Metodologias para Conversão de Esquemas em Sistemas de Bancos de Dados Heterogêneos

Ronaldo Lopes de Oliveira
Geovane Cayres Magalhães

Relatório Técnico DCC–17/93

Julho de 1993

Metodologias para Conversão de Esquemas em Sistemas de Bancos de Dados Heterogêneos

Ronaldo Lopes de Oliveira *
Geovane Cayres Magalhães †

SUMÁRIO

Sistemas de Bancos de Dados Heterogêneos (SBDHs) são sistemas que integram, num ambiente cooperativo, sistemas de bancos de dados (SBDs) autônomos e heterogêneos entre si, com relação à semântica dos seus dados e/ou às características dos seus SGBDs (modelo de dados, linguagens de manipulação de dados e aspectos de implementação). Uma propriedade desejável nesses sistemas é a transparência de modelos, que permite ao usuário enxergar e manipular os dados localizados em diferentes SBDs, através do modelo e da linguagem de manipulação de dados que ele utilizava no seu SBD local, antes do mesmo ser incorporado ao SBDH. A propriedade em questão é conseguida através do mapeamento entre os dados e operações dos SBDs componentes. Este trabalho apresenta metodologias para conversão de esquemas em SBDHs construídos para integrar SBDs que utilizam o modelo de dados rede ou o modelo de dados relacional. Essas metodologias são importantes para obtenção da transparência de modelos.

*Departamento de Ciência da Computação, Universidade Estadual de Campinas, 13081-970 Campinas-SP. Pesquisa desenvolvida com suporte financeiro da Telegoiás - Telecomunicações de Goiás S/A

†Departamento de Ciência da Computação, Universidade Estadual de Campinas, 13081-970 Campinas-SP.

1 Introdução

A integração de aplicações construídas em Sistemas de Bancos de Dados (SBDs) independentes é uma necessidade cada vez maior nos dias atuais em que a disponibilidade de informações é um fator essencial para o sucesso das atividades humanas. Por isso, um grande investimento tem sido feito em pesquisas que estudam alternativas para a construção de sistemas que permitam o compartilhamento de informações entre SBDs isolados, preservando-se a autonomia dos mesmos. Esses sistemas são conhecidos na literatura como *Sistemas de Bancos de Dados Heterogêneos* (SBDHs) [HK89, EP90], *Sistemas de Bancos de Dados Federados* (SBDFs) [HM85, SL90] ou *Multidatabases* [LA86, HB91].

1.1 Transparência de Modelos em SBDHs

SBDHs são formados tipicamente pela integração de Sistemas de Bancos de Dados (SBDs componentes) que utilizam modelos e linguagens de manipulação diferentes entre si.

Dentro desse contexto, a transparência de modelos em SBDHs é a propriedade que garante a cada usuário a possibilidade de visualizar e manipular os dados do sistema global através de um único modelo de dados e da linguagem de manipulação (LMD) associada a esse modelo.

A propriedade em questão é obtida em duas etapas através do mapeamento entre os dados e operações dos SBDs componentes. A primeira etapa é tipicamente estrutural e objetiva converter esquemas expressos em um modelo de dados M1 para esquemas equivalentes em um modelo de dados M2. A segunda etapa é essencialmente operacional e utiliza os mapeamentos estruturais para permitir que operações construídas na LMD do modelo M2 sejam transformadas em operações equivalentes na LMD do modelo M1.

Várias abordagens podem ser utilizadas para mapear os modelos e linguagens existentes em um SBDH [Hsi92].

Uma primeira possibilidade é fazer um mapeamento direto (*um para um*) entre cada par de modelos e linguagens. Alguns trabalhos que

propõem soluções para a tradução direta entre modelos de dados são [Zan79, VL80, Lie81, Dem83, Lar83, TYF86].

Uma segunda alternativa é adotar um único modelo de dados (**modelo unificado**) para o qual todos os esquemas dos SBDs componentes são traduzidos. A seguir os esquemas obtidos são integrados em um ou mais esquemas virtuais globais. Os usuários do SBDH atuam diretamente sobre um esquema virtual global através da linguagem de manipulação de dados do modelo unificado, e as operações globais são transformadas em operações locais usando a linguagem de manipulação de dados disponível no SBD componente. Os mapeamentos nesse caso são do tipo *um para muitos*. Vários projetos e protótipos descritos na literatura de SBDHs utilizam esse tipo de mapeamento [Chu90, ADD+91, BDK+88, TBC+87, DL87, LA86, LR82].

Outra solução possível é adotar um modelo de dados comum (MDC) e uma linguagem de manipulação de dados intermediária. Assim os esquemas originais dos SBDs componentes (**esquemas locais**) são convertidos em esquemas no MDC (**esquemas componentes**) que são então integrados em um ou mais esquemas virtuais no mesmo modelo (**esquemas federados**). Os esquemas federados por sua vez são traduzidos para esquemas nos modelos de dados originais (**esquemas externos**). As operações dos usuários federados construídas sobre o esquema externo são transformadas em operações sobre o esquema federado usando a linguagem intermediária. As operações sobre o esquema federado são, a seu turno, transformadas em operações equivalentes sobre os esquemas locais dos SBDs componentes, usando a LMD local. Trata-se, portanto, de mapeamentos do tipo *muitos para muitos*. Cardenas [Car87] utiliza essa abordagem no projeto HD-DBMS.

Uma última abordagem possível é a tradução da representação dos dados dos SBDs componentes para um único modelo de dados de mais baixo nível e a criação de esquemas virtuais nos diferentes modelos de dados para os diversos tipos de usuários. Os mapeamentos nesse caso são do tipo *muitos para um*.

A terceira solução apresentada, ao nosso ver, é uma solução mais eficiente do que a primeira, mais transparente e flexível do que a segunda, e

mais realista do que a última. A primeira solução aumenta em demasia o número de mapeamentos necessários à medida que o número de modelos e linguagens diferentes cresce no SBDH. A segunda solução exige que o usuário passe a dominar um outro modelo de dados e uma nova LMD. A última solução supõe uma migração de dados que nem sempre é possível ou desejável.

1.2 Modelo de dados comum

O modelo de dados comum (MDC) é um peça fundamental em um SBDH, e através desse modelo é possível obter a transparência de modelos de dados e a integração dos SBDs componentes. Para que um modelo de dados possa se adequar perfeitamente ao papel de MDC, ele precisa apresentar as seguintes propriedades:

1. Facilidades para representar informações contidas em outros modelos, de forma simples e clara;

2. Capacidade de suportar diferentes níveis de abstração usados nos projetos dos diversos SBDs componentes;

3. Possibilidade de acrescentar informações relevantes não contidas nos esquemas dos SBDs componentes como, por exemplo, restrições de integridade implícitas ou mantidas pelas aplicações;

4. Disponibilidade de uma linguagem de definição e manipulação de dados bem definida e abrangente, que possa ser utilizada como linguagem intermediária no processo de transformação de operações sobre os modelos de dados existentes no SBDH.

Vários projetos de pesquisa em SBDHs [Chu90, BDK+88, LA86, LR82] utilizam como modelo de dados comum o modelo relacional. Todavia, o modelo relacional não possui todas as propriedades citadas anteriormente. A limitação na representação de relacionamentos, de restrições de integridade e de abstrações como generalização e especialização, torna

o modelo relacional inadequado para os processos de conversão e integração de esquemas.

Os chamados modelos semânticos [PM88, HK87] são, por outro lado, candidatos mais naturais para servirem como modelo de dados comum, devido à sua natureza tipicamente conceitual e ao seu poder de representação semântica. Apesar disso, poucos projetos em SBDHs [Car87, LR82] utilizam modelos semânticos como MDC.

Neste trabalho são apresentadas metodologias para conversão de esquemas locais de SBDs componentes nos modelos relacional e rede para esquemas componentes no modelo de dados comum (MDC). O MDC utilizado é o modelo Entidade-Categoria-Relacionamento (modelo ECR) [EWH85]. O modelo ECR é um modelo semântico que estende o modelo Entidade-Relacionamento proposto em [Che76]. As metodologias propostas permitem a uniformização da representação dos dados para integração dos esquemas. Essas metodologias, em conjunto com as metodologias para conversão de esquemas federados em esquemas externos contidas em [Oli93], suportam a parte estrutural dos mapeamentos necessários à obtenção da transparência de modelos em SBDHs.

A próxima seção descreve sucintamente o modelo ECR e a linguagem associada a ele. A seção 3 apresenta a metodologia proposta para conversão de esquemas locais no modelo rede em esquemas componentes no modelo ECR. A seção 4 descreve a metodologia para conversão de esquemas locais no modelo relacional em esquemas componentes no modelo ECR. A última seção analisa as metodologias propostas comparando-as com outros trabalhos correlatos.

2 O modelo ECR e a linguagem GORDAS

2.1 O modelo ECR

O Modelo ECR [EWH85] é um dos vários modelos propostos para estender a capacidade de modelagem do MER, mantendo-se as suas vantagens originais. Além de utilizar os conceitos de entidade e relacionamento do MER, o Modelo ECR introduz o conceito de categoria, que permite mo-

delar grupos lógicos de entidades pertencentes a tipos de entidades distintas mas que desempenham um mesmo papel em um relacionamento. Os conceitos e características principais do Modelo ECR serão descritos a seguir.

2.1.1 Entidade, Tipo de Entidade e Categoria

Os conceitos de **entidade** e **tipo de entidade** no Modelo ECR são similares aos conceitos de entidade e conjunto de entidades no MER. Os tipos de entidades são conjuntos disjuntos na medida que uma entidade só pode pertencer a um único tipo de entidade.

Uma **categoria** é um grupo de entidades pertencentes a um ou mais tipos de entidade que desempenham um mesmo papel (função lógica) em um relacionamento. Esse relacionamento pode estar explícita ou implicitamente representado no esquema ECR. Um exemplo de relacionamento implicitamente representado em uma categoria é o relacionamento de vínculo empregatício com uma empresa que existe nas entidades pertencentes à categoria *Empregado*. Em termos formais, uma categoria é uma relação matemática assim definida:

$$C_i = T_1[S_1] \text{ U } T_2[S_2] \text{ U } T_3[S_3] \ldots \text{U } T_n[S_n] \text{ onde:}$$
$$C_i = \text{categoria} : 1 \leq i \leq n;\ T_j = \text{tipo de entidade} : 1 \leq j \leq n;$$
$$S_j = \text{predicado de seleção que especifica as entidades de } T_j \text{ que farão parte da categoria;}$$

As categorias, ao contrário dos tipos de entidades, não são necessariamente disjuntas visto que uma entidade pode participar em várias categorias diferentes. Uma categoria pode, inclusive, ser definida em termos de outras categorias.

A introdução do conceito de categoria no Modelo ECR permite a modelagem explícita de duas abstrações importantes: generalizações e hierarquias ISA.

Tanto o conceito de tipo de entidade como o conceito de categoria são usados para representar conjuntos de entidades do mundo real. Observando-se a definição formal de categoria pode-se perceber que uma

categoria pode se confundir com um tipo de entidade (quando j = 1 e não existe predicado de seleção restritivo). Então é possível usar o conceito de categoria para se definir de forma mais abrangente o conceito de relacionamento.

2.1.2 Relacionamentos

Um relacionamento é uma associação de entidades. Formalmente um relacionamento R é um subconjunto do produto cartesiano de n categorias, não necessariamente distintas:

$$R \subseteq X(C_1, C_2, ..., C_n) = \{(e_1, e_2, ..., e_n): e_j \in C_j \text{ para } 1 \leq j \leq n\} \text{ onde:}$$
e_j = entidade; C_j = categoria; X = produto cartesiano.

Um elemento de R (uma tupla) é denominado de **instância do relacionamento**.

