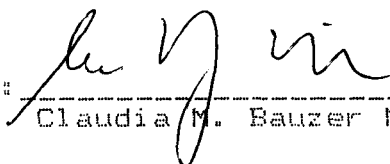


**Um Sistema para Manuseio de
Objetos Entidade-Relacionamento
no Modelo Relacional**

Este exemplar corresponde à redação
final da tese devidamente corrigida,
defendida por **Mônica de Lima Nogueira**
e aprovada pela Comissão Julgadora.

Campinas, 29 de março de 1989.

Prof.^a. Dr.^a. :



Claudia M. Bauzer Medeiros

Orientadora

Dissertação apresentada ao Instituto
de Matemática, Estatística e Ciência
da Computação, UNICAMP, como requisito
parcial para obtenção do título de
"Mestre em Ciência da Computação".

UNICAMP
BIBLIOTECA CENTRAL

Aos meus filhos
Heloisa, Marcus Vinicius, e "bebê"
que participaram "on-line" da elaboração desta tese.

Ao meu marido Geraldo
meu incentivador incondicional.

AGRADECIMENTOS

A Profa. Dra. Claudia Maria Bauzer Medeiros, pela orientação dedicada e paciente, e pelos muitos ensinamentos recebidos, os quais extrapolam os limites desta tese.

Ao Prof. Luis Tucherman, pelo interesse e incentivo demonstrados na execução e prosseguimento deste trabalho.

Ao Prof. Dr. Geovane Cayres Magalhães, pelas críticas e sugestões que muito contribuíram para melhoria desta tese.

Ao Prof. Dr. Leo Pini Magalhães, pelo oferecimento de material bibliográfico e sugestões apresentadas a este texto.

A IBM Brasil, na pessoa do Dr. Marco Antonio Casanova, Coordenador do Convênio IBM-UNICAMP, pelo recebimento de uma Bolsa de Estudos que possibilitou a realização deste trabalho.

A CAPES e CNPQ, que patrocinaram a fase inicial deste curso com uma Bolsa de Estudos.

Aos meus pais, Maria do Carmo e Ivens Lima, pelo apoio e incentivo permanentes, e pelo esforço ilimitado dedicado aos cuidados de meus filhos, fundamental para que esta tese pudesse ser finalizada.

SUMARIO

Esta tese descreve as características de uma interface E-R para um banco de dados relacional - o sistema REVER. O usuário do sistema pode utilizá-la tanto para fazer o projeto do banco de dados quanto para consultas e atualizações. A visão do usuário é a de criação de elementos de um diagrama E-R estendido e de manipulação de suas ocorrências. O sistema especificado se encarrega de mapear as solicitações do usuário para operações sobre o esquema e instâncias do banco de dados relacional correspondente.

Além da especificação, a tese descreve os detalhes da implementação e testes de um protótipo da interface, validando a especificação. O protótipo se ocupa das fases de geração e manuseio de esquemas E-R, sendo o diálogo com o usuário feito através de menus.

ABSTRACT

This thesis describes the features of an E-R interface to a relational database - the REVER system. The system allows the user to design the database using only the E-R specification, as well as to query and update it. The user's view is always that of an E-R (extended) diagram, in the design and in the operational phases. The system specified here maps the user's requests on the E-R scheme and instances to operations on the corresponding relational database.

The thesis also describes the details of developing and testing a prototype of the interface, thus validating the specification. This prototype allows the generation and manipulation of E-R schemes by means of menus.

Um Sistema para Manuseio de Objetos Entidade-Relacionamento no Modelo Relacional

Índice

	Página
1 - Introdução	01
1.1 - Variações de Modelos E-R	01
1.2 - Aplicação da Modelagem e Ferramentas	07
1.3 - Organização da Tese	18
2 - Conceitos e Nomenclatura	21
2.1 - Modelo Entidade-Relacionamento	21
2.2 - Modelo E-R Estendido	24
2.3 - Modelo Relacional	29
2.4 - Resumo	32
3 - Especificação da Interface REVER	33
3.1 - Descrição Geral	33
3.2 - Mapeamento E-R <--> Relacional	36
3.3 - Simplificações para o Presente Trabalho	41
3.4 - Especificação da Interface	47
3.4.1 - Módulo de Diálogo	47
3.4.2 - O Módulo Esquema	51
3.4.3 - O Módulo Manuseador	54
3.4.4 - O Módulo Tradutor	63
3.5 - Resumo	65

4 - Descrição da Implementação	66
4.1 - Descrição Geral	66
4.2 - Exemplo de Utilização	86
4.3 - Considerações de Eficiência	97
4.4 - Consultas: Navegação dentro do Esquema	99
4.5 - Resumo	102
5 - Conclusão	103
5.1 - Considerações Finais	103
5.2 - Extensões Sugeridas	105
5.3 - Resumo	110
BIBLIOGRAFIA	111

Índice de Figuras

	Página
1.1 - Quadro de Classificação dos Modelos E-R	03
2.1 - Esquema-exemplo mostrando Entidade, Relacionamento e Atributo	22
2.2 - Esquema-exemplo mostrando Relacionamento Total . .	23
2.3 - Esquema-exemplo mostrando Cardinalidade	24
2.4 - Simbologia usada para Generalização, Agregação e Hierarquia de Tipos	25
2.5 - Esquema-exemplo de Agregação	26
2.6 - Esquema-exemplo de Generalização	27
2.7 - Esquema-exemplo de Hierarquia de Tipos	27
3.1 - Arquitetura Proposta da Interface REVER	34
3.2 - Configurações Proibidas envolvendo mais de um Relacionamento Total	42
3.3 - Configuração Proibida envolvendo Agregação com Relacionamento Total e Árvores	42
3.4 - Configuração Proibida envolvendo Agregação e Relacionamentos Totais	43
3.5 - Configurações Proibidas envolvendo Árvores e Relacionamentos Totais	43
3.6 - Configurações Proibidas envolvendo Árvores	44
3.7 - Exemplo da Configuração Proibida 3.2a	45
3.8 - Exemplo da Configuração Proibida 3.2b	45

3.9 - Exemplo da Configuração Proibida 3.3	46
3.10- Exemplo envolvendo Dependência de Inclusão	55
3.11- Diagrama Esquemático para Navegação dentro de uma Estrutura	59
3.12- Exemplo de Arvore Heterogênea com Generalização e Hierarquia	60
3.13- Exemplo de Propagação de Atualizações quando há Dependência de Inclusão	62
3.14- Exemplo de Propagação de Atualizações em Entidades	62
4.1 - Estrutura Intermediária	69
4.2 - Estrutura de Dados para Objetos do Tipo ENTIDADE	72
4.3 - Estrutura de Dados para Objetos do Tipo RELACIONAMENTO	74
4.4 - Estrutura de Dados para Objetos do Tipo AGREGAÇÃO	75
4.5 - Estrutura de Dados para Objetos do Tipo ARVORE (GENERALIZAÇÃO/HIERARQUIA)	77
4.6 - Estrutura de Dados para o Registro ENTIDADE	80
4.7 - Sequência de Gravação para os Registros referentes ao Esquema de uma ENTIDADE	80
4.8 - Estrutura de Dados para o Registro RELACIONAMENTO	81
4.9 - Sequência de Gravação para os Registros referentes ao Esquema de um RELACIONAMENTO	81
4.10- Estrutura de Dados para o Registro AGREGAÇÃO	83
4.11- Sequência de Gravação para os Registros referentes ao Esquema de uma AGREGAÇÃO	83

4.12 - Estrutura de Dados para Registros do Tipo GENERALIZAÇÃO ou HIERARQUIA	84
4.13- Sequência de Gravação para os Registros referentes ao Esquema de uma GENERALIZAÇÃO ou HIERARQUIA	84
4.14- Estrutura de Dados para o Registro ATRIBUTO	84
4.15- Estrutura de Dados para o Registro ARVORE	85
4.16- Estrutura de Dados para o Registro LISTA DE RELACIONAMENTOS	85
4.17- Estrutura de Dados para o Registro LISTA DE NÓS-FILHO	86
4.18- Exemplo para mostrar Especificação e Armazenamento de Esquemas E-R	87
4.19- Tela-exemplo (1) para Geração de Objeto E-R (ENTIDADE)	88
4.20- Tela-exemplo (2) para Geração de Objeto E-R (ENTIDADE)	89
4.21- Estrutura de Dados para a Entidade C	92
4.22- Estrutura de Dados para a Entidade E	93
4.23- Estrutura de Dados para a Entidade G	94
4.24- Estrutura de Dados para a Generalização C	95
4.25- Estrutura de Dados para a Generalização E	96

1 - INTRODUÇÃO

Este capítulo dá uma breve descrição de trabalhos publicados sobre modelagem e sistemas utilizando formalismos Entidade-Relacionamento.

1.1 - VARIÁÇÕES DE MODELOS E-R

Até o surgimento da proposta de [CHEN76] para modelagem do mundo real em termos de entidades, relacionamentos e atributos, os bancos de dados existentes não eram capazes de expressar em correspondência direta os objetos de uma aplicação. Na realidade, o ambiente era moldado para um modelo de estrutura mais ou menos rígido.

Algumas das vantagens do Modelo Entidade-Relacionamento (E-R) segundo [CHUN83] são: permitir capturar e preservar algumas das importantes informações semânticas do mundo real, possibilitar linguagens de computador baseadas em conceitos mais naturais para usuários não-técnicos que as linguagens de manipulação de dados de sistemas convencionais, e a facilidade de tradução entre o modelo E-R e os modelos de dados convencionais. Estas vantagens, segundo o autor, indicam que o modelo deveria ser padronizado para modelagem de esquema conceitual nas arquiteturas ANSI/SPARC e para manipulação da tradução de dados em ambientes de banco de dados heterogêneos.

Desde a primeira proposta de [CHEN76] para Modelo Entidade-Relacionamento, inúmeras extensões têm sido defendidas ([ELMA85],

[FELD86], [PARE86], [SCHE80], [SMIT77], [TEOR86], [WAGN87]) para este modelo. No entanto, não existe um consenso a respeito da Linguagem de Manipulação de Dados (DML) mais adequada para o Modelo E-R, capaz de englobar todas essas extensões. Outro problema apontado é a fraqueza de definição semântica e a falta de poder para expressar propriedades temporais.

Muitos dos modelos propostos que usam conceitos de entidades, relacionamentos e atributos são distintos entre si nas definições e uso dos conceitos de relacionamento e atributo. Em [CHEN83b] é proposto um quadro (figura 1.1) para classificar os modelos E-R existentes e é discutido como estes modelos podem ser traduzidos de um para o outro. Segundo a classificação apresentada os modelos E-R são agrupados em duas categorias. A primeira chamada "Modelos E-R Generalizados" - GERM, n-ários, permite relacionamentos definidos em mais de duas entidades; e a segunda, chamada "Modelos E-R Binários" - BERM, que permite no máximo duas entidades envolvidas num mesmo relacionamento. Essas categorias sofrem ainda subdivisões dependendo do tratamento dado aos atributos (permitindo atributos tanto para entidades quanto para relacionamentos; só para entidades e não permitindo nenhum atributo).

Em [TEIC83] temos uma retrospectiva dos trabalhos baseados no Modelo E-R-Approach (ERA) e resultados obtidos na sua aplicação. O enfoque do modelo ERA surge como o mais geral, capaz de fornecer uma visão mais natural dos sistemas e de preencher a lacuna entre análise de sistemas, modelagem de dados e análise funcional.

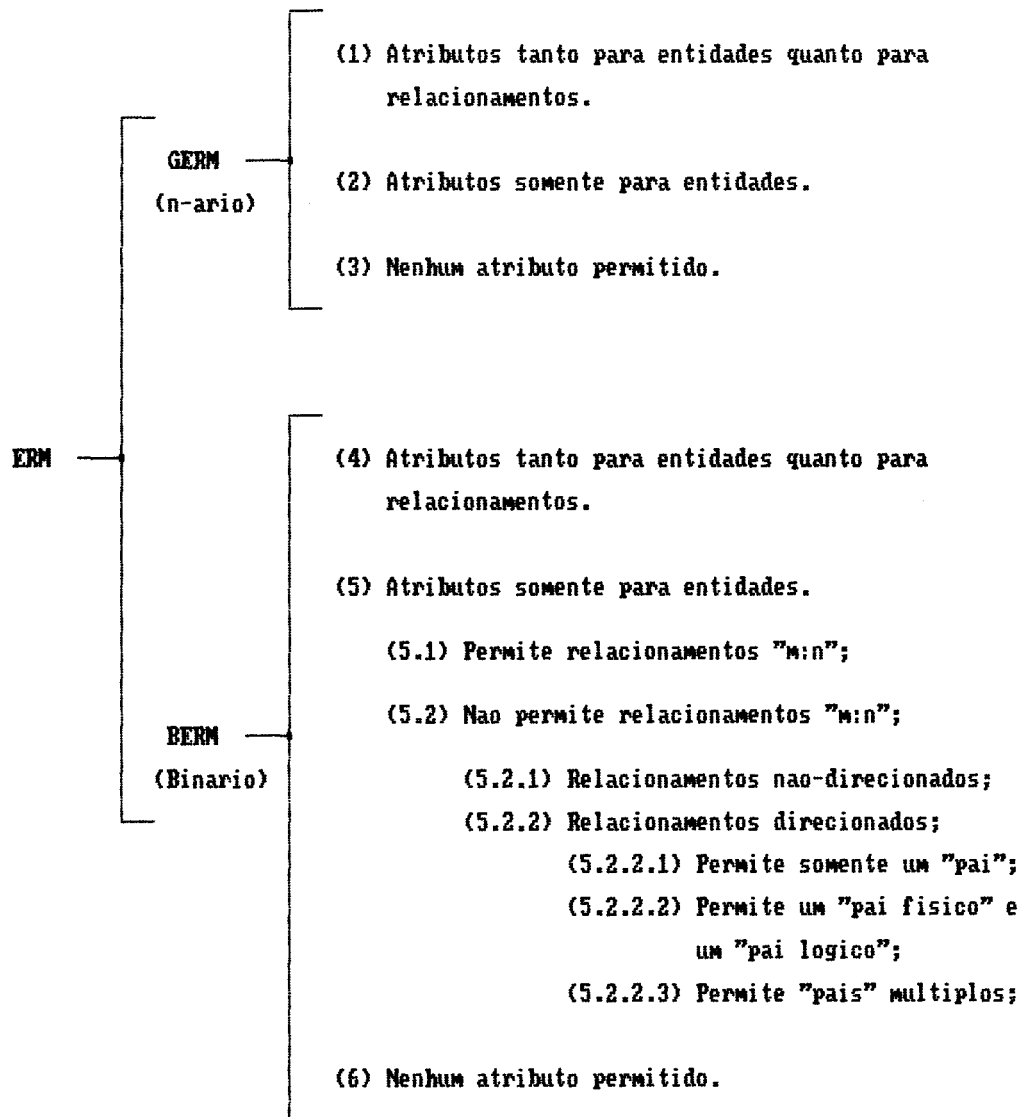


Figura 1.1 - Quadro de Classificação dos Modelos E-R.

Um outro exemplo de variação de modelagem E-R é [ROUS84], que propõe uma metodologia adaptável para projeto de banco de dados.

O método de projeto apresentado nesse artigo baseia-se na filosofia de que o projeto de banco de dados deve ser feito de "fora para dentro". A modelagem conceitual é uma técnica para especificar numa linguagem formal, conceitos e idéias que podem ser interpretadas por outras pessoas familiarizadas com o contexto da empresa e seu ambiente.

A metodologia analisa o comportamento operacional da empresa pois sem uma ampla compreensão de como uma empresa opera, nenhum projeto efetivo pode ser desenvolvido quer para operações correntes quer para futuros melhoramentos.

Outro objetivo da metodologia de "fora para dentro" ("outside-in") é incorporar técnicas reconhecidas e comprovadas como: técnicas de fluxo de dados, de especificação de fluxo de controle, de entrevistas estruturadas e questionários, técnicas de modelagem de dados bem-compreendidas e de otimização lógica.

A modelagem de dados conceitual consiste na identificação das entidades, propriedades (atributos) e relacionamentos envolvidos na aplicação, que podem ser expressos em uma linguagem de vários níveis. O primeiro nível de sintaxe da linguagem é uma versão estendida do modelo E-R de Chen, ficando a diferença por conta da adição de quantificadores numéricos (cardinalidades) que mostram as dependências entre os valores dos dados das entidades nos relacionamentos (podendo ser 1:m ou m:m).

O trabalho de [SCHE80] apresenta uma variação do modelo E-R proposto por [CHEN76] capaz de expressar relacionamentos entre relacionamentos além de abstrações tais como agregações e generalizações, conforme foram descritas por [SMIT77].

Um artigo anterior desses autores [SCHI79] já examinou as características do modelo E-R e as extensões agregação e generalização, com o objetivo de oferecer visões múltiplas ao usuário através do emprego dessas abstrações.

Nesses trabalhos é feita uma distinção entre abstrações de entidades (que inclui o conceito de generalização) e abstrações de relacionamentos (que inclui o conceito de agregação). Dado que uma agregação é uma abstração de um relacionamento, esse novo objeto (agregação) é associado a outros objetos através de relacionamentos, evoluindo naturalmente do conceito de agregação para relacionamento entre relacionamentos [SCHE80].

Um aspecto importante do trabalho de [SCHE80] é a representação de restrições através de construções intencionais e das propriedades invariantes a elas associadas que devem ser preservadas durante as operações de atualização.

As restrições de integridade abordadas concentram-se em relacionamentos totais e parciais, e relacionamentos fracos.

Para preservar as propriedades invariantes durante as operações de atualização é adotada a estratégia de propagar essas atualizações, sendo definidas regras de propagação para as diferentes construções intencionais do modelo E-R estendido. As regras apresentadas possibilitam determinar o "caminho de propagação" para um ponto de acesso e uma operação de atualização

específicos, mesmo quando ocorrem propagações cíclicas ou sobrepostas.

Uma outra metodologia para projeto lógico de banco de dados relacionais usando uma extensão do modelo E-R é apresentada em [TEOR86]. Os passos básicos desta metodologia são a análise dos dados e sua modelagem usando um modelo E-R estendido (que inclui relacionamentos opcionais, ternários, hierarquias de subconjuntos e de generalizações); transformação dos diagramas E-R obtidos em relações candidatas, usando um conjunto de regras de mapeamento, sendo ainda nesta fase eliminadas as relações redundantes; e normalização das relações candidatas, podendo eliminar redundâncias desde que seja preservada a integridade dos dados.

A análise e modelagem dos dados implica em classificar entidades e atributos (sendo fornecidas algumas diretrizes para facilitar a identificação dos papéis desempenhados por essas construções na aplicação modelada); identificar as hierarquias de subconjuntos e generalizações; definir relacionamentos (especificando grau, conectividade, classe dos membros e atributos); e por fim, integrar as múltiplas visões de entidades, atributos e relacionamentos, usando as ferramentas semânticas do modelo E-R estendido, sejam agregação, generalização e identificação de sinônimos.

As regras básicas de mapeamento utilizadas na metodologia são: transformar cada entidade em uma relação contendo atributos chave e não chave; cada entidade em uma hierarquia de generalização ou subconjunto é transformada em uma relação que contém a chave da

entidade genérica, a relação correspondente à entidade genérica contém os atributos comuns às outras entidades, e as demais relações contém os atributos específicos de cada uma das entidades subtipo; cada relacionamento binário do tipo $m:m$ é transformado em uma relação com as chaves das entidades que associa e os atributos do relacionamento, outros relacionamentos ($1:1$, $1:m$) são mapeados pela adição da chave estrangeira na relação apropriada, e os relacionamentos ternários (ou n -ários) são convertidos em uma relação contendo as chaves de todas as entidades envolvidas.

A fase de normalização das relações consiste em analisar as dependências funcionais e multivaloradas associadas às relações, e eliminar as redundâncias que ocorrem entre relações normalizadas.

Esta tese inclui as fases de modelagem dos dados em termos de objetos E-R e efetua o mapeamento para relações com regras semelhantes às adotadas pela metodologia proposta em [TEOR86]; no entanto, não abrange a fase de normalização das relações sendo indicada como um dos melhoramentos a serem adicionados ao sistema futuramente.

1.2 - APLICAÇÃO DA MODELAGEM E FERRAMENTAS

O modelo E-R é encontrado em larga escala em vários tipos de situações. Um exemplo de aplicação específica é [DELG86] que propõe uma extensão do modelo para tratamento de projetos PAC, mais tarde ampliado em sua tese de mestrado [DELG87]. Detalhes de uma parte da implementação foram posteriormente discutidos em

[RICA87]. Enquanto a proposta apresentada nesta tese é mapear operações E-R sobre o modelo relacional, o trabalho desses autores parte da criação de um SGBD específico, voltado para manuseio de objetos E-R.

O estágio atual desse projeto é apresentado em [MAGA89], onde são descritos os aspectos de implementação dos sistemas UnICOSMOS (que constitui o núcleo do sistema Gerenciador de Dados para PAC - GERPAC) e do próprio GERPAC.

O sistema UnICOSMOS compreende facilidades para controle de informação através da adoção de classes, permitindo a definição de instâncias como membros de uma classe, a definição de classes formadas a partir de outras e a associação entre classes.

As classes especificadas são do tipo objeto simples, conjunto e relação. A arquitetura do UnICOSMOS é constituída de domínios internos que armazenam as informações referentes às instâncias, aos tipos de objetos e às associações binárias existentes entre classes.

O sistema GERPAC oferece funções que possibilitam operar sobre as primitivas do modelo; efetuar operações mais específicas como por exemplo, consultas de restrições associadas e estruturais, definição de papéis e cardinalidades; e também verifica e valida as regras de consistência implícitas no modelo.

Além de primitivas básicas, o sistema suporta extensões tais como: esquema e agrupamento (agregação, generalização e objeto complexo). O sistema GERPAC permite também a definição de regras, mas este módulo ainda não foi implementado. Outros pontos para estudo futuro englobam modelos orientados a objetos, o tratamento

de restrições de consistência e integridade, e o desenvolvimento de uma linguagem gráfica para definição e manipulação de dados.

Outro exemplo de aplicação é o modelo EAS e a linguagem EAS-E, sistemas experimentais de desenvolvimento de aplicações baseados no formalismo de entidades, atributos, conjuntos(set) e eventos. Segundo [MARK83] as visões ER e EAS são similares no aspecto que possibilitam ao usuário modelar o mundo em vez de obrigá-lo a especificar como o computador deve representar este mundo e ambas fornecem um conjunto simples mas geral de conceitos de modelagem. O poder de cada visão é também sua fraqueza. Na visão EAS é necessário que o usuário declare antecipadamente os relacionamentos, o que leva à geração de um programa mais eficiente. Ainda comparado ao modelo E-R, o sistema EAS normalmente envolve mais problemas quando são efetuadas mudanças nas definições do banco de dados.

A implementação de uma linguagem (ERLANG) que suporta um modelo E-R restrito para banco de dados relacional é descrita em [MALH86]. ERLANG é uma linguagem compilada, implementada em um banco de dados relacional padrão (SQL/DS) a fim de usufruir das características de manutenção de integridade dos dados e controle de concorrências, entre outras que estão presentes nesse ambiente. A portabilidade da linguagem é acentuada, sendo possível, a partir de pequenas mudanças, utilizá-la em DB2 ou outros bancos de dados relacionais.

O mapeamento entre o modelo E-R restrito usado e o banco de dados relacional procura minimizar o impacto sobre a base de dados através da adoção das seguintes estratégias: possibilita ao usuário da ERLANG utilizar o banco de dados da mesma forma que outros usuários de comandos SQL; não requer uma rigorosa reorganização do banco de dados existente para viabilizar o uso da linguagem.

Projetistas que utilizam formalismos E-R precisam enfrentar o problema de que este é um modelo semântico, que deve depois ser analisado por administradores de banco de dados para permitir a geração do banco de dados que o represente utilizando algum sistema específico; excessão feita para os sistemas de banco de dados que trabalham com objetos E-R, como por exemplo os sistemas GERFAC [MAG89] e DAMOKLES [DITT87], onde não há essa necessidade. Portanto, embora ao leigo seja facilitado expressar suas especificações no modelo semântico, a implementação para SGBD's específicos fica a cargo de especialistas. Além disto, uma vez gerado o banco de dados, o usuário não mais dispõe das abstrações iniciais.

Dentre trabalhos que abordam soluções para este problema situam-se [SHAV81], que discute uma ferramenta para modelagem conceitual baseada em elementos E-R; o Projeto GAMBIT [BRAE85], que dentre outros pontos facilita ao usuário a definição de restrições de integridade; e o Sistema CHRIS [FURT87], uma implementação em PROLOG de uma interface E-R para um banco de

dados relacional. Alguns detalhes destes três projetos são discutidos a seguir.

