Volume 1

5 SEMINÁRIO INTEGRADO SOFTWARE E HARDWARE NACIONAIS

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Núcleo de Computação Eletrônica

CAPRE/FINEP

1978

INTERFACE RELACIONAL DO SISTEMA HYADES

Conferencistas e autores:

C. Saraiva dos Santos
A. L. Furtado

Departamento de Informática da
Pontifícia Universidade Católica do R.J.

RESUMO:

É descrita a implementação de um interface relacional , que permite a manipulação de bancos de dados através da álgebra de relações quocientes.

A implementação é feita sobre um suporte básico, cujas características são indicadas.

O suporte básico e o interface são componentes do sistema HYADES de gerencia de bancos de dados, sendo desenvolvido na PUC/RJ.

1. INTRODUÇÃO:

O presente trabalho descreve o interface relacional do sistema HYADES de gerencia de bancos de dados, em elaboração na Pontifícia Universidade Católica do Rio de Janeiro.

O sistema Hyades consiste em um suporte básico [1,2] , implementado como primeira fase do projeto, sobre o qual diversos interfaces podem ser desenvolvidos. O primeiro destes é o interface relacional.

A seção 2 dá uma breve descrição do suporte básico, enquanto a seção 3 introduz o conceito de interface relacional. A seção 4 explica através de exemplos as operações que fazem parte do interface e esquematiza a solução usada para sua implementação. A seção 5 contém as especificações dos algoritmos de cada operação. A seção 6 apresenta aspectos a serem pesquisados futuramente.

São assumidos conhecimentos básicos de bancos de dados, em particular dos conceitos de arquivos, registros, inversões, elos, e da terminologia do modelo relacional (relações dominos, tuplas) [3].

2. SUPORTE BÁSICO

O suporte básico consiste de tres estruturas de dados, que são:

a) os arquivos de dados, nos quais são armazenadas efetivamente as informações pertencentes ao banco de dados;

b) os arquivos de inversões, que permitem o acesso eficiente aos arquivos de dados para localizar os grupos de registros (classes) que possuam um valor especificado para um dominio (ou um conjunto de valores especificado para um conjunto de dominios);

c) os arquivos de elos, que visam ao acesso eficiente a registros de arquivos de dados logicamente "ligados" a registros de outros arquivos de dados;

e consiste, ainda, das operações definidas sobre essas estruturas.

As operações atuam a nivel de arquivo (criação, cancelamento, copia, etc.), ou a nivel de registros (inserção, remoção, modificação, seleção, etc. de registros de arquivos).

O suporte básico contem, além dessas estruturas e operações fundamentais, uma estrutura que guarda informações sobre a organização do banco de dados (um diretório de dados inicial), e diversas operações auxiliares, incluindo facilidades para recuperação do banco de dados.

Em paralelo ao suporte básico foi desenvolvido um sistema de medição de eficiência [2]; este sistema deverá indicar aspectos em que o suporte básico poderá ser aperfeiçoado, além de permitir aos usuários melhorar a eficiência para suas aplicações especificas.

A programação foi efetuada em PL/I, utilizando o méto-

do de acesso indexado sequencial; conforme a instalação, poderá ser usada a versão VSAM ou ISAM.

Para contornar a exigência imposta pelo sistema operacional de que todos os arquivos devem ser inicialmente declarados, lançou-se mão da seguinte estrategia: é declarado um número fixo de arquivos, os quais são subdivididos em segmentos, conforme as necessidades verificadas em tempo de execução; os segmentos é que são vistos pelos usuários como arquivos (de dados, elos, inversões) e podem ser criados ou cancelados através das operações já mencionadas.

3. O INTERFACE

Embora o suporte básico possa em principio ser usado diretamente em programas de aplicação, preferimos considerá -lo como um meio para a implementação de interfaces de nível mais alto.

Diversos interfaces estão sendo considerados. O interface que discutiremos neste artigo enquadra-se no modelo relacional, oferecendo uma linguagem baseada na álgebra de relações quocientes, proposta em [4].

