UMA LINGUAGEM DE CONSULTA PARA O MODELO ER E SUA COMPLETUDE

ANSELMO MORAES NETO

DISSERTAÇÃO APRESENTADA

AO

INSTITUTO DE MATEMÁTICA E ESTATÍSTICA

DA

UNIVERSIDADE DE SÃO PAULO

PARA OBTENÇÃO DO GRAU DE MESTRE

EM

MATEMÁTICA APLICADA

ORIENTADOR: PROF. DR. VALDEMAR W. SETZER

- SÃO PAULO, JULHO DE 1982 -

A minha avó,

a meus pais,

à Nilce.

AGRADECIMENTOS

Ao Professor Dr. Valdemar W. Setzer, pela orientação, dedicação e incentivo;

à Nilce, pela ajuda, compreensão, carinho e apoio constante;

ao Professor Dr. Imre Simon, pela orientação durante os cursos de pós-graduação;

aos amigos do MAP e da GCPD, pelo estímulo;

aos amigos Décio Teixeira e Wagner Tunis Martins, pelas valiosas trocas de idéias;

à Luzia do Carmo Namiki e Regina Helena da Silva, pelo eficiente trabalho de datilografia.

São Paulo, julho de 1982.

A.M.N.

ÍNDICE

pág.

CAPÍTULO I - Introdução I.1

CAPÍTULO II - GORDAS - Graph-Oriented Data Selection.. II.1

1 - Introdução II.1

2 - Grafos Associados ao Modelo ER II.4

Figura II.A II.19

Figura II.B II.20

Figura II.C II.21

3 - Introdução às Consultas II.22

4 - Valores Tratados pela Linguagem GORDAS II.25

5 - Expressões-de-Caminho II.28

6 - Predicados II.38

7 - Prefixos Livres e Predicados de Restrição II.47

8 - Consultas II.52

9 - Consultas bem Formadas II.65

10 - Expressões-de-Caminho bem Especificadas II.69

11 - Valor de uma Consulta II.77

12 - Valor de uma Expressão-de-Caminho II.81

13 - Comentários II.91

CAPÍTULO III - A Completude ER de Linguagens de Consulta III.1

1 - Introdução III.1

2 - O Cálculo ER III.2

3 - A Completude ER de Linguagens de Consulta..... III.8

4 - Comentários III.10

pág.

CAPÍTULO IV - A Completude ER da Linguagem GORDAS IV.1

CAPÍTULO V - Extensões V.1

REFERÊNCIAS

CAPÍTULO I

Introdução

O modelo Entidade-Relacionamento (ER), introduzido por P.P. Chen em 1976, tornou-se rapidamente conhecido e despertou interesse tanto no meio acadêmico como no meio empresarial.

No meio acadêmico, inúmeros trabalhos foram e continuam sendo realizados com base nesse modelo, tendo sido já realizadas duas conferências internacionais sobre a abordagem ER na análise e projeto de sistemas (Los Angeles, 1979 e Washington, D.C., 1981). Mais que um modelo para bases de dados, ele tornou-se uma ferramenta na análise e projeto de sistemas de informação.

No meio empresarial, os conceitos do modelo (entidade, relacionamento, conjunto-entidade, conjunto-relacionamento, atributo, conjunto-de-valores, classes de relacionamento 1:n, m:n,..., etc.) tornaram-se familiares aos profissionais em processamento de dados, e os fabricantes de Sistemas Gerenciadores de Bases de Dados vêm gradativamente utilizando esses conceitos para descrever os seus produtos (TOTAL, IMS, DMSII, etc.).

Certamente uma das razões desse sucesso deve-se ao fato desse modelo basear-se em conceitos simples e naturais , que podem ser facilmente compreendidos mesmo por pessoas com pouco conhecimento de bases de dados.

Com a difusão do modelo ER surgiram linguagens de consulta a bases de dados para o mesmo (CLEAR, CABLE, Executable Language, GORDAS), o que motivou a introdução do conceito de "Completude ER" (P.P. Chen e P. Atzeni, 1981). A introdução desse conceito estabeleceu um padrão para avaliação de linguagens de consulta para o modelo ER, o qual baseia-se na linguagem de Cálculo de Predicados da lógica formal, e determina a "potência de consulta" desejável para essas linguagens.

O objetivo inicial deste trabalho era tentar definir uma linguagem de consulta para o modelo ER, com estrutura simples porém poderosa. Durante a pesquisa bibliográfica verificou-se que uma das linguagens existentes, a GORDAS (R.Elmasri e G. Wiederhold, 1981), aproximava-se bastante do que se desejava. No entanto a sua definição original mostrava-se bastante incompleta: a semântica era introduzida informalmente através de exemplos e a sintaxe, embora formalmente definida, apresentava inconsistências. Ainda durante a pesquisa bibliográfica deparou-se com o conceito de Completude ER, e verificou-se que o mesmo não havia sido ainda considerado para a referida linguagem.

Optou-se então por elaborar um estudo detalhado e formal da sintaxe e da semântica da linguagem GORDAS (Capítulo II), e complementá-lo com uma avaliação da "potência de consulta" dessa linguagem em relação ao padrão estabelecido pelo con

ceito de Completude ER (apresentado no Capítulo III). Essa avaliação resultou em uma demonstração formal de que a linguagem GORDAS tem a propriedade da Completude ER (Capítulo IV).

Devido à extensão e à profundidade do trabalho acima descrito não foi possível abordar certas questões que surgiram durante a sua realização. Essas questões estão apresentadas sucintamente como temas para possíveis extensões deste trabalho (Capítulo V).

CAPÍTULO II

GORDAS - Graph-Oriented Data Selection

1 - INTRODUÇÃO

A linguagem GORDAS foi inicialmente desenvolvida com base no modelo Entidade-Categoria-Relacionamento (ECR).

No modelo ECR as entidades são agrupadas em categorias, e os relacionamentos são definidos entre categorias. As entidades agrupadas em uma categoria podem pertencer a conjuntos-entidade diferentes, e uma entidade pode pertencer a várias categorias. Uma entidade pode possuir atributos específicos de cada uma das categorias às quais ela pertence, além dos atributos básicos definidos para o conjunto-entidade ao qual ela pertence. Pode-se dizer que o modelo ER é um caso particular do modelo ECR em que as categorias coincidem exatamente com os conjuntos-entidade.

Uma descrição completa dessa versão da linguagem para o modelo ECR (incluindo uma descrição do modelo) pode ser encontrada em /ELMA81/, onde a linguagem é apresentada como uma linguagem de alto nível, auto-contida (não requer uma linguagem

hospedeira) e "non-procedural" (as consultas não são especificadas através de algoritmos construídos pelo usuário), dirigida para acessar a base de dados através de terminais, e que foi desenvolvida tendo em vista um compromisso entre os seguintes objetivos:

(i) ser uma linguagem que permita ao usuário especificar consultas de forma natural;

(ii) ser uma linguagem formal e não ambígua, de forma que sua implementação seja simples.

Essa versão da linguagem está sendo implementada como parte de um projeto experimental chamado "Distributed Database Testbed System (DDTS)", no "Honeywell Corporate Computer Sciences Center", que é apresentado em /DEWE80/.

Posteriormente foi desenvolvida uma nova versão da linguagem baseada diretamente no modelo ER, descrita em /ELWI81/.

Optou-se aqui por tratar essa segunda versão da linguagem, já que o modelo ER é mais simples e mais conhecido, e a parte da linguagem correspondente às consultas sobre a base de dados (cuja análise é o objetivo deste trabalho) é praticamente idêntica em ambas as versões.

De fato, a linguagem que será descrita e analisada a seguir não corresponde totalmente à linguagem descrita em /ELWI81/, já que foi necessário introduzir algumas modificações

para evitar inconsistências. Essas modificações estão ressaltadas e justificadas ao longo do texto. Além disso, como as descrições de ambas as versões da linguagem são bastante incompletas no que se refere a semântica da linguagem (que é introduzida apenas através de exemplos), é possível que o significado atribuído a certas construções da linguagem também não corresponda exatamente ao que foi imaginado pelos seus autores, apesar de permitir a mesma interpretação para os exemplos apresentados.

Como a descrição que segue em geral difere bastante da descrição existente em /ELWI81/, que daqui em diante será referenciada como *a descrição dos autores*, procurou-se destacar os trechos que são traduções praticamente diretas de trechos do referido texto, através de setas com sentidos opostos colocadas junto a margem esquerda dos mesmos (que apontam para o interior do trecho ao qual se referem). Procurou-se também destacar através de comentários as principais diferenças entre as descrições.

Além disso, para simplificar eventuais comparações entre as descrições, na primeira vez que é citado no texto o nome de um elemento da linguagem também é citado o nome atribuído ao mesmo na descrição dos autores.

2 - GRAFOS ASSOCIADOS AO MODELO ER

A linguagem GORDAS baseia-se numa representação abstrata da base de dados através de grafos construídos a partir dos conceitos do modelo ER.

São considerados dois tipos de grafos:

(i) *grafo-esquema* ("schema graph"):

Representa a estrutura da base de dados, correspondendo aproximadamente ao diagrama ER de Chen. O grafo-esquema é utilizado como referência na formulação de consultas.

(ii) *grafo-base-de-dados* ("database graph"):

Representa a própria base de dados em uma posição específica, isto é, representa as entidades e relacionamentos (e os valores de seus atributos) da base de dados em um certo instante.

O conceito de grafo-base-de-dados é utilizado para descrever a semântica da linguagem.

Definição 2.1 (grafo-esquema)

↓ ⌜ Um *grafo-esquema* é um grafo orientado e rotulado. Todo nó nesse grafo representa um conjunto-entidade (um *CE-nó*) ou um conjunto-relacionamento (um *CR-nó*). Toda aresta nesse

grafo liga um CE-nó a um CR-nó, e é dirigida do CE-nó para o CR-nó. Uma aresta de um CE-nó X para um CR-nó Y especifica a participação do conjunto-entidade X no conjunto-relacionamento Y.

Todo CR-nó está ligado a pelo menos duas arestas

Definição 2.2 (rótulos em um grafo-esquema)

Tanto os nós quanto as arestas do grafo-esquema são rotulados.

Um *rótulo de nó* é da forma

(tipo-do-nó, nome-do-nó: atr_1 =$cval_1$,...,atr_n=$cval_n$).

Para um CE-nó o *tipo-do-nó* é CE, e n > 0;
para um CR-nó o *tipo-do-nó* é CR, e n $\geq$ 0.

O *nome-do-nó* corresponde ao nome do conjunto-entidade ou do conjunto-relacionamento representado pelo nó. Os *nomes dos atributos*, atr_i, $1 \leq i \leq n$, correspondem aos nomes dos atributos associados ao conjunto-entidade ou conjunto-relacionamento representado pelo nó. Os *nomes dos conjuntos-de-valores*, $cval_i$, $1 \leq i \leq n$, são nomes de conjuntos-de-valores pré-definidos.

Um *rótulo de aresta* é da forma
($nome_1$, $nome_2$: i_1, i_2), onde $nome_1$ é um identificador arbitrá

rio associado ao sentido da aresta (de um CE-nó para um CR-nó), e $nome_2$ é um identificador (também arbitrário) associado ao sentido oposto.

Os inteiros i_1 e i_2 são opcionais, e restringem o número de R-nós ligados diretamente a um E-nó no grafo-base-de dados correspondente (ver definições 2.4 a 2.6) a um valor entre i_1 e i_2 (inclusive), com $i_1 \geq 0$, $i_2 > 0$ e $i_2 \geq i_1$.

A *parte de nome* de um rótulo de aresta é ($nome_1$, $nome_2$), e os componentes da mesma ($nome_1$ e $nome_2$) são chamados *nomes de conexão*.

A *parte de restrição* do rótulo é (i_1,i_2). Se i_1 não estiver especificado no rótulo o valor subentendido é $i_1=0$. Se i_2 não estiver especificado o valor subentendido é o maior número inteiro que pode ser tratado pela particular implementação da linguagem. |

Daqui em diante o termo *parte de restrição de uma aresta* designa sempre o par (i_1,i_2) associado à aresta, independentemente desses valores estarem ou não explicitamente especificados no rótulo da mesma.

A figura II.A mostra um diagrama ER, e a figura II.B mostra o grafo-esquema correspondente.

Note-se que para não sobrecarregar o exemplo foram omitidos os nomes dos conjuntos-de-valores.

Comparando os dois diagramas, observa-se que a única diferença de conteúdo entre os mesmos é a presença dos nomes de conexão que não existem no diagrama ER.

Os nomes de conexão são utilizados na linguagem GORDAS para especificar acessos a entidades ou relacionamentos a partir de outras entidades ou relacionamentos, conforme será descrito posteriormente.

Como os nomes de conexão são identificadores associados às arestas, e uma aresta representa a participação de um conjunto-entidade em um conjunto-relacionamento, pode ser feita uma analogia entre os nomes de conexão e os papéis associados às entidades no modelo ER.

É interessante observar que embora o grafo-esquema da figura II.B seja conexo isso não é obrigatório para todo grafo-esquema, podendo haver inclusive um grafo-esquema composto somente por CE-nós, sem nenhuma aresta.

Definição 2.3 (restrições sobre os nomes no grafo-esquema)

As seguintes restrições devem ser obedecidas em um grafo-esquema:

a) Cada nome-de-nó deve ser único.

b) Cada nome de atributo deve ser único para cada nó.

c) Para um CE-nó X, se as partes de nome dos rótulos de quais

quer duas arestas ligadas diretamente a X são (A,B) e (C,D), então deve-se ter A≠C.

d) Para um CR-nó X ligado diretamente a exatamente duas arestas (que representa um relacionamento binário), se a parte de nome do rótulo de uma das arestas ligadas a X é (A,B), então a parte de nome do rótulo da outra aresta deve ser (B,A). Além disso, os nomes dos atributos de X (se houver) devem ser diferentes dos nomes dos atributos dos dois CE-nós ligados diretamente a X.

e) Para um CR-nó X, se as partes de nome dos rótulos dequaisquer duas arestas ligadas a X são (A,B) e (C,D), então deve-se ter B ≠ D.

Essas restrições são necessárias pra tornar as consultas não-ambíguas. As restrições (a) e (b) são intuitivas.

As restrições (c) e (e) dizem essencialmente que os nomes de conexão "que saem" de cada nó devem ser diferentes.

A restrição (d) deve-se à forma especial como são tratados os relacionamentos binários na linguagem GORDAS, e seu objetivo somente ficará claro após a apresentação da linguagem.

Na descrição dos autores a restrição (e) aplica-se apenas aos CR-nós ligados a mais de duas arestas (que representam relacionamentos de grau n > 2). Dessa forma a parte de nome de ambas as arestas ligadas a um CR-nó que representa um relacionamento binário poderia ser (A,A), pois a restrição (d) também estaria satisfeita. Isso entretanto provocaria ambigüidade

nas consultas, já que o nome de conexão A estaria associado a dois caminhos diferentes com origem no mesmo CR-nó.

Definição 2.4 (grafo-base-de-dados

↓ ⌜ Um *grafo-base-de-dados* é um grafo orientado e rotulado.

Cada nó de um grafo-base-de-dados representa uma entidade (nesse caso diz-se que ele é um *E-nó*) ou um relacionamento (nesse caso diz-se que ele é um *R-nó*).

Cada aresta de um grafo-base-de-dados liga um E-nó a

↑ um R-nó, e é dirigida do E-nó para o R-nó. ⌟

Definição 2.5 (rótulos em um grafo-base-de-dados)

⌜ O rótulo de um nó x em um grafo-base-de-dados é da forma (nome-do-conjunto: atr_1=val_1,...,atr_n=val_n).

Se x é um E-nó o nome-do-conjunto é o nome do conjunto-entidade ao qual pertence a entidade representada por x; se x é um R-nó o nome-do-conjunto é o nome do conjunto-relacionamento ao qual pertence o relacionamento representado por x.

Para i=1,...,n, atr_i é o nome de um atributo associado ao conjunto-entidade (ou conjunto-relacionamento) cujo nome é nome-do-conjunto, e val_i é o valor do atributo atr_i para a entidade (ou relacionamento) representada pelo nó x.

O rótulo de uma aresta d em um grafo-base-de-dados

é da forma ($nome_1$, $nome_2$), onde $nome_1$ é um identificador arbitrário associado ao sentido da aresta e $nome_2$ é um identificador também arbitrário associado ao sentido inverso. Esses identificadores são chamados de *nomes de conexão*.

Definição 2.6 (restrições em um grafo-base-de-dados)

Um grafo-base-de-dados específico GBD deve *corresponder* a um grafo-esquema específico GE, isto é:

1 - Para todo nó x de GBD deve existir um nó X em GE tal que:

1.1 - Se x é um E-nó, X é um CE-nó; se x é um R-nó, X é um CR-nó.

1.2 - O nome-do-conjunto especificado no rótulo de x é o nome-do-nó especificado no rótulo de X.

1.3 - Os nomes dos atributos $atr_1,\ldots,atr_n$ especificados nos rótulos de x e de X são respectivamente iguais.

1.4 - Os valores dos atributos $val_1,\ldots,val_n$ especificados no rótulo de x devem pertencer respectivamente aos conjuntos-de-valores $cval_1,\ldots cval_n$ especificados no rótulo de X.

Diz-se que o nó x *pertence* ao nó X.

2 - Para toda aresta d em GBD que liga um E-nó x (que pertence a um CE-nó X em GE) a um R-nó y (que pertence a um CE-nó Y em GE) deve existir uma aresta D em GE que liga o CE-nó X

ao CR-nó Y tal que a parte de nome de seu rótulo seja idêntica à parte de nome do rótulo da aresta d.

Diz-se que a aresta d *pertence* à aresta D.

3 - (restrição básica dos relacionamentos)

Seja x um R-nó de GBD que pertence a um CR-nó X em GE.

Se X está ligado diretamente a exatamente n arestas $D_1,\ldots,D_n$, então x deve estar ligado diretamente a exatamente n arestas $d_1,\ldots,d_n$, que pertencem respectivamente a $D_1,\ldots,D_n$.

4 - (restrição de cardinalidade)

Seja x um E-nó de GBD que pertence a um CE-nó X de GE.

Seja D uma aresta em GE que liga X a um R-nó Y.

Se a parte de restrição associada a aresta D é (i_1,i_2) então x deve estar ligado a n R-nós $y_1,\ldots,y_n$ que pertencem a Y através de n arestas $d_1,\ldots,d_n$ que pertencem a D, onde $i_1 \leq n \leq i_2$. ⊥

A figura II.C mostra uma parte de um grafo-base-de-dados que corresponde ao grafo-esquema da figura II.B. Os nós $N_{11},\ldots,N_{18}$ pertencem ao nó N1 do grafo-esquema; os nós $N_{21},\ldots,N_{229}$ pertencem ao nó N2 do grafo-esquema, e assim sucessivamente.

Note-se que para não sobrecarregar a figura não fo

ram representados os rótulos de todas as arestas e os nomes dos atributos foram omitidos.

Na definição 2.6, os itens 1 e 2 definem um mapeamento dos nós e das arestas de um grafo-base-de-dados sobre os nós e as arestas do grafo-esquema correspondente.

Deve-se notar que devido às restrições impostas sobre um grafo-esquema pela definição 2.3 esse mapeamento é único, isto é, cada nó do grafo-base-de-dados pertence a um único nó do grafo-esquema, o mesmo ocorrendo com as arestas.

O item 3 procura assegurar a correta representação do conceito de relacionamento e conjunto-relacionamento através dos grafos. Cabe notar porém que essa definição não impede a existência em um grafo-base-de-dados de vários R-nós $y_1, y_2, \ldots, y_n$ ligados exatamente aos mesmos E-nós $x_1, x_2, \ldots, x_m$, com $y_1, y_2, \ldots, y_n$ pertencendo a um mesmo CR-nó Y no grafo-esquema correspondente, e com as arestas $d_{1j}, d_{2j}, \ldots, d_{nj}$, que ligam os R-nós $y_1, y_2, \ldots, y_n$ ao E-nó x_j, pertencendo à uma mesma aresta D_j do grafo-esquema (que liga o CR-nó Y ao CE-nó X_j ao qual pertence o E-nó x_j), para $j = 1,2,\ldots,m$. Isso significa que podem existir vários relacionamentos no conjunto-relacionamento representado por Y relacionando as mesmas entidades, o que é incompatível com o conceito de relação matemática associado a um conjunto-relacionamento.

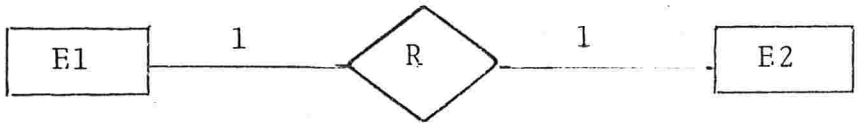
Como a descrição dos autores não ressalta essa possibilidade, acredita-se que a mesma não tenha sido introduzida propositalmente na linguagem, tratando-se apenas de uma impreci

são das definições. Entretanto, existem casos práticos em que essa possibilidade de existência de vários relacionamentos com as mesmas entidades é interessante, naturalmente desde que esses relacionamentos possuam atributos (ver /CHIC82/). Optou-se então por mantê-la, já que os comandos de consulta à base de dados independem dessa possibilidade.

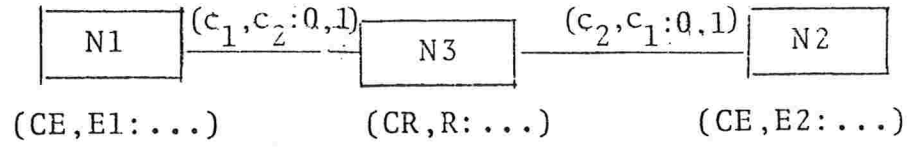
O item 4 associa restrições sobre os relacionamentos representados em um grafo-base-de-dados com as partes de restrição dos rótulos das arestas do grafo-esquema correspondente. Esse conceito permite especificar em um grafo-esquema a classe de relacionamento (classe 1:n, m:n, etc.) associada aos conjuntos-relacionamento representados nesse grafo, da seguinte maneira:

(i) relacionamento 1:1

parte de um diagrama ER:



parte do grafo-esquema correspondente:



Conforme a definição 2.6.4, em qualquer grafo-base-de-dados correspondente a esse grafo-esquema todo E-nó que pertence a N1 pode estar ligado a no máximo um R-nó

que pertence a N3. Da mesma forma, todo E-nó que pertence a N2 pode estar ligado a no máximo um R-nó que pertence a N3.

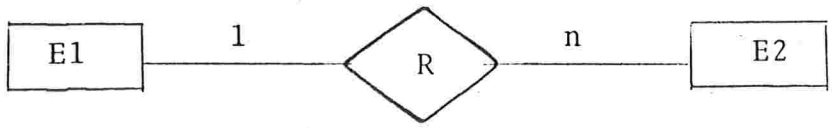
Como a definição 2.6.3 afirma que todo R-nó que pertence a N3 está ligado a um E-nó que pertence a N1 e a um E-nó que pertence a N2, conclui-se que todo E-nó que pertence a N1 pode estar ligado a no máximo a um E-nó que pertence a N2 através de um R-nó que pertence a N3, e vice-versa.

Isso significa que toda entidade de E1 pode estar relacionada por R a no máximo uma entidade de E2, e vice-versa. Esta é a própria definição de relacionamento tipo 1:1.