O modelo "Entity-Functional" [SHAV81] está fundamentado em três estruturas de dados: entidades, atributos e relacionamentos binários. Uma diferença em relação ao modelo E-R proposto por [CHEN76] é que não permite atributos de relacionamentos. Introduce ainda uma fase de análise funcional com o objetivo de identificar quais entidades e relacionamentos devem ser acessados, em qual ordem, de que forma (sequencial, indexada por chave, etc.) e para que ação (recuperação, criação, remoção, modificação), de forma que o sistema possa executar as tarefas necessárias.

A análise é dividida em duas fases, que inclui a análise das entidades (para determinar os objetos do sistema envolvidos) e a análise funcional para determinar a maneira como são usadas.

Uma vez completado o modelo conceitual de entidades, validado através da análise funcional, é usado como base para construção de um modelo Lógico ou Esquema.

Uma ferramenta para projeto interativo de banco de dados abrangendo desde a fase de geração do esquema até sua implementação é o sistema GAMBIT [BRAE85]. Ele é um dos módulos do sistema de banco de dados LIDAS baseado numa extensão do modelo Relacional voltada para o modelo E-R binário de Chen, e apresenta as relações como um novo tipo de dado, composto por um conjunto de elementos do tipo "registro". Inclui, ainda, tipos de associações estendidas para relacionamentos e generalizações de entidades. O

mapeamento entre os modelos E-R e relacional converte os seguintes objetos: entidades, relacionamentos (com representações implícitas e explícitas no modelo relacional estendido usado) e generalizações de entidades.

Esse sistema permite o desenvolvimento de versões diferentes de uma estrutura de dados e auxilia o usuário a projetar um esquema seguindo o método de refinamentos sucessivos ("top-down").

O usuário é guiado durante o processo de projeto podendo postergar decisões importantes (por exemplo, a definição de um atributo chave), ou definir entidades e relacionamentos em qualquer ordem, quando o sistema se encarrega de supervisionar as inconsistências geradas e informar o usuário de sua existência.

Uma característica importante é que possibilita ao usuário a definição de três classes de restrições de integridade, através de menus: 1. para qual operação deve ser verificada a restrição (inserção, remoção, modificação, ou combinações dessas operações); 2. qual a restrição (expressão, subrotina, restrição única para grupo de atributos, nenhuma); e 3. reação à violação de uma restrição (aviso, erro, gatilho). No entanto, uma desvantagem é a verificação de todas as restrições antes que qualquer operação do banco de dados seja efetuada.

Além de encarregar-se da consistência do banco de dados (verificação das restrições de integridade implícitas e aquelas definidas pelo usuário), mantém o controle das operações de atualização através do uso de tipos abstratos de dados.

CHRIS [FURT87] é um sistema especialista para auxiliar no projeto de sistemas de informação que utilizam banco de dados. A principal contribuição desta ferramenta é cobrir todo o ciclo de projeto, ou seja, mapear a especificação de um esquema em uma interface que transforma as operações em transações corretas. Compreende, ainda, um conjunto de construções/comandos para manter a integridade do sistema.

A estratégia de projeto é dividida em três fases:

1. Projeto Conceitual - onde o usuário especifica a aplicação em termos de objetos do modelo E-R, estendido para incluir hierarquias do tipo "is-a" e um conjunto de restrições de integridade;
2. Projeto Lógico - o esquema E-R obtido na fase anterior é mapeado para um esquema relacional normalizado;
3. Geração da Interface - o esquema relacional é mapeado para comandos SQL DDL, sendo então criada a base de dados. As restrições de integridade são também mapeadas em regras de bloqueio, que indicarão quando rejeitar operações, e regras de propagação, que controlarão a propagação de operações de atualização indicando operações extras que deverão ser acionadas.

Esse trabalho discute como transformar uma transação especificada pelo usuário em uma transação correta, incluindo a questão que envolve certos tipos de restrições de integridade as quais não determinam completamente essa transformação.

O protótipo implementado compreende as três fases de projeto, mas somente na fase de projeto conceitual é estabelecido um diálogo com o usuário; as fases de projeto lógico e geração da interface são totalmente automáticas. Algumas simplificações para agilizar a implementação do protótipo foram impostas, sendo as que seguem as mais importantes: cada entidade só pode ter uma chave; cada entidade só pode aparecer uma vez no lado esquerdo de uma restrição do tipo "is-a"; cada entidade só pode participar com um papel em um relacionamento; um relacionamento pode ser declarado total em relação a não mais que uma das entidades participantes da associação; somente relacionamentos binários podem ser declarados 1:n; na fase de projeto conceitual o projetista só pode escolher entre bloquear ou propagar operações em certos casos de restrições de integridade referencial.

Uma estrutura de dados específica para controle em hierarquias de abstrações é apresentada em [WAGN88].

O método tabular, chamado de Tabelas de Hierarquias (HT), consiste em representar em tabelas as ocorrências legais das entidades que compõem a hierarquia, adicionando informações sobre o relacionamento existente entre as entidades de sub e supertipo, o atributo discriminante das entidades e a característica de herança entre elas. A grande vantagem de manter essas informações é que possibilitam a rápida referência às ocorrências legais dos objetos, minimizando os problemas existentes para programar operações de inserção, remoção e modificação das entidades de uma hierarquia e garantir as restrições de integridade referencial.

São apresentadas também duas novas estruturas de acesso (as árvores AB e AB+S) com o objetivo de reduzir o número de acessos lógicos necessários para pesquisar dados em objetos de hierarquias e facilitar a manutenção das restrições de integridade. O método de indexação da árvore de Abstração B (árvore AB) é uma variação do tipo de índice da árvore B. A principal diferença entre as duas está na estrutura das entradas no nível das folhas da árvore. A árvore AB trabalha com um índice único para a hierarquia como um todo, em vez de usar índices separados para cada arquivo de dados. A árvore AB+S (árvore de Abstração B + Conjunto) é uma variação da árvore AB para eliminar deficiências existentes na consulta de conjuntos de objetos de um certo tipo, no banco de dados.

O uso das Tabelas de Hierarquias facilita refinamentos de hierarquias, pois para criar alguma restrição sobre qualquer entidade de uma hierarquia basta retirar ou acrescentar uma linha na tabela correspondente a essa hierarquia. Muitas vezes esses refinamentos implicam na criação de entidades abstratas no modelo conceitual que permitam a visualização do refinamento introduzido. Isto pode ser traduzido na Tabela de Hierarquia com a adição de uma nova coluna. A informação sobre a natureza da entidade (concreta ou abstrata) também está presente nas HTs, sendo utilizada para evitar acessos desnecessários a relações inexistentes que corresponderiam a entidades abstratas.

Esta tese não se preocupa com o conceito de entidade 'concreta' ou 'abstrata', nem caracteriza subconjuntos totais/parciais; sendo a diferenciação entre subconjuntos mutuamente

exclusivos e não-exclusivos explicitada com a adoção dos objetos 'generalização' e 'hierarquia de tipos'.

Para realizar operações de inserção/remoção/modificação em uma determinada relação, a árvore dispõe de um 'bit map' que fornece uma lista de todas as relações que devem ser também modificadas para manter a integridade do Banco de Dados. Nesta tese, as estruturas de dados propostas permitem deduzir esse caminho, mas o uso do 'bit map' das árvores AB e AB+S fornece diretamente as relações a serem modificadas.

Um trabalho recente sobre mapeamento ER - relacional aliado à análise de manutenção de dependências de inclusão é [CASABB] que descreve um monitor para manter as dependências de inclusão encontradas em um diagrama ER quando mapeado para o modelo relacional. O monitor apresentado utiliza duas estratégias básicas para garantir restrições de integridade a fim de manter o estado do banco de dados consistente de acordo com o esquema relacional: modificação dos comandos da linguagem de manipulação de dados e propagação da operação requerida usando dados coletados durante uma transação.

O monitor descrito estuda os problemas que envolvem o bloqueio ou a propagação de operações de remoção e inserção em esquemas com dependências de inclusão. Cada operação submetida pelo usuário é modificada para preservar a restrição existente, no caso de bloqueio da propagação. Caso a opção seja de propagação, são coletados os valores necessários para propagar a atualização e só então é executada a operação. Quando a transação é terminada pelo

usuário, são executados todos os gatilhos resultantes das operações de propagação usando os valores coletados durante o processo.

Um gatilho é implementado através de uma rotina que é acionada quando são satisfeitas as suas pré-condições. Gatilhos são frequentemente usados para manutenção de restrições de integridade.

Um problema abordado por [CASABB] é como ordenar as operações submetidas pelo monitor, o que pode levar a erros ou processamento desnecessário tanto na modificação de comandos quanto na propagação. A solução apontada é fixar uma ordem para atualizar de forma correta as relações e forçar o usuário a segui-la (caso de operação com opção de bloqueio), e postergar o acionamento de gatilhos advindos de operações de propagação até a finalização da transação, sendo mantido para tal o conjunto de valores necessários para disparar tais gatilhos.

A estratégia utilizada pelo monitor para definição de quais gatilhos ativar consiste em verificar se uma atualização viola determinada restrição e aplicar a opção de bloqueio ou propagação mais apropriada. Além disto, o monitor testa se a operação submetida requer a modificação de algum dos conjuntos de valores que são mantidos para disparar gatilhos posteriores.

Outro problema discutido é a interferência de gatilhos, solucionado através da modificação dos valores mantidos para disparo de gatilhos posteriores, sendo dada preferência a gatilhos de inserção para evitar o disparo de gatilhos de remoção desnecessários.

Algumas restrições impostas nesse trabalho são que não é permitido atualização de valores de atributos-chave e que o SGBD tem o encargo de garantir chaves, verificação de tipos e ausência de valores nulos.

1.3 - ORGANIZAÇÃO DA TESE

A presente tese enfoca alguns dos aspectos mencionados nos parágrafos anteriores, tentando solucionar o problema de permitir ao usuário a utilização de formalismos ER para manipular bases de dados. A solução proposta consiste em apresentar uma interface de uma extensão do modelo E-R que é mapeada em manipulação de um banco de dados relacional. REVER (RElacional Via Entidade-Relacionamento) é a interface proposta, que pretende ampliar o universo de usuários que virão a usufruir das qualidades do Banco de Dados Relacional sem que para isso lhes seja impingida a necessidade de se especializarem nessa área. Como resultado, o usuário poderá desfrutar das vantagens da modelagem semântica E-R aliadas ao rigor teórico existente para o Modelo Relacional.

Pressupõe-se que o usuário da interface já haja procedido à análise do ambiente e suas necessidades, havendo elaborado a especificação do sistema, ou esquema funcional [ROUS84]. Uma vez estabelecido um modelo funcional, ou seja, quando identificadas as principais atividades ou funções e as associações existentes entre elas, pode-se especificar um esquema conceitual E-R. De posse desta especificação, o usuário pode passar a utilizar a interface.

Em outras palavras, a interface não auxilia o usuário a modelar uma aplicação; restringe-se a permitir a interação entre a modelagem de um problema através do modelo E-R e a sua implementação relacional subjacente.

Um dos objetivos deste trabalho é integrar técnicas de projeto de banco de dados e de projeto de programas, dado que um sistema de banco de dados não é só um esquema (componente estrutural) ou só um conjunto de transações (componente procedimental), mas sim um esquema com transações sendo aplicadas a este esquema [ROUS84].

Elmasri [ELMA83], em sua proposta de uma linguagem (GORDAS) para o modelo E-R, classifica os diferentes tipos de interfaces em: procedimentais, dedicadas a profissionais para definição dos programas de aplicação que serão usados pelos usuários; não-procedimentais, para interação de usuários técnicos e semi-técnicos com o SGBD a partir de um terminal; e amigáveis, que inclui menus para orientar usuários não-técnicos e uso de linguagens naturais. O sistema REVER utiliza menus para interagir com o usuário.

Este capítulo fez uma breve descrição de alguns aspectos da modelagem E-R, levando à motivação para a tese. A tese possui 4 capítulos adicionais. O capítulo 2 trata dos conceitos e primitivas do modelo E-R estendido utilizado e seu mapeamento para o modelo Relacional. O capítulo 3 aborda a especificação da

interface, sendo discutidas as linguagens de definição dos objetos do modelo e de consulta/atualização. O capítulo 4 trata de alguns detalhes do protótipo implementado. O capítulo final apresenta problemas para pesquisas futuras e conclusões.

2 - CONCEITOS E NOMENCLATURA

Este capítulo pressupõe que o leitor já conheça as características básicas do modelo E-R tradicional e sua representação gráfica. Serão apenas ressaltados alguns termos utilizados na tese. Para maiores detalhes, sugere-se a leitura de [CHEN76].

2.1 - *MODELO ENTIDADE-RELACIONAMENTO*

Pode-se exprimir qualquer "acontecimento", ou qualquer dado do universo - sistema de informações, através de "Entidades" e "Relacionamentos".

Uma "entidade" representa um conjunto de ocorrências de informações de uma mesma espécie, com características comuns: "atributos". Por exemplo, na figura 2.1, a entidade 'MEDICO' indica a coleção de dados ('atributos') que dizem respeito à classe médica composta por um número variável de médicos, como por exemplo: NOME, CPF, CRM, ESPECIALIDADE, HOSPITAL/CLINICA, END_RES, FONE_RES, END_HOS, FONE_HOS, etc. As diferentes ocorrências de uma entidade são distinguidas através de um ou mais atributos identificadores ("chave"), escolhido(s) dentre seus atributos por caracterizar de forma única cada ocorrência.

Atributos podem ser compostos ou multivalorados. Atributos compostos são aqueles formados por mais de um atributo, identificadores ou não (descritores); atributos multivalorados

ocorrem quando para um único valor de atributo identificador corresponde mais de um valor de atributo descritor ou não-chave.

Um "relacionamento" é a representação das associações que ligam uma (auto-relacionamento) ou mais entidades, podendo ter atributos próprios. Por exemplo, na mesma figura, dadas duas entidades ('MEDICO' e 'ENFERMEIRO'), o relacionamento 'TRABALHA_COM' associa as entidades indicando quais médicos trabalham com quais enfermeiros. Este relacionamento possui o atributo NUMERO_ENFERMARIA, que indica para cada par (MEDICO-ENFERMEIRA), o(s) número(s) de enfermaria em que trabalham juntos.

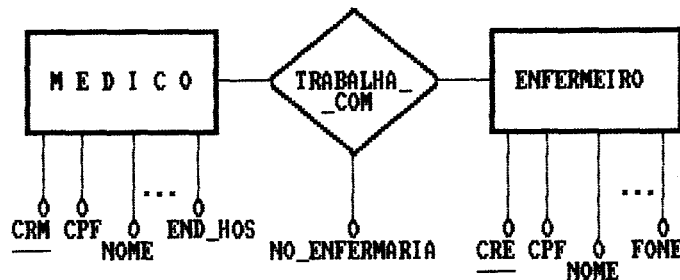


Figura 2.1 - Esquema-exemplo mostrando Entidade, Relacionamento e Atributo.

Um dos aspectos mais importantes na discriminação de diferenças entre modelos originalmente baseados no modelo proposto por Chen é a definição dos relacionamentos. Um exemplo são os conceitos de 'grau' (unário, binário, ternário), 'conectividade' (1:1, 1;n, m:n) e 'classe-de-membro' (mandatório ou opcional) dos relacionamentos, conforme adotados por [TEOR86].

Um relacionamento (entre entidades A e B) é dito "total" de B em relação a A quando a existência de cada ocorrência da entidade B "depende" da existência de uma ocorrência correspondente da entidade A. Essa entidade com dependência de existência é chamada de entidade "fraca" no modelo original de Chen. Na extensão E-R adotada, não se identifica uma entidade fraca, mas sim o relacionamento total, indicado graficamente através de um círculo escuro conforme exemplifica a figura 2.2.

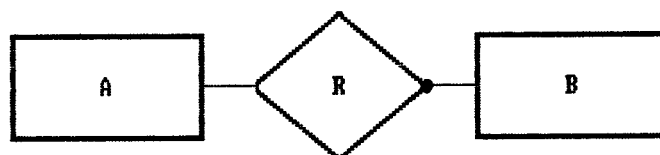


Figura 2.2 - Esquema-exemplo mostrando Relacionamento Total.

A modificação de representação (em função do relacionamento) é particularmente interessante, pois marca o relacionamento que estabelece a relação de dependência de existência, evitando ambiguidades quando essa mesma entidade participa de outros relacionamentos onde tal dependência não existe.

Caso a existência de uma entidade seja completamente independente da existência de outra entidade, o relacionamento que as interliga é dito "parcial".

O número de ocorrências de uma determinada entidade no contexto de um relacionamento, chamado "cardinalidade", está contido dentro de limites inferior e superior explicitados no diagrama do modelo E-R. Na figura 2.3, o mapeamento entre as entidades é de 1,1:0,n, ou seja, a cada ocorrência da entidade 'PAI' correspondem no mínimo zero a no máximo n ocorrências da entidade 'FILHO'.

As mesmas entidades podem estar associadas através de outros relacionamentos com cardinalidades distintas.



Figura 2.3 - Esquema-exemplo mostrando Cardinalidade.

2.2 - MODELO E-R ESTENDIDO

Das inúmeras contribuições ao modelo E-R ([DELG87], [ELMA83], [ELMA85], [FELDB6], [MALH86], [PARE86], [SCHE80], [SHAV81], [SMIT77], [TABO83], [WAGN87], [WEBR83]) existentes, a de Wagner e Medeiros [WAGN87] foi a adotada pela tese. Esta extensão do modelo reuniu um grupo de proposições cujo objetivo principal é

ampliar a riqueza semântica do modelo através do aumento dos níveis de abstração.

Assim, além dos elementos já expostos na seção 2.1, o modelo E-R estendido aqui utilizado será composto de blocos adicionais mais complexos formados por composição das primitivas do modelo original: 'Agregação', 'Generalização' e 'Hierarquia de Tipos'.

A figura 2.4 mostra a simbologia usada para tais elementos conforme foi proposta por Navathe e Cheng [NAVA83]. Maiores detalhes e exemplos da extensão adotada podem ser encontrados em [WAGN87].

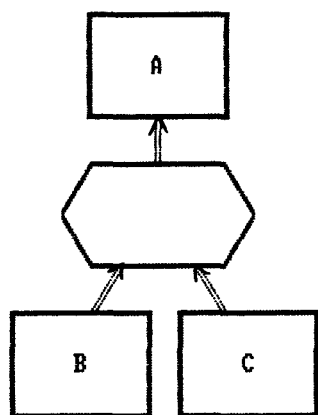


Fig. 2.4a: GENERALIZACAO

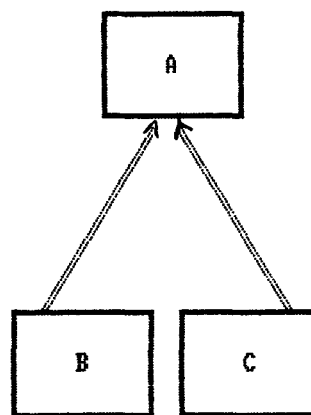


Fig. 2.4b: HIERARQUIA DE TIPOS

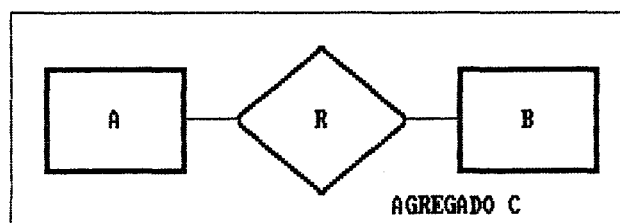


Fig. 2.4c: AGREGACAO

Figura 2.4 - Simbologia usada para Generalizacao, Agregacao e Hierarquia de Tipos.

Entende-se "agregação" como o encapsulamento de uma ou mais entidades e um relacionamento que as interliga (figura 2.4c). Na realidade, as ocorrências da agregação correspondem às associações expressas pelo relacionamento que encapsula. Na figura 2.5 temos a agregação 'ORQUESTRA' composta pela entidade 'MAESTRO' que 'REGE' a entidade 'INSTRUMENTOS'. Podem existir ocorrências de INSTRUMENTOS que não sejam REGIDOS por nenhum MAESTRO, logo não pertencem à agregação ORQUESTRA.

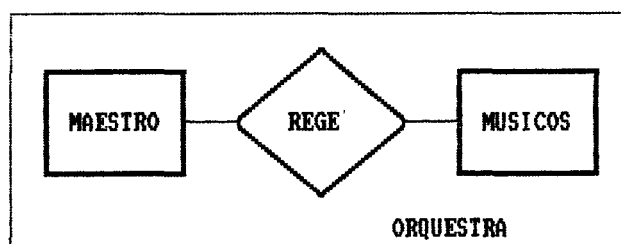


Figura 2.5 - Esquema-exemplo de Agregação.

A entidade complexa "generalização" pode ser caracterizada por uma árvore de dois níveis onde a raiz engloba os argumentos comuns a um conjunto de elementos (as folhas). Em outras palavras, o elemento-raiz "generaliza" as características dos elementos-filho. Cada elemento-folha tem como atributos aqueles herdados da raiz, bem como atributos que o distinguem das demais folhas (figura 2.6). A cada ocorrência de um elemento-raiz corresponde uma ocorrência de alguma folha; as folhas correspondem a conjuntos mutuamente exclusivos. Diz-se que a raiz

é uma "generalização" das folhas e que estas são uma "especialização" da raiz. Esta caracterização é recursiva, ou seja, cada folha pode ser por sua vez a raiz de uma nova árvore: o grafo correspondente adquire assim as características de uma árvore de nós heterogêneos.

Esta definição de componentes de uma generalização é semelhante àquela aplicada por [TEOR86] e [BRAE85]. Em [WELD80] a generalização é representada por um objeto geral com as características comuns de uma coleção de objetos (mais específicos e mutuamente exclusivos entre si). A existência de categorias não exclusivas implica em diferentes 'agrupamentos' de generalizações, sendo que em cada agrupamento as categorias são distintas.

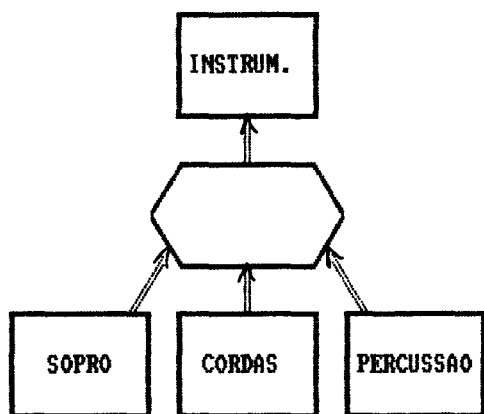


Fig. 2.6: Esquema-exemplo de Generalizacao.

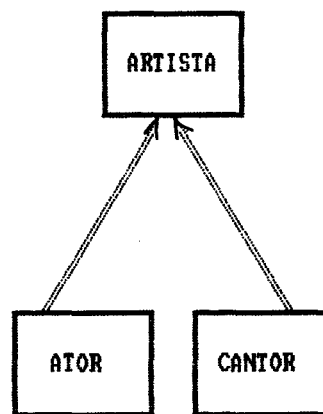


Fig. 2.7: Esquema-exemplo de Hierarquia de Tipos.

A generalização tem uma variante denominada "*hierarquia de tipos*" em [WAGN87]. Se em generalização ocorre exclusão mútua entre os elementos-filho participantes, o mesmo não é necessariamente verdade em "*hierarquia de tipos*", daqui por diante referenciada simplesmente como "*hierarquia*". No exemplo da figura 2.7, os conjuntos de ocorrências de 'ATOR' e 'CANTOR' podem não ser mutuamente exclusivos pois um determinado 'ARTISTA' pode ser ATOR e CANTOR ao mesmo tempo. O trabalho de [FURT87] implementa o conceito de hierarquia de tipos que também é conhecida por "*hierarquia is-a*". Em [TEOR86] temos o mesmo conceito indicado como 'hierarquia de subconjuntos' onde as ocorrências de entidades-folha não possuem exclusão mútua, ocorrendo sobreposição de papéis sem que haja qualquer prejuízo da estrutura; os únicos atributos repetidos são no caso aqueles identificadores das entidades, ou chaves.

Muitas vezes, generalização e hierarquia são chamadas indistintamente de "generalização", sendo acrescida a qualificação "E" (mutuamente exclusiva) ou "O" (overlapping, ou seja, sem exclusão mútua). Além disso, pode-se especificar se a descendência pai → filhos é total ou parcial. Uma dependência é total quando a cada ocorrência do pai corresponde obrigatoriamente (ao menos) uma ocorrência de algum filho. Uma descendência é parcial quando esta obrigatoriedade não existe.

