Universidade Federal da Paraíba
Centro de Ciências e Tecnologia
Coordenação de Pós-Graduação em Informática

Um Sistema Baseado em Conhecimento para Integração de Esquemas de Banco de Dados

Isaac Douglas Moreira

Campina Grande - PB
Dezembro de 1991

Isaac Douglas Moreira

Um Sistema Baseado em Conhecimento para Integração de Esquemas de Banco de Dados

Dissertação apresentada ao Curso de MESTRADO EM SISTEMAS E COMPUTAÇÃO da Universidade Federal da Paraíba – Campina Grande, em cumprimento às exigências para obtenção do Grau de Mestre.

Área de Concentração: Ciência da Computação

Prof. Marcus Costa Sampaio
(Orientador)

Prof. Giuseppe Mongiovi
(Co-orientador)

Campina Grande
Dezembro - 1991

M835s Moreira, Isaac Douglas
Um sistema baseado em conhecimento para integracao de esquemas de banco de dados / Isaac Douglas Moreira. - Campina Grande, 1991.
139 f.

Dissertacao (Mestrado em Sistemas e Computacao) - Universidade Federal da Paraiba, Centro de Ciencias e Tecnologia.

1. Banco de Dados - Integracao 2. Sistemas Baseados em Conhecimento 3. Dissertacao - Mestrado em Sistemas e Computacao I. Sampaio, Marcus Costa II. Mongiovi, Giuseppe III. Universidade Federal da Paraiba - Campina Grande (PB) IV. Título

CDU 004.65(043)

Ao meu falecido Pai, Sr. Vicente, que sempre foi fonte de inspiração

À minha mãe, D. Antônia, uma lutadora, que não se intimida diante das adversidades.

Aos meus irmãos, que sempre foram companheiros.

Agradecimentos

Agradeço a Deus por te me concedido a graça de uma vida plena e repleta de oportunidades e bons amigos. À minha família, que sempre me deu apoio e esteve ao meu lado nos momentos alegres e difíceis. A meu orientador, Prof. Marcus Sampaio, que muito mais que um professor e orientador, foi sempre um amigo e um verdadeiro exemplo de competência, determinação e sensibilidade. A meus professores do mestrado, em especial a Giuseppe e Jacques, que me inspiraram e me despertaram o desejo de um dia conseguir dar aulas como eles. À equipe de funcionários da COPIN, especialmente a Aninha e ao pessoal da Miniblio, que sempre se mostraram disponíveis e acolhedores e finalmente aos meus colegas, que lutaram junto e participaram de muitos dos enriquecedores momentos que vivi ao longo desse trabalho.

Resumo

Este trabalho apresenta um sistema baseado em conhecimento para a integração de esquemas de banco de dados. O sistema incorpora os principais aspectos das metodologias mais significativas de integração de esquemas. É um sistema evolutivo e flexível, na medida em que possibilita a inclusão de novas regras à sua base de conhecimento, extraídas de novas metodologias propostas ou da experiência do usuário com integração de esquemas de banco de dados. A base de conhecimento do sistema utiliza regras de produção para representação do conhecimento sobre integração de esquemas além de quadros (*frames*) e representação lógica para representação estrutural dos fatos, constituindo um sistema híbrido que aproveita as vantagens dessas três formas de representação do conhecimento. Apresenta uma interface gráfica para manipulação dos esquemas, além de prover explanação das decisões tomadas e perguntas feitas pelo sistema, o que possibilita o treinamento de usuários menos experientes.

Abstract

This work presents a Knowledge based system to integrate database schemas. The system incorporates the main aspects of the most significant methodology to integrate the schemas. It's an evolutionary and flexible system once it's possible to include new rules to its knowledge-base extracted from new proposed methodologies or from the users experience with database integration. The system knowledge base uses production rules to represents its knowledge over the schema integration. It also uses frames and logical representation to model structural facts, constituting a hybrid system that uses the advantages of these three forms of knowledge representation. It uses a graphical interface to manipulate the schemas, offering explanation of the decisions taken and questions asked by the system. That offers training to the least experienced users.

Sumário

Lista de Figuras

CAPÍTULO 1

INTRODUÇÃO

A crescente tendência em considerar dados como um recurso autônomo nas organizações, independentemente das funções correntemente em uso, e a necessidade de capturar o significado dos dados como um todo de modo a gerenciá-los efetivamente tem levado as organizações a desenvolverem projetos de Bancos de Dados (BDs) para suas aplicações. Um BD é uma coleção de dados que representa conhecimento sobre um determinado domínio, associado a uma parte arbitrária do mundo real [WIEDERHOLD 83].

Ao contrário dos tradicionais sistemas de arquivos, os sistemas de bancos de dados permitem definir e manipular uma visão unificada de todos os dados relevantes para uma dada aplicação, evitando dessa forma problemas de duplicação, múltiplas atualizações e inconsistências entre aplicações. Alem disso, os Sistemas Gerenciadores de Bancos de Dados (SGBDs) procuram prover mecanismos de acesso múltiplo dos usuários aos dados, mecanismos de segurança e de controle de integridade dos dados. Entretanto, para que esses objetivos sejam atingidos, é necessário que seja desenvolvido um projeto de BD consistente e que atenda os requisitos da aplicação.

O projeto de um BD normalmente produz diversos esquemas que representam visões dos usuários do BD. A diversidade de visões durante a fase de projeto leva à necessidade de metodologias para integração dessas visões. Por outro lado, algumas organizações têm sentido a necessidade de integrar diversos esquemas de BDs já existentes a fim de obter um esquema global único que represente o conteúdo desses BDs

[BATINI 86]. Diante dessas premissas, podemos analisar o problema da integração de esquemas de Bancos de Dados dentro de dois contextos:

a) No projeto de um BD, quando diversos esquemas representam visões dos usuários e se busca obter um esquema conceitual global que represente o conteúdo das diversas visões.

b) Para Bancos de Dados já existentes, quando diversos BDs já existem e estão em uso em uma organização, onde se busca obter um esquema global único que poderá ser usado como interface para os vários BDs. Dentro do contexto de integração de esquemas de BDs já existentes, diversos trabalhos propõem o uso de um modelo de dados semântico como modelo intermediário para facilitar a integração de BDs, de forma que a atividade de integração possa ser considerada como uma atividade de projeto de BDs [BATINI 86].

Diversas metodologias têm surgido com o objetivo de estabelecer um conjunto de procedimentos para a integração de esquemas de BDs, tanto para integração de visões quanto para integração de BDs já existentes [BATINI 86] [NAVATHE 89]. Nosso trabalho se concentra na tarefa de integrar visões, dentro do contexto do projeto de um BD, podendo ser eventualmente utilizado como apoio na integração de esquemas de BDs já existentes, apenas no que diz respeito à integração dos esquemas de dados.

Desenvolvemos um sistema baseado em conhecimento para o auxílio na integração de esquemas, que leva em consideração os principais aspectos das metodologias mais representativas de integração de esquemas e cujo algoritmo básico de integração se baseia no conceito da equivalência de atributos [LARSON 89]. A abordagem seguida é aquela cada vez mais aceita e constantemente proposta na área de modelagem conceitual de BDs, que é o desenvolvimento de sistemas baseados em conhecimento e sistemas especialistas

para auxílio no projeto [BOUZEGHOUB 85] [CHOOBINEH 88] [DOGAC 89]. O sistema guia o projetista na tarefa de integração, apresentando alternativas para a integração dos esquemas. É um sistema evolutivo e flexível, na medida em que novo conhecimento poderá ser incluído, baseado em novas metodologias que venham a ser desenvolvidas. Além disso, provê um mecanismo de explanação, que não só permite ao usuário acompanhar os passos do processo de integração, como também possibilita o treinamento de usuários inexperientes, como é o caso de projetistas iniciantes.

No capítulo 2 apresentamos as razões que levam à necessidade de se integrar esquemas e discutimos os problemas relacionados com a integração. No capítulo 3 analisamos diversas metodologias para a integração fazendo também um estudo comparativo dessas metodologias. O capítulo 4 é dedicado à apresentação da Teoria da Equivalência de Atributos em BDs, um trabalho que serviu de base teórica e formal para a nossa metodologia. No capítulo 5 apresentamos a nossa abordagem para a integração descrevendo os passos necessários para a sua realização. Apresentamos também um tratamento detalhado para o problema da integração de hierarquias de generalização, de subconjuntos e de conjuntos relacionamento. Já no capítulo 6 apresentamos a arquitetura utilizada para o sistema que desenvolvemos e apresentamos as características das estruturas de dados e de controle. Um exemplo de integração e a forma como os dados são armazenados são apresentados no capítulo 7. E finalmente no capítulo 8 apresentamos nossas conclusões e indicamos possíveis extensões ao nosso trabalho.

CAPÍTULO 2

PROBLEMAS RELACIONADOS COM A INTEGRAÇÃO DE ESQUEMAS

Antes de definirmos uma metodologia para integração de esquemas é necessário um estudo dos problemas envolvidos no processo de integração, identificando as razões que levam à necessidade de integrar esquemas de BD, além do contexto onde se deve aplicar a integração e as tarefas envolvidas no processo.

2.1 - PROJETO DE UM BD

A necessidade de integração de esquemas no projeto de BDs deve-se a dois aspectos fundamentais:

a) A estrutura de um BD para grandes aplicações e muito complexa para ser modelada por um único projetista em uma única visão;

b) Grupos de usuários normalmente operam independentemente em organizações e tem suas próprias necessidades e expectativas dos dados, que podem conflitar com outros grupos de usuários. Isso levou a proposta de desenvolvimento do projeto de BDs através de diversas fases.

Podemos considerar o projeto de um Banco de Dados como um conjunto de 4 tarefas básicas:

a) Análise das necessidades de informação

b) Projeto conceitual

c) Projeto lógico

d) Projeto físico

A figura 1, a seguir, mostra as etapas do projeto de um BD [CERI 85] [NAVATHE 86]. Durante a análise das necessidades de informação, o projetista, a partir de entrevistas feitas com os usuários e da observação de como ele atua, coleciona para cada ambiente da organização, os dados e as atividades relevantes para as aplicações que se deseja modelar, formalizando-os em termos de descrições de dados, operações, eventos e restrições associadas.

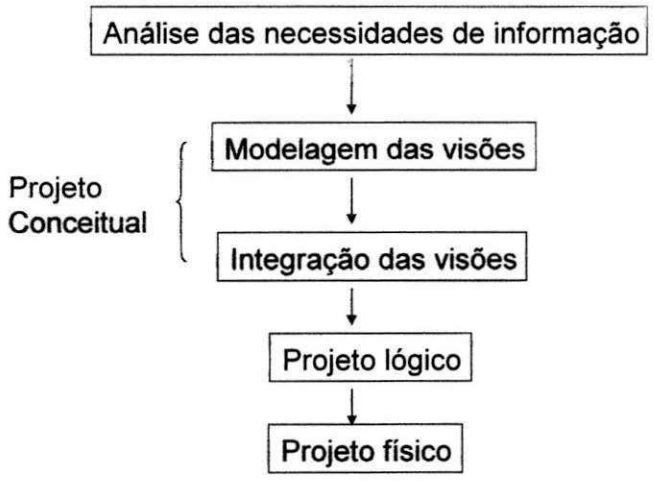

FIGURA 1: ETAPAS DO PROJETO DE UM BD

O projeto conceitual é dividido em duas fases: Modelagem das visões, onde as descrições das aplicações são expressas de maneira formal por meio de diversos esquemas conceituais orientados ao usuário, através do uso de um modelo de dados e Integração de esquemas, cujo objetivo é reunir tais esquemas em um esquema conceitual global único. O projeto conceitual tem como objetivo global obter uma especificação formal integrada, independente da implementação, de um conjunto de informações relativas a uma aplicação especifica de uma organização [BATINI 84].

O projeto lógico visa transformar um esquema conceitual em um esquema lógico de um SGBD. E finalmente no projeto físico o esquema lógico do BD é mapeado em uma

representação apropriada de armazenamento no SGBD, incluindo novos parâmetros físicos para otimização do desempenho do BD em relação a um conjunto de transações.

Acreditamos que o melhor momento para a integração de esquemas é durante o projeto conceitual, logo que se tenha um conjunto de esquemas conceituais que representem as visões dos usuários para as aplicações. Algumas metodologias entretanto consideram a possibilidade da integração de visões antes ou mesmo após o projeto conceitual.

A integração deve ser feita assim que se tenha uma representação completa e sem ambigüidades das visões dos usuários. Isso evita a proliferação de erros e inconsistências de dados durante o ciclo de vida do BD e das aplicações. Integrar visões antes do projeto conceitual pode ser negativo, pois as visões tendem a estar numa representação pouco formal, geralmente descrita como narrativas das necessidades de informação dos usuários. A integração durante o projeto lógico, apesar de mais fácil, normalmente carrega o inconveniente de não se ter uma representação com maior expressividade semântica, com riscos de obter resultados menos representativos das aplicações.

2.2 - Modelos de Dados

Um problema inerente à modelagem de qualquer subconjunto do mundo real é a diferença entre a percepção humana deste subconjunto, que denominaremos domínio de uma aplicação, e as necessidades computacionais para organizar as estruturas de uma maneira particular para armazenamento e recuperação eficientes [PECKHAM 88]. A fim de modelar uma aplicação, precisamos de um modelo de dados. Um modelo de dados tem por finalidade fornecer um meio formal de representar informações e um meio formal de manipular tal representação [DATE 91]. Para que um determinado modelo seja útil a uma

certa aplicação, devera existir, claramente, alguma correspondência simples entre os componentes do modelo e os elementos daquela aplicação.

As primeiras pesquisas em BD concentraram-se na estrutura física do Banco de Dados. Poucas considerações eram feitas sobre as percepções dos usuários sobre os dados. Os modelos Hierárquico e de Redes oferecem ao usuário uma forma de navegar no BD ao nível de registros [KORTH 89], assim provendo operações para derivar estruturas mais abstratas. O modelo relacional acrescentou um nível de estruturação dos dados, eliminando a necessidade de efetuar manipulações primitivas ao nível de registro do BD.

A capacidade de modelagem de Bancos de Dados com as abordagens anteriores esta ainda muito relacionada com a estrutura de registros do BD, o que dificulta a captação do significado mais amplo de uma aplicação.

Nos anos 70, pesquisadores tentaram simplificar o projeto e uso de BDs sugerindo estruturas de modelagem que eram capazes de suportar as visões dos usuários acerca dos dados de suas aplicações. Duas importantes idéias marcaram o surgimento dos chamados modelos de dados semânticos [PECKHAM 88]: A primeira idéia foi a da independência de dados. Em grande parte influenciados pelos desenvolvimentos em linguagens de programação, pesquisadores de BD sentiram que os usuários deveriam estar livres dos detalhes da estrutura física dos BDs. Dessa maneira, o usuário poderia modelar os dados em uma maneira similar a percepção humana da aplicação. A segunda idéia envolve capturar mais semântica no processo de modelagem dos dados. Os modelos de dados existentes até então, eram capazes de definir alguma semântica de dados, porém de baixo nível, como por exemplo dependências funcionais no modelo relacional.

Dentre os modelos semânticos, um dos que mais se destacaram e popularizaram foi o modelo Entidade-Relacionamento (ER) [CHEN 76], que posteriormente foi enriquecido

por diversas extensões, mas serviu como um referencial para a maioria dos modelos de dados denominados semânticos. O modelo ER original possui três tipos básicos de conceitos: Entidades, Relacionamentos e Atributos.

No modelo ER o mundo real é visto como um conjunto de entidades e um conjunto de relacionamentos entre essas entidades. Uma entidade representa um objeto que pode ser claramente identificado, como uma pessoa, um produto ou um evento. Um relacionamento representa uma associação entre entidades, como por exemplo "Uma pessoa comprou um produto" que estabelece uma associação entre pessoa e produto. Um atributo é uma propriedade elementar de uma entidade ou relacionamento que possui um conjunto de valores associados, denominado domínio. Denomina-se **Conjunto Entidade** o conjunto de todas as entidades que possuem os mesmos atributos. **Conjunto Relacionamento** é o conjunto de todos os relacionamentos envolvendo dois ou mais conjuntos entidade. Restrições de cardinalidade podem ser usadas para especificar o numero de instâncias que uma entidade pode participar em um relacionamento. A figura 2 mostra o exemplo de um diagrama ER incorporando as construções básicas do modelo.

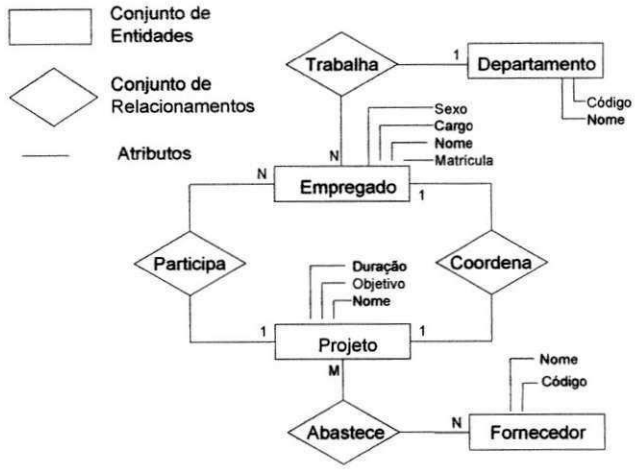

FIGURA 2: DIAGRAMA ENTIDADE-RELACIONAMENTO

Abstrações já empregadas por pesquisadores nas áreas de Inteligência Artificial e Psicologia foram utilizadas por [SMITH 77] a fim de incorporar mais semântica aos modelos de dados. A finalidade de uma abstração é permitir que o usuário concentre-se nos detalhes que ele considere mais relevantes da aplicação. Essas abstrações eram a Generalização e a Agregação, que estão hoje disponíveis em quase todos os modelos semânticos de dados [PECKHAM 88].

Agregação é uma forma de abstração em que um relacionamento entre objetos é considerado um objeto de nível mais elevado, com os detalhes de nível mais baixo suprimidos, enquanto que **Generalização** é o modo pelo qual, diferenças entre objetos similares são ignoradas para formar um tipo de maior ordem em que as similaridades podem ser enfatizadas [DATE 91]. Como característica da generalização, destacamos o conceito de herança hierárquica. Herança é a forma pela qual, atributos de um objeto mais geral são passados para um objeto mais especifico, ajudando a evitar redundância de dados. A figura 3 mostra um exemplo de generalização e agregação. Neste exemplo, Funcionários Efetivos e Funcionários Contratados herdam os atributos de Funcionários. Agregações, generalizações e herança hierárquica são de vital importância no escopo do nosso trabalho.