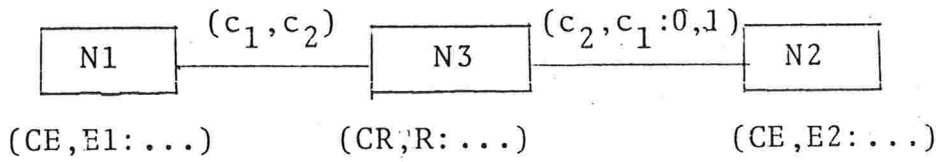
É interessante observar que se as partes de restrição dos rótulos das arestas do grafo-esquema fossem (1,1) ao invés de (0,1) isso implicaria que toda entidade de E1 deveria estar relacionada a exatamente uma entidade de E2, e vice-versa. Essa classe de relacionamento não foi tratada por Chen em seu diagrama ER (introduzido em /CHEN76/) mas existem diagramas ER generalizados que incluem essa possibilidade - ver /WEBR81/).

(ii) relacionamento 1:n

parte de um diagrama ER:



parte do grafo-esquema correspondente:



Como não está especificada a parte de restrição da aresta que liga N1 a N3, os valores subentendidos são $(0, i_{máximo})$. Assim, conforme a definição 2.6.4 em qualquer grafo-base-de-dados que corresponde a esse grafo-esquema todo E-nó que pertence a N1 pode estar ligado a um número qualquer (supondo que o valor de $i_{máximo}$ seja suficientemente grande para não representar uma restrição do ponto de vista prático) de R-nós que pertencem a N3.

Ainda segundo a definição 2.6.4, todo E-nó que pertence a N2 pode estar ligado a no máximo um R-nó que pertence a N3.

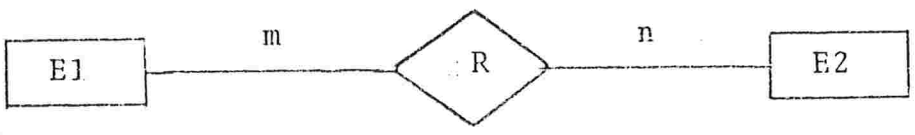
Como a definição 2.6.3 afirma que todo R-nó que pertence a N3 está ligado a um E-nó que pertence a N1 e a um E-nó que pertence a N2, conclui-se que todo E-nó que per-

tence a N1 pode estar ligado a um número qualquer (inclusive zero) de E-nós que pertencem a N2 através de R-nós que pertencem a N3, enquanto que todo E-nó que pertence a N2 pode estar ligado a no máximo um E-nó que pertence a N1 através de um R-nó que pertence a N3. Isso quer dizer que toda entidade de E1 pode estar relacionada por R a um número qualquer de entidades de E2, enquanto que toda entidade de E2 pode estar relacionada por R a no máximo uma entidade de E1, que é a própria definição de relacionamento 1:n.

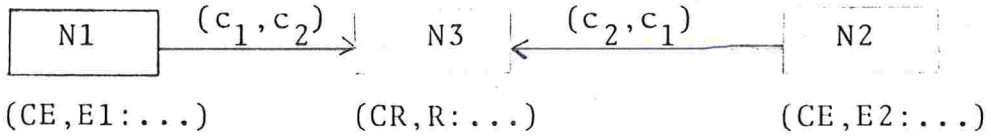
É interessante observar que se a parte de restrição desta aresta que liga N1 a N3 fosse um intervalo específico, por exemplo (3,5), estaria sendo especificada uma classe de relacionamento em que toda entidade de E1 estaria relacionada por R a um número entre 3 e 5 de entidades de E2, enquanto que toda entidade de E2 poderia estar relacionada a no máximo uma entidade de E1. Essa classe de relacionamento também não foi tratada por Chen em seu diagrama ER.

(iii) relacionamento m:n

parte de um diagrama ER:



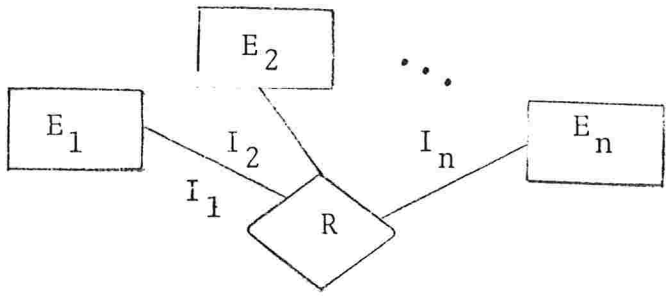
parte do grafo-esquema correspondente:



A verificação de que no grafo-esquema está especificada a classe de relacionamento m:n é análoga aos casos anteriores. Aqui também pode-se observar que a utilização de intervalos específicos nas partes de restrição das arestas permite especificar classes de relacionamento não consideradas por Chen em seu diagrama ER.

(iv) relacionamentos m:n:p, 1:m:n, etc.

O conceito de classe de relacionamento foi introduzido no modelo ER em /CHEN76/ (sem menção a essa terminologia) somente para conjuntos-relacionamento binários (que envolvem apenas dois conjuntos-entidade). Em /LAPY77/ esse conceito foi informalmente estendido para conjuntos-relacionamento de grau maior que 2, de um modo que pode ser resumidamente descrito como segue.



Na representação de um conjunto-relacionamento R que envolve conjuntos-entidade E_1, E_2, ..., E_n ($n \geq 2$) em um diagrama ER (como ilustrado acima), os símbolos

I_1, I_2,..., I_n (que podem ser o dígito 1 ou uma letra m,n, p, etc.) associados respectivamente as linhas do diagrama que indicam a participação de E_1, E_2,..., E_n em R determinam a classe desse relacionamento. Para cada j, se o símbolo I_j é o dígito 1 significa que não existem duas ênuplas de entidades $(e_1,e_2,\ldots,e_n)$ e $(e'_1,e'_2,\ldots,e'_n)$ em R tais que $e_j \neq e'_j$ e $e_i = e'_i$,para i=1,2,...,j-1,j+1,...,n. Em outras palavras, se I_j é o dígito 1 significa que cada ênupla $(e_1,e_2,\ldots,e_{j-1}, e_{j+1},\ldots,e_n)$ do conjunto $Z = E_1 \times E_2 \times \ldots \times E_{j-1} \times E_{j+1} \times \ldots \times E_n$ pode estar relacionada por R a *no máximo* uma entidade do conjunto-entidade E_j.

As classes de relacionamento $I_1:I_2:\ldots:I_n$ tais que I_j é o dígito 1 para algum j ($1 \leq j \leq n$) não podem ser especificadas nos grafos-esquema. Isto porque as restrições que podem ser especificadas nas arestas desses grafos referem-se sempre ao número de ênuplas de um conjunto-relacionamento em que cada entidade de cada conjunto-entidade que dele participa pode aparecer, *independentemente* de quais sejam as outras entidades que constituem essas ênuplas.

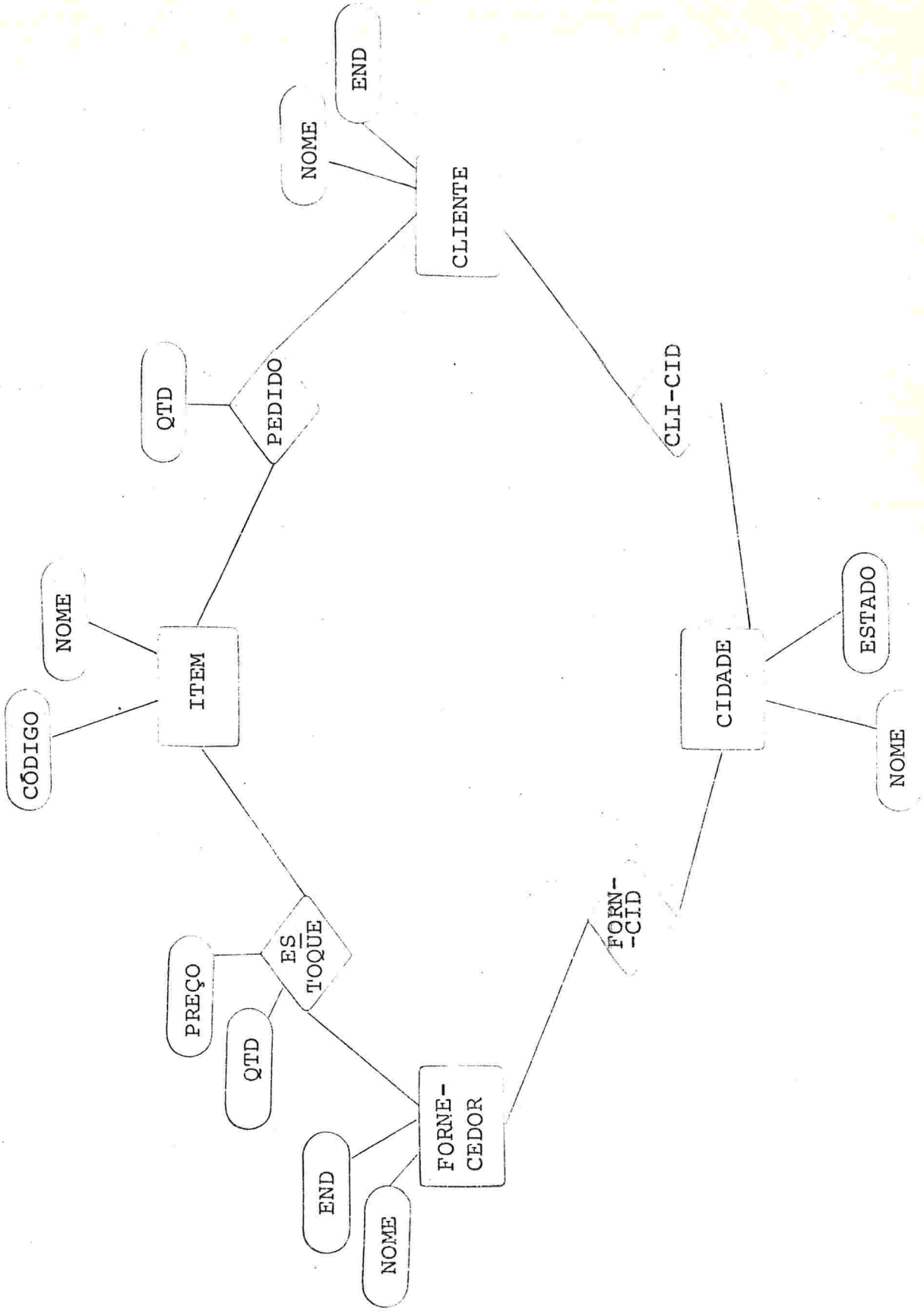

END
NOME
CLIENTE
QTD
PEDIDO
CLI-CID
NOME
CÓDIGO
ITEM
CIDADE
ESTADO
NOME
PREÇO
ESTOQUE
FORN-CID
QTD
END
NOME
FORNE-CEDOR

FIGURA II.A

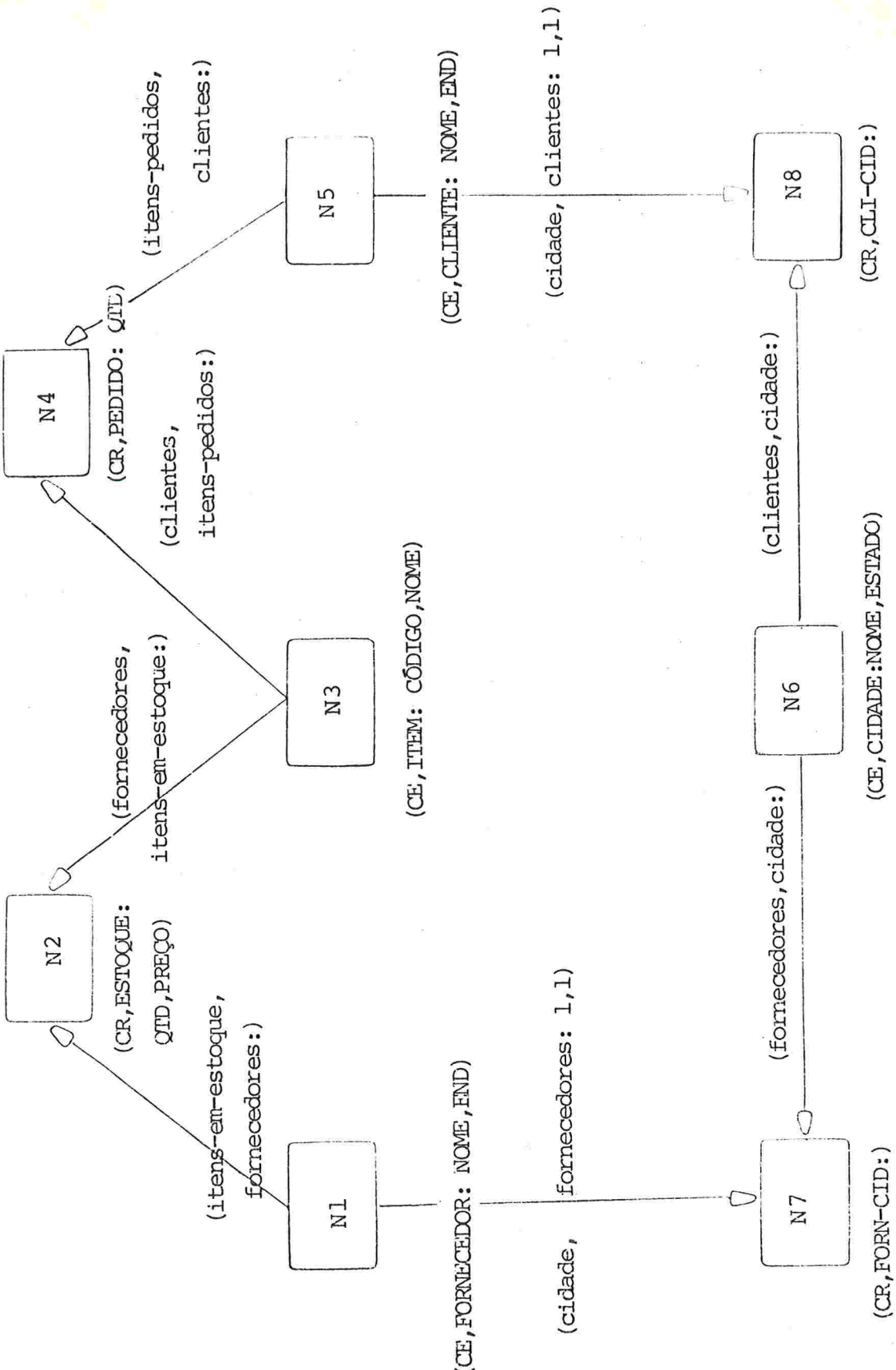

N1
N2
N3
N4
N5
N6
N7
N8
(CE,FORNECEDOR: NOME,END)
(cidade, fornecedores: 1,1)
(itens-em-estoque, fornecedores:)
(CR,ESTOQUE: QTD,PREÇO)
(fornecedores, itens-em-estoque:)
(CE,ITEM: CÓDIGO,NOME)
(clientes, itens-pedidos:)
(CR,PEDIDO: QTD)
(itens-pedidos, clientes:)
(CE,CLIENTE: NOME,END)
(cidade, clientes: 1,1)
(clientes,cidade:)
(fornecedores,cidade:)
(CE,CIDADE:NOME,ESTADO)
(CR,CLI-CID:)
(CR,FORN-CID:)

FIGURA II.B

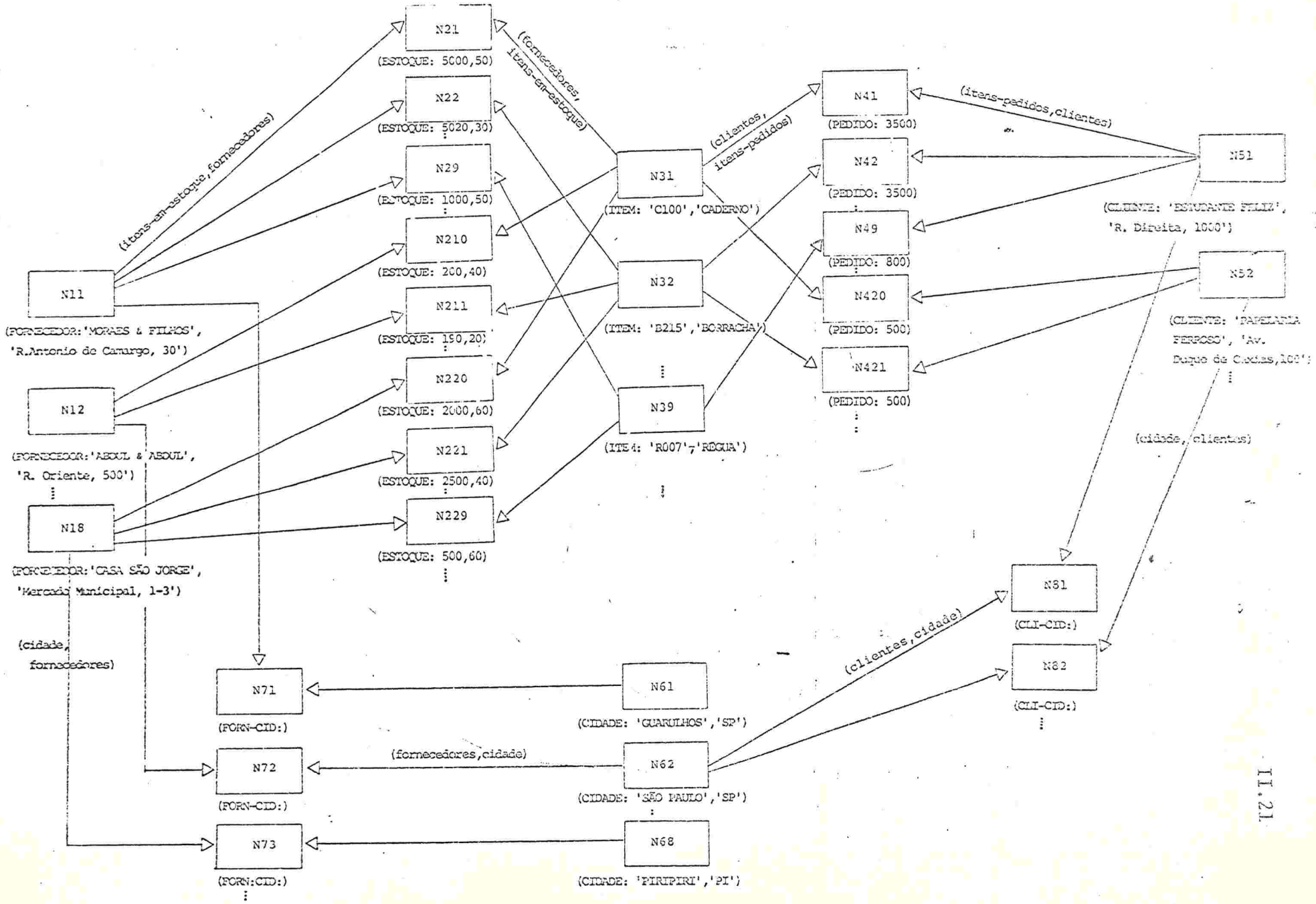

N11
(FORNECEDOR:'MORAES & FILHOS', 'R.Antonio de Camargo, 30')
N12
(FORNECEDOR:'ABDUL & ABDUL', 'R. Oriente, 500')
N18
(FORNECEDOR:'CASA SÃO JORGE', 'Mercado Municipal, 1-3')
(itens-em-estoque,fornecedores)
(cidade, fornecedores)
N21
(ESTOQUE: 5000,50)
N22
(ESTOQUE: 5020,30)
N29
(ESTOQUE: 1000,50)
N210
(ESTOQUE: 200,40)
N211
(ESTOQUE: 190,20)
N220
(ESTOQUE: 2000,60)
N221
(ESTOQUE: 2500,40)
N229
(ESTOQUE: 500,60)
(fornecedores, itens-em-estoque)
N31
(ITEM: 'C100','CADERNO')
N32
(ITEM: 'B215','BORRACHA')
N39
(ITEM: 'R007','RÉGUA')
(clientes, itens-pedidos)
N41
(PEDIDO: 3500)
N42
(PEDIDO: 3500)
N49
(PEDIDO: 800)
N420
(PEDIDO: 500)
N421
(PEDIDO: 500)
(itens-pedidos,clientes)
N51
(CLIENTE: 'ESTUDANTE FELIZ', 'R. Direita, 1000')
N52
(CLIENTE: 'PAPELARIA FERROSO', 'Av. Duque de Caxias,100')
(cidade, clientes)
N71
(FORN-CID:)
N72
(FORN-CID:)
N73
(FORN:CID:)
N61
(CIDADE: 'GUARULHOS','SP')
(fornecedores,cidade)
N62
(CIDADE: 'SÃO PAULO','SP')
N68
(CIDADE: 'PIRIPIRI','PI')
(clientes,cidade)
N81
(CLI-CID:)
N82
(CLI-CID:)

FIGURA II.C

3 - INTRODUÇÃO ÀS CONSULTAS

Antes de apresentar as definições formais, vamos considerar um exemplo de uma consulta ("query") formulada na linguagem GORDAS, baseada no modelo de dados representado nas figuras II.A e II.B.

Seja a seguinte consulta: "obtenha o nome de todos os fornecedores do estado de São Paulo".

Em GORDAS essa consulta poderia ser formulada da seguinte maneira:

```
get NOME of FORNECEDOR
  where ESTADO of cidade of FORNECEDOR = 'SP' .
```

Considerando a parte de uma base-de-dados representada na figura II.C o resultado dessa consulta é o conjunto {'MORAES & FILHOS', 'ABDUL & ABDUL'} .

Esse exemplo simples permite identificar algumas das principais construções da linguagem GORDAS, a saber:

(i) As consultas são constituídas de duas partes:

- a cláusula get, que especifica o conjunto de nós do grafo-base-de-dados que deve ser utilizado como base para a

consulta (no exemplo, o conjunto de nós que pertence ao nó FORNECEDOR do grafo-esquema) e os dados que devem ser obtidos desses nós (no exemplo, o valor do atributo NOME);

- a cláusula where, que especifica um predicado que deve ser utilizado para selecionar os nós do conjunto definido na cláusula get a partir dos quais deve ser construído o resultado da consulta.

Na verdade, pode-se especificar vários conjuntos de nós na cláusula get, e o predicado da cláusula where permite selecionar elementos do produto cartesiano desses conjuntos.

(ii) Tanto a cláusula get como o predicado da cláusula where são formados por construções básicas da linguagem chamadas expressões-de-caminho ("path expressions"), que no exemplo são NOME of FORNECEDOR e ESTADO of cidade of FORNECEDOR .

As expressões-de-caminho representam valores da base-de-dados. Por exemplo, a expressão NOME of FORNECEDOR representa o valor do atributo NOME para cada entidade do conjunto FORNECEDOR, e a expressão ESTADO of cidade of FORNECEDOR representa, para cada entidade f do conjunto FORNECEDOR, o valor do atributo ESTADO da entidade c do conjunto CIDADE que está relacionada com f através do relacionamento FORN-CID. Essa ligação entre as entidades c e f

está especificada pelo nome de conexão cidade .

Introduzidos esses conceitos básicos, pode-se descrever informalmente o processo de construção do resultado da consulta apresentada inicialmente, da seguinte maneira:

Para cada entidade do conjunto FORNECEDOR, é calculado o valor da expressão-de-caminho ESTADO of cidade of FORNECEDOR . Se esse valor for igual a 'SP', a entidade é selecionada e o valor da expressão-de-caminho NOME of FORNECEDOR é calculado para a mesma e incluído no resultado; caso contrário, a entidade é ignorada.