As propriedades estruturais de um relacionamento são as regras que governam as diversas maneiras que entidades de uma categoria podem participar em um relacionamento. O Modelo ECR especifica a participação de uma categoria C_j em um relacionamento R através de um par de números inteiros: (i_1, i_2), $0 \leq i_1 \leq i_2$ e $i_2 > 0$. Esses números indicam que cada entidade pertencente à categoria C_j, pode participar em, no mínimo, i_1 e em, no máximo, i_2 instâncias do relacionamento R. Utilizando-se esse par de números podem ser diferenciados os seguintes relacionamentos: total ($i_1 \geq 1$), parcial ($i_1 = 0$) e funcional ($i_2 = 1$).

Um relacionamento é **específico** em relação à participação de C_j se ele é total, e, se uma entidade de C_j, uma vez ligada a uma instância de R, não puder ser removida dessa instância a menos que a própria entidade seja excluída do banco de dados.

2.1.3 Atributos

Entidades e relacionamentos possuem atributos que descrevem suas características e propriedades. Um atributo a é definido formalmente como uma função com domínio definido em um tipo de entidade T, categoria C

ou relacionamento R e com imagem definida em um conjunto de valores (*valueset*) V ou no produto cartesiano de n conjuntos de valores:

$$a : T \rightarrow P(V) \text{ ou } a : C \rightarrow P(V) \text{ ou } a : R \rightarrow P(V) \text{ onde:}$$
P(V) é o conjunto potencial de valores;

Atributos definidos sobre um único conjunto de valores são chamados **atributos simples** e atributos definidos sobre o produto cartesiano de conjuntos de valores são chamados **atributos compostos**.

Restrições de cardinalidade e unicidade podem ser impostas sobre os atributos para representar certas situações do mundo real. A cardinalidade de um atributo determina o número de valores, obtidos de um conjunto de valores V, que podem ser associados a esse atributo. No Modelo ECR, a restrição de cardinalidade é especificada por um par de inteiros (i_1, i_2), onde $0 \leq i_1 \leq i_2$ e $i_2 > 0$. O número i_1 indica o número mínimo de valores e i_2 o número máximo.

Um atributo é **monovalorado** se $i_2 = 1$ e **multivalorado** se $i_2 > 1$. Um atributo é **total** se não lhe é permitido conter um valor nulo, isto é, se $i_1 \geq 1$. Caso contrário é um atributo **parcial**. O valor default para i_1 e i_2 é 1, definindo um atributo monovalorado e total.

A unicidade de atributos é usada para especificar que determinados atributos identificam univocamente uma entidade. Um atributo a é **único (ou chave)** se cada valor v de $P(V)$ aparece como um valor de a em no máximo uma entidade pertencente ao domínio, para cada estado válido do banco de dados. Ao contrário do MER, o Modelo ECR não exige que um tipo de entidade possua um conjunto de atributos chave já que entidades com atributos contendo o mesmo valor podem ser diferenciadas pelos seus relacionamentos com outras entidades.

2.1.4 Diagramas ECR

Assim como o MER, o Modelo ECR possui uma representação diagramática dos construtores utilizados na modelagem dos dados.

Tipos de entidade e relacionamentos são representados, respectivamente, por um retângulo e por um losângulo. Uma categoria é representada por um hexágono. Quando uma categoria é também um tipo de

entidade, o hexágono é sobreposto ao retângulo que representa o tipo de entidade.

Atributos são representados pelos seus nomes ligados por um arco[1] ao tipo de entidade, categoria ou relacionamento a que pertencem.

Os diferentes tipos de participação das categorias nos relacionamentos podem ser distinguidos nos diagramas ECR[2]. Relacionamentos sem restrição com relação à participação da categoria ($i_1 = 0$ e $i_2 = \infty$) são representados com uma linha simples entre a categoria e o relacionamento em que a categoria participa. Um relacionamento total é representado por uma linha com um ponto na extremidade que toca o losângulo que representa o relacionamento. Um relacionamento específico é representado por uma linha com um ponto em ambas as extremidades. Um relacionamento funcional é representado por uma linha com uma seta na extremidade em que está localizada a categoria que é funcionalmente determinada. Relacionamentos parcial e funcional, total e funcional e específico e funcional são representados usando combinações das convenções já citadas. A figura 1 ilustra a utilização dos símbolos em um diagrama ECR.

2.2 A linguagem GORDAS

A linguagem GORDAS[3] é utilizada como linguagem de definição e manipulação de dados no Modelo ECR.

A definição formal dessa linguagem é baseada na representação do modelo através de grafos orientados e rotulados [EW83]. Um esquema ECR é representado por um grafo em que cada vértice representa um tipo de entidade (vértice-TE), uma categoria (vértice-CA) ou um relacionamento (vértice-RE) e cada aresta A conecta um vértice-TE/CA a um

[1] Em [EWH85] os nomes dos atributos são envolvidos por formas ovais. Entretanto para tornar os diagramas menos "carregados" preferimos mudar ligeiramente essa representação.

[2] A representação dos relacionamentos apresentadas neste texto difere um pouco da representação usada em [EWH85] por questões de clareza e concisão dos diagramas

[3] acrônimo para Graph Oriented Data Selection

vértice-RE, indicando a participação do tipo de entidade ou categoria naquele relacionamento. O rótulo de cada vértice contém o nome do tipo de entidade ou categoria ou relacionamento representado, o tipo do vértice (RE, CA ou TE) e os atributos com seus respectivos *valuesets*. As arestas são rotuladas com dois nomes que servem para referenciar o relacionamento a partir do tipo de entidade/categoria (*nome-1*) e para referenciar o tipo de entidade/categoria a partir do relacionamento (*nome-2*). Além desses nomes, os rótulos das arestas contém dois números inteiros i_1 e i_2 que definem as restrições de cardinalidade e dependência de existência já descritas na seção 2.1.2.

Da mesma forma como um esquema ECR é representado pelo **grafo de esquema** (GE), um banco de dados ECR é definido como um grafo orientado e rotulado denominado **grafo de banco de dados** (GBD). O GBD é uma representação abstrata do banco de dados real formado por entidades individuais e instâncias dos relacionamentos. Portanto, o GBD é uma instância de um GE particular e existe uma correspondência entre os vértices e arestas dos dois grafos.

Uma operação GORDAS nada mais é que a definição de um caminho no GE que gera, a seu turno, caminhos no GBD. Para definição desses caminhos são fundamentais os nomes existentes nos rótulos das arestas porque eles estabelecem ligações tanto no sentido entidade-relacionamento como no sentido relacionamento-entidade. Os nomes dos rótulos das arestas são conhecidos no Modelo ECR como **nomes de conexão**. São os nomes de conexão que possibilitam a referência direta entre entidades pertencentes às diferentes categorias envolvidas nos relacionamentos.

A figura 1 mostra um diagrama ECR contendo os nomes de conexão associados aos relacionamentos. Os nomes de conexão podem ser vistos como nomes "invertidos" dos papéis que uma categoria desempenha em um relacionamento. Desse modo, o nome de conexão *empregados* associado ao relacionamento *Dep-Emp* permite uma referência do tipo *empregados of Departamento*. De maneira similar, o nome de conexão *departamento* permite referenciar entidades pertencentes ao tipo de entidade *Departamento* a partir de entidades do tipo de entidade *Empregado*, através da expressão *departamento of Empregado*.

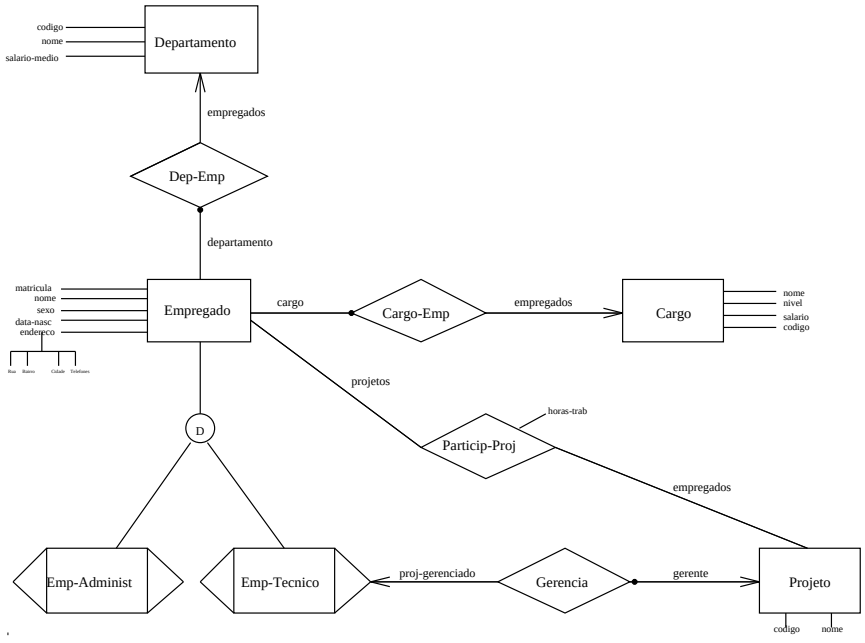

Figura 1: Diagrama ECR com nomes de conexão

A linguagem GORDAS permite a declaração de todos os construtores utilizados pelo Modelo ECR: conjunto de valores (*valueset*), tipo de entidade, categoria e relacionamento. A figura 2 exemplifica a definição de dados em GORDAS.

```
DEFINE VALUESET data-nasc AS DATE 1932:1978.

DEFINE ENTITY TYPE Empregado
       ATTRIBUTES
             matricula  VALUESET codigo  KEY,
             nome       VALUESET nome,
             data-nasc  VALUESET data-nasc,
             sexo       VALUESET sexo,
             endereco   COMPOUND
                            (rua       VALUESET nome,
                             bairro    VALUESET bairro,
                             cidade    VALUESET nome,
                             telefones VALUESET fone
                                         MINIMUM VALUES 0,
                                         MAXIMUM VALUES 3).

DEFINE  CATEGORY Emp-Tecnico FROM Empregado
  (codigo OF cargo OF Empregado INCLUDES {1111, 2222, 3333}).

DEFINE RELATIONSHIP Dep-Emp FROM
       Empregado (departamento, empregados)
                                  MINIMUM PARTICIPATION 1
                                  MAXIMUM PARTICIPATION 1,
       Departamento (empregados, departamento).
```

Figura 2: Exemplos de definição de dados em GORDAS

3 Conversão Rede-ECR

3.1 Considerações gerais

Uma metodologia para conversão Rede-ECR que possa ser utilizada no contexto de SBDHs deve levar em consideração cinco pontos importantes.

Primeiro, a separação que existe entre entidades e relacionamentos no modelo rede, que possui construtores distintos para modelar esses dois conceitos semânticos, embora em algumas situações específicas, como no caso de relacionamentos N:M, essa separação não aconteça plenamente.

Segundo, a forma como são representados os relacionamentos no esquema rede. O único tipo de relacionamento representado diretamente no modelo rede é o relacionamento 1:N. Outros tipos de relacionamento ou são garantidos nos programas de aplicação (relacionamento 1:1) ou são simulados por técnicas de projeto (relacionamentos N:M, relacionamentos não binários).

Terceiro, as restrições impostas à participação dos registros membros em um *set-type*. Essas restrições devem ser consideradas na definição do tipo de participação das entidades nos relacionamentos representados no esquema ECR.

Quarto, a presença no esquema rede de informações referentes à organização física dos dados. Essas informações não podem ser transferidas para o esquema ECR mas precisam ser mantidas localmente para que possam ser utilizadas no processo de transformação de operações GORDAS em operações sobre o esquema rede.

Quinto e último, a maior riqueza semântica do modelo ECR que implica na possibilidade de inclusão no esquema ECR de informações que não podem ser obtidas diretamente do esquema rede e devem ser supridas pelo usuário, no caso o DBA local.

3.2 Trabalhos correlatos

São poucos os trabalhos na literatura que apresentam metodologias para conversão do modelo rede para modelos baseados no modelo entidade-relacionamento (MER).

Dumpala e Arora [DA83] propuseram um algoritmo bastante limitado para converter diagramas rede em diagramas ER. O algoritmo baseia-se quase que exclusivamente na correspondência entre *record-type* e tipo de entidade e entre *set-type* e relacionamento. Entretanto, essa correspondência não é sempre verdadeira devido à presença dos registros conectores que simulam relacionamentos M:N, relacionamentos de grau maior do que dois e auto-relacionamentos.