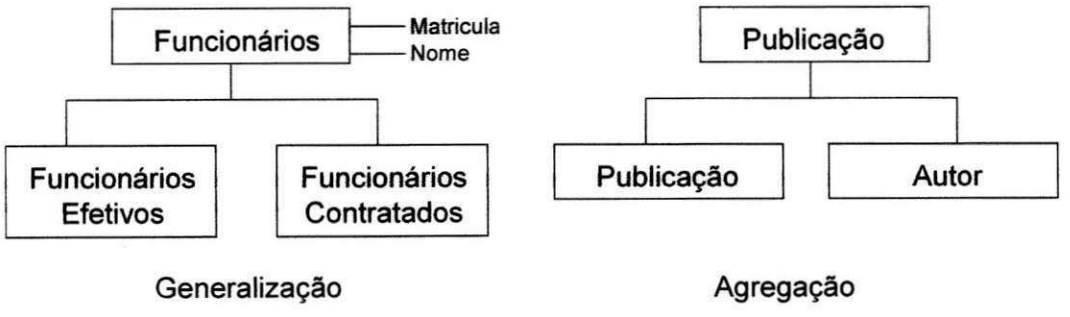

FIGURA 3: EXEMPLOS DE ABSTRAÇÕES

Os modelos semânticos além de facilitarem a manutenção da integridade do BD, já que integram certas restrições de integridade ao esquema conceitual, permitem uma grande flexibilidade na modelagem, uma vez que os usuários podem ver os dados em muitos níveis, através do uso de abstrações. Eles aumentam a eficiência na modelagem, permitindo que o projetista ignore os detalhes de implementação do BD. Em contrapartida, esses modelos possibilitam diversas formas de representação dos dados para uma mesma aplicação.

2.3 - ETAPAS DO PROCESSO DE INTEGRAÇÃO

Podemos considerar o processo de integração de esquemas de BDs como sendo composto de 4 atividades principais:

a) Pre-integração;

b) Comparação dos esquemas;

c) Assemelhação dos esquemas;

d) Fusão e reestruturação.

As metodologias que iremos avaliar e comparar neste trabalho podem ser consideradas como um conjunto de algumas ou todas essas atividades [BATINI 86]. A figura 4 apresenta dois esquemas ER que usaremos como ilustração na discussão das etapas do processo de integração. Os esquemas representam diferentes visões de usuários para uma mesma aplicação em uma dada organização.

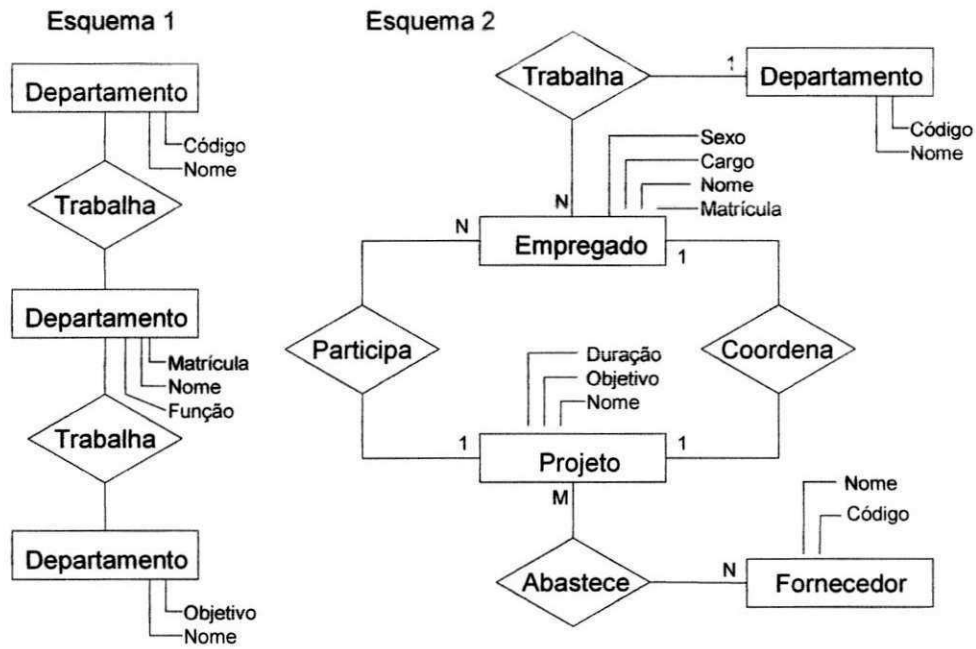

FIGURA 4: ESQUEMAS ENTIDADE-RELACIONAMENTO

2.3.1 - PRE-INTEGRAÇÃO

Durante a atividade de pre-integração, o projetista decide sobre qual estratégia de integração adotar, qual o nível de interação com o usuário e cataloga informações adicionais como asserções ou restrições entre visões e propriedades intra-esquema.

Apesar de algumas metodologias não citarem explicitamente qual a estratégia de integração adotada, a decisão de quantos esquemas serão integrados por vez e a ordem de integração devem ser inevitavelmente decididos pelo projetista. Existem dois tipos de estratégias para o processo de integração: As estratégias binárias que se dividem nas

categorias tipo escada e balanceada, e as estratégias n-árias que são do tipo "*one-shot*" e interativas. A figura 5 mostra os tipos de estratégias para o processo de integração.

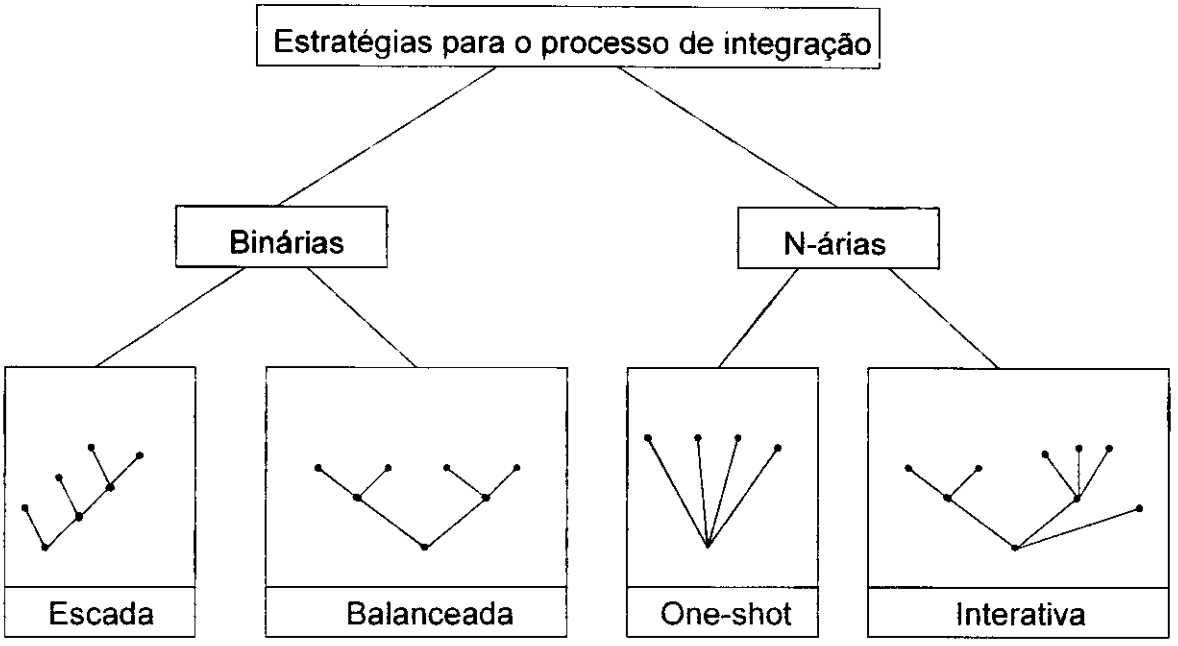

FIGURA 5: ESTRATÉGIAS PARA INTEGRAÇÃO DE ESQUEMAS

As estratégias binárias possuem a vantagem de simplificar o processo de comparação e assemelhação dos esquemas, diminuindo a complexidade do algoritmo de integração, além de permitirem que os esquemas a integrar, que denominaremos Esquemas Componentes, sejam organizados em uma ordem decrescente de importância, com o objetivo de obter resultados intermediários cada vez mais expressivos da aplicação modelada. Naturalmente o principal problema associado às estratégias binárias é o aumento da quantidade de passos necessários para a integração dos diversos esquemas e a necessidade eventual de análise do esquema global integrado a fim de detectar propriedades que possam ter sido perdidas durante o processo.

As estratégias n-árias têm como principal vantagem a redução da quantidade de passos para a integração, e a possibilidade de se eliminar analises posteriores através do aumento da análise semântica dos esquemas antes da fusão. A principal desvantagem,

como já dissemos, é o aumento da complexidade do algoritmo de integração, que é de $O(n^2)$ para a fusão de n esquemas [BATINI 86].

2.3.2 - COMPARAÇÃO DOS ESQUEMAS

Esta atividade visa detectar as similaridades e conflitos entre os dois esquemas. A maioria das metodologias para integração distingue dois tipos de conflitos entre esquemas de BD: Os conflitos de nome e os conflitos estruturais.

Como nomes devem ser atribuídos aos diversos objetos dos esquemas por diferentes pessoas de diversas áreas em uma organização, e existe um significado associado a cada nome, é inevitável que ocorram dois tipos de problemas na associação de nomes aos objetos. O primeiro problema diz respeito à sinonímia, quando nomes diferentes são associados a um mesmo objeto em um esquema ou em esquemas diferentes. No exemplo da figura 4, Funcionários no Esquema_1 representa o mesmo conjunto de entidades que Empregados no Esquema_2. Os conceitos são portanto sinônimos. Já o conjunto relacionamento *Trabalham* no Esquema_1, apesar de possuir um conjunto relacionamento de mesmo nome no Esquema_2, possui significado diferente do segundo, representando uma homonímia entre conceitos.

O problema da homonímia é de fácil solução uma vez que uma simples comparação de nomes é suficiente para a sua detecção. Entretanto sinonímia é um problema mais complexo e normalmente requer para a solução a obtenção de novas informações sobre os objetos. O uso de dicionários de dados pode ajudar no processo de gerenciamento de nomes e facilitar a atividade de integração.

Conflitos estruturais também podem ocorrer durante a modelagem. Um mesmo objeto pode ser modelado de diferentes maneiras em diferentes esquemas, ou também,

restrições de integridade incompatíveis em diversos esquemas podem ser associadas a um mesmo objeto. A existência de conflitos estruturais pode ser atribuída a três razões:

a) Projetistas podem possuir perspectivas diferentes acerca do problema, adotando diferentes pontos de vista na modelagem de um mesmo objeto.

b) O modelo de dados adotado permite modelar uma mesma aplicação através de diferentes tipos de construções, existindo equivalência entre diversas construções do modelo.

c) O projetista pode modelar de forma errada alguns dos conceitos da aplicação, escolhendo nomes, tipos e restrições de integridade incorretos.

Podemos identificar 4 tipos de conflitos estruturais entre esquemas de BDs, independente da metodologia utilizada para integração e do modelo de dados adotado [BATINI 86]:

a) Conflitos de tipo ocorrem quando diferentes construções de um modelo são utilizadas para modelar um mesmo objeto em diferentes esquemas. Por exemplo, podemos representar em um esquema Sexo como um atributo de domínio (Masc,Fem), e em outro esquema como uma entidade.

b) Conflitos de dependência ocorrem quando diferentes conceitos são relacionados entre si com diferentes restrições de cardinalidade em diferentes esquemas. Um relacionamento Casado entre Homem e Mulher pode ser representado como (1:1) em um esquema e (M:N) em outro esquema para representar um histórico de casamentos.

c) Conflitos de chave ocorrem quando chaves diferentes são associadas a um mesmo conceito em diferentes esquemas. Por exemplo, CPF e Matrícula podem ser a chave de Empregado em dois esquemas componentes.

d) Conflitos comportamentais ocorrem apenas quando o modelo de dados permite a representação de propriedades comportamentais dos objetos. Por exemplo, um departamento deve existir apenas se houver empregados associados a ele. No caso do ultimo empregado ser excluído, o departamento correspondente deve ser também excluído.

2.3.3 - ASSEMELHAÇÃO DOS ESQUEMAS

Durante o assemelhação dos esquemas os conflitos identificados devem ser resolvidos, a fim de tornar os esquemas compatíveis para integração. A resolução automática de conflitos nem sempre é possível. Deve haver uma forte interação com o projetista antes que decisões sejam tomadas.

Muitas vezes a resolução de conflitos requer que transformações sejam feitas nos esquemas. O conceito de transformação de esquemas é fundamental para a resolução de conflitos, e conseqüentemente para a atividade de assemelhação de esquemas. A simples troca de nomes pode ser usada para resolver conflitos de nomes entre conceitos de dois esquemas, entretanto a resolução de conflitos estruturais pode requerer a remodelagem de conceitos através de construções equivalentes.

A maioria das metodologias para integração de esquemas de BDs não faz uma análise ou prova de completeza das operações para transformação de esquemas com o objetivo de resolver os tipos de conflito que possam ocorrer. Entretanto, elas tratam a atividade de assemelhação de esquemas como um processo de obtenção de esquemas de consenso, através de transformações sobre os esquemas componentes.

2.3.4 - FUSÃO E REESTRUTURAÇÃO

A atividade de fusão dos esquemas normalmente compreende a superposição dos conceitos comuns e transporte destes e dos conceitos não integrados para o esquema

resultante. Já durante a reestruturação procura-se melhorar o esquema global obtido no que diz respeito a:

a) Completeza, no sentido de obter um esquema integrado que contenha todos os conceitos presentes em qualquer dos esquemas componentes, correspondendo o esquema integrado a visão dos domínios da aplicação associados com os esquemas componentes;
b) Minimalidade, significando que, se um conceito aparece em mais de um esquema componente, ele deve ser representado uma única vez no esquema integrado e
c) Expressividade, que significa obter um esquema integrado de fácil compreensão, procurando escolher entre as diversas alternativas de modelagem aquelas que mais claramente expressem o domínio da aplicação.

CAPÍTULO 3

ANÁLISE DE METODOLOGIAS PARA INTEGRAÇÃO DE ESQUEMAS

Para que possamos definir um conjunto de procedimentos a serem adotados no processo de integração, precisamos ainda fazer uma avaliação das principais metodologias para integração de esquemas desenvolvidas até o presente momento. Procuramos no nosso trabalho ter contato com o máximo de trabalhos possível [BATINI 86] e estudar de forma mais aprofundada algumas das metodologias que consideramos mais significativas e que, sob o nosso ponto de vista, tenham acrescentado maiores contribuições ao estudo do problema da integração. Na escolha desses trabalhos levamos em consideração principalmente o modelo de dados escolhido, o nível de abstração utilizado na modelagem dos conceitos, as etapas contempladas durante o processo de integração e o nível de formalismo apresentado.

3.1 - DESCRIÇÃO DE METODOLOGIAS PARA INTEGRAÇÃO DE ESQUEMAS

Procuraremos descrever de forma sucinta os seguintes trabalhos: [Al-FEDAGHI 81], [YAO 82], [CASANOVA 83], [BATINI 84], [DAYAL 84], e [NAVATHE 86]. A partir dessas descrições e analises, procuraremos desenvolver a nossa abordagem para o problema da integração, que terá também como base o trabalho intitulado A *Theory of Attribute Equivalence in Databases with Application to Schema Integration* [LARSON 89], além de diversos aspectos das metodologias estudadas.

3.1.1 - METODOLOGIA DE AL-FEDAGHI E SCHEUERMANN

Al-Fedaghi e Scheuermann [AL-FEDAGHI 81] desenvolveram uma metodologia para projeto de BD e integração de visões baseada no uso do Modelo Relacional. O

modelo relacional é constituído de um conjunto de relações definidas sobre certos atributos [MAIER 83]. Por exemplo, uma relação denominada Empregado pode ser definida sobre o conjunto de atributos {Matrícula, Nome, Cargo}. Representaremos esta relação na forma EMPREGADO(Matrícula, Nome,Cargo).

Denominamos a estrutura de uma relação como intenção e o conjunto de tuplas que a compõem, ou seja, seus valores num dado momento no tempo, como sua extensão. Podemos também definir alguns tipos de associação entre atributos. Uma dependência funcional, representada como **FD**:**X** → **Y**, vale em uma relação se e somente se, para todo instante, cada valor do conjunto de atributos **X** na relação esta associado a exatamente um valor do conjunto de atributos **Y**. Existem outros tipos de dependência, que contudo não são abordados por essa metodologia. Regras de inferência (axiomas) são também utilizadas para explorar as propriedades das dependências funcionais, estas regras são:

Reflexividade: **FD**:**X** → **X**,

Aumentação: Se **FD**:**X** → **T**, então **FD**:**X**,**Z** → **T** e

Pseudo-transitividade: Se **FD**:**X** → **Y** e **FD**:**Y**,**Z** → **W**, então **FD**:**X**,**Z** → **W**.

Essa metodologia utiliza dependências funcionais para o projeto e integração de esquemas de BD. O objetivo é obter um esquema conceitual integrado a partir de um conjunto de visões dos usuários, dadas em termos de dependências funcionais, e um conjunto de mapeamentos entre esquemas externos e o esquema conceitual integrado.

A metodologia considera a integração de visões como uma das etapas do projeto de um BD, e julga esta uma das atividades de maior complexidade no projeto. Não requer entretanto uma forte interação com o projetista na tentativa de verificar e eliminar inconsistências, uma vez que considera o conjunto de FDs e axiomas de inferência suficiente para o processo de integração. Conflitos de nomes não são tratados por essa

metodologia, certamente como conseqüência da suposição que as relações que constituem as visões são projeções de uma relação universal, e assim, os nomes são considerados únicos para quaisquer das relações a integrar. A metodologia procura obter mapeamentos que preservem a compatibilidade entre as visões e o esquema global integrado. Claramente, utiliza uma estratégia do tipo n-ária "one-shot" para integração das visões.

O algoritmo External-Conceptual (EC) é apresentado para o projeto de um BD compreendendo a integração de diversas visões. O algoritmo recebe as visões, como já dissemos, em termos de um conjunto de dependências funcionais, produzindo um esquema conceitual integrado. Utilizando a abordagem de síntese, constrói diversos conjuntos de relações em terceira forma normal, que representam os esquemas externos dos usuários. O esquema conceitual integrado é a união desses esquemas externos, eliminadas as redundâncias que tenham ocorrido.

3.1.2 - METODOLOGIA DE CASANOVA E VIDAL

Essa metodologia também se baseia no uso do modelo relacional para o projeto de BD e integração de visões [CASANOVA 83] . Entretanto, outros tipos de dependências inter-relacionais são utilizados tentando incorporar mais semântica ao modelo e facilitar o processo de integração, evitando que este seja baseado apenas na sobreposição de estruturas ou similaridades de nomes. Os autores sugerem o uso de três novas classes de dependências, além das dependências funcionais: dependência de inclusão, dependência de exclusão e dependência de união funcional. Além disso, fazem referência a construções do tipo conjunto entidade/relacionamento, significando esquemas de relações que representem construções semânticas semelhantes ao modelo ER.

Dependências de inclusão (INDs) podem ser utilizadas para expressar que dois conjuntos entidade/relacionamento de mesmo tipo, porém de diferentes visões são iguais

ou um é subconjunto do outro, sendo por isso muito úteis no processo de integração de visões. Por exemplo, se em uma visão **V1** temos o esquema de relações **S1** = *{Empregado(Matricula,Dept)}* e numa visão **V2** temos **S2** = *{Secretaria(Matrícula,Velocidade)}*, podemos escrever *Secretaria[Matrícula]* ⊆ *Empregado[Matrícula]* para indicar que Secretaria na visão **V1** é subconjunto de Empregado na visão **V2**.

