Capítulo 3

Álgebra relacional I

Neste capítulo e no que segue, é apresentada uma das linguagens teóricas de consulta, a álgebra relacional.

O presente capítulo apresenta as operações básicas da álgebra relacional, juntamente com duas formas de expressar consultas, uma textual e outra gráfica.

Além disso, é detalhado o tratamento de campos vazios e a lógica de três valores empregada nas linguagens relacionais.

3.1 Álgebra, árvore sintática e álgebra relacional

Na Matemática, uma álgebra é definida como sendo um conjunto de objetos juntamente com um conjunto de operações definidas sobre os objetos deste conjunto. Uma álgebra bem conhecida é a Aritmética. Nela, os objetos são os números e as operações são a soma, a subtração, a multiplicação e a divisão, entre outras.

Fecho algébrico

Uma propriedade interessante de uma operação é ser *fechada*. Uma operação é fechada quando tanto seus operandos quanto seu resultado pertencem ao conjunto de objetos sobre o qual a álgebra está definida. Exemplificando, na Aritmética, os operandos e os resultados são sempre números.

Esta propriedade permite escrever expressões que combinam vários operadores, já que o resultado de uma operação pode ser usado como operando para a próxima. Como exemplo, considere uma expressão aritmética, como

$$((5 - 2) * 3) + (8 \div 2)$$

onde o resultado de uma operação serve de operando para outra.

Representação em árvore

Uma conseqüência do fato de as operações de uma álgebra serem fechadas é que suas expressões podem ser representadas na forma de uma árvore, denominada *árvore sintática*. Nesta árvore, cada nó não-folha representa uma operação, sendo seus operandos representados pelos nós filhos. Exemplificando, a Figura 3.1 ⇒ p. 84 apresenta a árvore sintática para a expressão aritmética acima. Quando

a expressão algébrica for grande e complexa, ela pode ser mais facilmente compreendida através da observação da correspondente árvore sintática.

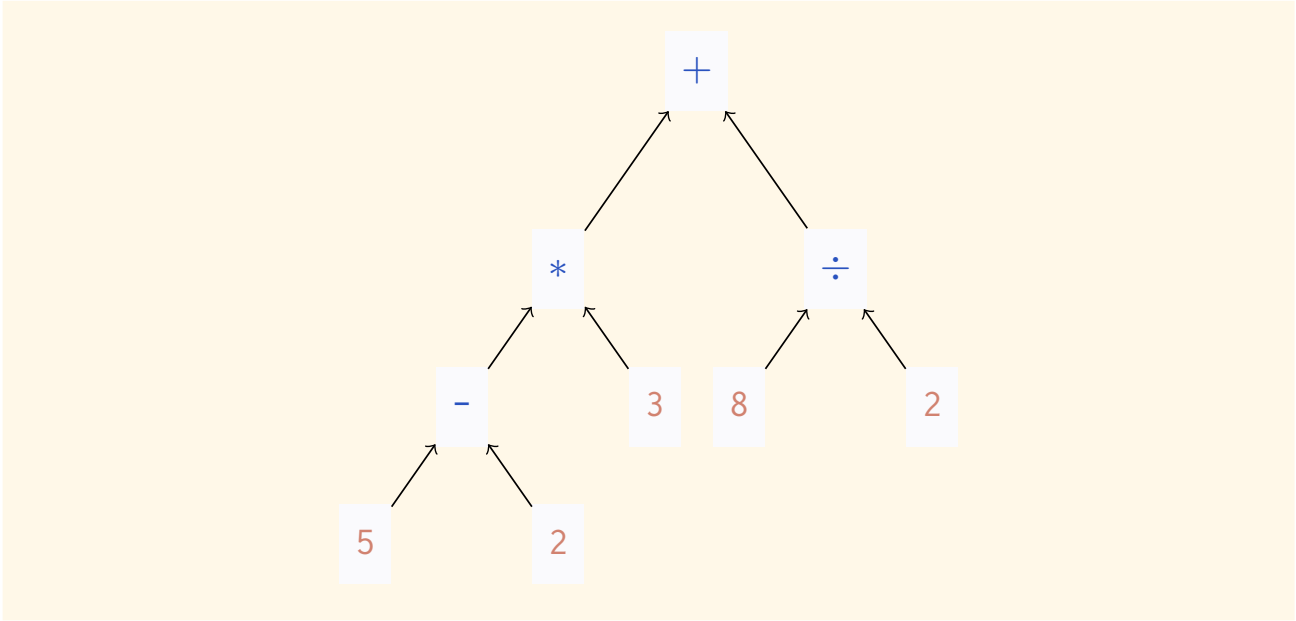


Figura 3.1: Árvore sintática para a expressão $((5 - 2) * 3) + (8 \div 2)$

Neste livro, para facilitar a compreensão das operações de álgebra relacional, vamos usar ambas as representações, tanto a representação textual quanto a árvore sintática equivalente.

Álgebra relacional

Codd, o criador da abordagem relacional, usou o conceito de álgebra para definir uma das linguagens da abordagem relacional.

Codd considerou que o resultado de uma consulta pode ser representado por uma tabela e que uma consulta pode ser vista como uma série de operações sobre tabelas. Assim, definiu uma álgebra que opera sobre tabelas. Uma operação desta álgebra relacional tem como operandos tabelas, resultando igualmente em uma tabela.

Como tabelas são conjuntos no sentido matemático da palavra,

para conceber a álgebra relacional, Codd buscou inspiração em uma álgebra de conjuntos. As operações da álgebra relacional podem ser classificadas em:

- Operações adaptadas das operações tradicionais de álgebra de conjuntos: *união*, *intersecção*, *diferença* e *produto cartesiano*.
- Operações unárias, específicas da álgebra relacional, para selecionar linhas (operação de *seleção*) e selecionar colunas (operação de *projeção*) de uma tabela.
- Uma operação de *renomeação*, que serve para definir novos nomes para tabelas ou colunas.