A característica mais importante da algebra de relações quocientes é o tratamento da quantificação através do particionamento das relações, o que permite efetuar comparações entre conjuntos de tuplas.

A implementação utiliza arquivos de inversões para denotar o particionamento das relações, as quais consistem em arquivos de dados. Como seria de esperar, cada tupla corresponde a um registro de um arquivo de dados.

Os arquivos de elos não são utilizados. O conceito de elos foi incorporado ao suporte para a eventual implementação de interfaces hierarquicos ou de redes.

4. IMPLEMENTAÇÃO DOS OPERADORES

Cada operador definido na Álgebra de Relações Quocientes (ARQ) é implementado como uma procedure na linguagem PL/I . Todos os operadores dão como resultado uma relação quociente , que é identificada por um "Character String" de comprimento igual ou menor que 7.

Uma relação quociente R é representada por um arquivo do tipo inversão, no suporte básico, que recebe o mesmo nome ("Character String") dado à relação. As tuplas propriamente ditas são armazenadas em um arquivo de dados, também do suporte básico, aqui denotado for D_R.

Uma relação quociente P, com o conjunto de domínios {N T B L}, particionada pelo subconjunto p_p={ N T } dos domínios, é ilustrada na figura 1.

Por simplicidade, nas ilustrações seguintes, uma relação quociente é representada apenas por suas tuplas, identificados os blocos da partição como na figura 2.

O conjunto de operadores definido na ARQ é composto dos seguintes:

a) Partição
b) Departição
c) Projeção
d) Restrição
e) União
f) Produto Cartesiano

Estes operadores são descritos a seguir. Na descrição, são comentadas algumas características dos algoritmos que os implementam, os quais são apresentados, na íntegra, no item 5 deste artigo.

Para ilustração das operações, são usadas as relações

P, S e T mostradas na figura 2. O símbolo R é usado em todos os exemplos, para denotar a relação resultante de uma operação.

4.1 - Partição

A operação de partição, simbolicamente representada por R ← Q/{A}, onde Q é uma relação quociente e A um subconjunto dos domínios de Q, gera a relação R a partir de Q, por particionamento pelo conjunto A de domínios.

Consideremos, como exemplo, a operação R ← P/{L}, a qual é consistente, por ser P (Fig.2) uma relação quociente e L um dos domínios de P.

A relação resultante R (fig.3) possuirá exatamente o mesmo conjunto de tuplas que P e será particionada pelo conjunto p_R de domínios, resultante da união de {L} com o conjunto p_P de domínios, pelo qual P está particionada, ou seja:

$$p_R = \{L\} \cup p_P$$
$$p_R = \{L\} \cup \{N\ T\}$$
$$p_R = \{N\ T\ L\}$$

Do ponto de vista operacional, é feita uma cópia D_R do arquivo D_P das tuplas de P, e criada a inversão R sobre D_R, pelo conjunto p_R domínios.

Uma alternativa de implementação, seria a criação da inversão R diretamente sobre o arquivo D_P, protelando a operação de cópia até o momento em que R(ou P) viesse a sofrer uma inserção, remoção ou alteração de tupla.

Com isto, haveria um ganho de eficiência caso R e P não fossem alterados. No entanto, esta alternativa exigiria modificações nos primitivos do suporte básico, fato este que determinou a adoção da cópia em todos os casos.

4.2 - Departição

Esta operação, denotada por R ← Q * { A }, é semelhante à partição, deferindo apenas no modo de obtenção do conjunto P_R de domínios.

Consideremos, como exemplo, a operação R ← P* {N}. O conjunto P_R é obtido pela diferença

$$P_R = p_P - \{N\}$$
$$P_R = \{N\ T\} - \{N\}$$
$$P_R = \{T\}$$

A relação R é mostrada na Figura 4.

4.3 - Projeção

A operação de projeção, denotada por R← Q[{A}] , onde A é um subconjunto dos domínios de Q, gera a relação R a pertir de Q, pela eliminação dos domínios de Q não constantes de A. Ela é definida somente no caso em que $P_Q \subseteq A$, sendo R particionada por $P_R = P_Q$.