Segundo tal notação, é permitido a cada nível de uma generalização ter uma combinação diferente desta classificação (T,E), (T,O), (P,E), (P,O), caracterizando uma árvore em que, a cada nível, o tipo de herança pode ser diferente. Os problemas

de representação e propagação de atualizações em árvores com estas características são estudados em [WAGN88].

Nesta tese, é feita a hipótese de que árvores têm um único tipo de herança em todos os nós: (T,E), quando são chamadas de generalização ou (T,O), quando são chamadas hierarquias. As razões desta simplificação serão vistas na descrição da interface e mapeamento e têm por objetivo diminuir os problemas de propagação de atualizações.

A princípio nada impede a existência de árvores cujos nós são relacionamentos (vide [WAGN87]), ou mesmo árvores heterogêneas (onde os nós correspondem a elementos de natureza diversa, por exemplo, folhas compostas por hierarquias e agregações). Nesta tese, todas as árvores são supostas homogêneas. Permite-se, no entanto, árvores de agregações, o que corresponde a árvores onde a semântica dos nós está ligada aos relacionamentos encapsulados.

2.3 - MODELO RELACIONAL

Esta seção discute apenas os conceitos do Modelo Relacional relevantes à tese. Para maiores detalhes, sugere-se a leitura dos capítulos 1 a 8 de [MAIE83].

Simplificadamente, um banco de dados relacional é composto por um conjunto de relações, também chamadas tabelas. Um "esquema" $R(i)$ é formado pelo conjunto de (nomes de) atributos $\{a(1)...a(n)\}$ onde cada um destes é definido sobre um domínio,

$\text{dom}\{a(i)\}$. Um "esquema relacional" RR é composto de um conjunto de esquemas $R(i)$. Por analogia, esta tese denomina esquema E-R ao projeto conceitual no modelo E-R estendido, onde se especificam os objetos do projeto e suas interligações, sem preocupação com ocorrências.

Uma "relação" $r(i)$ é uma ocorrência de um esquema $R(i)$. Diz-se também que uma relação é uma tabela formada por um conjunto de tuplas $\{t\}$. Uma "tupla" $t(i)$ é um conjunto de valores ordenados segundo os atributos do esquema da relação, com significação válida dentro do domínio dos atributos correspondentes. De novo, por analogia, a tese referencia "ocorrências" de um objeto E-R: por exemplo, para uma entidade MEDICO, cada ocorrência é o conjunto de dados relativo a um determinado médico. Um "banco de dados relacional" é um conjunto de relações $\{r(i)\}$. Um estado do banco de dados é uma ocorrência do esquema relacional RR , sobre o qual está definido.

Um atributo ou conjunto de atributos constitui uma "chave" X do esquema $R(i)$ $\{a(1)..a(n)\}$ se $X \subseteq \{a(1)..a(n)\}$ e não existem duas tuplas em qualquer $r(i)$ com o mesmo valor de X , e ainda, nenhum subconjunto de X tem esta propriedade.

Um atributo Y é "funcionalmente dependente" de um ou mais atributos X se não é válido existir duas tuplas com o mesmo valor de X mas diferentes valores para Y .

Dados dois atributos X e Y de R ocorre uma "dependência funcional" (FD), $X \rightarrow Y$, se a cada valor de X em R corresponde exatamente um valor de Y . A existência de duas tuplas com o mesmo valor de X implica na repetição dos valores de Y .

Um relacionamento total é tratado nesta tese como um tipo de "dependência de inclusão". O nome advém do fato de que a inclusão de uma tupla em uma relação depende da existência de uma tupla correspondente em outra relação (aquela à qual está ligada através do relacionamento total). De uma forma mais geral, dado um conjunto de atributos $A = \{a(1) \dots a(n)\}$ e dois conjuntos de atributos $X \subseteq A$ e $Y \subseteq A$, a dependência de inclusão de X em Y ocorre se $X \subseteq Y$. A dependência é não-trivial se $X \subsetneq Y \subseteq A$. No caso de um relacionamento total, sejam B e C os conjuntos de atributos de dois esquemas $R(i)$ e $R(j)$; se existir $X \subseteq B$ e $Y \subseteq C$ tal que $X \subseteq Y$, então o relacionamento é total de $R(j)$ para $R(i)$.

O estudo de dependências de inclusão no contexto de modelagem entidade-relacionamento vem recebendo muita atenção, por exemplo [CASAB88, BRAG88]. A razão é que, além de aparecerem no contexto de relacionamentos totais, dependências de inclusão são utilizadas para expressar herança de propriedades em árvores (já que a existência de uma ocorrência em um nó filho é condicionada à existência da ocorrência no nó pai). Além disto, ocorrências em relacionamentos (e agregações) são também sujeitas a dependências

de inclusão, já que são condicionadas à existência de ocorrências nas entidades por eles relacionadas.

2.4 - RESUMO

Este capítulo descreveu formalismos e notação que serão usados no resto da tese, tanto do modelo E-R quanto do relacional.

O próximo capítulo fornece a especificação funcional da interface REVER.

3 - ESPECIFICAÇÃO DA INTERFACE REVER

3.1 - DESCRIÇÃO GERAL

Esta tese especifica uma interface que permite ao usuário definir e manipular objetos E-R e suas ocorrências, traduzindo suas solicitações sobre objetos E-R em operações sobre um sistema relacional.

A arquitetura proposta pode ser descrita pelo diagrama 3.1 a seguir, sendo composta de quatro componentes principais:

- * Módulo de DIALOGO;
- * Módulo GERADOR-ESQUEMA-OBJETOS;
- * Módulo MANUSEADOR-OBJETOS;
- * Módulo TRADUTOR - DML/DDL.

Para facilitar a leitura do texto, estes componentes serão denominados DIALOGO, ESQUEMA, MANUSEADOR e TRADUTOR.

O módulo DIALOGO é o que trata da interação direta com o usuário. Este módulo intercepta as solicitações do usuário e ativa os módulos necessários para seu atendimento (criação/manipulação de esquema; consulta/manipulação de ocorrências E-R). O protótipo implementado nesta tese optou por um diálogo interativo através de menus. Uma das características é agilizar o processo de definição, dirigindo o usuário para o leque de opções adequado em cada momento. Este tipo de arquitetura torna o sistema independente do tipo de interação com o usuário. Caso se deseje uma interação de

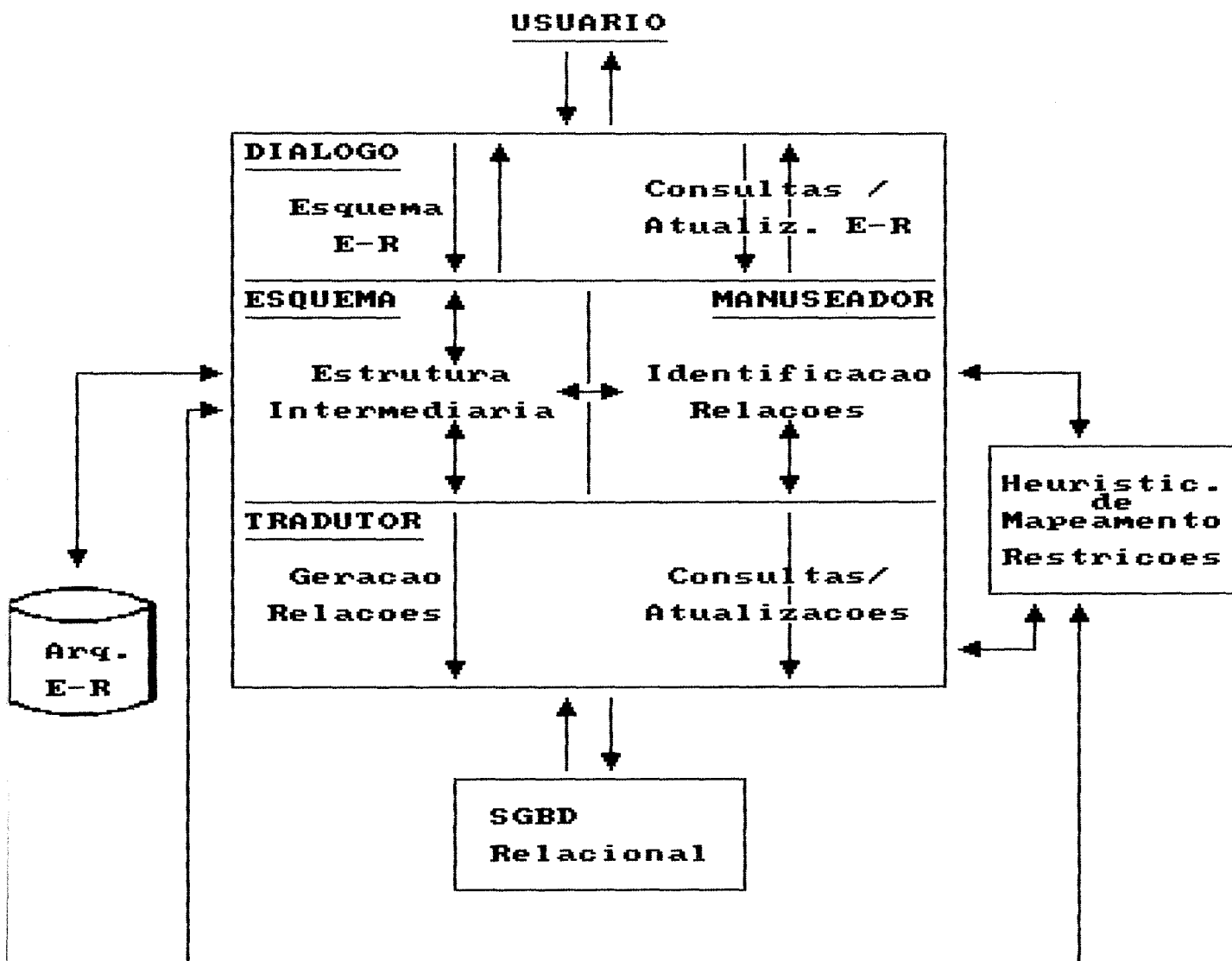


Figura 3.1 - Arquitetura proposta da Interface REVER.

outro tipo (por exemplo, através de interface gráfica), basta acoplar um novo módulo DIALOGO ao sistema, que permita tal interação, sem com isto afetar os demais componentes.

O módulo ESQUEMA tem duas funções básicas: permitir ao usuário gerar e modificar o esquema E-R; e manter a descrição deste esquema em uma estrutura intermediária que é acessada quando o usuário quer manusear ocorrências de objetos E-R. Este módulo efetua a tradução de esquemas E-R para relacionais.

Dentre as suas características gerais desejáveis, destacam-se:

- * utilização de esquemas já cadastrados;
- * definição dos esquemas dos objetos e seus atributos;
- * consulta aos esquemas dos objetos (subdividida em consulta aos seus atributos e consulta às estruturas de que participa);
- * modificação dos esquemas dos objetos, seja de seus atributos, seja de outros campos que envolvem as ligações das quais participa;
- * eliminação de partes de esquema.

No capítulo 4, será mostrado que todas estas características foram mantidas no protótipo implementado.

O módulo MANUSEADOR traduz as solicitações do usuário de manuseio de ocorrências de objetos E-R em operações sobre relações. Para tanto, utiliza a estrutura gerada e mantida pelo módulo ESQUEMA. Além disto, deve verificar que gatilhos devem ser acionados para manutenção de dependências de inclusão.

Como características importantes temos:

- * processamento de consultas e atualizações a arquivos de dados (relações);
- * determinação das relações e dados envolvidos numa operação de consulta ou de atualização de forma totalmente transparente ao usuário.

Finalmente, o módulo TRADUTOR traduz as estruturas internas e solicitações geradas pelos módulos ESQUEMA e MANUSEADOR em comandos para o SGBD hospedeiro.

Esta arquitetura torna o sistema independente do SGBD usado, bastando apenas modificar o módulo TRADUTOR.

O protótipo implementado, descrito no capítulo 4, não inclui o módulo MANUSEADOR e tem apenas uma versão primitiva do módulo TRADUTOR.

3.2 - MAPEAMENTO E-R \longleftrightarrow RELACIONAL

A interface projetada exige que se estabeleçam regras de mapeamento entre o modelo E-R estendido adotado e o modelo relacional. A interface tem dois níveis de mapeamento: para geração do esquema e para consultas/atualizações de relações. No primeiro caso, o usuário especifica seu projeto semântico em termos de objetos E-R e a interface gera o esquema relacional correspondente segundo regras previamente definidas de mapeamento. No segundo caso, o usuário consulta/atualiza ocorrências de objetos E-R, o que é traduzido pela interface em consultas e

atualizações às tuplas das relações base. Há dois aspectos a se considerar: o primeiro se refere a encontrar a correspondência entre os dois modelos; e o segundo busca sintetizar algumas propriedades dos dois modelos em uma forma compreensível [CHEN83a].

O mapeamento entre modelos na fase de geração de esquema precisa considerar o tratamento de atributos identificadores externos (chaves estrangeiras) e o processamento de atributos compostos e multivalorados. No presente trabalho, propõe-se processar as chaves estrangeiras através dos mecanismos para manter as dependências de inclusão. É ignorada a existência de atributos agregados (por exemplo: max, min) ou compostos. Seguindo o proposto por [TEOR86] os atributos multivalorados devem ser transformados em entidades ainda na fase de projeto. Portanto, não há necessidade de mapeá-los como tal para o modelo relacional. Desta forma, a cada atributo do modelo E-R corresponderá um único atributo relacional, todos monovalorados.

A fase de mapeamento de uma especificação E-R para uma operação sobre esquemas e relações engloba a transformação de entidades, relacionamentos e dos objetos complexos: generalização, hierarquia e agregação. O mapeamento de um esquema E-R para um esquema relacional obedece às regras a seguir, que não consideram otimização ou minimização de esquema.

* Cada entidade simples é convertida em uma relação com mesmo nome. O esquema relacional é formado dos atributos da entidade, identificando atributos chave e não-chave.

* Cada relacionamento é transformado em um esquema composto das chaves das entidades que liga, acrescido dos atributos específicos ao relacionamento. Esse mapeamento é adotado porque ressalta as seguintes vantagens:

1. o uso de uma relação separada para relacionamentos permite armazenar ocorrências de relacionamentos sem modificar outras relações do Banco de Dados Relacional;
2. esta implementação permite tratamento uniforme para relacionamentos onde entidades apareçam com qualquer cardinalidade.
3. possibilita de forma simples a criação e remoção de relacionamentos sem modificação das tabelas das entidades. No caso de eliminação de relacionamento total, supõe-se que as entidades não mais dependem uma da outra.

* As entidades complexas Generalização e Hierarquia seguem o padrão de mapeamento adotado por [WELDBO]:

- o nó 'pai' é transformado em um esquema com a chave e atributos desse nó (comuns a todas as entidades 'filho');
- os nós 'filhos' são transformados em esquemas contendo cada um a chave da entidade 'pai' mais os atributos específicos de cada filho. Os atributos não-chave do pai não são reproduzidos em seus filhos, já que se supõe que sejam recuperados através de dependências funcionais e junção de relações.

* A entidade complexa Agregação é convertida obedecendo os mesmos critérios usados para relacionamentos. O mapeamento descrito por [WELD80] neste caso define cada agregação como uma relação contendo como atributos as chaves dos objetos (entidades) encapsulados mais quaisquer dados adicionais do objeto agregação sendo mapeado.

O modelo semântico proposto considera dependências de chave e de existência. Estas dependências são mapeadas, no modelo relacional, para dependências funcionais (dentro de uma relação) e dependências de inclusão (correspondentes a ocorrências de relacionamentos e manutenção de árvores com heranças de propriedades).

A herança de propriedades em uma árvore é garantida pela inclusão da chave do pai entre os atributos dos filhos, sendo mantida portanto através dos mecanismos de integridade associados a chaves. Uma das chaves do filho corresponde obrigatoriamente à chave do pai.

A manutenção de dependências de inclusão e de propriedades como exclusão mútua (de ocorrências de filhos em uma generalização) apresenta problemas teóricos complexos. O artigo de [CASA84] discute uma axiomatização de dependências de inclusão e funcionais. Os prós e contras de manter relações separadas que estejam relacionadas por dependências de inclusão são discutidos em [BRAG88]. A análise teórica destes problemas escapa ao âmbito desta tese. A seção 3.3 apresenta algumas simplificações do trabalho que permitem eliminar alguns destes problemas.

Vale notar que o mapeamento sugerido é adotado também por [BRAG88], que prova formalmente que este mapeamento do tipo 1:1 (um objeto: um esquema) é correto, embora ignore agregações. Enquanto que esta tese se atém ao conjunto de regras definido nesta seção, em [BRAG89] são analisadas outras opções, visando diminuir o número de esquemas relacionais e assim facilitar a manutenção de dependências de inclusão.

Por exemplo, seja uma generalização com n componentes. Dependendo do tipo de propagação de atualização especificada, [BRAG89] mapeia a árvore em um único esquema relacional, propondo que os componentes sejam recuperados através de mecanismos de materialização de visões. Nesta tese, a mesma árvore dá origem a n esquemas relacionais, sendo proposto que as dependências de inclusão que as interligam sejam mantidas através de gatilhos [CASA88].

O enfoque de [BRAG89] facilita manutenção de integridade de ocorrências e o adotado na tese facilita o processamento de consultas sobre ocorrências e modificação de esquema.

Taborier [TAB083] define restrições de integridade como "todas aquelas restrições que devem ser mantidas todo o tempo para que um banco de dados seja válido". Ele as classifica em três categorias: restrições de valor, estruturais e mistas. Existem dois tipos principais de enfoques para especificar restrições de integridade: estático - tenta defini-las diretamente no esquema conceitual; dinâmico - considera as consequências de ações na base de informações (criação, remoção, atualização e recuperação). O

enfoque desta tese é do primeiro tipo, enquanto que o de [BRAG89] é do segundo tipo.

3.3 - SIMPLIFICAÇÕES PARA O PRESENTE TRABALHO

A interface proposta considera apenas relacionamentos binários. As razões para tal são o alto grau de complexidade embutido na definição e representação de relacionamentos n-ários para $n > 2$, e na incapacidade de evitar-se ambiguidade de interpretação em relacionamentos envolvendo entidades fracas e cardinalidades mínima e máxima, usando-se uma simbologia tão reduzida.

As relações criadas a partir da especificação E-R estarão, pelo menos, na Primeira Forma Normal. Este trabalho se restringirá ao manuseio de árvores homogêneas (compostas do mesmo tipo de nó). Árvores de relacionamentos serão igualmente ignoradas, embora se aceite árvores de agregações.

Para minimizar os problemas advindos de manutenção de integridade para dependência de inclusão, proíbe-se a uma dada entidade estar associada a mais de um relacionamento total, considerando também o fato de que a participação em uma árvore já implica uma dependência de inclusão. Este parágrafo resulta nas seguintes restrições:

- * Uma dada entidade só pode estar associada a no máximo um relacionamento total (ou seja, só pode participar de no

máximo uma dependência de inclusão - vide figuras 3.2a, 3.2b e 3.2c). Estas figuras representam configurações proibidas pelo trabalho.

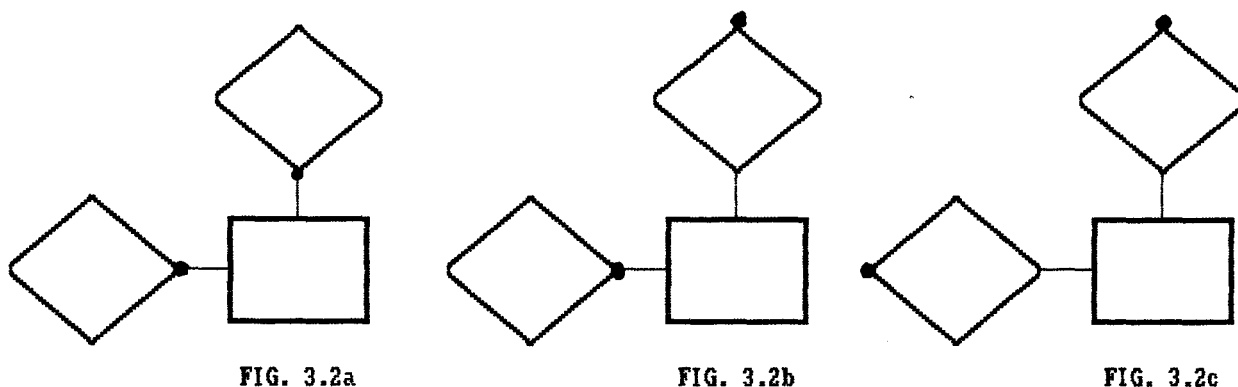


Figura 3.2 - Configurações Proibidas envolvendo mais de um Relacionamento Total.

* Agregações que participem de árvores não podem englobar relacionamentos totais (a participação em uma hierarquia já implica participação em uma dependência de inclusão). A figura 3.3 mostra esta proibição.

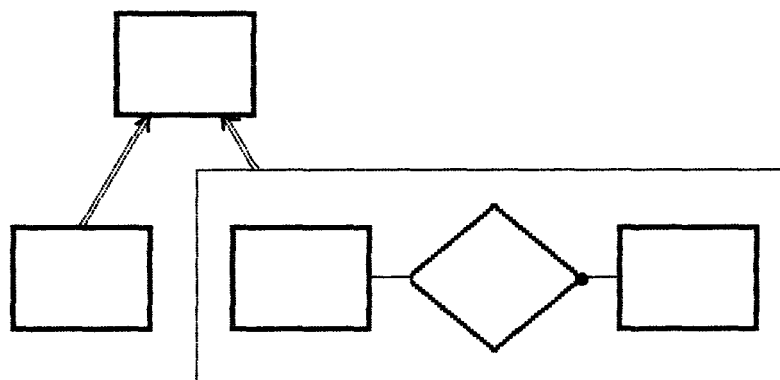


Figura 3.3 - Configuração Proibida envolvendo Agregação com Relacionamento Total e Árvores.

* Agregações que englobem relacionamentos totais não podem por sua vez estar associadas a outros elementos através de relacionamentos totais, conforme figura 3.4.

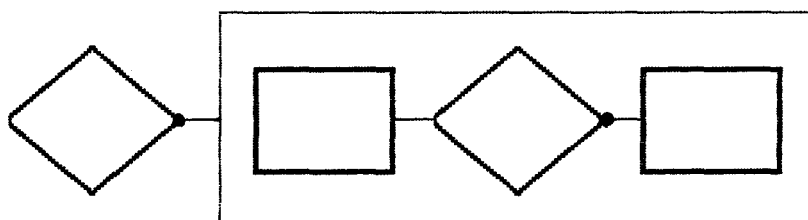


Figura 3.4 - Configuracao Proibida envolvendo Agregacao e Relacionamentos Totais.

* Folhas de uma árvore não podem estar associadas por relacionamentos totais, conforme figura 3.5a.

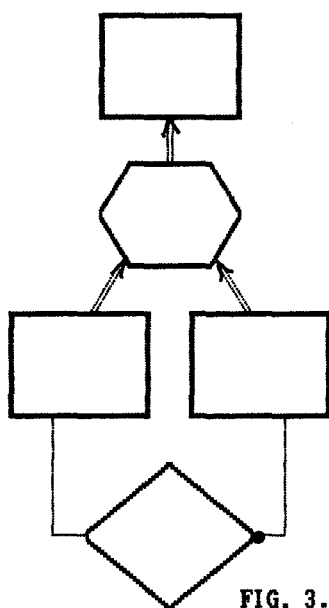


FIG. 3.5a

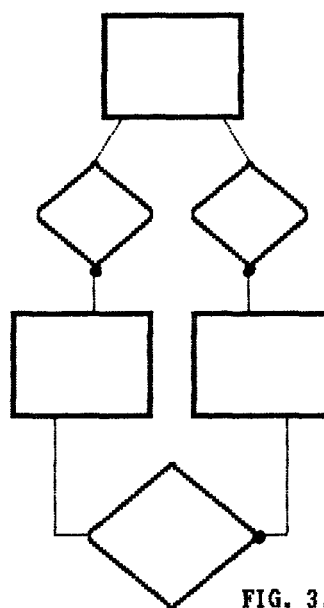


FIG. 3.5b

Figura 3.5 - Configuracoes Proibidas envolvendo Arvores e Relacionamentos Totais.

* Nenhum elemento pode pertencer a mais de uma árvore (figura 3.6a).