Por tratar-se de uma linguagem apenas teórica, sem implementação direta em computador, e para manter as consultas como expressões compactas, Codd empregou letras gregas e símbolos usuais da álgebra de conjuntos para os operadores, ao invés de empregar palavras-chave, como é usual em linguagens de programação. Para compatibilidade com a literatura, empregamos os mesmos símbolos nesta apresentação.

Representação textual x representação em árvore

Conforme já mencionado, ao longo do livro, cada consulta de álgebra relacional é representada tanto textualmente quanto em árvore.

Do ponto de vista didático, considero a representação em árvore mais adequada. Principalmente em consultas maiores, ela explicita a sequência das operações, sem que exigir que o leitor faça o reconhecimento da consulta, procurando parênteses, como é o caso da representação textual. Além disso, a representação em árvore guarda semelhança com um plano de execução (ver Seção 15.2 ⇒ p. 676), o que deve facilitar a compreensão da relação entre o plano de consulta e a consulta da qual ele é derivado.

Por outro lado, deve-se reconhecer que o restante da literatura na área de banco de dados utiliza exclusivamente a representação textual.

Como o livro sempre apresenta as duas alternativas, fica com o leitor a decisão de escolher a representação que mais lhe convém, a representação em árvore, mais didática, ou a representação textual, mais difundida.

Seleção

O operador usado para a *seleção*[1] ("*selection*"em Inglês) é a letra grega Sigma (σ), que corresponde à letra "s" em nosso alfabeto. A operação tem a seguinte sintaxe:

```
seleção

σ condiçãoSel ( tabela )
```

Nesta definição:

- `tabela` é o nome de uma tabela do banco de dados ou uma expressão de álgebra relacional.
- `condiçãoSel`, chamada de condição de seleção, é uma expressão lógica que, avaliada sobre uma linha da `tabela`, resulta em *falso* ou *verdadeiro*[2].

O resultado da operação de seleção é uma nova tabela que contém aquelas linhas da tabela operando (`tabela`) que satisfazem a condição de seleção (`condiçãoSel`).

Exemplo 3.1: (banco de dados de embarques – Apêndice A ⇒ p. 690)
Obter as peças que são da cidade `'Rio'` *e cujo peso excede* 15.

Esta consulta envolve somente a tabela peca, filtrando algumas linhas de mesma. Para resolve-la, aplica-se o operador

[1] Na definição original da álgebra relacional, a operação de seleção tinha o nome de *restrição*. Entretanto, a maior parte dos autores preferiu a denominação *seleção*, que usamos neste texto.

[2] Como é mostrado ao final deste capítulo, na realidade, a condição de seleção pode resultar em *falso*, *verdadeiro* e *desconhecido*. Para simplificar a discussão, começamos considerando apenas os dois valores lógicos usuais.

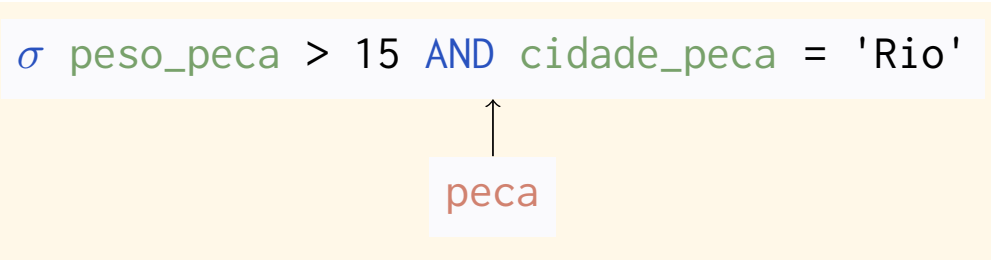
de seleção sobre esta tabela.

```
σ peso_peca > 15 AND cidade_peca = 'Rio'
    (peca)
```

Esta expressão de álgebra resulta em uma tabela contendo todas linhas da tabela peca cujo peso excede 15 e cuja cidade é 'Rio'. Se considerarmos o conteúdo do banco de dados que é apresentado na Figura 2.11 ⇒ p. 55, o resultado desta consulta é a tabela abaixo:

cod_peca	nome_peca	peso_peca	cor_peca	cidade_peca
P3	Mancal	25	Vermelho	Rio

Conforme mencionado na Seção 3.1 ⇒ p. 83, vamos mostrar cada consulta de duas formas, através da expressão na forma textual vista acima e através de sua árvore sintática:



Condição de seleção

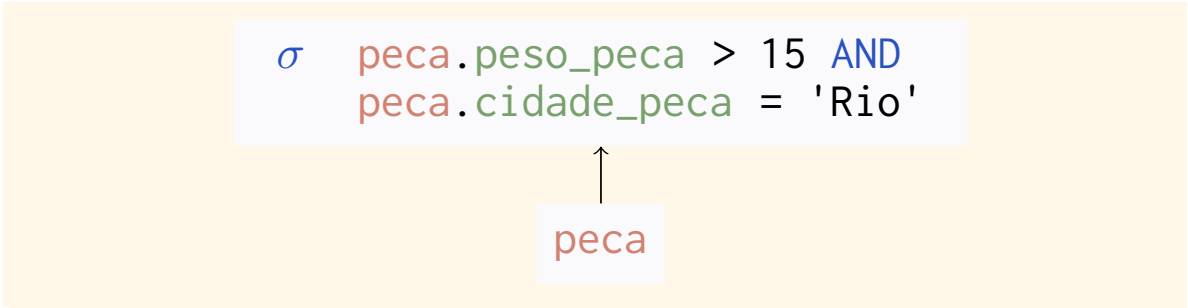
A condição de seleção (condiçãoSel) é uma expressão lógica (expressão que resulta em falso ou verdadeiro) como aquelas que aparecem em linguagens convencionais de programação (p.ex.: Java ou C).