O problema maior na implementação desta operação é a remoção das duplicatas resultantes da eliminação de alguns domínios da relação Q. Ele é resolvido pela efetivação dos dois passos ilustrados na figura 5, na qual é considera a operação R = P [{N T L}].

No primeiro passo, é criada a inversão R^{-} sobre D_p, pelos domínios A. No segundo, cada classe de R^{-} gera uma tupla de D_R-pela remoção do domínio B, sendo após criada a inversão R sobre D_r, pelos domínios $p_R = p_P$.

Com isto, o problema da remoção das duplicatas é trivialmente resolvido, por serem todos os conjuntos de duplicatas agrupados como classes da inversão R^{-}, bastando, pois, tomar um representante de cada classe.

É evidente, que está operação é razoavelmente dispendiosa, por exigir a criação da inversão R' como um passo intermediário. No entanto, isto é uma decorrência da complexidade do próprio problema de remoção das duplicatas.

4.4 - Restrição

A restrição, denotada por R ← Q [<condição>], gera a relação R, a partir de Q, pela eliminação dos blocos de Q que não satisfazem à <condição>. É importante salientar que a condição de restrição se refere a blocos de tuplas e não a tuplas individuais.

<Condição> é uma expressão com a forma $A_1\ \theta\ A_2$, onde A_1 e A_2 são subconjuntos dos domínios de Q, sendo que A_2 pode, ainda, ser um conjunto de constantes, e θ é um operador de comparação. A_1 e A_2 devem possuir a mesma cardinalidade e os seus domínios correspondentes devem ser do mesmo tipo. Detalhes sobre os operadores θ permitidos são encontrados em [4]

Tomemos, como exemplo, a operação R ← S [V > C] . Neste caso, estamos selecionando os blocos de S tais que o valor de V em alguma tupla seja maior do que o valor de C em alguma tupla, não necessariamente a mesma.

A relação R, resultante, é mostrada na figura 6.

Esta comparação (V > C) é implementada pela determinação do valor máximo do domínio V e do mínimo de C, em cada bloco. Aqueles blocos nos quais $max_V > min_C$, satisfazem à condição de restrição e irão compor a relação R, a qual é participada pelo conjunto de domínios $p_R = p_S$.

Os operadores θ permitidos na restrição, dividem-se em dois grupos:

a) Comparação tupla a tupla
b) Comparação de conjuntos

No grupo a, cada tupla é considerada isoladamente , sendo um bloco aceito se a <condição> for satisfeita por todas as suas tuplas, em alguns operadores, ou por alguma tupla, em outros. A verificação da condição é efetuada, em uma única varredura das tuplas de cada bloco, sendo, na maioria dos casos , um bloco aceito ou rejeitado sem que haja a necessidade de examinar todas as tuplas.

No grupo b, a maioria das comparações é verificada com base nos valores máximo e mínimo de cada domínio, os quais são determinados por uma varredura das tuplas de cada bloco. Em alguns casos, não é necessária a determinação dos valores máximo e mínimo, como no exemplo, onde o exame da primeira tupla do segundo bloco (v=30 > 20=c) já decide a aceitação deste bloco.

Os casos nos quais é exigida a confrontação de dois conjuntos (θ = c, $\subseteq$, etc.), são os que acarretam os maiores problemas de eficiência. A estratégia usada consiste em projetar cada bloco sobre os domínios A_1 e sobre os domínios A_2 formando dois conjuntos que são a seguir ordenados e comparados. Como estes conjuntos podem assumir grande cardinalidade, são armazenados em arquivos temporários, o que, evidentemente, torna o processamento mais demorado.

4.5 - União

A operação de união, denotada por $R \leftarrow Q \cup Z$, onde Q e Z são relações quocientes, gera a relação R contendo as tuplas de Q e as de Z, sem repetições.

É exigido que Q e Z sejam compatíveis com respeito à união, isto é, possuam o mesmo número de domínios, sendo de mesmo tipo os domínios correspondentes (de mesma ordem) de Q e Z. Além disto, os domínios que particionam Q devem corresponder aos que particionam Z.