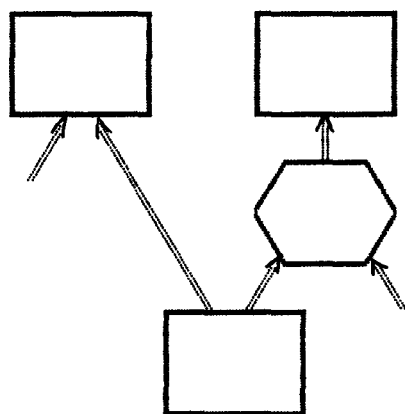


FIG. 3.6a

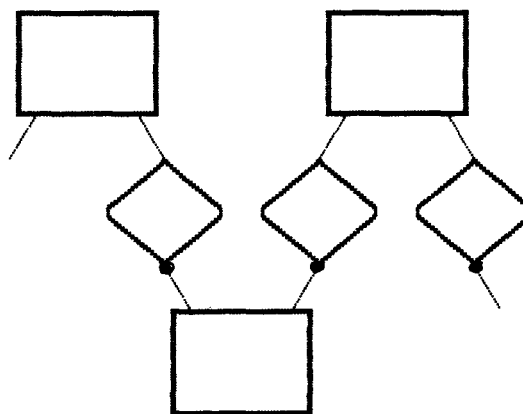


FIG. 3.6b

Figura 3.6 - Configurações Proibidas envolvendo Árvores.

A figura 3.7 exemplifica um dos casos proibidos, o da figura 3.2a. Tomando como base o exemplo da figura 3.7, caso seja permitido a uma entidade (HOSPEDE) estar associada a mais de um relacionamento total, ocorre propagação de inserções, ou seja, para se efetuar uma inserção em HOSPEDE é necessário verificar se é possível inserir ocorrências correspondentes nos relacionamentos - HOSPEDA e E_USADO. Estas inserções podem exigir inserções em HOTEL e QUARTO. Este é um exemplo de caso em que [CASAB8] considera a opção de bloqueio em que uma atualização não é efetuada se houver necessidade de propagação. Em outras palavras, este tipo de configuração é proibido na tese por corresponder a situações de cascadeamento não controlado de atualizações.

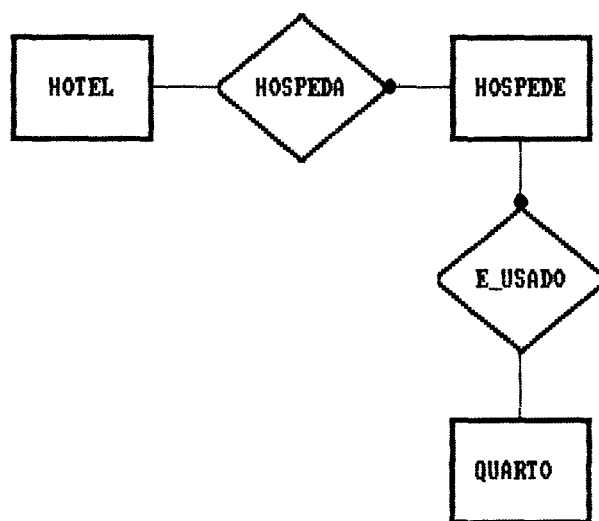


Figura 3.7 - Exemplo da configuração proibida 3.2a.

A figura 3.8 é um exemplo da proibição da configuração 3.2b. Uma inserção em HOSPEDE precisa efetuar inserções em HOSPEDA e talvez em HOTEL. Caso haja necessidade de inserção em HOTEL, é preciso continuar a verificação para o relacionamento CONSTROI e a entidade CONSTRUTORA.

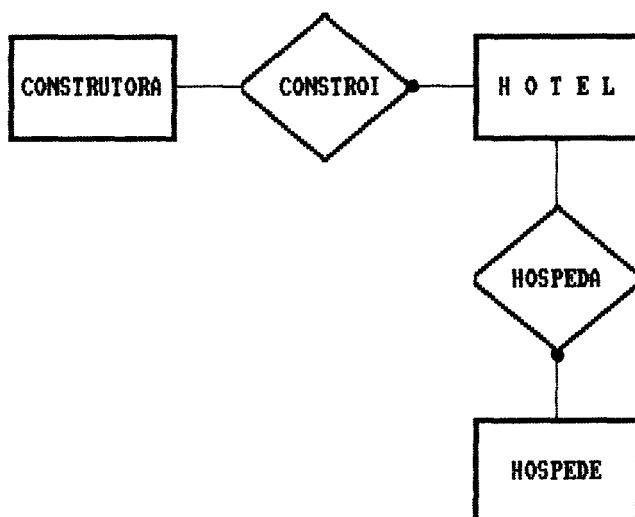


Figura 3.8 - Exemplo da configuração proibida 3.2b.

Também ocorre propagação de atualizações quando se modifica agregações que englobam relacionamentos totais e que participam de árvore. Na figura 3.9, quando se deseja realizar uma remoção em A, é necessário verificar se existe uma ocorrência correspondente em R (e B). Caso seja verdade, remove-se de A, R e B. Além disto, como se trata de hierarquia, é necessário propagar a remoção para a entidade genérica D a menos que exista em C uma ocorrência de mesma chave.

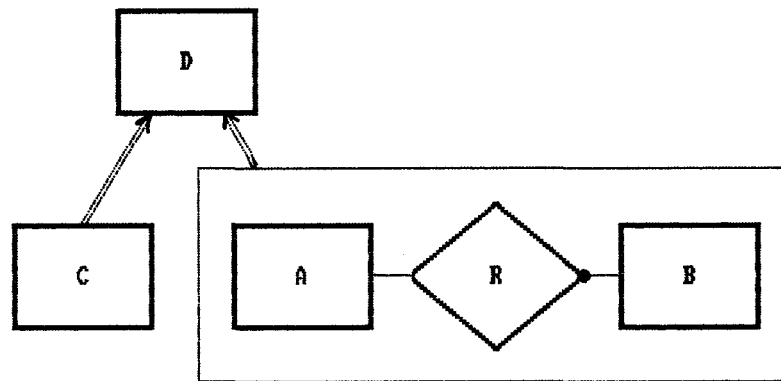


Figura 3.9 - Exemplo da configuração proibida 3.3.

Note que a figura 3.5a (redesenhada em 3.5b) é um caso que recai nas figuras 3.2a e 3.2b, já que a ligação de um filho ao pai em uma hierarquia representa um relacionamento total de pai para filho. Da mesma forma, a figura 3.6a (redesenhada em 3.6b) cai em problema semelhante ao da figura 3.2a.

Neste último caso, por exemplo, para remoção de ocorrência de um elemento-folha que participa de mais de uma árvore é necessário verificar: a) quando a árvore é uma hierarquia, se existe outro

elemento com mesma chave, o que implica que não deve ser propagada a remoção para a entidade raiz da hierarquia; b) quando a árvore é uma generalização, deve-se propagar a remoção para a entidade raiz da generalização. Em outras palavras, perde-se o controle da extensão da propagação.

Conforme já mencionado, o artigo de [CASAB88] contorna os problemas de algumas destas configurações através de especificações de propagação ou bloqueio de atualizações. Se especificado bloqueio (de propagação), cessa a preocupação de extensão da cadeia de atualizações. Esta tese, no entanto, não considera esta possibilidade e só analisa o que o modelo de [CASAB88] chama de condição de "propaga imediatamente". Por este motivo, foi necessário delimitar as configurações permitidas, visando assim previsão da extensão da propagação ainda durante a fase de projeto.

3.4 - ESPECIFICAÇÃO DA INTERFACE

3.4.1 - MÓDULO DE DIALOGO

O sistema precisa considerar dois tipos de operação: operações sobre o esquema e sobre os dados. Em ambos os casos, devem ser permitidas consultas e atualizações. Saliente-se que é possível modificar o esquema mesmo após criado o banco de dados. Na proposta atual do REVER, as modificações permitidas são ainda limitadas (por exemplo, não se considera possível mudar o tipo de um atributo uma vez criada uma relação).

A necessidade de definir uma sintaxe para operações de geração, consulta e atualização permitidas sobre o modelo E-R estendido aqui utilizado leva-nos à discussão da linguagem apropriada, ou seja, o módulo de DIALOGO.

Inúmeras linguagens de consulta E-R foram propostas sobre uma base de dados relacional (por exemplo, [MALH86]). Todas tentam aliar a riqueza semântica do modelo E-R ao formalismo relacional.

Atzeni [ATZE83], por exemplo, baseia-se no cálculo relacional para mostrar que linguagens de consulta para o modelo E-R podem ser 'completas', que é um requisito fundamental das linguagens relacionais. Fornece duas definições formais: linguagens E-R completas e completas simplificadas. As linguagens completas, segundo esses autores, possuem a habilidade para extrair dados de um número variável de diferentes conjuntos de entidades e/ou conjuntos de relacionamentos, enquanto às linguagens completas simplificadas basta possuir a habilidade de selecionar ocorrências de um único conjunto de entidades ou conjunto de relacionamentos, com condições que possam envolver qualquer objeto do esquema.

As linguagens completas simplificadas são consideradas muito importantes pois efetuam a recuperação de entidades e relacionamentos necessários a algumas operações de atualização (tais como inserção de relacionamentos, remoção ou modificação de entidades ou relacionamentos), enquanto os conjuntos de entidades e relacionamentos (que as linguagens completas devem necessariamente recuperar) são mais usados em modelos que se preocupam em capturar a sintaxe e não a semântica do mundo real.

Segundo [DATE77], uma linguagem relacional completa "significa para o usuário que, se uma informação desejada está num banco de dados, então ela pode ser recuperada através de uma consulta simples". As linguagens completas, entretanto, não possuem funções como as de agregação ou que permitam operações aritméticas implícitas, que são propriedades desejáveis para facilitar a interação com o usuário.

O componente de DIALOGO da interface REVER se comunica com o usuário através de menus, de forma interativa. As principais vantagens desse tipo de interação com o usuário são:

- * facilidade de manuseio, permitindo que o usuário aprenda rapidamente a usar o sistema;
- * diminuição da possibilidade de erros de digitação;
- * diminuição do trabalho de digitação a ser executado pelo usuário;
- * aumento do universo de usuários: dependendo da clareza e precisão do texto que compõe o menu, qualquer usuário consegue utilizar o sistema, mesmo aqueles menos especializados;
- * rapidez de implementação do módulo DIALOGO, pois são evitados os problemas de implementação decorrentes de outros tipos de linguagens (por exemplo, se o módulo interage com o usuário através de uma linguagem especial para modelo E-R, há necessidade de interpretar seus comandos; se através de linguagem gráfica envolve questões como controle de imagens ou tela);
- * rapidez de implementação para implantação de novos bancos de

dados, pois não existe a necessidade de modificar quaisquer comandos do módulo DIALOGO quando da mudança do banco de dados utilizado;

* facilidade de inclusão de novas rotinas, acionadas através da inclusão de novas opções no menu.

O capítulo 4 mostra exemplos de utilização da interface implementada.

Quaisquer operações (consultas ou atualizações) sobre dados são especificadas pelo usuário em termos do projeto E-R. Desta forma, o sistema precisa primeiro verificar que elemento(s) do esquema relacional corresponde(m) a um objeto E-R para depois efetuar a operação desejada sobre as relações correspondentes (ocorrências do esquema E-R) e mostrar o resultado ao usuário.

Vale observar que o módulo DIALOGO implementado restringiu-se à parte de manuseio de esquemas, onde as opções permitidas são mais limitadas e portanto o uso de menus é adequado.

No entanto, é preciso considerar que este módulo deve também permitir consultas e atualizações de ocorrências, caso em que a opção por uma interface do tipo linguagem pode ser considerada. A tese, no entanto, parte do pressuposto de que consultas a ocorrências (quer para verificação de seu conteúdo, quer como passo que antecede uma atualização) podem igualmente ser feitas através de menus, como descrito a seguir.

Detalhes do módulo no que diz respeito a consulta e atualização do esquema são descritos no capítulo 4. Para consulta/atualização de uma ocorrência, o usuário deve primeiro indicar o objeto que quer manipular, o que é feito em etapas, através de

menus, onde indica o tipo de objeto e seu nome. Uma vez localizado o objeto, o usuário deve indicar o valor do(s) identificador(es) das ocorrências desejadas, sendo instado a fornecer o tipo de operação e os demais atributos necessários à operação.

Uma opção a ser considerada no caso de consultas mais complexas é utilizar uma interface do tipo Query-by Example (vide descrição a respeito em [MAIE83]).

3.4.2 - O MÓDULO ESQUEMA

O módulo ESQUEMA é responsável pela geração, consulta e atualização do esquema relacional correspondente ao mapeamento do esquema E-R, usando as regras de 3.2. Este módulo é ativado pelo módulo DIALOGO. Como este módulo foi implementado, alguns dos seus detalhes foram deixados para o capítulo 4, e esta seção fornece apenas a sua filosofia de funcionamento.

Optou-se por processar o esquema internamente como um conjunto de estruturas de dados. Apenas quando o usuário se considera satisfeito com o projeto do esquema E-R é que são gerados os comandos para criação das tabelas correspondentes.

É opção do usuário salvar este estado do esquema em um arquivo, que também pode ser recuperado em qualquer momento. A geração dos comandos para o SGBD é feita através de acionamento do módulo TRADUTOR.

A utilização de uma estrutura intermediária visa também otimização de transações, pois facilita a execução de procedimentos para consulta ao banco de dados. A idéia é que só

sejam efetuados acessos ao banco de dados após exame da estrutura do esquema E-R para determinação das relações envolvidas. Por exemplo, o exame prévio do esquema armazenado desta forma pode indicar que uma operação não terá sucesso, evitando acessos desnecessários ao banco de dados. A estrutura intermediária faz as vezes de diretório para o esquema E-R, permitindo a ligação diretório E-R → diretório relacional.

A geração de um esquema é, desta forma, uma atividade interativa que pode se prolongar por várias sessões.

As operações de consulta são aquelas que permitem ao usuário navegar pelo esquema E-R, o que o módulo ESQUEMA realiza navegando dentro das estruturas de dados. No caso de haver vários níveis de abstração (por exemplo, agregações), deve ser permitido ao usuário percorrer estes vários níveis.

O usuário deve poder obter informações de dois tipos: estruturais e de integridade. Informações estruturais são aquelas que indicam interrelacionamentos entre objetos, seus atributos e domínios; informações de integridade permitem ao usuário determinar a que restrição de integridade um objeto está submetido.

As operações de atualização sobre o esquema E-R são, na verdade, de dois tipos: operações permitidas sobre um esquema vazio e operações permitidas quando o banco de dados já contém dados. Muitas atualizações triviais sobre o esquema de um banco de dados vazio deixam de sê-lo a partir do momento em que dados são carregados. Parte-se do pressuposto que a definição de qualquer esquema E-R pode sofrer alteração (por exemplo, quantidade de

atributos de uma entidade, restrições de integridade ou interrelacionamentos entre objetos). No entanto, modificações de esquema se tornam mais difíceis após a geração de dados.

Enquanto não houver dados, modificações em esquemas são mapeadas para modificações dos esquemas relacionais correspondentes, não havendo problemas a menos que estas mudanças introduzam configurações proibidas ilustradas nas figuras da seção 3.3. Por outro lado, modificações de um projeto E-R que corresponde a um banco de dados carregado podem resultar em modificações radicais no conteúdo das relações. Se, por exemplo, um relacionamento é modificado de parcial para total, é preciso percorrer todas as ocorrências das entidades a que está ligado, para garantir a manutenção da nova dependência de inclusão.

Os tipos de atualizações desejadas sobre esquema vazio são de duas espécies: a. modificações sintáticas; b. modificações estruturais/semânticas. O primeiro caso, já discutido aqui, corresponde a modificação de nomes e domínios de objetos e atributos, ou cardinalidades, e onde só se prevê problemas caso envolva dependências de inclusão. No exemplo da figura 3.8, a eliminação de atributos da entidade HOSPEDE talvez deva ser propagada a HOTEL e seguir em cascata até CONSTRUTORA. Modificações estruturais são aquelas em que há criação ou remoção de objetos.

Um exemplo do tipo de situação que pode ocorrer é a fragmentação de entidades, que pode ter ao menos duas soluções: ou é necessário optar pela criação de uma chave para cada uma das novas entidades, o que implica internamente em normalização e que

pode levar à geração/remoção de relações pela inclusão/remoção de dependências funcionais; ou pode se decidir repetir a mesma chave para as duas entidades, executando uma divisão lógica. A tese não tem como objetivo um sistema "inteligente" que oriente o usuário quanto à forma de reorganização. Assim, para este exemplo, caberia ao usuário eliminar a entidade original e criar duas novas entidades, determinando seus atributos.

As estruturas de dados utilizadas, bem como a interação entre os módulos DIALOGO, ESQUEMA e TRADUTOR, são discutidas no capítulo 4.

3.4.3 - O MÓDULO MANUSEADOR

O módulo MANUSEADOR é responsável por mapear o manuseio de ocorrências E-R para o manuseio das relações correspondentes. Esta seção discute como deve ser feito este mapeamento. Modificações de esquema para bancos de dados já carregados envolvem também intervenção deste módulo.

O módulo MANUSEADOR é ativado pelo módulo DIALOGO, que com ele interage e lhe passa informações sobre o objeto E-R que se deseja manusear, o tipo de operação e a identificação das ocorrências.

O MANUSEADOR identifica a(s) relação(s) correspondente(s) através de consulta às estruturas de dados geradas e mantidas pelo módulo ESQUEMA. Uma vez feita a identificação, deve processar as solicitações do usuário, interpretando-as e passando-as ao módulo TRADUTOR (que as traduz para linguagem do SGBD hospedeiro).

O MANUSEADOR é também responsável pela interpretação das mensagens recebidas do SGBD e sua comunicação ao usuário, indicando o sucesso ou insucesso das operações solicitadas. Além disto, deve verificar, auxiliado pelas informações das estruturas do ESQUEMA, se uma determinada operação exige acionamento de gatilhos e agir adequadamente.

Seja, por exemplo, o esquema da figura 3.10 a seguir.



Figura 3.10 - Exemplo envolvendo Dependência de Inclusão.

Se o usuário solicitar inserção na entidade HOSPEDE, o MANUSEADOR, após consulta às estruturas de dados do ESQUEMA, deve realizar as seguintes operações:

- * pedir dados de HOSPEDE (valores dos atributos da tupla), a partir dos nomes de atributos armazenados na estrutura;
- * informar ao usuário que HOSPEDE precisa ser inserido em algum hotel (a entidade é fraca em relação ao relacionamento hotel) e pedir identificação do hotel. Em outras palavras, verifica que há um gatilho para manutenção da dependência que deve ser acionado.

* solicitar ao TRADUTOR:

- verificação se já existe o hotel indicado;
- transação para inserção nas relações correspondentes (insucesso na inserção de HOSPEDE indica que a transação não pode ser efetuada).

Vale notar que as operações descritas supõem um ambiente monousuário, caso contrário seria necessário ao MANUSEADOR providenciar as trancas adequadas, bem como garantir a geração de uma transação completa de consultas/atualizações.

O trabalho de [CASAB8] discute a implementação de monitores para garantir integridade referencial e que geram transações corretas. Os problemas relativos a atualização de ocorrências e execução de gatilhos seriam por si só objeto de uma tese, e não serão desta forma discutidos mais a fundo. Sugere-se, no caso de manutenção de dependências de inclusão, solução do tipo descrita por [CASAB8], já implementada e relatada na introdução. O resto desta seção propõe a solução de alguns destes problemas visando o modelo de mapeamento e restrições desta tese.

No caso mais geral, o mapeamento de uma consulta sobre um objeto E-R é simples porque existe uma correspondência de 1:1 entre os objetos do esquema E-R e as relações, e qualquer operação de consulta a um objeto pode ser traduzida em combinações de seleção (σ) e/ou projeção (π) sobre a relação correspondente.

Existem vários problemas no que diz respeito a operações sobre ocorrências. Uma consulta a um objeto complexo E-R (por exemplo, uma generalização) pode corresponder a consultas a várias

relações. Sugere-se, neste caso, informar ao usuário sobre o esquema consultado para que ele possa decidir quais os dados que deseja realmente ver. Assim, por exemplo, se for consultada uma generalização, o usuário recebe como resposta os nomes dos nós componentes, podendo então escolher qual nó examinar.

Da mesma forma, atualizações sobre um objeto podem ser propagadas a outros (razão pela qual se optou pelas restrições mostradas na seção 3.3). Apesar destas simplificações, o sistema é obrigado a propagar atualizações em árvores e em casos de relacionamentos totais.

A simplicidade obtida na implementação de consultas é decorrente do mapeamento proposto que é de 1:1. No entanto, se forem considerados mapeamentos mais complexos (por exemplo, [BRAG88]), um objeto E-R pode corresponder a um fragmento de relação, ou a uma visão. No caso de atualização, pode-se cair em situação de atualização de visões, em que a atualização de uma ocorrência requeira atualização de várias relações. Por outro lado, a atualização de um conjunto de objetos pode ser mapeada para atualização de uma única relação.

Os problemas encontrados na atualização de ocorrências E-R são inúmeros e têm merecido destaque na bibliografia. Os parágrafos que se seguem discutem alguns destes problemas encontrados na propagação de atualizações, dadas as restrições feitas sobre o tipo de esquema conceitual válido para a tese.

Serão consideradas apenas atualizações (de dados) do tipo inserção, remoção e modificação de valores não-chave. Qualquer pedido de atualização é suposto como sendo requisitado em cima de

um dos seguintes elementos: a) entidade simples; b) relacionamento; c) agregação; d) nó de uma árvore (que pode ser dos tipos a) ou c)). De agora em diante, qualquer atualização em um elemento é entendida como mapeada para a relação correspondente.

O caminho de propagação de atualizações, descrito a seguir, será obtido através de navegação dentro da estrutura de dados do esquema, acompanhado, quando necessário, de consultas ao usuário sobre opções a serem tomadas.

Para melhor entendimento do que se segue, serão definidos os seguintes termos (figura 3.11):

a) Caminho em uma árvore

E o mesmo conceito utilizado em estruturas de dados do tipo árvore.

b) Caminho ancestral de um nó

E o caminho que sai da raiz de uma árvore até atingir o nó em questão.

c) Irmãos de um nó

Dado o ancestral (pai) do nó em uma árvore, são os outros filhos daquele pai, no mesmo nível.

d) Sub-árvore de um nó

E a sub-árvore da qual o nó é a raiz.

Para inserção de ocorrências em nós de hierarquias ou generalizações, é necessário ao usuário indicar valores para todos os atributos no caminho na árvore.

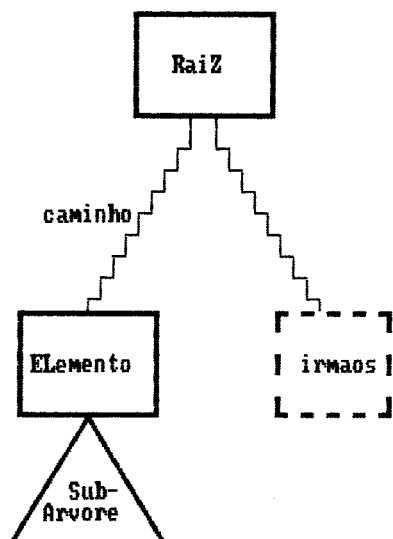


Figura 3.11 - Diagrama esquemático para navegação dentro de uma estrutura.

O exemplo da figura 3.12 consiste na mesclagem dos conceitos de generalização e hierarquia, onde DESENHISTA, CONTADOR e SECRETARIA são generalizações de EMPREGADO, que pode ser MENSALISTA e/ou HORISTA. Podem existir ocorrências de DESENHISTA com ocorrências em MENSALISTA e HORISTA, por exemplo.

Para inserir uma ocorrência em CONTADOR é necessário verificar em EMPREGADO se já existe tal chave. Se existir, não é possível realizar a inserção. Caso não exista, é inserida a chave e todos os atributos de EMPREGADO, e é repetida em CONTADOR a mesma chave de EMPREGADO acrescida de todos os seus atributos específicos. Além disto, é preciso inserir a mesma chave em MENSALISTA e/ou em HORISTA, dependendo da semântica da aplicação. Para inserir em SECRETARIA, além dos passos já descritos, referentes ao caminho ancestral, deve-se também inserir uma ocorrência de mesma chave na sub-árvore, em SIMPLES ou BILINGUE, além de todos os atributos complementares (por exemplo, LING_NATAL, SEG_LINGUA, etc).

