\_cli}\ (temp)$

- Renomeia
  - nome da relação
  - nomes dos atributos da relação
  - nome da relação e nomes dos atributos



```
select f.fun_nome as funcionario, g.fun_nome as gerente
    from tbl_funcionarios as f, tbl_funcionarios as g
    where f.cod_gerente = g.fun_id;
```

- Notação: ρ_S (R) ou $\rho_{S(A1,A2, ...)}$ (R)
  - Renomeia R para S ou renomeia R para S com atributos renomeados A1, A2, ...
- Entrada: Tabela (R)
- Propósito: redefinir nome tabelas / ou colunas num contexto
- Saída: Tabela renomeada com mesmas linhas de R
- Usada para
  - Útil para auto-relacionamentos, onde precisamos fazer a junção de uma tabela com ela mesma, e nesse caso cada versão da tabela precisa receber um nome diferente da outra.
  - Cria colunas idênticas numa junção natural

- Unárias
  - seleção
  - projeção
  - renomear

operam sobre uma única relação

- Binárias
  - produto cartesiano
  - união
  - diferença de conjuntos
  - intersecção de conjuntos
  - junção natural
  - divisão

operam sobre duas relações

- Notação: R X S
- Entrada: Tabela (R) e Tabela (S)
- Propósito: gera combinações de linhas das duas tabelas
- Saída: Para cada linha r em R e cada linha sem S, gerar a tupla rs

```
select p.nome_professor, d.nome_disciplina
        from PROFESSORES p, DISCIPLINAS d,
              PROFESSORES_DISCIPLINAS r
        where r.cod_prof = p.codigo_professor
        and   r.cod_disc = d.codigo_disciplina;
```

Álgebra Relacional
Produto cartesiano

nro_cli	nome_cli	end_cli	saldo	cliente. cod_vend	vendedor. cod_vend	nome_vend
1	Márcia	Rua X	100,00	1	1	Adriana
1	Márcia	Rua X	100,00	1	2	Roberto
2	Cristina	Avenida 1	10,00	1	1	Adriana
2	Cristina	Avenida 1	10,00	1	2	Roberto
3	Manoel	Avenida 3	234,00	1	1	Adriana
3	Manoel	Avenida 3	234,00	1	2	Roberto
4	Rodrigo	Rua X	137,00	2	1	Adriana
4	Rodrigo	Rua X	137,00	2	2	Roberto

grau: número de atributos de cliente + número de atributos de vendedor

número de tuplas: número de tuplas de cliente * número de tuplas de vendedor

Exemplo

Matricula m

SID	CID
142	CPS114
142	CPS116
183	CPS114
898	CPS116

Estudante e

SID	Nome	Idade	CR
142	Roberto	23	5,8
183	Andrea	20	8,1
898	Carlos	18	9,2
453	Bruno	20	7,5

SID	Nome	Idade	CR	SID	CID
142	Roberto	23	5,8	142	CPS114
142	Roberto	23	5,8	142	CPS116
183	Andrea	20	8,1	183	CPS114
898	Carlos	18	9,2	898	CPS116

- Notação: R ⋈ S
- Entrada: Tabela (R) e Tabela (S)
- Propósito: relaciona linhas das tabelas
  - Reforça a igualdade de seus atributos
  - Elimina 1 cópia dos atributos comuns
- Saída: Para cada linha r em R e cada linha sem S, gerar a tupla rs se, e somente se,atenderem a condição p

```
-- nome do professor e disciplina
-- sem inner join

select p.nome_professor, d.nome_disciplina
            from PROFESSORES p, DISCIPLINAS d, PROFESSORES_DISCIPLINAS r
            where r.cod_prof = p.codigo_professor
            and   r.cod_disc = d.codigo_disciplina;



-- nome do professor e disciplina
-- com inner join

select p.nome_professor, d.nome_disciplina
            from PROFESSORES p
  inner join PROFESSORES_DISCIPLINAS r on r.cod_prof = p.codigo_professor
  inner join DISCIPLINAS as d on d.codigo_disciplina = r.cod_disc;
```

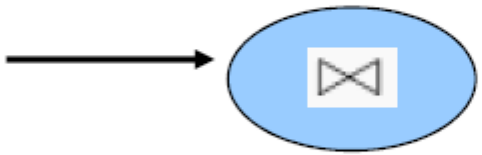
Exemplo

Matricula m

SID	CID
142	CPS114
142	CPS116
183	CPS114
898	CPS116

Estudante e

SID	Nome	Idade	CR
142	Roberto	23	5,8
183	Andrea	20	8,1
898	Carlos	18	9,2
453	Bruno	20	7,5