A condição de seleção deve ser avaliável sobre uma única linha da tabela sobre a qual a seleção opera. Isto significa que, como operandos da condição de seleção, somente podem ser usados campos de uma linha ou constantes. Como operadores, podem ser usados os

operadores lógicos `AND`, `OR` e `NOT`, os operadores relacionais `>`, `<`, `>=`, `<=`, e `<>`, bem como os operadores aritméticos usuais (`+`, `-`, `*`, `/`, ...).

Ao referenciar um campo de uma tabela, pode-se usar simplesmente o nome da coluna, como no exemplo acima, ou então, usar o nome da coluna prefixado pelo nome da tabela, como no exemplo a seguir.

Exemplo 3.2: (banco de dados de embarques – Apêndice A ⇒ p. 690)
Obter as peças que são da cidade `'Rio'` *e cujo peso excede* 15.
(Enunciado idêntico ao do Exemplo 3.1 ⇒ p. 87).




```
σ peca.peso_peca > 15 AND
  peca.cidade_peca = 'Rio'
     (peca)
```

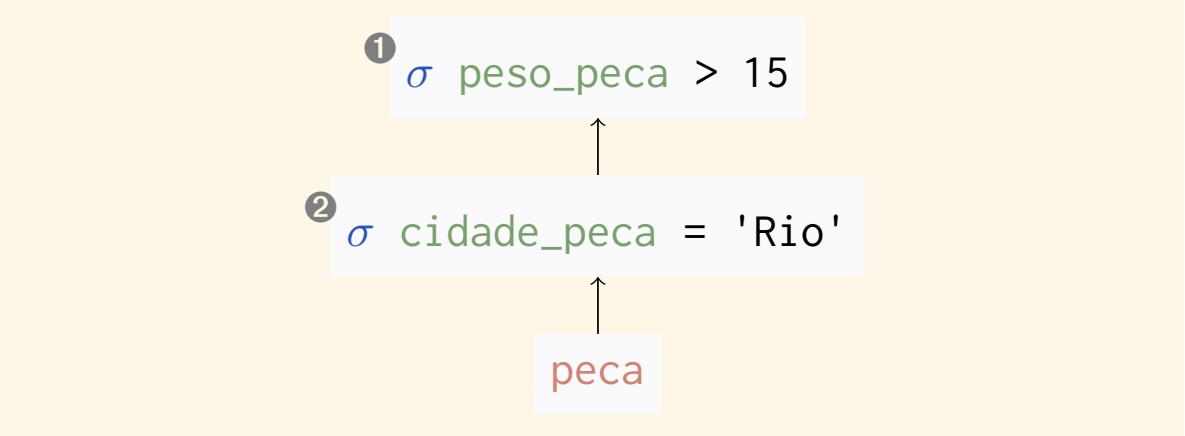
A única diferença desta solução em relação a anterior (Exemplo 3.1 ⇒ p. 87) é que aqui os nomes das colunas são qualificados com o nome da respectiva tabela.

Esta forma de referenciar colunas é útil quando várias tabelas são combinadas em uma consulta, como é mostrado mais adiante neste capítulo.

Expressão de álgebra relacional como operando

O termo `tabela` pode ser tanto o nome de uma tabela do banco de dados, como nos exemplos acima, quanto uma nova expressão de álgebra relacional.

Exemplo 3.3: (banco de dados de embarques – Apêndice A ⇒ p. 690)
Obter as peças que são da cidade `'Rio'` *e cujo peso excede* 15.
(Enunciado idêntico ao do Exemplo 3.1 ⇒ p. 87)




```
1    σ peso_peca > 15
2      (σ cidade_peca = 'Rio'
3         (peca)
4      )
```

Esta expressão envolve duas operações de seleção. A seleção mais interna (operação ❷ - linha 2)[a]. Já a operação mais externa (operação ❶ - linha 1) recebe como operando o resultado da operação mais interna. Obviamente, o resultado desta consulta é idêntico ao da consulta anterior (Exemplo 3.2 ⇒ p. 89). Ambas estas consultas são equivalentes.

[a] Ao longo da apresentação de álgebra relacional, sempre que for necessário fazer referência a uma parte de uma consulta, vamos rotular o nó cor-

> respondente à operação na árvore sintática com um número. Muitas vezes, para fazer referência a uma operação, vamos usar uma frase na forma 'operação *xx* - linha *yy*" onde *xx* é o número do nó na árvore e *yy* é o número da linha correspondente na expressão algébrica textual.

Exercício 3.1*:* (banco de dados de embarques – Apêndice A ⇒ p. 690) Escreva uma consulta em álgebra relacional[3] que obtenha todas linhas de fornecedor cujo status seja diferente de sete e cuja cidade seja `'Curitiba'`.
(*Solução* ⇒ p. 140)

Exercício 3.2*:* (banco de dados acadêmico – Apêndice B ⇒ p. 695) Escreva uma consulta em álgebra relacional que obtenha as linhas de `disciplina` que correspondem a disciplinas com mais que quatro créditos.
(*Solução* ⇒ p. 141)

[3] Ao resolver os exercícios de álgebra relacional, você pode usar tanto a forma textual quanto a correspondente árvore sintática. Nos primeiros exercícios, pode ser uma boa idéia resolver a consulta em ambas as formas, até dominá-las.

Consulta é conceitual

Observando as consultas dos exemplos 3.2 e 3.3, o leitor mais preocupado com questões de desempenho poderá estar se perguntando: Qual destas duas consultas consultas tem o melhor desempenho? À primeira vista, a segunda alternativa (Exemplo 3.3 ⇒ p. 90) deveria ter um desempenho pior, já que parece exigir duas varreduras de tabela (uma que percorre a tabela peca, obtendo as peças de Rio, e uma segunda varredura, que percorre o resultado da primeira, obtendo as peças cujo peso excede 15).

Entretanto, uma consulta em uma linguagem relacional é apenas uma especificação conceitual do que deve ser obtido e não um plano de execução a ser seguido literalmente pelo SGBD. Como veremos na Seção 15.2 ⇒ p. 676, ao receber uma consulta para execução, o SGBD procura a melhor forma de executá-la, com o objetivo de otimizar o desempenho global do sistema.