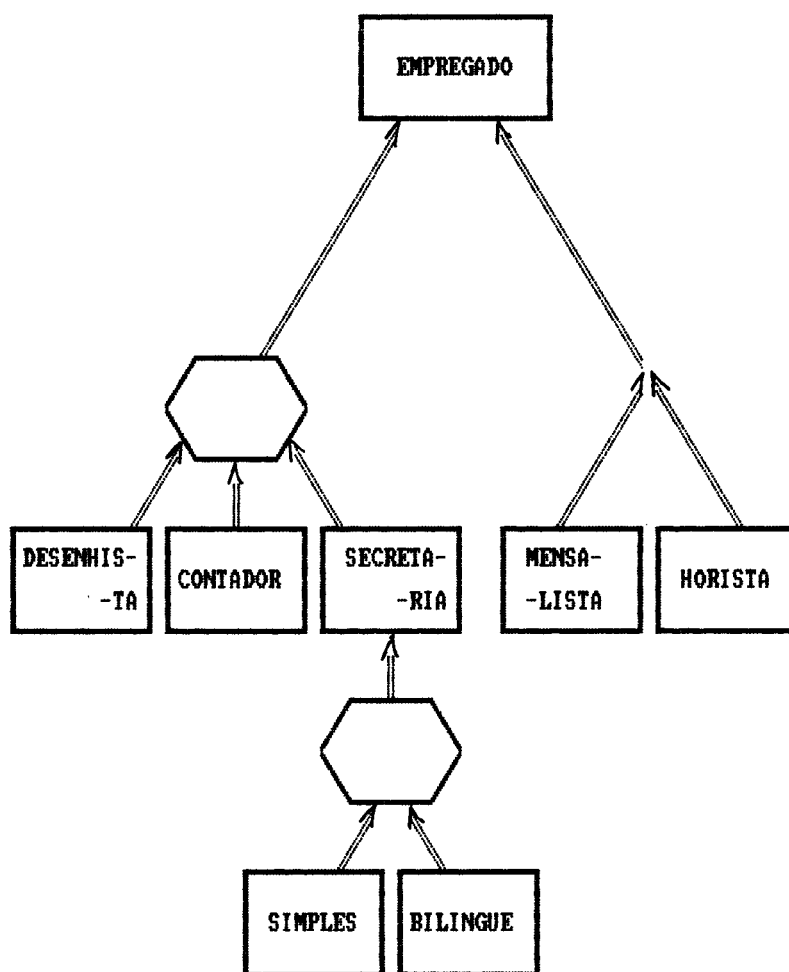


Figura 3.12 - Exemplo de Árvore Heterogênea, com Generalização e Hierarquia.

Para remoção em nó de generalizações, devem ser eliminados todos os elementos correspondentes ao caminho na árvore; para remoção em hierarquias, serão eliminados todos os elementos correspondentes ao caminho ancestral na árvore, se o elemento a remover não tiver correspondente em seus irmãos na árvore; e serão eliminados todos os elementos da sub-árvore da qual ele é raiz.

A remoção de uma ocorrência de SECRETARIA, na figura 3.12, implica na remoção da ocorrência de mesma chave em EMPREGADO, MENSALISTA e/ou HORISTA caso existam, e ainda na remoção da ocorrência correspondente em SIMPLES ou BILINGUE.

Modificações de valores de atributos não acarretam problemas, a menos que se trate de chave estrangeira, o que é manipulado por regras de dependências de inclusão.

Tendo em vista estas definições, e considerando-se as atualizações permitidas, observa-se o seguinte:

- * modificações não exigem propagação, restringindo-se ao elemento considerado (pois não se permite mudar conteúdo de chaves). No caso de modificação de valor de chave estrangeira, a modificação só é permitida se o novo valor corresponder a um (outro) valor de chave já existente.
- * as inserções e remoções discutidas a seguir são as que requerem propagação. As demais atualizações se restringem ao elemento atualizado.

Casos de propagação de atualizações:

a) Quando há dependência de inclusão (vide figura 3.13):

- * se inserir em B e/ou R --> inserir se necessário em A;
- * se excluir de A e/ou R --> excluir se necessário de B;

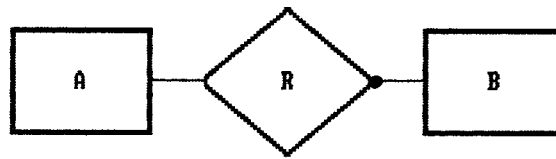


Figura 3.13 - Exemplo de Propagacao de Atualizacoes quando ha Dependencia de Inclusao.

b) Em entidades (figura 3.14):

- * se excluir de A/B --> excluir se necessário de R;
- * se incluir em R --> incluir se necessário em A e B;

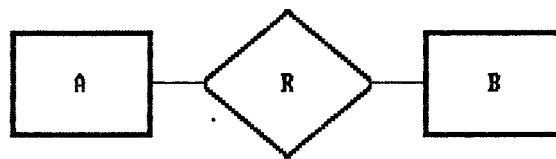


Figura 3.14 - Exemplo de Propagacao de Atualizacoes em Entidades.

c) Em agregações:

- * obedece as regras de (a) e (b);

d) Em hierarquias de tipo (figura 3.11):

d.1) se inserir em ELEMENTO

* Propagação no caminho ancestral:

- se já existem irmãos com mesma chave do pai, não fazer nada;
- senão, propagar pelo caminho ancestral até RAIZ.

* Propagar por um caminho em SA.

d.2) se remover de ELEMENTO

* Propagação no caminho ancestral:

- se existe irmão com mesma chave, não fazer nada;
- senão, propagar pelo caminho ancestral, respeitando a existência de irmãos no caminho.

* Propagar por todos os caminhos necessários em SUB-ARVORE.

e) Em generalizações (figura 3.11):

* se inserir em ELEMENTO, propagar para o caminho ancestral e para algum caminho em SUB-ARVORE;

* se excluir em ELEMENTO, propagar para o caminho ancestral e para o caminho (de mesma chave) em SUB-ARVORE.

3.4.4 - O MÓDULO TRADUTOR

O módulo TRADUTOR é responsável pela tradução das solicitações enviadas pelos módulos ESQUEMA e MANUSEADOR para as linguagens de definição e manipulação de dados do SGBD hospedeiro. Traduz também as mensagens do SGBD para os módulos em questão.

Para a concretização de uma operação requerida pelo usuário a nível de ESQUEMA, o módulo TRADUTOR gera uma transação para executar os comandos disparados por essa operação, que irão modificar o estado do banco de dados.

Para realizar tais tarefas o TRADUTOR deve ser capaz de entender as estruturas do ESQUEMA para, a partir delas, gerar solicitações de criação de relações.

A implementação do TRADUTOR recebe como entrada ou a) estruturas de dados de ESQUEMA, ou b) dados do MANUSEADOR; e devolve: a) mensagens do SGBD; b) resultados de consultas e atualizações, possivelmente em "buffers" que possam ser manipulados pelo MANUSEADOR, conforme descrito em [SOUZ88].

A geração de relações não acarreta problemas já que o mapeamento é feito pelo ESQUEMA. O manuseio de consultas também não gera dúvidas pois o MANUSEADOR traduz a consulta em termos de acesso a relações antes de acessar o TRADUTOR. Para o manuseio de atualizações, o módulo MANUSEADOR indica a atualização e gatilhos que devem ser acionados. No entanto, é preciso que o TRADUTOR para o banco de dados permita a execução desses gatilhos e devolva ao módulo MANUSEADOR informações adicionais que devem ser interpretadas e passadas ao usuário, tais como:

- * violação de dependências funcionais;
- * violação de dependências de inclusão;
- * propagação das atualizações e seus reflexos em outros objetos do modelo, devido a dependências de inclusão.

Duas estratégias podem ser adotadas caso ocorra alguma das violações previstas:

- a) abandonar a execução pelo encerramento da transação, sem que seja efetuada a operação, devolvendo uma mensagem de insucesso ao usuário; ou

b) indicar ao usuário qual violação ocorreu, oferecendo opção para modificação da operação submetida.

O primeiro caso implica somente em retardar o disparo dos gatilhos envolvidos até que seja verificada a existência de qualquer das violações descritas, e se afirmativo abortar toda a transação. O segundo caso envolve a modificação da "transação", além de exigir do MANUSEADOR a especificação das novas atualizações que as modificações do usuário irão acarretar.

3.5 - RESUMO

Este capítulo apresentou a especificação da interface ER - relacional, com detalhamento dos módulos componentes (DIALOGO, ESQUEMA, TRADUTOR e MANUSEADOR). Foram discutidas regras de mapeamento e problemas existentes na atualização de esquemas e ocorrências.

O próximo capítulo apresenta o protótipo implementado.

4 - DESCRIÇÃO DA IMPLEMENTAÇÃO

O capítulo anterior descreve as características de uma interface que permita ao usuário tanto definir objetos E-R quanto manipulá-los, ficando a cargo da interface traduzir as solicitações do usuário a consultas e atualizações a relações. A especificação salienta os pontos que devem ser considerados na implementação.

Tendo em vista a complexidade de um sistema deste porte, o protótipo implementado como parte da tese se ateve à primeira fase do projeto E-R. Em outras palavras, foram implementados os módulos ESQUEMA e DIALOGO descritos no capítulo anterior, para geração de relações.

4.1 - DESCRIÇÃO GERAL

O protótipo implementado permite ao usuário especificar, gerar e consultar esquemas de objetos E-R.

Estes esquemas são armazenados nas estruturas descritas a seguir. Ao fim de uma sessão, as estruturas geradas são armazenadas em um arquivo, que pode servir de entrada para o módulo TRADUTOR e também ser utilizado em sessões futuras.

O protótipo foi desenvolvido em Turbo Pascal 4.0, para computadores do tipo PC compatível, ocupando 136 Kb de código executável para 3.500 linhas de código fonte.

E permitido ao usuário navegar dentro do esquema, tanto para consultas quanto atualizações. A navegação permite ao usuário, por exemplo, "entrar" em agregações, ou percorrer caminhos específicos dentro de árvores. O acesso às várias estruturas internas é feito via "funções de hash", cuja eficiência é monitorada por rotinas auxiliares.

Os principais módulos dentro do módulo DIALOGO têm as seguintes funções:

- * formatação das telas que apresentam os menus de opções para o usuário, através da utilização de rotinas gráficas pertencentes à linguagem Turbo Pascal;
- * rotinas que efetuam a leitura dos dados inseridos pelo usuário, armazenando-os em variáveis de trabalho.

As opções disponíveis neste módulo são:

- * criação de esquemas E-R;
- * consulta a esquemas E-R;
- * recuperação de esquemas E-R previamente armazenados em arquivos;
- * gravação de esquemas E-R em arquivo;
- * impressão do esquema E-R sendo manipulado (parcialmente implementada);
- * atualização de esquemas E-R (não implementada).

Os principais módulos dentro do módulo ESQUEMA têm as seguintes funções:

- * geração de objetos E-R;

- * consulta a objetos E-R;
- * busca/recuperação de um objeto E-R nas estruturas internas;
- * execução da(s) função(ões) *hash* utilizadas para acesso às estruturas internas;
- * monitoramento da eficiência da distribuição obtida pela aplicação da(s) função(ões) *hash*.

Para implementação do esquema é utilizada a estrutura de dados mostrada na figura 4.1, chamada de "estrutura intermediária", que permanece residente em memória durante todo o período de geração e manipulação de um determinado esquema, sendo sobre ela realizadas as operações de geração e consulta de objetos E-R.

Esta estrutura armazena quatro diferentes tipos de registros (estrutura OBJETO) implementados em listas ligadas. O acesso a essas listas é conseguido através de ponteiros que estão armazenados em um vetor (estrutura CONTROL-OBJ). Para acessar determinado ponteiro é calculada a posição do vetor onde tal ponteiro se encontra armazenado. O parâmetro da função *hash* é o nome do objeto E-R procurado. São duas as subestruturas que compõem a estrutura intermediária:

1. Estrutura de controle do objeto manipulado (CONTROL-OBJ), que controla o acesso aos objetos E-R. É dividida em duas outras subestruturas: a primeira consiste em um conjunto de informações para controle da eficiência dos acessos efetuados e dados que apontam para o último objeto E-R acessado (CONTROL-ACESS); a segunda é um vetor que armazena os ponteiros que apontam para as listas de objetos E-R.

CONTROL-ACCESS

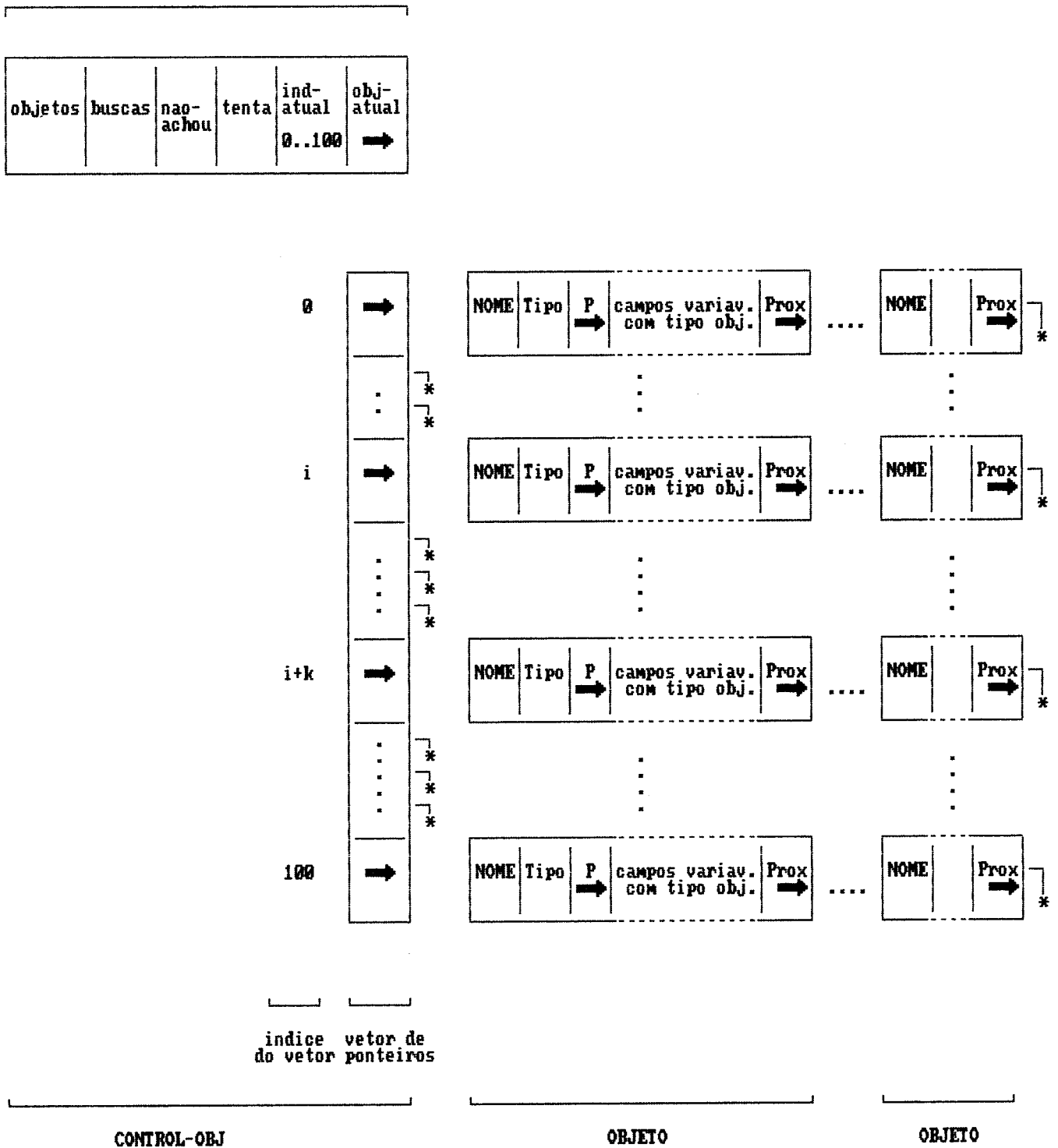


Figura 4.1 - ESTRUTURA INTERMEDIARIA

2. Estrutura que representa um objeto E-R (OBJETO). Possui um conjunto de campos fixos, que inclui o nome e tipo do objeto em questão e um ponteiro para o próximo objeto da lista; e um apontador para um conjunto de campos variáveis de acordo com o tipo do objeto. Estruturas OBJETO estão ligadas em cadeia originada no vetor de ponteiros.

As idéias contidas nessas estruturas de acesso seguem sugestões contidas em [FLOY87] e serão descritas com mais detalhes na sessão 4.3. Parte dessas idéias já foram apresentadas em [NOGU88], no entanto inúmeras modificações foram introduzidas e são explicadas nesta tese.

A estrutura intermediária pode ser pesquisada de duas formas:

1. *Sequencial* - acessando cada um dos ponteiros do vetor de ponteiros seguindo a ordem de seus índices, e a partir desse ponteiro percorrer toda a lista de objetos para a qual ele aponta. Em outras palavras, é acessado o ponteiro do vetor com índice igual a 0 (zero) e a partir dele o objeto E-R que ele aponta, seguindo depois todos os outros objetos a este ligados. Uma vez percorrida toda a lista é acessado o ponteiro de índice 1 e percorrida toda a lista correspondente de objetos, e assim por diante até que seja percorrida toda a lista $n-1$ de objetos, onde n é o número máximo de elementos que o vetor pode armazenar.
2. *Indexada* - dado o nome de um objeto E-R sendo procurado, é calculado o número *hash*, correspondente à aplicação de uma determinada função *hash* sobre esse nome. O resultado é o índice do vetor de ponteiros para acessar a lista de

objetos que contém o objeto E-R procurado. Esta lista é percorrida de forma sequencial para encontrar o objeto em questão.

A primeira forma de pesquisa corresponde a uma consulta em que o usuário queira ver todo o esquema; a segunda se refere a consulta de objeto específico.

Cada conjunto de registros (OBJETOS) é identificado por um campo comum chamado *TIPO*, onde: *E* = entidade simples, *R* = relacionamento, *AG* = agregação, *G* = generalização e *H* = hierarquia de tipos.

Todos os OBJETOS contém um ponteiro (*Prox*) para o próximo OBJETO da lista, o identificador de *TIPO*, o *NOME* do elemento que representam e um ponteiro (*P*) para o conjunto variável de campos específicos de cada tipo de OBJETO.

Um objeto do tipo *entidade simples*, mostrado na figura 4.2, contém as seguintes informações específicas: o atributo *Chave*, o número total de seus atributos *Natr*, o número de relacionamentos aos quais está ligado - *Nrel*, o número de objetos complexos (árvores e agregações) dos quais esta entidade participa - *Narv*, um apontador *Cad(atr)* para uma cadeia de enumeração de seus atributos, um apontador *Cad(rel)* para uma cadeia que armazenará os *NOMES* dos relacionamentos aos quais esta entidade está ligada, um apontador *Cad(arv)* para uma cadeia que conterà os *NOMES* dos objetos complexos (árvores e agregações) dos quais esta entidade participa.

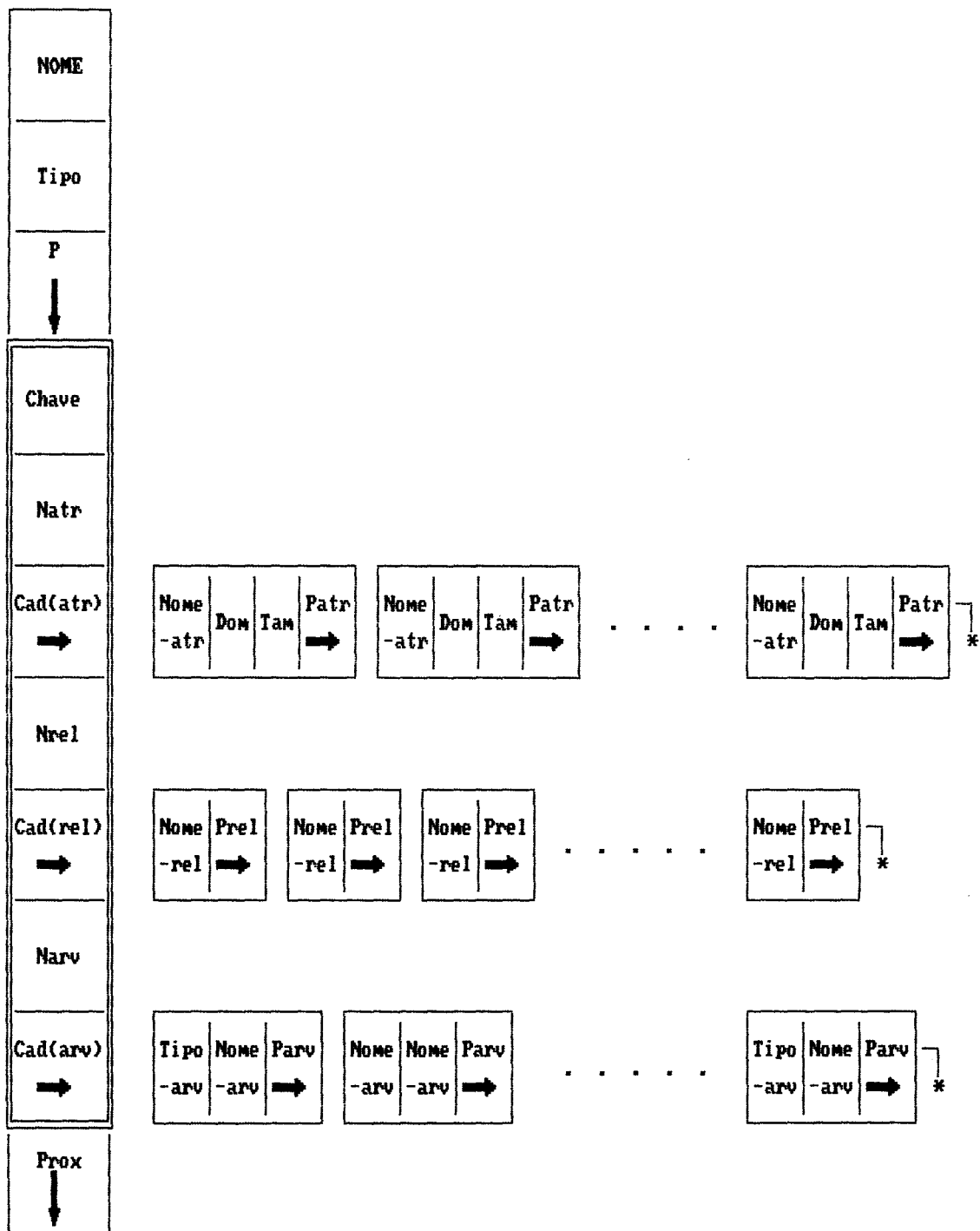


Figura 4.2 - Estrutura de dados para objetos do tipo ENTIDADE.

Os elementos da cadeia de atributos contém o nome do atributo (*Nome-atr*), o domínio (*Dom*) sobre o qual o atributo está definido, o tamanho (*Tam*) máximo permitido dentro desse domínio e um ponteiro (*Patr*) para o próximo atributo da cadeia. Os domínios considerados são: cadeia de caracteres, inteiros e reais.

Os elementos da cadeia de relacionamentos contém o nome do relacionamento (*Nome-rel*), e um ponteiro para o próximo elemento da cadeia (*Pre1*).

Os elementos da cadeia de objetos complexos contém o tipo (*Tipo-obj*) do objeto (agregação, generalização ou hierarquia), o *Nome-obj* do objeto e um ponteiro para o próximo da cadeia, *Parv*.

Um objeto do tipo *relacionamento*, mostrado na figura 4.3, engloba os seguintes campos: o atributo *Chave*, a indicação se é um relacionamento *Total/Parcial*, a *Direção* para relacionamentos totais, os ponteiros (*PE1* e *PE2*) para as entidades que liga, as cardinalidades (*Card1* e *Card2*) com que essas entidades participam do relacionamento, o número de atributos que possui - *Natr*, o apontador para uma cadeia contendo seus atributos - *Cad(atr)*, o número de objetos complexos (agregações e árvores) de que participa *Narv*, o ponteiro para uma cadeia indicando os nomes dos objetos complexos de que participa - *Cad(arv)*.