Fong [Fon92] apresenta uma metodologia para conversão de esquemas para ser usada no caso de uma migração de dados de um SBD rede ou de um SBD hierárquico para um SBD relacional. A metodologia usa um modelo entidade-relacionamento estendido para capturar a semântica do esquema rede original, particularmente com relação as dependências funcionais e as restrições de integridade referenciais existentes entre os *record-type*. As dependências funcionais são obtidas pelo exame dos itens de dados dos *record-type*, e as restrições de integridade referenciais através do exame do tipo de retenção e inserção dos *set-type* em que os *record-type* estão envolvidos. Durante todo o processo de conversão o usuário pode intervir para validar ou modificar os resultados obtidos. A representação no modelo entidade-relacionamento estendido é traduzida para um esquema relacional em que são mantidas as restrições estruturais do modelo original. Nesse esquema resultante todas as relações bases estão pelo menos na terceira forma normal.

3.3 Equivalência de Construtores

A conversão de esquemas locais dos SBDs componentes representados no modelo rede para esquemas componentes no modelo de dados comum (modelo ECR) pode ser feita baseada na correspondência entre os construtores dos dois modelos. A seguir serão analisadas essas correspondências.

3.3.1 Record-type e Set-type

De maneira geral, os *record-type* do modelo rede são equivalentes aos tipos de entidade no modelo ECR e os *set-type* são equivalentes a re-

lacionamentos binários funcionais (1:N) entre o tipo de entidade que representa o *record-type* mestre e o tipo de entidade que representa o *record-type* membro de um *set-type*.

Contudo, certos tipos de relacionamentos são modelados usando registros conectores: relacionamentos N:M, relacionamentos recursivos e relacionamentos que envolvem mais de dois tipos de entidades. Os conectores, apesar de serem representados por *record-type* no modelo rede, devem ser traduzidos para relacionamentos no modelo ECR, como ilustra a figura 3.

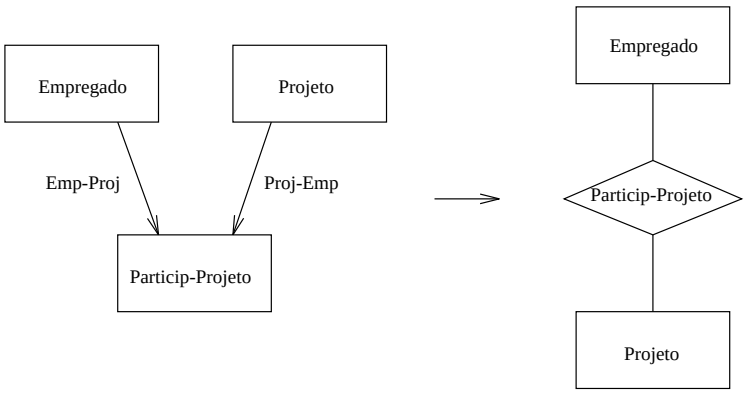

Figura 3: Mapeamento de relacionamento N:M no modelo rede para o modelo ECR

Uma metodologia para tradução de esquemas rede para esquemas ECR deverá incluir um algoritmo para identificar esses possíveis conectores. Sabe-se que os *record-type* conectores são membros em mais de um *set-type* e ao mesmo tempo não podem ser o *record-type* mestre em nenhum *set-type*, já que nesse caso estariam fortemente caracterizados como um tipo de entidade propriamente dito e não como uma estrutura auxiliar. Contudo, pode ser que um *record-type* satisfaça essas condições e ainda assim não represente um conector. Logo, os candidatos a conectores devem ser apresentados ao usuário (DBA local) para que ele os confirme ou não como tal.

3.3.2 Set Membership (Set Insertion e Set Retention)

Na descrição de um *set-type* em um esquema rede são definidas algumas restrições em relação à participação de seus registros membros. São declaradas restrições quanto à inserção (AUTOMATIC ou MANUAL) e quanto à retenção desses registros (*FIXED*, *MANDATORY*, *OPTIONAL*). Um tipo de retenção *FIXED* ou *MANDATORY* é geralmente acompanhada de um tipo de inserção AUTOMATIC e um tipo de retenção *OPTIONAL* é normalmente acompanhada de um tipo de inserção MANUAL.

A retenção *FIXED* caracteriza um tipo de relacionamento *específico* entre as entidades que representam registros mestres e membros, e a retenção *MANDATORY* caracteriza um tipo de relacionamento *total* entre essas entidades. Ambos as retenções indicam que, para todo estado válido do banco de dados, a entidade que representa o registro membro deve estar obrigatoriamente ligada por uma instância desse relacionamento a uma entidade que representa o registro mestre. Logo, a cardinalidade mínima do relacionamento em relação à participação das entidades que representam os registros membros deverá ser em ambos os casos igual a 1. No caso do relacionamento específico, além dessa restrição, é imposta uma outra que proíbe a transferência de uma entidade que representa um registro membro para outra instância do relacionamento.

Por outro lado, o tipo de retenção *OPTIONAL* caracteriza um tipo de relacionamento *parcial* das entidades que representam os registros membros com as entidades que representam os registros mestres. Assim, uma entidade que representa um registro membro poderá existir em um estado válido do banco de dados sem estar incluída em uma instância do relacionamento. Nesse caso, a cardinalidade mínima do relacionamento em relação à participação das entidades que representam os registros membros deverá ser igual a 0.

Para todos os tipos de retenção a cardinalidade máxima do relacionamento em relação à participação do tipo de entidade que representa o *record-type* membro será igual a N, a menos que, durante o processo

de tradução, o DBA local informe que a cardinalidade máxima será outra. Esse tratamento de exceção se justifica porque no modelo rede pressupõe-se que os relacionamentos são todos 1:N e outros tipos de relacionamentos, como por exemplo 1:1, só podem ser estabelecidos através de restrições impostas sobre o valor de N nos programas de aplicação.

Se o relacionamento modelado for funcional (1:N), a cardinalidade mínima e máxima do relacionamento em relação à participação do tipo de entidade que representa o *record-type* mestre são iguais a 0 e 1, respectivamente. Se o relacionamento for não funcional (N:M), a cardinalidade máxima é igual a N.

3.3.3 Itens de dados

Os itens de dados descritos na definição dos *record-type* no modelo rede são equivalentes aos atributos dos tipos de entidade e relacionamentos no modelo ECR. Os itens de dados dos *record-type* que foram convertidos em tipos de entidade ou categorias no modelo ECR, são neles incluídos. Os itens de dados pertencentes aos conectores no esquema rede são incluídos nos relacionamentos que esses conectores geraram no esquema ECR.

Cada atributo no modelo ECR tem um conjunto de valores associados (*valueset*) que podem ser obtidos a partir da descrição do tipo e do tamanho dos itens de dados no esquema rede. Os itens de dados compostos ou multivalorados podem ser representados diretamente no modelo ECR. A cardinalidade mínima e máxima dos atributos monovalorados são iguais a 1 (um). A cardinalidade mínima e máxima dos atributos multivalorados são iguais a 1 e a N, respectivamente, onde N é o número de ocorrência definido na declaração dos itens de dados no esquema rede.

No modelo ECR, os tipos de entidade não precisam apresentar atributos identificadores (chaves) uma vez que as entidades com mesmos conteúdos de atributos podem ser distinguidas pelos relacionamentos que possuem com outras entidades. Entretanto, alguns atributos identificadores podem ser inferidos a partir do esquema rede. Por exemplo, os registros que têm a cláusula *LOCATION MODE CALC DUPLICATES NOT ALLOWED* podem ser identificados univocamente pelos atri-

butos que representam os itens de dados contidos nessa cláusula. Da mesma forma, os registros membros de um *set-type* que são ordenados pela cláusula *SORTED BY DEFINED KEY DUPLICATES NOT ALLOWED* podem ser identificados univocamente pela concatenação dos atributos definidos nessa cláusula com o(s) atributo(s) que identifica(m) o seu mestre. O valor da *DATABASE-KEY* que em um banco de dados rede identifica univocamente cada registro, não pode ser usada como identificador no modelo ECR porque esse valor expressa unicamente a identidade física do registro e, portanto, não tem utilidade no nível lógico (conceitual) em que o modelo de dados comum é utilizado.

3.3.4 Cláusulas referentes à organização física dos dados

Os esquemas no modelo rede trazem algumas informações relativas às particularidades da organização física dos dados, como localização física (*LOCATION MODE*), métodos para identificação de uma ocorrência de um *set-type* (*SET SELECTION*), partições lógicas de registros (*AREA*) e ordenamento dos registros membros dentro de um *set-type* (*SET ORDER*).

Essas informações são relevantes para a definição de estratégias de recuperação e armazenamento dos dados e por conseguinte são importantes no processamento das operações nesse modelo. Porém, o modelo ECR, enquanto modelo conceitual, não possui construtores para modelar essa características físicas. Por esse motivo, essas informações devem ser armazenadas em um repositório de dados local que pode ser um dicionário de dados (DD) ou um banco de dados auxiliar. As informações nele contidas serão utilizadas pelo processador responsável pela transformação de operações expressas no modelo ECR em operações equivalentes no modelo rede. Além disso, essas informações precisam ser armazenadas a nível global para que seja possível criar esquemas externos no modelo rede a partir de esquemas federados no modelo ECR.

3.3.5 Construtores específicos do modelo ECR

Alguns construtores importantes utilizados no modelo ECR não possuem equivalentes no modelo rede e são usados para modelar explicitamente conceitos semânticos importantes. Entre esses construtores destacam-se as categorias (de generalização e de especialização) e os nomes de conexão, discutidos no capítulo 2 dessa dissertação.

As abstrações de generalização e especialização são modeladas de forma intuitiva no modelo rede. Considere, por exemplo, o *set-type Pessoa-Empreg* da figura 4 que associa *Pessoa* e *Empregado*. Pode ser que a intenção do projetista seja a de representar o fato de que todo empregado é uma pessoa, isto é, que *Empregado* é uma subclasse de *Pessoa*. Da mesma forma, o *set-type Tipo-Empreg* da mesma figura pode sugerir que *Empregado* seja uma generalização de *Empregado Técnico* e *Empregado Administrativo*. O problema é que a intenção do projetista não pode ser percebida através de um exame objetivo do esquema rede. Fica evidente a necessidade de intervenção humana para a inclusão de categorias no esquema componente ECR.

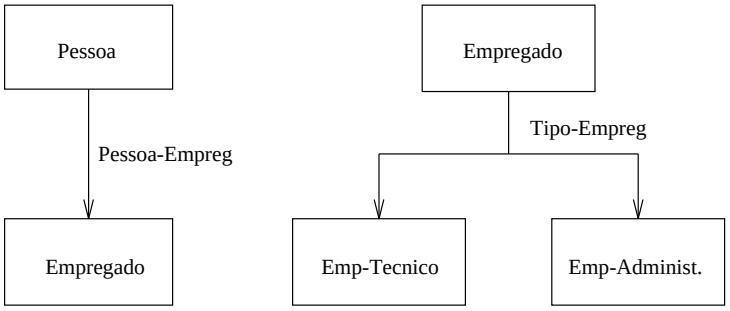

Figura 4: Exemplos de Set-type simulando relacionamento ISA e generalização

Contudo, a inclusão de categorias no esquema componente, embora desejável do ponto de vista da clareza de representação não é uma boa alternativa no contexto de SBDHs pois modifica sensivelmente a organização original dos dados no esquema local, aumentando o número de

mapeamentos necessários entre os construtores nos vários níveis da arquitetura do SBDH. Conseqüentemente os processos de tradução de esquemas e, principalmente, de transformação de operações se torna muito mais complexo.

Os nomes de conexão são usados no modelo ECR para permitir que se façam referências às entidades de uma categoria a partir de uma outra entidade que está relacionada com as primeiras através de um determinado relacionamento. Esses nomes de conexão estabelecem, de certa maneira, caminhos de navegação entre os registros no modelo rede.