R é particionada da mesma forma que Q e Z.

Tomando-se as relações S e T da figura 2, a operação $R \leftarrow S \cup T$ resultaria na relação mostrada na figura 7. Note-se que a tupla <c, 30, 20, 1, b> aparece nas relações S e T.

A união é implementada gerando-se o arquivo de dados D_R contendo o arquivo D_S das tuplas de S concatenado com o arquivo D_T das tuplas de T. A seguir, D_R é classificado por todos os domínios e removidas as duplicatas, resultando o arquivo das tuplas de R.

Após, é criada a inversão R de D_R pelos domínios $p_R = p_S = p_T$, sendo R a relação resultante da união.

4.6 - Produto Cartesiano

O produto cartesiano é a operação mais dispendiosa dentre aquelas implementadas, devendo ser evitada a sua utilização, sempre que possível. Na verdade, esta operação raramente se torna necessária para a obtenção de informações de um banco de dados, principalmente se for disponível a operação de junção ("JOIN"), a, qual não integra o elenco original da Álgebra de Relações Quocientes mas que deverá ser implementada, no futuro, como uma extensão.

A operação produto cartesiano, denotada por $R \leftarrow S \oplus T$, obtem a relação R pela concatenação de cada tupla de S com todas as de T. A relação resultante R é particionada pelo conjunto p_R que resulta da união disjunta de p_S com p_T.

O resultado da operação $R \leftarrow S \oplus T$ é mostrado, como exemplo, na figura 8. Note-se que os domínios de T foram renomeados em R para ser cumprido o requisito de não haver repetição de nomes de domínios em uma mesma relação.

CHAVE		TUPLAS
a	1	t_2 t_5 t_8
b	1	t_1
c	2	t_3 t_6
d	2	t_7
e	2	t_4

Inversão P p_P = {NT}

Fig 1: Representação de uma relação quociente P, em termos do suporte básico

	N	T	B	L
t_1	b	1	7	x
t_2	a	1	3	x
t_3	c	2	3	x
t_4	e	2	3	x
t_5	a	1	12	j
t_6	c	2	12	j
t_7	d	2	3	x
t_8	a	1	7	x

Tuplas de P = D_p

N	T	B	L
a	1	3	x
a	1	12	j
a	1	7	x
b	1	7	x
c	2	3	x
c	2	12	j
d	2	3	x
e	2	3	x

Relação P

p_P = {NT }

N	C	V	T	M
a	10	20	1	b
b	20	30	1	e
b	20	15	2	d
c	30	20	1	b

Relação S

p_S = {NC}

N	C	V	T	M
c	30	20	1	b
e	40	20	1	b

Relação T

p_T = {NC}

Figura 2: Relações quocientes P, S e T usadas nos exemplos, representadas na forma simplificada

N	T	B	L
a	1	3	x
a	1	7	x
a	1	12	j
b	1	7	x
c	2	3	x
c	2	12	j
d	2	3	x
e	2	3	x

R = P / {L}

Figura 3: Operação de Partição

N	T	B	L
a	1	3	x
a	1	7	x
a	1	12	y
b	1	7	x
c	2	3	x
c	2	12	j
d	2	3	x
e	2	3	x

R = P * { N }

Figura 4: Operação de Departição

N	T	B	L
a	1	3	x
a	1	7	x
a	1	12	j
b	1	7	x
c	2	3	x
c	2	12	j
d	2	3	x
e	2	3	x

R´= P/{N T L}

N	T	L
a	1	x
a	1	j
b	1	x
c	2	x
c	2	j
d	2	x
e	2	x

R = P [N T L]

p_R= {N T}

Figura 5: Operação de Projeção

N	C	V	T	M
a	10	20	1	b
b	20	30	1	e
b	20	15	2	d

R = S [v > c]