Dependências de exclusão (EXDs) podem representar que dois conjuntos entidade/relacionamento de mesmo tipo, porém de visões diferentes, são disjuntos. Por exemplo, se em uma visão **V1** temos *Pós-graduado(Matrícula,Departamento)* e em **V2** temos *Graduando (Matrícula,Departamento)*, podemos escrever *Pós-graduando[Matrícula]* | *Graduando[Matrícula]* para indicar que os conjuntos de estudantes nas diferentes visões são disjuntos.

Dependências de união funcional (UFDs) servem para representar sinonímia entre atributos de diferentes conjuntos entidade/relacionamento de visões diferentes ou de uma mesma visão. Por exemplo, consideremos o esquema de relações **S** = *{Estudante(Matrícula, Departamento), Professor-assistente(Matricula,Dept)}*. Se Departamento no primeiro esquema significa o mesmo que Dept no segundo, Matrícula pode ser uma chave comum aos dois, e podemos expressar a sinonímia entre Departamento e Dept na forma *[Estudante: Matrícula → Departamento, Professor-assistente: Matrícula → Dept]*.

A entrada para o algoritmo de integração é um conjunto de visões dos usuários, dadas em termos de esquemas de relações, e o resultado é um esquema conceitual integrado. Os autores assumem que todas as visões já passaram por um processo de pré-integração para detectar quando construções em diferentes visões são do mesmo tipo.

Nessa metodologia adota-se uma estratégia de integração binária do tipo balanceada. As visões são organizadas numa ordem hierárquica de importância, agrupando-se inicialmente visões que mais estejam relacionadas entre si. O processo de integração é dividido em duas etapas: combinação e otimização.

Durante a combinação das visões, que são informadas em termos de relações, INDs, EXDs e FDs, o projetista define novas dependências de inclusão e exclusão para indicar associações interesquemas entre conjuntos entidade/relacionamento e novas dependências de união funcional para indicar sinônimos entre atributos. A seguir, as visões são superpostas formando um esquema composto dos conteúdos das duas visões e das novas dependências informadas. O esquema resultante do processo de combinação constitui o que os autores denominam esquema restrito. Num esquema restrito, as relações estão na forma normal de Boyce-Codd, dependências de inclusão são assinaladas exclusivamente pelas chaves das relações, dependências de exclusão denotam conjuntos disjuntos e a noção de sinônimos é transitiva, devendo-se assim evitar redundâncias na indicação de sinônimos.

Durante a otimização, tenta-se minimizar redundâncias no esquema restrito, resultante do processo de combinação, independente dos esquemas componentes possuírem ou não redundâncias. Procura-se ainda reduzir o tamanho do esquema de relações e do conjunto de dependências, tentando-se fundir esquemas de relações quando apropriado.

Quatro objetivos principais devem ser alcançados pela otimização:

a) Não devem existir INDs que indiquem o armazenamento redundante de uma relação;

b) Não devem existir INDs que representem construções diferentes para um mesmo conjunto de objetos;

c) Deve-se reduzir ao máximo o conjunto de UFDs, já que ele representa potenciais redundâncias e

d) Devem-se excluir relações que possam ser potencial mente vazias, isso pode ser verificado através das EXDs.

Essas transformações devem ser feitas de maneira que se preserve a equivalência entre o esquema restrito inicial **R** e o resultado da otimização **R'**, de forma que qualquer estado consistente de **R** possa ser mapeado em um estado consistente de **R'**.

3.1.3 - METODOLOGIA DE YAO, WADDLE E HOUSEL

Essa metodologia também se aplica à integração de visões no projeto de BD [YAO 82]. Todavia os autores optaram pelo uso do Modelo Funcional, que é semanticamente mais rico que o modelo relacional, oferecendo maiores facilidades na modelagem das aplicações. O modelo funcional utiliza como construções básicas Entidades e Funções. Entidades, ou nodos, servem para representar objetos do mundo real, enquanto funções representam propriedades desses objetos ou relacionamentos entre eles. Entidades possuem conjuntos de atributos que as distinguem, porém funções não possuem atributos associados. Uma construção importante para o processo de integração e que é utilizada nesse trabalho é o nodo tupla. Um nodo tupla é usado para representar relacionamentos muitos para muitos (não funcionais) entre nodos. Isto pode ser útil para representar um relacionamento do tipo subconjunto entre nodos.

Aspectos dinâmicos da modelagem são considerados, sendo representados através de uma linguagem de especificação de processamento denominada *Transactions Specification Language* (TASL). As visões estão representadas nesta linguagem, que serve

tanto para especificar os esquemas no modelo funcional quanto para modelar caminhos de acesso, volumes de consultas ao BD e transações de manipulação de dados.

Conflitos de nome não são claramente abordados, porém conflitos estruturais são resolvidos durante o processo de fusão dos esquemas.

Segundo os autores, o principal problema existente no processo de integração de visões é determinar e remover funções (relacionamentos entre nodos) redundantes. Dentro desse ponto de vista, eles identificam duas abordagens para integração: A abordagem automática, que assume a existência de no máximo uma função entre quaisquer dois itens de dados, implicando que se existe mais de uma função entre dois nodos, uma deve ser redundante, e a abordagem manual, que permite múltiplas funções entre itens de dados, requerendo a determinação das funções redundantes por parte do projetista manualmente.

Naturalmente a abordagem automática é pouco realista, além de restringir a capacidade de modelagem das visões não permitindo indicar mais que um tipo de associação entre nodos, enquanto que a abordagem manual, ao deixar todas as decisões a cargo do projetista, tende a tornar a atividade de integração cansativa e ineficaz. O ideal seria obter um método de integração que pudesse capturar características positivas das duas abordagens anteriores.

Os autores desenvolveram heurísticas, para filtrar muitas funções que não são redundantes do ponto de vista do projetista, com o propósito de evitar comparações desnecessárias durante o processo de integração. Assim, o projetista irá se concentrar apenas nas funções que tenham uma alta probabilidade de serem realmente redundantes. Conseqüentemente evita-se a abordagem automática, sem que haja necessidade de testes exaustivos ou projeto completamente manual.

As heurísticas propostas pelos autores para ajudar na detecção de funções potencialmente redundantes utilizam tanto informações relativas a tamanho dos nodos e funções (cardinalidade) no projeto do BD, quanto percentuais de participação do nodo em uma função. Elas se dividem em quatro tipos:

a) As que utilizam informações sobre como os domínios das funções se interceptam;

b) As que usam informações sobre a cardinalidade máxima das funções;

c) As que utilizam informações sobre as cardinalidades medias das funções e

d) As que avaliam os tipos de funções envolvidas.

O primeiro tipo de heurística visa detectar potenciais propriedades de subconjuntos entre visões, enquanto que os três últimos procuram detectar funções redundantes entre nodos.

No inicio do processo de integração, são especificados mapeamentos entre nodos de diferentes visões que compartilhem domínios comuns, obtendo-se um grafo integrado onde estão representados estes mapeamentos. O algoritmo de integração então segue os seguintes passos:

a) Faz a fusão dos nodos com o mesmo valor (cardinalidade), desde que não existam funções determinando que os nodos não são idênticos (não-identidade).

b) Faz a fusão dos nodos subconjunto de outros nodos, desde que não existam funções não-identidade.

c) Remove funções redundantes, verificando ligações entre todos os nodos que não foram fundidos, sendo removidas funções após confirmação do projetista.

3.1.4 - METODOLOGIA DE DAYAL E HWANG

A metodologia de Dayal e Hwang [DAYAL 84] aplica-se a integração de BDs. Ela aborda muitos dos problemas envolvidos na integração de visões e é de grande importância no contexto do nosso trabalho, apresentando heurísticas para a integração dos esquemas de dados.

A tarefa de integrar BDs é normalmente complexa, pois estes podem estar expressos em diferentes modelos de dados e implementados em diferentes SGBDs, cada qual com suas próprias linguagens de consulta. Além disso, a tarefa compreende não só a integração dos esquemas de dados, a intenção do BD, como também a integração dos dados correntemente armazenados, a extensão do BD. Nessa metodologia os dois aspectos são devidamente tratados, além de se analisar o problema do mapeamento das consultas dos BDs a integrar para o BD integrado.

O esquema resultante da integração é expresso no modelo funcional, estendido com o uso de hierarquias de generalização, e é usada uma linguagem para mapeamento das consultas baseada no modelo relacional, também estendido, denominada DAPLEX. A abordagem seguida pelos autores pressupõe o mapeamento dos BDs locais em esquemas auxiliares expressos no modelo funcional estendido e das consultas em termos da linguagem DAPLEX. Os esquemas são então integrados e as consultas compatibilizadas com o novo BD resultante. A integração de dados considera tanto redundâncias de dados, quanto dados incompatíveis.

Essa metodologia separa a atividade de integração de BDs em 3 fases: a) Integração de esquemas; b) Integração de dados e c) Modificação de consultas.

A integração de esquemas compreende a resolução de conflitos e compatibilização dos diversos esquemas de dados dos BDs, ora mapeados em termos do modelo funcional.

São analisados e resolvidos conflitos de nomes, diferenças de escalas, diferenças estruturais e diferenças em abstrações.

Conflitos de nomes, como homonímia e sinonímia, são resolvidos pela simples troca de nomes por parte do projetista.

Diferenças de escala, como a representação de um atributo em centímetros em um BD e polegadas em outro, podem ser resolvidas através da unificação das escalas e incorporação de uma formula de conversão como informação adicional, ou uma tabela de conversão para casos onde não exista uma formula simples associável ao atributo. Apesar dos autores apresentarem estas soluções, consideradas razoáveis e eficientes, não são discutidos procedimentos claros para a efetivação da solução proposta, como criação de catálogos ou incorporação de disparadores automáticos para atributos com formulas associadas.

Diferenças estruturais ocorrem quando um mesmo objeto é modelado como uma entidade em um esquema e como uma função em um outro esquema, ou quando existe diferença em agregações de um mesmo objeto em diferentes esquemas. O problema da diferença de agregações é resolvido através do uso de generalização, colocando-se apenas as funções comuns a todos os subtipos em um supertipo. O problema da modelagem incompatível é resolvido através da definição de entidades e funções virtuais e então de uma generalização.

Diferenças em abstrações ocorrem quando tipos entidade e funções foram definidos em diferentes níveis de generalização. A maneira de integrar estes esquemas é também via generalização.

A integração de dados tem por objetivo compatibilizar a extensão do BD integrado com as extensões dos BDs locais. Naturalmente as extensões dos BDs locais podem ser discrepantes nos valores de algumas funções. Duas situações são abordadas pelos autores:

a) Os BDs locais são mutuamente inconsistentes, porém corretos. Isso ocorre quando, por exemplo, a um empregado **E1** no BD1 está associado um salário *S1* e para este mesmo empregado, **E1** no BD2, esta associado um outro valor de salário *S2*. Discrepâncias desse tipo podem ocorrer por três razões:

    1) As entidades dos dois BDs, apesar de momentaneamente possuírem o mesmo valor de chave, não representam a mesma entidade no mundo real. Isso pode ocorrer quando chaves não são escolhidas de forma adequada para as entidades, e pode ser resolvido através da redefinição das chaves das entidades;

    2) As chaves são adequadamente escolhidas, porém as funções são diferentes, por exemplo, salário do BD1 pode significar salário de um cargo exercido por **E1**, digamos professor com dedicação parcial, e salário do BD2 corresponder ao salário de função exercida como técnico numa mesma instituição. Neste caso os atributos são considerados homônimos e uma simples troca de nomes resolve o problema;

    3) A terceira razão para discrepâncias pode ser a obsolescência. O valor do salário em um dos BDs não esta devidamente atualizado. Neste caso, podemos utilizar no BD integrado o valor mais atual de salário, se uma data esta associada ao atributo, ou o valor mais confiável.

b) Os BDs locais são mutuamente inconsistentes e incorretos. Temos aqui também diversas alternativas. Uma é usar o valor mais confiável dos dados, se

isso pode ser definido *a priori*. Uma outra possibilidade é deixar a alternativa de escolha a critério do usuário, notificando-o sempre que ocorrerem inconsistências.

O processo de modificação de consultas tem por objetivo mapear as consultas em termos das visões locais em consultas semanticamente equivalentes contra o esquema global integrado e conseqüentemente o BD integrado. Um algoritmo de modificação de consultas é apresentado pelos autores, além de uma discussão sobre o problema do controle na eliminação de duplicidades.

3.1.5 - METODOLOGIA DE BATINI E LENZERINI

A metodologia de Batini e Lenzerini [BATINI 84] aplica-se a integração de visões no projeto de BDs. O modelo de dados utilizado é o ER acrescido de algumas extensões que visam incorporar mais semântica ao modelo original com o intuito de facilitar tanto o trabalho do projetista de BDs quanto o processo de integração de esquemas. O conceito de restrições de cardinalidade foi ampliado permitindo-se especificar cardinalidade máxima e cardinalidade mínima para indicar o numero máximo e mínimo de vezes que cada ocorrência de uma entidade pode ser envolvida em uma ocorrência de um relacionamento. Também são aceitos atributos multivalorados, com cardinalidade máxima maior que 1, tentando-se simplificar os esquemas reduzindo-se o numero de entidades. Além disso os autores fazem uso das abstrações hierarquia de subconjunto e generalizações. Uma entidade **E1** é subconjunto de outra entidade **E2** se toda ocorrência de **E1** é também uma ocorrência de **E2**. A figura 6 a seguir mostra exemplos de esquemas representando hierarquias de subconjunto e de generalização.

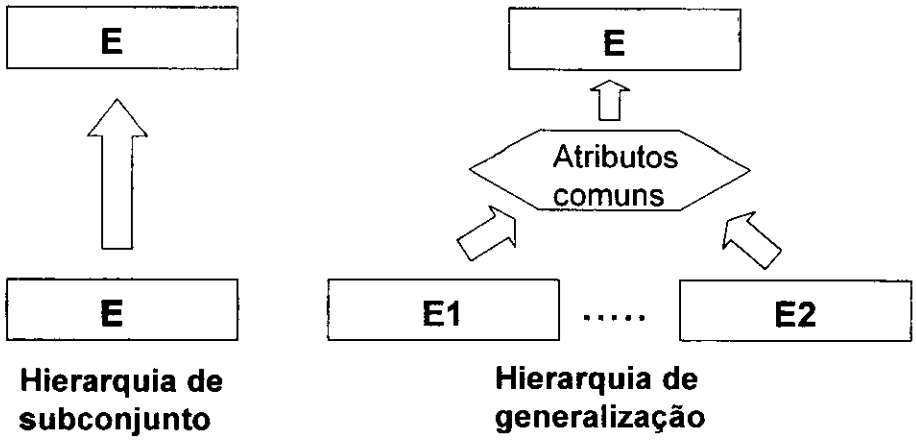

FIGURA 6: HIERARQUIAS DE SUBCONJUNTO E GENERALIZAÇÃO

A estratégia de integração é a binária do tipo escada. Dois esquemas são integrados de cada vez, sendo atribuídos pesos aos esquemas dos usuários e esquemas parciais integrados, na tentativa de associar a eles uma importância relativa no processo de integração, que determine o grau de confiabilidade de um esquema em relação a outro ou a um resultado parcial. Isso, segundo os autores, faz com que os esquemas parciais integrados sejam privilegiados em relação a esquemas dos usuários, obtendo-se uma melhor convergência e estabilidade em direção ao esquema global integrado.

O processo de integração divide-se em 3 fases:

A) Análise de conflitos;

B) Fusão dos esquemas e

C) Enriquecimento e reestruturação do esquema integrado.

A fase de análise de conflitos tem por objetivo detectar e resolver incoerências que existam na modelagem de uma mesma classe de objetos em diferentes esquemas. O processo de análise é guiado através de indicações dos conflitos detectados ao projetista, no caso de uma ferramenta automatizada. Possíveis cenários para a resolução desses conflitos são apresentados, cabe ao projetista escolher a melhor alternativa dentre as

propostas para resolução do conflito. As duas principais tarefas durante esta fase são a análise de conflitos de nomes e a análise de compatibilidades de modelagem.

Na análise de conflitos de nomes, as sinonímias e homonímias são detectadas e resolvidas. Esta tarefa é completamente guiada por indicações e o projetista é chamado a analisar cada indicação e resolver o conflito se ele realmente ocorrer. Esse trabalho pode levar normalmente a detecção de propriedades interesquemas, que nesta fase são apenas registrados pelo projetista para posterior análise. Os resultados desta etapa são dois esquemas melhorados, devido à eliminação de sinônimos e homônimos, e uma lista de conflitos e propriedades entre os esquemas. Para evitar uma explosão combinatorial de comparações durante esta tarefa, a metodologia apresenta algumas heurísticas de comparação. Para conceitos de mesmo tipo, são analisados inicialmente os conjuntos de valores para atributos, atributos e relacionamentos para entidades, e atributos e entidades envolvidas para relacionamentos. Para conceitos de tipos diferentes, são analisados os conjuntos de valores dos atributos e o conjunto de valores dos identificadores para o caso de entidade e atributo, e os atributos e o conjunto de valores dos identificadores para o caso de entidade e relacionamento.

Na análise de compatibilidades da modelagem, as representações escolhidas nos dois esquemas para uma mesma classe de objetos do mundo real são comparadas para verificar se elas podem ou não ser fundidas. O primeiro passo é realizar reestruturações sintáticas nos esquemas a fim de associar o mesmo tipo aos conceitos que representem a mesma classe de objetos e tem tipos diferentes nos dois esquemas. A metodologia trata apenas transformações envolvendo conceitos atômicos: entidades, relacionamentos e atributos, apresentando os passos e as condições necessárias para a transformação de um conceito em outro. O segundo passo na análise de compatibilidades da modelagem é

determinar se as diferentes representações escolhidas nos dois esquemas para um mesmo objeto são ou não compatíveis e escolher de uma representação comum.

Conceitos são considerados idênticos quando eles têm exatamente as mesmas características de modelagem associadas. São compatíveis quando as restrições de integridade não são contraditórias. Caso as condições anteriores não sejam satisfeitas, os conceitos são considerados incompatíveis. A existência de diferentes cardinalidades associadas a uma mesma entidade ou o fato de um identificador em um esquema não ser também identificador no outro esquema são exemplos de incompatibilidades. Durante a fase posterior de fusão de esquemas os conceitos idênticos e compatíveis serão superpostos.

Na fase de fusão, os esquemas componentes são ligados sintaticamente obtendo-se um esquema integrado parcial composto dos conceitos pertencentes a um ou outro esquema e conceitos comuns.

A ultima fase no processo de integração nessa metodologia é a de enriquecimento e reestruturação dos esquemas. O objetivo nesta fase é obter um esquema global integrado mais confiável, auto-explicativo e o mais claro e simples possível. Para atingir estes objetivos, os autores dividem esta fase em três tarefas: A) Análise das propriedades interesquemas; B) análise de redundâncias e C) reestruturação.

A análise das propriedades interesquemas compreende a avaliação de propriedades detectadas entre conceitos nos passos anteriores, como novas dependências funcionais que possam existir novas entidades resultantes da fusão de duas outras.

Análise de redundâncias procura eliminar ciclos que ocorrem em decorrência da fusão dos esquemas, através da eliminação de relacionamentos redundantes. A eliminação

de relacionamentos leva a necessidade de reestruturações do esquema e deve ser feita sem perda de informação.