Os nomes de conexão não podem ser obtidos das estruturas existentes no esquema rede e, portanto, devem ser supridos pelo DBA local que deverá ter o cuidado de escolher nomes significativos para tornar mais claras as referências entre as entidades. As correspondências entre os nomes dos *set-type* e os nomes de conexão deverão ser guardadas no DD local para serem usadas no processo de transformação de operações GORDAS em operações na LMD rede.

3.3.6 Restrições de integridade explícitas

O modelo rede provê, pelo menos em tese, um mecanismo que pode ser utilizado para a verificação de restrições de integridade semânticas. Esse mecanismo é baseado em procedimentos especiais, conhecidos como *database procedures* [TF76], que são invocados sempre que determinadas condições sobre os dados não são satisfeitas.

Na verdade, a maioria dos SGBDs rede comercialmente disponíveis não suporta esse tipo de facilidade, deixando à cargo dos programas de aplicação a manutenção das restrições de integridade que não são intrínsecas ao modelo rede.

De qualquer forma, as restrições semânticas podem ser incorporadas ao esquema componente utilizando-se os recursos providos pela linguagem GORDAS.

3.4 Metodologia proposta

Nas seção anterior foram feitas várias considerações sobre a tradução de esquemas rede para esquemas ECR. Foram discutidas, em particular, as correspondências entre os construtores utilizados nos dois modelos e a interação do DBA local no processo de tradução para tornar possível a inclusão no esquema alvo de informações que não podem ser inferidas diretamente do esquema original.

Baseado nessas discussões, pode-se então sugerir uma metodologia para a tradução Rede-ECR composta do seguintes passos:

1. Exame dos *record-type* identificando tipos de entidades provisórios;
2. Exame dos *set-type* identificando relacionamentos provisórios;
3. Identificação dos conectores que representam relacionamentos N:M e relacionamentos recursivos;
4. Definição dos nomes de conexão;
5. Refinamento do esquema ECR.

Cada um desses passos será discutido a seguir.

3.4.1 Exame dos record-type

Cada *record-type* deverá dar origem a um tipo de entidade provisória. Esse mapeamento é provisório porque alguns dos *record-type* podem representar conectores que são convertidos para relacionamentos no modelo ECR. Porém, só será possível identificar exatamente o conceito semântico representado por um *record-type* ao serem examinados os *set-type* em que ele participa.

Os itens de dados dos *record-type* serão transformados em atributos dos tipos de entidade gerados, juntamente com seus respectivos *valueset*, que serão definidos no esquema ECR com o mesmo nome dos atributos e com as mesmas características de tipo e tamanho descritas nos itens de dados correspondentes do esquema rede.

As informações relativas às cláusulas *LOCATION MODE* e *AREA* deverão ser armazenadas no DD local para que possam ser utilizadas na transformação de operações GORDAS para operações rede interpretáveis pelo SGBD local.

As figuras 6 e 7 mostram, respectivamente, o trecho do esquema ECR gerado (expresso em GORDAS) a partir da definição do *record-type Empregado* da figura 5 e a entrada do DD local incluída nessa etapa da metodologia.

```
RECORD Empregado
LOCATION MODE CALC USING matricula
DUPLICATES NOT ALLOWED WITHIN Area-Pes.

02 matricula         type is decimal 6.
02 nome              type is char    40.
02 sexo              type is char    1.
02 data-nasc         type is decimal 8.
02 endereco.
   03 rua            type is char    30.
   03 bairro         type is char    40.
   03 cidade         type is char    40.
   03 telefones      type is decimal 7  OCCURS 3.
```

Figura 5: Definição de *record-type*

3.4.2 Exame dos set-type

Para se identificar quais dentre os *record-type* existentes no esquema rede representam conectores é preciso examinar todos os *set-type* do esquema, porque um *record-type* só é considerado conector se ele participa como *record-type* membro em mais de um *set-type* e não participa como *record-type* mestre em nenhum outro *set-type*.

Entretanto, deve-se gerar para cada *set-type* examinado um relacionamento binário 1:N provisório. Dessa forma, evita-se um novo exame

```
DEFINE VALUESET matricula AS INTEGER RANGE 0:999999.
DEFINE VALUESET nome      AS STRING  40.
                .
                .
                .
DEFINE VALUESET fone      AS INTEGER RANGE 0:9999999.

DEFINE ENTITY-TYPE Empregado
       ATTRIBUTES
            matricula   VALUESET matricula KEY,
            nome        VALUESET nome,
            sexo        VALUESET sexo,
            data-nasc   VALUESET data-nasc,
            endereco    COMPOUND
                         (rua       VALUESET rua,
                          bairro    VALUESET bairro,
                          cidade    VALUESET nome,
                          telefones VALUESET fone
                                    MINIMUM VALUES 0
                                    MAXIMUM VALUES 3)
```

Figura 6: Trecho do esquema ECR obtido no passo 1 da metodologia Rede-ECR

Entrada	Tipo	Construtor Rede	Location Mode	Area
Empregado	entidade	record-type	calc matricula	A1

Figura 7: Mapeamento gerado no passo 1 da metodologia Rede-ECR

dos *set-type* para gerar esses relacionamentos após a identificação dos conectores.

O tipo de relacionamento (específico, total ou parcial) é definido pelo número de participações mínima e máxima das entidades nesse relacionamento, que podem ser obtidos através do exame das cláusulas de inserção e retenção dos *record-type* membros, como foi mostrado na seção 3.3.2. A participação do tipo de entidade que representa o *record-type* mestre é assumida como não funcional (caracterizando um relacionamento 1:N). Contudo essa cardinalidade poderá ser alterada no passo de refinamento do esquema ECR (passo 5).

Para os *set-type* com mais de um *record-type* membro são criados relacionamentos binários 1:N entre o *record-type* mestre e cada um dos *record-type* membros. O nome de cada relacionamento é formado pelo nome do *set-type* original concatenado com um sufixo diferenciador que pode ser, por exemplo, um número inteiro seqüencial e crescente. Todos os *set-type* criados são mapeados para o *set-type* original para que as operações sobre o esquema componente possam ser transformadas em operações usando os construtores do esquema local.

As informações contidas nas cláusulas *SET SELECTION* e *SET ORDER* de cada *set-type* examinado devem ser armazenadas no DD local para serem usadas na transformação de operações. Além disso, para cada *record-type* devem ser guardados os *set-type* em que ele participa como membro e os *set-type* em que ele participa como mestre. Essas informações serão usadas no próximo passo da metodologia.

A figura 9 mostra o trecho do esquema ECR obtido pela aplicação do passo 2 sobre a definição dos *set-type* da figura 8. A figura 10 mostra o mapeamento gerado no DD local.

3.4.3 Identificar conectores

A identificação dos possíveis conectores é feita examinando-se as entradas do DD local que associam cada *record-type* com os *set-type* em que ele participa.

Os candidatos a conectores são apresentados ao DBA local que deve

```
SET Gerente-Projeto
OWNER IS Projeto
ORDER IS BY SYSTEM DEFAULT
MEMBER IS Empregado
RETENTION IS OPTIONAL
INSERTION IS MANUAL
SET SELECTION IS BY VALUE OF
               cod-projeto IN Projeto

SET Dep-Emp
OWNER IS Departamento
ORDER IS LAST
MEMBER IS Empregado
RETENTION IS MANDATORY
INSERTION IS AUTOMATIC
SET SELECTION IS BY VALUE OF
               cod-depto IN Departamento

SET Emp-Proj
OWNER IS Empregado
ORDER IS NEXT
MEMBER IS Particip-Projeto
RETENTION IS MANDATORY
INSERTION IS AUTOMATIC
SET SELECTION IS BY APPLICATION

SET Proj-Emp
OWNER IS Projeto
ORDER IS NEXT
MEMBER IS Particip-Projeto
RETENTION IS MANDATORY
INSERTION IS AUTOMATIC
SET SELECTION IS BY APPLICATION
```

Figura 8: Definição de *set-type*

```
DEFINE RELATIONSHIP Dep-Emp FROM
                    Departamento,
                    Empregado        MINIMUM PARTICIPATION 1,
                                     MAXIMUM PARTICIPATION 1.
DEFINE RELATIONSHIP Emp-Proj FROM
                    Empregado,
                    Particip-projeto MINIMUM PARTICIPATION 1,
                                     MAXIMUM PARTICIPATION 1.
DEFINE RELATIONSHIP Proj-Emp FROM
                    Projeto,
                    Particip-projeto MINIMUM PARTICIPATION 1,
                                     MAXIMUM PARTICIPATION 1.
DEFINE RELATIONSHIP Gerente-Proj FROM
                    Projeto,
                    Empregado        MINIMUM PARTICIPATION 0,
                                     MAXIMUM PARTICIPATION 1.
```

Figura 9: Trecho do esquema obtido no passo 2 da metodologia Rede-ECR

Entrada	Tipo	Const. Rede	Location Mode	Mestre nos sets	Membro nos sets
Empregado	entidade (TE)	Rec-type (RT)	Calc (matricula)	Emp-Proj,	Dep-Emp, Gerente-Proj,
Departamento	TE	RT	Calc (cod-depto)	Dep-Emp	
Projeto	TE	RT	Calc (cod-projeto)	Gerente-Proj, Proj-Emp	
Particip-projeto	TE	RT	Via (Emp-Proj)		Emp-Proj, Proj-Emp

Entrada	Tipo	Construtor Rede	Set Selection	Set Order
Dep-Emp	Relac.bin (R1)	set-type	value of (cod-depto)	Last
Emp-Proj	R1	set-type	Application	Next
Proj-Emp	R1	set-type	Application	Next
Gerente-Proj	R1	set-type	Application	System default

Figura 10: Mapeamentos gerados no passo 2 da metodologia Rede-ECR

determinar se o *record-type* em questão é um conector ou não. Se o *record-type* não for considerado um conector nenhuma ação é executada, porque os relacionamentos definidos pelos *set-type* em que ele participa já foram identificados e declarados no passo anterior da metodologia.

Se um *record-type* é identificado como conector, então o próximo passo é determinar se esse conector está sendo usado para representar um relacionamento N:M ou um auto-relacionamento. Se os *record-type* mestres dos *set-type* em que um conector participa como membro são distintos, fica caracterizado um relacionamento N:M entre esses *record-type* mestres. No caso dos *record-type* mestres serem idênticos fica caracterizado um auto-relacionamento entre eles.

As declarações dos tipos de entidade que representavam os registros conectores no esquema ECR provisório devem ser substituídas pelas declarações dos relacionamentos que eles realmente representam. Os atributos do tipo de entidade passam a ser atributos do relacionamento. A correspondência entre os *set-type* originais e os relacionamentos gerados

deve ser guardada no DD local. A figura 11 apresenta as modificações introduzidas no esquema ECR ocasionadas pela identificação do conector *Particip-projeto* e a figura 12 apresenta as modificações introduzidas nos mapeamentos armazenados no DD local.

```
DEFINE  ENTITY-TYPE  Particip-Projeto...   ---> Excluido
DEFINE  RELATIONSHIP Emp-Proj  FROM...     ---> Excluido
DEFINE  RELATIONSHIP Proj-Emp  FROM...     ---> Excluido

DEFINE  RELATIONSHIP Particip-Projeto FROM  |
                     Empregado              |---> Incluido
                     Projeto.               |
```

Figura 11: Modificações no esquema feitas no passo 3 da metodologia Rede-ECR

Entrada	Tipo	Const. Rede	Location Mode	Area	Mestre nos sets	Membro nos sets
Particip-Projeto	relac.N:M (R2)	RT	Via (Emp-proj)	Area-Pes		Emp-Proj, Proj-Emp

Entrada	Tipo	Construtor Rede	Set Selection	Set Order
Emp-Proj		set-type	Application	
Proj-Emp		set-type	Application	

Figura 12: Mapeamentos modificados no passo 3 da metodologia Rede-ECR

3.4.4 Definir nomes de conexão

A declaração dos relacionamentos feita nos passos 2 e 3 da metodologia ainda não está completa porque falta incluir os nomes de conexão

para cada tipo de entidade participante. Como foi visto na seção 2.2, cada participação de uma classe[4] em um relacionamento está associada a dois nomes de conexão, *nome-conexão-1* e *nome-conexão-2*, que determinam um caminho da classe para o relacionamento e um caminho do relacionamento para a classe, respectivamente. Os nomes de conexão possibilitam que as classes participantes em um relacionamento se referenciem umas às outras. No caso de relacionamentos binários, o *nome-conexão-1* permite referenciar não só o relacionamento e seus eventuais atributos mas também a outra classe participante. Para um relacionamento que associa mais de duas classes, a referência à uma classe B a partir de uma classe A é feita através do caminho definido pela concatenação do *nome-conexão-2* da classe B com o *nome-conexão-1* de A. Nos relacionamentos recursivos, a referência às entidades de uma classe, que participam do relacionamento desempenhando um papel Y, a partir de uma entidade da mesma classe que desempenha um papel X, é feita utilizando o *nome-conexão-1* associado ao tipo de participação X.