Resumindo, ao escrever consultas, a preocupação central deve estar em expressar corretamente o desejado e não em atingir melhor desempenho, exceção feita a raros casos em que o programador seja um especialista no SGBD em questão e domine perfeitamente todos detalhes de como consultas são executadas neste SGBD.

3.2 Projeção

Enquanto que a operação de seleção destina-se a obter linhas de uma tabela, a operação de projeção tem por função obter colunas de uma tabela.

O operador usado para a *projeção* ("*projection*" em Inglês) é a letra grega Pi (π), que corresponde à letra "p" em nosso alfabeto. A operação tem a seguinte sintaxe:

```
projeção

π coluna [ , ... ] ( tabela )
```

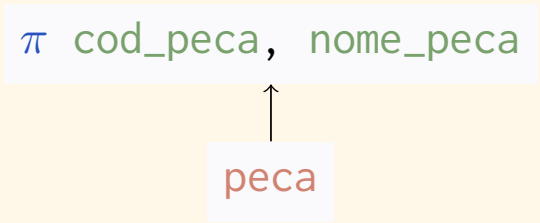
```
coluna

[
  nomeColuna |
  nomeTabela.nomeColuna |
  expressãoColuna
]
```

Nesta definição, `expressãoColuna` corresponde a uma expressão que computa um valor a partir de colunas do resultado (ver Exemplo 3.7 ⇒ p. 97).

O resultado da projeção é uma nova tabela obtida pela extração de colunas da tabela operando.

Para um primeiro exemplo, vamos considerar que os termos `coluna` sejam simplesmente nomes de coluna.

Exemplo 3.4: (banco de dados de embarques – Apêndice A ⇒ p. 690)
Obter os códigos e os nomes de todas peças.




```
π cod_peca, nome_peca (peca)
```

O resultado desta consulta é uma tabela que contém os códigos e os nomes de todas peças que aparecem na tabela de peças. Se considerarmos o conteúdo do banco de dados que é apresentado na Figura 2.11 ⇒ p. 55, o resultado desta consulta é a tabela abaixo:

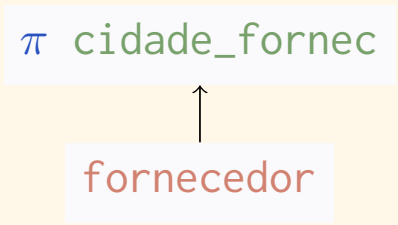
cod_peca	nome_peca
P1	Parafuso
P2	Arruela
P3	Mancal
P4	Eixo
P5	Motor

Duplicatas são eliminadas

Conforme visto na Página 32, uma tabela é um conjunto (no sentido matemático da palavra) de linhas. Portanto, uma linha não pode aparecer mais de uma vez na tabela. Isto significa que a operação de projeção, além de eliminar colunas, elimina também linhas duplica-

das do resultado[4].

Exemplo 3.5: (banco de dados de embarques – Apêndice A ⇒ p. 690)
Obter as cidades em que há fornecedores




```
π cidade_fornec (fornecedor)
```

Considerando o conteúdo do banco de dados que é apresentado na Figura 2.11 ⇒ p. 55, o resultado da consulta é a seguinte tabela:

cidade_fornec
Porto Alegre
Curitiba
Rio
<N>

Pela eliminação de duplicatas, cidades em que há vários fornecedores constam uma única vez do resultado. Um exemplo é a cidade `'Rio'`.

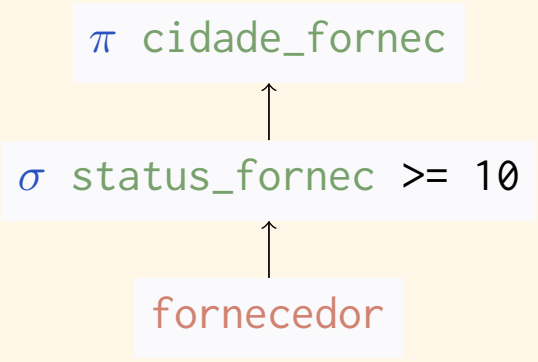
Expressão de álgebra relacional como operando

Como na seleção e nas demais operações de álgebra relacional, o operando da projeção não está restrito a um nome de tabela, mas

[4] A restrição de uma tabela não conter duplicatas refere-se à versão de álgebra relacional aqui apresentada. Na linguagem SQL, tabelas podem conter duplicatas.

pode ser qualquer expressão de álgebra relacional.

Exemplo 3.6: (banco de dados de embarques – Apêndice A ⇒ p. 690)
Obter as cidades em que há fornecedores com status maior ou igual a `10`.




```
π cidade_fornec
    (σ status_fornec >= 10
        (fornecedor)
    )
```

Neste caso, a projeção opera sobre o resultado de uma seleção.

Exercício 3.3*:* (banco de dados de embarques – Apêndice A ⇒ p. 690)
Escreva uma consulta em álgebra relacional que obtenha o código de peça, a data de embarque e a quantidade embarcada para cada embarque do fornecedor de código `'F1'`.

(*Solução* ⇒ p. 142)

Exercício 3.4*:* (banco de dados de embarques – Apêndice A ⇒ p. 690)
Escreva uma consulta em álgebra relacional que obtenha as diferentes datas nas quais houve embarques do fornecedor de código `'F1'`.

(*Solução* ⇒ p. 143)

Exercício 3.5*:* (banco de dados acadêmico – Apêndice B ⇒ p. 695)
Escreva uma consulta em álgebra relacional que, para cada disciplina com mais que quatro créditos, obtenha o código de seu departamento, o seu número de disciplina bem como o seu nome.