SID	Nome	Idade	CR		CID
142	Roberto	23	5,8		CPS114
142	Roberto	23	5,8		CPS116
183	Andrea	20	8,1		CPS114
898	Carlos	18	9,2		CPS116

A álgebra relacional empresta da teoria de conjuntos quatro operadores:
União, Intersecção, Diferença e Produto Cartesiano

Sintaxe da **União:** <tabela> 1 ∪ <tabela> 2

Sintaxe da **Intersecção**: <tabela> 1 ∩ <tabela> 2

Sintaxe da **Diferença**: <tabela> 1 – <tabela> 2

Nos três casos, a operação possui **duas tabelas** como **operando**.E as tabelas devem ser **compatíveis:**

- possuir o mesmo número de colunas;
- o mesmo domínio para cada posição da lista de atributos;
- quando os nomes das colunas forem diferentes, adota- se os nomes das colunas da primeira tabela.

- Notação: R U S
  - R e S devem ter o mesmo esquema
- Entrada: Tabela (R) e Tabela (S)
- Propósito: gera linhas de acordo com um critério
- Saída: Contém todas as linhas de R e de S
  - O esquema é o mesmo das tabelas de entrada
  - Duplicidade é eliminada

```
select * from TABLE_A
UNION
select * from TABLE_B
```

- Notação: R - S
  - R e S devem ter o mesmo esquema
- Entrada: Tabela (R) e Tabela (S)
- Propósito: gera linhas de acordo com um critério
- Saída: Contém todas as linhas de R e que não são encontradas em S
  - O esquema é o mesmo das tabelas de entrada

```
select A_KEY from TABLE_A
where A_KEY not in (select A_KEY from TABLE_B)
```

- Notação: R ∩ S
  - R e S devem ter o mesmo esquema
- Entrada: Tabela (R) e Tabela (S)
- Propósito: gera linhas de acordo com um critério
- Saída: Contém todas as linhas de R que são encontradas em S também
  - O esquema é o mesmo das tabelas de entrada
  - R – (R – S) ou S – (S – R) ou R ⋈ S

```
select KEY from TABLE_A
where TABLE_A.KEY in (select TABLE_B.KEY from TABLE_B)
```

- Notação: R ÷ S
- Entrada: Tabela (R) e Tabela (S)
  - Seja grau a medida de atributos de mesmo nome
    - R tem grau (“m”+”n”)
    - S tem grau “n”
- Propósito: gera linhas de acordo com um critério
- Saída: atributos de S cujos valores associam-se com todos os valores de R
  - Grau “m”

```
select distinct A_KEY from TABLE_C C
where not exists
        ( select B_KEY from TABLE_B B where not exists
                ( select * from TABLE_C CC
                        where A.A_KEY = CC.A_KEY
                                and B.B_KEY = CC.B_KEY ))
```

- Para aquelas consultas que não podem ser resolvidas simplesmente através da álgebra relacional, introduz-se um conjunto de funções agregadas
- Funções comumente aplicadas a conjuntos de dados são: Média, Máximo, Mínimo, Soma, Contador

• **Funções agregadas:** são aquelas que, quando aplicadas, tomam uma coleção de valores e retornam um valor simples como resultado.

trabalhador_integral

nome_empregado	nome_agencia	salario
José	NOH-1	5000
Ana	POA-1	4800
Flávia	SAL-1	3200
Maria	POA-1	6500

- **função sum:** descobrir a soma total dos salários de todos os empregados de tempo integral.

$\text{sum}_{\text{salario}}$ (trabalhador_integral)

sum salario
14505

-**função count:** descobrir o número de agências existentes na tabela de tempo integral.

$\text{count}_{\text{nome\_agencia}}$ (trabalhador_integral)

count nome_agencia
4

$\text{count-distinct}_{\text{nome\_agencia}}$ (trabalhador_integral)

count-distinct nome_agencia
3

- função avg: descobrir a média dos salários.

$\text{avg}_{\text{salario}}$ (trabalhador_integral)

avg salario
3626.25

- função min: descobrir o menor salário.

$\text{min}_{\text{salario}}$ (trabalhador_integral)

min salario
3200

- função max: descobrir o maior salário.