Para o objeto do tipo *agregação*, figura 4.4, as informações são: o ponteiro para o relacionamento que identifica a agregação - *Pre1*, o número de atributos da agregação *Natr*, um apontador *Cad(atr)* para a cadeia de enumeração de seus atributos, o número

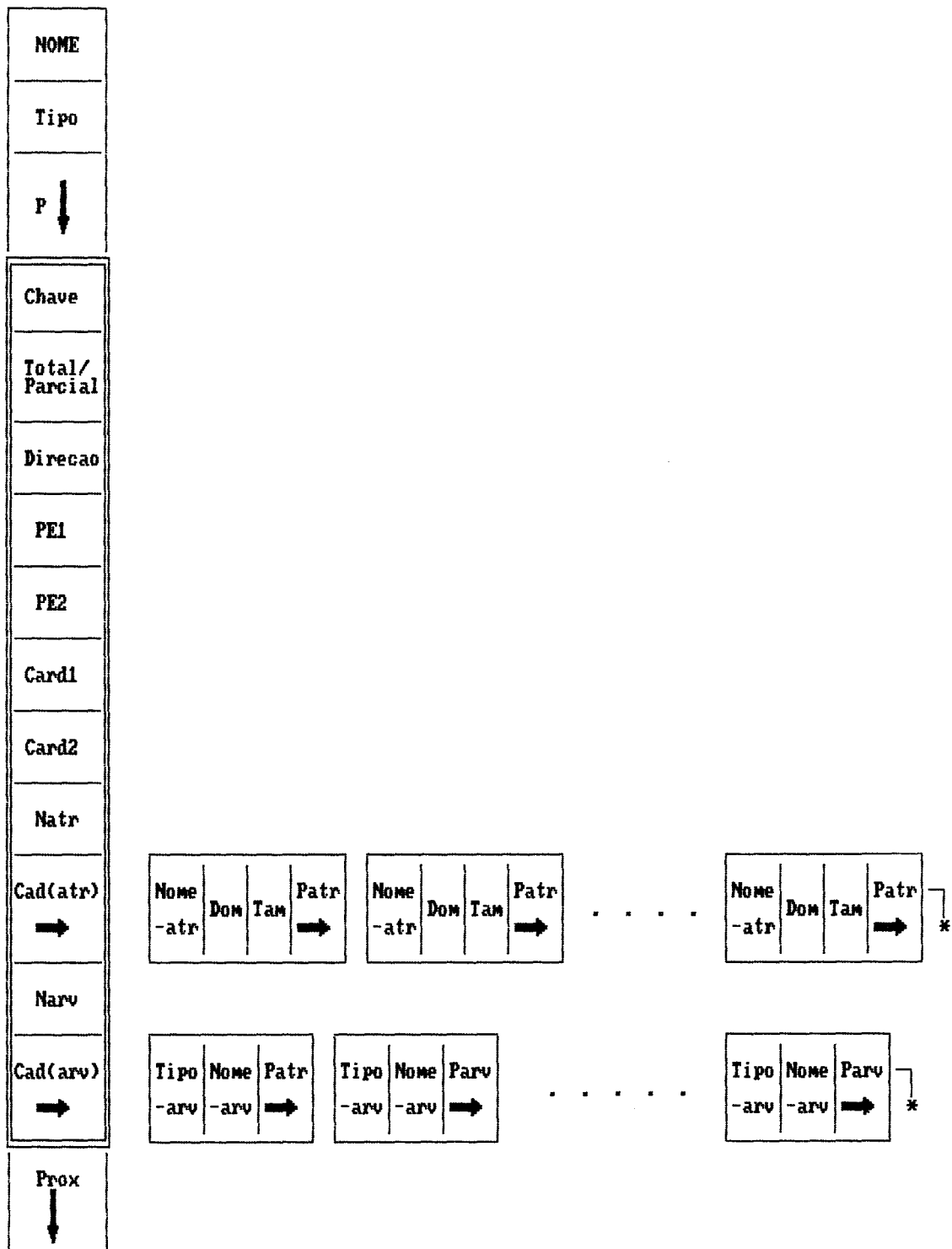


Figura 4.3 - Estrutura de dados para objetos do tipo RELACIONAMENTO.

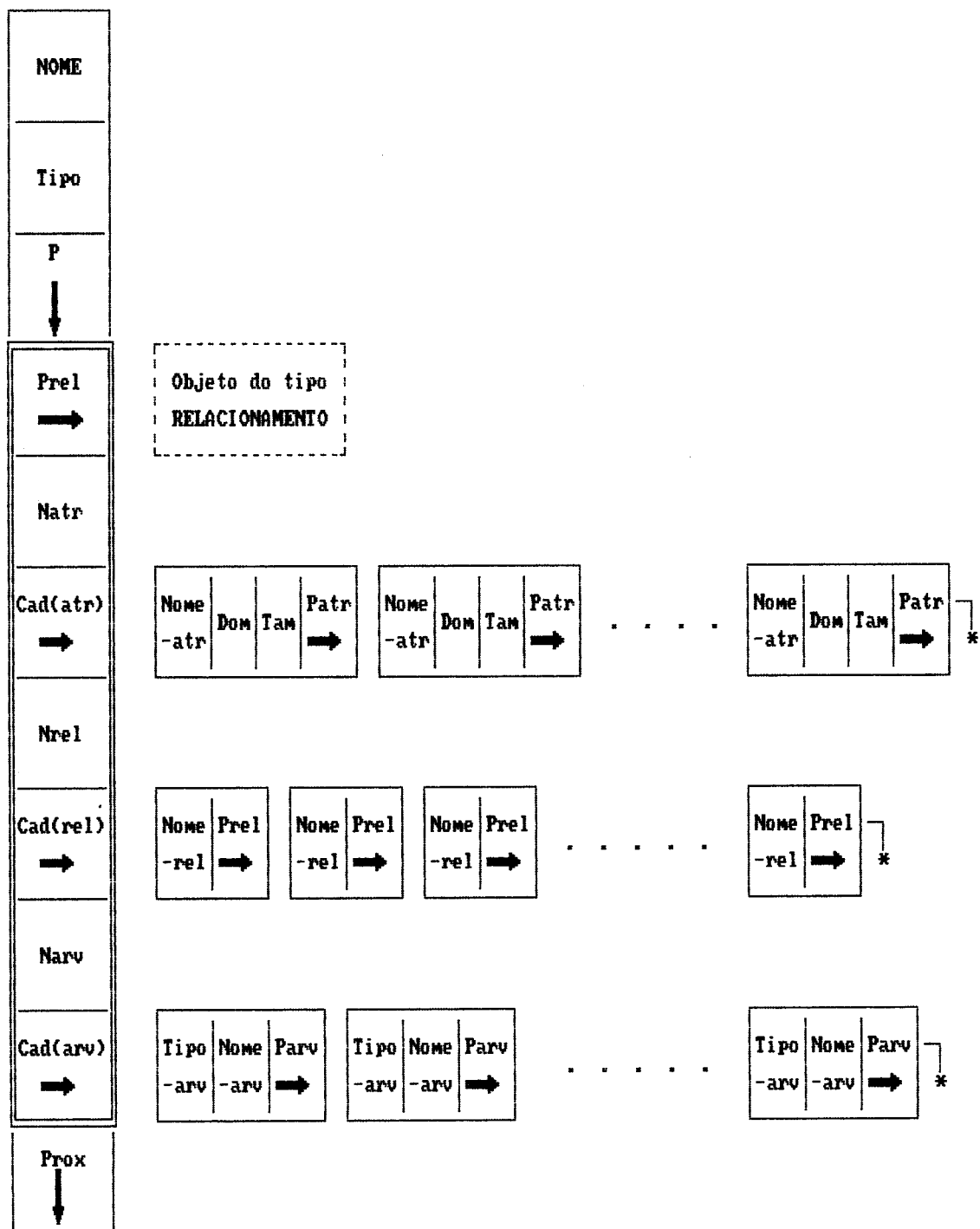


Figura 4.4 - Estrutura de dados para objetos do tipo AGREGAÇÃO.

Nrel de relacionamentos aos quais está ligada, um apontador para uma cadeia que contém os nomes dos relacionamentos aos quais a agregação está ligada - *Cad(rel)*, o número *Narv* de objetos complexos de que participa, um ponteiro para uma cadeia que informa os nomes dos objetos complexos dos quais a agregação participa - *Cad(arv)*.

As estruturas de árvores (generalizações ou hierarquias), figura 4.5, têm registros semelhantes para armazenar, nó a nó, os seguintes campos: um ponteiro *Pobj* para o objeto raiz da árvore (generalização ou hierarquia), um apontador *Ppai* para o nó-pai do objeto em questão (no caso da raiz há uma repetição de ponteiros), o número de nós-filho deste nó - *Nfilh*, um apontador *Cad(fil)* para uma cadeia que engloba os nomes dos nós-filho desse nó.

Os elementos da cadeia de nós-filhos contêm os nomes dos nós-filho (*Nome-filh*) e um ponteiro para o próximo elemento da cadeia - *Pfil*.

Com as informações constantes desses registros é possível navegar em todos os sentidos para a recuperação/alteração de dados.

Optou-se por dar às relações geradas os mesmos nomes que os elementos E-R (campo *NOME*).

O esquema E-R, gerado e manipulado através desta estrutura intermediária, pode ser armazenado em um arquivo. Para tal é percorrida a estrutura intermediária de forma sequencial, conforme

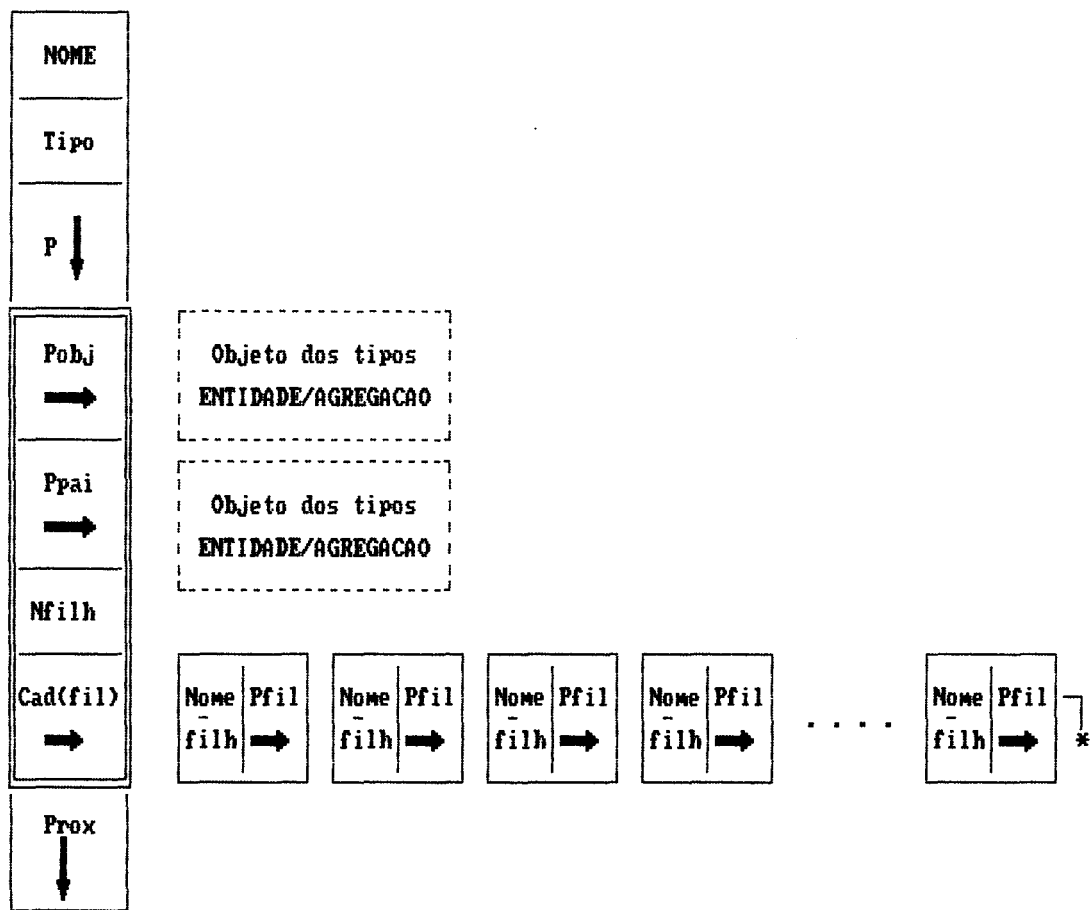


Figura 4.5 - Estrutura de dados para objetos do tipo ARVORE (GENERALIZACAO/HIERARQUIA).

explicado acima, sendo os dados referentes a cada objeto acessado transferidos para uma outra estrutura que corresponde ao registro físico a ser gravado em arquivo. Os passos descritos a seguir dão a sequência lógica para se realizar a gravação em arquivo dos objetos armazenados na estrutura intermediária:

- a. É gerado o índice 0 (zero) para possibilitar o acesso ao ponteiro armazenado no vetor de ponteiros, e a partir desse ponteiro é acessado o objeto para o qual ele aponta, caso exista;
- b. É efetuada a transferência dos dados do objeto acessado para a "estrutura de gravação" adequada (existem vários tipos diferentes de registros, conforme explicaremos posteriormente), sendo em seguida efetuada a gravação desse registro em arquivo;
- c. É percorrida a lista de objetos ligada a esse ponteiro até o seu final, sendo executada a transferência dos dados e gravação para cada um dos objetos da lista;
- d. É gerado novo índice do vetor de ponteiros (imediatamente consecutivo ao anterior), e executados os passos anteriores até que tenham sido percorridas todas as listas de objetos ligadas a todos os ponteiros do vetor.

Como as informações variam para diferentes tipos de objetos, optou-se por armazená-las em registros diferentes para em seguida realizar sua gravação. A única diferença existente entre as estruturas de dados dos objetos da estrutura intermediária e as estruturas de dados dos registros do arquivo é a inexistência dos ponteiros na segunda. Para mostrar a organização dos registros

dentro do arquivo físico segue-se uma descrição dos diferentes tipos de registros e sua distribuição no arquivo.

Todos os registros contêm um identificador de tipo (*TIPO*) e o *NOME* do elemento que representam. Os *TIPOS* de registros existentes são: Er = Entidade, Rr = Relacionamento, AGr = Agregação, Gr = Generalização, Hr = Hierarquia, Br = Atributo, Vr = Árvore, Lr = Lista de relacionamentos e Fr = Lista de nós-filho.

O registro referente a *entidade simples*, figura 4.6, contém o atributo *Chave*, o número de atributos *Natr* usado para controlar os registros de tipo "atributo" que são armazenados após o registro da entidade, o número de relacionamentos *Nrel* aos quais esta entidade está ligada, usado de forma semelhante ao anterior para controlar o registro que contém os nomes desses relacionamentos; e o número de objetos complexos *Narv* dos quais esta entidade participa para controlar os registros do tipo "árvore" que seguem. A sequência de gravação para os registros do tipo *entidade simples* é ilustrada na figura 4.7.

O registro referente a *relacionamentos*, figura 4.8, engloba os seguintes campos: o atributo *Chave*, a indicação se é um relacionamento *Total/Parcial*, a *Direção* para relacionamentos totais, os nomes das entidades que liga (*NE1* e *NE2*), os valores das cardinalidades envolvidas nesse relacionamento (*Card1* e *Card2*), o número *Natr* de atributos próprios desse relacionamento - usado para controlar os registros do tipo "atributo" que seguem

Tipo	NOME	Chave	Natr	Nrel	Narv
------	------	-------	------	------	------

Figura 4.6 - Estrutura de dados para o registro ENTIDADE.

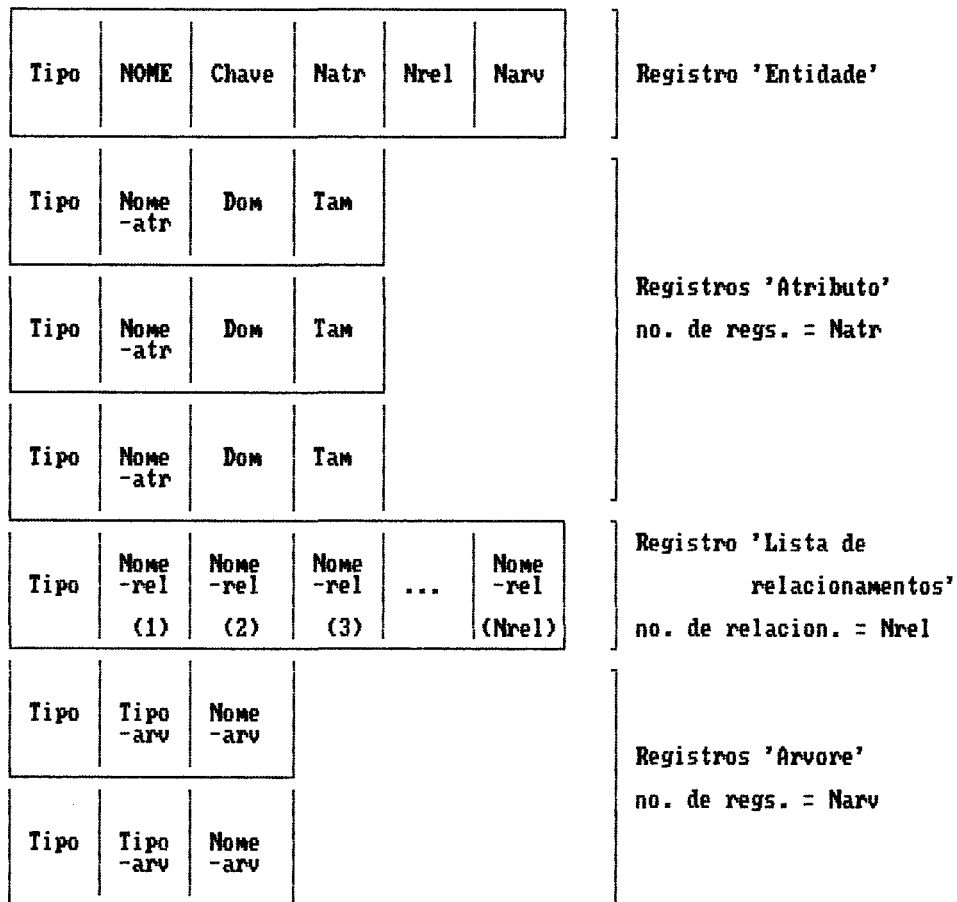


Figura 4.7 - Sequencia de gravacao para os registros referentes ao esquema de uma ENTIDADE.

Tipo	NOME	Chave	Total/ Parcial	Dir	NE1	NE2	Card1	Card2	Natr	Narv
------	------	-------	-------------------	-----	-----	-----	-------	-------	------	------

Figura 4.8 - Estrutura de dados para o registro RELACIONAMENTO.

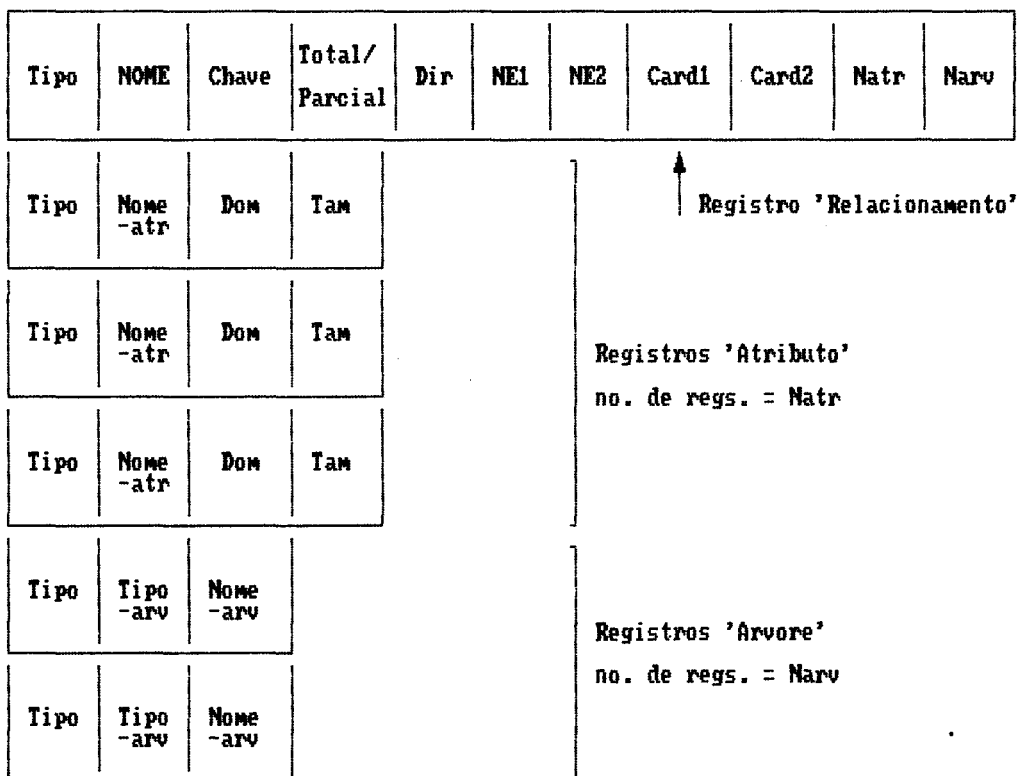


Figura 4.9 - Sequencia de gravacao para os registros referentes ao esquema de um RELACIONAMENTO.

este registro, e o número de objetos complexos de que participa - *Narv*. A sequência de gravação para os registros do tipo *relacionamento* é mostrada pela figura 4.9.

Para o registro de *agregações*, figura 4.10, as informações são: nome do relacionamento que identifica a agregação - *Nome-rel*, o número de atributos da agregação *Natr* - cujos registros seguem o da agregação, o número dos relacionamentos *Nrel* aos quais está ligada a agregação e que controla o registro com os nomes dos relacionamentos que segue os registros de atributos; o número de objetos complexos *Narv* dos quais a agregação participa e que controla os registros que contém os nomes desses objetos. Os registros do tipo *agregação* são gravados na sequência mostrada pela figura 4.11.

Os registros do tipo *generalização* ou *hierarquia*, figura 4.12, têm registros semelhantes para armazenar, nó a nó, os seguintes campos: *NOME* da árvore, nome do nó-pai (*Nome-pai*), o número de nós-filho do nó em questão (*Nfilh*) para controlar o registro que contém os nomes dos nós-filho desse nó. Os registros do tipo *generalização* ou *hierarquia* têm a mesma estrutura e são gravados de forma análoga conforme a sequência mostrada pela figura 4.13.

O registro de *atributos*, figura 4.14, contém os seguintes campos: nome do atributo - *Nome-atr*, domínio - *Dom* (que pode assumir os valores: caracter, real e inteiro), tamanho - *Tam* (caso o domínio seja caracter indica o número máximo de caracteres

Tipo	NOME	Nome -rel	Natr	Nrel	Narv
------	------	--------------	------	------	------

Figura 4.10 - Estrutura de dados para o registro AGREGACAO.

Tipo	NOME	Nome -rel	Natr	Nrel	Narv	Registro 'Agregacao'	
Tipo	Nome -atr	Dom	Tam				Registros 'Atributo' no. de regs. = Natr
Tipo	Nome -atr	Dom	Tam				
Tipo	Nome -atr	Dom	Tam				
Tipo	Nome -rel (1)	Nome -rel (2)	Nome -rel (3)	...	Nome -rel (Nrel)		Registro 'Lista de relacionamentos' no. de relacion. = Nrel
Tipo	Tipo -arv	Nome -arv					Registros 'Arvore' no. de regs. = Narv
Tipo	Tipo -arv	Nome -arv					

Figura 4.11 - Sequencia de gravacao para os registros referentes ao esquema de uma AGREGACAO.

Tipo	NOME	Nome -pai	Nfilh
------	------	--------------	-------

Figura 4.12 - Estrutura de dados para registros do tipo GENERALIZACAO ou HIERARQUIA.

Tipo	NOME	Nome -pai	Nfilh		
					Registro 'Generalizacao' ou 'Hierarquia'
Tipo	Nome -filh (1)	Nome -filh (2)	Nome -filh (3)	...	Nome -filh (Nfilh)
					Registro 'Lista de nos-filho' no. de filhos = Nfilh

Figura 4.13 - Sequencia de gravacao para os registros referentes ao esquema de uma GENERALIZACAO ou HIERARQUIA.

Tipo	Nome -atr	Dom	Tam
------	--------------	-----	-----

Figura 4.14 - Estrutura de dados para o registro ATRIBUTO.

do atributo; se o domínio for inteiro, número máximo de dígitos desse número; caso real, número de dígitos e de casas decimais que formam o número real.

O registro do tipo *árvore*, figura 4.15, possui os campos: *Tipo* para indicar que é *árvore*, *Tipo-arv* que distingue generalizações e hierarquias, e *Nome-arv* que contém o nome da *árvore*.

Tipo	Tipo -arv	Nome -arv
------	--------------	--------------

Figura 4.15 - Estrutura de dados para o registro ARVORE.

O registro do tipo *lista de relacionamentos*, figura 4.16, é formado por um vetor que armazena nomes de relacionamentos - *Nome-rel*.

Tipo	Nome -rel (1)	Nome -rel (2)	Nome -rel (3)	...	Nome -rel (Nrel)
------	---------------------	---------------------	---------------------	-----	------------------------

Figura 4.16 - Estrutura de dados para o registro 'LISTA DE RELACIONAMENTOS'.