(*Solução* ⇒ p. 144)

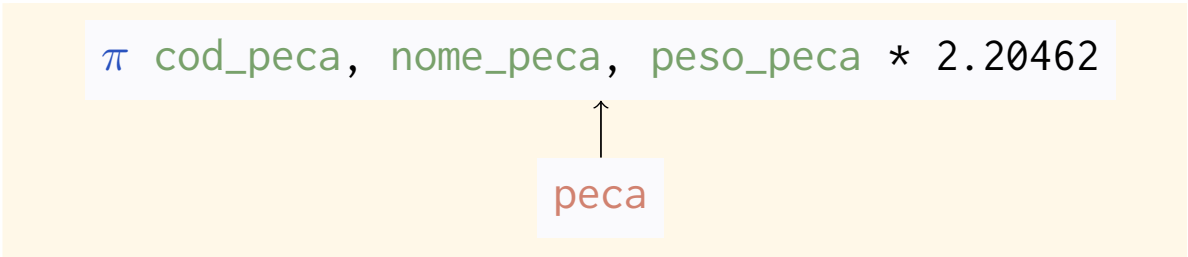
Exercício 3.6*:* (banco de dados acadêmico – Apêndice B ⇒ p. 695)
Escreva uma consulta em álgebra relacional que obtenha os diferentes nomes de disciplinas que têm mais que quatro créditos.

(*Solução* ⇒ p. 145)

Expressão de coluna

Nos exemplos vistos até aqui, as colunas que vão para o resultado são sempre especificadas através de seu nome. Entretanto, como visto na definição coluna ⇒ p. 93, colunas do resultado podem também ser definidas por uma expressão envolvendo constantes e nomes de colunas.

Exemplo 3.7: (banco de dados de embarques – Apêndice A ⇒ p. 690)
Obter uma tabela com três colunas, contendo o código, o nome, e o peso em libras de cada peça. A coluna peso_peca *da tabela* peca *contém o peso em quilogramas. O peso em libras é igual ao peso em quilogramas multiplicado por* 2,20462.



```
π cod_peca,
  nome_peca,
  peso_peca*2.20462
     (peca)
```

Neste caso, a terceira coluna não possui nome. Na Seção 3.5 ⇒ p. 114, é mostrada a operação de renomeação, que permite atribuir nomes a colunas no resultado de uma consulta.

3.3 Operações de conjunto

As operações de seleção e projeção são unárias, isto é, tem uma única tabela como operando. Nesta seção, são apresentadas operações binárias, que servem para combinar linhas de duas tabelas. Especificamente, são vistas as operações *união*, *interseção* e *diferença*, que Codd tomou emprestado da álgebra de conjuntos. Genericamente, estas operações têm a denominação de *operações de conjunto*.

operaçãoConjunto

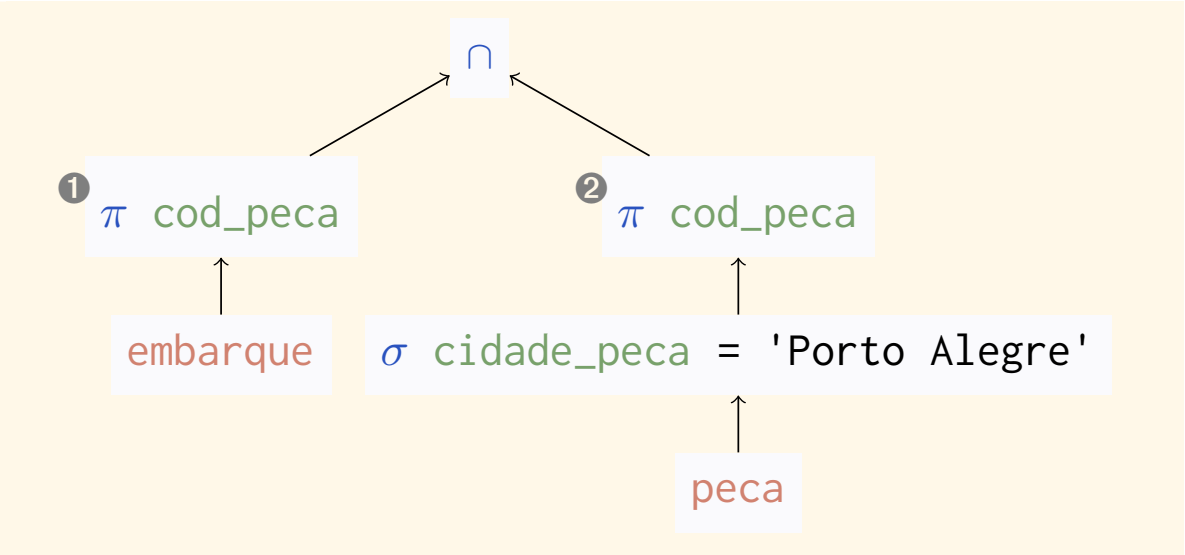
$\texttt{tabela}_1 \{ \cup \mid \cap \mid - \} \texttt{tabela}_2$

Os operadores são os mesmos de teoria de conjuntos:

- A *união* ($\cup$) tem como resultado o conjunto das linhas que aparecem em uma *ou* outra tabela.
- A *interseção* ($\cap$) tem como resultado o conjunto de linhas que aparecem em *ambas* as tabelas.
- A *diferença* ($-$) tem como resultado o conjunto das linhas que aparecem na primeira tabela ($\texttt{tabela}_1$) e não aparecem na segunda tabela ($\texttt{tabela}_2$).

Exemplo 3.8: (banco de dados de embarques – Apêndice A ⇒ p. 690)
Obter os códigos das peças cuja cidade é `'Porto Alegre'` *e que constam em embarques.*

A consulta em álgebra relacional pode ser representada pela árvore sintática abaixo:



A sub-árvore ❶ obtém os códigos das peças que têm embarques. Já a sub-árvore ❷ obtém os códigos das peças de Porto Alegre. A operação de interseção (∩) obtém os códigos das peças que encontram-se nestes dois conjuntos e, portanto, corresponde ao resultado.