Figura 6: Operação de Restrição

N	C	V	T	M
a	10	20	1	b
b	20	30	1	e
b	20	15	2	d
c	30	20	1	b
e	40	20	1	b

R = S U T

Figura 7: Operação de União

N	C	V	T	M	N´	C´	V´	T´	M´
a	10	20	1	b	c	30	20	1	b
a	10	20	1	b	e	40	20	1	b
b	20	30	1	e	c	30	20	1	b
b	20	15	2	d	c	30	20	1	b
b	20	30	1	e	e	40	20	1	b
b	20	15	2	d	e	40	20	1	b
c	30	20	1	b	c	30	20	1	b
c	30	20	1	b	e	40	20	1	b

R = S ⊕ T

Figura 8: Produto Cartesiano

```
PARTIC (R1,NEWDOMS,R2PTDOMS):
      /* Particiona a relação quociente R2 pelos dominios   */
      /* PTDOMS e atribui o resultado a R1, mudando         */
      /* os nomes dos dominios para NEWDOMS                 */
      /* Verifica se R2 resultou de um erro em uma          */
      /* operação anterior                                  */
      If   R2 = '***xxx'
      Then Return (R2);
      /* Verifica se R2 é uma relação quociente             */
      If  Tipo (R2) ≠ RQ
      Then Return ('***PT01');
      /* Obtem os nomes dos dominios da relação R2          */
      DOMSR2 = Dominios (R2);
      /* Verifica se os dominios especificados para         */
      /* o particionamento pertencem  a  R2                 */
      If PTDOMS  ⊈  DOMSR2
      Then Return ('***PTO2');
      /* Obtem os índices dos dominios pelos quais          */
      /* R2 já está particionada                            */
      DOMSPR2 = Dominios _Partição (R2);
      /* Obtem os índices dos dominios de PTDOMS em         */
      /* relação aos dominios de R2 e faz a união           */
      /* com DOMSPR2, Resultando os dominios da partidão de R1*/
      DOMSPR1 = Indices (PTDOMS,DOMSR2) ∪ DOMSPR2;
      /* Gera arquivo das tuplas de R1 pela cópia           */
      /* das tuplas de R2, mudando os nomes dos             */
      /* dominios para NEWDOMS                              */
      BASER1 = Copia (Base (R2), NEWDOMS);
      /* Obtem o resultado da operação e atribui a R1       */
      R1 = Inversão (BASER1,DOMSPR1);
      Return (R1);
END   PARTIC.
```

```
DEPART  ( R1,NEWDOMS, R2, DPDOMS):

    /* Departiciona a relação quociente R2 pelos dominios     */
    /* PTDOMS, e atribui o resultado à R1, mudando            */
    /* os nomes dos domínios para NEWDOMS                     */
    /* verifica se R2 resulta de um erro em                   */
    /* uma operação anterior                                  */
    If  R2= '***xxx'
    Then Return (R2);
    /* Verifica se R2 é uma relação quociente                 */
    If  Tipo (R2) ≠  RQ
    Then Return ('***DP01');
    /* Obtem os nomes dos dominios da relação R2              */
    DOMSR2 = Dominios (R2);
    /* Verifica se os dominios espeficiados para o            */
    /* departicionamento pertencem à R2                       */
    If DPDOMS  ⊈  DOMSR2
    Then Return ('***DP02');
    /* Obtem os indices (Nºs de ordem) dos dominios           */
    /* pelos quais R2 está particionada                       */
    DOMSPR2 = Dominios _ Partição (R2);
    /* Obtem os dominios da partição de R1 pela di-           */
    /* ferença entre DOMSPR2 e os índices dos do-             */
    /* minios de DPDOMS em relação A DOMSR2                   */
    DOMSPR1 = DOMSPR2 - Indices (DPDOMS , DOMSR2);
    /* Gera arquivo das tuplas de R1 pela cópia               */
    /* das tuplas de R2, mudando os nomes dos                 */
    /* dominios para NEWDOMS                                  */
    BASER1 = Cópia (Base(R2),NEWDOMS);
    /* Obtem o resultado da operação e atribui a R1           */
    R1 = Inversão(BASER1, DOMSPR1);
    RETURN (R1);
END DEPART.
```

```
PROJEC (R1, NEWDOMS, R2, PJDOMS):
    /* Projeta a relação R2 sobre os dominios PJDOMS          */
    /* e atribui o resultado à R1, mudando os nomes           */
    /* dos dominios para NEWDOMS                              */
    /* verifica se R2 resulta de um erro em uma               */
    /* operação anterior                                      */
    If R2 = '***xxxx'
    Then Return (R2);
    /* Verifica se R2 é uma relação quociente                 */
    If Tipo (R2) ≠ RQ