Na tarefa de reestruturação de esquemas, o esquema obtido é analisado detalhadamente na tentativa de representar com o uso do próprio modelo, todas as restrições de integridade que estejam expressas em linguagem natural, procurando tornar o esquema auto-explicativo. Procura-se também aumentar a simplicidade e clareza na representação da aplicação.

3.1.6 - METODOLOGIA DE NAVATHE, ELMASRI E LARSON

Em [NAVATHE 86] os autores descrevem uma medotologia para integração de esquemas no projeto de BDs baseada no uso de um modelo de dados semântico denominado Entidade-Categoria-Relacionamento (ECR). O modelo de dados ECR [ELMASRI 85] é uma extensão ao modelo ER que incorpora o conceito de categoria para representar hierarquias de generalização e subconjuntos e amplia o uso de restrições de cardinalidade permitindo representar cardinalidade mínima e cardinalidade máxima da participação de objetos em um relacionamento, como será melhor discutido na seção 5.1. Essa metodologia aplica-se a integração de visões no projeto de BDs já existentes. O processo de integração consiste de duas tarefas que são executadas repetidamente: a integração de classes de objetos (entidades e categorias) e a integração de relacionamentos entre objetos. O processo de integração nessa metodologia divide-se em três fases:

A) A pre-integração;

B) A integração de classes de objetos e

C) A integração de relacionamentos.

Durante a pre-integração, são estabelecidas diversas asserções e correspondências entre as visões a serem integradas, como informações sobre sinônimos, chaves candidatas

e domínios das classes de objetos. Uma chave candidata corresponde a um conjunto de atributos cujos valores podem potencialmente servir para identificar cada entidade. O domínio corresponde a todas as instâncias da classe de objetos em algum momento no tempo.

Os domínios das classes de objetos nessa metodologia são de grande importância no processo de integração. A metodologia tenta estabelecer um conjunto de relacionamentos possíveis entre diferentes classes de objetos. Esses relacionamentos são equivalentes a relacionamentos entre conjuntos. Dois domínios de duas classes de objetos diferentes podem ser idênticos, contidos um no outro, sobrepostos ou disjuntos. Sejam duas classes de objetos **A** e **B**:

Domínios idênticos ==> Dom (**A**) = Dom (**B**)

Domínios contidos ==> Dom (**A**) $\subseteq$ Dom (**B**) ou

Dom (**B**) $\subseteq$ Dom (**A**)

Domínios sobrepostos ==> Não existem restrições sobre os domínios, havendo uma interseção entre eles.

Domínios disjuntos ==> Dom (**A**) $\cap$ Dom (**B**) = $\varnothing$

As asserções sobre domínio são especificadas quando valem sobre os domínios das classes de objetos em todos os momentos no tempo. Ficando sob responsabilidade do projetista a especificação correta das correspondências entre as diferentes classes de objetos.

Também durante a fase de pre-integração devem ser informados os mapeamentos existentes entre atributos equivalentes das classes de objetos. Se duas classes de objetos possuem atributos similares, porém os atributos estão expressos em diferentes unidades, então um mapeamento deve ser especificado entre os dois atributos. Por exemplo, se o

atributo distancia de uma classe de objetos esta expressa em metros e um atributo equivalente em outra classe de objetos esta expresso em milhas, então deve ser especificada uma função de mapeamento entre os dois atributos.

A partir das asserções de domínios entre as classes de objetos, as classes de objetos que representam conceitos similares são integradas.

Nessa metodologia é utilizada uma estratégia de integração n-ária do tipo "*one shot*". Contudo, os autores descrevem mais claramente os passos para a integração através do uso de uma estratégia binária.

Os autores assumem que atributos equivalentes em diferentes visões devem ser identificados e possuir o mesmo nome. Dessa forma, o problema da homonímia deve ser resolvido pelo projetista antes mesmo de iniciar o processo de integração.

Após a fase de pre-integração, as classes de objetos similares são integradas de acordo com as asserções informadas pelo projetista. Quando dois domínios são informados como idênticos, a metodologia assume que as classes de objetos nas duas visões são as mesmas em todos os momentos no tempo. Nesse caso, uma nova classe de objetos é criada no esquema integrado tendo como atributos a união dos atributos das classes de objeto integradas.

Quando os domínios de duas entidades **A** e **B** de diferentes esquemas são informados como contidos, o esquema integrado conterá duas novas classes de objetos **A1** e **B1** onde **A1** é subconjunto de **B1**, ou usando a terminologia do modelo ECR, **A1** é uma categoria de **B1**. Os atributos de **B1** serão os mesmos atributos de **B**, enquanto que os atributos de **A1** serão os atributos de **A** que não possuem equivalente em **B**. Todas as entidades individuais na classe **B** existirão na classe **B1** no esquema integrado e todas as entidades na classe **A** irão existir na categoria **A1**.

No caso de domínios sobrepostos serão criados no esquema integrado três classes de objetos. Uma classe, que podemos denominar **AB**, incluirá a união das entidades da classe **A** e da classe **B**. Os atributos da classe **AB** serão a interseção dos atributos da classe **A** com os da classe **B**. São criados dois subconjuntos da classe **AB**. Uma classe **A1** cujas entidades serão as mesmas da classe **A** e os atributos serão aqueles da classe **A** que não aparecem em **AB**, e uma classe **B1** cujas entidades são as mesmas de **B**, e os atributos são os atributos da classe **B** que não aparecem em **AB**.

Para domínios disjuntos, duas situações são abordadas pelos autores. No primeiro caso, mesmo tendo o projetista especificado as classes de objetos como similares, eles nunca possuem entidades em comum. A decisão de integrar ou não as classes de objeto fica a cargo do projetista. Um critério de decisão é se as classes de objeto possuem qualquer atributo em comum, porém, mesmo com atributos em comum, o projetista pode ainda decidir não integrar as classes de objetos. Um exemplo pode ser: Dadas as duas classes de objeto Secretaria e Engenheiro abaixo, estas podem ser integrados como Empregados mesmo tendo os seus domínios sempre disjuntos:

Secretaria(matrícula, salário, nome, grau)

Engenheiro(matrícula, salário, nome, tipo_engenheiro),

Quando dois objetos compartilham alguns atributos, e o projetista decide integrá-los, eles podem ser integrados da mesma forma que domínios sobrepostos. A única diferença é que as classes **A1** e **B1** são disjuntas no esquema integrado.

Na fase de integração de relacionamentos, são analisados para integração os relacionamentos que estavam associados a classes de objetos integradas durante a fase anterior. São consideradas nesta fase informações semânticas e estruturais, essas informações são providas pelo projetista e são: os papeis das classes de objetos nos

relacionamentos, o grau dos relacionamentos e suas restrições de cardinalidade. São integrados dois conjuntos relacionamento de cada vez. A integração de mais de dois relacionamentos ao mesmo tempo é considerada pelos autores como uma extensão ao seu trabalho.

Para o processo de integração, os conjuntos relacionamento são classificados com relação a três características:

A) O grau do relacionamento, que corresponde ao numero de classes de objetos, não necessariamente distintos, participantes em um conjunto relacionamento;

B) O papel de uma classe de objetos em um relacionamento, que deve ser informada pelo projetista antes da integração, durante a fase de pre-integração;

C) Restrições estruturais, que associam instâncias de uma classe de objetos com instâncias de outra. Nessa metodologia, os autores utilizam restrições de cardinalidade no processo de integração.

Diversos problemas são discutidos pelos autores como ainda passíveis de resolução, dentre eles podemos destacar a necessidade de definir operações que capacitem o projetista a organizar visões apropriadamente para comparação e integração. Estudar a ordem em que assemelhação de esquemas, integração e modificação podem ser aplicadas. Estudar as vantagens e desvantagens de estratégias binárias de integração *versus* estratégias n-árias. Desenvolver técnicas para mapear operações da visão original para a visão integrada e vice-versa.

3.2 - COMPARAÇÃO ENTRE AS METODOLOGIAS PARA INTEGRAÇÃO

Faremos uma análise comparativa das metodologias citadas anteriormente, procurando identificar suas características mais importantes. Consideraremos os seguintes aspectos:

A) Aplicabilidade da metodologia;

B) Modelo de dados utilizado;

C) Etapas consideradas durante o processo de integração;

D) Estratégia de integração utilizada e

E) Tipos de conflitos considerados.

3.2.1 - APLICABILIDADE DA METODOLOGIA

Dentre as metodologias apresentadas, apenas a de [DAYAL 84] se aplica à integração de BDs. Apesar disso, ela aborda muitos dos problemas envolvidos na integração de visões. A metodologia de [NAVATHE 86] pode ser usada tanto para a integração de visões quanto para a integração de BDs, enquanto que as demais se destinam apenas à integração de visões.

Entre as metodologias para integração de visões, as de [CASANOVA 83] e [YAO 82] são aplicáveis durante a fase do projeto lógico, naturalmente como conseqüência do modelo de dados escolhido, enquanto que as demais são aplicáveis durante o projeto conceitual, que consideramos o melhor momento durante o projeto de um BD para a integração dos esquemas.

3.2.2 - MODELO DE DADOS UTILIZADO

Durante a escolha das metodologias a estudar, procuramos identificar aquelas que seriam as mais representativas em termos do modelo de dados utilizado. Metodologias baseadas no uso do modelo relacional como [CASANOVA 83] e [AL-FEDAGHI 81] assumem o uso do conceito de relação universal, fazendo com que cada atributo tenha um nome único.

Como conseqüência do uso do modelo relacional, o processo de integração fica mais facilitado devido a três principais fatores:

A) As possibilidades de conflitos de tipo são menores;

B) As operações de transformações são mais simples e

C) A fusão envolve menos operações primitivas. Contudo, o uso desse modelo restringe a capacidade de modelagem das aplicações por parte do projetista. Consciente dessa limitação, [CASANOVA 83] utiliza os conceitos de inclusão, exclusão e união funcional para incorporar mais semântica ao modelo utilizado no projeto.

As demais metodologias fazem uso de modelos semânticos para a representação dos esquemas, tendo como base fundamental extensões ao modelo ER [CHEN 76]. Essas metodologias têm naturalmente que lidar com diversos tipos de conflitos, como conseqüência da expressividade desses modelos, porém com a vantagem de permitirem comparações em um nível mais alto de abstração.

3.2.3 - Etapas do Processo de Integração

Quanto às etapas cumpridas durante o processo de integração, podemos identificar apenas a metodologia de [NAVATHE 86] como possuindo as fases de pre-integração, comparação dos esquemas, assemelhação dos esquemas e fusão e reestruturação. [CASANOVA 83] possui uma abordagem de integração com uma fraca interação com o projetista é considera apenas as etapas de assemelhação dos esquemas e de fusão e reestruturação. [AL- FEDAGHI 81], [DAYAL 84] e [YAO 82] também realizam apenas essas duas atividades, porém com uma maior interação com o projetista. [BATINI 84] não define claramente a existência de uma fase de pre-integração, apesar de exigir do projetista a atribuição de pesos aos esquemas componentes. [BATINI 84] e [NAVATHE 86] possuem uma forte interação com o projetista durante a comparação e o assemelhação dos esquemas, até a tentativa de fusão dos mesmos.

3.2.4 - ESTRATÉGIAS DE INTEGRAÇÃO

O numero de esquemas a ser considerado para integração de cada vez é um fator bastante importante e deve ser decidido durante a fase de pre-integração. Dentre as metodologias aqui mencionadas, [BATINI 84] utilizam uma estratégia de integração binária do tipo escada, atribuindo pesos aos esquemas a integrar e esquemas integrados intermediários. [CASANOVA 83] utilizam uma estratégia de integração binária balanceada e [DAYAL 84] utiliza também uma estratégia binária sem contudo especificar o tipo. As demais metodologias utilizam estratégias n-árias do tipo "*one shot*" para integração. A escolha do tipo de estratégia tem forte influencia na complexidade do algoritmo de integração, conforme foi discutido no capítulo 2.3.1.

3.2.5 - TIPOS DE CONFLITOS CONSIDERADOS

Podemos identificar dois tipos de conflitos na comparação de esquemas: os conflitos de nomes e os conflitos estruturais. As metodologias baseadas no uso do modelo relacional [AL-FEDAGHI 81] [CASANOVA 83] ao assumirem o uso do esquema relacional universal, ignoram problemas relacionados com nomes e especificações contraditórias, não avaliando qualquer tipo de conflito. Metodologias baseadas no uso de modelos de dados semânticos permitem uma maior liberdade em termos de nomes e perspectivas de projeto compatíveis e incompatíveis, naturalmente lidando com um universo mais amplo de conflitos.

Apesar de fazer uso do modelo funcional, [YAO 82] não avalia conflitos de nomes nem conflitos estruturais. A metodologia de [DAYAL 84] que, como já dissemos, aplica-se à integração de BDs já existentes, avalia homonímia e sinonímia, além de conflitos estruturais como: diferenças de escala, diferenças em abstrações e o nível de confiabilidade e obsolescência dos dados. [BATINI 84] analisa conflitos de nomes, como

sinônimos e homônimos e conflitos estruturais, como inconsistências de tipos e conflitos entre restrições de integridade informadas. E [NAVATHE 86] avalia como conflitos de nomes: homônimos, sinônimos, asserções sobre equivalência de atributos e equivalências entre classes de entidades, e como conflitos estruturais: diferenças em níveis de abstração, diferenças entre papeis, grau e cardinalidade de relacionamentos, sendo a que considera o maior conjunto de conflitos entre esquemas.

Papeis referem-se às funções desempenhadas por conjuntos entidade em um relacionamento. Podemos também dizer que dois relacionamentos possuem o mesmo papel quando possuem a mesma finalidade em diferentes esquemas.

A identificação e resolução dos conflitos é um dos aspectos fundamentais no processo de integração de esquemas. Podemos observar que dentre as metodologias mencionadas, nenhuma fornece uma análise ou prova de completeza das operações de transformação de esquemas do ponto de vista de serem capazes de resolver qualquer tipo de conflito que possa existir. Todas elas têm como objetivo da atividade de assemelhação dos esquemas a obtenção de um único *esquema de consenso*, possivelmente através de alterações em algumas visões dos usuários.

A tabela 1 a seguir apresenta um resumo comparativo das metodologias para integração de esquemas analisadas.

Metodologia	Aplicabilidade	Modelo de Dados	Etapas *				Estratégia	Tipo de conflito	
			1	2	3	4		De nomes	Estruturais
Al-Fedaghi 81	Int. de visões no proj. conceitual	Relacional			X	X	N-ária One-shot	---	---
Casanova 83	Int. de visões no projeto lógico	Relacional estendido			X	X	Binária balanceada	---	---
Yao 82	Int. de visões no projeto lógico	Funcional			X	X	N-ária one-shot	---	---
Dayal 84	Integração de BDs	Funcional estendido		X	X	X	Binária	Homônimos Sinônimos	Dif. escalas Dif.abstrações Nível de obsolescência e confiabilidade dos dados
Batini 84	Int. de visões no Projeto Conceitual	Entidade-relacionamento estendido		X	X	X	Binária escada	Sinônimos Homônimos	Inconsist. Tipos Conflito entre restrições de integridade
Navathe 86	Integração de visões no Projeto Conceitual e Integração de BDs	Entidade-Categoria-Relacionamento	X	X	X	X	N-ária One-shot	Homônimos Sinônimos Asserções sobre equiv. De atributos Asserções sobre equiv. Classes de Entidades	Dif. Em níveis de abstrações Dif. Em papéis, graus e restric. De cardinalidade de relacionamentos

* Etapas

1 – Pré-integração
2 – Comparação dos esquemas
3 – Assemelhamento dos esquemas
4 – Fusão e reestruturação

TABELA 1: COMPARATIVO ENTRE METODOLOGIAS PARA INTEGRAÇÃO DE ESQUEMAS DE BDS.

CAPÍTULO 4

TEORIA DA EQUIVALÊNCIA DE ATRIBUTOS EM BDS

Em [LARSON 89] os autores apresentam uma teoria para a determinação de equivalências entre atributos que pode ser utilizada na integração de esquemas de BDs. O modelo de dados utilizado no trabalho é o modelo ECR. Segundo esse trabalho, duas classes de objetos podem ser integradas se seus atributos de identificação podem ser integrados. Para que dois atributos possam ser integrados, eles necessitam satisfazer algumas condições de equivalência entre si. Atributos que são equivalentes possuem várias características em comum. Essas características descrevem os atributos com respeito à classe de objetos a que eles pertencem e a correspondência entre as características de dois atributos fornece a base para o estabelecimento de equivalências entre eles e de equivalências entre classes de objetos e relacionamentos.

4.1 - EQUIVALÊNCIA DE ATRIBUTOS

São descritas oito características básicas dos atributos, que podem servir de apoio na determinação de equivalências. Entretanto, outras características também podem ser utilizadas pelos projetistas. Dentre as características apresentadas, as mais importantes na determinação de equivalências são:

A) A Unicidade (UN (**a**)), que corresponde ao conjunto de atributos, incluindo **a**, que juntos identificam uma instância de uma classe de objetos;

B) A cardinalidade, que é uma restrição no numero de valores que podem estar presentes em cada instância de uma classe de objetos, sendo definida em termos de limite mínimo (LB(**a**)) e limite máximo (UB(**a**)), com LB(**a**) = 0 significando

que o atributo pode possuir instâncias sem valores associados (valores nulos) e UB(**a**) > 1 significando

C) Domínio (DOM(**a**)) que diz respeito ao conjunto de valores que o atributo **a** pode assumir;

D) As restrições de integridade semânticas estáticas (*Static Integrity Constraints*) (SIC(**a**)) que são restrições envolvendo o atributo **a** e possivelmente outro atributo da mesma ou de outra classe de objetos, como dependências funcionais ou restrições de integridade referenciais;

E) Restrições de integridade semânticas dinâmicas (*State-Change Constraints*) (SCC(**a**)), que descrevem restrições nas alterações que os valores dos atributos podem sofrer e

F) Restrições de segurança (SEC(**a**)) que limitam o uso dos valores de um atributo.

Considerando $\mathbf{a_i}$ um atributo da classe de objetos **A** e $\mathbf{b_i}$ um atributo da classe de objetos **B**. $\mathbf{D_i}$ o maior subconjunto não nulo de DOM($\mathbf{a_i}$) e $\mathbf{R_i}$ o maior subconjunto não nulo de DOM($\mathbf{b_i}$), tal que existe uma função de mapeamento $\mathbf{f_i}$: $\mathbf{D_i} \rightarrow \mathbf{R_i}$, e $\mathbf{f_i}$ *: $\mathbf{R_i} \rightarrow \mathbf{D_i}$ (sua inversa). A função $\mathbf{f_i}$, a ser definida pelo administrador de banco de dados (**ABD**), deve possuir as seguintes propriedades, que são denominadas Propriedades Básicas de Equivalência:

A) $\mathbf{f_i}$ é uma função bijetora. Existe uma correspondência um a um entre os elementos de $\mathbf{D_i}$ e $\mathbf{R_i}$.

B) Para cada operação permitida sobre $\mathbf{a_i}$ existe uma operação permitida equivalente sobre $\mathbf{b_i}$ e vice-versa.