4 - VALORES TRATADOS PELA LINGUAGEM GORDAS

Na linguagem GORDAS, tal como foi descrita pelos seus autores em /ELWI81/, um valor pode ser um conjunto ou um valor isolado (escalar). Por exemplo, considerando os valores apresentados na Seção anterior, o valor 'SP' é um valor isolado e o valor {'MORAES & FILHOS', 'ABDUL & ABDUL'} é (obviamente) um conjunto.

Essa distinção entre valores isolados e conjuntos complica a descrição da linguagem, tanto que existe uma inconsistência na descrição dos próprios autores gerada por essa distinção, a qual será descrita na Seção 13.

Optou-se então por modificar a linguagem, passando a considerar apenas os valores que são conjuntos. Assim, na versão da linguagem aqui descrita, o valor 'SP' citado anteriormente é considerado como o conjunto unitário {'SP'} . Essa modificação torna a linguagem mais "uniforme", e permite também a utilização de atributos "multi-valorados" ("multi-valued attributes") de forma natural.

Não há na descrição dos autores nenhuma indicação sobre os valores "primitivos" tratados pela linguagem. Na versão da linguagem para o modelo ECR descrita em /ELMA81/, são tratados os valores tipo string, integer, real e date.

Por simplicidade optou-se por considerar apenas a exis-

tência dos valores primitivos tipo n (números inteiros com sinal) e tipo a (cadeias de caracteres).

Definição 4.1 (valor-básico)

⌐ Um *valor-básico* pode ser:

(i) um número inteiro com sinal;

(ii) uma cadeia de caracteres quaisquer;

(iii) um nó de um grafo-base-de-dados.

O *tipo* do valor-básico no primeiro caso é n, no segundo é a e no terceiro é nó. ⌙

No restante deste texto, os valores-básicos tipo n serão denotados por cadeias de dígitos (eventualmente precedidas por um sinal) da forma usual (p. ex.: 3,5,-7,17, etc.) e os valores-básicos tipo a serão denotados pelas próprias cadeias de caracteres que os constituem, delimitadas por apóstrofes (p.ex.: 'A', 'CADEIA', 'VALOR-BÁSICO', etc.).

Os valores-básicos tipo nó são valores "internos" (não são tratados pelo usuário) que correspondem a nós de um grafo-base-de-dados. Portanto, um valor-básico tipo nó corresponde a uma entidade ou a um relacionamento da base de dados representada pelo referido grafo. Dado um diagrama de representa um grafo-base-de-dados específico, os valores-básicos tipo nó correspondentes poderão ser denotados pelos símbolos associados aos nós nesse diagrama. Por exemplo, considerando o grafo-base-de-da

dos representado na figura II.C, pode-se dizer que os valores-básicos tipo nó correspondentes são N11, N39, N41, etc..

A definição seguinte formaliza o conceito de valor , que até aqui vinha sendo tratada intuitivamente.

Definição 4.2 (valor)

⌜

a) Um *valor* tipo t é um conjunto de valores-básicos de mesmo tipo t ou um conjunto de valores-ênupla de mesmo tipo t.

b) Sejam $u_1,u_2,\ldots,u_n$ valores cujos tipos são respectivamente $t_1,t_2,\ldots,t_n$. A ênupla ordenada $(u_1,u_2,\ldots,u_n)$ é um *valor-ênupla* cujo tipo é $\langle t_1,t_2,\ldots,t_n\rangle$. ⌟

No restante deste texto, os valores serão denotados através do uso de chaves da forma usual (p. ex., o valor cujos elementos são 3,5,7 será denotado por {3,5,7}). Os valores que são conjuntos unitários poderão ser também denotados pela denotação do elemento que o constitui (assim, 5 e {5} denotam o mesmo valor). Os valores-ênupla serão denotados por $\langle u_1,u_2,\ldots,u_n\rangle$, onde $u_1,u_2,\ldots,u_n$ são as denotações dos valores que são seus componentes. Os valores-ênupla poderão ser também referenciados como *ênuplas de valores* ou simplesmente *ênuplas*.

5 - EXPRESSÕES-DE-CAMINHO

As definições apresentadas a seguir diferem das definições correspondentes na descrição dos autores nos seguintes pontos:

(i) Foram eliminadas algumas falhas das definições dos autores, que não prevêem algumas construções utilizadas nos exemplos apresentados no próprio texto que contem as definições.

(ii) Foi generalizado o conceito de tipo associado a uma expressão-de-caminho, que ficou diretamente relacionado com o tipo de valor representado pela expressão. Dessa forma pode-se evitar a construção de expressões-de-conjunto (ver seção 6) que definem operações com valores incompatíveis (como por exemplo a união de conjuntos de ênuplas em que o número de componentes das ênuplas de cada conjunto é diferente), o que está feito de forma muito restrita na descrição dos autores.

(iii) Foi formalmente introduzida a possibilidade de acréscimo de um sufixo a um nome-de-nó para formar expressões-de-caminho, que os autores introduziram informalmente (através de uma única frase, sem nem ao menos dar um exemplo.

(iv) Foi introduzido o conceito de variável, para simplificar a descrição da semântica da linguagem. De fato não foi in

troduzido nenhum elemento adicional na liguagem, tratando-se apenas de uma forma diferente de interpretar um elemento já existente.

Uma expressão-de-caminho é tipicamente composta por um prefixo ("prefix") e por uma variável, que serão definidos inicialmente.

Definição 5.1 (prefixo)

⌈Um *prefixo* em um grafo-esquema GE é definido recursivamente da seguinte maneira:

1) Um nome de atributo A de um nó X de GE é um prefixo cujo tipo é o tipo dos valores que o atributo A pode assumir.

2) Um nome de conexão c associado a uma aresta de GE é um prefixo tipo nó.

3) Se f é um prefixo e c um nome de conexão, então f of c é um prefixo de mesmo tipo que f.

4) Se $f_1, f_2, \ldots, f_n$, n>1, são prefixos, então $\langle f_1, f_2, \ldots, f_n \rangle$ (uma ênupla de prefixos) é um prefixo cujo tipo é $\langle t_1, t_2, \ldots, t_n \rangle$, onde t_i é o tipo de f_i, $1 \leq i \leq n$.

5) Se f é um prefixo, c um nome de conexão e p um predicado (ver definição 6.3), então c:(p) é um prefixo tipo nó e f of c:(p) é um prefixo de mesmo tipo que f.

Um predicado p assim inserido em um prefixo é chamado *predicado de restrição* ("restricting predicate").

6) Se f é um prefixo tipo t, então count f é um prefixo tipo n e <count,f> é um prefixo tipo <n,t>. ⌋

Definição 5.2 (variável)

⌈Uma *variável* em um grafo-esquema GE é uma expressão da forma nome ou nome.j , onde nome é o nome-do-nó para algum nó X de GE, e j é uma cadeia de dígitos em que o primeiro é diferente de zero. Diz-se que o nó X cujo nome é referenciado na variável é o *domínio* dessa variável⌋.

Alguns exemplos de variáveis no grafo-esquema da figura II-B são: FORNECEDOR, ITEM.2, ESTOQUE.10, CIDADE.

Uma variável assume valores tipo nó. Se v é uma variável em um grafo-esquema GE cujo domínio é X, e GBD é um grafo-base-de-dados correspondente a GE, os valores que v pode assumir são os nós de GBD que pertencem a X.

Na descrição dos autores não é utilizado esse conceito de variável. Ao contrário, há uma afirmação explícita de que a linguagem GORDAS não utiliza variáveis. Porém, as construções definidas em 5.2 correspondem exatamente ao conceito de variável utilizado nas linguagens em geral, pois são identificadores especificados nos comandos da linguagem que representam valores mani

pulados durante a execução desses comandos. Talvez essa afirmação dos autores se refira ao fato de que na linguagem GORDAS não é possível definir variáveis ou atribuir valores às mesmas através de comandos específicos: as variáveis ficam inplicitamente definidas quando é especificada uma consulta e seus valores são atribuídos automaticamente durante a execução dessa consulta (ver seção 8).

Definição 5.3 (expressão-de-caminho)

⌜Uma *expressão-de-caminho* em um grafo-esquema GE é uma variável em GE ou uma expressão da forma f of v , onde f é um prefixo em GE e v uma variável em GE. No primeiro caso o *tipo* da expressão-de-caminho é nó, e no segundo é igual ao tipo do prefixo f.

Diz-se que v é a variável *associada* à expressão-de-caminho.⌟

As expressões-de-caminho são construções básicas da linguagem, pois através delas é que são "extraídos" valores da base de dados. Esses valores podem ser valores de atributos de entidades ou relacionamentos, valores tipo nó que representam entidades ou relacionamentos ou ainda valores compostos (ênuplas) desses valores já citados.

Uma expressão-de-caminho pode ser interpretada como uma função que para cada valor da variável associada fornece um

valor "extraído" da base de dados. Essa função é definida pelo prefixo da expressão-de-caminho ou é a função identidade, caso a expressão não tenha prefixo.

Dado um grafo-base-de-dados GBD correspondente a um certo grafo-esquema GE, o valor de uma expressão-de-caminho em GE é calculado para cada nó de GBD que pertence ao domínio da variável associada à expressão através de um "percurso" realizado sobre o grafo-base-de-dados, a partir daquele nó. Note-se que esse "percurso" não se trata necessariamente de um passeio linear sobre o grafo (ver /Lucc77/), mas sim de um passeio ramificado (em forma de árvore), em que os ramos com origem comum são percorridos"em paralelo".

Na seção 12 será apresentada uma definição formal do processo de cálculo do valor de uma expressão-de-caminho. Os exemplos apresentados a seguir introduzem informalmente esse processo.

Exemplos:

Todos os exemplos seguintes referem-se ao grafo-esquema GE da figura II-B e ao grafo-base-de-dados GBD representado (parcialmente) na figura II.C.

Não foram incluídos exemplos para ilustrar o uso de predicados de restrição, que serão apresentados na seção 7, após a definição de predicado.

Deve-se recordar que os valores (e seus componentes) tratados pela linguagem são sempre conjuntos (cf. seção 4). Na discussão que segue cada exemplo, os valores que são conjuntos unitários estão representados pelos seus elementos, para simplificar o texto.

a) NOME of CLIENTE (tipo: a)

Para cada nó de GBD que pertence ao nó de GE cujo nome é CLIENTE, essa expressão assume o valor do atributo NOME. Assim, para o nó 51 (fig.II.C) ela assume o valor 'ESTUDANTE FELIZ' e para o nó 52 ela assume o valor 'PAPELARIA FERROSO'.

b) <NOME,END> of CLIENTE.2 (tipo: <a,a>)

Para cada nó de GBD que pertence ao nó de GE cujo nome é CLIENTE, que é o domínio da variável CLIENTE.2, essa expressão assume como valor o par formado pelos valores dos atributos NOME e END. Assim, para o nó 51 ela assume o valor <'ESTUDANTE FELIZ', 'R.Direita,1000'> e para o nó 52 ela assume o valor < 'PAPELARIA FERROSO', 'Av. Duque de Caxias,100'>.

c) ITEM (tipo: nó)

Para cada nó de GBD que pertence ao nó de GE cujo nome é ITEM, essa expressão assume como valor o próprio nó.

d) fornecedores of ITEM (tipo: nó)

Para cada nó de GBD que pertence ao nó de GE cujo no

me é ITEM, essa expressão assume como valor o conjunto de nós de GBD que pertencem ao nó FORNECEDOR (de GE) e que estão "ligados" (esse conceito se tornará claro ao longo do texto) ao nó em questão através de nós que pertencem ao nó ESTOQUE de GE. Isto porque o nome de conexão fornecedores "liga" o nó ITEM ao nó FORNECEDOR (nesse sentido) através do nó ESTOQUE (deve-se recordar que o primeiro nome de conexão corresponde ao mesmo sentido da aresta à qual ele está associado, e o segundo nome corresponde ao sentido contrário).

Assim, para os nós 31 e 32 essa expressão assume o valor {N11, N12, N18}, e para o nó 39 ela assume o valor {N11, N18}.

e) <NOME,QTD> of itens-em-estoque of FORNECEDOR (tipo:<a,n>)

Para cada nó de GBD que pertence ao nó de GE cujo nome é FORNECEDOR, essa expressão assume como valor o conjunto dos pares formados pelos valores dos atributos NOME (atributo do nó ITEM de GE) e QTD (atributo do nó ESTOQUE de GE) para os nós "ligados" ao nó em questão através do nome de conexão itens-em-estoque.

Assim, para o nó 11 ela assume o valor {<'CADERNO',5000>, <'BORRACHA',5020>, <'RÉGUA',1000>}, para o nó 12 ela assume o valor {<'CADERNO',200>,<'BORRACHA',190>} e para o nó 18 ela assume o valor {<'CADERNO',2000> , <'BORRACHA',2500> , <'RÉGUA',500>}.

f) <NOME, <NOME,ESTADO> of cidade of clientes,
<NOME,ESTADO> of cidade of fornecedores> of ITEM
(tipo: <a, <a,a>, <a,a>>)

Este exemplo é análogo ao anterior, porém envolvendo vários nomes de conexão.

Para os nós 31 e 32 essa expressão assume o valor <'CADERNO', <'SAO PAULO', 'SP'>, {<'GUARULHOS', 'SP'>, <'SÃO PAULO', 'SP'>, <'PIRIPIRI', 'PI'>} > e para o nó 39 ela assume o valor <'RÉGUA' , <'SÃO PAULO', 'SP'> , {<'GUARULHOS','SP'>, <'PIRIPIRI', 'PI'>}>

g) count fornecedores of ITEM (tipo: n)

Para cada nó de GBD que pertence ao nó de GE cujo nome é ITEM essa expressão assume como valor o número de nós de GBD que pertencem ao nó de GE cujo nome é FORNECEDOR e que estão "ligados" ao nó em questão através do nome de conexão fornecedores.

Assim, para os nós 31 e 32 ela assume o valor 3 e para o nó 39 ela assume o valor 2.

h) count ESTADO of cidade of fornecedores of ITEM (tipo: n)

Este exemplo é análogo ao anterior, exceto que a contagem não se refere diretamente a um conjunto de nós mas sim a um conjunto de valores de atributos.

Esta expressão assume o valor 2 para os nós 31, 32 e 39.

i) <NOME, count fornecedores> of ITEM (tipo: <a,n>)

Este exemplo é uma extensão do exemplo g.

Para o nó 31 ela assume o valor <'CADERNO',3>, para o nó 32 ela assume o valor <'BORRACHA',3> e para o nó 39 ela assume o valor <'RÉGUA',2>.

j) <NOME of itens-pedidos, count fornecedores of itens-pedidos > of CLIENTE (tipo: <a,n>)

Para o nó 51 essa expressão assume o valor <{'CADERNO', 'BORRACHA', 'RÉGUA'},3> e para o nó 52 o valor <{ 'BORRACHA', 'CADERNO'},3>.

k) <NOME, count fornecedores> of itens-pedidos of CLIENTE (tipo: <a,n>)

Este exemplo, apesar de sua semelhança com o anterior, tem um significado completamente diferente.

Para o nó 51, ela assume o valor {<'CADERNO',3>, <'BORRACHA',3>, <'RÉGUA',2>} e para o nó 52 o valor {<'CADERNO',3>, <'BORRACHA',3> }.

ℓ) <count <NOME,QTD>> of itens-em-estoque of FORNECEDOR (tipo: <n, <a,n>>)

Essa expressão assume os seguintes valores:

- nó 11:

<3,{<'CADERNO',5000>,<'BORRACHA',5020>,<'RÉGUA',1000>}>

- nó 12:

<2,{<'CADERNO',200>,<'BORRACHA',190>}>

- nó 18:

<3,{<'CADERNO',2000>,<'BORRACHA',2500>,<'RÉGUA',500>}>

6 - PREDICADOS

Os predicados são utilizados na linguagem GORDAS para selecionar os nós de um grafo-base-de-dados (que representam entidades ou relacionamentos) dos quais devem ser "extraídos" valores (através de expressões-de-caminho) para formar o resultado de uma consulta.

Eles se baseiam em comparações entre valores representados por expressões-de-conjunto ("set expression"), que por sua vez são especificadas através de operações com valores constantes representados por expressões-de-constantes ("constant expression") ou valores obtidos da base de dados através de expressões-de-caminho ou de consultas.

As definições aqui apresentadas diferem das definições existentes na descrição dos autores nos seguintes pontos:

(i) Foram eliminadas algumas falhas das definições dos autores, que não prevêem algumas construções utilizadas nos exemplos apresentados no próprio texto que contêm as descrições, tais como o uso do operador count nas expressões-de-conjunto (def. 6.2) e o uso de parênteses em geral.

(ii) Foi generalizado o conceito de tipo associado a umaexpressão-de-constantes, coerentemente com o que foi feito na seção 5 para as expressões-de-caminho.

(iii) As definições foram compatibilizadas com o conceito introduzido na seção 4 de que os valores tratados pela linguagem são sempre conjuntos. Em particular as definições relativas ao uso dos operadores $<$, $\leq$, $>$ e $\geq$ ficaram completamente diferentes das definições dos autores, conforme será descrito posteriormente.

(iv) Nas expressões-de-constantes (def. 6.1) foi eliminada a possibilidade de serem construídos conjuntos de conjuntos, já que nenhum outro elemento da linguagem pode representar valores dessa forma.

(v) Foi definida a seqüência de execução das operações especificadas nas expressões-de-conjunto e nos predicados.

Definição 6.1 (Expressão-de-constantes)

1) Uma cadeia de dígitos ou uma cadeia de dígitos precedida pelo símbolo "-" é uma *constante* tipo n.

2) Uma cadeia de caracteres quaisquer, delimitada por apóstrofes é uma *constante* tipo a.

3) Se $k_1, k_2, \ldots, k_n$, $n \geq 1$, são constantes de mesmo tipo t, então $\{k_1, k_2, \ldots, k_n\}$ (um conjunto de constantes) é uma *expressão-de-constantes* tipo t.

4) Toda constante tipo t é uma *expressão-de-constantes* tipo t.

5) Se $k_1, k_2, \ldots, k_n$, $n \geq 2$, são expressões-de-constantes, então $\langle k_1, k_2, \ldots, k_n \rangle$ (uma ênupla de expressões-de-constantes) é uma *constante* tipo $\langle t_1, t_2, \ldots, t_n \rangle$, onde $t_1, t_2, \ldots, t_n$ são respectivamente os tipos de $k_1, k_2, \ldots, k_n$. ⌋

Uma expressão-de-constantes representa um valor, e o tipo da expressão é o tipo do valor por ela representado.

Como os valores tratados pela linguagem são sempre conjuntos (conforme seção 4), toda constante representa o conjunto unitário cujo elemento é denotado pela constante, e toda expressão-de-constantes representa a união dos valores denotados pelas constantes que a constituem.

Nos exemplos apresentados a seguir as expressões-de-constantes agrupadas em cada item representam o mesmo valor.

Exemplos:

a) 5, {5}

b) <'XYZ', 37> , <'XYZ',{37}>, <{'XYZ'}, 37>, <{'XYZ'},{37} > , {<'XYZ', 37>},{<'XYZ',{37} > } ,etc.

c) <3, <'ABC', -7>>, <{3} ,<'ABC', -7>>, <3,<'ABC',{-7} >> , {<{3},{<{'ABC'}, {-7}>}>}, etc.

Definição 6.2 (Expressão-de-conjunto)

⌈

1) Qualquer prefixo, expressão-de-caminho ou expressão-de-cons

tantes tipo t é uma *expressão-de-conjunto* tipo t.

2) Se q é uma consulta tipo t (ver Definição 8.1) então (q) é uma *expressão-de-conjunto* tipo t e count (q) é uma *expressão-de-conjunto* tipo n .

3) Se s_1 e s_2 são expressões-de-conjunto de mesmo tipo t, então s_1 union s_2 , s_1 inters s_2 e s_1 dif s_2 são *expressões-de-conjunto* tipo t.

4) Se s é uma expressão-de-conjunto tipo t, então (s) é uma *expressão-de-conjunto* tipo t e count (s) é uma *expressão-de-conjunto* tipo n. |

Uma expressão-de-conjunto representa valores, e o tipo da expressão é o tipo dos valores que ela pode assumir.

Esses valores podem ser valores da base de dados, representados por prefixos, expressões-de-caminho ou consultas, valores constantes representados por expressões-de-constantes ou ainda valores correspondentes aos resultados de operações realizadas com esses valores já citados.

As operações são especificadas nas expressões-de-conjunto através dos operadores union, inters e dif, que fornecem respectivamente como resultado a união, intersecção e diferença dos valores (que são conjuntos) aos quais eles são aplicados, ou através do operador count, que fornece como resultado o conjunto unitário cujo elemento é a cardinalidade do valor ao qual

ele é aplicado (cabe lembrar que todo valor é um conjunto, cf. seção 4).

A seqüência de operações é definida pela seguinte regra:

Regra 6.A

⌜ Numa expressão-de-conjunto os parênteses determinam a seqüência das operações da forma usual. Quando não há parênreses as operações são interpretadas da esquerda para a direita, isto é, uma expressão da forma $s_1\ op_1\ s_2 \ldots op_n\ s_{n+1}$, onde $n \geq 2$, op_i $(1 \leq i \leq n)$ é um dos operadores union, inters ou dif e s_i $(1 \leq i \leq n+1)$ é uma expressão-de-conjunto tal que não existem duas expressões-de-conjunto s_{i1} e s_{i2} tais que $s_i = s_{i1}\ op\ s_{i2}$, onde op é um dos operadores union, inters ou dif, é equivalente à expressão

$$((\ldots((s_1\ op_1\ s_2)\ op_2\ s_3)\ op_3 \ldots s_n)\ op_n\ s_{n+1}).$$ ⌟

Definição 6.3 (Predicados)

⌜ 1) Se s_1 e s_2 são expressões-de-conjunto de mesmo tipo, então $s_1 = s_2$, $s_1 \neq s_2$ e s_1 includes s_2 são *predicados*.

2) Se s_1 e s_2 são expressões-de-conjunto de mesmo tipo t, onde $t = n$ ou $t = a$, então $s_1 < s_2$, $s_1 \leq s_2$, $s_1 > s_2$ e $s_1 \geq s_2$ são *predicados*.

3) Se p_1 e p_2 são predicados, então p_1 and p_2 e p_1 or p_2 são *predicados*.

4) Se p é um predicado, então not (p) e (p) são *predicados*. ⊣

Exemplos:

a) NOME of ITEM = 'CADERNO'

b) cidade of CLIENTE = cidade of FORNECEDOR

c) itens-em-estoque of FORNECEDOR includes
itens-pedidos of CLIENTE

d) count (itens-em-estoque of FORNECEDOR inters
itens-pedidos of CLIENTE) > 3

e) <NOME,ESTADO> of cidade of CLIENTE
= <'PIRIPIRI', 'PI'>

f) (NOME of itens-em-estoque of FORNECEDOR
union {'CANETA', 'TINTA'}) =
NOME of itens-pedidos of CLIENTE

g) (<NOME,END> of fornecedores of ITEM dif
<NOME,END> of clientes of ITEM) includes
{<'MORAES & FILHOS', 'R. Antonio de Camargo, 30'>}

Um predicado é falso ou verdadeiro para cada particular combinação de valores das expressões-de-conjunto que o constituem.