A responsabilidade da definição dos nomes de conexão é do DBA local juntamente com o DBA do sistema global. Para facilitar essa tarefa seria interessante prover uma visão gráfica do esquema ECR construído até o passo 3 da metodologia.

As correspondências entre os caminhos de busca de dados definidos pelos nomes de conexão no esquema ECR e os caminhos de navegação no esquema rede deverão ser guardados no DD local para que sejam utilizados durante o processo de transformação de operações GORDAS em operações sobre o esquema rede. Esses mapeamentos podem ser obtidos examinando-se os relacionamentos gerados no esquema ECR.

Seja $R = \{R_1, R_2, \cdots, R_n\}$ o conjunto dos relacionamentos gerados no esquema ECR.

Seja $R_i = \{(E_1, na_1, nb_1), (E_2, na_2, nb_2), \cdots, (E_n, na_n, nb_n)\}$, $1 \leq i \leq n$, um relacionamento qualquer de R que associa as classes $E_1, E_2, \cdots, E_n$ através dos nomes de conexão na_i e nb_i.

Sejam na_j o *nome-conexão-1* e nb_j o *nome-conexão-2* definidos para

[4] Uma classe é definida aqui como um conjunto de entidades, que pode ser tanto um tipo de entidade com uma categoria

a classe E_j, $1 \leq j \leq n$, que participa do relacionamento R_i. Para cada E_j devem ser gerados os seguintes mapeamentos:

1. Um mapeamento do caminho definido pela expressão $<of\ E_j>$ para o *record-type* correspondente a E_j;

2. Um mapeamento do caminho definido pela expressão $<na_j\ of\ E_j>$ para o caminho definido pelo *set-type* S correspondente a R_i, se R_i é um relacionamento 1:N. O caminho $E_j \rightarrow S$ pode ter duas formas: (*mestre-S* $\rightarrow$ *membro-S*) ou (*membro-S* $\rightarrow$ *mestre-S*) conforme E_j seja mestre ou membro de S;

3. Um mapeamento do caminho definido pela expressão $<na_j\ of\ E_j>$ para o caminho definido pelo *set-type* T que tem como membro o *record-type* correspondente a R_i e como mestre o *record-type* correspondente a E_j, se R_i é um relacionamento N:M ou um relacionamento de grau maior do que dois. O caminho definido pelo *set-type* T é da forma: (*mestre-T* $\rightarrow$ *membro-T*). Se o relacionamento R_i é binário deve ser gerado um mapeamento adicional da expressão $<na_j\ of\ E_j>$ para o *record-type* correspondente à outra classe $E_k, k \neq j$, do relacionamento. Esse mapeamento é da forma (*mestre-T* $\rightarrow$ *membro-T/membro-U* $\rightarrow$ *mestre-U*) onde U é o *set-type* que tem o *record-type* correspondente a E_k como mestre e o *record-type* correspondente a R_i como membro.

4. Um mapeamento do caminho definido pela expressão $<na_j\ of\ E_j>$ para o caminho que vai do *record-type* que corresponde a E_j até o *record-type* que corresponde a R_i através do *set-type* V que liga R_i às entidades de E_j que desempenham um determinado papel P_1, se R_i é um relacionamento recursivo. O caminho correspondente no esquema rede tem a forma (*mestre-V* $\rightarrow$ *membro-V*). Deve ser gerado também um mapeamento adicional do caminho definido pela expressão $<na_j\ of\ E_j>$ para o caminho que vai do *record-type* correspondente a E_j até ele mesmo, passando pelo *set-type* V e pelo *set-type* W que liga R_i às entidades de E_j que representam o

papel P_2 no relacionamento. Esse caminho tem a forma (*mestre-V* $\rightarrow$ *membro-V/membro-W* $\rightarrow$ *mestre-W*).

5. Um mapeamento do caminho definido por cada expressão $<nb_k$ *of* na_j *of* $E_j>$, $1 \leq k \leq n$, $k \neq j$, para o *record-type* correspondente a E_k, se R_i é um relacionamento que envolve mais de duas classes. O caminho no esquema rede tem a forma (*mestre-X* $\rightarrow$ *membro-X/membro-Y* $\rightarrow$ *mestre-Y*), onde X é o *set-type* que tem como mestre o *record-type* correspondente a E_j e como membro o *record-type* correspondente a R_i e Y é o *set-type* que que tem como mestre o *record-type* correspondente a E_k e como membro o *record-type* correspondente a R_i.

A figura 13 mostra a definição de alguns dos relacionamentos das figura 9 e 11 já com os nomes de conexão incluídos. A figura 14 mostra os mapeamentos que serão guardados no DD local. O leitor deve perceber que os mapeamentos de caminhos como *<projetos of empregados of Departamento>*, embora não apareçam explicitamente no DD local, podem ser facilmente obtidos com a substituição da subexpressão *<empregados of Departamento>* pelo *record-type* equivalente, no caso *Empregado*.

```
DEFINE RELATIONSHIP Dep-Emp FROM
                    Departamento (empregados, departamento),
                    Empregado    (departamento, empregados)
                                 MINIMUM PARTICIPATION 1,
                                 MAXIMUM PARTICIPATION 1.
DEFINE RELATIONSHIP Particip-projeto FROM
                    Projeto      (empregados, projetos),
                    Empregado    (projetos, empregados).
```

Figura 13: Nomes de conexão incluídos no passo 4 da metodologia Rede-ECR

Caminho ECR	Caminho rede 1	Caminho rede 2	Record-type alcançado
OF Departamento	Departamento		Departamento
OF Empregado	Empregado		Empregado
OF Projeto	Projeto		Projeto
empregados OF Departamento	Dep-Emp		Empregado
departamento OF Empregado	Dep-Emp		Departamento
empregados OF Projeto	Proj-Emp → Emp-Proj	Proj-Emp	Particip-Projeto, Empregado
projetos OF Empregado	Emp-Proj → Proj-Emp	Emp-Proj	Particip-Projeto, Projeto
gerente OF Projeto	Gerente-Proj		Empregado
projeto-gerenciado OF Empregado	Gerente-Proj		Projeto

Figura 14: Mapeamentos gerados no passo 4 da metodologia

3.4.5 Refinar esquema

Como resultado da execução dos quatro passos anteriores obtém-se um esquema ECR estruturalmente completo e equivalente ao esquema rede original. Contudo, o modelo ECR permite que certas informações importantes não contidas no esquema rede possam ser incorporadas ao esquema ECR gerado.

Restrições de integridade que estavam sendo verificadas nos programas de aplicação locais podem ser definidas diretamente no esquema componente ECR. Podem ser incorporadas ao esquema componente, por exemplo, restrições sobre a cardinalidade dos relacionamentos representados pelos *set-type* e restrições sobre os valores permitidos para um atributo.

A inclusão de restrições de integridade semânticas no esquema componente é muito importante, porque permite que o próprio sistema se responsabilize pela sua verificação e manutenção, durante o processo

de transformação de operações submetidas pelos usuários federados em operações sobre os SBDs componentes. Logo, o esquema componente funciona como uma espécie de interface semântica [Oli92] para os SBDs que formam o SBDH. Essa interface semântica permite uma maior independência dos dados em relação as aplicações dos usuários. Em contrapartida, exige-se uma maior complexidade dos processadores que constituem o sistema gerenciador do SBDH.

No último passo da metodologia proposta, o DBA local e o DBA global devem avaliar a conveniência de se incluir no esquema componente ECR as restrições de integridade que não são controladas pelo SGBD rede local. Essa decisão representa uma espécie de compromisso entre a simplicidade e o grau de segurança desejados para o ambiente. Se por acaso, a opção escolhida for a de não incluir as restrições de integridade no esquema ECR, então essas restrições devem ser divulgadas entre a comunidade de usuários federados para que as operações submetidas por eles não violem a integridade dos dados.

A figura 15 mostra alguns exemplos de modificações feitas durante o passo de refinamento para incluir no esquema componente restrições de integridade não modeladas no esquema local rede. Os *valuesets* dos atributos *sexo* e *data-nasc* são alterados para permitir apenas determinados valores. Assim, o atributo *sexo* só admite os valores *"m"* para sexo masculino e *"f"* para sexo feminino. Da mesma forma, o atributo *data-nasc* só admite datas válidas entre os anos de 1943 e 1977 indicando que a empresa só admite empregados em determinada faixa etária. No relacionamento *Gerente-Proj* são alteradas a participação mínima e máxima do tipo de entidade *Projeto* para indicar que o relacionamento é 1:1 e não 1:N como foi modelado durante o passo 2 da metodologia.

4 Conversão Relacional-ECR

4.1 Considerações gerais

O modelo relacional possui um único construtor (*relação*) para modelar entidades e relacionamentos entre entidades. Como conseqüência dessa

```
DEFINE VALUESET sexo AS {``m'',``f''}.
DEFINE VALUESET data-nasc AS DATE 01011943:31121977.

DEFINE RELATIONSHIP Gerente-Proj FROM
                    Projeto   (projeto-gerenciado, gerente)
                              MINIMUM PARTICIPATION 1,
                              MAXIMUM PARTICIPATION 1
                    Empregado (gerente, projeto-gerenciado)
                              MINIMUM PARTICIPATION 0,
                              MAXIMUM PARTICIPATION 1.
```

Figura 15: Exemplos de alterações feitas durante refinamento do esquema na metodologia Rede-ECR

sobrecarga semântica existe a necessidade de se definir relacionamentos através de domínios comuns entre os atributos das relações. Além disso, o modelo relacional apresenta pouquíssimas restrições de integridade estruturais intrínsecas o que permite alternativas diferentes para se modelar certos conceitos semânticos. Por exemplo, um relacionamento 1:N pode ser modelado utilizando-se uma chave estrangeira na relação com cardinalidade N ou através de uma relação separada cuja chave primária seja a concatenação das chaves das relações que representam as entidades participantes.

Dentre as alternativas possíveis para o projeto de uma aplicação em um banco de dados relacional, algumas podem facilitar a percepção das entidades e relacionamentos e outras podem dificultar essa percepção, omitindo certas relações que representam tipos de entidade e representando apenas o seu relacionamento com outros tipos de entidade. Em geral, a normalização das relações ajuda na obtenção de "bons" projetos, tornando mais claro o papel desempenhado por cada relação [NA88] e impedindo uma série de anomalias durante a atualização dos dados [Dat86]. Contudo, nem sempre a normalização é suficiente para contornar todos os problemas de representação do modelo relacional [Ken79].

As limitações semânticas do modelo relacional dificultam a conversão Relacional-ECR, exigindo uma grande interação com o DBA local para que ele complemente informações omitidas no esquema relacional e resolva certas ambigüidades de projeto.