C) Todas as restrições de integridade valem sobre os mapeamentos de $\mathbf{f_i}$ e $\mathbf{f_i}$ *.

D) Todas as restrições de troca de estado valem sobre os mapeamentos.

E) Todas as restrições de segurança valem sobre os mapeamentos.

F) As funções de mapeamento preservam as dependências funcionais.

G) As funções de mapeamento preservam os identificadores.

Se o **ABD** consegue determinar uma função de mapeamento entre os atributos $\mathbf{a_i}$ e $\mathbf{b_i}$ que preserve essas propriedades, então os atributos $\mathbf{a_i}$ e $\mathbf{b_i}$ são equivalentes e podem ser integrados.

Os autores apresentam um teorema que estabelece as condições sobre as quais a propriedade básica de equivalência A implica nas propriedades **C** a **G** restantes:

"Teorema 1. Dado o seguinte:

1) $\mathbf{f_i}$, uma função de mapeamento de $\mathbf{D_i}$ contida em DOM($\mathbf{a_i}$) para $\mathbf{R_i}$ contida em DOM($\mathbf{b_i}$) que satisfaz as propriedades básicas de equivalência.
2) $\mathbf{fa_{ij}}$, uma dependência funcional do atributo $\mathbf{a_i}$ para o atributo $\mathbf{a_j}$ e
3) $\mathbf{fb_{ij}}$, uma dependência funcional do atributo $\mathbf{b_i}$ para $\mathbf{b_j}$ e
4) Existe um mapeamento, digamos $\mathbf{f_j}$, de $\mathbf{D_j}$ contido em DOM($\mathbf{a_j}$) para $\mathbf{R_j}$ contido em DOM($\mathbf{b_j}$) que satisfaz a propriedade básica de equivalência A. Então a função de mapeamento $\mathbf{f_j}$ também satisfaz todas as propriedades básicas de equivalência".

Esse teorema garante que, uma vez assegurado que uma função de mapeamento entre dois atributos $\mathbf{a_i}$ e $\mathbf{b_i}$ satisfaz as propriedades básicas de equivalência, os atributos restantes que sejam funcionalmente dependentes de $\mathbf{a_i}$ e $\mathbf{b_i}$, respectivamente, e possuam função de mapeamento com uma inversa, também serão considerados equivalentes.

Equivalências entre dois atributos podem ser de três tipos: Fraca, Forte e Disjunta. A equivalência forte implica que atualizações são possíveis sobre o esquema integrado, na equivalência fraca, apenas recuperações são permitidas, enquanto que a equivalência

disjunta ocorre quando, apesar de não ser possível estabelecer uma função f que possa mapear valores de um atributo para valores de outro atributo, os papeis dos dois atributos são os mesmos.

Dois atributos possuem o mesmo papel quando, independente de possuírem ou não o mesmo domínio, são utilizados para modelar uma mesma característica de diferentes objetos do mundo real. Por exemplo, em uma entidade Professor o atributo Qualificação com domínio {doutor, mestre, especialista, bacharel} tem o mesmo papel do atributo Nivel_escolaridade da entidade Aluno, com domínio {2° grau, 1° grau}, pois, apesar de possuírem domínios disjuntos, tem a mesma finalidade.

Dois atributos **a** e **b** possuem equivalência forte se: A) É possível definir uma função $\mathbf{f_i}$ que mapeia elementos do domínio de **a** em elementos do domínio de **b** e essa função possui uma inversa e B) As duas funções satisfazem as propriedades básicas de equivalência.

A equivalência forte pode ser do tipo α, quando relaciona os valores de um atributo em uma instância de um conjunto entidade com os valores de outro atributo em uma instância de outro conjunto entidade em algum momento no tempo, e pode ser do tipo β quando relaciona os valores de um atributo de um conjunto entidade com os valores de um outro atributo de outro conjunto entidade sobre todos os momentos no tempo, ou seja, todos os estados do banco de dados . Para equivalências do tipo α–Forte (*Strong-*α), podem existir 4 tipos de equivalência. Sejam os atributos **a** e **b** de diferentes classes de objetos:

A) **a** *Strong-*α *Equal* **b** - Se existe uma correspondência um a um entre o conjunto de todos os valores de **a** e o conjunto de todos os valores de **b**.

B) **a** *Strong-α Contains* **b** - Se existe uma correspondência um a um entre um subconjunto dos valores de **a** e todos os valores de **b**.

C) **a** *Strong-α Contained-in* **b** - Se existe uma correspondência um a um entre todos os valores de **a** e um subconjunto dos valores de **b**.

D) **a** *Strong-α Overlaps* **b** - Se existe uma correspondência um a um entre um subconjunto dos valores de **a** e um subconjunto dos valores de **b**.

A definição das equivalências β é obtida a partir das equivalências α, da seguinte forma:

Dados um atributo **a** de uma classe de objetos **A** e um atributo **b** da classe de objetos **B**,

A) Se **a** *Strong-α Equal* **b** vale para todos os pontos no tempo para os quais ambos **A** e **B** existem, então **a** *Strong-β Equal* **b**.

B) Se para todo ponto no tempo, um dos dois **a** *Strong-α Equal* **b** ou **a** *Strong-α Contains* **b** vale, então **a** *Strong-β Contains* **b**.

C) Se para todo ponto no tempo, um dos dois **a** *Strong-α Equal* **b** ou **a** *Strong-α Contained-in* **b** vale, então **a** *Strong-β Contained-in* **b**.

D) Se **a** *Strong-β Equal* **b**, ou **a** *Strong-β Contains* **b**, ou **a** *Strong-β Contained-in* **b** vale em diferentes instâncias no tempo, então **a** *Strong-β Overlap* **b**.

Equivalência Forte permite aos usuários executar atualizações nos valores em **D** e **R**.

A equivalências fracas (*Weak*) entre dois atributos também podem ser do tipo α e β, com as mesmas características das equivalências fortes. Entretanto, na equivalência fraca, atualizações nos valores dos atributos não são possíveis.

Dois atributos **a** e **b** possuem equivalência fraca se todas as condições de equivalência forte valem, com as seguintes exceções: A) Nenhuma função inversa precisa existir; B) As propriedades básicas de equivalência **C**, **D** e **E** são substituídas por: Sejam $\mathbf{f_1},\mathbf{f_2},...,\mathbf{f_n}$ as funções de mapeamento associadas respectivamente com atributos $\mathbf{a_1},\mathbf{a_2},...,\mathbf{a_n}$. Cada restrição em SIC(**a**) envolvendo atributos $\mathbf{a_1},\mathbf{a_2},...,\mathbf{a_n}$ deve valer em SIC(**b**) depois de trocar $\mathbf{a_i}$ por $\mathbf{f_i}(\mathbf{a_i})$, i = 1,...n. Dessa forma cada restrição em SCC(**a**) e SEC(**a**) vale em SCC(**b**) e SEC(**b**). Assim, o relacionamento fraco não é reflexivo.

A diferença básica entre equivalência forte e fraca entre atributos é a habilidade para atualizações. Na equivalência forte são possíveis atualizações devido ao fato das funções de mapeamento possuírem uma inversa, enquanto que na equivalência fraca, as funções de mapeamento não possuem necessariamente uma inversa e, dessa forma, atualizações que preservem as propriedades básicas de equivalência podem não ser possíveis. O **ABD** deve determinar se o esquema integrado será utilizado para atualizações ou apenas consultas, e assim, determinar se devem ser usadas equivalências fortes ou fracas.

Mesmo quando equivalências forte e fraca não existem, atributos ainda podem ser combinados, desde que possuam os mesmos papeis, ainda que não exista uma função f que possa mapear valores de um atributo para os do outro atributo.

Dois atributos **a** e **b** são equivalentes disjuntos (**a** α-*Disjoint-role-equal* **b**) sempre que os domínios de **a** e **b** são disjuntos, não permitindo a existência de equivalências Forte ou Fraca e **a** e **b** tem o mesmo papel. A equivalência α–*Disjoint* implica em β–*Disjoint* porque os domínios dos atributos não possuem interseção em qualquer momento no tempo.

4.2 - EQUIVALÊNCIA DE CLASSES DE OBJETOS

A equivalência entre duas classes de objetos será analisada através da verificação da equivalência entre os valores dos seus atributos de identificação (chaves). As classes de objetos serão equivalentes da mesma maneira que seus atributos de identificação.

Um par de classes de objetos pode ter as equivalências *Equal*, *Contains*, *Contained-in*, *Overlap* e *Disjoint*. Entretanto os autores utilizam ρ para representar equivalências entre classes de objetos em lugar de α e β.

Para representar as equivalências entre classes de objetos os autores utilizam o conceito de Estados do Mundo Real (RWS). Um estado do mundo real de uma classe de objetos **A** (RWS(**A**)) corresponde às instâncias da classe de objetos **A** em um dado momento no tempo. O conceito de RWS é similar ao de extensão de um BD, exceto que, quando duas diferentes extensões de duas classes de objetos **A** e **B** em dois diferentes BDs representam momentaneamente as instâncias dos mesmos objetos do mundo real, temos que a extensão de **A** é diferente da extensão de **B**, enquanto que RWS(**A**) é igual a RWS(**B**).

Sejam duas classes de objetos **A**, com identificador $\mathbf{k_1}$ e atributos $(\mathbf{a_1},\mathbf{a_2},...,\mathbf{a_m})$, e **B** com identificador $\mathbf{k_2}$ e atributos $(\mathbf{b_1},\mathbf{b_2},...,\mathbf{b_n})$. Tomando $T(\mathbf{k_1})$ como sendo o conjunto dos valores de $\mathbf{k_1}$ para todas as instâncias da classe de objetos **A** em um certo ponto no tempo e $T(\mathbf{k_2})$ os valores de $\mathbf{k_2}$ para todas as instâncias da classe de objetos **B** no mesmo ponto no tempo, então é possível definir uma função de mapeamento f: $\mathbf{D} \rightarrow \mathbf{R}$ para cada ponto no tempo. **D** é o maior subconjunto não nulo de $T(\mathbf{k_1})$ e **R** é o maior subconjunto não nulo contido em $T(\mathbf{k_2})$. A equivalência entre instâncias das classes de objetos **A** e **B** pode ser definida usando esse mapeamento, dependendo do relacionamento entre $\mathbf{k_1}$ e **D** e do

relacionamento entre $\mathbf{k_2}$ e **R**. Os autores consideram a equivalência entre classes de objetos (ρ) sempre do tipo β, ou seja, vale para todos os RWS's.

Caso 1: Se $\mathbf{k_1}$ *Strong* β–*Equal* $\mathbf{k_2}$, então

RWS(**A**) ρ–*Equal* RWS(**B**)e as classes de objetos são integradas como na figura 7.

Caso 2: Se $\mathbf{k_1}$ *Strong* β-*Contains* $\mathbf{k_2}$, então

RWS(**A**) ρ–*Contains* RWS(**B**), como na figura 7.

Caso 3: Se $\mathbf{k_1}$ *Strong* β-*Contained-in* $\mathbf{k_2}$, então

RWS(**A**) ρ-*Contained-in* RWS(**B**), como na figura 7.

Caso 4: Se $\mathbf{k_1}$ *Strong* β-*Domain-disjoint-role-equal* $\mathbf{k_2}$ então

RWS(**A**) ρ–*Disjoint* RWS(**B**), sendo os objetos integrados como no casos 1 ou 5.

Caso 5: Se $\mathbf{k_1}$ *Strong* β–*Overlap* $\mathbf{k_2}$ então

RWS(**A**) ρ–*Overlap* RWS(**B**), provocando a criação de uma nova classe de objetos, como na figura 7.

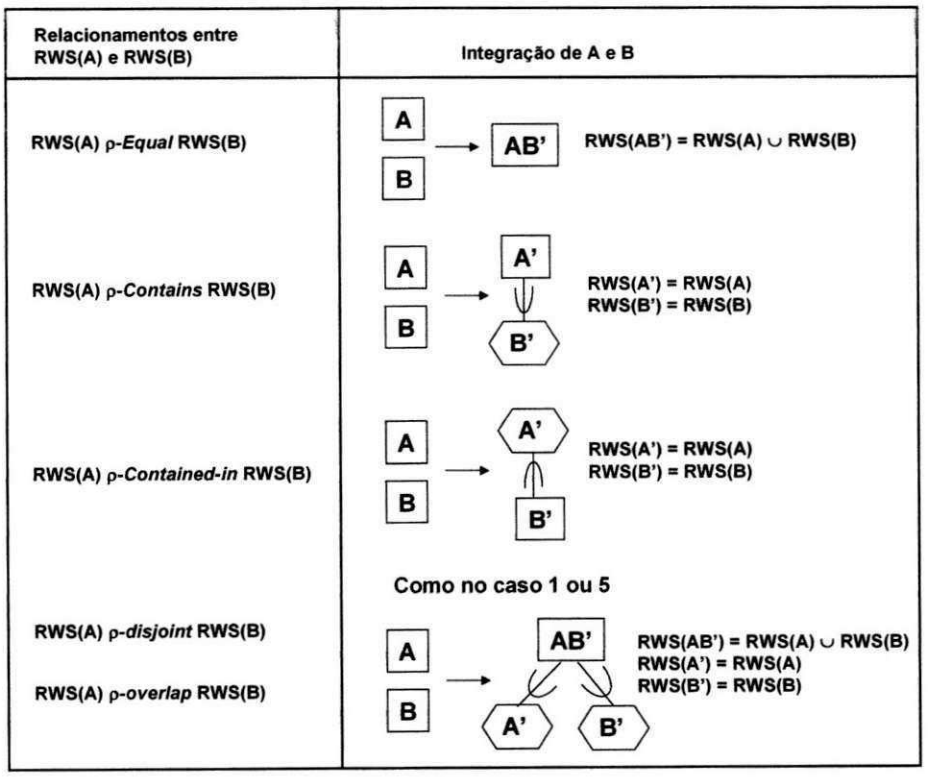

FIGURA 7: INTEGRAÇÕES POSSÍVEIS ENTRE DOIS OBJETOS

A equivalência ρ entre duas classes de objetos **A** e **B** implica em certas equivalências β entre os atributos de **A** e **B**, que são resumidas nos teoremas a seguir:

Teorema 2: Dado **a** um atributo de **A**, e **b** um atributo de **B**. Se RWS(**A**) ρ–*Equal* RWS(**B**), então a única equivalência possível entre **a** e **b** é **a** β–*Equal* **b**.

Teorema 3: Se RWS(**A**) ρ–*Contains* RWS(**B**), então **a** β–*Contains* **b**, ou **a** β–*Equal* **b**.

Teorema 4: Se RWS(**A**) ρ–*Contained-in* RWS(**B**), então **a** β–*Contained-in* **b**, ou **a** β–*Equal* **b**.

4.3 - EQUIVALÊNCIA DE RELACIONAMENTOS

A equivalência entre relacionamentos é verificada de maneira similar a equivalência de classes de objetos, sendo que é utilizado γ para denotar equivalência entre relacionamentos.

Uma instância de um relacionamento **RA**, de grau **n**, composto das classes de objetos $\mathbf{A_1}$, $\mathbf{A_2}$,..., $\mathbf{A_n}$ pode ser interpretada como a agregação dos atributos de identificação das classes de objetos que participam do relacionamento. Dessa forma, uma instância de **RA** pode ser identificada pelos valores de ($\mathbf{a_1}$, $\mathbf{a_2}$,..., $\mathbf{a_n}$).

Se em um dado instante, T($\mathbf{a_1}$, $\mathbf{a_2}$,..., $\mathbf{a_n}$) denota os valores de ($\mathbf{a_1}$, $\mathbf{a_2}$,..., $\mathbf{a_n}$) para todas as possíveis instâncias do conjunto relacionamento **RA** e T($\mathbf{b_1}$, $\mathbf{b_2}$,..., $\mathbf{b_n}$) denota os valores de ($\mathbf{b_1}$, $\mathbf{b_2}$,..., $\mathbf{b_n}$) para todas as possíveis instâncias do conjunto relacionamento **RB**, então é possível definir uma função de mapeamento f: **D** → **R**. **D** é o maior subconjunto contido em T($\mathbf{a_1}$, $\mathbf{a_2}$,..., $\mathbf{a_n}$) é **R** o maior subconjunto contido em T($\mathbf{b_1}$, $\mathbf{b_2}$,..., $\mathbf{b_n}$). A equivalência γ entre RWS (**RA**) e RWS (**RB**) se dará da seguinte forma:

Caso 1: Se **D** = T($\mathbf{a_1}$, $\mathbf{a_2}$,..., $\mathbf{a_n}$) e **R** = T($\mathbf{b_1}$, $\mathbf{b_2}$,..., $\mathbf{b_n}$) em todos os instantes, então RWS (**RA**) γ–*Equal* RWS (**RB**), e os relacionamentos **RA** e **RB** são integrados como na figura 8.

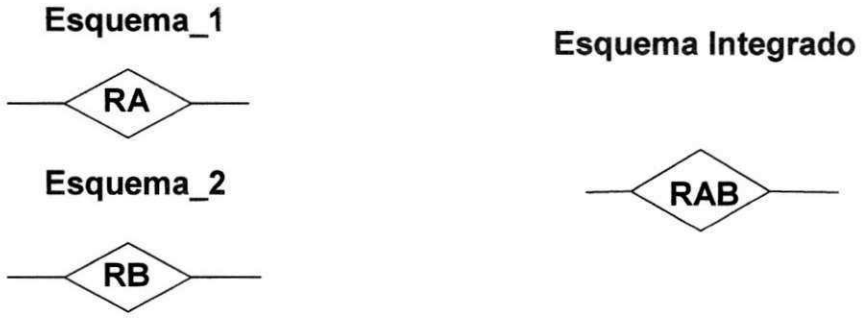

FIGURA 8: INTEGRAÇÃO DE RELACIONAMENTOS γ–EQUAL

Caso 2: Se **D** = T($\mathbf{a_1}$, $\mathbf{a_2}$,..., $\mathbf{a_n}$) em todos os instantes e T($\mathbf{b_1}$, $\mathbf{b_2}$,..., $\mathbf{b_n}$) $\subseteq$ ou T($\mathbf{b_1}$, $\mathbf{b_2}$,..., $\mathbf{b_n}$) = **R**, então RWS (**RA**) γ–*Contains* RWS (**RB**), e os relacionamentos serão integrados como na figura 9 abaixo.

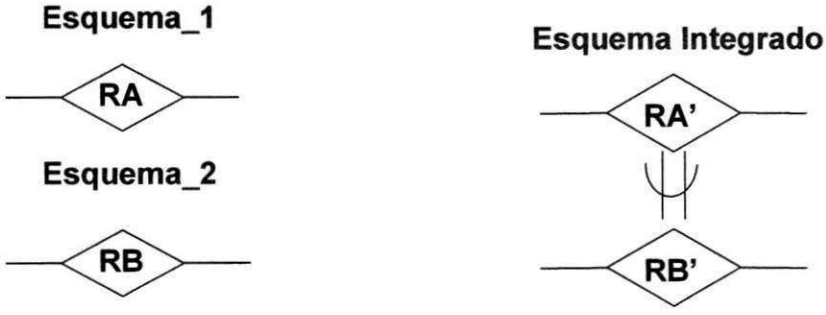

FIGURA 9: INTEGRAÇÃO DE RELACIONAMENTOS γ–CONTAINS

Caso 3: Se T($\mathbf{a_1}$, $\mathbf{a_2}$,..., $\mathbf{a_n}$) $\subseteq$ **D** ou T($\mathbf{a_1}$, $\mathbf{a_2}$,..., $\mathbf{a_n}$) = **D** e **R** = T($\mathbf{b_1}$, $\mathbf{b_2}$,..., $\mathbf{b_n}$) em todos os instantes, então RWS (**RA**) γ–*Contained-in* RWS (**RB**). Os relacionamentos serão integrados como na figura 10 a seguir.

