Uma "expressão do cálculo relacional de tuplas" tem a forma geral:

$\{t \mid F(t)\}$

onde se deve ler: "o conjunto de tuplas t tal que F(t) é verdadeiro".

O resultado de uma expressão do CRT é, portanto, uma relação contendo as tuplas t satisfazendo a condição ou "fórmula" F(t) que é uma expressão booleana construída de forma a ser detalhada a seguir. Uma expressão do CRT é também chamada de consulta do CRT.

Na prática é comum explicitar a relação R de onde provêm as tuplas t e uma projeção como resultado desejado da consulta, ou seja, o formato abaixo para uma expressão do CRT é comumente usado:

$\{t.a, t.b, \ldots, t.k \mid t \in R \textit{ and } F(t)\}$

F(t) é uma expressão booleana composta de átomos e operadores e definida recursivamente da seguinte forma:

(i) $s \in S$ é um átomo (cujo valor é verdadeiro se s pertence a S e falso caso contrário)

(ii) $s.a\ \theta\ t.b$ é um átomo (θ é o operador de comparação visto na álgebra relacional)

(iii) $s.a\ \theta\ c$ é um átomo (c é uma constante do mesmo domínio que a)

(iv) um átomo é uma fórmula.

(v) se F e G são fórmulas então *F and G*, *F or G* e *not F* são fórmulas (com o significado usual).

(vi) $\exists s \in S(F(s))$ é uma fórmula e deve ser lida como: "existe pelo menos uma tupla s pertencente a S tal que a fórmula F(s) é verdadeira". Exemplo:

$\exists s \in S(s.nome\ =\ 'Paulo')$

é verdadeiro se existe pelo menos uma tupla de S para a qual o atributo nome é igual a 'Paulo'.

(vii) $\forall s \in S(F(s))$ é uma fórmula e deve ser lida como "para toda tupla s pertencente a S a fórmula F(s) é verdadeira". Exemplo:

$\forall s \in S(s.salario > 1000)$

é verdadeiro se para todas as tuplas de S o atributo salário tem valor maior que 1000.

(viii) se F é uma fórmula então (F) também é uma fórmula.

(ix) ordem de precedência para avaliação de uma fórmula, de maior para menor:
θ , $\exists$ e $\forall$ *not*, *and*, *or*

A seguinte equivalência de expressões é bastante útil para passar de consultas usando o operador $\forall$ para usar $\exists$ e vice versa:
$\forall s \in S(F(s)) \equiv$ *not* $\exists\ s \in S(notF(s))$
que quer dizer, em palavras: "para todo s pertencente a S tal que F(s) é verdadeiro", então "não existe s pertencente a S tal que F(s) seja falso", e vice-versa.

A notação apresentada permite traduzir de forma direta as operações básicas da álgebra relacional, conforme se pode ver abaixo:

1. União: $R \bigcup S$
   $\{t|\ t \in R\ or\ t \in S\}$
2. Diferença: $R \quad S$
   $\{t|\ t \in R\ and\ not\ t \in S\}$
3. Interseção: $R \bigcap S$
   $\{t \mid t \in R\ and\ t \in S\}$
4. Produto Cartesiano: $R \times S$
   $\{t, s \mid t \in R\ and\ s \in S\}$
5. Projeção de R nas colunas c e d: $\Pi_{c,d}(R)$
   $\{t.c, t.d \mid t \in R\}$
6. Seleção: seja F uma fórmula válida sobre colunas de R e a seleção $\sigma_F(R)$
   $\{t \mid t \in R\ and\ F\}$
   Exemplo: $\{t \mid t \in R\ and\ (t.nome = 'Paulo'\ or\ t.nome = 'Jose')\}$
7. Junção θ de R com S sobre as colunas a de R e b de S: $\underset{a\ \theta\ b}{R \mid \times \mid S}$
   $\{t, s \mid t \in R\ and\ s \in S\ and\ t.a\ \theta\ s.b\}$

Observações sobre a notação:

- Fórmulas do CRT podem ser arbitrariamente aninhadas com operadores $\exists$ *e* $\forall$ e e uma definição do "escopo" das varáveis tuplas nelas usadas se faz necessária. Uma analogia com linguagens de programação com estrutura de blocos é particularmente útil para este fim: $\forall s \in S(\ldots)$ e $\exists\ s \in S(\ldots)$ podem ser vistas como declarações de uma variável local s cujo escopo é o da fórmula contida no par de parênteses que se segue $(\ldots)$. Nesse caso se diz que s está "ligada" (em inglês, "bound") ao operador $\exists$ ou ao operador $\forall$. Já a variável t em $\{t \mid F(t)\}$ é dita "livre" e pode ser vista como uma variável global a F(t). As únicas variáveis livres ou globais de uma expressão do CRT são as que aparecem "declaradas" à esquerda do símbolo |.