O cálculo de um predicado é sempre feito a partir de comparações entre valores, através dos operadores =, $\neq$, <, $\leq$, >, $\geq$. Como os valores tratados pela linguagem são sempre conjuntos, os operadores = e $\neq$ especificam as operações usuais de teste de igualdade ou desigualdade entre conjuntos. Já os operadores <, $\leq$, > e $\geq$ são tratados de forma especial na linguagem, descrita pela seguinte regra:

Regra 6.B

⌜ No cálculo de um predicado da forma s_1 op s_2 , onde s_1 e s_2 são expressões-de-conjunto e op é um dos operadores <, $\leq$, > ou $\geq$, se os valores de s_1 e s_2 são conjuntos-unitários é realizada a comparação entre os elementos desses conjuntos da forma usual. Se um dos valores é o conjunto-vazio ou um conjunto com mais de um elemento o cálculo do predicado é interrompido com uma indicação de erro. ⌟

Conforme citado na introdução desta seção, as definições relativas ao uso dos operadores <, $\leq$, > e $\geq$ aqui apresentadas diferem bastante das definições correspondentes na descrição dos autores. Na seção 13 essas diferenças serão descritas e justificadas.

As comparações entre valores resultam em valores ló

gicos (falso ou verdadeiro), aos quais são aplicados operadores lógicos (and, or, not) para formar o resultado do predicado. O cálculo das operações lógicas obedece às seguintes regras:

Regra 6.C

⌐ Num predicado os parênteses determinam a seqüência das operações lógicas da forma usual. Quando não há parênteses as operações são interpretadas da esquerda para a direita, isto é, uma expressão da forma $p_1\ op_1\ p_2 \ldots op_n\ p_{n+1}$, onde $n \geq 2$, op_i $(1 \leq i \leq n)$ é o operador and ou o operador or, e p_i $(1 \leq i \leq n+1)$ é um predicado tal que não existem dois predicados p_{i1} e p_{i2} tais que $p_i = p_{i1}$ and p_{i2} ou $p_i = p_{i1}$ or p_{i2}, é equivalente à expressão $((\ldots((p_1\ op_1\ p_2)op_2\ p_3)op_3 \ldots p_n)op_n\ p_{n+1})$. ⌙

Regra 6.D

⌐ No cálculo de um predicado da forma p_1 and p_2 o predicado p_1 é calculado em primeiro lugar. Se p_1 resultar no valor falso o predicado p_2 *não é calculado*, e o resultado é o valor falso.

No cálculo de um predicado da forma p_1 or p_2 o predicado p_1 é calculado em primeiro lugar. Se p_1 resultar no valor verdadeiro o predicado p_2 *não é calculado*, e o resultado é o valor verdadeiro. ⌙

A regra 6.C define a seqüência de execução das operações lógicas. A regra 6.D define um processo de cálculo que, além de otimizar o cálculo dos predicados, permite ao usuário evitar a ocorrência do erro descrito na regra 6.B, conforme será descrito a seguir.

Sempre que for necessário utilizar um predicado do tipo s_1 op s_2 (op ε {<, $\leq$, >, $\geq$}) em que s_1 ou s_2 podem resultar em valores que não são conjuntos unitários, ele pode ser composto através dos operadores and e or com os predicados count(s_1)=1 e count(s_2)=1 , que se forem verdadeiros garantem que a comparação entre s_1 e s_2 não irá resultar em erro. Por exemplo, um predicado do tipo $s_1 > s_2$ cujo valor deve ser considerado falso se s_1 e s_2 não resultarem em conjuntos unitários deve ser especificado pela seguinte composição:

count(s_1) = 1 and count(s_2) = 1 and $s_1 > s_2$.

Se no mesmo exemplo o valor do predicado devesse ser considerado verdadeiro caso s_1 ou s_2 não resultasse em um conjunto unitário, o predicado deveria se especificado pela seguinte composição:

count(s_1) $\neq$ 1 or count(s_2) $\neq$ 1 or $s_1 > s_2$.

7 - PREFIXOS LIVRES E PREDICADOS DE RESTRIÇÃO

Na seção anterior foi definido que uma expressão-de--conjunto pode ser formada simplesmente por um prefixo.

Não está explicitamente definida no texto dos autores a semântica dessas expressões: ela é introduzida apenas informalmente através de alguns exemplos.

Essas expressões-de-conjunto formadas simplesmente por um prefixo, que passarão a ser denominadas prefixos livres, são necessárias para a construção de predicados de restrição, que são predicados inseridos em expressões-de-caminho. Sua função nesses predicados será formalmente definida na seção 12, como parte da definição da semântica das expressões-de-caminho.

O objetivo desta seção é introduzir informalmente o conceito e estabelecer precisamente a ligação entre os prefixos livres e os predicados onde eles são utilizados. Para isso é necessário inicialmente formalizar algumas relações existentes entre os elementos da linguagem que formam os predicados.

Definição 7.1 (fatores de expressões-de-conjunto)

⌜Sejam s e t duas expressões-de-conjunto em um grafo--esquema GE.

Diz-se que t é um *fator* de s se e somente se:

1) s = t

2) Existe uma expressão-de-conjunto s' tal que s=(s') ou s= count(s') , e t é um fator de s'.

3) Existem duas expressões-de-conjunto s_1 e s_2 tais que s = s_1 union s_2 ou s= s_1 inters s_2 ou s= s_1 dif s_2 , e t é um fator de s_1 ou de s_2. ⌋

Exemplo:

As expressões-de-conjunto {'CANETA','TINTA'} e NOME of ITEM são fatores da expressão-de-conjunto NOME of ITEM inters {'CANETA','TINTA'} , e cada uma dessas expressões é um fator de si própria.

Definição 7.2 (componentes de um predicado)

⌈ Sejam p um predicado e s uma expressão-de-conjunto em um grafo-esquema GE.

Diz-se que s é um *componente imediato* de p se e somente se:

1) Existem expressões-de-conjunto s_1 e s_2 tais que p= s_1 op s_2, onde op é um dos operadores =, ≠, includes, <, ≤, >, ≥, e s é um fator de s_1 ou s_2.

2) Existem predicados p_1 e p_2 tais que p = p_1 and p_2 ou

$p = p_1$ or p_2 , e s é um componente imediato de p_1 ou de p_2.

3) Existe um predicado p' tal que p=(p') ou p=not(p') e s é um componente imediato de p'. ⌟

Exemplo:

As expressões-de-conjunto {'CANETA','TINTA'} , NOME of ITEM e 1 (entre outras) são componentes imediatos do predicado count (NOME of ITEM inters {'CANETA','TINTA'}) = 1 .

Notas:

(i) Os conceitos de fator de uma expressão-de-conjunto e de componente imediato de um predicado poderiam ter sido introduzidos diretamente nas definições de expressão-de-conjunto (def. 6.2) e de predicado (def. 6.3). Optou-se por introduzir separadamente esses conceitos para não complicar aquelas definições.

(ii) Na definição 7.2, o termo *imediato* tem por objetivo salientar que o conceito que está sendo definido não corresponde ao conceito usual de *componente*. Por exemplo, a expressão-de-caminho NOME of cidade *não é* um componente imediato do predicado

clientes:(NOME of cidade = 'SÃO PAULO') of ITEM includes CLIENTE, embora seja um de seus componentes.

Definição 7.3: (Prefixo livre em um predicado)

⌜Sejam p um predicado e f um prefixo em um grafo-esquema GE.

Diz-se que f é um *prefixo livre* em p se e somente se a expressão-de-conjunto f é um componente imediato de p.⌟

Os exemplos que seguem ilustram o uso de predicados de restrição e o papel dos prefixos livres nesses predicados:

a) NOME of fornecedores: (<NOME,ESTADO> of cidade =
= <'SÃO PAULO','SP'>) of ITEM

Se não houvesse o predicado de restrição, a expressão assumiria para cada nó de GBD que pertence ao nó ITEM (de GE) o valor correspondente a união dos valores do atributo NOME para os nós de GBD que pertencem ao nó FORNECEDOR e estão "ligados" ao referido nó através do nome de conexão fornecedores.

Com o predicado de restrição, o valor da expressão-de-caminho para cada nó de GBD que pertence ao nó ITEM é formado com os valores do atributo NOME dos nós de GBD que pertencem ao nó FORNECEDOR, estão "ligados" ao nó em questão através do nome de conexão fornecedores e satisfazem ao predicado de restrição.

O prefixo livre <NOME,ESTADO> of CIDADE no predicado de restrição "funciona" como uma expressão-de-caminho

associada a uma variável cujo domínio é o nó CIDADE que é calculada para cada nó de GBD para o qual o predicado é calculado.

Para os nós 31 e 32 essa expressão assume o valor {'ABDUL' & 'ABDUL'}, e para o nó 39 ela assume o valor ϕ (conjunto vazio).

Vê-se então que essa expressão-de-caminho representa para cada item o conjunto dos nomes dos fornecedores desse item que estão localizados na cidade de São Paulo (SP).

b) NOME of cidade : (ESTADO = 'SP') of fornecedores of ITEM

Nesse caso, para cada nó de GBD que pertence ao nó ITEM o predicado de restrição é testado para os nós de GBD que pertencem ao nó CIDADE e estão "ligados" ao referido nó através dos nomes de conexão cidade e fornecedores. Portanto o prefixo livre ESTADO "funciona" como uma expressão-de-caminho associada a uma variável cujo domínio é o nó CIDADE.

Para os nós 31 e 32 essa expressão assume o valor {'GUARULHOS','SÃO PAULO'} e para o nó 39 ela assume o valor {'GUARULHOS'}.

Vê-se que essa expressão-de-caminho representa para cada item o conjunto dos nomes das cidades do estado cuja sigla é SP onde estão localizados fornecedores daquele item.

8 - CONSULTAS

As consultas são os elementos da linguagem GORDAS através dos quais o usuário acessa a base de dados. Todos os elementos da linguagem apresentados nas seções anteriores têm por objetivo a construção de consultas.

Definição 8.1 (Consulta em um grafo-esquema)

⌜Uma *consulta* em um grafo-esquema GE é uma expressão da forma

get $r_1, r_2, \ldots, r_n$ (cláusula get)

⌊where p⌋ , (cláusula where)

onde $r_1, r_2, \ldots, r_n$, $n \geq 1$, são expressões-de-caminho tais que as variáveis associadas às mesmas (def. 5.3) são distintas entre si, e p é um predicado.

Os símbolos '⌊' e '⌋' são meta-símbolos que indicam que a cláusula where pode não existir.

Se n = 1, o *tipo* da consulta é o tipo da expressão-de-caminho r_1.

Se n>1 o tipo da consulta é $\langle t_1, t_2, \ldots, t_n \rangle$, onde $t_1, t_2, \ldots, t_n$ são os tipos das expressões-de-caminho $r_1, r_2, \ldots, r_n$ respectivamente.

Diz-se que as variáveis $v_1, v_2, \ldots, v_n$, associadas respectivamente às expressões-de-caminho $r_1, r_2, \ldots, r_n$, são as variáveis *declaradas* na consulta. ⌋

Essa definição difere da definição correspondente na descrição dos autores nos seguintes pontos:

(i) Foi generalizado o conceito de tipo de uma consulta, analogamente ao que foi feito na seção 5 para as expressões-de-caminho e na seção 6 para as expressões-de-constantes e para as expressões-de-conjuntos.

(ii) Foi incluída na definição a restrição de que as variáveis associadas às expressões-de-caminho da cláusula get devem ser distintas (que os autores introduziram informalmente).

(iii) Não foi definida a semântica das consultas, o que será feito na seção 11 através de uma definição específica (a parte da definição dos autores relativa à semântica das consultas é muito rudimentar como definição formal, e será apresentada nesta seção como parte da descrição informal da semântica).

Os papéis desempenhados pelos vários elementos da linguagem que formam uma consulta podem ser resumidos como segue:

- as variáveis associadas às expressões-de-caminho especificadas na cláusula get determinam os conjuntos-entidade e os conjuntos-relacionamento sobre os quais será feita a consulta;

- os prefixos dessas expressões-de-caminho determinam quais os dados que devem ser obtidos das entidades e relacionamentos dos referidos conjuntos para formar o resultado da consulta;

- o predicado especificado na cláusula where seleciona, dentre as várias combinações de entidades e relacionamentos possíveis, aquelas que devem efetivamente ser utilizadas para formar o resultado da consulta.

A definição da semântica de uma consulta depende de alguns conceitos preliminares, que serão considerados nas seções seguintes. A descrição (rudimentar) dos passos envolvidos no processo de cálculo do valor de uma consulta, que será apresentada em seguida, introduz informalmente essa semântica.

O processo de cálculo do valor de uma consulta pode ser imaginado como sendo o seguinte:

(i) É formado o conjunto $Z = Z_1 x Z_2 x \ldots x Z_n$, onde Z_i é o conjunto de nós do grafo-base-de-dados (que representa a base de dados que está sendo consultada) que pertencem ao domínio da variável v_i (ver def. 5.2) associada à expressão-de-caminho r_i da cláusula get, para $i=1,2,\ldots,n$.
(Cada elemento de Z representa uma combinação de entidades e/ou relacionamentos da base de dados).

(ii) Se existe a cláusula where na consulta, são excluídos do conjunto Z todos os elementos para os quais o predicado p (especificado na cláusula where) é falso, isto é, são excluídos os elementos $z = (x_1, x_2, \ldots, x_n) \in Z$ tais que atri

buindo-se os valores $x_1, x_2, \ldots, x_n$ respectivamente às variáveis $v_1, v_2, \ldots, v_n$ declaradas na consulta e calculando-se o predicado obtém-se o valor falso.

(iii) É construído então o valor da consulta, da seguinte maneira:

Se n=1, é calculado o valor da expressão-de-caminho r_1 para cada elemento de Z (que nesse caso é um nó do grafo-base-de-dados) e o valor da consulta é a união desses valores.

Se n>1, para cada elemento $(x_1, x_2, \ldots, x_n)$ do conjunto Z é calculada a ênupla $\langle u_1, u_2, \ldots, u_n \rangle$, cujos componentes são os valores das expressões-de-caminho $r_1, r_2, \ldots, r_n$ respectivamente para os nós $x_1, x_2, \ldots, x_n$, e o valor da consulta é o conjunto das ênuplas assim calculadas.

Exemplos de Consultas:

(Em todos os exemplos seguintes os colchetes são utilizados para delimitar as partes de cada consulta que poderiam ser omitidas sem alterar o seu resultado, o que é possível devido a uma regra adicional sobre a especificação de consultas que será apresentada posteriormente).

↓ (a) Obtenha os nomes e endereços de todos os fornecedores localizados em SP:

get <NOME,END,NOME of cidade> of FORNECEDOR

↑ where ESTADO of cidade [of FORNECEDOR] = 'SP'

Considerando a parte de uma base de dados representada na figura II-C o valor dessa consulta é {<'MORAES & FILHOS', 'R.Antonio de Camargo, 30', 'GUARULHOS'>, <'ABDUL & ABDUL', 'R. Oriente, 500', 'SÃO PAULO'>}.

↓ (b) Para cada cidade, obtenha o nome da cidade e os nomes de todos os fornecedores e clientes localizados na cidade:

get <NOME, <NOME of fornecedores, NOME of clientes>>

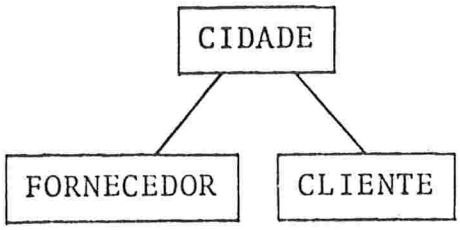
↑ of CIDADE

Considerando a figura II-C o valor dessa consulta é
{<'GUARULHOS', <'MORAES & FILHOS', ϕ>> ,
<'SÃO PAULO', <'ABDUL & ABDUL', {'ESTUDANTE FELIZ','PAPELARIA FERROSO'}>>,
<'PIRIPIRI', <'CASA SÃO JORGE', ϕ>>}

(Cabe recordar que todos os valores componentes das ênuplas são conjuntos, mas por simplicidade os conjuntos unitários estão denotados por seus elementos).

↓ Este exemplo mostra que as expressões-de-caminho permitem uma visão hierárquica da base de dados, na qual o domínio da variável associada à expressão é a raiz da hierarquia.

A hierarquia correspondente à expressão-de-caminho utilizada neste exemplo é



Esta característica da linguagem permite obter um agrupamento natural dos valores que formam o resultado da consulta.

(c) Obtenha os pares de nomes de clientes e de fornecedores localizados na mesma cidade tais que o fornecedor possui em estoque todos os itens requeridos pelo cliente:

```
get NOME of CLIENTE, NOME of FORNECEDOR
where (cidade of CLIENTE = cidade of FORNECEDOR)
  and (itens-em-estoque of FORNECEDOR includes
       itens-pedidos of CLIENTE)
```

Considerando a figura II-C o valor dessa consulta é
{<'PAPELARIA FERROSO', 'ABDUL & ABDUL'>}

(d) Para cada cliente obtenha o nome do cliente e os nomes dos fornecedores localizados na mesma cidade e que possuem em em estoque todos os itens requeridos pelo cliente:

```
get <NOME,NOME of fornecedores: (itens-em-estoque
        includes itens-pedidos of CLIENTE)
        of cidade>  of CLIENTE
```

Esta consulta obtém os mesmos dados que a consulta anterior, porém agrupados de forma diferente (infelizmente para a figura II-C o resultado é o mesmo!).

↓ (e) Obtenha os nomes de todos os fornecedores que estão localizados na cidade do cliente ESTUDANTE FELIZ e que possuem todos os itens por ele requeridos em estoque:

```
get NOME of FORNECEDOR
 where  (cidade [of FORNECEDOR]  =
          (get cidade of CLIENTE
            where NOME [of CLIENTE] = 'ESTUDANTE FELIZ'))
 and (itens-em-estoque  of FORNECEDOR  includes
          (get itens-pedidos of CLIENTE
            where NOME [of CLIENTE] ='ESTUDANTE FELIZ'))
```

Este exemplo ilustra o uso de consultas como expressões-de-conjuntos, para formar outras consultas.

É interessante observar que a variável definida por

ambas as consultas internas é a mesma, o que não traz proble-
↑ mas porque elas são independentes.

Para a parte de uma base de dados representada na figura II-C o valor dessa consulta é o conjunto vazio.

Os mesmos resultados podem ser obtidos através da seguinte consulta:

get NOME of fornecedores: (itens-em-estoque includes
itens-pedidos of CLIENTE) of cidade of CLIENTE
where NOME [of CLIENTE] = 'ESTUDANTE FELIZ'.

É razoável esperar que essa consulta seja mais eficiente que a anterior, independentemente da implementação da linguagem. Isto porque naquele caso para cada fornecedor são calculadas duas consultas (as consultas internas), que testam o predicado NOME of CLIENTE = 'ESTUDANTE FELIZ' para todos os clientes. Mesmo que a implementação seja tal que o valor das consultas internas seja calculado uma única vez (já que elas não dependem da variável declarada na consulta externa) ainda terão que ser realizados três cálculos de consulta.

↓ (f) Obtenha os pares de nomes de clientes e fornecedores tais que o fornecedor possui em estoque todos os itens requeridos pelo cliente, em quantidade maiores que aquelas por
↑ ele requeridas:

```
get  NOME of CLIENTE, NOME of FORNECEDOR

 where  (get ITEM

        where (fornecedores [of ITEM] includes FORNECEDOR)

         and (clientes [of ITEM] includes CLIENTE)

         and (QTD of fornecedores:(NOME = NOME of

                 FORNECEDOR) [of ITEM] >

              QTD of clientes:(NOME = NOME of

                 CLIENTE) [of ITEM]))

        = itens-pedidos of CLIENTE
```

Essa consulta é bastante complexa, e ilustra a "potência" da linguagem.

Para cada par (cliente, fornecedor) a consulta interna forma o conjunto dos itens que são fornecidos pelo fornecedor, requeridos pelo cliente e cuja quantidade em estoque do fornecedor é maior que a quantidade requerida pelo cliente. Nessa consulta o predicado fornecedores of ITEM includes FORNECEDOR testa se o item é fornecido pelo fornecedor; o predicado clientes of ITEM includes CLIENTE testa se o item é requerido pelo cliente e o predicado QTD of fornecedores:(NOME=NOME of FORNECEDOR) of ITEM > QTD of clientes:(NOME=NOME of CLIENTE) of ITEM testa se a quantidade em estoque do fornecedor é maior que a quantidade requerida pelo cliente. Cabe observar que esse

último predicado somente é testado se os dois primeiros são verdadeiros, o que garante que as expressões-de-caminho que são comparadas pelo operador > não resultam no conjunto vazio; além disso, os predicados de restrição inseridos nessas expressões garantem que as mesmas não resultam em conjuntos com mais de um elemento, e que portanto a comparação não resulta em erro (ver regras 6B e 6D na seção 6). Esses predicados de restrição são necessários porque um item pode ser fornecido por outros fornecedores e requerido por outros clientes além do par (cliente, fornecedor) específico que está sendo testado em cada cálculo da consulta interna. É interessante notar que o prefixo livre NOME no primeiro predicado de restrição se refere ao nome de cada fornecedor do item que está sendo testado, enquanto que no segundo o mesmo prefixo se refere ao nome de cada cliente que requer aquele item.

O predicado da consulta externa apenas testa (para cada par (cliente, fornecedor)) se os itens que o fornecedor possui em estoque em quantidade maior que a requerida pelo cliente (resultado da consulta interna) correspondem exatamente aos itens requeridos pelo cliente.

Considerando a figura II.C, o valor da consulta é

{<'ESTUDANTE FELIZ','MORAES & FILHOS'>

<'PAPELARIA FERROSO', 'MORAES & FILHOS'>,

<'PAPELARIA FERROSO', 'CASA SÃO JORGE'>}.

Pode-se obter consultas equivalentes à consulta apresentada através da substituição de alguns predicados por predica

dos equivalentes, como por exemplo:

```
get NOME of CLIENTE, NOME of FORNECEDOR
 where (get ITEM
        where (itens-em-estoque of FORNECEDOR includes ITEM)
        and   (itens-pedidos of CLIENTE includes ITEM)
        and   (QTD of itens-em-estoque:(CÓDIGO = CÓDIGO
               of ITEM) of FORNECEDOR >
               QTD of itens-pedidos: (CÓDIGO = CÓDIGO
               of ITEM) of CLIENTE))
        = itens-pedidos of CLIENTE
```

(g) Obtenha os pares de nomes de fornecedores tais que os itens em estoque de ambos sejam exatamente os mesmos:

```
get NOME of FORNECEDOR.1, NOME of FORNECEDOR.2
 where FORNECEDOR.1 ≠ FORNECEDOR.2
  and  itens-em-estoque of FORNECEDOR.1
       = itens-em-estoque of FORNECEDOR.2
```

Considerando a figura II.C, o resultado dessa consulta é {<'MORAES & FILHOS','CASA SÃO JORGE'>, <'CASA SÃO JORGE' , 'MORAES & FILHOS'>}.