4.2 Trabalhos correlatos

Alguns trabalhos que discutem a conversão de esquemas relacionais para esquemas no MER (ou extensões do MER) como forma de documentação ou como técnica de projeto lógico de bancos de dados são: [CS83], [DA83], [BDH+88], [DA88], [NA88], [JK90].

[CS83] apresenta uma metodologia que restringe excessivamente o tipo de relações que podem ser tratadas o que se contrapõe à autonomia de projeto que deve ser garantida aos SBDs componentes. Além disso, a metodologia não analisa a cardinalidade dos relacionamentos gerados no esquema conceitual.

A metodologia proposta em [DA83] não trata alguns tipos de relacionamentos como, por exemplo, relacionamentos existentes entre duas relações primárias com chaves interdependentes. A metodologia também apresenta alguns critérios ambíguos para classificação das relações e atributos.

[BDH+88] apresenta uma metodologia de pouca praticidade porque exige que o usuário forneça a cobertura mínima das dependências funcionais.

[DA88] apresenta um critério para definição de tipos de entidade e relacionamentos baseado na correspondência entre as chaves das relações o que pode ocasionar problemas se os nomes dos atributos que compõem a chave não forem compatíveis, refletindo a semântica das relações.

[NA88] apresenta uma metodologia que prevê alguns passos para a modificação de nomes e a substituição de chaves primárias por chaves alternativas (ou vice-versa), uniformizando as referências entre as relações. Essa uniformidade é necessária porque o critério de classificação das relações é baseado na comparação do nome das suas chaves primárias. Contudo, a substituição de nomes de atributos não é desejável

no contexto de SBDHs porque exige que sejam feitos muitos mapeamentos adicionais entre os nomes originais e os nomes modificados, tornando o sistema muito mais complexo. Um outro problema da metodologia é não tratar conceitos importantes para o modelo ECR como as restrições em relação à participação de entidades nos relacionamentos.

A metodologia proposta em [JK90] apresenta um grande vantagem em relação às demais. Ela utiliza dependências de inclusão ao invés de dependências funcionais ou comparação de chaves. Uma dependência de inclusão (DI) entre duas relações R e S, denotada por $R(x) \ll S(y)$, indica a existência de uma dependência entre os valores do atributo (ou conjunto de atributos) x da relação R e os valores do atributo (ou conjunto de atributos) y da relação S, isto é, os valores de x formam um subconjunto dos valores de y em todas as tuplas das relações R e S. As dependências de inclusão, além de caracterizarem as ligações entre as relações, são independentes de nomes de atributos e aparecem em menor número do que as dependências funcionais. Todavia, a metodologia exige uma interação excessiva com o usuário pela presença de chaves alternativas nas DIs e pelo fato de que muitas ligações entre os tipos de entidades e os relacionamentos terem sido feitas em uma etapa posterior à identificação dos relacionamentos.

4.3 Metodologia Proposta

A seguir será apresentada uma metodologia para conversão de esquemas relacionais em esquemas ECR. A metodologia proposta leva em consideração a autonomia de projeto dos SBDs componentes e os mapeamentos que devem ser guardados durante a conversão para permitir a transformação de operações sobre o esquema ECR em operações equivalentes sobre o esquema relacional. O processo de conversão é dividido em quatro passos:

1. Classificação de relações e atributos;
2. Definição de tipos de entidades e relacionamentos;
3. Definição dos nomes de conexão;

4. Refinamento e validação do esquema ECR resultante;

4.3.1 Classificação de relações e atributos

A metodologia usa as dependências de inclusão existentes entre os atributos das relações para classificar essas relações em primárias, secundárias, terciárias e quaternárias.

Se nenhum subconjunto da chave primária de uma relação aparece do lado esquerdo de uma dependência de inclusão (DI), a relação é uma **relação primária (RP)**. Isto significa que a relação não depende da existência de outras relações. As DIs em que uma chave primária de uma RP aparece do lado direito serão denominadas **referências primárias**.

Se a chave primária de uma relação é formada pela concatenação de dois ou mais atributos[5] que aparecem do lado esquerdo de referências primárias, a relação é uma **relação secundária (RS)**. As RS servem como elo de ligação entre outras relações. As DIs que apresentam a chave de uma RS no seu lado direito são chamadas **referências secundárias**.

Se a chave de uma relação é composta por alguns atributos que aparecem do lado esquerdo de referências primárias ou secundárias e por outros atributos que não aparecem do lado esquerdo de nenhuma DI, então a relação é uma **relação terciária (RT)**. As DIs que possuem do seu lado direito a chave primária de uma RT são denominadas **referências terciárias**.

Uma relação que não puder ser classificada como RP, RS ou RT é uma **relação quaternária (RQ)**. As RQ possuem na composição de sua chave primária pelo menos um atributo que aparece do lado esquerdo de uma referência terciária.

Segundo as DIs definidas na figura 17 para as relações da figura 16, as relações *Cliente*, *Empregado*, *Projeto* e *Cargo* são classificadas como RP, as relações *Cliente-Proj* e *Gerente-Proj* são classificadas como RS, a relação *Etapa-Projeto* como RT e a relação *Empreg-Proj* como RQ.

Os atributos são classificados, conforme a função que desempenham na relação, em atributos chave-primária (ACP), atributos chave-alternativa

[5] O termo atributo está se referindo a atributos simples ou compostos

```
Departamento(cod-depto,nome)
Empregado(matricula,nome,salario,cargo,depto)
Cargo(cod-cargo,nome-cargo,salario-medio)
Projeto(cod-projeto,nome-projeto,contrato,valor-cont)
Cliente(cod-cliente,nome-cliente,endereco,telefone)
Etapa-Projeto(cod-projeto,num-etapa,data-inic,data-fim)
Empreg-Proj(matric,cod-projeto,num-etapa,horas-trab)
Particip-Projeto(matric,cod-projeto,horas-trab)
Gerente-Proj(matric,cod-projeto)
Cliente-Proj(cod-cliente,cod-projeto)
```

Figura 16: Relações do esquema local relacional

```
Empregado.cargo << Cargo.cod-cargo
Empregado.depto << Depto.cod-depto
Empreg-Proj.matric << Empregado.matricula
Empreg-Proj.num-etapa << Etapa-Projeto.num-etapa
Empreg-Proj.cod-projeto << Etapa-Projeto.cod-projeto
Etapa-Projeto.cod-projeto << Projeto.cod-projeto
Cliente-Proj.cod-cliente << Cliente.cod-cliente
Cliente-Proj.cod-projeto << Projeto.cod-projeto
Gerente-Proj.matric << Empregado.matricula
Gerente-Proj.cod-projeto << Projeto.cod-projeto
```

Figura 17: Dependências de inclusão nas relações da figura

(ACA) e atributos não-chave (ANC). Os ANC são subdivididos em atributos não-chave incluídos na chave primária ($ANC1$) e atributos não-chave não incluídos na chave primária ($ANC2$).

4.3.2 Definição de tipos de entidade e relacionamentos

Para se identificar os tipos de entidade e os relacionamentos representados pelas relações são examinados os grupos obtidos no passo anterior na seguinte ordem: 1) RP; 2) RS; 3) RT; 4) RQ.

Relações Primárias

Cada RP origina um tipo de entidade provisório no esquema ECR. Os atributos da relação tornam-se atributos do tipo de entidade criado. Para cada atributo do tipo de entidade será definido um *valueset* correspondente com mesmo nome, tipo e tamanho do atributo correspondente no esquema relacional. Os atributos do tipo de entidade que correspondem à chave primária ou à uma chave alternativa da relação serão identificados como tal no esquema ECR, através das cláusulas KEY e UNIQUE, respectivamente. Chaves primárias ou alternativas compostas por mais de um atributo geram atributos compostos no tipo de entidade.

Normalmente, os SGBDs relacionais trabalham com relações em 1FN, embora existam propostas para modelos relacionais que não apresentem essa limitação [Set86, SS86]. Para que as relações obedeçam a 1FN é necessário que os atributos multivalorados de um tipo de entidade sejam colocados em uma relação a parte juntamente com o atributo identificador da relação que representa esse tipo de entidade. A chave da relação usada para modelar um atributo multivalorado é formada pela concatenação do próprio atributo multivalorado com o atributo correspondente à chave da relação que representa o tipo de entidade.

Alguns SGBDs relacionais aceitam valores nulos para os atributos. No modelo ECR essa possibilidade é expressa atribuindo-se à cardinalidade mínima do atributo o valor zero. Se a cardinalidade mínima for omitida no esquema ECR, assume-se que o atributo não pode ter valores nulos.

As correspondências entre os tipos de entidade gerados e as relações originais devem ser armazenadas no DD local para que possam ser utilizadas na transformação de operações GORDAS em operações em uma LMD relacional. A figura 18 mostra parte do esquema ECR gerado a partir das relações primárias da figura 16. A figura 19 mostra os mapeamentos armazenados no DD local.

```
DEFINE VALUESET matricula AS INTEGER RANGE 0:99999.
                .
                .
                .
DEFINE ENTITY-TYPE Empregado
       ATTRIBUTES
           matricula VALUESET matricula KEY,
           nome      VALUESET nome,
           salario   VALUESET salario,
           cargo     VALUESET cargo,
           depto     VALUESET depto.
```

Figura 18: Trecho de esquema ECR obtido pelo exame de relações primárias

Entrada	Tipo	Relação
Empregado	TE	Empregado

Figura 19: Mapeamento gerado durante o exame de relações primárias

Relações Secundárias

Cada *RS* pode originar um tipo de entidade ou um relacionamento. Se a chave primária ou um atributo não contido na chave primária aparecer em uma *DI*, então a *RS* é convertida em um tipo de entidade

fraca. Caso contrário, a RS origina um relacionamento que associa os tipos de entidade referenciados pelos atributos que compõem a sua chave primária. Os atributos da relação tornam-se atributos do tipo de entidade ou do relacionamento gerado.

Se a RS for convertida em um tipo de entidade fraca, cria-se também um relacionamento associando o tipo de entidade fraca e os tipos de entidade referenciados pelos atributos que compõem a chave primária da RS. A participação do tipo de entidade fraca no relacionamento é definida como específica e funcional, caracterizando a sua dependência em relação aos demais tipos de entidade que participam do relacionamento. A participação dos outros tipos de entidade é definida como parcial e não funcional.

No caso da RS corresponder a um relacionamento, as dependências funcionais entre os atributos componentes da chave dessa relação podem indicar se o tipo de participação dos tipos de entidade envolvidos no relacionamento é funcional ou não. Por exemplo, se a chave da RS examinada é composta pelos atributos x, y e z que referenciam os tipos de entidade $E1$, $E2$ e $E3$, respectivamente, e existe a dependência funcional $x \rightarrow yz$, então a participação do tipo de entidade $E1$ deve ser funcional. Todavia, para evitar o tratamento de dependências funcionais, pode-se assumir que a participação dos tipos de entidade seja não funcional e deixar para o passo de refinamento e validação do esquema ECR as correções necessárias. O tipo de participação total ou parcial dos tipos de entidade pode ser avaliado diretamente das DIs. Se existir uma DI da relação correspondente ao tipo de entidade para a RS que deu origem ao relacionamento, a participação do tipo de entidade é total. Caso contrário é parcial.

Devem ser guardados no DD local as correspondências entre os tipos de entidade e relacionamentos gerados e as relações originais. A figura 20 apresenta parte do esquema ECR obtido através do exame das relações secundárias da figura 16. A figura 21 mostra os mapeamentos incluídos no DD local.

Relações Terciárias

```
DEFINE RELATIONSHIP Cliente-Proj FROM
       Cliente, Projeto
       ATTRIBUTES cod-cliente VALUESET cod-cliente,
                  cod-projeto VALUESET cod-projeto.
```

Figura 20: Trecho de esquema ECR obtido pelo exame de relações secundárias

Entrada	Tipo	Relação	Dependências de Inclusão
Cliente-Proj	Rel. Bin.	Cliente-Proj	Cliente-Proj.cod-cliente << Cliente.cod-cliente, Cliente-Proj.cod-projeto << Projeto.cod-projeto

Figura 21: Mapeamento gerado durante o exame de relações secundárias

Se um atributo não contido na chave primária ($ANC2$ ou ACA) da RT aparecer em uma DI, então a RT é convertida em um tipo de entidade fraca. Todos os atributos da relação, tornam-se atributos do tipo de entidade criado. É gerado também um relacionamento que associa o tipo de entidade fraca com os tipos de entidade referenciados pelos atributos que compõem a chave da RT. A participação do tipo de entidade fraca no relacionamento é específica e funcional enquanto que a participação dos demais é parcial e não funcional.