$\text{max}_{\text{salario}}$ (trabalhador_integral)

max salario
6500

Álgebra Relacional
Resumo

Símbolo	*Operação*	*Sintaxe*	*Tipo*
σ	Seleção / Restrição	$\sigma_{\text{condição}}$ (Relação)	Primitiva
π	Projeção	$\pi_{\text{expressões}}$ (Relação)	Primitiva
∪	União	Relação1 ∪ Relação2	Primitiva
∩	Intersecção	Relação1 ∩ Relação2	Adicional
-	Diferença de conjuntos	Relação1 - Relação2	Primitiva
x	Produto cartesiano	Relação1 x Relação2	Primitiva
|x|	Junção	Relação1 |x| Relação2	Adicional
÷	Divisão	Relação1 ÷ Relação2	Adicional
ρ	Renomeação	ρ_{nome} (Relação)	Primitiva
←	Atribuição	variável ← Relação	Adicional

Álgebra Relacional
Resumo

SÍMBOLO	OPERAÇÃO	SINTAXE	TIPO
<-----	Atribuição	Variável <----- Relação	Primitiva
σ	Seleção (Select)	σ <condicao de elecao>(Relação)	Primitiva
π	Projeção (Project)	π <lista de atributos>(Relação)	Primitiva
∪	União (Union)	(Relação 1) ∪ (Relação 2)	Primitiva
∩	Interseção (Intersection)	(Relação 1) ∩ (Relação 2)	Adicional
–	Diferença (Difference)	Sintaxe: (Relação 1) – (Relação 2)	Primitiva
X	Produto Cartesiano (Product)	(Relação 1) X (Relação 2)	Primitiva
|x|	Junção (Join)	(Relação 1) |x| <condição de junção> (Relação 2)	Adicional
÷	Divisão (Divide)	(Relação 1) ÷ (Relação 2)	Adicional

- **Formato:** $RelResultado = \sigma_{predicado}(RelEntrada)$
- Ex.:Selecione as tuplas da relação EMPRÉSTIMOS para quais o nome da agência é "Perryridge"

$$R = \sigma_{\text{e-agência='Perryridge'}}(\text{EMPRÉSTIMOS})$$

```
SELECT *
FROM EMPRESTIMOS
WHERE E_AGENCIA = 'PERRYRIDGE'
```

– Formato: $RelResult = \pi_{colunas\ a\ copiar}(RelEntrada)$

– Ex.: Obter uma tabela que relacione os clientes do banco com as agências onde fizeram empréstimos:

$$Res = \pi_{e\text{-}agencia,\ e\text{-}nome}(EMPRÉSTIMOS)$$

```
SELECT E_AGENCIA, E_NOME
FROM EMPRESTIMOS
```

- É possível compor operações mais complexas da álgebra relacional através do aninhamento de operações mais simples
- Exemplo: listar os nomes dos clientes que fizeram empréstimos superiores a 1200:

$$\text{RelResult} = \pi_{\text{e-nome}}(\sigma_{\text{e-valor}>1200}(\text{EMPRÉSTIMOS}))$$

```
SELECT DISTINCT E_NOME
FROM (SELECT E_NOME
        FROM EMPRESTIMOS
         WHERE E_VALOR >1200)
```

- A operação de seleção que gera uma relação como resultado pode ser usada como relação de entrada para a operação de projeção

Em SQL proporciona um mecanismo para renomear tanto atributos quanto relações, usando a cláusula **as**, como segue:

```
select Atributo as Novo_nome
from relação;
```

- Exemplo: listar o nome dos clientes que moram em Rye e fizeram empréstimo de menos de 1000.

  a)RelResult1= CLIENTES x EMPRÉSTIMOS

  b)RelResult2= $\sigma_{\text{c-name = e-name}}$ (RelResult1)

  c)RelResult3= $\sigma_{\text{e-valor<1000 and c-cidade = Rye}}$ (RelResult2)

```
SELECT C_NOME
FROM CLIENTES, EMPRESTIMOS
WHERE C_NOME = E_NOME AND
E_VALOR < 1000 AND C_CIDADE = 'RYE'
```

Faça uma lista com todos os números de projetos nos quais esteja envolvido algum empregado cujo último nome seja 'Smith'; ou como empregado, ou como gerente do departamento que controla o projeto.

```
(select distinct pnumero
from projeto, departamento, empregado
where Dnum = Dnumero and GerSSN = ssn and
unome = 'Smith')
union
(select distinct pnumero
from trabalha_em, projeto, empregado
where pnumero = Pno and essn = SSN and
unome = 'Smith');
```

Obs: Diferença entre conjuntos → except
Interseção de conjuntos → intersect