    Then Return ('***PJ01');
    /* Obtem os nomes dos dominios de R2 e os                 */
    /* indices (nºs de ordem) daqueles pelos quais            */
    /* R2 está particionada                                   */
    DOMSR2 = Dominios (R2);
    DOMSPR2= Dominios_Partição  (R2);
    /* Verifica se os domínios da projeção pertencem          */
    /* à R2 e se incluem todos os da partição de R2           */
    If  JPDOMS   ⊄ DOMSR2  or
        DOMSPR2  ⊄ Indices (PJDOMS,DOMSR2)
    Then Return ('***PJ02');
    /*  Obtem índices dos dominios da projeção                */
    DOMSPJ = Índices(PJDOMS,DOMSR2);
    /* Cria inversão de R2 pelos domínios da projeção         */
    T1 = Inversão (Base(R2), DOMSPJ);
    /* Cria arquivo das tuplas de R1, no qual cada            */
    /* Tupla é uma das classes da inversão T1. Os             */
    /* nomes dos domínios são mudados para NEWDOMS            */
    BASEAR1 = Cria _Tuplas (T1,NEWDOMS);
    /* Obtem índices dos domínios da partição de R1           */
    /* ajustando os índices de DOMSPR2, considerando          */
    /* que alguns dominios de R2 não estarão pre-             */
    /* sentes em R1                                           */
    DOMSPR1 = Ajusta(DOMSPR2, DOMSPJ);
```

```
    /* Obtem o resultado da operação e atribui a R1          */
    R1 = Inversão (BASER1,DOMSPR1);
    Return(R1)
END PROJEC.
```

```
RESTRIC  (R1,NEWDOMS, R2, DOMSA, TETA, DOMSB):
    /* R1 é obtida pela restrição de R2 aos blocos B que sa  */
    /* tisfazem à condição B(DOMSA) TETA B(DOMSB), onde DOMSA e */
    /* e DOMSB são sequências de dominios Teta-comparáveis    */
    /* de R2. DOMSB pode, também , ser um conjunto de         */
    /* constantes, ao invés de dominios                       */
    /* Os dominios de R1 recebem os nomes constantes de NEWDOMS */
    /* Verifica se R2 resulta de um erro em uma ope-          */
    /* ração anterior ou não é uma relação quociente          */
    If R2= '***xxxx'
    Then Return (R2);
    If Tipo (R2) ≠ RQ
    Then Returm ('***RT01');
    /* Verifica se DOMSA e DOMSB são teta-comparaveis em R2   */
    If   Teta_comparaveis (DOMSA,DOMSB,R2)
    Then Return('***RT02');
    /* Se o segundo operando for constante, cria ar-          */
    /* quivo TB com as tuplas nele contidas e coloca em MINB  */
    /* e MAXB a minima e a maxima, respectivamente            */
    If DOMSB = '*......' Then
    Do;
        Constantes = True;
        Gera_Arq_Constantes(TB,DOMSB,MINB,MAXB);
    End;
    /* Examina cada bloco da Partição de R2: Se o             */
    /* bloco satisfaz à condição de restrição, é inserido     */
    /* no arquivo das tuplas da relação resultante            */
    Do each Bloco(B2) of R2;
        /* Examina o bloco de acordo com o grupo              */
        /* ao qual pertence operador Teta                     */
        If Grupo (TETA) = 1 then
        /* Grupo 1: =, ≃, ⊂, ⊆, ⊃, ⊇, ≠, ⋣, ⊄, ⊈, ⊅, ⊉       */
        Do;
           /* Projeta o bloco sobre os dominios DOMSA,        */
```

```
    /* obtendo TA e sobre DOMSB, obtendo TB,  se        */
    /* DOMSB não é constante                            */
    Gera_Arq_Proj (TA,B2DOMSA);
    If  CONSTANTES Then
    Gera_Arq_Proj (TB,B2,DOMSB);
    /* Verifica se o bloco satisfaz à restrição         */
    ACEITO = TA . TETA.TB;
END; Else

IF  Grupo(TETA) = 2 Then
/*  Grupo 2 : >,.>,>.,.>., <,.<,<., .<.,                */
/*            >=, .>=, >=., .>=., <=,.<=, <=., .<=.     */
Do;
    /* Obtem valores Máximo e Mínimo das projeções      */
    /* de B2 sobre DOMSA e DOMSB(se DOMSB não é         */
    /* DOMSB não é constante). A obtenção dos           */
    /* quatro parâmetros é feita em uma única           */
    /* varredura de B2                                  */
     MINA = Mínimo (B2,DOMSA);
     MAXA = Máximo (B2,DOMSA);
     If  Constantes Then
     Do;
         MINB = Mínimo (B2,DOMSB);
         MAXB = Máximo (B2,DOMSB);
     End;
    /* Verifica se o bloco satisfaz à restrição         */
    ACEITO = (MINA,MAXA) .TETA.(MINB,MAXB);