Expressa de forma textual, a consulta fica como segue:

```
1    (π cod_peca
2        (embarque)
3    )
4    ∩
5    (π cod_peca
6        (σ cidade_peca = 'Porto Alegre'
7          (peca)
8        )
9    )
```

Nesta expressão, as linhas 1-3 correspondem a sub-árvore ❶ da árvores sintática e as linhas 5-9 correspondem à sub-árvore ❷.

Considerando o conteúdo do banco de dados de embarques apresentado na Figura 2.11 ⇒ p. 55, o resultado da consulta é a tabela abaixo:

cod_peca
P1
P2

Compatibilidade para operação de conjunto

Diferentemente da álgebra de conjuntos, onde as operações estão definidas sobre quaisquer dois conjuntos, na álgebra relacional, é necessário que as tabelas operando atendam a certos requisitos. Antes de discutir estes requisitos, um exemplo de duas tabelas incompatíveis é mostrado.

Exemplo 3.9: Considere as tabelas `professor` e `titulacao` mostradas na Figura 3.2 ⇒ p. 101. Qual seria a tabela resultante da união destas duas tabelas? Como as linhas de ambas

professor

cod_prof	cod_depto	cod_tit	nome_prof
1	INF01	4	Souza
2	INF01	2	Antunes
3	INF01	4	Macedo
4	INF01	3	Machado

titulacao

cod_tit	nome_tit
1	Graduado
3	Mestre
4	Doutor

Figura 3.2: Tabelas incompatíveis para operações de conjuntos

> tabelas têm estruturas diferentes, não há como incluí-las em uma mesma tabela. Recorde que, conforme a definição de tabela (Seção 2.3 ⇒ p. 30), todas as linhas de uma tabela devem ter o mesmo número de colunas.

Os requisitos para que duas tabelas sejam *compatíveis para operação de conjunto* são os que seguem:

- As duas tabelas devem ter o mesmo número de colunas.
- O domínio da i-ésima coluna de uma tabela deve ser igual ao domínio da i-ésima coluna da outra tabela.

Por convenção, definiu-se que os nomes das colunas da tabela resultante da operação são os nomes das colunas da primeira tabela operando.

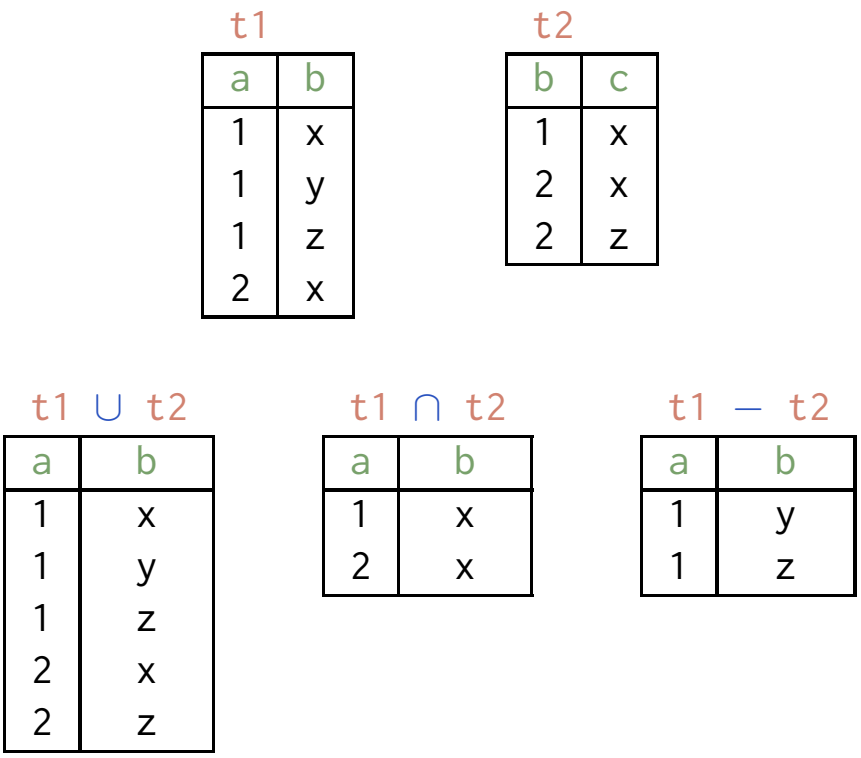
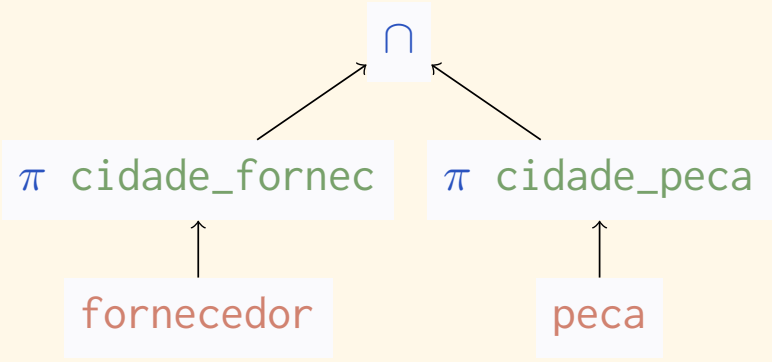


Figura 3.3: Exemplos de operações de conjuntos

Exemplo 3.10: A Figura 3.3 ⇒ p. 102 apresenta alguns exemplos de aplicação das operações de conjunto sobre as tabelas t1 e t2 mostradas na mesma figura. Estas tabelas são compatíveis para as operações de conjuntos.

Observe que, para que duas tabelas sejam compatíveis, não é exigido que elas tenham os mesmos nomes de colunas. Apenas as posições das colunas são relevantes. Assim, os nomes das colunas da primeira tabela (t1) somente são usados para definir os nomes das colunas do resultado. Na definição de compatibilidade e também na definição de quais colunas devem ser comparadas, apenas a posição da coluna é usada. No exemplo, são comparadas as colunas t1.a com t2.b (primeira coluna de cada tabela) e t1.b com t2.c (segunda coluna de cada tabela).