O registro do tipo *lista de nós-filho*, figura 4.17, é formado por um vetor que armazena nomes de nós-filho de árvores (*Nome-filh*).

Tipo	Nome -filh	Nome -filh	Nome -filh	...	Nome -filh
	(1)	(2)	(3)		(Nfilh)

Figura 4.17 - Estrutura de dados para o registro 'LISTA DE NOS-FILHO'.

4.2 - EXEMPLO DE UTILIZAÇÃO

Para melhor entendimento do funcionamento do sistema desenvolvido, segue-se um exemplo (figura 4.18) de especificação de um esquema E-R e seu armazenamento interno.

Para especificação da entidade A, por exemplo, o usuário precisa definir os seguintes campos:

- * tipo do objeto - Entidade;
- * nome do objeto (entidade) - A;
- * nome do atributo chave de A (K_A), seu domínio e tamanho;
- * nome dos atributos de A (A_1, A_2), domínios e tamanhos.

As figuras 4.19 e 4.20, mostram as telas com as quais o usuário define o esquema de A.

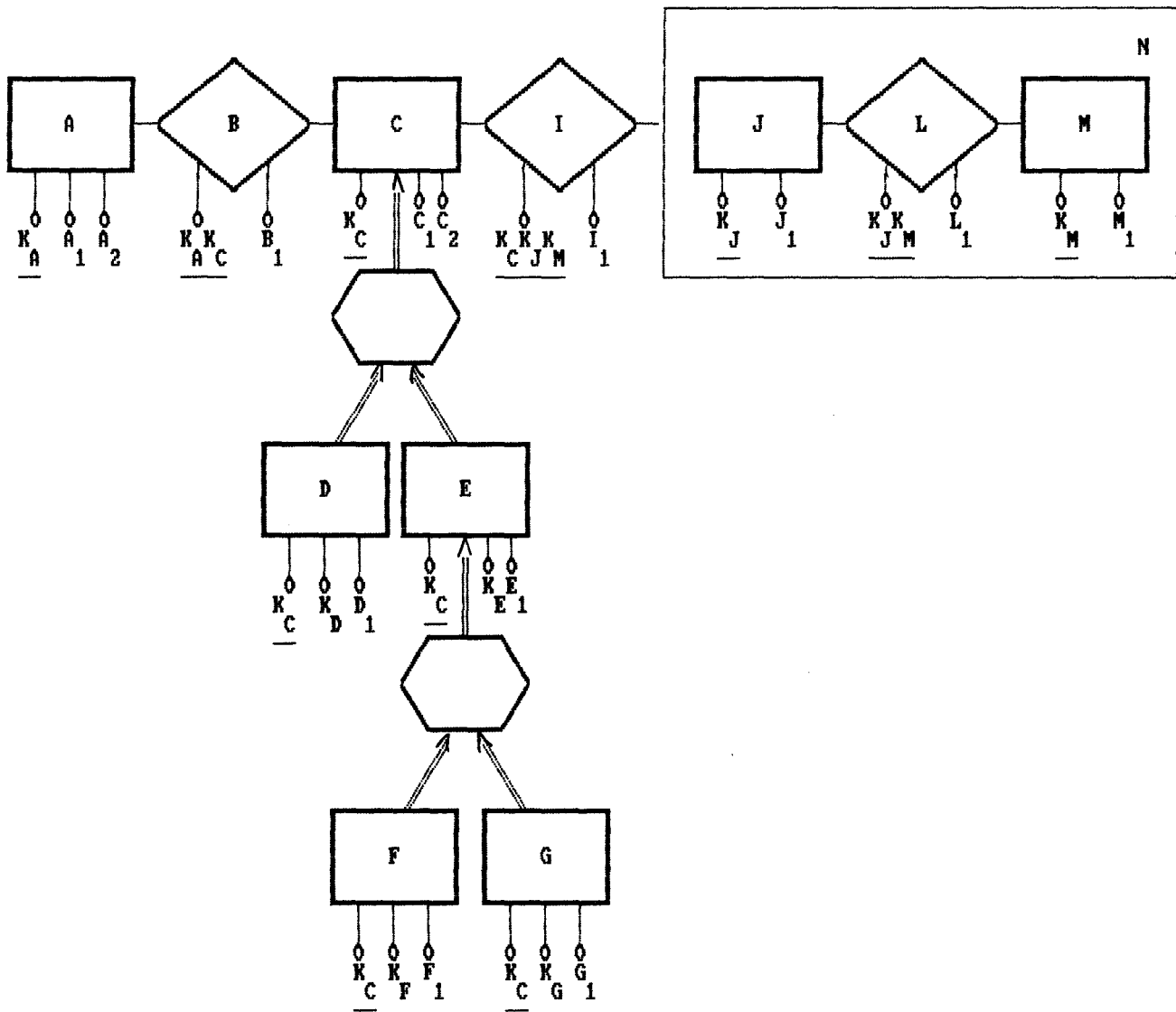


Figura 4.18 - Exemplo para mostrar Especificação e Armazenamento de Esquemas E-R.

Projeto Logico

ESQUEMA E - R

Menu Principal

1. Criar
2. Consultar
3. Atualizar
4. Imprimir
5. Ler
6. Gravar
7. Encerrar

Tecla sua opcao: 1

Tecla nome do esquema:

ESCOLA

Selecione objeto:

- Entidade
- Relacionamento
- Agregacao
- Generalizacao
- Hierarquia

Tecla sua opcao: E

Tecla nome do objeto

ALUNO

Figura 4.19 - Tela-exemplo (1) para geracao de objeto E-R (Entidade).

Entidade: **ALUNO**

Tecla atributo-chave:

matricula

Selecione dominio:

Caracter

Inteiro

Real

Tecla sua opcao: I

Tecla tamanho:

6

Nome do atributo	Dominio	Tamanho
nome-aluno	C	25
endereço	C	35
idade	I	2
mensalidade	R	4.2

Figura 4.20 - Tela-exemplo (2) para geracao de objeto E-R (Entidade).

A definição de um esquema deve obedecer uma ordem de especificação, de forma que um relacionamento só possa ser especificado após a definição dos objetos que relaciona. Da mesma forma, um objeto complexo só pode ser definido após a criação de seus componentes. É mostrada, a seguir, a especificação completa de uma generalização usando o exemplo da figura 4.18.

Para gerar uma generalização (hierarquia ou árvore de agregações) é necessário que todos os seus nós já tenham sido previamente definidos. As entidades C, D, E, F e G são geradas de forma análoga à especificação da entidade A. Como é usual dar a uma generalização (hierarquia) o mesmo nome do elemento-raiz, adotou-se que o nome da árvore será o mesmo da raiz, sendo feita a distinção entre os dois objetos através do campo *Tipo*, que indicará se generalização/hierarquia ou entidade/agregação (pois são permitidas árvores de agregações).

Para especificar a generalização C, o usuário é instado a definir os seguintes campos:

- * tipo do objeto - Generalização;
- * nome do objeto (raiz da generalização) - C;
- * nome do nó-pai - C;
- * tipo dos nós da árvore (Entidade ou Agregação);
- * nomes dos nós-filho - D, E.

A interface se encarrega de modificar as chaves das entidades D e E, substituindo pela chave da entidade C, a fim de usufruir dos mecanismos de integridade associados a chaves para manutenção das dependências de inclusão e exclusão mútua entre os nós-filho

(D e E) da generalização C. As outras chaves de D e E passam a ser atributo não-chave. São também atualizados os campos que envolvem participação em árvores - *Narv* e *Cad(arv)*, de todas as entidades envolvidas (C, D e E). A seguir, é inserido na estrutura intermediária o nó correspondente a generalização C.

Para a geração da sub-árvore da generalização C, cuja raiz é a entidade E, são necessárias as seguintes definições do usuário:

- * tipo do objeto - Generalização;
- * nome do objeto - E;
- * nome do nó-pai - C;
- * tipo dos nós da árvore (E ou AG);
- * nomes dos nós-filho - F, G.

São executadas então as modificações nos nós-filho para manter a herança das chaves, atualizados os campos relativos a árvores, conforme já foi explicado. E então realizada a inserção do registro correspondente a generalização E, na estrutura intermediária.

As estruturas de dados mais interessantes que resultaram da geração desses objetos são mostradas nas figuras 4.21 a 4.25.

O objeto que segue a entidade C (apontado por *Prox*) é determinado através de função *hash*, o mesmo ocorrendo com todos os demais objetos gerados. A distribuição dos objetos do exemplo da figura 4.18, na estrutura intermediária, depende somente de seus nomes e da função *hash* aplicada sobre eles.

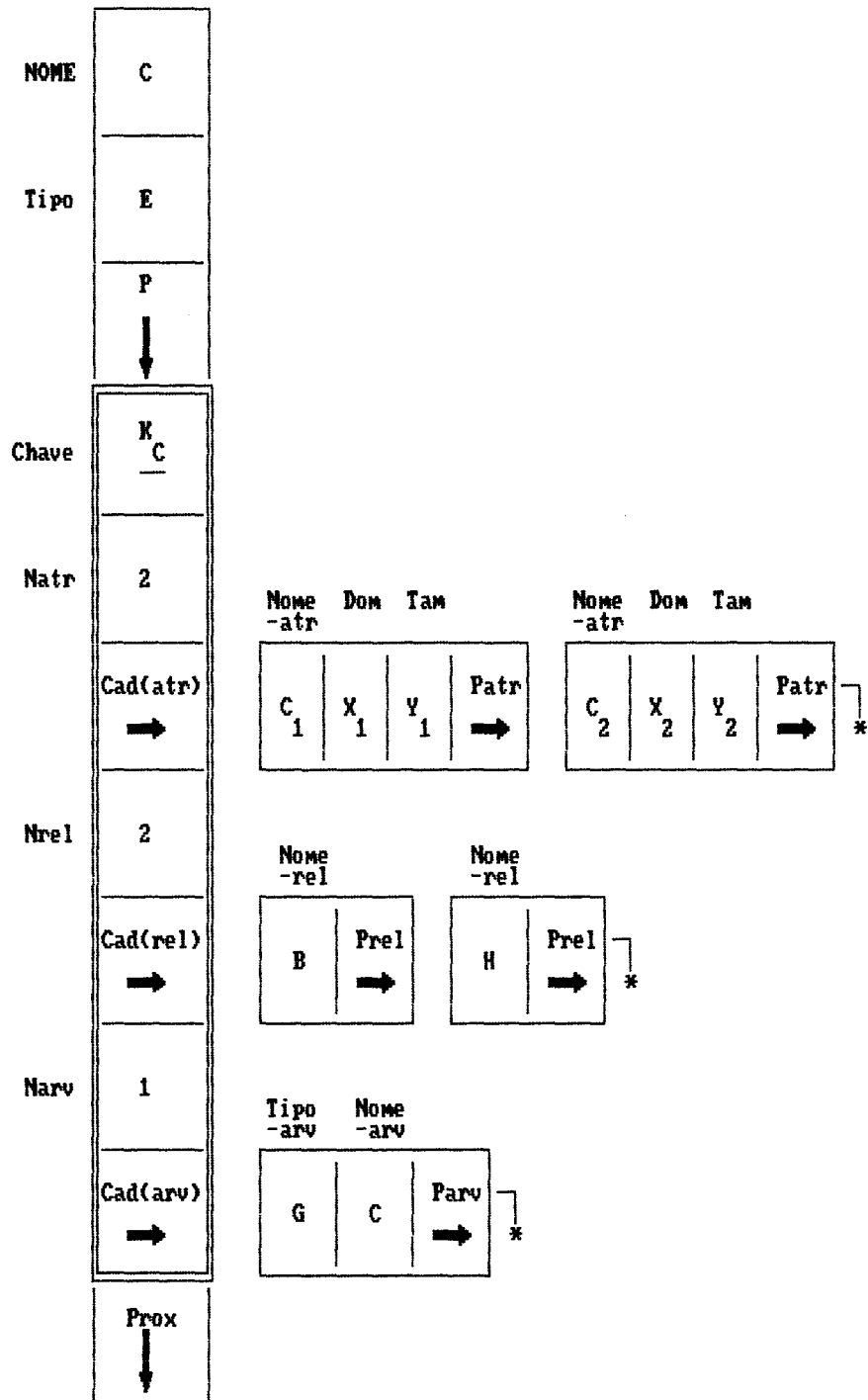


Figura 4.21 - Estrutura de dados para a entidade C.

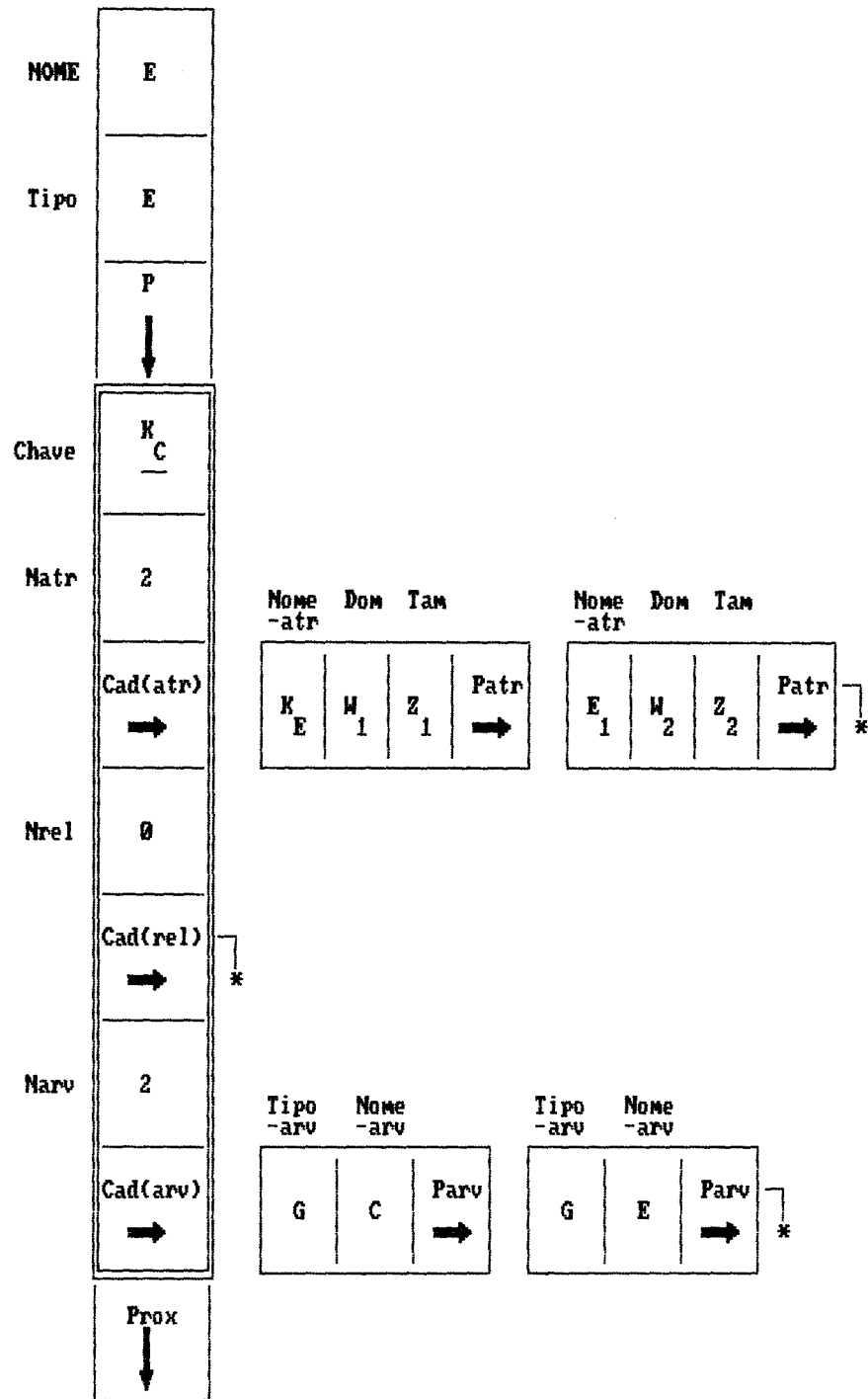


Figura 4.22 - Estrutura de dados para a entidade E.

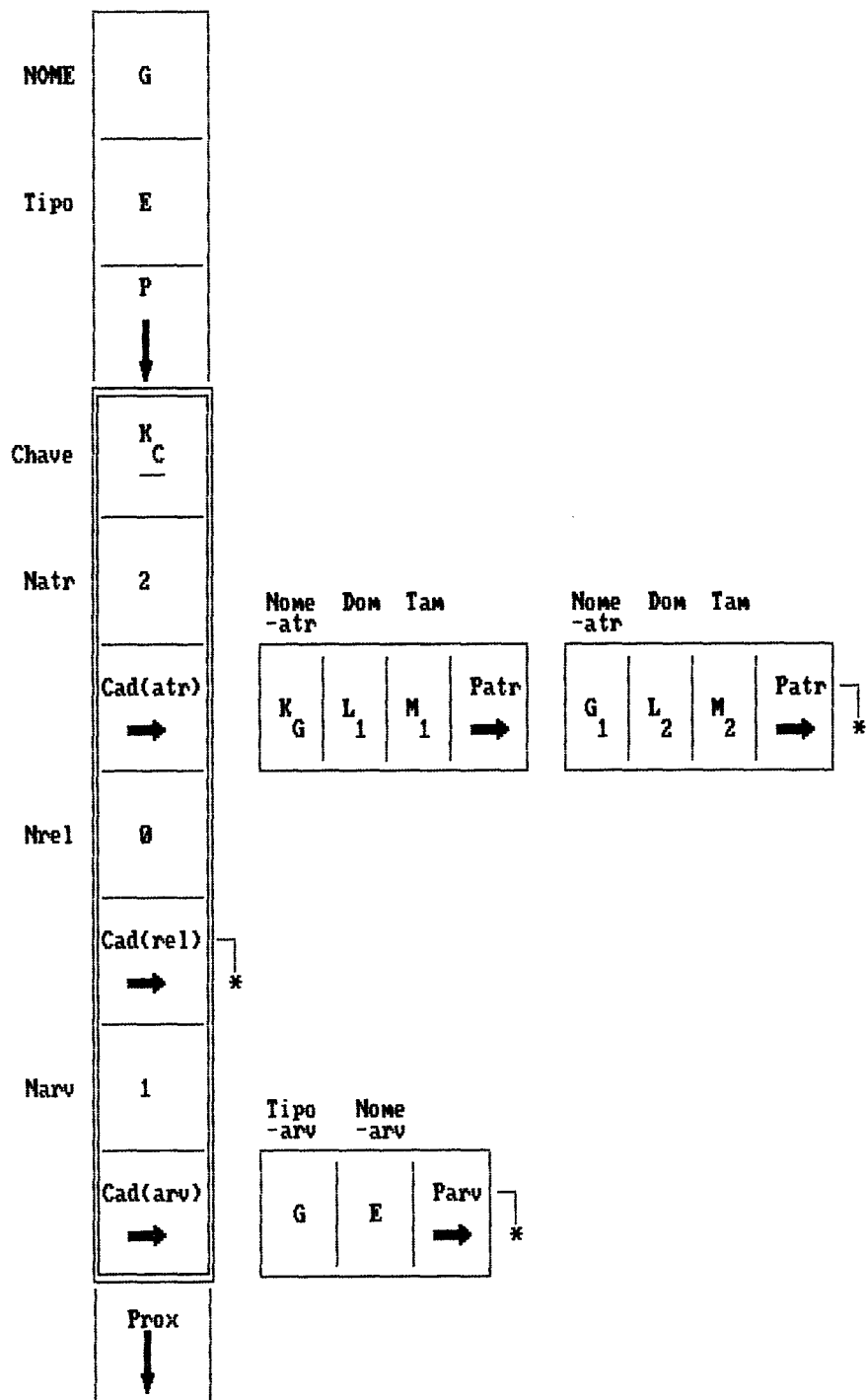


Figura 4.23 - Estrutura de dados para a entidade G.

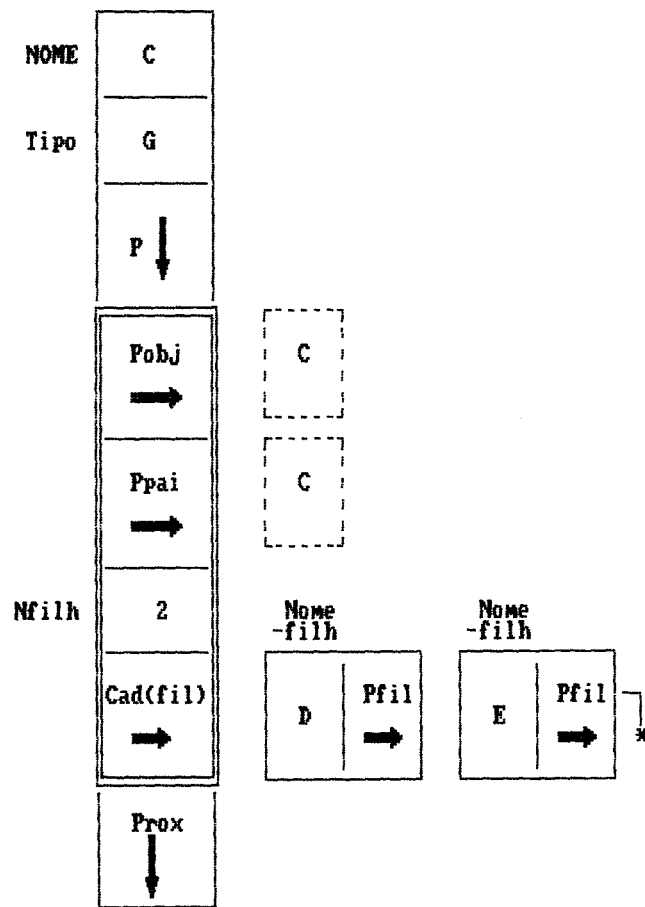


Figura 4.24 - Estrutura de dados para a generalizacão C.

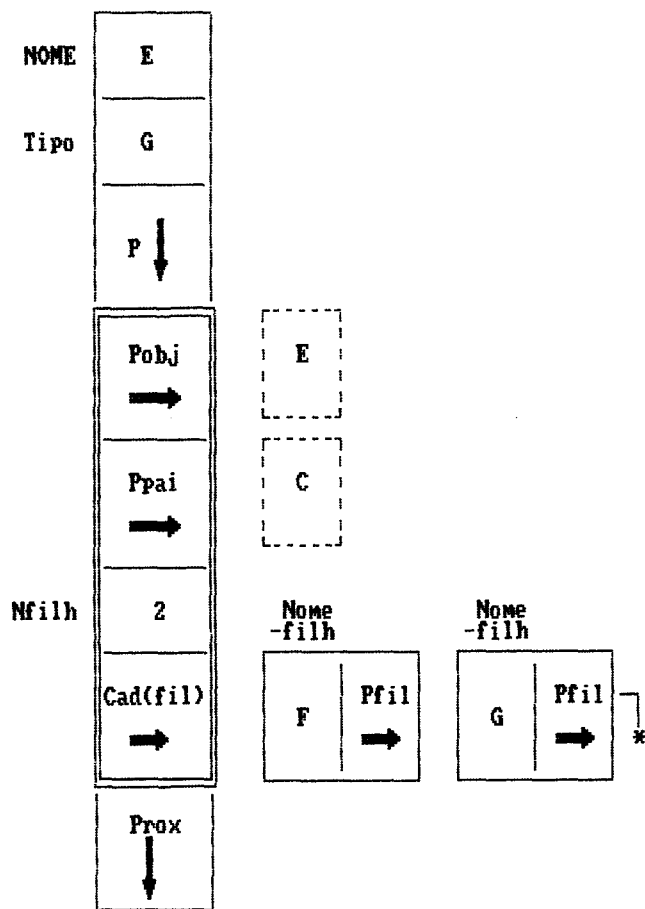


Figura 4.25 - Estrutura de dados para a generalizacão E.

4.3 - CONSIDERAÇÕES DE EFICIENCIA

Conforme mencionado anteriormente, o protótipo verifica as funções de acesso às tabelas através de módulos que efetuam mapeamento de *hashing*. Os algoritmos utilizados estão descritos em [FLOY87]. A técnica de acesso utilizada é aquela conhecida como "*open hashing*", que é descrita em [AHO86].

Uma tabela de "*open hash*" consiste em um vetor linear de ponteiros, cada um deles apontando para o início de uma lista de símbolos.