Para não obter dois pares de nomes (em ordem inversa) para cada par de fornecedores, pode ser utilizada a seguinte consulta:

```
get NOME of FORNECEDOR.1, NOME of FORNECEDOR.2
 where (NOME of FORNECEDOR.1
        < NOME of FORNECEDOR.2)
 and  (itens-em-estoque of FORNECEDOR.1
        = itens-em-estoque of FORNECEDOR.2)
```

A possibilidade de omissão da variável em certas expressões-de-caminho, que foi mostrada nos exemplos através de colchetes, é definida pela seguinte regra:

Regra 8.A

⌜Se q é uma consulta em um grafo-esquema GE tal que q é da forma

```
get r
  where p
```

(a cláusula get contém uma única expressão-de-caminho), então qualquer expressão-de-caminho r' que é um componente imediato de p (ver def. 7.2) tal que r' = f of v, onde f é um prefixo e v a variável associada à expressão-de-caminho r, pode ser denotada simplesmente pelo prefixo f.⌟

Alguns dos exemplos apresentados mostram a utilização de consultas "internas", especificadas como partes de outras consultas (exemplos (e), (f)).

É interessante observar que o fato de uma consulta ser ou não uma consulta interna não é uma característica da consulta em si. Por exemplo, na consulta

get NOME of ITEM

where NOME of fornecedores of ITEM

= (get NOME of FORNECEDOR)

(essa consulta obtém os nomes dos itens fornecidos por todos os fornecedores) é utilizada a consulta get NOME of FORNECEDOR , que também pode ser utilizada como uma consulta "independente" (essa consulta obtém os nomes de todos os fornecedores). Por sua vez, a primeira consulta, que foi apresentada como uma consulta independente, também pode ser utilizada para formar outras consultas, tornando-se assim uma consulta interna.

Portanto, os conceitos de "consulta independente" e de "consulta interna" são conceitos relativos.

Uma característica importante da linguagem GORDAS é que nas expressões-de-caminho de uma consulta interna podem ser utilizadas variáveis declaradas em qualquer consulta da qual ela é um componente (ver exemplo (f)). Fazendo uma analogia com as linguagens de programação tipo ALGOL pode-se dizer que as variáveis declaradas em uma consulta são variáveis globais a todas as consultas internas que dela fazem parte.

9 - CONSULTAS BEM FORMADAS

As definições até aqui apresentadas permitem especificar consultas inconsistentes, cujos significados não estão definidos.

Por exemplo, as consultas

get ESTADO of ITEM

e get NOME of CLIENTE

where count (fornecedores of ITEM)>0

são inconsistentes. No primeiro caso a expressão-de-caminho ESTADO of ITEM é inconsistente, já que ESTADO não é um atributo de ITEM; no segundo caso, embora os componentes da consulta sejam válidos, há uma incompatibilidade entre eles, pois o predicado se refere à variável ITEM enquanto que a variável declarada na consulta é CLIENTE.

Esses exemplos ilustram os dois principais motivos pelos quais uma consulta pode ser inconsistente: por envolver expressões-de-caminho inconsistentes ou por conter componentes que especificam variáveis que são incompatíveis com as variáveis declaradas na consulta. O problema específico das expressões-de-caminho inconsistentes será tratado na seção seguinte.

Para tratar do problema das consultas inconsistentes em geral, que não é considerado na descrição dos autores, intro-

duziu-se na linguagem o conceito de consulta bem formada , de forma que as consultas definidas como sendo bem formadas são exatamente as consultas consistentes, às quais será formalmente associado um significado na seção 11.

Um dos aspectos que devem ser considerados na caracterização das consultas bem formadas é a compatibilidade entre as variáveis nelas declaradas e as variáveis utilizadas em seus componentes, o que é feito através do conceito de contexto.

Definição 9.1 (contexto em um grafo-esquema)

⌜Um *contexto* em um grafo-esquema GE é um conjunto de variáveis em GE.⌟

Por exemplo, se a consulta

get NOME of CLIENTE, NOME of ITEM

where ...

é especificada como uma consulta independente, as variáveis que podem ser utilizadas no predicado da cláusula where são identificadas através do contexto {CLIENTE,ITEM}.

A própria definição de consulta bem formada é feita em função de um contexto, isto é, a definição caracteriza uma consulta bem formada *em relação a um certo contexto*. Isto é necessário porque a definição é utilizada recursivamente para tra

tar as consultas internas à consulta a qual ela se refere, e é preciso "relacionar" as variáveis declaradas nessa consulta com as referidas consultas internas.

Para efeito de verificação dessa definição para uma consulta independente deve ser utilizado o contexto $K=\phi$ (conjunto vazio).

Definição 9.2 (consulta bem formada)

$$\left\lceil \text{Sejam } q=\underline{\text{get}}\ r_1,r_2,\ldots,r_n \left\lfloor \underline{\text{where}}\ p \right\rfloor \right.$$

uma consulta e K um contexto, em um grafo-esquema GE.

Seja K' o conjunto formado pela união de K e do conjunto das variáveis declaradas em q.

Diz-se que a consulta q é *bem formada* em relação ao contexto K se e somente se as seguintes condições são satisfeitas:

1) Nenhuma variável declarada em q ocorre no contexto K.

2) As expressões-de-caminho $r_1,r_2,\ldots,r_n$ são bem especificadas (def. 10.3) em relação ao contexto K'.

3) Toda expressão-de-caminho que é um componente imediato de p (def. 7.2) é bem especificada em relação ao contexto K'.

4) Se existe um prefixo livre em p (def. 7.3) as seguintes condições são satisfeitas:

   (i) n=1

   (ii) Se f é um prefixo livre em p então a expressão-de-caminho f of v é bem especificada em relação ao contexto K', onde v é a variável declarada em q.

5) Toda consulta q' tal que a expressão-de-conjunto (q') ou count (q') é um componente imediato de p é bem formada em relação ao contexto K'. ⌋

Deve-se observar que se uma consulta é bem formada então todas as consultas internas à mesma são bem formadas. Isto está explícito na definição 9.2. Apenas para as consultas internas que são "componentes imediatos" do predicado da referida consulta (item 5), mas está implícito para as demais consultas nos itens 2, 3 e 4, pois as mesmas devem ser componentes das expressões-de-caminho a que se referem esses itens, e estas por sua vez são bem especificadas (def. 10.3) se e somente se as consultas nelas contidas são bem formadas (as definições 9.2 e 10.3 são mutuamente recursivas).

10 - EXPRESSÕES-DE-CAMINHO BEM ESPECIFICADAS

A definição de expressão-de-caminho apresentada na seção 5 permite a construção de expressões-de-caminho inconsistentes, cujo significado não está definido.

Uma expressão-de-caminho pode ser inconsistente pelas seguintes razões:

(i) por não definir um "percurso" coerente no grafo-esquema ao qual se refere;

(ii) por envolver um predicado de restrição que por sua vez contem elementos inconsistentes.

Por exemplo, a expressão-de-caminho PREÇO of cidade of ITEM (relativa ao grafo-esquema da figura II-B) é inconsistente, pois não há no grafo-esquema uma aresta ligada ao nó ITEM cujo rótulo inclua o nome de conexão cidade. Além disso, nenhuma das arestas que incluem esse nome de conexão em seu rótulo estão ligadas diretamente ao nó ESTOQUE, que é o único nó do grafo-esquema ao qual está associado o atributo PREÇO. Toda expressão-de-caminho que utiliza um predicado do qual a expressão-de-caminho PREÇO of cidade of ITEM é um componente é também inconsistente.

Por esse motivo foi introduzido na linguagem o conceito de expressão-de-caminho bem especificada ("well specified path expression"), de forma que as expressões-de-caminho defini

das como sendo bem especificadas são exatamente aquelas que são consistentes, às quais será formalmente atribuído um significado na seção 12.

A definição apresentada nesta seção difere da definição correspondente na descrição dos autores principalmente nos seguintes pontos:

(i) são tratados todos os tipos de expressões-de-caminho, e não apenas aquelas cujos prefixos são sequências de nomes de conexão como na definição dos autores;

(ii) é considerada a "consistência" de todos os componentes dos predicados de restrição eventualmente especificados no prefixo da expressão-de-caminho;

(iii) é considerada a existência das variáveis referenciadas pela expressão-de-caminho e seus componentes no contexto associado à expressão pela definição 9.2, completando assim aquela definição.

Os principais conceitos envolvidos nessa definição estão relacionados com a coerência do "percurso" sobre o grafo-esquema que é especificado pelo prefixo da expressão-de-caminho. Esses conceitos são tratados na definição de "percurso coerente", na qual se baseia a definição de expressão-de-caminho bem especificada.

Para tratar esses "percursos" sobre o grafo-esquema é necessário inicialmente formalizar algumas relações existentes

entre um nó de um grafo-esquema, as arestas ligadas a esse nó e os nomes de conexão especificados nos rótulos dessas arestas.

Definição 10.1 (nome de conexão com origem em um nó; aresta apontada por um nome de conexão)

⌜Seja X um nó de um grafo-esquema GE. Sejam $d_1, d_2, \ldots, d_n$ as arestas ligadas nesse nó e $(c_{11}, c_{12}), (c_{21}, c_{22}), \ldots, (c_{n1}, c_{n2})$ as partes de nome (def. 2.2) dos rótulos dessas arestas (respectivamente). Sejam $c_1, c_2, \ldots, c_n$ nomes de conexão tais que:

(i) se X é um CE-nó (nesse caso X é o extremo inicial (ver /Lucc77/) das arestas $d_1, d_2, \ldots, d_n$) então $c_i = c_{i1}$, $i=1,2,\ldots,n$;

(ii) se X é um CR-nó (nesse caso X é o extremo final das arestas $d_1, d_2, \ldots, d_n$) então $c_i = c_{i2}$, $i=1,2,\ldots,n$.

Diz-se que $c_1, c_2, \ldots, c_n$ são os nomes de conexão *com origem em* X, e que a aresta d_i é *apontada* pelo nome de conexão c_i a partir de X, para $i=1,2,\ldots,n$.⌟

É interessante observar que as restrições impostas sobre os rótulos de um grafo-esquema pela definição 2.3 implicam que, para qualquer nó X de um grafo-esquema, se c é um nome de conexão com origem em X então existe uma única aresta apontada por c a partir de X.

Definição 10.2 (percurso coerente)

Sejam f um prefixo, K um contexto e (X,X') um par de nós de um grafo-esquema GE. Diz-se que f define um *percurso coerente* com o par (X,X') e com o contexto K se e somente se uma das seguintes condições é satisfeita:

1) f é o nome de um atributo associado ao rótulo do nó X ou do nó X'.

2) f é um nome de conexão com origem em X.

3) f é da forma f' of c , onde f' é um prefixo e c um nome de conexão, e as seguintes condições são satisfeitas:

   (i) c é um nome de conexão com origem em X

   (ii) o prefixo f' define um percurso coerente com o par (Y,Y') e com o contexto K, onde Y' é o nó ligado a X pela aresta apontada pelo nome de conexão c a partir de X, e Y é definido como segue:
   Se Y' é um CE-nó (representa um conjunto-entidade) ou é um CR-nó ligado a mais de duas arestas (representa um conjunto-relacionamento de grau n>2), então Y=Y'.
   Se Y' é um CR-nó ligado a exatamente duas arestas (representa um conjunto-relacionamento binário), então Y é o nó ligado a Y' pela aresta apontada pelo nome de conexão c a partir de Y'.

4) f é forma $<f_1,f_2,\ldots,f_n>$ e f_i é um prefixo que define um per

curso coerente com o par (X,X') e com o contexto K, para i=1,2,...,n.

5) (f envolve um predicado de restrição)

5.1 - f é da forma c:(p) , onde c é um nome de conexão e p um predicado, e as seguintes condições são satisfeitas:

(i) c é um nome de conexão com origem em X.

(ii) Existe um prefixo f_ℓ tal que f_ℓ é um prefixo livre em p (def. 7-3).

(iii) Se f_ℓ é um prefixo livre em p, então f_ℓ define um percurso coerente com o par (Y,Y') e com o contexto K, onde Y e Y' são definidos em função de X e c como no item 3.

(iv) Toda expressão-de-caminho que é um componente imediato de p (def. 7-2) é bem especificada em relação ao contexto K (def. 10.3).

(v) Toda consulta q tal que a expressão-de-conjunto (q) ou count (q) é um componente imediato de p é bem formada em relação ao contexto K (def. 9.2).

5.2 - f é da forma f' of c:(p) , onde f' é um prefixo, c um nome de conexão e p um predicado, e as seguintes condições são satisfeitas:

(i) Todas as condições especificadas no item 5.1 são satisfeitas.

(ii) O prefixo f' define um percurso coerente com par (Y,Y') e com o contexto K, onde Y e Y' são definidos em função de X e c como no item 3.

6) f é da forma count f' ou <count,f'> , e f' é um prefixo que define um percurso coerente com o par (X,X') e com o contexto K. ⌟

Nota: Os números dos itens nessa definição identificam os itens na definição de prefixo (def. 5.1) onde são definidos os prefixos tratados em cada um dos referidos itens.

Essa definição pode ser interpretada como um algoritmo recursivo que "percorre" o grafo-esquema ao mesmo tempo que analisa o prefixo, e em cada passo verifica se o "trecho" do grafo que está sendo percorrido é coerente com a parte do prefixo que está sendo analisada. Quando a parte do prefixo a ser analisada é uma ênupla, são iniciados vários "percursos" em paralelo (um para cada prefixo que compõe a ênupla), todos com origem no mesmo "trecho" do grafo-esquema. Os "trechos" do grafo-esquema considerados nessa definição são pares (X,X') de nós tais que X=X' ou X representa um conjunto-entidade e X' representa um conjunto-relacionamento binário que envolve o conjunto-entidade representado por X.

A verificação da definição 10.2 para um prefixo es

pecífico apresentada a seguir ilustra esse algoritmo.

Proposição:

"No grafo-esquema da figura II.b, o prefixo

<NOME,NOME of cidade, QTD> of fornecedores

define um percurso coerente com o par (ITEM,ITEM) e com o contexto K={ITEM}" (os nós estão sendo denotados pelos nomes associados aos seus rótulos).

Prova:

⌜Pelo item 3 (def. 10.2) essa proposição é verdadeira sse fornecedores é um nome de conexão com origem em ITEM (o que de fato ocorre, de acordo com a definição 10.1) e o prefixo <NOME,NOME of cidade, QTD> define um percurso coerente com o par (FORNECEDOR,ESTOQUE) e com o contexto K.

Segue então pelo item 4 que a proposição é verdadeira sse os prefixos NOME,NOME of cidade e QTD definem um percurso coerente com o par (FORNECEDOR,ESTOQUE) e com o contexto K.

Pelo item 1, essa afirmação é verdadeira para os prefixos NOME e QTD, e segue então pelo item 3 que a proposição é verdadeira sse cidade é um nome de conexão com origem em FORNECEDOR (o que de fato ocorre) e o prefixo NOME define um percurso coerente com o par (CIDADE,FORN-CID) e com o contexto K.

Essa última afirmação é verdadeira de acordo com o item 1, e conclui-se então que a proposição é verdadeira. ⌋

Com base na definição 10.2 pode-se então concluir esta seção com a definição que sintetiza todos os conceitos aqui tratados:

Definição 10.3 (expressão-de-caminho bem especificada)

⌈ Sejam r uma expressão-de-caminho e K um contexto, em um grafo-esquema GE.

Diz-se que r é *bem especificada* em relação ao contexto K se e somente se a variável v associada a r (def. 5.3) existe no contexto K e o prefixo de r (se houver) define um percurso coerente com o par (X,X) e com o contexto K, onde X é o domínio da variável v. ⌋

11 - VALOR DE UMA CONSULTA

O objetivo desta seção é definir formalmente o processo de cálculo do valor de uma consulta, que foi introduzido informalmente na seção 8 (essa definição formal não existe na descrição dos autores).

Naturalmente o cálculo do valor de uma consulta envolve o cálculo dos valores das expressões-de-caminho nela utilizadas. Essa questão específica do cálculo do valor de uma expressão-de-caminho será tratado na seção seguinte.

O cálculo do valor de uma consulta que contém outras consultas depende obviamente do cálculo dos valores das mesmas, e por esse motivo o processo de cálculo aqui apresentado é recursivo. Como o valor dessas consultas internas pode depender dos valores de variáveis declaradas na consulta "externa", é necessário definir os valores dessas variáveis para cada "acionamento recursivo" do processo, o que é feito através do conceito de "estado".

Definição 11.1 (Estado)

Sejam GE um grafo-esquema e GBD um grafo-base-de-dados que correspode a GE.

Um *estado* relativo ao par (GE, GBD) é um conjunto $\{w_1=y_1, w_2=y_2, \ldots, w_m=y_m\}$, $m \geq 0$, onde $w_1, w_2, \ldots, w_m$ são variá-

veis distintas em GE e $y_1, y_2, \ldots, y_m$ são nós de GBD que pertencem (def. 2.6) respectivamente aos domínios (def. 5.2) de $w_1, w_2, \ldots, w_m$.⌋

Por exemplo, no cálculo do valor de uma consulta da forma

get NOME of CLIENTE, NOME of ITEM

where (get ...) = ... ,

é necessário calcular o valor da consulta interna para cada par de valores (x,y) das variáveis CLIENTE e ITEM. Cada um desses cálculos da consulta interna é especificado como o "cálculo do valor da consulta get ... para o estado {CLIENTE=x, ITEM=y} ".

O cálculo do valor de uma consulta independente deve ser iniciado com o estado E=ϕ (conjunto vazio). O próprio processo de cálculo define os estados necessários para o cálculo do valor das consultas internas.

Definição 11.2 (valor de uma consulta)

⌈ Sejam GE um grafo-esquema e GBD um grafo-base-de-dados que corresponde a GE.

Sejam q = get $r_1, r_2, \ldots, r_n$

⌊ where p ⌋

uma consulta em GE e E= $\{w_1=y_1, w_2=y_2,\ldots,w_m=y_m\}$ um estado relativo ao par (GE, GBD) tais que a consulta q é bem formada em relação ao contexto $\{w_1,w_2,\ldots,w_m\}$. Sejam $v_1,v_2,\ldots,v_n$ as variáveis associadas (def. 5.2) respectivamente às expressões-de-caminho $r_1,r_2,\ldots,r_n$.

O *valor* da consulta q para o par (GBD, E) é o valor V obtido através do seguinte processo:

1) É tomado como valor inicial o valor V= ϕ (conjunto vazio).

2) É construído o conjunto $Z = Z_1 x\ Z_2\ x \ldots x\ Z_n$, onde Z_i é o conjunto dos nós de GBD que pertencem ao domínio da variável v_i, para i=1,2,...,n (Z_i é o conjunto dos valores que a variável v_i pode assumir).

3) Para cada elemento $z = (x_1,x_2,\ldots,x_n) \in Z$:

   3.1 - É construído o estado

   $$E' = \{w_1=y_1,w_2=y_2,\ldots,w_m=y_m,v_1=x_1,v_2=x_2,\ldots,v_n=x_n\} .$$

   3.2 - Se existe a claúsula where, o predicado p é calculado a partir dos seguintes valores:

   (i) para toda expressão-de-caminho r que é um componente imediato de p (def. 7.2) é tomado o valor de r para o par (GBD, E') (def. 12.4);

   (ii) para todo prefixo f tal que f é um prefixo li-

vre em p (def. 7.3), é tomado o valor da expressão-de-caminho f of v_1 para o par (GBD,E');

(iii) para toda consulta q' tal que a expressão-de-conjunto (q') ou count(q') é um componente imediato de p é tomado o valor de q' para o par (GBD, E').

3.3 - Se não existe a cláusula where, ou se ela existe e o valor do predicado p é verdadeiro:

3.3.1 - Se n=1, o valor V é substituído por (V ∪ u), onde u é o valor da expressão-de-caminho r_1 para o par (GBD, E').

3.3.2 - Se n > 1, o valor V é substituído por

(V ∪ {< u_1,u_2,...,u_n>}), onde u_1,u_2,...,u_n são respectivamente os valores das expressões-de-caminho r_1,r_2,...,r_n para o par (GBD, E'). (a ênupla de valores <u_1,u_2,...,u_n> torna-se um elemento do valor V). ⌟

12 - VALOR DE UMA EXPRESSÃO-DE-CAMINHO

O objetivo desta seção é definir formalmente a semântica das expressões-de-caminho, que foi introduzida informalmente na seção 5.

As expressões-de-caminho são utilizadas para formar consultas, e sua função nas mesmas é obter os valores da base de dados que são necessários para construir os valores dessas consultas.

O valor que é obtido da base de dados por uma expressão-de-caminho depende do valor da variável associada à mesma, e, se ela envolver predicados de restrição, depende também dos valores das variáveis associadas às expressões-de-caminho especificadas nesses predicados. (Cabe recordar que o valor de uma variável é um nó de grafo-base-de-dados que representa a base de dados que está sendo consultada, e portanto representa uma entidade ou um relacionamento dessa base de dados).

Para definir formalmente os valores das variáveis que determinam cada valor específico de uma expressão-de-caminho será utilizado o conceito de estado (def. 11.1). Por exemplo, cada particular valor da expressão-de-caminho r = QTD of clientes : (NOME = NOME of CLIENTE) of ITEM pode ser referido como "valor de r para o estado {ITEM=x, CLIENTE=y}".

As definições que serão apresentadas nesta seção po

dem ser interpretadas como elementos de um algoritmo recursivo que determina o valor de uma expressão-de-caminho para um particular grafo-base-de-dados e um particular estado. Esse algoritmo percorre o grafo-base-de-dados ao mesmo tempo em "decompõe" o prefixo em "subprefixos", construindo conjuntos de nós chamados "bases" (def. 12.1). Cada "iteração" do algoritmo consiste em "aplicar" um subprefixo a uma base.

O algoritmo tem início (def. 12.4) com o prefixo da expressão-de-caminho e com a base formada pelo nó que é o valor da variável associada à mesma.

Quando numa iteração o subprefixo a ser aplicado à base é da forma f of c , onde f é outro subprefixo e c um nome de conexão, é gerada uma nova base com os nós "ligados" aos nós da base anterior pelo nome de conexão c (def. 12.2), e a próxima iteração consiste em aplicar o subprefixo f a essa nova base.

Esse processo de decomposição do prefixo em subprefixos prossegue até que sejam obtidos "subprefixos elementares"(um nome de atributo, um nome de conexão, o operador count), cuja aplicação à base gera então os "subvalores" que constituem o valor da expressão-de-caminho.

Após as definições formais será apresentado um exemplo de cálculo do valor de uma expressão-de-caminho que ilustra esse algoritmo.

Cabe ressaltar que o conceito de base, o qual foi introduzido como um "conjunto de nós", de fato envolve um conjunto

de pares de nós. A necessidade de se "guardar" pares de nós durante o processo de cálculo se tornará evidente com a apresentação das definições.

Definição 12.1 (base)

⌜Sejam GE um grafo-esquema e GBD um grafo-base-de-dados que corresponde a GE.

Uma *base* relativa ao par (GE,GBD) é uma tripla (X,X', H), onde X e X' são nós de GE e H é um conjunto de pares de nós de GBD tais que uma das seguintes afirmações é verdadeira:

(i) X=X', e todo par em H é da forma (x,x), onde x pertence (def. 2.6) a X;

(ii) X é um CE-nó, X' um CR-nó ligado a exatamente duas arestas, existe uma aresta D em GE que liga X a X', e todo par em H é da forma (x,x'), onde x pertence a X e está ligado a x' por uma aresta que pertence (def. 2.6) a D. (Observar que pela def. 2.6 o fato da aresta que liga x a x' pertencer a D implica que x' pertence a X')

⌟

Nota: É interessante observar que pode haver mais de uma aresta ligando um CE-nó X a um CR-nó X' em um grafo-esquema, o que indica que o conjunto-entidade representado por X participa mais de uma vez do conjunto-relacionamento representado por X', mas a definição apresentada se refere à uma

aresta específica D.