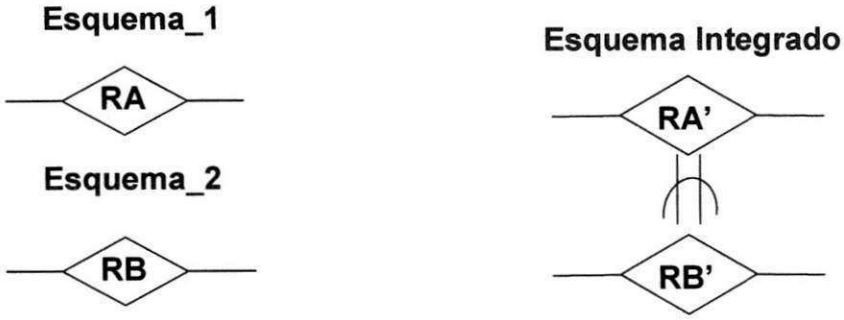

FIGURA 10: INTEGRAÇÃO DE RELACIONAMENTOS γ–CONTAINED-IN

Caso 4: Se **D R** = ∅ então RWS (**RA**) γ–*Disjoint* RWS (**RB**). Neste caso, os relacionamentos não são integrados, a menos que possuam o mesmo papel, ou seja, ($\mathbf{a_1}$, $\mathbf{a_2}$,...,$\mathbf{a_n}$) Domains-*disjoint-role-Equal* ($\mathbf{b_1}$, $\mathbf{b_2}$,..., $\mathbf{b_n}$), sendo a integração feita como no caso 1 ou 5.

Caso 5: Se quaisquer das restrições 1 a 4 pode valer em diferentes instantes de tempo, então RWS (**RA**) γ–*Overlaps* RWS (**RB**). Neste caso a integração dos relacionamentos se dera como na figura 11 a seguir. No exemplo, as classes de objetos $\mathbf{A_1}$ e $\mathbf{B_1}$ foram integradas considerando uma equivalência do tipo ρ–*Overlaps* entre seus domínios e $\mathbf{A_2}$ e $\mathbf{B_2}$ como possuindo uma equivalência do tipo ρ–*Equal*.

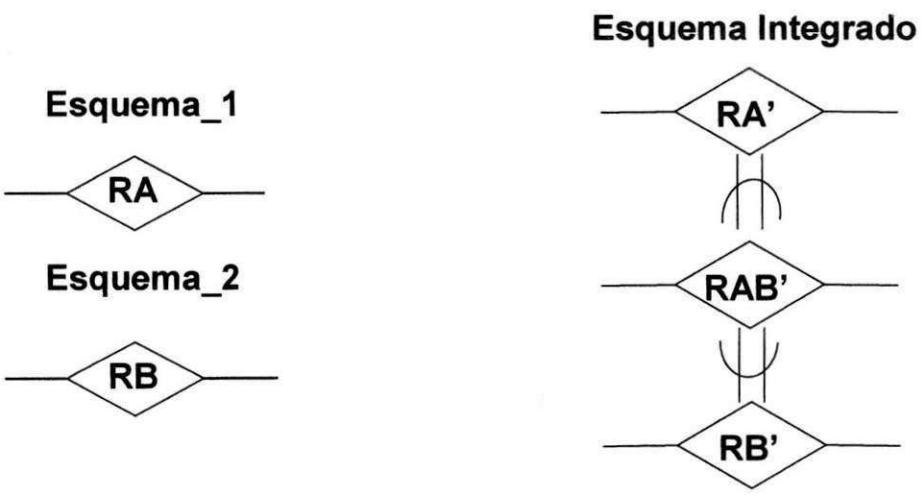

FIGURA 11: INTEGRAÇÃO DE RELACIONAMENTOS γ–OVERLAPS

Apesar da integração de relacionamentos geralmente envolver relacionamentos binários, também é possível através dessa metodologia integrar conjuntos relacionamento de graus diferentes. Além disso, seguindo a abordagem da verificação das equivalências entre atributos e conjuntos de atributos, também é possível integrar relacionamentos com classes de objetos.

Quando são integrados relacionamentos de graus diferentes, o maior grau entre eles deve prevalecer no esquema integrado.

Uma outra abordagem para integração de relacionamentos é também apresentada, considerando-se os conjuntos relacionamento como atributos, possivelmente multivalorados, de cada classe de objetos participante do relacionamento. Esses atributos são atributos agregados que serão constituídos de todos os atributos do relacionamento original e mais os atributos de identificação das classes de objetos que participam no relacionamento. Dessa forma, os relacionamentos podem ser integrados, sendo tratados como atributos agregados. Os autores consideram que dois atributos agregados são β equivalentes somente se cada um dos respectivos subatributos que compõem o atributo agregado são β equivalentes. Após a integração dos atributos agregados, eles são re-alocados em seus relacionamentos correspondentes.

4.4 - ESTRATÉGIAS PARA INTEGRAÇÃO DOS ATRIBUTOS

Quatro estratégias são apresentadas para a integração dos atributos das classes de objetos e dos relacionamentos que foram integrados.

A primeira estratégia consiste em integrar todos os atributos não disjuntos. Se os atributos **a** da classe de objetos **A** e **b** da classe de objetos **B** possuem algum tipo de equivalência entre si, seja ela β-*Equal*, β–*Contains*, β–*Contained-in* ou β–*Overlaps*, então eles serão integrados formando um novo atributo **c'** de uma classe de objetos **C** no esquema integrado. Das quatro estratégias para integração de atributos, esta é a que apresenta um esquema resultante com menor numero de atributos. Sendo a mais indicada para usuários ocasionais ou inexperientes que desejam evitar a complexidade de muitos atributos similares.

A segunda estratégia para a integração de atributos consiste em integrar apenas os atributos que são β–*Equal*. São integrados os atributos **a** e **b** para formar o atributo **c'** apenas se **a** β–*Equal* **b**. Para as outras equivalências, os atributos **a** e **b** não serão

integrados. Esta estratégia produz um esquema integrado com um numero bastante elevado de atributos, já que apenas os atributos β–*Equal* equivalentes serão integrados, sendo os demais atributos apenas transportados para o esquema integrado.

A terceira estratégia compreende a integração apenas dos atributos que são β–*Equal*, porém com a indicação de associações entre atributos similares não integrados. O esquema obtido poderá ter um pouco mais de atributos que na estratégia 2, porém o esquema integrado resultante explicitamente descreve a equivalência dos atributos, oferecendo ao usuário mais informação semântica.

A quarta estratégia consiste em integrar todos os atributos não disjuntos e migrar valores entre atributos. Além de integrar todos os atributos que possuem qualquer tipo de equivalência, como na estratégia 1, esta estratégia permite que valores comuns entre dois dos atributos originais apareçam apenas uma vez no esquema integrado.

4.5 - AVALIAÇÃO DA TEORIA DA EQUIVALÊNCIA DE ATRIBUTOS

A Teoria da Equivalência de Atributos oferece importantes contribuições às pesquisas na área de integração de visões e integração de Bancos de Dados, avaliando aspectos formais do problema da equivalência entre classes de objetos, relacionamentos e atributos. Fornece também uma base segura para o desenvolvimento de ferramentas automatizadas para o auxílio no projeto de BDs, e em particular, para a integração de esquemas de BDs. Muitos dos problemas encontrados durante a integração de esquemas, como conflitos de nomes, conflitos estruturais e diferenças de escala, podem ser reavaliados em termos da equivalência de atributos.

CAPÍTULO 5

NOSSA METODOLOGIA PARA A INTEGRAÇÃO DE ESQUEMAS

Nossa proposta é o desenvolvimento de uma ferramenta automatizada para o auxílio na integração de esquemas, que leva em consideração os principais aspectos das metodologias analisadas no capítulo 3, e cujo algoritmo principal de integração se baseia no conceito da equivalência de atributos descrito no capítulo 4. O modelo de dados usado chama-se Entidade-Categoria-Relacionamento (ECR) [ELMASRI 85], uma extensão ao modelo ER.

A abordagem seguida consiste no desenvolvimento de um sistema baseado em conhecimento para a integração dos esquemas de BD. O sistema guia o projetista durante o processo de integração, apresentando alternativas para a integração das classes de objetos, relacionamentos e atributos, além de possibilitar uma constante interação com o projetista, que tem a facilidade de discordar dos resultados obtidos durante o processo de integração.

As decisões tomadas pelo sistema durante a integração são baseadas em um conjunto de regras de produção, como será descrito de forma mais detalhada no capítulo 6. Essas regras constituem a parte mais importante da base de conhecimento e é o que caracteriza o sistema como um especialista em integração de esquemas. Esse conjunto de regras foi desenvolvido a partir das analises das metodologias existentes para a integração de esquemas e da nossa experiência, adquirida a partir desses estudos.

Como a atividade de integração ainda depende fortemente do projetista do BD e do modelo de dados escolhido para o projeto, procuramos oferecer um sistema que evitasse a tomada de decisões incondicionais em situações onde existissem mais de uma alternativa de integração. Apesar desses cuidados, o sistema ainda correria o risco de se tornar

obsoleto na medida que surgissem novas metodologias de integração baseadas no uso de extensões do modelo ER. A par desses problemas, desenvolvemos um sistema evolutivo, que permite ao usuário incorporar novo conhecimento a base de regras do sistema, à medida que novas metodologias venham a surgir ou o próprio usuário formule regras consistentes para integração a partir de sua experiência com o uso da ferramenta.

Conseqüentemente, o sistema pode evoluir para versões mais completas e aprimoradas sem a necessidade da intervenção de seus implementadores originais [MOREIRA 90], uma vez que a lógica e funcionamento do sistema esta associada às regras, que estão acessíveis ao usuário, e não dependente do controle, como em sistemas convencionais.

A ferramenta possui ainda um módulo de explanação, que permite ao usuário acompanhar os passos do processo de integração, com explicações sobre as ações tomadas e perguntas feitas pelo sistema, e também possibilita o treinamento de usuários inexperientes. Cada regra possui uma explanação associada que será informada ao usuário sempre que ele tenha duvidas quanto às ações tomadas pelo sistema.

5.1 - Estratégia de Integração

Optamos por utilizar uma estratégia de integração binária do tipo escada. Conseqüentemente, dois esquemas são integrados de cada vez. Essa decisão tem o objetivo de reduzir a complexidade do algoritmo de integração é simplificar as atividades de comparação e assemelhação dos esquemas.

Não são atribuídos pesos aos esquemas, porém, um novo esquema componente é sempre integrado com um esquema intermediário, resultante de uma integração anterior, como mostra a figura 12. A ordem em que os esquemas parciais deverão ser integrados

fica inteiramente a critério do projetista, que devera levar em consideração a importância relativa de cada um desses esquemas parciais.

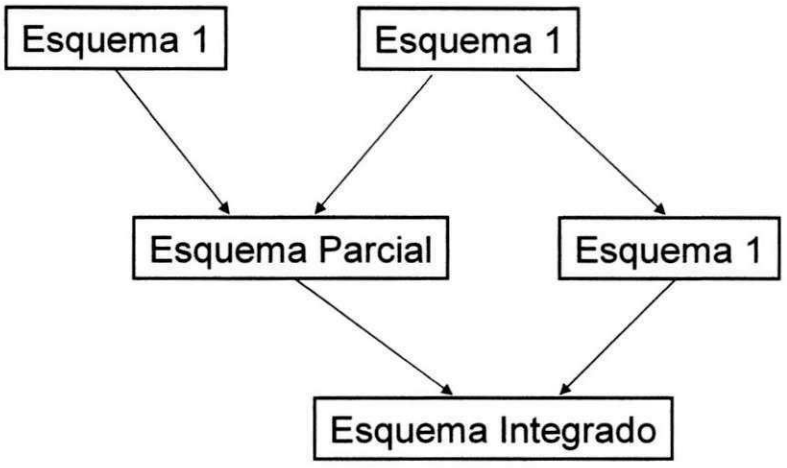

FIGURA 12: ESTRATÉGIA DE INTEGRAÇÃO

O grau de importância associado aos esquemas pode ser medido levando-se em consideração fatores como:

A) O nível de especialização do usuário entrevistado;

B) A posição ocupada pelo usuário na Organização;

C) O grau de confiança que pode ser associado às informações obtidas.

A atribuição de grau de importância aos esquemas compreende uma etapa imediatamente posterior a Modelagem das Visões descrita na seção 2.1. Deixamos essa tarefa a cargo do projetista, que terá a responsabilidade de fornecer ao nosso sistema os esquemas a integrar na ordem decrescente de sua importância associada.

Esquemas parciais integrados, entretanto, prevalecem sobre esquemas de usuários. Com isso buscamos uma maior convergência em direção a um esquema global integrado que seja o mais representativo em relação aos diversos esquemas componentes, sem que se tenha um significativo aumento da complexidade dos algoritmos de integração, dando ao

projetista alguma liberdade na escolha da ordem em que os esquemas serão apresentados para integração.

5.2 - MODELO DE DADOS

Optamos pelo uso de um modelo de dados semântico para a representação dos esquemas a serem integrados. A nossa escolha recaiu sobre o modelo ECR [ELMASRI 85]. O modelo ECR é uma extensão ao modelo ER [CHEN 76]. Essa escolha deve-se a dois fatores em especial: 1) O modelo ER é um dos modelos mais utilizados atualmente para o projeto de BDs, e um dos mais conhecidos nos meios acadêmicos e de pesquisa [BATINI 86]; 2) Necessitávamos de uma extensão do modelo ER original que incorporasse os conceitos de hierarquias de generalização e subconjunto, o que é feito pelo modelo ECR através do conceito de Categoria. Além disso, o modelo ECR amplia o uso de restrições estruturais sobre os relacionamentos, para especificar como entidades podem estar associadas.

O modelo ECR descreve a semântica dos dados classificando o domínio da aplicação em Entidades (que podem ser objetos ou eventos) e relacionamentos entre as entidades. O modelo inclui como extensão ao modelo ER o conceito de Categoria, para representar hierarquias de generalização e subconjunto. Restrições Estruturais são usadas para especificar de que maneira entidades podem participar em relacionamentos. Um exemplo são as restrições de cardinalidade, que são úteis para capturar mais semântica.

O modelo utiliza as construções: Conjunto de Entidades, Conjunto de Relacionamentos, Categoria e Atributo. O termo Classe de Objetos refere-se tanto a um conjunto de entidades, quanto a uma categoria. Num conjunto de entidades, todas as entidades têm os mesmos atributos. Uma categoria pode ser usada para modelar um subconjunto de uma classe de objetos, herdando os atributos da classe de objetos da qual

ela é um subconjunto. A figura 13 mostra um exemplo de um esquema ECR. Restrições sobre o numero de instâncias de um relacionamento, do qual uma entidade pode participar, são representadas por um par de inteiros (Min,Max). Cada entidade pode participar de, no mínimo, (Min) instâncias e, no máximo, (Max) instâncias do relacionamento.

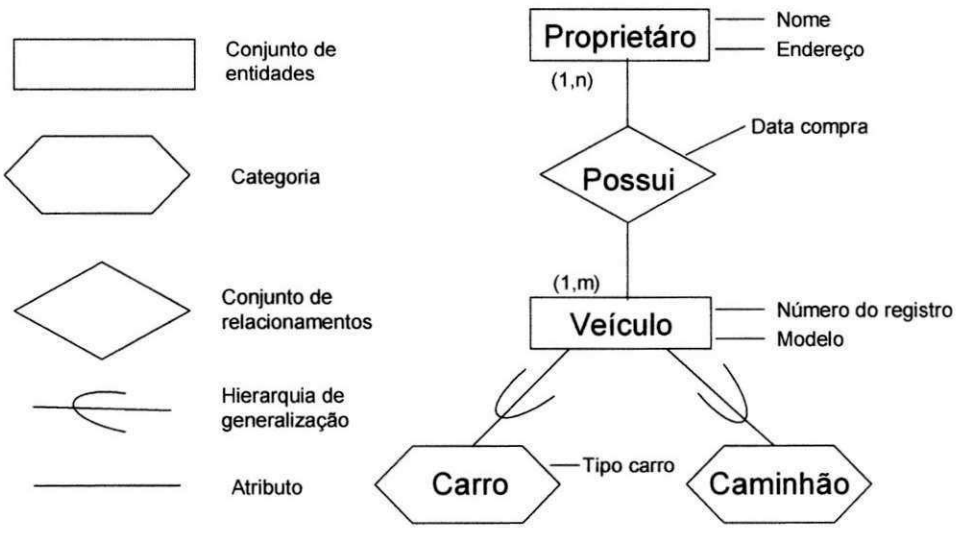

FIGURA 13: EXEMPLO DE UM ESQUEMA ECR

5.3 - ETAPAS DO PROCESSO DE INTEGRAÇÃO

Como em outras metodologias para integração de esquemas, nosso sistema consiste da realização de diversas atividades para que possamos obter o esquema final integrado. Separamos o processo de integração em quatro etapas. Essas etapas compreendem todas as atividades descritas na seção 2.3 para a integração de esquemas. As etapas são:

A) Obtenção dos esquemas a integrar;

B) Especificação de asserções;

C) Integração dos esquemas e

D) Melhoramento e reestruturação do esquema global integrado. O fluxo de processamento dessas etapas é apresentado na figura 14 a seguir:

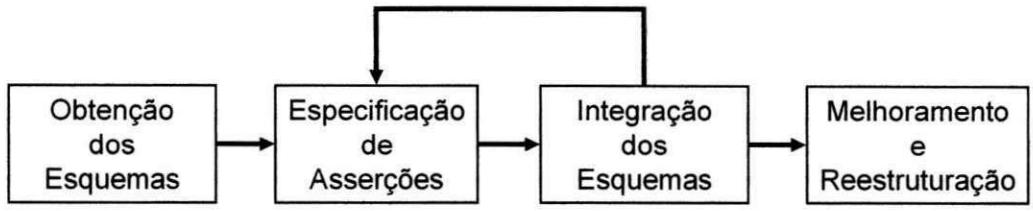

FIGURA 14: FLUXO DAS ETAPAS DO PROCESSO DE INTEGRAÇÃO

5.3.1 - OBTENÇÃO DOS ESQUEMAS A INTEGRAR

Nesta etapa, os esquemas são checados quanto a sua consistência semântica e completeza na medida em que são fornecidos pelo projetista. Isso abrange a fase de pre-integração, onde transformações intra-esquema são realizadas, caso necessário, antes que ele sirva como entrada para o processo de integração.