Se a condição anterior não for satisfeita, a RT pode representar um tipo de entidade fraca, um atributo multivalorado de um tipo de entidade ou um relacionamento. Na primeira opção, os atributos que compõem a chave primária e não aparecem do lado esquerdo de nenhuma DI são considerados identificadores próprios do tipo de entidade fraca. Na segunda opção, esses mesmos atributos são considerados atributos multivalorados do tipo de entidade referenciado pelos outros atributos que compõem a chave. Na terceira opção, tais atributos são considerados como chaves primárias de tipos de entidades implicitamente referenciadas no projeto e que podem ser explicitadas no esquema ECR. O problema é que não

existem critérios objetivos genéricos para se decidir qual o conceito representado. A decisão depende do conhecimento do contexto da aplicação e, portanto, deve ser tomada pelo DBA local.

Se o DBA caracterizar a RT como um tipo de entidade fraca, deve-se executar as mesmas ações já descritas para este caso.

Se, por outro lado, o DBA considerar que a relação está sendo usada para modelar um atributo multivalorado de um tipo de entidade, então esse atributo deve ser incluído no tipo de entidade e o DBA deve informar qual a cardinalidade máxima permitida para ele.

Finalmente, se o DBA decidir que a relação representa um relacionamento, as seguintes ações são executadas. Em primeiro lugar, é criado um tipo de entidade no esquema ECR para cada atributo componente da chave que não aparece do lado esquerdo de nenhuma DI. O único atributo de um tipo de entidade criado é o próprio atributo que deu origem a ele. Em segundo lugar, é criado um relacionamento de grau k, $k \geq 2$, associando os k tipos de entidade referenciados na chave primária da RT examinada. A participação de cada tipo de entidade no relacionamento é parcial e não funcional. Os atributos da RT tornam-se atributos do relacionamento criado.

As DIs que referenciam no seu lado direito os atributos que motivaram a criação dos tipos de entidade devem ser substituídas por DIs que passam a referenciar as relações implicitamente identificadas por esses atributos, que foram explicitadas no esquema ECR. Assim, se o atributo a da RT $R1(\underline{a,b,c,d})$ deu origem ao tipo de entidade E, então a DI $S(x) \ll R1(a)$ deve ser substituída pela DI $S(x) \ll E(a)$. Essa substituição é importante porque permite que essas dependências de inclusão sejam tratadas como referências primárias nos próximos passos da metodologia.

Nas figuras 22 e 23 são apresentadas a conversão da RT *Etapa-projeto* da figura 16, interpretada como um tipo de entidade fraca, e os mapeamentos gerados no DD local a partir dessa conversão.

Relações Quaternárias

```
DEFINE VALUESET descr-etapa AS STRING 40.
DEFINE VALUESET data-prev-inic AS DATE.
                  .
                  .
                  .
DEFINE ENTITY-TYPE Etapa-Projeto
       ATTRIBUTES
             etapa-projeto-id COMPOUND
                (cod-projeto VALUESET cod-projeto,
                 num-etapa   VALUESET num-etapa) KEY,
             descr-etapa     VALUESET descr-etapa,
             data-inic       VALUESET data-inic,
             data-fim        VALUESET data-fim.

DEFINE RELATIONSHIP Cronog-Projeto FROM
       Projeto,
       Etapa-Projeto SPECIFIC
                     MINIMUM PARTICIPATION 1
                     MAXIMUM PARTICIPATION 1.
```

Figura 22: Trecho de esquema ECR obtido pelo exame de relações terciárias

Entrada	Tipo	Relação	Dependências de Inclusão
Etapa-projeto	entid.fraca (TE2)	Etapa-projeto	Etapa-projeto.cod-projeto << Projeto.cod-projeto

Entrada	Tipo	Entidades Participantes
Cronog-Projeto	Relac.fraco	Projeto, Etapa-Projeto

Figura 23: Mapeamentos gerados durante o exame de relações terciárias

As *RQ* são tratadas da mesma forma que as relações terciárias que representam relacionamentos. É importante destacar, no entanto, que a separação entre *RT* e *RQ* é essencial para a metodologia proposta. As *RQ* precisam ser examinadas após as *RT*, uma vez que as primeiras referenciam as últimas e, portanto, o resultado das modificações nas dependências de inclusão que envolvem as RTs são importantes no exame das *RQ*.

As figuras 24 e 25 mostram, respectivamente, a conversão da *RQ* *Empreg-Proj* da figura 16 e os mapeamentos armazenados no DD local.

```
DEFINE VALUESET horas-trab AS INTEGER.

DEFINE RELATIONSHIP Empreg-Proj FROM
       Empregado,
       Etapa-projeto
               ATTRIBUTES
                  empreg-proj-id COMPOUND
                        (matric VALUESET matricula,
                         cod-projeto VALUESET cod-projeto,
                         num-etapa   VALUESET num-etapa) KEY,
                  horas-trab VALUESET horas-trab.
```

Figura 24: Trecho de esquema ECR obtido pelo exame de relações quaternárias

Entrada	Tipo	Relação	Dependências de Inclusão
Empreg-Proj	Relac.	Empreg-Proj	matricula $\ll$ Empregado.matricula, cod-projeto $\ll$ Etapa-proj.cod-projeto, num-etapa $\ll$ Etapa-proj.num-etapa,

Figura 25: Mapeamento gerado durante o exame de relações quaternárias

Chaves estrangeiras

Dependências de inclusão envolvendo atributos não-chave ($ANC1$ e $ANC2$) de relações que originaram tipos de entidade podem indicar relacionamentos entre esses tipos de entidade. As DIs em que aparecem $ANC1$ do lado esquerdo foram consideradas durante o exame das RS, RT e RQ. Nesse momento, são tratadas as DIs em que atributos $ANC1$ ou $ANC2$ aparecem do lado direito e as DIs em que atributos $ANC2$ aparecem do lado esquerdo.

Cada uma das DIs examinada dá origem a um relacionamento binário associando os tipos de entidade referenciadas do seu lado esquerdo e do seu lado direito. Se o atributo do tipo de entidade referenciado do lado esquerdo da DI pode conter valores nulos, a participação do tipo de entidade em questão no relacionamento é funcional e parcial. Caso contrário a participação é funcional e total. Para o tipo de entidade referenciado no lado direito de uma DI é atribuída uma participação não funcional e parcial.

A figura 26 mostra o relacionamento entre as relações *Empregado* e *Cargo* da figura 16. A figura 27 mostra os mapeamentos armazenados no DD local.

```
DEFINE RELATIONSHIP Cargo-Empregado FROM
            Empregado MINIMUM PARTICIPATION 1
                      MAXIMUM PARTICIPATION 1,
            Cargo.
```

Figura 26: Trecho de esquema ECR obtido pelo exame de atributos não chave

4.3.3 Definição dos nomes de conexão

Neste passo da metodologia são definidos pelo DBA local e pelo DBA do SBDH, os nomes de conexão que serão associados à participação de cada classe nos relacionamentos.

Após a inclusão dos nomes de conexão no esquema ECR devem ser

Entrada	Tipo	Relação	Relação Referenciada	Chave Estrang.	Chave Referenciada
Cargo-Empregado	chave estrangeira	Empregado	Cargo	cargo	cod-cargo

Figura 27: Mapeamento gerado durante o exame de atributos não chave

definidos os mapeamentos entre eles e as relações no esquema relacional. Esses mapeamentos serão utilizados para transformar os caminhos de busca expressos na linguagem GORDAS em referências às relações na linguagem utilizada pelo SGBD relacional. Os mapeamentos em questão são obtidos examinando-se a definição dos relacionamentos no esquema ECR.

Seja $R = \{R_1, R_2, \cdots, R_n\}$ o conjunto dos relacionamentos gerados no esquema ECR.

Seja $R_i = \{(E_1, na_1, nb_1), (E_2, na_2, nb_2), \cdots, (E_n, na_n, nb_n)\}, 1 \leq i \leq n$, um relacionamento qualquer de R que associa as classes $E_1, E_2, \cdots, E_n$ através dos nomes de conexão na_i e nb_i.

Sejam na_j o *nome-conexão-1* e nb_j o *nome-conexão-2* definidos para a classe E_j, $1 \leq j \leq n$, que participa do relacionamento R_i. Para cada E_j devem ser gerados os seguintes mapeamentos:

1. Um mapeamento do caminho definido pela expressão $<of\ E_j>$ para a relação correspondente a E_j;

2. Um mapeamento do caminho definido pela expressão $<na_j\ of\ E_j>$ para a relação correspondente a R_i no esquema relacional, se tal relação existir. Se o relacionamento R_i é binário deve ser gerado um mapeamento adicional da expressão $<na_j\ of\ E_j>$ para a relação correspondente à outra classe E_k que participa do relacionamento. A classe E_k pode ser inclusive igual a E_j, no caso de R_i ser um relacionamento recursivo.

3. Um mapeamento do caminho definido pela expressão $<na_j\ of\ E_j>$ para a relação referenciada pela chave estrangeira existente na

relação correspondente a E_j, caso o relacionamento R_i tenha sido definido pela presença de uma chave estrangeira.
A relação referenciada pela chave estrangeira pode ser obtida na entrada do DD local que guardou a correspondência entre o relacionamento R_i e os construtores do esquema relacional (ver figura 27).

4. Mapeamentos do caminho definido por cada expressão $<nb_k$ *of* na_j *of* $E_j>$, $1 \leq k \leq n$, $k \neq j$, para cada relação correspondente a E_k, se R_i é um relacionamento que envolve mais de duas classes.

As figuras 28 e 29 apresentam, respectivamente, parte de um esquema ECR contendo os nomes de conexão e os mapeamentos que serão guardados no DD local. O leitor deve observar que os mapeamentos de caminhos como <*departamento of empregados of Cargo*> embora não declarados explicitamente no DD local, podem ser facilmente obtidos com a substituição da subexpressão <*empregados of Cargo*> pela relação equivalente, no caso a relação *Empregado*.

```
DEFINE RELATIONSHIP Cargo-Empregado FROM
              Cargo     (empregados, cargo),
              Empregado (cargo, empregados)
                        MIMIMUM PARTICIPATION 1,
                        MAXIMUM PARTICIPATION 1.

DEFINE RELATIONSHIP Departamento-Empregado FROM
              Departamento (empregados, departamento),
              Empregado    (departamento, empregados)
                       MIMIMUM PARTICIPATION 1
                       MAXIMUM PARTICIPATION 1.
```

Figura 28: Nomes de conexão incluídos no passo 3 da metodologia Relac.-ECR

Caminho ECR	Relação
OF Empregado	Empregado
OF Departamento	Departamento
OF Cargo	Cargo
OF Projeto	Projeto
⋮	⋮
departamento OF Empregado	Departamento
cargo OF Empregado	Cargo
empregados OF Departamento	Empregado
empregados OF Cargo	Empregado

Figura 29: Mapeamentos gerados no passo 3 da metodologia Relac.-ECR

4.3.4 Refinamento e validação

No último passo da metodologia, o DBA local e o DBA do SBDH devem examinar o esquema obtido até o passo anterior para avaliar a necessidade de inclusão ou modificação de certas informações nesse esquema. Assim como na conversão Rede-ECR, também na conversão Relacional-ECR é importante garantir que todas as informações relevantes não contidas explicitamente no esquema local sejam inseridas no esquema componente para permitir uma maior independência dos dados em relação às aplicações dos usuários federados.