End ; Else
/* Operadores "Tupla a Tupla": >,<, =,> =,< =          */
If Grupo(TETA) = 3A Then
/* Grupo 3A: a condição é exigida para alguma tupla de */
/* B2                                                  */
Do;
    ACEITO = False;
    Do each Tupla(T) of B2 While (¬ACEITO);
       ACEITO = T [DOMSA] . TETA. T [DOMSB];
    End;
```

```
      End; Else
      /* Grupo 3B: a condição é exigida para todos as tuplas */
      Do;
         ACEITO = True;
         Do each Tupla(T) of B2 While (ACEITO);
            ACEITO = T [DOMSA ].TETA.T [DOMSB ] ;
         End;
      End;
      /* Se o bloco foi aceito, insere no arquivo            */
      /* das tuplas de B1                                    */
      If ACEITO
      Then Insere_Bloco(B2,BASER1);
   End;
   /* Obtem índices dos domínios da partição de R1=partição  */
   /* de R2                                                  */
   DOMSPR1= Dominios_Partição(R2);
   /* Obtem relação resultante e atribui a R1                */
   R1 = Inversão (BASER1,DOMSPR1);
   Return(R1);
End  RESTRIC.
```

```
PRODUTO (R1,NEWDOMS,R2,R3):
    /* Obtem o produto cartesiano das relações R2 e R3 e      */
    /* atribui o resultado a R1, mudando os nomes dos         */
    /* domínios para NEWDOMS                                  */
    /* verifica se R2 (ou R3) resultou de um erro em          */
    /* uma operação anterior                                  */
    If R2 = '*** xxxx'
    Then Return (R2);
    If R3 = '*** xxxx'
    Then Return (R3);
    /* Verifica se R2 e R3 são relações quocientes            */
    If tipo (R2) ≠ RQ or
       tipo (R3) ≠ RQ
    Then Return ('***PC01');
    /* Cria arquivo vazio que irá conter as tuplas de R1      */
    DIMR1 = Dimensão (R2)* Dimensão (R3);
    DOMSR1= Dominios (R2)// Dominios (R3);
    BASER1 = Cria-Dummy (DIMR1,DOMSR1)
    /* Gera as tuplas do produto cratesiano pela conca-       */
    /*tenação de cada tupla de R2 com cada uma                */
    /* de R3, bloco a bloco                                   */
    Do each Bloco(B2) of R2;
       Do each Bloco(B3) of R3;
          Do each Tupla (T2) of B2;
             Do each Tupla(T3) of B3;
                T1 = T2 || T3;
                Insere  Tupla(BASER1,T1);
          End;
       End;
    End;
End ;
/* obtem índices dos domínios da partição de R1              */
/* pela concatenação dos de R2 com os de R3, ajus-           */
/* tados pela soma do nº de domínios da relação R2           */
```

```
    DOMSPR2= Domínios_Partição(R2);
    DOMSPR3= Dominios_Partição(R3) + # (Dominios(R2));
    DOMSPR1= DOMSPR2 || DOMSPR3;
    /* Obtem relação resultante e atribui a R1
    R1 = Inversão (BASER1,DOMSPR1);
    Return (R1);
End PRODUTO.
```

```
UNIÃO (R1,NEWDOMS, R2,R3):
   /* Faz a união das relações R2 e R3 e atribui o re-        */
   /* sultado à R1, mudando os nomes dos dominios             */
   /* para NEWDOMS                                            */
   /* Verifica se R2 (ou R3) resultou de um erro              */
   /* em uma operação anterior                                */
   If R2 = '*** xxxx'
   Then Return (R2);
   If R3 = '*** xxxx'
   Then Return (R3);
   /* Verifica se R2 e R3 são relações quocientes e           */
   /* compativeis com respeito à união                        */
   If Tipo (R2) ≠ RQ  or
      Tipo (R3) ≠ RQ  or
      ¬ União-compativeis (R2,R3)
   Then Return ('***UNO1');
   /* Cria BASER1 = concatenação do arquivo das tuplas        */
   /* de R2 com o das tuplas de R3 e remove as                */
   /* Tuplas duplicadas, resultando o arquivo das             */
   /* Tuplas de R1                                            */
   BASER1 = Cat-Dados (Base(R2), BASE(R3), NEWDOMS);
   BASER1 = Remove_Duplos (BASER1);
   /* Obtem os índices dos dominios da partição               */
   /* de R2, que serão os mesmos da partição                  */
   /* de R1, e obtem a relação resultante R1                  */
   DOMSPR1 = Dominios Partição(R2);
   R1 = Inversão (BASER1,DOMSPR1)
   Return(R1)
End UNIÃO.
```