Definição 12.2 (base gerada por um nome de conexão)

Sejam GE um grafo-esquema e GBD um grafo-base-de-dados que corresponde a GE.

Sejam b = (X,X',H) uma base relativa ao par (GE,GBD) e c um nome de conexão com origem em X.

A base *gerada* por c a partir de b é a base $\bar{b}$ = (Y,Y', $\bar{H}$), relativa ao par (GE,GBD), onde Y' é o nó ligado a X pela aresta D que é apontada pelo nome de conexão c a partir de X (def. 10.1), e Y' e $\bar{H}$ são definidos como segue:

Seja Z o conjunto de nós de GBD (que pertencem a Y') tal que y' $\in$ Z se e somente se existe um par (x,x') em H tal que y' está ligado a x por uma aresta que pertence a D.

Se Y' é um CE-nó (representa um conjunto-entidade)ou é um CR-nó ligado a mais de duas arestas (representa um conjunto-relacionamento de grau n>2), então Y= Y' e $\bar{H}$ é o conjunto de todos os pares (y',y') tais que y' $\in$ Z.

Se Y' é um CR-nó ligado a exatamente duas arestas(representa um conjunto-relacionamento binário), então Y é o nó ligado a Y' pela aresta D' que é apontada pelo nome de conexão c a

partir de Y', e $\bar{H}$ é o conjunto de pares de nós de GBD (nos quais o primeiro componente pertence a Y e o segundo partence a Y') tal que $(y,y') \in \bar{H}$ se e somente se $y' \in Z$ e y está ligado a y' por uma aresta que pertence a D'. ⌟

Exemplos:

(a) A base $\bar{b}$ gerada pelo nome de conexão fornecedores a partir da base

b = (ESTOQUE, ESTOQUE, {(N211,N211)}) é

$\bar{b}$ = (FORNECEDOR,FORNECEDOR, {(N12,N12)}).

(b) A base $\bar{b}$ gerada pelo nome de conexão itens-pedidos a partir da base

b = (CLIENTE,CLI-CID, {(N51,N81),(N52,N82)}) é

$\bar{b}$ = (ITEM,PEDIDO,{(N31,N41),(N31,N420),(N32,N42),
(N32,N421),(N39,N49)}).

(c) A base $\bar{b}$ gerada pelo nome de conexão cidade a partir da base

b = (FORNECEDOR,ESTOQUE,{(N11,N21),(N11,N22),(N11,N29)}) é

$\bar{b}$ = (CIDADE,CLI-CID,{(N71,N61)}).

Definição 12.3 (valor coletado por um prefixo)

Sejam GE um grafo-esquema e GBD um grafo-base-de-dados que corresponde a GE.

Sejam f um prefixo em GE, $E=\{w_1=y_1, w_2=y_2, \ldots, w_m=y_m\}$ um estado relativo ao par (GE,GBD) e b = (X,X',H) uma base relativa ao par (GE,GBD) tais que f define um percurso coerente (def. 10.2) com o par (X,X') e com o contexto $\{w_1, w_2, \ldots, w_m\}$.

O valor *coletado* em GBD pelo prefixo f a partir de b e E, daqui em diante denotado por f*(GBD,b,E), é definido recursivamente como segue:

1) Se f é um nome de atributo A:
Seja V o conjunto de valores tal que u ∈ V se e somente se existe um par (x,x') em H tal que a lista de valores de atributos do rótulo de x (def. 2.5) ou a lista de valores de atributos do rótulo de x' inclui o termo A=u (u é o valor do atributo A para o nó x ou para o nó x'). O valor f*(GBD,b,E) é a união de todos os valores que formam o conjunto V.

2) Se f é um nome de conexão c:
Seja $\bar{b} = (Y,Y',\bar{H})$ a base gerada pelo nome de conexão c a partir de b.
O valor f*(GBD,b,E) é o conjunto de nós de GBD (que pertencem a Y) tal que y ∈ f*(GBD,b,E) se e somente se existe um nó y' em GBD (que pertence a Y') tal que $(y,y') \in \bar{H}$.

3) Se f é da forma f' of c , onde f' é um prefixo e c um nome de conexão, então $f^*(GBD,b,E) = f'^*(GBD,\bar{b},E)$, onde $\bar{b}$ é a base gerada pelo nome de conexão c a partir de b.

4) Se f é da forma $\langle f_1,f_2,\ldots,f_n\rangle$, onde $f_1,f_2,\ldots,f_n$ são prefixos, então $f^*(GBD,b,E)$ é o conjunto de ênuplas de valores tal que $\langle u_1,u_2,\ldots,u_n\rangle \in f^*(GBD,b,E)$ se e somente se existe um par (x,x') em H tal que $u_i = f_i^*(GBD,(X,X',\{(x,x')\}),E)$, para $i=1,2,\ldots,n$.

5) (f envolve um predicado de restrição)

5.1 - Se f é da forma c:(p) , onde c é um nome de conexão e p um predicado:

Seja $b_1=(Y,Y',H_1)$ a base gerada pelo nome de conexão c a partir de b.

Seja $b_2=(Y,Y',H_2)$ a base tal que um par $(x,x') \in H_2$ se e somente se $(x,x') \in H_1$ e o predicado p resulta no valor verdadeiro quando é calculado a partir dos seguintes valores:

(i) para toda expressão-de-caminho r que é um componente imediato de p é tomado o valor de r (def. 12.4) para o par (GBD,E);

(ii) para todo prefixo f_ℓ tal que f_ℓ é um prefixo livre em p é tomado o valor $f_\ell^*(GBD,(Y,Y',\{(x,x')\}),E)$;

(iii) para toda consulta q tal que a expressão-de-conjunto (q) ou count(q) é um componente imediato de p é tomado o valor de q para o par (GBD,E).

O valor f*(GBD,b,E) é o conjunto de nós de GBD (que pertencem a Y) tal que y ∈ f*(GBD,b,E) se e somente se existe um nó y' em GBD (que pertence a Y') tal que (y, y') ∈ H_2.

5.2 - Se f é da forma f' of c:(p) , onde f' é um prefixo , c um nome de conexão e p um predicado, então f*(GBD,b,E) = f'*(GBD,b_2,E), onde b_2 é a base construída a partir de b, c e p conforme descrito em 5.1.

6) (f envolve o operador count)

6.1 - Se f é da forma count f' , onde f' é um prefixo tal que não existem um prefixo f'' e um nome de conexão c tais que f' = f'' of c , então f*(GBD,b,E) = n, onde n é a cardinalidade (número de elementos) do valor f'*(GBD,b,E).

6.2 - Se f é da forma <count,f'>, onde f' é um prefixo:
Seja Z o conjunto de nós de GBD (que pertencem a X) tal que x ∈ Z se e somente se existe um nó x' em GBD (que pertence a X') tal que (x,x') ∈ H.
O valor f*(GBD,b,E) é a ênupla de valores <n,f'*(GBD , b,E)>, onde n é a cardinalidade do conjunto Z. ⌟

Nota: Com relação ao caso tratado no item 1, em que o prefixo em questão é um nome de atributo A, cabe observar que a hipótese de que o prefixo A define um percurso coerente com o par (X,X') garante que A é um atributo associado ao rótulo do nó X ou do nó X', e portanto que para qualquer par de nós (x,x') de H existe um valor u tal que o termo A=u es

tá especificado na lista de valores de atributos do rótulo de x ou de x'. Mais ainda, as restrições sobre os nomes em um grafo-esquema impostas pela def.2.3, item d, garantem que se X ≠ X' então A não pode ser um atributo associado aos rótulos de X e de X' ao mesmo tempo.

Definição 12.4 (valor de uma expressão-de-caminho)

⌜Sejam GE um grafo-esquema e GBD um grafo-base-de-dados que corresponde a GE.

Sejam r uma expressão-de-caminho em GE e E = $\{w_1=y_i, w_2=y_2,\ldots,w_m=y_m\}$ um estado relativo ao par (GE,GBD) tais que r é bem especificada em relação ao contexto $\{w_1,w_2,\ldots,w_m\}$.

Seja j o índice tal que $1\leq j\leq m$ e w_j é a variável associada (def. 5.3) à r. (A afirmação de que r é bem especificada em relação a $\{w_1,w_2,\ldots,w_m\}$ garante que existe um índice j nessas condições, e a definição de estado garante que ele é único.)

Se r = w_j, então o *valor* de r para o par (GBD,E) é y_j. Se r é da forma f of w_j, onde f é um prefixo, então o *valor* de r para o par (GBD,E) é o valor coletado em GBD pelo prefixo f a partir de b = $(X,X,\{(y_j,y_j)\})$ e E, onde X é o domínio (def.5.2) de w_j.⌟

As definições apresentadas nesta seção são ilustradas pelo cálculo do valor u da expressão-de-caminho

<NOME,QTD> of itens-em-estoque of FORNECEDOR (relativa ao grafo-esquema GE da figura II-B) para o grafo-base-de-dados GBD da figura II-C e para o estado E = {FORNECEDOR = N11} . (Esse cálculo já foi tratado informalmente no exemplo (e) da seção 5).

Pela def. 12.4 esse valor é dado por $f_1^*(GBD,b_1,E)$, onde f_1 = <NOME,QTD> of itens-em-estoque e b_1 = (FORNECEDOR , FORNECEDOR,{(N11,N11)}).

Pela def. 12.3, item 3, $u = f_2^*(GBD,b_2,E)$, onde f_2 = <NOME,QTD> e b_2 é a base gerada pelo nome de conexão itens-em-estoque a partir de b_1.

Pela def. 12.2,

b_2 = (ITEM,ESTOQUE,{(N31,N21),(N32,N22),(N39,N29)}).

Sejam as bases b_{21}=(ITEM,ESTOQUE,{(N31,N21)}), b_{22} = = (ITEM,ESTOQUE,{(N32,N22)}) e b_{23} = (ITEM,ESTOQUE,{(N39,N29)}).

Pela def. 12.3, item 4, tem-se

u = {<NOME*(GBD,b_{21},E), QTD*(GBD,b_{21},E)>,

<NOME*(GBD,b_{22},E), QTD*(GBD,b_{22},E)>,

<NOME*(GBD,b_{23},E), QTD*(GBD,b_{23},E)>}.

Pela def. 12.3, item 1, conclui-se então que

u = {<{'CADERNO'},{5000}>, <{'BORRACHA'},{5020}> ,

<{'RÉGUA"},{1000}>} .

13 - COMENTÁRIOS

Conforme citado na introdução, a linguagem descrita neste capítulo não corresponde exatamente à linguagem GORDAS, tal como descrita pelos seus autores em /ELWI81/. A principal diferença é que a linguagem original trata valores isolados (escalares) e conjuntos, enquanto que a linguagem aqui descrita trata apenas conjuntos. A conveniência da introdução dessa modificação na linguagem foi evidenciada pela análise de uma inconsistência existente na descrição da linguagem original, causada pela distinção entre valores isolados e os conjuntos unitários correspondentes, conforme será descrito a seguir.

Na linguagem original, uma expressão-de-caminho pode representar valores isolados ou conjuntos. Por exemplo, a expressão-de-caminho NOME of cidade of CLIENTE representa valores isolados (para cada cliente, o nome da cidade onde ele está localizado) e a expressão-de-caminho NOME of clientes of CIDADE representa conjuntos (para cada cidade, o conjunto dos nomes dos clientes nela localizados). O fato de uma expressão-de-caminho representar valores isolados ou conjuntos depende da expressão em si e das partes de restrição (def. 2.2) associadas às arestas do grafo-esquema ao qual ela se refere.

Analogamente, uma expressão-de-conjunto também pode representar valores isolados ou conjuntos. Uma expressão-de-conjunto pode representar valores isolados nos seguintes casos:

(i) ela é da forma count (...);

(ii) ela é uma constante que denota um valor isolado;

(iii) ela é uma expressão-de-caminho (ou um prefixo que "funciona" como uma expressão-de-caminho, conforme descrito na seção 7) que representa valores isolados.

Conforme descrito na seção 6, uma expressão-de-conjunto pode ser utilizada para formar outras expressões-de-conjunto, através do uso dos operadores count, union, inters e dif, ou para formar predicados, através do uso dos operadores includes, $=$, $\neq$, $<$, $\leq$, $>$, $\geq$. Na linguagem original, somente as expressões-de-conjunto que representam valores isolados podem ser utilizadas para formar predicados da forma s_1 op s_2, onde op é um dos operadores $<$, $\leq$, $>$, $\geq$. Nos demais casos de utilização de expressões-de-conjunto não há qualquer consideração quanto ao fato delas representarem valores isolados ou conjuntos, donde se conclui que sempre que necessário os valores isolados são "convertidos" nos conjuntos unitários correspondentes. Por exemplo, num predicado da forma s_1 includes s_2 em que a expressão-de-conjunto s_2 representa valores isolados, toda vez que ela é calculada o valor obtido é convertido no conjunto unitário correspondente, para compatibilizá-lo com o operador includes.

Apesar da citada restrição quanto ao uso dos operadores $<$, $\leq$, $>$, $\geq$, o texto dos autores apresenta um exemplo de uma consulta (ao qual é dado bastante destaque) onde aparece um predicado da forma $s_1 > s_2$ no qual s_1 e s_2 representam conjuntos. Esse exemplo corresponde ao exemplo (f) da seção 8, e as expressões-de-conjunto s_1 e s_2 correspondem respectivamente as

expressões QTD of fornecedores: (NOME = NOME of FORNECEDOR) of ITEM e QTD of clientes: (NOME = NOME of CLIENTE) of ITEM. A consulta tratada no referido exemplo (que não está coerente com a definição da linguagem original) é bastante razoável do ponto de vista prático, pois de fato as expressões-de-conjunto s_1 e s_2 não representam conjuntos quaisquer, mas, devido aos predicados de restrição, representam apenas conjuntos unitários ou o conjunto vazio. A "lógica" dessa consulta supõe que o predicado $s_1 > s_2$ assume o valor falso quando uma das duas expressões-de-conjunto resulta no conjunto vazio (o que não é razoável, pois nesse caso o predicado not$(s_1 > s_2)$ não seria equivalente a $s_1 \leq s_2$), e quando ambas resultam em conjuntos unitários os seus elementos são comparados da forma usual para determinar o valor do predicado.

Para não impedir desnecessariamente a construção de consultas desse tipo julgou-se que seria conveniente eliminar a citada restrição da linguagem, permitindo a utilização de expressões-de-conjunto que representam conjuntos na construção dos predicados que envolvem comparações através dos operadores $<$, $\leq$, $>$, $\geq$, e transferindo para o usuário a responsabilidade de assegurar que as comparações sejam realizadas somente entre conjuntos unitários (ver regras 6.B e 6.D na seção 6).

Porém, com essa modificação, passaria a ser indiferente para o usuário da linguagem o fato de uma expressão-de-conjunto representar um valor isolado ou um conjunto, pois para todos os efeitos um valor isolado seria equivalente ao conjunto unitário correspondente.

Diante desse fato decidiu-se introduzir uma modificação mais abrangente na linguagem original, eliminando o tratamento de valores isolados. Conforme citado na seção 4 a linguagem assim modificada passou a permitir o tratamento de atributos multi-valorados ("multi-valued attributes") de forma natural, e a sua descrição tornou-se mais simples e uniforme.

Para encerrar este capítulo cabe mencionar que a linguagem GORDAS possui outros recursos que não foram mencionados até então, por serem pouco significativos do ponto de vista da "potência" de consulta da linguagem, e por não estarem claramente definidos no texto dos autores, onde eles são apenas citados. Alguns desses recursos são: as expressões aritméticas, as funções sum, avg, sd , que retornam respectivamente a soma, média e desvio padrão de um conjunto cujos elementos são valores-básicos numéricos (na linguagem aqui apresentada essas funções seriam aplicáveis a valores tipo n), e as funções max[i] e min [i] , que retornam respectivamente o i-ésimo maior e o i-ésimo menor valor-básico de um conjunto cujos elementos são valores-básicos numéricos ou alfabéticos (na linguagem aqui apresentada essas funções seriam aplicáveis a valores tipo a ou n). Também não foram considerados neste trabalho os recursos da linguagem para definição de esquemas e para atualização da base de dados, tal como foi feito em /ELWI81/. Na versão da linguagem para o modelo ECR, descrita em /ELMA81/, estão definidos também esses recursos.

CAPÍTULO III

A Completude ER de Linguagens de Consulta

1 - INTRODUÇÃO

O conceito de "completude" de uma linguagem de consulta a bases de dados foi apresentado inicialmente por E.F.Codd em /CODD72/, para as linguagens baseadas no modelo relacional de dados, quando então ele foi denominado "completude relacional".

Segundo Date (em /DATE77/), o fato de uma linguagem ser caracterizada como "completa" pode ser intuitivamente interpretado pelo seu usuário da seguinte maneira: "se a informação requerida está na base de dados, então ela pode ser obtida através de um único comando de consulta".

É interessante observar que esse conceito se refere exclusivamente aos recursos oferecidos pela linguagem para a obtenção de dados da base de dados, sem considerar outras características da linguagem (capacidade de realização de operações aritméticas, funções pré-definidas disponíveis, etc.). Portanto, do ponto de vista do usuário, uma boa linguagem de consulta de-

ve ser "mais que completa".

Formalmente, o conceito de completude é definido a partir de uma linguagem de consulta a bases de dados baseada no cálculo de predicados (ver p.ex. /HEGE73/), que serve como padrão para avaliação das outras linguagens. Diz-se que uma linguagem de consulta é completa se toda consulta formulada através da linguagem padrão pode também ser formulada através da linguagem em questão.

Esse conceito foi definido para as linguagens de consulta baseadas no modelo ER por P.P. Chen (que introduziu o modelo ER) e P. Atzeni em /ChAt81/, onde ele foi denominado "completude ER".

2 - O CÁLCULO ER

O Cálculo ER é uma linguagem de consulta a bases de dados para o modelo ER baseada no cálculo de predicados, introduzida por P.P. Chen e P. Atzeni em /ChAt81/. As definições formais apresentadas nesta seção são traduções praticamente diretas de definições dos autores, os quais utilizam o termo *conjunto-objeto* para designar um conjunto-entidade ou um conjunto-relacionamento, e o termo *objeto* para designar uma entidade ou um relacionamento.

No que segue, supõe-se conhecidos os conceitos elementares do cálculo de predicados, em particular os significados dos símbolos $\vee$, $\wedge$, $\neg$, $\exists$, $\forall$ e os conceitos de fórmula, variá

vel ligada, variável livre e sentença.

Definição 2.1 (expressão do Cálculo ER)

Uma *expressão do Cálculo ER* é da forma

<lista-de-domínios> (<lista-objetivo>): <predicado> ,

onde:

1) <lista-de-domínios> é da forma $x_1:X_1$, $x_2:X_2$,..., $x_m:X_m$, $x_1,x_2,\ldots,x_m$ são variáveis distintas que aparecem no <predicado> e $X_1,X_2,\ldots,X_m$ são conjuntos-objeto, não necessariamente distintos, que constituem os domínios respectivamente das variáveis $x_1,x_2,\ldots,x_m$ (os valores que a variável x_i pode assumir são os elementos do conjunto-objeto X_i, $1 \leq i \leq m$);

2) <lista-objetivo> é da forma $x_{j_1},x_{j_2},\ldots,x_{j_n}$, onde $x_{j_1},x_{j_2},\ldots,x_{j_n}$ são variáveis distintas que aparecem na <lista-de-domínios> ;

3) <predicado> é uma fórmula, construída a partir de átomos e operadores, que satisfaz as seguintes condições:

3.1 - Os átomos são de uma das três seguintes formas:

(i) $x_h.X_k = x_k$, onde x_h é uma variável cujo domínio X_h é um conjunto-relacionamento que envolve o conjunto-entidade X_k, que é o domínio da variável x_k. Esse átomo assume o valor verdadeiro para valores $\bar{x}_h$ e $\bar{x}_k$ das

variáveis x_h e x_k se e somente se o relacionamento $\bar{x}_h$ envolve a entidade $\bar{x}_k$;

(ii) $x_h.A \; \nu \; x_k.A'$, onde x_h e x_k são variáveis (não necessariamente distintas) associadas ao mesmo domínio X_h, $\nu \in \{=, \neq, <, \leq, >, \geq\}$, e A, A' são atributos do conjunto-objeto X_h definidos sobre conjuntos-de-valores D e D' cujos elementos são comparáveis através do operador ν, isto é, para quaisquer $d \in D$ e $d' \in D'$ o valor $d \; \nu \; d$ está definido (é falso ou verdadeiro). Esse átomo assume o valor verdadeiro para valores $\bar{x}_h$ e $\bar{x}_k$ das variáveis x_h e x_k se e somente se o valor do atributo A para o objeto $\bar{x}_h$ está na relação ν com o valor do atributo A' para o objeto $\bar{x}_h$;

(iii) $x_h.A \; \nu \; k$, onde x_h é uma variável, A é um atributo do conjunto-objeto X_h (domínio de x_h), $\nu \in \{=, \neq, <, \leq, >, \geq\}$, e K é uma constante que pode ser comparada através do operador ν com qualquer elemento do conjunto-de-valores sobre o qual está definido o atributo A. Esse átomo assume o valor verdadeiro para um valor $\bar{x}_h$ da variável x_h se e somente se o valor do atributo A para o objeto $\bar{x}_h$ está na relação ν com a constante K.

3.2 - A fórmula é construída segundo a seguinte definição:

(i) um átomo é uma fórmula;

(ii) se ψ e Ω são fórmulas, então $\psi \wedge \Omega$, $\psi \vee \Omega$, $\neg \psi$, e (ψ) são fórmulas;

(iii) se ψ é uma fórmula que contém pelo menos duas variáveis livres x e x', então $\exists x(\psi)$ e $\forall x(\psi)$ são fórmulas.

3.3 - As variáveis livres do predicado são exatamente as variáveis $x_{j_1}, x_{j_2}, \ldots, x_{j_n}$ da <lista-objetivo> . ⊣

Definição 2.2 (expressão do cálculo ER simplificado)

⌜ Uma *expressão do Cálculo ER simplificado* é uma expressão do Cálculo ER cuja lista-objetivo é constituída por uma única variável. ⊣

Como não há no texto dos autores nenhuma consideração sobre a "precedência" dos conectivos $\neg$, $\vee$, $\wedge$ na interpretação de uma fórmula, será adotada neste texto a seguinte convenção: o conectivo $\neg$ tem a maior precedência, e a precedência dos conectivos $\vee$ e $\wedge$ é determinada "da esquerda para a direita", ou seja, dadas duas ocorrências desses conectivos na fórmula, a ocorrência que aparece mais à esquerda tem a maior precedência

(supondo que não haja parênteses determinando o contrário, naturalmente). Com essa convenção, a fórmula $(A_1 \wedge A_2 \vee \neg A_3 \vee \neg A_4 \wedge A_5)$, onde $A_1, A_2, \ldots, A_5$ são átomos, é equivalente à fórmula $((((A_1 \wedge A_2) \vee (\neg A_3)) \vee (\neg A_4)) \wedge A_5)$.