5 Conclusão

Neste trabalho foram apresentadas metodologias para conversão de esquemas em SBDHs que integram SBDs relacionais e SBDs rede. Essas metodologias fazem parte de um conjunto de soluções propostas em [Oli93] para garantir a transparência de modelos e linguagens em SBDHs.

As metodologias propostas diferem de outras metodologias apresentadas na literatura [CS83, DA83, EWH85, BDH+88, DA88, NA88, JK90, Fon92] por apresentarem características específicas que permitem a sua aplicação em SBDHs. Algumas dessas características são:

- a preservação da autonomia dos SBDs componentes;
- a construção e armazenamento de mapeamentos que são utilizados na transformação das operações dos usuários em operações equivalentes nas LMDs dos SBDs componentes;
- a utilização de um modelo de dados comum semântico que permite a incorporação ao esquema componente de informações não contidas nos esquemas locais;

Espera-se obter com a implementação das metodologias um conjunto de ferramentas que sirva como base para um projeto de maior porte que envolva outras áreas de pesquisa em SBDHs como integração de esquemas, controle de acesso, processamento e otimização de transações, gerenciamento de transações e administração do SBDH.

Referências

[ADD+91] R. Ahmed, P. DeSmedt, W. Du, W. Kent, M. Ketabchi, W. Litwin, A. Rafii, and M. Shan. The PEGASUS heterogeneous multidatabase system. *Computer*, pages 19–27, December 1991.

[BDH+88] H. Briand, C. Ducateau, Y. Hebrail, D. Herin-Aime, and J. Kouloumdjian. From minimal cover to entity-relationship diagram. In S. T. March, editor, *Entity-Relationship Approach*, pages 287–304. North-Holland, 1988.

[BDK+88] V. Belcastro, A. Dutkowski, W. Kominski, M. Kowalewski, C. Mallamaci, and S. Mezyic. An overview of the distributed query system DQS. In *Lecture Notes in Computer Science*, pages 170–189. Springer-Verlag, 1988. Volume 303.

[Car87] A. Cardenas. Heterogeneous distributed database management: The HD-DBMS. *Proceedings of the IEEE*, 75(5):588–600, May 1987.

[Che76] P. Chen. The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[Chu90] C. Chung. DATAPLEX: An access to heterogeneous distributed databases. *Communications of ACM*, 33(1):70–80, January 1990.

[CS83] M. A. Casanova and J. E. Sa. Design entity-relationship schemas for conventional information systems. In *Entity-Relationship Approach to Software Engineering*, pages 265–277. North-Holland, 1983.

[DA83] S. Dumpala and S. Arora. Schema translation using the entity-relationship approach. In P. P. Chen, editor, *Entity-Relationship Approach to Information Modeling and Analysis*, pages 337–356. North-Holland, 1983.

[DA88] K. Davis and A. Arora. Converting a relational database model into an entity-relationship model. In S. T. March, editor, *Entity-Relationship Approach*, pages 271–285. North-Holland, 1988.

[Dat86] C.J. Date. Introdução a sistemas de bancos de dados, 1986. Capítulo 1, páginas 26-47, Editora Campus ltda., terceira edição.

[Dem83] B. Demo. Program analysis for convertion from a navigational to a specification database interface. In *Proceedings of the 9th Int. Conference on Very Large DataBases*, pages 387–398, Florence, Italy, October 1983.

[DL87] P. Dwyer and J. Larson. Some experiences with a distributed database testbed system. *Proceedings of the IEEE*, 75(5):633–647, May 1987.

[EP90] A. Elmargamid and C. Pu. Guest editors introduction to the special issue on heterogeneous databases. *ACM Computing Surveys*, 22(3):175–178, September 1990.

[EW83] R. Elmasri and G. Wiederhold. GORDAS: a formal high-level query language for the entity-relationship model. In P. P. Chen, editor, *Entity-Relationship Approach to Information Modeling and Analysis*, pages 49–72. North-Holland, 1983.

[EWH85] R. Elmasri, J. Weeldreyer, and A. Hevner. The category concept: An extension to entity-relationship model. In H. J. Schneider, editor, *Data and Knowledge Engineering*, pages 75–116. North-Holland, June 1985.

[Fon92] J.S. Fong. Methodology for schema translation from hierarquical or network into relational. *Information and Software Technology*, 34(3):159–174, March 1992.

[HB91] A. Hurson and M. Bright. Multidatabase systems: An advanced concept in handling distributed data. In M. Yovits, editor, *Advances in Computers*, pages 149–200. Academic Press, 1991.

[HK87] R. Hull and R. King. Semantic database modeling: Survey, applications, and research issues. *ACM Computing Surveys*, 19(3):45–62, September 1987.

[HK89] D. Hsiao and M. Kamel. Heterogeneous databases: Proliferation, issues, and solutions. *IEEE Transactions on Knowledge Data Engineering*, 1(1):45–62, 1989.

[HM85] D. Heimbigner and D. McLeod. A federated architecture for information management. *ACM Transactions on Office Systems*, 3(3):253–278, July 1985.

[Hsi92] D. Hsiao. Tutorial on federated databases and systems (Part I). *The VLDB Journal*, 1(1):127–179, jul 1992.

[JK90] P. Johannesson and K. Kalman. A method for translating relational schemas into conceptual schemas. In F. H. Lochovsky, editor, *Entity-Relationship Approach to Database Design and Querying*, pages 271–285. North-Holland, 1990.

[Ken79] W. Kent. Limitations of record based information models. *ACM Transactions on Database Systems*, 4(1):107–131, 1979.

[LA86] W. Litwin and A. Abdellatif. Multidatabase interoperability. *Computer*, 19(12):10–18, December 1986.

[Lar83] J. Larson. Bridging the gap between network and relational management systems. *Computer*, pages 82–92, September 1983.

[Lie81] Y. Lien. Hierarquical schemata for relational databases. *ACM Transactions on Database Systems*, 4(1):107–131, 1981.

[LR82] T. Landers and R. Rosenberg. An overview of multibase. In H. J. Schneider, editor, *Distributed DataBases*, pages 153–184. North-Holland, December 1982.

[NA88] S. Navathe and A. Awong. Abstracting relational and hierarquical data with a semantic data model. In S. T. March, editor, *Entity-Relationship Approach*, pages 305–333. North-Holland, 1988.

[Oli92] Ricardo Oliveira. S-SQL: uma interface semântica. Master's thesis, DCC-IMECC-UNICAMP, 1992.

[Oli93] Ronaldo Lopes de Oliveira. Transparência de modelos em sistemas de bancos de dados heterogêneos. Master's thesis, DCC-IMECC-UNICAMP, 1993. To be published.

[PM88] J. Peckham and R. Maryanski. Semantic data models. *ACM Computing Surveys*, 20(3):153–189, September 1988.

[Set86] W. Setzer. Projeto lógico e projeto físico de banco de dados, 1986. V Escola de Computação, Belo Horizonte, Capítulos 3 e 4.

[SL90] A. Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, September 1990.

[SS86] H. Schek and M. Scholl. The relational model with relation-valued atributes. *Information Systems*, 11(2), 1986.

[TBC+87] M. Templeton, D. Brill, A. Chen, S. Dao, E. Lund, R. McGregor, and P. Ward. MERMAID: a front-end to distributed heterogeneous databases. *Proceedings of the IEEE*, 75(5):695–708, May 1987.

[TF76] R. Taylor and R. Frank. CODASYL database management systems. *ACM Computing Surveys*, 8(1):67–103, March 1976.

[TYF86] T. Teorey, D. Yang, and J. Fry. A logical design metodology for relational databases using the extended entity-relationship model. *ACM Computing Surveys*, 18(2):197–222, June 1986.

[VL80] Y. Vassiliou and F. Lochovsky. DBMS transaction translation. In *Proceedings COMPSAC 80, IEEE Computer Software and Application Conference*, 1980.

[Zan79] C. Zaniolo. Design of relational views over network schemas. In *Proceedings of ACM SIGMOD Int. Conference on Management of Data*, pages 179–190, 1979.

Relatórios Técnicos – 1992

01/92 **Applications of Finite Automata Representing Large Vocabularies,** *C. L. Lucchesi, T. Kowaltowski*

02/92 **Point Set Pattern Matching in d-Dimensions,** *P. J. de Rezende, D. T. Lee*

03/92 **On the Irrelevance of Edge Orientations on the Acyclic Directed Two Disjoint Paths Problem,** *C. L. Lucchesi, M. C. M. T. Giglio*

04/92 **A Note on Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams,** *W. Jacometti*

05/92 **An (l, u)-Transversal Theorem for Bipartite Graphs,** *C. L. Lucchesi, D. H. Younger*

06/92 **Implementing Integrity Control in Active Databases,** *C. B. Medeiros, M. J. Andrade*

07/92 **New Experimental Results For Bipartite Matching,** *J. C. Setubal*

08/92 **Maintaining Integrity Constraints across Versions in a Database,** *C. B. Medeiros, G. Jomier, W. Cellary*

09/92 **On Clique-Complete Graphs,** *C. L. Lucchesi, C. P. Mello, J. L. Szwarcfiter*

10/92 **Examples of Informal but Rigorous Correctness Proofs for Tree Traversing Algorithms,** *T. Kowaltowski*

11/92 **Debugging Aids for Statechart-Based Systems,** *V. G. S. Elias, H. Liesenberg*

12/92 **Browsing and Querying in Object-Oriented Databases,** *J. L. de Oliveira, R. de O. Anido*

Relatórios Técnicos – 1993

01/93 **Transforming Statecharts into Reactive Systems,** *Antonio G. Figueiredo Filho, Hans K. E. Liesenberg*

02/93 **The Hierarchical Ring Protocol: An Efficient Scheme for Reading Replicated Data,** *Nabor das C. Mendonça, Ricardo de O. Anido*

03/93 **Matching Algorithms for Bipartite Graphs,** *Herbert A. Baier Saip, Cláudio L. Lucchesi*

04/93 **A lexBFS Algorithm for Proper Interval Graph Recognition,** *Celina M. H. de Figueiredo, João Meidanis, Célia P. de Mello*

05/93 **Sistema Gerenciador de Processamento Cooperativo,** *Ivonne. M. Carrazana, Nelson. C. Machado, Célio. C. Guimarães*

06/93 **Implementação de um Banco de Dados Relacional Dotado de uma Interface Cooperativa,** *Nascif A. Abousalh Neto, Ariadne M. B. R. Carvalho*

07/93 **Estadogramas no Desenvolvimento de Interfaces,** *Fábio N. de Lucena, Hans K. E. Liesenberg*

08/93 **Introspection and Projection in Reasoning about Other Agents,** *Jacques Wainer*

09/93 **Codificação de Seqüências de Imagens com Quantização Vetorial,** *Carlos Antonio Reinaldo Costa, Paulo Lício de Geus*

10/93 **Minimização do Consumo de Energia em um Sistema para Aquisição de Dados Controlado por Microcomputador,** *Paulo Cesar Centoducatte, Nelson Castro Machado*

11/93 **An Implementation Structure for RM-OSI/ISO Transaction Processing Application Contexts,** *Flávio Morais de Assis Silva, Edmundo Roberto Mauro Madeira*

12/93 **Boole's conditions of possible experience and reasoning under uncertainty,** *Pierre Hansen, Brigitte Jaumard, Marcus Poggi de Aragão*

13/93 **Modelling Geographic Information Systems using an Object Oriented Framework,** *Fatima Pires, Claudia Bauzer Medeiros, Ardemiris Barros Silva*

14/93 **Managing Time in Object-Oriented Databases,** *Lincoln M. Oliveira, Claudia Bauzer Medeiros*

15/93 **Using Extended Hierarchical Quorum Consensus to Control Replicated Data: from Traditional Voting to Logical Structures,** *Nabor das Chagas Mendonça, Ricardo de Oliveira Anido*

Departamento de Ciência da Computação — IMECC
Caixa Postal 6065
Universidade Estadual de Campinas
13081-970 – Campinas – SP
BRASIL

reltec@dcc.unicamp.br