Exemplo 3.11: (banco de dados de embarques – Apêndice A ⇒ p. 690)
Obter as cidades em que há tanto uma peça quanto um fornecedor.




```
(π cidade_fornec (fornecedor))
   ∩
(π cidade_peca (peca))
```

O resultado desta consulta é uma tabela com uma coluna, denominada `cidade_fornec`, contendo as cidades em que há um fornecedor *e* uma peça. Considerando o conteúdo do banco de dados de embarques apresentado na Figura 2.11 ⇒ p. 55, o resultado desta consulta é a tabela abaixo:

cidade_fornec
Porto Alegre
Rio
<N>

Exercício 3.7*:* (banco de dados de embarques – Apêndice A ⇒ p. 690)
Usando álgebra relacional, escreva uma consulta que obtenha todas cidades cadastradas no banco de dados. Em outras palavras, deseja-se obter as cidades que aparecem na tabela `peca` ou na tabela `fornecedor`.

(*Solução* ⇒ p. 146)

Exercício 3.8*:* (banco de dados de embarques – Apêndice A ⇒ p. 690) Usando álgebra relacional, escreva uma consulta que obtenha os códigos das peças para as quais não existem embarques. Em outras palavras, deseja-se obter os códigos das peças que aparecem na tabela `peca`, mas não aparecem na tabela `embarque`.

(*Solução* ⇒ p. 147)

Exercício 3.9*:* (banco de dados de embarques – Apêndice A ⇒ p. 690) Usando álgebra relacional, escreva uma consulta que obtenha os códigos das peças para as quais ao menos uma das seguintes afirmativas é válida:

- A peça tem ao menos um embarque em `'2000-01-12'`.
- O peso da peça é maior ou igual a cinco.

(*Solução* ⇒ p. 148)

Exercício 3.10*:* (banco de dados acadêmico – Apêndice B ⇒ p. 695) Usando álgebra relacional, escreva uma consulta que obtenha os códigos dos departamentos que possuem ao menos uma disciplina com seis créditos e ao menos um professor cuja titulação tem código igual a quatro.

(*Solução* ⇒ p. 150)

3.4 Produto cartesiano

Na álgebra de conjuntos, o produto cartesiano de dois conjuntos é o conjunto de todos pares que podem ser formados tomando cada elemento do primeiro conjunto como primeiro elemento do par e cada elemento de segundo conjunto como segundo elemento do par.

Exemplificando, se

$$A = \{a_1, a_2, a_3\}$$

e

$$B = \{b_1, b_2\}$$

então:

$$A \times B = \{\langle a_1, b_1\rangle, \langle a_2, b_1\rangle, \langle a_3, b_1\rangle, \langle a_1, b_2\rangle, \langle a_2, b_2\rangle, \langle a_3, b_2\rangle\}$$

Na álgebra relacional, o produto cartesiano de duas tabelas é o conjunto de todas linhas formadas pela concatenação de cada linha da primeira tabela com cada linha da segunda tabela.

O operador é semelhante ao produto em outras álgebras:

produtoCartesiano

$$\texttt{tabela}_1 \times \texttt{tabela}_2$$

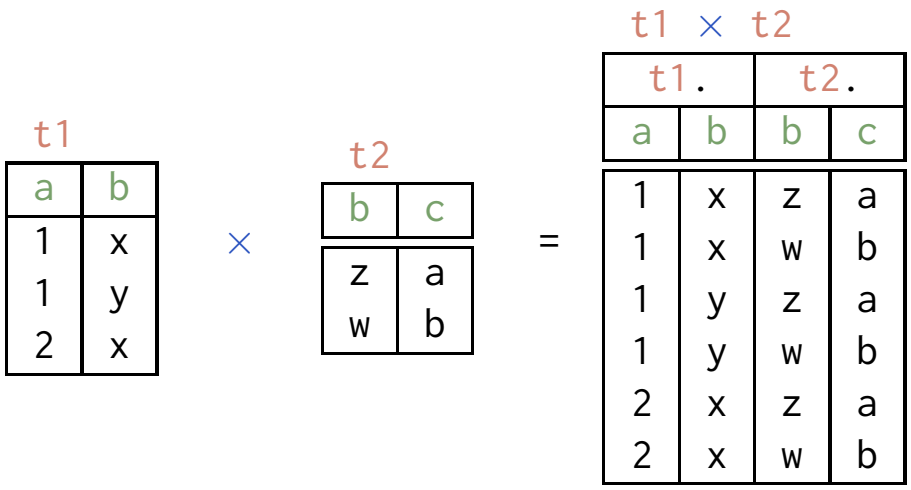


Figura 3.4: Resultado do produto cartesiano

Nomeação de colunas no resultado

O resultado de um produto cartesiano é mostrado na Figura 3.4 ⇒ p. 105. Na tabela resultante, os nomes das colunas são prefixados pelo nome da tabela da qual se originam. Quando o nome da coluna é único no resultado, este prefixo é opcional. Assim, no caso do exemplo, as colunas são denominadas, respectivamente:

1. Primeira coluna: coluna a ou coluna t1.a (o nome de coluna a aparece somente na tabela t1).

2. Segunda coluna: coluna `t1.b` (o nome `b` é ambíguo, pois aparece em ambas as tabelas – o prefixo com o nome da tabela é requerido).

3. Terceira coluna: coluna `t2.b` (o nome `b` é ambíguo, pois aparece em ambas as tabelas – o prefixo com o nome da tabela é requerido).