Para definir o significado de uma expressão do Cálculo ER é necessário salientar que, embora não esteja mencionado explicitamente na definição, cada expressão do Cálculo ER é definida com base em um "esquema ER" específico, ou seja, em um modelo de dados do "tipo ER" específico. (É interessante observar que uma mesma expressão do Cálculo ER pode ser definida com base em vários esquemas ER diferentes.)

Uma expressão do Cálculo ER referente a um certo esquema ER é uma consulta a qualquer base de dados correspondente a esse esquema (ou seja, "modelada" por esse esquema). Para cada posição (estado, instância) de uma dessas bases de dados a expressão define um *conjunto-resposta* que é o conjunto das ênuplas de valores das variáveis $x_{j_1}, x_{j_2}, \ldots x_{j_n}$ da <lista-objetivo> que "satisfazem" o predicado. Formalmente pode-se dizer que uma ênupla $(\bar{x}_{j_1}, \bar{x}_{j_2}, \ldots, \bar{x}_{j_n})$, onde $\bar{x}_{j_1}, \bar{x}_{j_2}, \ldots, \bar{x}_{j_n}$ são objetos que pertencem respectivamente aos domínios das variáveis $x_{j_1}, x_{j_2}, \ldots, x_{j_n}$, pertence ao conjunto-resposta da expressão se e somente se a substituição, no <predicado> da expressão, de toda ocorrência livre da variável x_{j_i} pelo objeto x_{j_i}, para $i = 1,2,\ldots,n$, transforma esse <predicado> em uma sentença verdadeira.

Os exemplos seguintes mostram consultas formuladas em

linguagem natural (relativas a uma base de dados correspondente ao esquema ER parcialmente representado na figura II.A) e as expressões do Cálculo ER cujos conjuntos-resposta são os conjuntos de objetos (ou de ênuplas de objetos) requeridos por essas consultas.

- Exemplos:

(a) Obtenha as cidades do estado de São Paulo:

x_1:CIDADE(x_1): x_1.ESTADO = 'SP'

(b) Obtenha os itens requeridos pelo cliente Estudante Feliz:

x_1:CLIENTE, x_2:PEDIDO, x_3:ITEM(x_3):

$\exists x_1(x_1.\text{NOME} = \text{'ESTUDANTE FELIZ'} \wedge \exists x_2(x_2.\text{CLIENTE}=x_1 \wedge x_2.\text{ITEM}=x_3))$

(c) Obtenha os fornecedores que possuem em estoque todos os itens requeridos pelo cliente Estudante Feliz:

x_1:CLIENTE, x_2:PEDIDO, x_3:ITEM, x_4:ESTOQUE, x_5:FORNECEDOR (x_5):

$\exists x_1(x_1.\text{NOME} = \text{'ESTUDANTE FELIZ'} \wedge$

$\forall x_3(\neg \exists x_2(x_2.\text{CLIENTE} = x_1 \wedge x_2.\text{ITEM} = x_3) \vee$

$\exists x_4(x_4.\text{ITEM} = x_3 \wedge x_4.\text{FORNECEDOR} = x_5)))$

(d) Obtenha os pares (fornecedor, cliente) tais que o fornecedor possui em estoque pelo menos um dos itens requeridos pelo cliente:

x_1:CLIENTE, x_2:PEDIDO, x_3:ITEM, x_4:ESTOQUE, x_5:FORNECEDOR (x_5, x_1):

$$\exists x_3(\exists x_2(x_2.\text{CLIENTE}=x_1 \wedge x_2.\text{ITEM}=x_3) \wedge \exists x_4(x_4.\text{ITEM}=x_3 \wedge x_4.\text{FORNECEDOR}=x_5))$$

(e) Obtenha os pares (fornecedor, cliente) tais que o fornecedor está localizado na mesma cidade que o cliente e possui em estoque todos os itens por ele requeridos:

x_1:CLIENTE, x_2:PEDIDO, x_3:ITEM, x_4:ESTOQUE, x_5:FORNECEDOR, x_6:FORN-CID, x_7:CIDADE, x_8:CLI-CID (x_5, x_1):

$$\exists x_7(\exists x_6(x_6.\text{FORNECEDOR}=x_5 \wedge x_6.\text{CIDADE}=x_7) \wedge \exists x_8(x_8.\text{CLIENTE}=x_1 \wedge x_8.\text{CIDADE}=x_7)) \wedge \forall x_3(\neg\exists x_2(x_2.\text{CLIENTE}=x_1 \wedge x_2.\text{ITEM}=x_3) \vee \exists x_4(x_4.\text{FORNECEDOR}=x_5 \wedge x_4.\text{ITEM}=x_3))$$

3 - A COMPLETUDE ER DE LINGUAGENS DE CONSULTA

A partir do Cálculo ER pode ser definido o conceito de "completude ER" para as linguagens de consulta baseadas no modelo ER.

Definição 3.1 (completude ER)

Uma linguagem de consulta L tem a propriedade da *completude ER* se para toda expressão do Cálculo ER existe uma ex-

pressão equivalente na linguagem L. ⌋

Por "expressão na linguagem L" designa-se uma cadeia de caracteres dessa linguagem que representa uma consulta à base de dados. Por exemplo, para a linguagem GORDAS o termo "expressão" deve ser interpretado como "consulta".

Dadas uma expressão do Cálculo ER e uma consulta em uma linguagem L, ambas referentes a um mesmo esquema ER, diz-se que elas são *equivalentes* se para qualquer posição de uma base de dados correspondente ao referido esquema ER ambas "selecionam" o mesmo conjunto de ênuplas de objetos.

Definição 3.2 (completude ER simplificada)

⌈ Uma linguagem de consulta L tem a propriedade da *completude ER simplificada* se para toda expressão do Cálculo ER simplificado existe uma expressão equivalente na linguagem L. ⌋

No texto /ChAt81/, onde os conceitos de completude ER e completude ER simplificada são introduzidos, os autores salientam que nenhuma das linguagens de consulta baseadas no modelo ER conhecidos até aquela data (esse trabalho foi publicado no fim de 1981) tem a propriedade da completude ER simplificada, e portanto nenhuma delas tem a propriedade da completude ER. São citadas especificamente as linguagens CLEAR /Poon79/, CABLE /Shos78/ e Executable Language /AtVi81/. Nesse mesmo trabalho é apresentada uma nova versão da Executable Language que tem a propriedade da completude ER simplificada.

No Capítulo seguinte será demonstrado que a linguagem GORDAS, descrita no Capítulo II, tem a propriedade da completude ER. A descrição dessa linguagem foi publicada na mesma época (fim de 1981) que o referido texto, o que poderia explicar o fato dela não ter sido considerada no mesmo.

4 - COMENTÁRIOS

O Cálculo ER, tal como foi definido em /ChAt81/ e apresentado na Seção 2, tem duas limitações que devem ser destacadas.

A primeira limitação é que o Cálculo ER não considera a possibilidade de um relacionamento envolver mais de uma entidade de um mesmo conjunto-entidade. Por exemplo,cada relacionamento do conjunto-relacionamento CASAMENTO envolve duas entidades do conjunto-entidade PESSOA, e nesse caso o significado de um átomo da forma x . PESSOA = y , onde x é uma variável cujo domínio é CASAMENTO e y uma variável cujo domínio é PESSOA, não está bem definido.

Para superar essa limitação é necessário que o Cálculo ER permita caracterizar cada participação de um conjunto-entidade em um conjunto-relacionamento, o que requer a introdução do conceito de papel ("role" em /CHEN76/) de uma entidade em um relacionamento. Uma possibilidade seria a introdução no Cálculo ER de átomos da forma $x_h . X_k(P) = x_k$, onde x_h é uma variável cujo domínio é um conjunto-relacionamento X_h que envolve o conjunto-entidade X_k, domínio de x_k, e P é um particular pa-

pel das entidades do conjunto-entidade X_k no conjunto-relacionamento X_h. Um átomo dessa forma assumiria o valor verdadeiro para valores $\bar{x}_h$ e $\bar{x}_k$ das variáveis x_h e x_k se e somente se a entidade $\bar{x}_k$ participasse do relacionamento $\bar{x}_h$ no papel P.

A segunda e mais grave limitação é a impossibilidade de comparar valores de atributos de objetos pertencentes a conjuntos-objeto diferentes (nos átomos da forma $x_h.A \; \nu \, x_k.A'$, pelos quais são especificadas as comparações, as variáveis x_h e x_k devem ter o mesmo domínio). Essa limitação provavelmente foi introduzida no Cálculo ER para assegurar o princípio adotado pelos seus criadores de que "uma linguagem completa não precisa ser capaz de expressar consultas envolvendo comparações entre objetos não relacionados", onde o termo "objetos não relacionados " designa objetos pertencentes a conjuntos-objeto cujas figuras no diagrama ER não estão "ligadas" por um caminho. De fato, se nos átomos da forma $x_h.A \; \nu \; x_k.A'$ os domínios de x_h e x_k pudessem ser quaisquer conjuntos-objeto esse princípio seria violado.

Ocorre porém que a exigência de que x_h e x_k tenham o mesmo domínio é muito forte, e impede também comparações entre valores de atributos de objetos "relacionados". Por exemplo, considerando o esquema ER representado parcialmente pelo diagrama ER da figura II.A, não é possível construir uma expressão do Cálculo ER para a consulta "obtenha os pares de fornecedores e clientes tais que o fornecedor possui em estoque todos os itens requeridos pelo cliente, em quantidades maiores que aquelas por ele requeridas", pois é necessário comparar o valor do atributo QTD de um relacionamento do conjunto-relacionamento ESTOQUE com

o valor do atributo QTD de um relacionamento do conjunto-relacionamento PEDIDO, o que é impossível no Cálculo ER.

Para superar essa limitação sem violar o referido princípio é necessário formalizar o conceito de "objetos relacionados" e introduzí-lo na definição dos átomos da forma $x_h.A \ \nu \ x_k.A'$.

Não é razoável que uma linguagem de consulta que tenha alguma das citadas limitações seja considerada "completa" , o que sugere que essas limitações devem ser eliminadas do Cálculo ER. Não há interesse aqui em formalizar essas extensões,pois para efeito de estudar a completude da linguagem GORDAS elas não são relevantes, já que essa linguagem permite caracterizar o papel de uma entidade em um relacionamento (através de um nome de conexão) e permite comparar valores de atributos de quaisquer objetos, estejam eles "relacionados" ou não. Assim, mesmo que essas extensões fossem introduzidas no Cálculo ER, a linguagem GORDAS não perderia a propriedade da completude ER.

Além dessas limitações, há outras características "estranhas" do Cálculo ER que devem ser mencionadas.

Uma dessas características é ilustrada pela expressão do Cálculo ER

$$x_1 : \text{FORNECEDOR}(x_1) \ : \ x_1.\text{NOME} = x_1.\text{NOME}$$

que corresponde à consulta "obtenha todos os fornecedores". O <predicado> "artificial" dessa expressão foi especificado ape-

nas para satisfazer a exigência da presença de um < predicado > em uma expressão do Cálculo ER. Essa característica decorre da introdução no Cálculo ER do conceito de <lista-de-domínios> que do ponto de vista da lógica formal corresponderia também a uma fórmula como $x_1 \varepsilon$FORNECEDOR que no exemplo citado assume o valor verdadeiro para um valor qualquer $\bar{x}_1$ da variável x_1 se e somente se o objeto x_1 pertence ao conjunto-objeto FORNECEDOR (ver p. ex., /HEGE73/).

Outra característica "estranha" refere-se ao modo como o uso de quantificadores foi introduzido nas fórmulas. Na definição 2.1, item 3.3, aparece o seguinte: "se Ψ é uma fórmula que contém pelo menos *duas* variáveis livres x e x', então $\exists x(\Psi)$ e $\neg\forall x\,(\Psi)$ são fórmulas". A exigência da presença da variável livre x', não usual na lógica formal, provavelmente deve-se a considerações semânticas, tendo os autores considerado "sem sentido" a utilização de uma sentença (fórmula que não contém variáveis livres) na construção de um <predicado> . Ocorre porém que essa restrição imposta na definição é inócua, conforme ilustrado pela expressão

$$x_1 : \text{ITEM},\ x_2 : \text{CLIENTE}(x_1) :$$

$$\exists x_2 (x_2.\text{END} = \text{'AV.MARGINAL S/N'} \wedge x_1.\text{NOME} \neq \text{'LÁPIS'}).$$

Do ponto de vista da lógica formal, esse <predicado> é equivalente à fórmula (na terminologia da lógica)

$$x_1.\text{NOME} \neq \text{'LÁPIS'} \wedge \exists x_2 (x_2.\text{END} = \text{'AV.MARGINAL S/N'}) \quad ,$$

o que mostra que é possível utilizar uma sentença em um <predicado> , apesar da referida restrição. Outra maneira de "burlar" a restrição seria através do uso de um átomo "artificial", como por exemplo na seguinte expressão (que é equivalente à anterior):

$$x_1\text{:ITEM},\ x_2\text{:CLIENTE}\ (x_1):\ x_1.\text{NOME} \neq \text{'LÁPIS'} \wedge$$

$$\exists x_2(x_2.\text{END}=\text{'AV.MARGINAL S/N'} \wedge x_1.\text{END}=x_1.\text{END}).$$

É fácil verificar que os "truques" ilustrados por esses exemplos sempre podem ser utilizados, o que mostra que a referida restrição é de fato inócua.

CAPÍTULO IV

A Completude ER da Linguagem GORDAS

Neste capítulo será demonstrado que a linguagem GORDAS tem a propriedade da completudo ER. Isso será feito através do conceito de ξ-expressão, que é uma generalização do conceito de expressão do Cálculo ER.

Definição 1 (ξ-expressão)

⌜Uma *ξ-expressão* é da forma

<lista-de-domínios>(<lista-objetivo>):<predicado>,

onde:

1) <lista-de-domínios> é da forma $x_1:X_1,x_2:X_2,\ldots,x_n:X_m$, onde $x_1,x_2,\ldots,x_m$ são variáveis distintas e $X_1,X_2,\ldots,X_m$ são conjuntos-objeto, não necessariamente distintos, que são os domínios respectivamente das variáveis $x_1,x_2,\ldots,x_m$.

2) <lista-objetivo> é da forma $x_{j_1},x_{j_2},\ldots,x_{j_n}$, onde $x_{j_1},x_{j_2},\ldots,$

x_{j_n} são variáveis distintas que aparecem na <lista-de-domínios>.

3) <predicado> é uma fórmula, definida como no cap. III , def. 2.1, itens 3.1 e 3.2, cujos átomos envolvem apenas variáveis que aparecem na <lista-de-domínios> e cujas variáveis livres aparecem na <lista-objetivo>.

Dessa definição decorre imediatamente que toda expressão do Cálculo ER é uma ξ-expressão, mas a recíproca não é verdadeira, porque na <lista-objetivo> de uma ξ-expressão pode aparecer uma variável que não é uma variável livre do <predicado>, o que não pode ocorrer em uma expressão do Cálculo ER.

Definição 2 (ênupla de objetos de uma base de dados que satisfaz uma ξ-expressão)

⌜ Seja Γ um esquema ER e

$$x_1:X_1, x_2:X_2, \ldots, x_m:X_m(x_{j_1}, x_{j_2}, \ldots, x_{j_n}):\Pi$$

uma ξ-expressão referente ao esquema Γ

Seja β uma posição específica de uma base de dados correspondente a Γ, e seja $t=(z_1, z_2, \ldots, z_n)$ uma ênupla de objetos de β.

Diz-se que a ênupla t *satisfaz* a ξ-expressão δ se e somente se:

(i) para i=1,2,...,n, o objeto z_i pertence ao conjunto-objeto X_{j_i};

(ii) a substituição em Π de toda ocorrência livre da variável x_{j_i} pelo objeto z_i, para i=1,2,..,n, transforma Π em uma sentença verdadeira. ⌋

Definição 3 (ênupla de objetos de uma base de dados representada por uma ênupla de nós de um grafo-base-de-dados)

⌈ Seja Γ um esquema ER, e GE um grafo-esquema (cap.II, def. 2.1) que representa Γ.

Seja β uma posição específica de uma base de dados correspondente a Γ, e GBD o grafo-base-de-dados (cap.II, def.2.4) que corresponde (cap. II, def. 2.6) a GE e que representa β.

Seja $u = \langle y_1, y_2, \ldots, y_n \rangle$ uma ênupla de nós de GBD, e seja $t=(z_1, z_2, \ldots, z_n)$ a ênupla de objetos de β tal que z_i é o objeto representado pelo nó y_i, para i=1,2,...,n.

Diz-se que a ênupla u *representa* a ênupla t, e que a ênupla t é *representada* pela ênupla u. ⌋

Definição 4 (ξ-equivalência)

⌈ Seja Γ um esquema ER, e GE um grafo-esquema que representa Γ.

Seja $\delta = x_1:X_1, x_2:X_2, \ldots, x_m:X_m(x_{j_1}, x_{j_2}, \ldots, x_{j_n}):\Pi$ uma ξ-expressão referente ao esquema Γ.

Diz-se que uma consulta q (cap. II, def. 8.1) no grafo-esquema GE é *ξ-equivalente* a δ se e somente se as seguintes condições são satisfeitas:

(i) a cláusula get da consulta q é da forma get $v_1, v_2, \ldots, v_n$, onde os domínios (cap. II, def. 5.2) das variáveis $v_1, v_2, \ldots, v_n$ (que são nós de GE) representam respectivamente os domínios das variáveis $x_{j_1}, x_{j_2}, \ldots, x_{j_n}$ (que são conjuntos-objeto);

(ii) para qualquer posição β de uma base de dados correspondente a Γ o valor V da consulta q para o par (GBD,ϕ) (cap. II, def. 11.2), onde GBD é o grafo-base-de-dados que corresponde a GE e que representa β, é formado exatamente pelas ênuplas de nós de GBD que representam as ênuplas de objetos de β que satisfazem a ξ-expressão δ. ⌟

Proposição:

Seja Γ um esquema ER, e GE um grafo-esquema que representa Γ.

Para toda ξ-expressão

$$\delta = x_1:X_1, x_2:X_2, \ldots, x_m:X_m(x_{j_1}, x_{j_2}, \ldots, x_{j_n}):\Pi$$

referente ao esquema Γ, existe uma consulta q em GE tal que q é ξ-equivalente a δ.

Demonstração:

Pode-se supor sem perda de generalidade que o <predicado> Π não é da forma (Ψ), onde Ψ é uma fórmula, pois obivamente se existe uma consulta que é ξ-equivalente a uma ξ-expressão δ cujo <predicado> é Ω essa consulta é também ξ-equivalente à ξ-expressão δ' que se obtem pela substituição em δ do <predicado> Ω por (Ω).

A demonstração será feita por indução sobre o *índice* do <predicado> Π, que é o número de ocorrências de conectivos $(\neg, \vee, \wedge)$ e quantificadores $(\exists, \forall)$ em Π.

Base da Indução:

Se o índice de Π é 0 (zero), então Π é um átomo, é pode-se ter um dos três seguintes casos:

(i) Π é da forma $x_{j_h} . X_{j_k} = x_{j_k}$, onde $h \neq k$ e $1 \leq h,k \leq n$.

Seja c o segundo nome de conexão especificado no rótulo (cap.II,def. 2.2) da aresta D que liga o nó cujo nome é X_{j_k} ao nó cujo nome é X_{j_h} no grafo-esquema GE (c é o nome de conexão associado ao sentido oposto ao sentido da referida aresta).

A consulta

$$q = \underline{\text{get}}\ X_{j_1}.j_1, X_{j_2}.j_2, \ldots, X_{j_n}$$

$$\underline{\text{where}}\ c\ \text{of}\ X_{j_h}.j_h = X_{j_k}.j_k$$

é ξ-equivalente a δ, conforme será justificado a seguir.

Por construção a condição (i) da ξ-equivalência é verificada por essa consulta.

Seja β uma posição de uma base de dados correspondente a Γ, e GBD o grafo-base-de-dados que corresponde a GE e representa β.

Pela definição 11.2 do capítulo II, o valor da consulta q para o par (GBD,ϕ) é o conjunto V de ênuplas de nós de GBD tal que uma ênupla $\langle y_1, y_2, \ldots, y_n \rangle$ pertence a V se e somente se os nós $y_1, y_2, \ldots, y_n$ pertencem (cap. II, def. 2.6) respectivamente aos nós de GE cujos nomes são $X_{j_1}, X_{j_2}, \ldots, X_{j_n}$ e o nó y_{j_k} está ligado ao nó y_{j_h} por uma aresta que pertence (cap. II, def. 2.6) a D.

Para i=1,2,..,n, o nó de GE cujo nome é X_{j_i} representa o conjunto-objeto X_{j_i}, e um nó y de GBD pertence ao nó X_{j_i} se e somente se y representa um objeto do conjunto-objeto X_{j_i}.

Segue então que uma ênupla de nós $\langle y_i, y_2, \ldots, y_n \rangle$ pertence a V se e somente se a ênupla de objetos $(\bar{x}_{j_1}, \bar{x}_{j_2}, \ldots, \bar{x}_{j_n})$ por ela representada pertence ao conjunto

$X_{j_1} \times X_{j_2} \times \ldots \times X_{j_n}$ e o relacionamento $\bar{x}_{j_h}$ envolve a entidade $\bar{x}_{j_k}$.

Conclui-se então que V é formado exatamente pelas ênuplas de nós de GBD que representam as ênuplas de objetos de β que satisfazem à ξ-expressão δ.

Como os argumentos utilizados se aplicam a qualquer posição β de uma base de dados que corresponde a Γ, conclui-se que q é de fato ξ-equivalente a δ.

É interessante observar que poderia haver mais de uma aresta ligando o nó X_{j_k} ao nó X_{j_h}, caso os relacionamentos do conjunto-relacionamento X_{j_h} envolvessem mais de uma entidade do conjunto-entidade X_{j_k}.

Porém, conforme citado no cap.III, seção 4, o Cálculo ER não prevê esse caso, e pode-se supor então que a aresta D é única.

Caso o Cálculo ER fosse estendido para tratar esse caso a escolha do nome de conexão c deveria ser compatível com o papel especificado no átomo em questão.

Nota: De um modo geral a verificação de ξ-equivalência em cada um dos casos previstos nesta demonstração é relativamente simples, embora trabalhosa. Por esse motivo na maioria dos casos restantes essa verificação será omitida.

(ii) Π é da forma $x_{j_h}.A \; \nu \; x_{j_k}.A'$, com $1 \leq h,k \leq n$, $\nu \in \{=, \neq, <, \leq, >, \geq\}$

Nesse caso a consulta

$$q = \underline{\text{get}}\ X_{j_1}.j_1, X_{j_2}.j_2, \ldots, X_{j_n}.j_n$$

$$\underline{\text{where}}\ A\ \underline{\text{of}}\ X_{j_h}.j_h \vee A'\ \underline{\text{of}}\ X_{j_k}.j_k$$

é ξ-equivalente a δ.

iii) Π é da forma $x_{j_h}.A \vee K$, com $1 \leq h \leq n$.

Nesse caso a consulta

$$q = \underline{\text{get}}\ x_{j_1}.j_1, X_{j_2}.j_2, \ldots, X_{j_n}.j_n$$

$$\underline{\text{where}}\ A\ \underline{\text{of}}\ X_{j_h}.j_h \vee K$$

é ξ-equivalente a δ.