Transformações intra-esquemas podem ser realizadas a partir de:

A) Existência de homônimos ou sinônimos entre os conceitos do esquema;

B) Existência no esquema de entidade com atributo único, que pode eventualmente ser transformada em atributo multivalorado de outra entidade a qual esteja relacionada;

C) Existência de entidades isoladas, ou seja, que não estejam relacionadas ou sejam categorias de outras entidades, podendo ocorrer essa hipótese apenas no caso de falhas durante um processo anterior de integração;

D) Existência de ciclos provocados por relacionamentos ou hierarquias de generalização redundantes, implicando na eliminação de um relacionamento ou de uma ligação de subconjunto;

E) Existência de conjuntos relacionamento que, juntamente com as entidades associadas, podem ser transformados em agregação.

Essas transformações somente serão realizadas com a concordância do projetista e poderão implicar na integração de conceitos dentro do próprio esquema que esta sendo

analisado. Tanto os esquemas componentes quanto os esquemas parciais integrados passam pelo processo de pre-integração.

5.3.2 - ESPECIFICAÇÃO DE ASSERÇÕES

Esta etapa compreende a fase de comparação dos esquemas. São fornecidas pelo projetista informações sobre as classes de objetos, relacionamentos e atributos dos esquemas a serem integrados. O usuário devera informar, para cada um dos dois esquemas a integrar, os conceitos sinônimos entre os dois esquemas, as funções de mapeamento entre os atributos dos esquemas, os tipos de equivalência entre relacionamentos e os papeis das classes de objetos nos relacionamentos, se necessários para o processo de integração. O sistema avalia a necessidade de obter tais informações, de acordo com o conjunto de afirmações feitas pelo usuário e resultados já obtidos pelo sistema. Papeis dos relacionamentos, por exemplo, são necessários apenas quando eles são de mesmo grau e envolvem entidades já integradas. Dessa forma, o usuário não necessita perder tempo fornecendo informações que não serão efetivamente utilizadas.

São passíveis de integração entidades sinônimas, homônimas, que possuam chaves do mesmo tipo, atributos das chave equivalentes ou que estejam relacionadas com outras entidades que já foram integradas. São passíveis de integração relacionamentos que envolvam entidades integradas e sejam sinônimos, homônimos, equivalentes ou cujos papeis sejam os mesmos. Os atributos passíveis de integração são os pertencentes a entidades ou relacionamentos integrados e que sejam sinônimos, homônimos ou equivalentes.

As informações de sinonímia são de iniciativa do usuário, no inicio do processo de especificação de asserções. O sistema apenas solicita a informação dos sinônimos existentes. As informações de homonímia são obtidas diretamente pelo sistema e as

demais informações, que serão armazenadas como asserções, serão prestadas pelo projetista na medida em que o sistema as solicite. São solicitadas informações apenas para os conceitos passíveis de integração. Ao final, o projetista ainda poderá incluir novas asserções que julgue convenientes, como mapeamentos entre conceitos. Todas essas asserções servem de base para a integração dos conceitos pertencentes aos dois esquemas. Algumas das decisões tomadas pelo sistema durante a especificação de asserções dependem de resultados obtidos durante a etapa de integração dos esquemas. Dessa forma, após a integração dos esquemas, o sistema precisa retornar e verificar as regras de especificação de asserções.

5.3.3 - INTEGRAÇÃO DOS ESQUEMAS

Nesta etapa os esquemas são integrados de acordo com as asserções especificadas. Inicialmente são integradas as classes de objetos de acordo com as equivalências existentes entre os atributos chave, em seguida, todos os relacionamentos que envolviam classes de objetos já integradas são analisados para integração. Informações relativas a esses relacionamentos são pedidas ao usuário para que o sistema possa decidir se eles devem ser ou não integrados. Finalmente, os atributos são integrados de acordo com as estratégias para integração de atributos descritas no capítulo 4.4.

A integração das classes de objetos compreende as fases de integração de hierarquias de generalização e integração de entidades. Os primeiros objetos a serem integrados são as hierarquias de generalização. No modelo utilizado, tratam-se dos conjuntos de entidades que possuem categorias subordinadas, como é apresentado na figura 13 da seção 5.2. A integração de hierarquias de generalização exige um tratamento especial das categorias envolvidas no processo, e dos atributos que serão herdados por essas categorias. A participação de categorias no processo será dependente do modo como

foram integradas as entidades as quais estejam subordinadas. A integração direta de categorias com outras classes de objetos não é realizada, pois pode implicar em herança inadequada de atributos.

5.3.3.1 - INTEGRAÇÃO DE HIERARQUIAS DE GENERALIZAÇÃO

Considerando a necessidade de integrar os objetos apresentados na figura 15 a seguir, temos 5 alternativas para a integração de uma hierarquia de generalização (**A**) com um conjunto entidade (**B**), dependendo do tipo de equivalência existente entre os domínios de **A** e **B**.

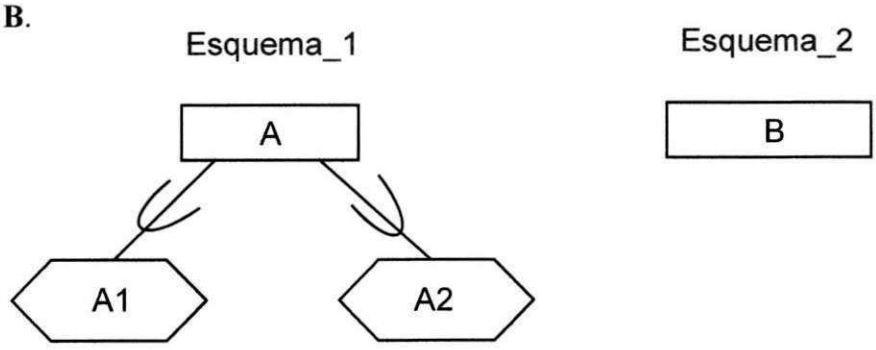

FIGURA 15: INTEGRAÇÃO DE UMA HIERARQUIA DE GENERALIZAÇÃO COM UM CONJUNTO ENTIDADE

Se RWS(**A**) ρ–*Equal* RWS(**B**), **A** e **B** são integrados gerando um novo objeto **AB'**. São criadas duas novas categorias $\mathbf{A_{1'}}$ e $\mathbf{A_{2'}}$ subordinadas a **AB'**. Os atributos de **AB'** serão a união dos atributos de **A** com os de **B**. $\mathbf{A_{1'}}$ e $\mathbf{A_{2'}}$ passam a herdar os atributos de **AB'**, como mostra a figura 16 a seguir.

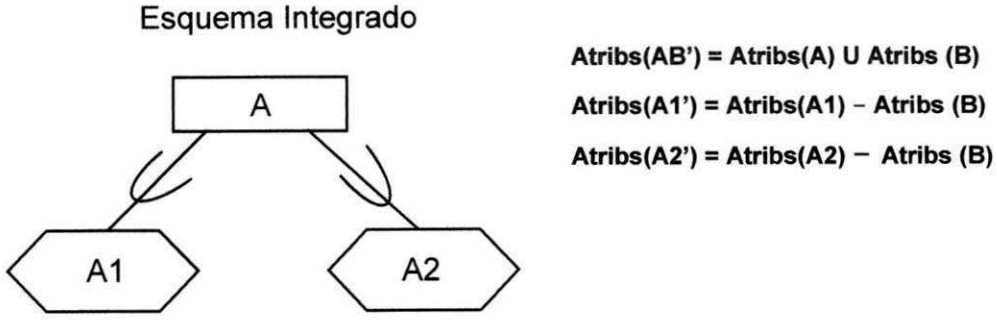

FIGURA 16: INTEGRAÇÃO DE HIERARQUIAS DE GENERALIZAÇÃO. CASO 1.

Se RWS(**A**) ρ–*contained-in* RWS(**B**), é criada uma nova entidade **B'** e uma nova categoria **A'** a ela subordinada. As categorias $\mathbf{A_1}$ e $\mathbf{A_2}$ são transportadas para o esquema integrado sem altercações. Os atributos de **B'** serão os mesmos atributos de **B** e os atributos de **A'** serão os atributos de **A** menos os atributos de **B**. O resultado da integração é apresentado na figura 17.

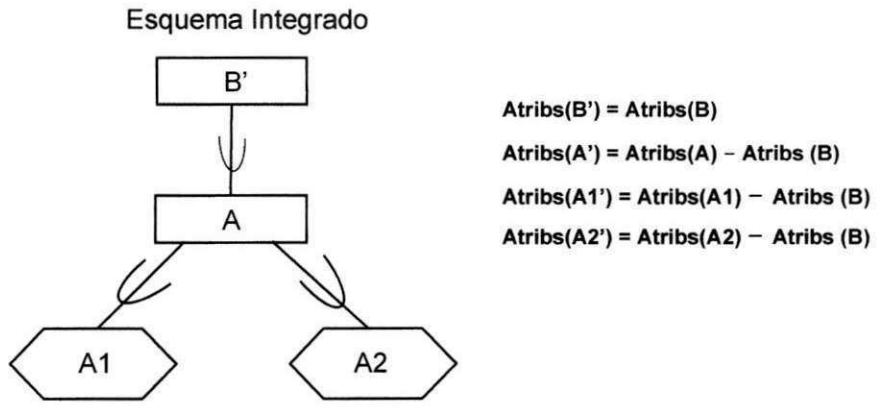

FIGURA 17: INTEGRAÇÃO DE HIERARQUIAS DE GENERALIZAÇÃO. CASO 2.

Se RWS(**A**) ρ–*contains* RWS(**B**), será criada uma nova entidade **A'** e uma categoria **B'** a ela subordinada. $\mathbf{A_1}$ e $\mathbf{A_2}$ serão transportados para o esquema integrado subordinados a **A'**. Os atributos de **A'** serão os mesmos atributos de **A**, e os atributos de **B'** serão os atributos de **B** menos os atributos de **A**. O sistema ainda avalia a possibilidade de integrar **B'** com $\mathbf{A_{1'}}$ ou $\mathbf{A_{2'}}$ dentro dos critérios para integração de entidades descritos no capítulo 4. Um resultado possível para a integração é apresentado na figura 18.

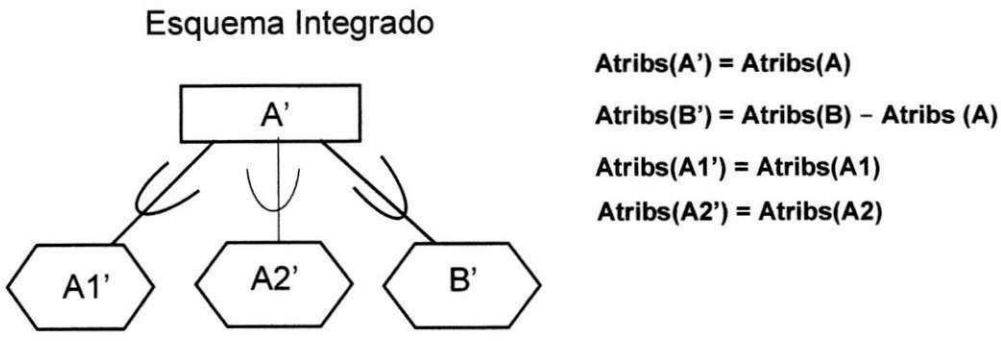

FIGURA 18: INTEGRAÇÃO DE HIERARQUIAS DE GENERALIZAÇÃO. CASO 3.

Se RWS(**A**) ρ–*overlap* RWS(**B**) será criada uma nova entidade **AB'** e as entidades **A** e **B** serão transportadas para o esquema integrado juntamente com $\mathbf{A_1}$ e $\mathbf{A_2}$. Os atributos de **AB'** serão a interseção entre os atributos de **A** com os de **B**. Os atributos de **A'** serão os atributos de **A** menos os de **B** e os de **B'** serão os de **B** menos os de **A**. A figura 19 apresenta o resultado da integração.

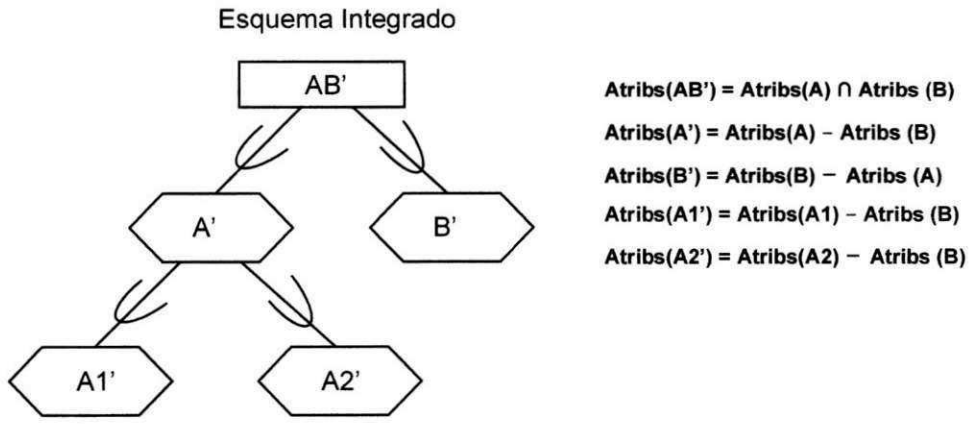

FIGURA 19: INTEGRAÇÃO DE HIERARQUIAS DE GENERALIZAÇÃO. CASO 4.

Se RWS(**A**) ρ–*disjoint* RWS(**B**). Os esquemas não serão integrados, ou caso o projetista deseje, poderão ser integrados de forma similar ao caso 1 ou 4, analogamente ao discutido na seção 4.2.

A integração de duas hierarquias de generalização também admite 5 possíveis alternativas. A figura 20 apresenta dois esquemas a integrar contendo hierarquias de generalização.

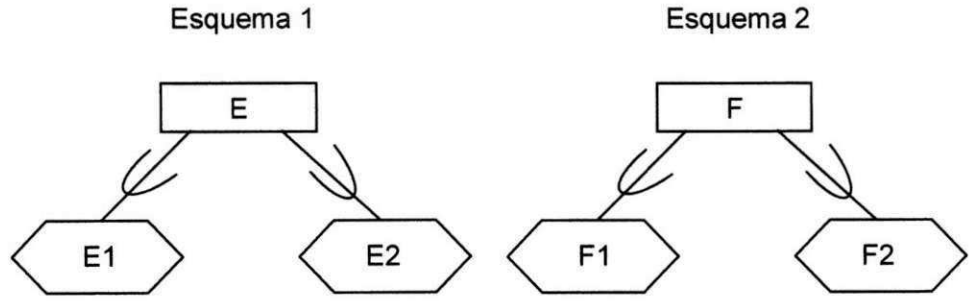

FIGURA 20: INTEGRAÇÃO DE DUAS HIERARQUIAS DE GENERALIZAÇÃO

Se RWS(**E**) ρ–*equal* RWS(**F**), será criada uma nova entidade **EF'** no esquema integrado e as categorias $\mathbf{E_1}$, $\mathbf{E_2}$, $\mathbf{F_1}$, $\mathbf{F_2}$ serão transportadas para o esquema integrado. Os atributos de **EF'** serão a união dos atributos de **E** e **F**. As categorias serão ainda analisadas para possível integração de $\mathbf{E_1}$ ou $\mathbf{E_2}$ com $\mathbf{F_1}$ ou $\mathbf{F_2}$. Um possível resultado é apresentado na figura 21 a seguir.

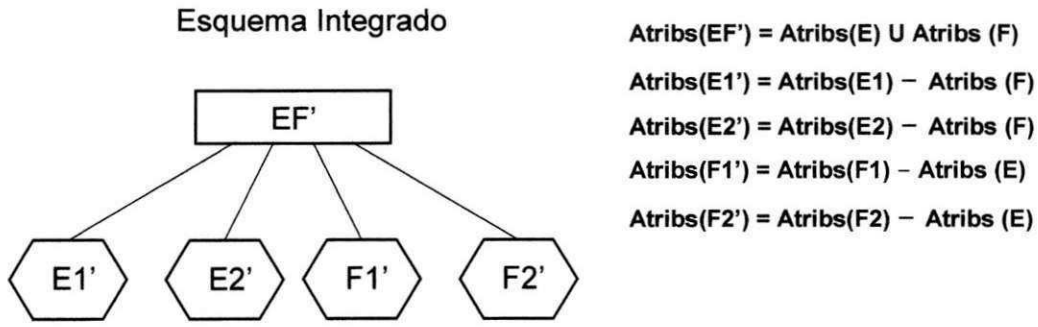

FIGURA 21: INTEGRAÇÃO DE HIERARQUIAS DE GENERALIZAÇÃO. CASO 6.

Se RWS(**E**) ρ–*contained-in* RWS(**F**), será criada uma nova entidade **F'** e uma categoria **E'** a ela subordinada. $\mathbf{F_1}$ e $\mathbf{F_2}$ serão transportadas subordinadas a **F'**. $\mathbf{E_1}$ e $\mathbf{E_2}$ serão transportadas subordinadas a **E'**. Os atributos de **F'** serão os atributos de **F** enquanto que os atributos de **E'** serão os atributos de **E** menos atributos de **F**. O resultado é apresentado na figura 22.

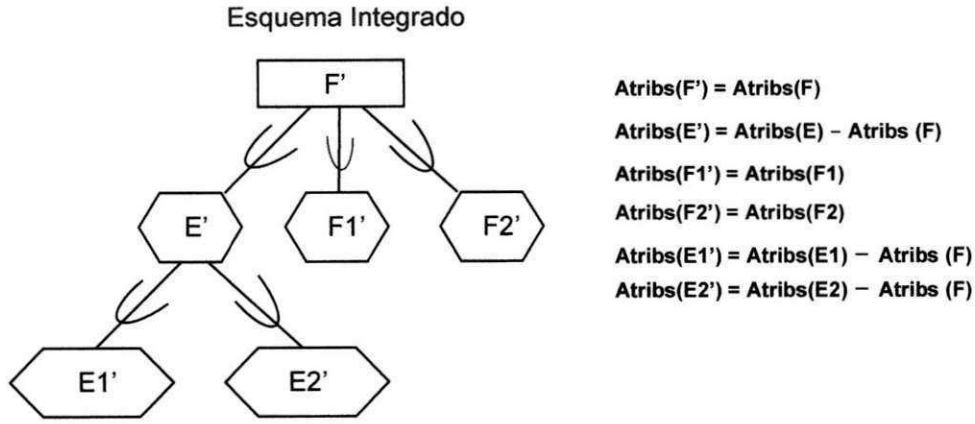

FIGURA 22: INTEGRAÇÃO DE HIERARQUIAS DE GENERALIZAÇÃO. CASO 7.

Se RWS(**E**) ρ *contains* RWS(**F**), a integração será feita de forma análoga ao caso anterior.

Se RWS(**E**) ρ *overlaps* RWS(**F**) uma nova entidade **EF'** e criada. **E'** e **F'** serão categorias subordinadas a **EF'**. Os atributos de **EF'** serão os atributos comuns a **E** e **F**. Os atributos de **E'** serão os atributos de **E** que não estão em **F** e os de **F'** serão os atributos de **F** que não estão em **E**. Não é possível a integração de categorias neste caso. O resultado da integração é apresentado na figura 23.