- a notação $\forall s \in S(F(s))$ é uma simplificação não cosmética de:
  $(\forall s)(s \notin S\ or\ s \in S\ and\ (F(s))$
  garantindo que a expressão que usa $\forall$ seja "segura".[1] Nos parece que esta notação simplifica consideravelmente as consultas envolvendo o quantificador universal $\forall$ como veremos nos exemplos a seguir.

- a notação $\exists\ s \in S(F(s))$ é uma simplificação cosmética da notação $(\exists\ s)(s \in S\ and\ F(s))$

4.2.1 Exercícios resolvidos sobre CRT

Vamos apresentar a seguir expressões do CRT para resolver as consultas sobre Funcionários (*F(numf, nomef)*) e seus Dependentes (*D(numf, nomed, par)*), vistas na introdução à Álgebra Relacional.

1. Quais os nomes e parentescos de todos os dependentes?

   $\{t.nomed, t.par \mid t \in D\}$

2. Quais os números dos funcionários que possuem filhas?

   $\{t.numf \mid t \in D\ and\ t.par = 'filha'\}$

3. Dê os nomes dos funcionários que possuem algum dependente.

   $\{t.nomef \mid t \in F\ and\ \exists s \in D(t.numf = s.numf)\}$

   Este é um exemplo típico de junção natural expressa no CRT. A solução fica fácil de entender se imaginarmos t como uma variável global que "varre" todas as tuplas de F e para cada valor de t, num laço interno, a variável s "varre" as tuplas de D. Esta interpretação "algoritmica" é condizente com o escopo das variáveis tupla t e s, vistos na apresentação teórica sobre o CRT.

[1] Uma expressão do CRT se diz "segura" quando ela não gera relações com infinitas tuplas, o que não faria sentido [Ullman88].

4. Quais funcionários não possuem dependentes?

   $\{t.numf \mid t \in F \; and \; not \; \exists s \in D(t.numf = s.numf)\}$

5. Dê o nome de cada funcionário que possui uma dependente chamada ´Alice´.

   $\{t.nomef \mid t \in F \; and \; \exists s \in D(t.numf = s.numf \; and \; s.nomed = 'Alice')\}$

6. Quais funcionários possuem mais de um dependente?

   $\{t.numf \mid t \in D \; and \; \exists s \in D(t.numf = s.numf \; and \; t.nomed \neq s.nomed)\}$

7. Quais funcionários possuem exatamente um dependente?

   $\{t.numf \mid t \in D \; and \; not \; \exists s \in D(t.numf = s.numf \; and \; t.nomed \neq s.nomed)\}$

8. Quais funcionários não têm Alice como dependente, isto é, nenhuma dependente chamada Alice?

   $\{t.numf \mid t \in D \; and \; not \; \exists s \in D(t.numf = s.numf \; and \; s.nomed = 'Alice')\}$

9. Para cada funcionário que tem uma dependente chamada Alice, dê o número do funcionário e o nome dos outros dependentes, se houverem.

   $\{t.numf, t.nomed \mid t \in D \; and \; t.nomed \neq 'Alice' \; and$
   $\exists s \in D(t.numf = s.numf \; and \; s.nomed = 'Alice')\}$

10. Dê os nomes dos funcionários que possuem exatamente um dependente.

    $\{t.nomef \mid t \in F \; and \; \exists u \in D(t.numf = u.numf \; and \; not \; \exists s \in D(u.numf = s.numf \; and \; u.nomed \neq s.nomed)\}$

4.3 Exercícios

1. Suponha que as relações R e S possuem n e m tuplas, respectivamente. Dependendo das instâncias de R e S dê o mínimo e o máximo de tuplas que cada uma das expressões abaixo pode ter. Justifique!

   (a) $R \cup S$

   (b) $R \mid \times \mid S$

   (c) $\sigma_C(R) \times S$ para alguma condição C

   (d) $\Pi_l(R) \quad S$, para alguma lista de atributos l