Passo da Indução:

Se Π é de índice $N>0$, pode-se ter um dos casos seguintes:

(i) $\Pi = \Psi_1 \vee \Psi_2$ ou $\Pi = \Psi_1 \wedge \Psi_2$, onde Ψ_2 é uma fórmula tal que não existem fórmulas Ω_1 e Ω_2 tais que $\Psi_2 = \Omega_1 \vee \Omega_2$ ou $\Psi_1 = \Omega_1 \wedge \Omega_2$.

Sejam as ξ-expressões

$$\delta_1 = x_1:X_1, x_2:X_2, \ldots, x_m:X_m(x_{j_1}, x_{j_2}, \ldots, x_{j_n}):\Psi_1$$

e $$\delta_2 = x_1:X_1, x_2:X_2, \ldots, x_m:X_m(x_{j_1}, x_{j_2}, \ldots, x_{j_n}):\Psi_2$$

Os índices de Ψ_1 e Ψ_2 são menores que N, e pela hipótese da indução existem consultas q_1 e q_2 que são ξ-equivalentes respectivamente as ξ-expressões s_1 e s_2.

Pela condição (i) da definição 4 (ξ-equivalência) as cláusulas get dessas duas consultas devem ser da forma get $v_1, v_2, \ldots, v_n$, onde $v_1, v_2, \ldots, v_n$ são variáveis cujos domínios devem ser os nós de GE que representam respectivamente os conjuntos-objeto $X_{j_1}, X_{j_2}, \ldots, X_{j_n}$.

Como a substituição, em uma consulta qualquer, de todas as ocorrências de uma variável v por uma variável w , que tem o mesmo domínio que v, gera uma consulta equivalente, pode-se supor que as cláusulas get das consultas q_1 e q_2 são exatamente iguais.

Sejam então

q_1 = get $v_1, v_2, \ldots, v_n$

where p_1

e q_2 = get $v_1, v_2, \ldots, v_n$

where p_2

essas consultas, onde p_1 e p_2 representam as cadeias de caracteres que constituem os predicados (capítulo II, def.6.3) das mesmas.

Se $\Pi = \Psi_1 \vee \Psi_2$, a consulta

$$q = \underline{\text{get}}\ v_1, v_2, \ldots, v_n$$

$$\underline{\text{where}}\ (p_1)\ \underline{\text{or}}\ (p_2)$$

é ξ-equivalente a δ.

Se $\Pi = \Psi_1 \wedge \Psi_2$, a consulta

$$\underline{\text{get}}\ v_1, v_2, \ldots, v_n$$

$$\underline{\text{where}}\ (p_1)\ \underline{\text{and}}\ (p_2)$$

é ξ-equivalente a δ.

Obs.: Em ambos os casos os parênteses que envolvem os predicados p_1 e p_2 não são realmente necessários, mas a sua omissão torna a verificação da ξ-equivalência entre a consulta q e a ξ-expressão δ não trivial.

(ii) $\Pi = \neg\Psi$, onde Ψ é uma fórmula tal que não existem Ω_1 e Ω_2 tais que $\Psi = \Omega_1 \vee \Omega_2$ ou $\Psi = \Omega_1 \wedge \Omega_2$.

Seja a ξ-expressão

$$\delta' = x_1 : X_1, x_2 : X_2, \ldots, x_m : X_m (x_{j_1}, x_{j_2}, \ldots, x_{j_n}) : \Psi$$

O índice de Ψ é (N-1), e pela hipótese da indução existe uma consulta ξ-equivalente a δ'.

Seja $q' = \underline{\text{get}}\ v_1, v_2, \ldots, v_n$

$$\underline{\text{where}}\ p$$

essa consulta, onde p representa a cadeia de caracteres que constitui o seu predicado.

Segue então que a consulta

$$q = \underline{\text{get}}\ v_1, v_2, \ldots, v_n$$

$$\underline{\text{where}}\ \underline{\text{not}}\ (p)$$

é ξ-equivalente a δ.

(iii) $\Pi = \exists\ x_h(\Psi)$

Pode-se supor sem perda de generalidade que a variável x_h não aparece na <lista-objetivo> de δ, pois introduzindo-se na <lista-de-domínios> de δ uma nova variável com o mesmo domínio de x_h, e substituindo-se todas as ocorrências da variável x_h ligadas ao quantificador que inicia o <predicado> Π por essa nova variável obtém-se uma ξ-expressão equivalente.

Segue então que

$$\delta' = x_1{:}X_1, x_2{:}X_2, \ldots, x_m{:}X_m(x_{j_1}, x_{j_2}, \ldots, x_{j_n}, x_h){:}\Psi$$

é uma ξ-expressão, e, como o índice de Ψ é $(N-1)$, pela hipótese da indução existe uma consulta

$$q' = \underline{\text{get}}\ v_1, v_2, \ldots, v_n, w$$

$$\underline{\text{where}}\ p$$

que é ξ-equivalente a δ'.

Seja a consulta

q = get $v_1, v_2, \ldots, v_n$

where count (get w

where p) > 0.

Da ξ-equivalência entre q' e s' decorre que os domínios das variáveis $v_1, v_2, \ldots, v_n$ são os nós de GE que representam os conjuntos-objeto $X_{j_1}, X_{j_2}, \ldots, X_{j_n}$, de forma que a condição (i) da ξ-equivalência entre q e s está satisfeita.

Seja β uma posição de uma base de dados correspondente a Γ, e GBD o grafo-base-de-dados que corresponde a GE e representa β.

Pela definição 11.2 do capítulo II, uma ênupla de nós de GBD $\langle y_1, y_2, \ldots, y_n \rangle$ pertence ao valor V da consulta q para o par (GBD, ϕ) se e somente se existe pelo menos um nó z de GBD (que pertence ao domínio da variável w) tal que a ênupla de nós $\langle y_1, y_2, \ldots, y_n, z \rangle$ pertence ao valor V' da consulta q' para o par (GBD, ϕ).

Como q' é equivalente a s' pela hipótese da indução uma ênupla de nós $\langle y_1, y_2, \ldots, y_n, z \rangle$ pertence ao valor V' se e somente se a ênupla de objetos $(\bar{x}_{j_1}, \bar{x}_{j_2}, \ldots, \bar{x}_{j_n}, \bar{x}_h)$ por ela representada satisfaz a ξ-expressão s', ou seja, se os objetos $\bar{x}_{j_1}, \bar{x}_{j_2}, \ldots, \bar{x}_{j_n}, \bar{x}_h$ pertencem respectivamente aos conjuntos-objeto $X_{j_1}, X_{j_2}, \ldots, X_{j_n}, X_h$ e a substituição em Ψ de toda ocorrência livre da variável x_{j_i} pelo objeto $\bar{x}_{j_i}$,

para i=1,2,...,n, e de toda ocorrência livre da variável x_h pelo objeto $\bar{x}_h$, transforma o <predicado> Ψ em uma sentença verdadeira.

Logo uma ênupla de nós de GBD $(y_1, y_2, \ldots, y_n)$ pertence ao valor V se e somente se existe um objeto $\bar{x}_h$ em β (que pertence ao conjunto-objeto X_h) tal que a ênupla de objetos $(\bar{x}_{j_1}, \bar{x}_{j_2}, \ldots, \bar{x}_{j_n}, \bar{x}_h)$ satisfaz a ξ-expressão δ', onde $\bar{x}_{j_i}$ é o objeto representado por y_i, para i=1,2,...,n.

Conclui-se então que uma ênupla de nós de GBD $<y_1, y_2, \ldots, y_n>$ pertence ao valor V se e somente se a ênupla de objetos $(\bar{x}_{j_1}, \bar{x}_{j_2}, \ldots, \bar{x}_{j_n})$ por ela representada satisfaz a ξ-expressão δ.

Como os argumentos utilizados se aplicam a qualquer posição β de uma base de dados correspondente a Γ, conclui-se então que a consulta q é de fato ξ-equivalente a δ.

(iv) $\Pi = \forall x_h(\Psi)$

Conforme justificado no item anterior pode-se supor sem perda de generalidade que a variável x_h não aparece na lista-objetivo de δ.

Segue então que

$$\delta' = x_1:X_1, x_2:X_2, \ldots, x_m:X_m(x_{j_1}, x_{j_2}, \ldots, x_{j_n}, x_h):\Psi$$

é uma ξ-expressão, e, como o índice de Ψ é (N-1), pela hipótese da indução existe uma consulta

q' = get $v_1, v_2, \ldots, v_n, w$

where p

que é ξ-equivalente a δ'.

De modo análogo ao que foi feito no caso anterior pode-se verificar que a consulta

q = get $v_1, v_2, \ldots, v_n$

where (get w

where p) = (get w)

é ξ-equivalente a ξ-expressão δ. ⌟

Nota: Poderiam ter sido utilizados alguns resultados conhecidos da lógica formal sobre a equivalência de fórmulas para não tratar todos os conectivos e quantificadores, como usualmente é feito nesse tipo de demonstração. Optou-se por tratar todos para salientar a "implementação" de cada um deles na linguagem GORDAS, de modo que essa demonstração conduz a um algoritmo recursivo para conversão de uma ξ-expressão em uma consulta, o qual será exemplificado a seguir.

Exemplo:

Considerando o esquema ER representado parcialmente na figura II-A, pode-se construir a seguinte expressão do cálcu-

lo ER

$$\delta = x_1:\text{CLIENTE}, x_2:\text{PEDIDO}, x_3:\text{ITEM}(x_1, x_3)$$

$$\exists x_2(x_2.\text{CLIENTE} = x_1 \wedge x_2.\text{ITEM} = x_3$$

$$\wedge \; x_2.\text{QTD} > 1000) \;,$$

que corresponde à consulta "obtenha os pares de clientes e itens tais que o cliente tem um pedido de uma quantidade superior a 1000 unidades daquele item "

Com base na demonstração da proposição anterior pode-se construir uma consulta q na linguagem GORDAS (referente ao grafo-esquema representado na figura II-B) a qual é ξ-equivalente a δ.

Seguindo o critério adotado nos casos (i), (ii) e (iii) da base da indução as variáveis x_1, x_2, x_3 de δ serão implementadas na consulta q respectivamente através das variáveis CLIENTE.1 , PEDIDO.2 e ITEM.3.

Conforme o caso (iii) do passo da indução a consulta ξ-equivalente a δ é

q = get CLIENTE.1, ITEM.3

where count (get PEDIDO.2

where p_1) > 0,

onde p_1 é o predicado da consulta q_1 que é ξ-equivalente à ξ-expressão

$$\varepsilon_1 = x_1{:}\text{CLIENTE}, x_2{:}\text{PEDIDO}, x_3{:}\text{ITEM}(x_1, x_3, x_2):$$

$$x_2.\text{CLIENTE} = x_1 \wedge x_2.\text{ITEM} = x_3 \wedge x_2.\text{QTD} > 1000.$$

Segundo o caso (i) do passo da indução tem-se

q_1 = get CLIENTE.1, ITEM.3, PEDIDO.2

where (p_{11}) and (p_{12}) ,

onde p_{11} é o predicado da consulta q_{11} que é ξ-equivalente à ξ-expressão

$$\varepsilon_{11} = x_1{:}\text{CLIENTE}, x_2{:}\text{PEDIDO}, x_3{:}\text{ITEM}(x_1, x_3, x_2):$$

$$x_2.\text{CLIENTE} = x_1 \wedge x_2.\text{ITEM} = x_3 ,$$

e p_{12} é o predicado da consulta q_{12} que é ξ-equivalente à ξ-expressão

$$\varepsilon_{12} = x_1{:}\text{CLIENTE}, x_2{:}\text{PEDIDO}, x_3{:}\text{ITEM}(x_1, x_3, x_2): x_2.\text{QTD} > 1000$$

Ainda segundo o caso (i) do passo da indução tem-se

q_{11} = get CLIENTE.1, ITEM.3, PEDIDO.2

where (p_{111}) and (p_{112}) ,

onde p_{111} é o predicado da consulta q_{111} que é ξ-equivalente à ξ-expressão.

$$\varepsilon_{111} = x_1{:}\text{CLIENTE}, x_2{:}\text{PEDIDO}, x_3{:}\text{ITEM}(x_1, x_3, x_2): x_2.\text{CLIENTE} = x_1,$$

e p_{112} é o predicado da consulta q_{112} que é equivalente à ξ-expressão

$$\xi_{112} = x_1{:}\text{CLIENTE}, x_2{:}\text{PEDIDO}, x_2{:}\text{ITEM}(x_1, x_3, x_2) : x_2.\text{ITEM} = x_3.$$

Segundo o caso (i) da base da indução tem-se

q_{111} = get CLIENTE.1, ITEM.3, PEDIDO.2

where clientes of PEDIDO.2 = CLIENTE.1 e

q_{112} = get CLIENTE.1, ITEM.3, PEDIDO.2

where itens-pedidos of PEDIDO.2 = ITEM.3.

Segue então que

q_{11} = get CLIENTE.1, ITEM.3, PEDIDO.2

where (clientes of PEDIDO.2 = CLIENTE.1)

and (itens-pedidos of PEDIDO.2 = ITEM.3).

Segundo o caso (iii) da base da indução tem-se

q_{12} = get CLIENTE.1, ITEM.3, PEDIDO.2

where QTD of PEDIDO.2 > 1000, e segue então que

q_1 = get CLIENTE.1, ITEM.3, PEDIDO.2

where((clientes of PEDIDO.2 = CLIENTE.1) and

(itens-pedidos of PEDIDO.2 = ITEM.3))

and (QTD of PEDIDO.2 > 1000).

Conclui-se então que a consulta procurada é

```
q=  get CLIENTE.1,ITEM.3
      where count (get PEDIDO.2
                   where ((clientes of PEDIDO.2=CLIENTE.1) and
                          (itens-pedidos of PEDIDO.2 = ITEM.3))
                   and (QTD of PEDIDO.2 > 1000))
    > 0.
```

O processo utilizado para construção dessa consulta a partir da ξ-expressão pode ser generalizado, obtendo-se assim um algoritmo que transforma qualquer ξ-expressão na consulta ξ-equivalente. A definição desse algoritmo está fora dos objetivos deste trabalho.

Teorema:

A linguagem GORDAS tem a propriedade da completude ER (cap. III, def. 3.1).

Demonstração:

⌜Decorre trivialmente da proposição anterior, uma vez que toda expressão do Cálculo ER é uma ξ-expressão.⌟

Comparando as expressões do Cálculo ER com as consul

tas da linguagem GORDAS pode-se verificar uma característica interessante desta última.

Por exemplo, seja a consulta: "obtenha todos os ítens requeridos por clientes localizados na cidade de São Paulo". No Cálculo ER, essa consulta pode ser formulada pela expressão

$$x_1:\text{ITEM}, x_2:\text{PEDIDO}, x_3:\text{CLIENTE}, x_4:\text{CLI-CID}, x_5:\text{CIDADE}(x_1):$$

$$\exists x_5(x_5.\text{NOME} = \text{'SÃO PAULO'} \wedge$$

$$\exists x_2(\exists x_3(\exists x_4(x_2.\text{ITEM} = x_1 \wedge x_2.\text{CLIENTE} = x_3$$

$$\wedge\ x_4.\text{CLIENTE}=x_3 \wedge x_4.\text{CIDADE} = x_5)))).$$

Na linguagem GORDAS essa consulta pode ser formulada da seguinte maneira:

get ITEM

where cidade of clientes of ITEM includes 'SÃO PAULO'

Observa-se então que as expressões-de-caminho "escondem" variáveis e quantificadores existenciais.

Essa mesma consulta poderia também ser formulada da seguinte maneira:

get itens-pedidos of clientes of CIDADE

where NOME of CIDADE = 'SÃO PAULO'.

Comparando essa consulta com a expressão do Cálculo ER, observa-se que também as variáveis livres podem ser "escondidas" em expressões-de-caminhos.

CAPÍTULO V

Extensões

As seguintes linhas de pesquisa foram identificadas durante a elaboração deste trabalho:

1) Possibilidade e consequências da eliminação dos predicados de restrição da linguagem GORDAS (Cap. II, seção 7).

A consulta "obtenha os pares de nomes de clientes e fornecedores tais que o fornecedor possui em estoque todos os itens requeridos pelo cliente, em quantidades maiores que aquelas por ele requeridas" (referente ao diagrama ER da figura II-A) foi formulada através da linguagem GORDAS no Cap. II, seção 8, da seguinte maneira:

```
get  NOME of  CLIENTE, NOME of  FORNECEDOR
where (get  ITEM
        where (fornecedores of  ITEM includes  FORNECEDOR)
         and  (clientes of  ITEM includes  CLIENTE)
         and  (QTD of  fornecedores: (NOME = NOME of
                 FORNECEDOR) of  ITEM >
               QTD of  clientes: (NOME = NOME of
                 CLIENTE) of  ITEM))
       = itens-pedidos of  CLIENTE.
```

Essa mesma consulta pode ser formulada sem utilizar predicados de restrição, da seguinte maneira:

```
get NOME of CLIENTE, NOME of FORNECEDOR
where (get ITEM where
        count (get ESTOQUE where
                fornecedores of ESTOQUE = FORNECEDOR and
                itens-em-estoque of ESTOQUE = ITEM and
                count (get PEDIDO where
                        clientes of PEDIDO = CLIENTE and
                        itens-pedidos of PEDIDO = ITEM and
                        QTD of ESTOQUE > QTD of PEDIDO)
                    > 0 ) > 0 )
      = itens-pedidos of CLIENTE.
```

Essa segunda consulta é mais "complicada" que a anterior, e provavelmente menos eficiente, pois envolve duas consultas internas adicionais.

Como os predicados de restrição não foram necessários para a demonstração da Completude ER da linguagem GORDAS, conclui -se que o exemplo apresentado não é um caso isolado , pois para toda consulta na qual são utilizados predicados de restrição deve existir uma consulta equivalente (que obtém os mesmos dados) na qual os mesmos não são utilizados. Portanto , esses predicados poderiam ser eliminados da linguagem sem comprometer a sua "potência". Diante disso, uma questão interessante que se coloca é pesquisar o papel dos mesmos na simplificação e/ou otimização de consultas, conforme sugerido pelo

exemplo apresentado inicialmente.

Outra questão a ser pesquisada é a importância dos predicados de restrição para o agrupamento dos resultados de uma consulta, conforme sugerido pelos exemplos (c) e (d) apresentados no Cap. II, seção 8.

2) A "Completude relacional" da linguagem GORDAS.

Pode-se representar o modelo de uma base de dados relacional através de um grafo-esquema (Cap. II, seção 2), no qual não existem CR-nós nem arestas e cada CE-nó representa uma relação. Analogamente, a base de dados pode ser representada por um grafo-base-de-dados no qual não existem R-nós nem arestas, e os E-nós representam as ênuplas que constituem as relações.

Diante disso a linguagem GORDAS pode ser também imaginada como uma linguagem de consulta para o modelo relacional, e uma questão interessante que se coloca é investigar se ela possui a propriedade da "Completude relacional" (/CODD72/). É razoável esperar que ela tenha essa propriedade, já que ela tem a propriedade da completude ER e ambos os conceitos são baseados no Cálculo de Predicados.

Nessa mesma linha poderia ser interessante comparar a linguagem GORDAS "relacional" com as linguagens de consulta para o modelo relacional, tais como SEQUEL2 (/CHAM76/), QUEL (/SWKH76/), etc.

3) A definição da linguagem GORDAS sem a utilização de grafos.

Outra linha de pesquisa interessante seria tentar definir uma linguagem com a mesma estrutura que a linguagem GORDAS na qual as consultas fossem formuladas diretamente a partir do diagrama ER, e não de um grafo-esquema construído a partir dele.

Nessa linguagem os "caminhos" no diagrama ER seriam especificados através de sequências de nomes de conjunto-entidade e/ou conjuntos-relacionamento, ao invés das sequências de nomes de conexão. Nos casos de existência de dois "caminhos" entre um conjunto-entidade e um conjunto-relacionamento (como nos auto-relacionamentos, por exemplo) surgiriam ambiguidades, e para resolvê-las seria necessário que os papéis ("roles" em /CHEN76/) associados a cada participação de um conjunto-entidade em um conjunto-relacionamento fossem introduzidos no diagrama ER.

R E F E R Ê N C I A S

/ATVI81/ Atzeni, P., Villanelli, F. - "A Query and Manipulation Language for the Entity Relationship Model" (texto em Italiano, submetido à Conferência AICA, Itália, 1981).

/CHAM76/ Chamberlin, D.D. e outros - "SEQUEL2 : A Unified Approach to Data Definition, Manipulation, and Control", *IBM Journal of Research and Development*, Vol. 20, No. 6, November 1976.

/CHAT81/ Chen, P.P., Atzeni, P. - "Completeness of E-R Query Languages", *Entity-Relationship Approach to Information Modeling and Analysis* (ed. P.P. Chen), ER Institute, 1981.

/CHEN76) Chen, Peter Pin-Shan - "The Entity-Relationship Model -Toward a Unified View of Data", ACM *TODS*, Vol. 1 , No. 1, March 1976.

/CHIC82/ Chicca, Ricardo Junior - "Decisões Críticas em Projetos de Bancos de Dados", *Anais do II Simpósio Sobre Banco de Dados*, ed. SUCESU-SP, São Paulo, 1982.

/CODD72/ Codd, E.F. - "Relational Completeness of Data Base Sublanguages" - *Data Base Systems* (ed. R. Rustin), Prentice Hall, 1972.

/DATE77/ Date, C.J. - *An Introduction to Data Base Systems* 2nd ed., Addison Wesley, 1977.

/DEWE80/ Devor, C., Weeldreyer, J., "DDTS: A Testbed for Distributed Database Research", *Proceedings of the ACM Pacific 80 Conference*, San Francisco, California, 1980.

/ELMA81/ Elmasri, R., "GORDAS: A Data Definition, Query and Update Language for the Entity-Category-Relationship Model of Data", *Report* HR-81-250:17-38, Honeywell CCSC, Bloomington, Minnesota, 1981.

/ELWI81/ Elmasri, R., Wiederhold, G. - "GORDAS: A Formal High-Level Query Language", *Entity-Relationship Approach to Information Modeling and Analysis* (ed. P.P.Chen), ER Institute, 1981.

/HEGE73/ Hegenberg, Leônidas - *LÓGICA: O Cálculo de Predicados*, Herder, 1973.

/LUCC77/ Lucchesi, C. e outros - *Aspectos Teóricos da Computação*, 11º. Colóquio Brasileiro de Matemática, IMPA, 1977.

/POON79/ Poonen, G. - "CLEAR: A Conceptual Language for Entities and Relationships", reimpresso em *Tutorial: Centralized and Distributed Data Base Systems* (ed. W. Chu e P. P. Chen), IEEE, 1979.

/SHOS78/ Shoshani, Arie - *CABLE: A Language Based on the Entity-Relationship Model*, UCID-8005, Lawrence Berkeley Laboratory, 1978.

/SWKH76/ Stonebraker, M., Wong, E., Kreps, P., Held, G. - "The Design and Implementation of INGRES", *ACM TODS*, Vol. 1, No. 3, September 1976.

/WEBR81/ Webre, N.W. - "An Extended Entity-Relationship Model and its Use on a Defense Project", *Entity-Relationship Approach to Information Modeling and Analysis* (ed. P.P. Chen), ER Institute, 1981.