A idéia geral é que, para cada consulta ou atualização, seja gerado o valor *hash* que determina o índice do vetor onde se encontra o ponteiro para uma lista de objetos, dentre os quais estará aquele procurado. Desta forma, é necessário apenas percorrer a lista associada a tal ponteiro. A utilização desse método leva a uma busca que gira em torno de 1% da lista completa, em média [FLOY87]. Em outras palavras, o tratamento dado para colisões é a criação de uma cadeia de *overflow*, onde idealmente não há colisões. Em tal caso, o número de acessos necessários para encontrar um dado objeto é aproximadamente igual ao número de objetos armazenados dividido pelo número de ponteiros (tamanho máximo do vetor). Quanto maior o vetor de ponteiros, menos colisões no cálculo *hash* e, por conseguinte, listas menores de objetos para pesquisar.

A escolha de uma função *hash* adequada é essencial, a fim de manter uma distribuição uniforme, equilibrada dentro do vetor de

ponteiros, de forma a evitar que alguns sejam sobrecarregados com extensas listas de objetos, em prejuízo de outros sub-utilizados.

Floyd [FLOY87] analisa quatro funções *hash* para três diferentes tipos de texto:

- a) (soma dos caracteres (do identificador) + comprimento) mod 101;
- b) (primeiro caracter x constante + último + comprimento) mod 101;
- c) HASHPJW mod 101, função *hash* descrita e analisada em [AH086];
- d) CRC-16 mod 101, função que computa a verificação de redundâncias cíclicas em códigos.

Vale observar que o valor 'mod 101' é decorrente da adoção de um tamanho máximo de 101 posições para o vetor de ponteiros.

Seguindo [FLOY87], a tese aplica uma função estatística, $U(h,t)$, descrita por Aho, Sethi e Ullman em [AH086]. Esta função caracteriza a uniformidade da distribuição de um conjunto de símbolos (t), pelos ponteiros de um vetor, quando do cálculo da função *hash* (h). Nesta tese, os símbolos são os objetos: entidade, relacionamento, agregação, generalização e hierarquia. Quanto menor o número obtido através de $U(h,t)$ mais uniforme é a distribuição - uma distribuição perfeitamente randômica está próxima de 1,00.

Como o tempo de busca de um elemento (objeto) em uma lista decai com a proximidade do elemento à cabeça da lista, [FLOY87]

aplica a estratégia de organizar a lista no sentido de que a cada busca o elemento seja deslocado para o início. Isto leva a uma otimização da organização para buscas. Em geral, quanto menos uniforme for a distribuição, ou seja, maiores as listas ligadas aos ponteiros do vetor, então maior o melhoramento obtido com a aplicação da técnica de deslocamento do elemento procurado para a cabeça da lista. No entanto, essa melhoria não é mensurável com a aplicação da função $U(h,t)$ porque esta baseia-se somente na função *hash* utilizada e no tipo de texto considerado (nesta tese, os nomes dos objetos).

4.4 - CONSULTAS: NAVEGAÇÃO DENTRO DO ESQUEMA

A interface permite dois tipos distintos de consultas: a. consultas simples, ou seja, com objetivo de visualizar um determinado esquema; b. consultas que precedem uma atualização. O primeiro não envolve maiores problemas, já o segundo caso implica grandes discussões sendo sugerida a solução apresentada em [CASAB88], que mostra o estudo e soluções possíveis para os problemas que envolvem o bloqueio ou a propagação de operações de remoção e inserção em esquemas com dependência de inclusão. Para atualizações do tipo eliminação ou inserção de um objeto inteiro bastaria a geração ou eliminação de um objeto em alguma lista de objetos da estrutura intermediária.

Para consultar um objeto o usuário deve definir as seguintes informações:

* nome do esquema (caso não esteja trabalhando com algum);

* tipo do objeto;

* nome do objeto.

A partir desses dados é realizada uma busca dentro da estrutura intermediária para verificar a existência desse objeto, sendo em seguida exibidas para o usuário as informações que constam da estrutura acessada.

Utilizando o exemplo da figura 4.18, e as estruturas de dados geradas para esses objetos (figuras 4.21 a 4.26), é mostrado a seguir como é possível navegar dentro dessas estruturas e acessar todos os objetos que compõem a generalização C.

A consulta à generalização C exige que o usuário informe apenas o tipo e nome do objeto que quer consultar. A partir do nome do objeto (C) é gerado um valor *hash* e selecionado o ponteiro da lista onde a generalização C está armazenada.

Como existe uma repetição de nomes (há uma entidade e uma generalização nomeadas de C), o valor *hash* gerado é igual para os dois casos, o que leva a armazenar os dois objetos na mesma lista (ou seja, ligados a um único ponteiro do vetor de ponteiros). Isto pode ser evitado se a(s) função(ões) *hash* levarem em consideração também o valor do campo *Tipo*, o que não é interessante pois esses dois objetos estão intimamente associados, sendo que uma vez requerido o acesso a um deles, é quase certa a necessidade de acessar também o outro. Essa proximidade ainda não foi completamente aproveitada mas esse arranjo é o mais interessante para futuros melhoramentos na cadeia de acessos a árvores.

Ao acessar a generalização C verifica-se através da cadeia de nós-filho de C, a existência de D e E (vide figura 4.24), sendo o usuário instado a decidir qual nó deve ser consultado a seguir.

Uma vez decidido pelo nó-filho E (entidade), observa-se que é raiz de uma generalização (vide figura 4.22), sendo solicitada uma consulta à generalização E, quando verifica-se a existência dos nós-filho F e G (vide figura 4.25); o usuário é então levado a escolher qual caminho deve ser percorrido na sequência.

Para melhor visualização das facilidades para a navegação de consultas obtidas com as estruturas implementadas, é explicado a seguir como seria possível percorrer toda a árvore de generalização anterior, pelo caminho inverso ao que foi gerada.

Supõe-se que o usuário solicite uma consulta a entidade G, sendo-lhe então fornecida a informação de que G participa da generalização E. Observe-se que o atributo chave de E já possibilita ao usuário perceber que E pertence a uma estrutura.

Decidida a próxima consulta cuja escolha recai sobre a generalização E, o usuário é informado da existência de uma entidade F, irmã do nó G, e que a generalização E é sub-árvore da generalização C. A consulta a C permite acessar a entidade D que é irmã do nó-entidade E, e pertence a essa generalização.

Outros tipos de consulta são processados interativamente de forma análoga. Por exemplo para uma dada agregação, o usuário é informado sobre os objetos que encapsula e os relacionamentos aos quais está ligada.

O exemplo desta seção foi escolhido para ilustrar algumas das características do sistema, bem como indicar a utilização das estruturas de dados implementadas.

4.5 - RESUMO

Este capítulo discutiu o protótipo implementado que se restringiu aos módulos DIALOGO e ESQUEMA. Foram apresentadas as estruturas de dados utilizadas na implementação do protótipo para validação das idéias propostas nesta tese. E mostrado também um exemplo de utilização dessas estruturas e discutidos alguns pontos concernentes à eficiência alcançada pelo sistema.

O próximo capítulo apresenta extensões do trabalho, questões que exigem estudo futuro e conclusões.

5 - CONCLUSÃO

Este capítulo apresenta um breve perfil do trabalho realizado e aponta algumas áreas onde devem ser envidados maiores esforços no sentido de complementar a ferramenta especificada.

5.1 - *CONSIDERAÇÕES FINAIS*

Esta tese descreveu uma interface E-R para o modelo relacional, que permite ao usuário definição e manipulação de objetos E-R, traduzindo-os sob forma de esquemas e relações. A tese descreve, no capítulo 4, um protótipo implementado que corresponde à especificação no que diz respeito a diálogo e manuseio de esquema. Além disso, a tese se ocupou da definição de um conjunto de funções que permitisse a consulta e atualização quer do projeto lógico E-R, quer dos dados relacionais, e a discussão da viabilidade de implementação dessas funções. O objetivo desse sistema é automatizar o processo de mapeamento do esquema E-R para o Banco de Dados Relacional e a criação das tabelas relacionais correspondentes, as duas etapas que exigem alto grau de conhecimento do Modelo Relacional, envolvendo a aplicação de heurísticas e comandos de uma forma organizada.

Com essa automatização obtemos uma ferramenta que possibilita ao usuário centralizar seus esforços na criação de um projeto lógico de banco de dados utilizando uma versão estendida do

Modelo E-R, um dos modelos que maior riqueza semântica possui e que é de fácil assimilação para usuários pouco especializados.

Uma interface com tais características possibilitaria a um usuário pouco especializado projetar um banco de dados concentrando seus esforços na fase mais importante dessa tarefa que é capturar a semântica do mundo real, e expressá-la através de um dos modelos considerado mais adequado para tal - o E-R.

O protótipo da interface obedeceu à filosofia de que o mapeamento entidade-relacionamento para um banco de dados relacional deveria ser transparente ao usuário. O protótipo permite a definição e armazenamento de esquemas de "objetos" descritos segundo o modelo E-R aqui utilizado, com extensões que incluem agregação, generalização e hierarquia de tipos, independente dos dados que serão armazenados no Banco de Dados Relacional físico.

O sistema proposto foi estruturado em módulos que realizam funções estanques: o Módulo ESQUEMA (que gera o esquema E-R), Módulo MANUSEADOR (efetua a manipulação das relações e sua transação), Módulo TRADUTOR (que verifica a entrada e consulta dos dados relacionais e é responsável pela interação com o SGBD hospedeiro); e o Módulo DIALOGO (para interação com o usuário). Esta arquitetura permite que o sistema se adapte a diferentes SGBDs e possa utilizar linguagens distintas para interação com o usuário.

O protótipo implementado permite a modificação de um esquema previamente definido através da inclusão, eliminação ou modificação de seus objetos, tantas vezes quantas necessárias durante a fase inicial de projeto. Uma vez concluída esta etapa de geração do esquema E-R é acionado o módulo TRADUTOR que gera os comandos de definição das relações correspondentes ao esquema. Não há, portanto, durante o projeto de um esquema, interação com o SGBD. Consultas e modificações dos objetos do esquema, durante toda a fase de sua geração, são feitas diretamente às estruturas de dados manipuladas pela interface.

5.2 - EXTENSÕES SUGERIDAS

O trabalho desenvolvido nesta tese comporta várias extensões, tanto de implementação quanto de especificação. Dentre as extensões sugeridas pode ser citada, por exemplo, a implementação de procedimentos que permitam a manipulação de ocorrências dos objetos E-R, com consultas e atualizações, traduzidas em operações relacionais e incorporando gatilhos, correspondendo ao módulo MANUSEADOR.

Vale observar que este tipo de extensão não é de forma alguma trivial, pois exige incorporar conceitos como: otimização de consultas, manutenção de integridade, atendimento a multi-usuários. Além disto, o problema torna-se maior quando se considera que o esquema do banco de dados não é estático, desta forma, procedimentos-padrão de otimização de consultas ou mapeamento de atualizações não podem ser utilizados na ferramenta

já que esta tem como um dos objetivos prover o usuário com facilidades para modificar dinamicamente o esquema. Assim, é possível que uma mesma consulta precise ser processada de duas maneiras totalmente diferentes se houver modificações de esquema durante uma sessão.

Um problema menos complexo é a mudança de esquema do banco de dados já carregado ignorando aspectos de eficiência ou de possíveis consultas e atualizações. Ainda assim, permitir este tipo de atividade envolve uma série de complicações adicionais principalmente no que diz respeito à manutenção de integridade local e global.

Outro ponto a considerar desta classe de extensão é o que ocorre quando se modifica alguma restrição de integridade do esquema de um banco de dados. Se o banco de dados está vazio, é necessário percorrer todo o esquema para fazer os ajustes necessários nos demais objetos, de forma a garantir a consistência global. Em alguns casos, é possível que seja necessário codificar novos gatilhos e até mesmo modificar gatilhos já existentes. O problema se torna crítico se o banco de dados já está carregado. Neste caso, além dos procedimentos anteriores é preciso fazer a consistência entre todas as ocorrências afetadas pela mudança de esquema ou sujeitas aos novos gatilhos.

Os parágrafos anteriores indicam alguns dos itens que devem ser considerados na implementação do MANUSEADOR. Desta forma, este tipo de extensão pode até não ser factível dependendo do número de graus de liberdade que se lhe queira adicionar.

Uma outra extensão é a ampliação do módulo ESQUEMA de modo a permitir a modificação do esquema E-R após a geração das relações correspondentes, com todas suas implicações (por exemplo, novo mapeamento dos objetos modificados e criação de novas relações ou alteração daquelas existentes). Ainda outra sugestão é incorporar ao sistema opções que impliquem num mapeamento mais compacto para facilitar manutenção de dependências de inclusão [BRAG88]. A implementação de um Módulo IMPRESSOR completo viria a permitir a impressão de detalhes do esquema E-R e seu mapeamento relacional.

Seria ainda de grande interesse o estudo das consequências geradas caso se modificassem algumas das restrições (proibições) impostas neste trabalho a título de simplificação, por exemplo:

- a. relaxar as proibições de relacionamentos totais encadeados;
- b. permissão de aumentar/diminuir o número de atributos em relações após sua criação (como definir valores de atributos caso haja aumento, e como efetuar o tratamento de nulos);
- c. permitir ao usuário incluir/eliminar restrições de dependências de inclusão e seu reflexo sobre as relações geradas.

Algumas restrições da tese, por exemplo a referente ao encadeamento de relacionamentos totais, podem ser relaxadas. Outras, como por exemplo considerar relacionamentos n-ários, ao

invés de binários apenas, não podem ser consideradas sem mudar completamente as premissas básicas da tese.

Motro [MOTR80] defende a idéia de que estruturas rígidas que possuem arquitetura estruturada, baseada em registros, facilitam a recuperação de informações; no entanto, impedem que o banco de dados acompanhe a evolução constante do ambiente que retrata. Propõe como solução uma arquitetura binária baseada em coleções desestruturadas de fatos que podem ser consultados através de uma linguagem 'padrão' fundamentada em predicados. A recuperação de um fato, definido da mesma forma que uma agregação - duas entidades associadas via um relacionamento binário - é realizada através de dois mecanismos: pesquisa, que é exame da vizinhança do fato; e tentativa, que é uma consulta seguida de falha seguida de consulta.

Em uma estrutura composta de registros de tamanhos variáveis como nesta interface, a recuperação de informações não é de todo trivial. A implementação via listas ligadas impõe um caminho a ser percorrido para localizar o dado, o que ocasiona problemas com o tempo de acesso. Outro ponto crucial é a necessidade de se trabalhar num ambiente heterogêneo, formado pelos dois modelos envolvidos, e extrair os aspectos mais favoráveis e fundamentais de cada um sem perdas no outro lado.

Uma deficiência da interface aqui proposta é que estabelece um mapeamento único para implementação do banco de dados. Além

disto, não há preocupação em otimizar o banco de dados relacional gerado.

Como é finalidade deste trabalho usufruir da semântica do modelo E-R para proporcionar maior compreensão e documentação do sistema, e como as fases de planejamento e análise do desenvolvimento do sistema levam a representações extensas do modelo que requerem sucessivos refinamentos, acreditamos que um futuro melhoramento seria a implementação dos conceitos descritos no "Entity Model Clustering" de Feldman e Miller [FELD86].

Acredita-se que após a aplicação desse modelo de agrupamento de entidades, outro melhoramento seria a implementação de uma linguagem gráfica para a interface, que viria a constituir o módulo DIALOGO, pois um dos objetivos do agrupamento é reduzir o número de entidades num contexto (tela) para facilitar sua compreensão (representação visual).

O que julgamos mais adequado seria a existência de três linguagens a serem suportadas pela interface:

- a. gráfica - dirigida aos usuários não-técnicos;
- b. orientada por menus - para atender usuários mais familiarizados com a área de banco de dados, capazes de estabelecer restrições de integridade, por exemplo;
- c. formal, semelhante no formalismo do cálculo ou álgebra relacional - para servir aos profissionais que administrem o banco de dados.

A inexistência de formalismos para o modelo E-R nos obriga a refletir sobre as questões de normalização e otimização das relações geradas pelo mapeamento imposto. O trabalho de [CHUN83] define a noção de Relações E-R e introduz formas normais E-R para decompor o diagrama - esquema E-R até normalizá-lo e obter uma correspondência de 1:1 com o esquema relacional.

5.3 - RESUMO

Este capítulo apresentou uma visão geral do trabalho realizado e as suas conclusões. Foram sugeridas futuras extensões que seriam de interesse para tornar a interface mais operacional e indicados alguns pontos que requerem um estudo mais profundo.

BIBLIOGRAFIA

- [AHO86] - Aho, A.V.; Sethi, R. e Ullman, J. D. "Compilers". Addison-Wesley Publish Company, pag. 433-440, 1986.
- [ATZE83] - Atzeni, P. e Chen, P.P. "Completeness of Query Languages for the Entity-Relationship Model" in [CHEN83a].
- [BRAE85] - Braegger, R.F.; Dudler, A.M.; Rebsamen, J. e Zehnder, C.A. "Gambit: An Interactive Database Design Tool for Data Structures, Integrity Constraints, and Transactions". IEEE Transactions on Software Engineering, SE-11(7), 1985.
- [BRAG88] - Braga, A. P. "Tradução de Esquemas Entidade-Relacionamento Estendido para Esquemas Relacionais". IBM Centro Científico Rio, Relatório Técnico CCR 068, Nov/1988.
- [BRAG89] - Braga, A. P.; Casanova, M. A. e Tucherman, L. "Trivialização de Dependências de Inclusão em Esquemas Conceituais Relacionais". Anais do IV Simpósio Brasileiro de Banco de Dados, Campinas, 1989.
- [CASAB4] - Casanova, M. A.; Fagin, R. e Papadimitriou, C. "Inclusion Dependencies and their Interaction with Functional Dependencies". Journal of Computer and System Sciences 28(1), February 1984.
- [CASAB8] - Casanova, M.A.; Tucherman, L. e Furtado, A.L. "A Monitor Enforcing Referencial Integrity". Anais do

III Simpósio Brasileiro de Banco de Dados, 1-16, 1988.

- [CHEN76] - Chen, P.P. "The Entity-Relationship Model: Toward a Unified View of Data". ACM Transactions on Database Systems 1(1), 1976.
- [CHEN83a] Chen, P.P. "Entity-Relationship Approach to Information Modeling and Analysis". North-Holland, 1983.
- [CHEN83b] Chen, P.P. "A Preliminary Framework for Entity-Relationship Models" in [CHEN83a].
- [CODD79] - Codd, E.F. "Extending the Database Relational Model to Capture More Meaning". ACM Transactions on Database Systems 4(4), 1979.
- [CHUN83] - Chung, I.; Nakamura, F. e Chen, P.P. "A Decomposition of Relations Using the Entity-Relationship Approach" in [CHEN83a].
- [DATE77] - Date, C.J. "An Introduction to Data Base Systems". 2nd Ed., Addison Wesley, 1977.
- [DELG86] - Delgado, A.L.N. e Magalhães, L.P. "Assimilando a Semântica de PAC através de um Modelo de Dados". Anais do VI Congresso da SBC, Recife, 1986.
- [DELG87] - Delgado, A.L.N. "Banco de Dados no Contexto de Projeto Auxiliado por Computador". Tese de Mestrado em Engenharia Elétrica - UNICAMP, Janeiro/1987.
- [DITT87] - Dittrich, K. R.; Gotthard, W. e Lockemann, P. C. "DAMOKLES - The Database System for the UNIBASE Software Engineering Environment". Database Engineering, March 1987.

- [ELMA83] - Elmasri, R. e Wiederhold, G. "GORDAS: A Formal High-Level Query Language for the Entity-Relationship Model" in [CHEN83a].
- [ELMA85] - Elmasri, R.; Weeldreyer, J. e Hevner, A. "The Category Concept: An Extension to the Entity-Relationship Model". North-Holland, 1985.
- [FELD86] - Feldman, P. e Miller, D. "Entity Model Clustering: Structuring a Data Model By Abstraction". The Computer Journal 29(4), 1986.
- [FLOY87] - Floyd, E.T. "Hashing for High-Performance Searching". Dr. Dobb's Journal, February, 1987.
- [FURT87] - Furtado, A.L.; Casanova, M.A. e Tucheran, L. "The CHRIS Consultant". Proc. 6th International Conference on Entity-Relationship Approach, 1987.
- [KENT83] - Kent, W. "A Simple Guide to Five Normal Forms in Relational Database Theory". Communications of the ACM 26(2), 1983.
- [MAIE83] - Maier, D. "The Theory of Relational Databases". Computer Science Press, 1983.
- [MAGA89] - Magalhães, L.P.; Delgado, A.L.N.; Ricarte, I.L.M.; Ruschel, R.C. e Olguin, C.J.M. "Implementação de um Banco de Dados Não-Convencional: A Experiência GERFAC/UnicOSMOS". Anais do IV Simpósio Brasileiro de Banco de Dados, Campinas, 1989.
- [MALH86] - Malhotra, A.; Tsalalikhin, Y.; Markowitz, H.M.; Pazel, D.P. e Burns, L.M. "Implementing an Entity-Relationship Language on a Relational Data Base".

Relatório Técnico IBM, 1986.

- [MARK83] - Markowitz, H.M., Malhotra, A. e Pazel, D.P. "The ER and EAS Formalisms for System Modeling, and the EAS-E Language" in [CHEN83a].
- [MOTR84] - Motro, A. "Browsing in a Loosely Structured Database". ACM SIGMOD, 1984.
- [NAVAB3] - Navathe, S. e Cheng, A. "Database Schema Mapping" in Entity-Relationship Approach to Software Engineering, North Holland, 1983.
- [NOGUS88] - Nogueira, M.L. e Medeiros, C.B. "REVER: Uma Interface Semântica para Bancos de Dados Relacionais". XIV Conferencia Latinoamericana de Informática e 17as. Jornadas Argentinas de Informática e Investigación Operativa, Buenos Aires, Setembro 1988.
- [PARE86] - Parent, C. e Spaccapietra, S. "An Algebra for a General Entity-Relationship Model". IEEE Transactions on Software Engineering SE-11(7), 1986.
- [PARK80] - Parkin, A. "Systems Analysis by Entity-Relationship Modelling: A Case Study". Computer Bulletin, December 1980.
- [PARK82] - Parkin, A. "Data Analysis and System Design by Entity-Relationship Modelling". The Computer Journal 25(4), 1982.
- [RICAB7] - Ricarte, I.L.M. "Sistemas de Gerência de Dados para Projeto Auxiliado por Computador". Tese de Mestrado em Engenharia Elétrica - UNICAMP, Maio/1987.
- [ROUSS84] - Roussopoulos, N. e Yeh, R.T. "An Adaptable Methodology

for Database Design". Computer, May, 1984.

- [SCHE80] - Scheuermann, P., Schiffner, G. e Weber, H. "Abstraction Capabilities and Invariant Properties Modelling Within the Entity-Relationship Approach" in Entity-Relationship Approach to Systems Analysis and Design, North Holland, 1980.
- [SCHI79] - Schiffner, G. e Scheuermann, P. "Multiple Views and Abstractions with an Extended-Entity-Relationship Model". Computer Languages, vol 4, 1979.
- [SHAV81] - Shave, M.J.R. "Entities, Functions and Binary Relations: Steps to a Conceptual Schema". The Computer Journal 24(1), 1981.
- [SMIT77] - Smith, J.M. e Smith, D.C.P. "Database Abstractions: Aggregation and Generalization". ACM Transactions on Database Systems 2(2), 1977.
- [SOUZ88] - Souza, S.M.F.M. "Manipulação de Banco de Dados através de Formulários". Tese de Mestrado em Ciência da Computação - UNICAMP, Setembro/1988.
- [TABO83] - Taborier, Y. e Nanci, D. "The Occurrence Structure Concept: An Approach to Structural Integrity Constraints in the Entity-Relationship Model" in [CHEN83a].
- [TEIC83] - Teichroew, D.; Germano, F. e Silva, L. "Applications of the Entity-Relationship Approach" in [CHEN83a].
- [TEOR86] - Teorey, T.J.; Yang, D. e Fry, J.P. "A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model". Computing

Surveys 28(2), 1986.

- [WAGN87] - Wagner, C.F. e Medeiros, C.B. "Um Modelo Binário Estendido para Entidade-Relacionamento". Anais do VII Congresso da SBC, Salvador, 1987.
- [WAGN88] - Wagner, C.F. "Implementing Abstraction Hierarchies". Proc. 7th International Conference on Entity-Relationship Approach, Roma, 1988.
- [WEBR83] - Webre, N.W. "An Extended Entity-Relationship Model and Its Use on a Defense Project" in [CHEN83a].
- [WELD80] - Weldon, J.L. "Using Data Base Abstractions for Logical Design". The Computer Journal 23(1), 1980.