6. CONCLUSÕES:

Ao implementar a interface ficou evidenciado que o suporte basico tem de fato as características necessárias à implementação facil e eficiente de interfaces.

A pesquisa futura visará principalmente a:

a) Adicionar ao interface operações que, embora não sejam estritamente necessárias (podem ser definidas em termos das operações implementadas) são convenientes; as principais são: junção, interseção, diferença.

b) Aperfeiçoar os algoritmos quanto à eficiência. Várias situações particulares poderão ser aproveitadas com este fim; por exemplo, se p_S = A e desejamos R ← S [A = a] , onde a é um conjunto de constantes do tipo de A, podemos obter R selecionando a classe correspondnete a a da inversão associada a S.

AGRADECIMENTOS

Agradecemos o auxilio do Conselho Nacional de Desenvolvimento Científico e Tencológico para o desenvolvimento desta pesquisa.

BIBLIOGRAFIA

1. Vasques,R.P. - "Desenvolvimento, implementação, descrição lógica e documentação de um sistema de banco de dados" - tese de mestrado -PUC/RJ (1978)

2. Passos,Sonia,M.M. - "Desenvolvimento, implementação e avaliação da eficiência de um sistema de banco de dados" - tese de mestrado - PUC/RJ (1978).

3. Codd,E.F. - "Relational completeness of data base sublanguages" - in "Data Base Systems" - Rustin (ed.) - (1971).

4. Furtado,A.L. e Kerschberg,L. - "An algebra of quotient relations" - anais da conferencia SIGMO (1977).

CLESIO SARAIVA DOS SANTOS: Engenheiro Civil pela Universidade Federal do Rio Grande do Sul (1970), Mestre em Ciências em Informática (PUC/RJ,1972), Professor Assistente da UFRGS desde 1971, desempenhando atividades de ensino e pesquisa no CPD e no Curso de Pós-Graduação em Ciência da Computação. Atualmente integrando o Programa de Doutoramento em Informática da PUC RJ.

ANTONIO LUZ FURTADO: Economista pela Universidade do Estado do Rio de Janeiro (1964), Mestre em Ciências em Informática (PUC-RJ, 1969), Ph.D. em Ciência da Computação pela Universidade de Toronto (1974), professor associado do Departamento de Informática da PUC/RJ desde 1968.