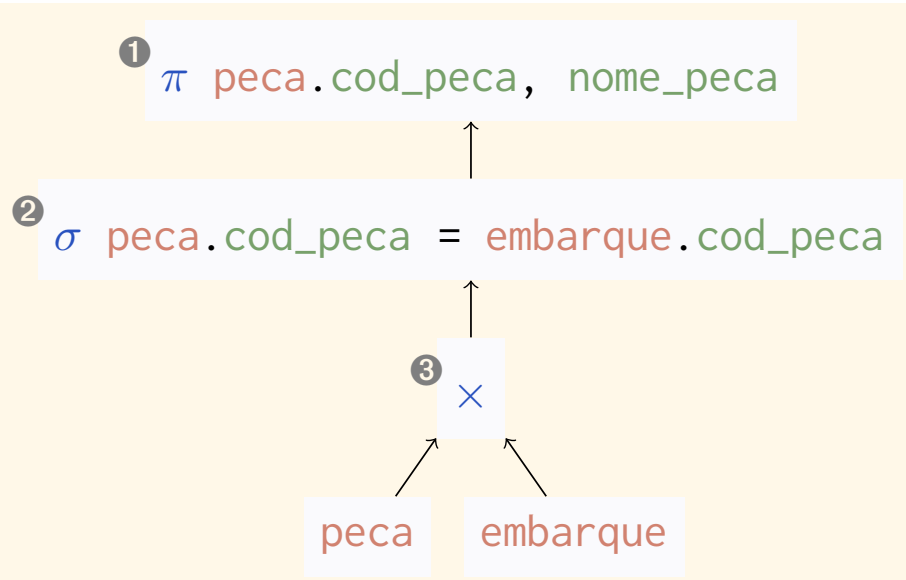
4. Quarta coluna: coluna `c` ou coluna `t2.c` (o nome de coluna `c` aparece somente na tabela `t2`).

Produto cartesiano e seleção

São raras as aplicações nas quais o produto cartesiano é empregado isoladamente. Normalmente, ele é usado em combinação com alguma outra operação, muito frequentemente em combinação com a operação de seleção. Como esta combinação de produto cartesiano e seleção ocorre com frequência na prática, para ela foi criada uma operação específica, chamada de *junção*, que é explicada no capítulo que segue.

Exemplos de combinações de produto cartesiano e seleção são mostrados a seguir.

> **Exemplo 3.12:** (banco de dados de embarques – Apêndice A ⇒ p. 690)
> *Obter o código e o nome de cada peça para a qual exista um embarque.*




```
1  π peca.cod_peca, nome_peca
2     (σ peca.cod_peca = embarque.cod_peca
3        (peca × embarque)
4     )
```

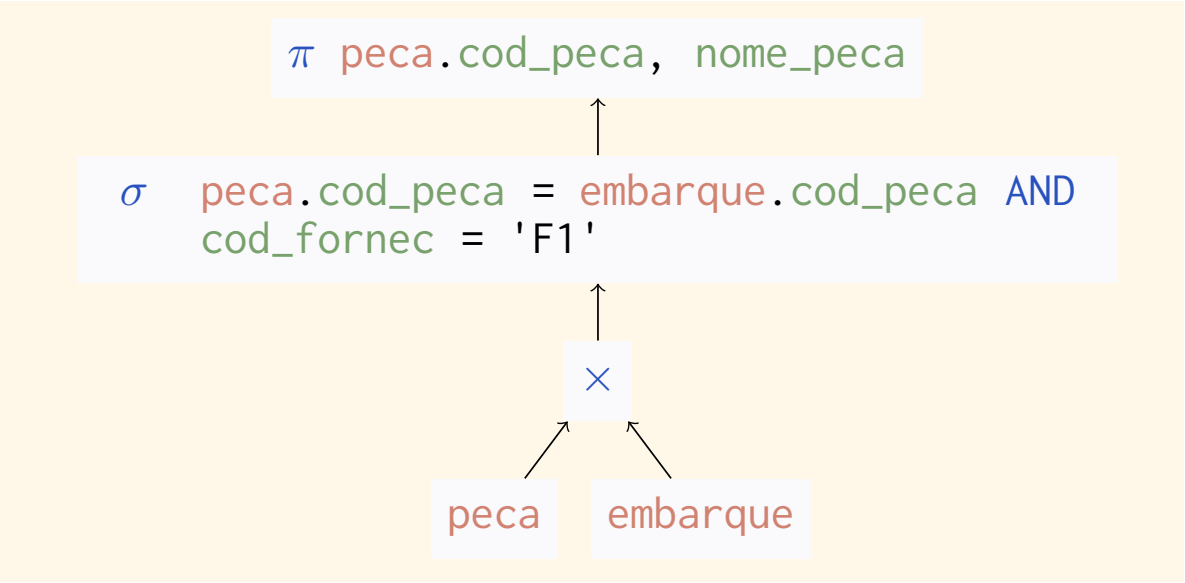
A operação ❸ (linha 3) é o produto cartesiano das tabelas peca e embarque.

A seleção ❷ (linha 2) especifica que, do resultado do produto cartesiano, devem ser selecionadas apenas aquelas linhas que casam uma peça com um de seus embarques. Isto significa que peças para as quais não há embarques, não aparecem no resultado desta seleção.

Finalmente, na operação ❶ (linha 1), a projeção obtém as colunas que contém o código e nome de cada peça.

Observe a forma através da qual são referenciadas as colunas. No resultado do produto cartesiano, aparecem duas colunas denominadas cod_peca. Portanto, o nome desta coluna deve ser prefixado com o nome da tabela de origem. Já a coluna denominada nome_peca aparece somente na tabela peca e portanto não necessita ser prefixada com o nome da tabela.

Exemplo 3.13: (banco de dados de embarques – Apêndice A ⇒ p. 690)
Obter os códigos e nomes das peças que têm pelo menos um embarque feito pelo fornecedor de código `'F1'`.




```
π peca.cod_peca, nome_peca
   (σ peca.cod_peca =
       embarque.cod_peca AND
      cod_fornec = 'F1'
       (peca ×
        embarque
       )
   )
```

Comutatividade e associatividade

Na álgebra de conjuntos, o produto cartesiano tem as propriedades de *associatividade* e *comutatividade*. A comutatividade quer dizer que

$$a \times b = b \times a$$