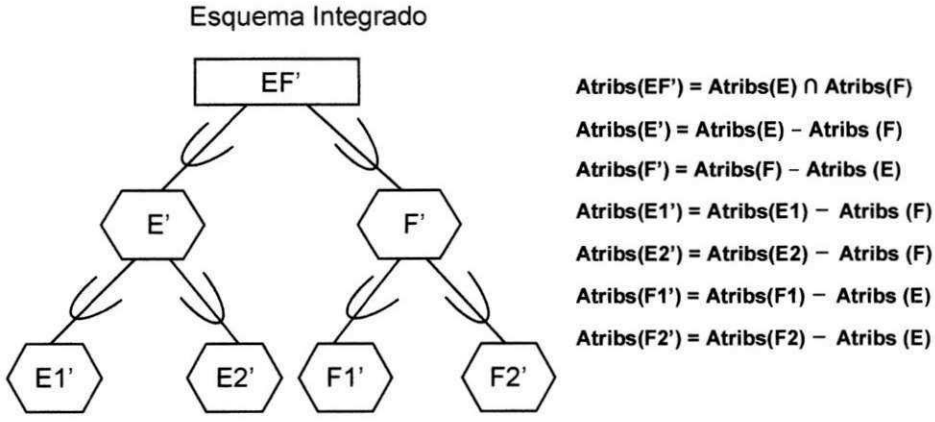

FIGURA 23: INTEGRAÇÃO DE HIERARQUIAS DE GENERALIZAÇÃO. CASO 9.

Se RWS(**E**) ρ *disjoint* RWS(**F**) os esquemas não serão integrados. Se o projetista desejar, poderão ser integrados de forma análoga aos casos 6 ou 9.

O sistema suporta herança de atributos, tornando possíveis as integrações anteriormente descritas.

5.3.3.2 - INTEGRAÇÃO DE CONJUNTOS ENTIDADE

A integração de entidades que não possuem categorias dependentes se da de acordo com os tipos de equivalência existentes entre os domínios de seus atributos chave, como mostrado na figura 7 da seção 4.2. Mesmo para conceitos homônimos, sinônimos, ou com chaves possuidoras de domínios de mesmo tipo, será necessário identificar a equivalência existente entre os domínios de suas chaves.

O procedimento utilizado para a verificação de entidades equivalentes segue a seguinte ordem:

A) Entidades sinônimas;

B) Entidades homônimas;

C) Entidades com domínios de mesmo tipo;

D) Entidades relacionadas com outras entidades já integradas;

E) Entidades que possuem atributos em comum.

Dessa forma, procuramos evitar uma explosão combinatorial na comparação dos conceitos dos dois esquemas. Entidades que já possuem asserções que levem a sua integração ou que já tenham sido integradas não são avaliadas novamente.

5.3.3.3 - INTEGRAÇÃO DE CONJUNTOS RELACIONAMENTO

A integração de relacionamentos compreende os conjuntos relacionamento associados a classes de objetos anteriormente integradas. Além de realizarmos a integração baseada na equivalência de atributos, que leva em consideração as equivalências existentes entre os atributos chave das classes de objetos envolvidas, utilizamos também no processo de integração as restrições de cardinalidade das entidades associadas e papeis das entidades nos relacionamentos.

5.3.3.3.1 - INTEGRAÇÃO DE RELACIONAMENTOS DE MESMO GRAU

Analisamos as restrições de cardinalidade e o tipo de equivalência existente entre os relacionamentos. O tipo de equivalência é obtido a partir das equivalências existentes entre as chaves das entidades envolvidas nos relacionamentos a integrar e é confirmada com o projetista.

No caso de equivalência γ *equal*, as ligações do novo relacionamento integrado poderão se suceder de duas diferentes maneiras, dependendo das restrições de cardinalidade existentes:

A) Havendo as mesmas restrições de cardinalidade e supondo que as entidades foram integradas com uma equivalência ρ *overlap*, os relacionamentos serão integrados conforme a figura 24.

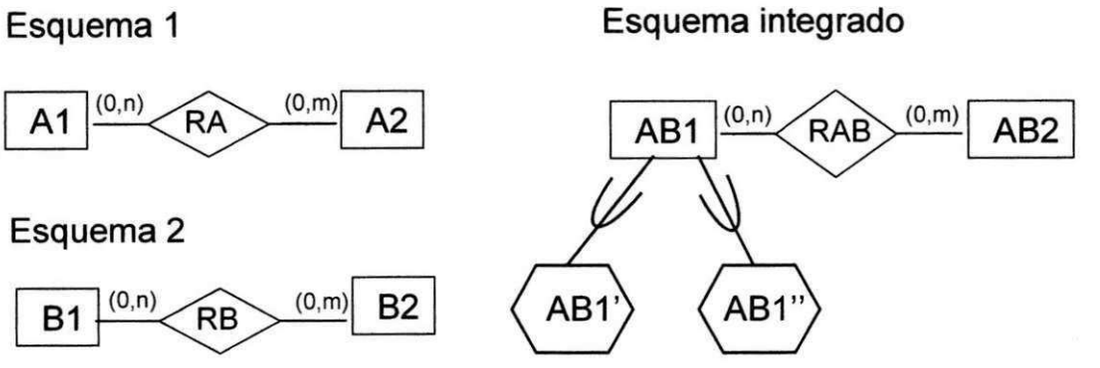

FIGURA 24: INTEGRAÇÃO DE RELACIONAMENTOS γ EQUAL. CASO 1.

B) Havendo restrições de cardinalidade diferentes, onde uma visão é mais restrita que a outra, e supondo que as entidades foram integradas com uma equivalência do tipo ρ *contains*, o relacionamento ficara ligado a classe de objetos mais restrita no esquema integrado, como é mostrado na figura 25.

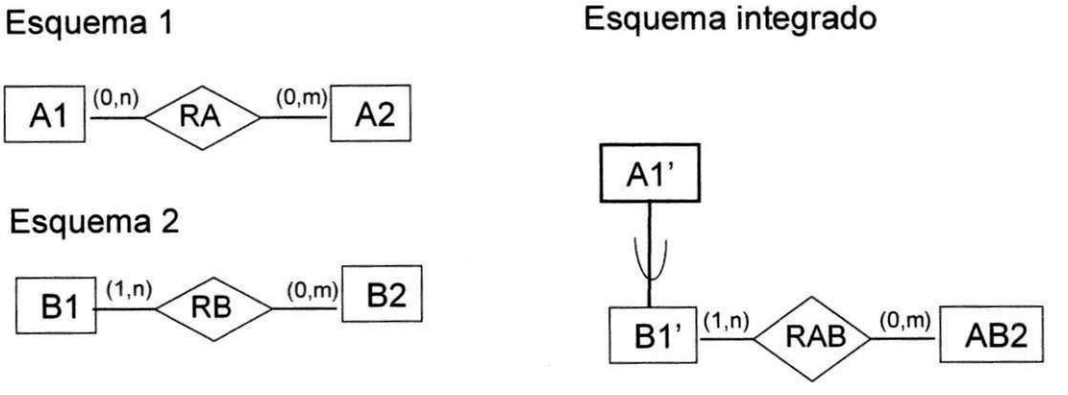

FIGURA 25: INTEGRAÇÃO DE RELACIONAMENTOS γ EQUAL. CASO 2.

No caso de equivalência γ *contains* ou γ *contained-in*, os dois relacionamentos são transportados para o esquema integrado, e suas restrições de cardinalidade são mantidas, criando-se uma ligação do tipo subconjunto entre os novos relacionamentos, de forma similar ao apresentado nas figuras 9 e 10 da seção 4.3.

Na equivalência γ *disjoint*, os relacionamentos não poderão ser integrados, sendo apenas transportados para o esquema integrado.

Na equivalência γ *overlap*, são criados 3 novos relacionamentos no esquema integrado. O resultado é similar ao apresentado na figura 11 da seção 4.3.

5.3.3.3.2- INTEGRAÇÃO DE RELACIONAMENTOS DE GRAUS DIFERENTES

Nesse tipo de integração prevalecera no esquema integrado o maior grau entre os relacionamentos. Três situações podem ocorrer quando integramos relacionamentos de graus diferentes: Os relacionamentos podem ser integráveis, condicionalmente integráveis ou não integráveis.

Os relacionamentos são integráveis quando, apesar de apresentarem graus diferentes, representam essencialmente a mesma informação. Essa informação pode ser obtida a partir da verificação da existência de atributos comuns ou equivalentes nas entidades envolvidas. Ainda assim, a integração dos relacionamentos devera ser confirmada pelo projetista.

Nesta situação se enquadra o caso especial da integração de uma entidade com um conjunto relacionamento. A figura 26 apresenta um exemplo desse tipo de integração. A entidade Automóvel no Esquema_2 pode ser compreendida como um conjunto relacionamento de grau 1 [NAVATHE 86]. Neste caso, o resultado da integração será o mesmo do Esquema_1, já que o relacionamento de maior grau deve prevalecer no esquema integrado. Outros relacionamentos que estejam associados a Automóvel no

Esquema_2 serão associados a Pessoa ou Veiculo no esquema integrado, a critério do projetista. Por exemplo, se uma entidade Multa existisse no Esquema_2 e estivesse associada a Automóvel, poderia ser associada a Veiculo no esquema integrado.

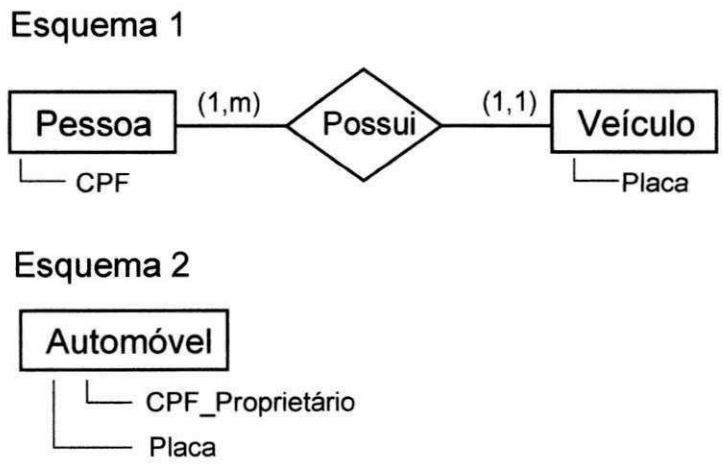

FIGURA 26: INTEGRAÇÃO DE UM RELACIONAMENTO COM UMA ENTIDADE

Os relacionamentos são condicionalmente integráveis quando o relacionamento de menor grau pode ser derivado do de maior grau. O sistema não suporta completamente esse tipo de verificação, porém pode dar alguma indicação ao projetista a partir da análise dos atributos dos objetos associados aos relacionamentos e dos atributos dos próprios relacionamentos. A figura 27 apresenta um exemplo de relacionamentos condicionalmente integráveis.

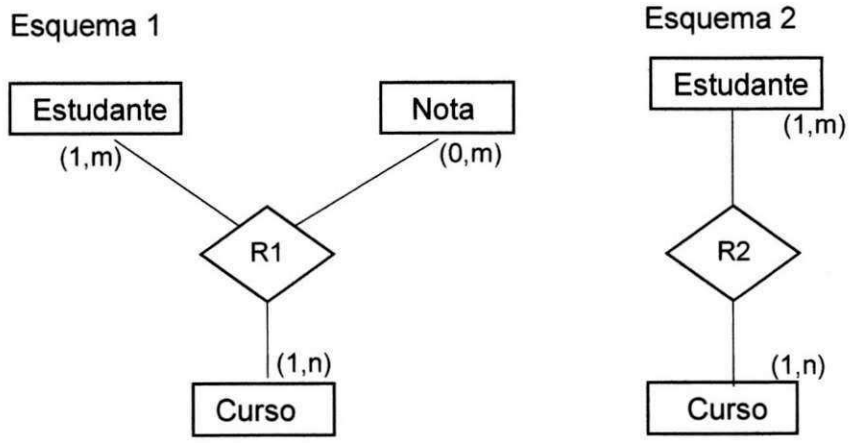

FIGURA 27: CONJUNTOS RELACIONAMENTO CONDICIONALMENTE INTEGRÁVEIS

Os relacionamentos são não integráveis quando, apesar de envolverem objetos equivalentes nos dois esquemas, a semântica dos relacionamentos não é a mesma, ou seja, os relacionamentos foram modelados com finalidades diferentes.

5.3.4 - MELHORAMENTO E REESTRUTURAÇÃO

Esta etapa compreende a avaliação do esquema integrado no que diz respeito a sua consistência, além da análise de possíveis ciclos ou inconsistências porventura gerados durante o processo de integração. Esta atividade ainda não esta implementada em nosso trabalho.

5.4 - ANÁLISE DA NOSSA METODOLOGIA

Nossa metodologia aplica-se a integração de esquemas de BDs na fase de projeto. Utiliza um modelo de dados semântico derivado do modelo ER. Isso permite ao usuário mais liberdade durante a modelagem de suas aplicações que o uso de modelos orientados a registro. Entretanto, isso possibilita diversas formas de representação para um mesmo subconjunto do mundo real, aumentando a responsabilidade do sistema e o compromisso do projetista com os resultados obtidos durante a integração.

Todas as etapas discutidas na seção 2.3 são consideradas em nossa metodologia, o que a torna bastante completa.

A escolha de uma estratégia de integração binária reduz a complexidade do tratamento dos esquemas, inclusive para o projetista. Em contrapartida, ha um aumento no numero de passos necessários para a obtenção do esquema global integrado. Apesar disso, acreditamos que essa ainda é uma das melhores alternativas para integração.

Todos os conflitos de nome tratados pelas outras metodologias são também avaliados pelo nosso sistema. Quanto aos conflitos estruturais, nosso trabalho não

considera apenas aqueles relacionados exclusivamente com a integração de BDs já existentes.

Procuramos incorporar os aspectos positivos das metodologias analisadas, principalmente na questão relativa a análise de conflitos. Nosso trabalho deriva principalmente da Teoria da Equivalência de Atributos em BDs, porém utilizamos um tratamento de conjuntos relacionamento semelhante à metodologia de [NAVATHE 86]. A verificação de conflitos entre restrições de cardinalidade também pode ser encontrada no trabalho de [BATINI 84].

CAPÍTULO 6

ARQUITETURA DO SISTEMA

Procuramos definir uma estrutura modular, que privilegiasse a independência entre os diversos módulos do sistema e ao mesmo tempo facilitasse o trabalho de implementação, documentação e compreensão do projeto, sem contudo comprometer o desempenho geral do sistema.

Optamos por uma arquitetura do tipo Sistema Baseado em Regras [BOUZEGHOUB 85] [WATERMAN 86]. Essa escolha é justificada principalmente pelo fato de haver uma tendência por parte da maioria das pessoas em expressar seus conhecimentos e técnicas para resolução de problemas em termos de regras do tipo SE Condições ENTÃO Ações [HAYES-ROTH 85]. Além disso, procuramos utilizar uma maquina de inferência que fosse a mais independente possível da aplicação, tentando permitir a atualização da base conhecimento, inclusive das regras do sistema, sem alteração na estrutura de controle do sistema.

Nosso projeto segue uma tendência atual de desenvolvimento de sistemas baseados em conhecimento, utilizando diversas formas de representação do conhecimento [MONGIOVI 90]. Utilizamos no nosso sistema Regras de Produção, Quadros (*Frames*) e representação lógica através de predicados Prolog.

A figura 27 apresenta a arquitetura do sistema e a seguir são discutidos os diversos aspectos dos módulos envolvidos.

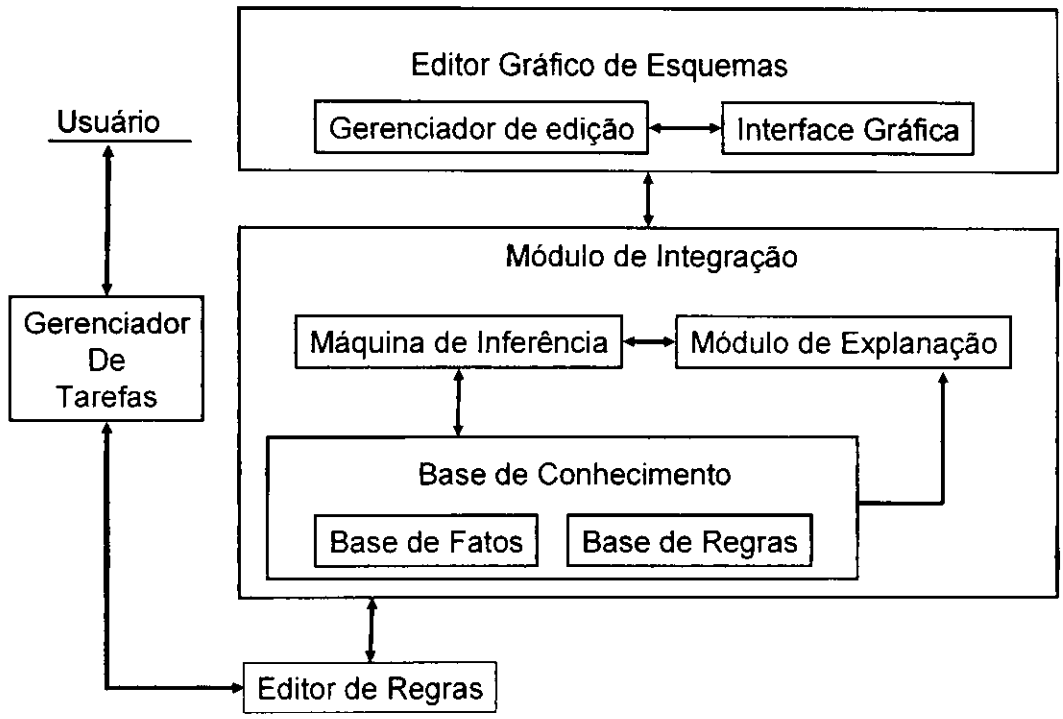

FIGURA 27: ARQUITETURA DO SISTEMA

6.1 - GERENCIADOR DE TAREFAS

O Gerenciador de Tarefas é responsável pelo monitoramento do sistema, auxiliando o usuário na seleção da tarefa desejada e inicializando o sistema de acordo com as opções do usuário. Apresenta uma interface baseada em menu, onde se pode optar entre as seguintes funções: A) Edição gráfica de esquemas; B) Integração de esquemas e C) Edição de regras.

A edição gráfica de esquemas compreende tanto a inclusão de novos esquemas de dados pelo usuário quanto à alteração de esquemas anteriormente cadastrados. A integração de esquemas compreende as atividades realizadas para a obtenção do esquema global integrado a partir dos esquemas informados pelo usuário. A edição de regras permite excluir, alterar ou incluir novas regras a Base de Regras do sistema, e só pode ser utilizada por um usuário realmente habilitado a manipular a Base de Conhecimento do sistema.

6.2 - EDITOR GRÁFICO DE ESQUEMAS

Permite a edição gráfica dos esquemas que serão integrados, além do próprio esquema resultante, oferecendo ainda facilidade para navegação na estrutura dos esquemas, já que estes não podem ser mostrados por completo em apenas uma tela de edição. O editor possibilita a inclusão, exclusão e alteração de esquemas, que são fatos do sistema. O módulo gerenciador de edição é assistido por uma interface gráfica que procura tornar a manipulação do sistema mais agradável para o usuário, já que ele pode trabalhar com os esquemas ECR na forma em que eles são representados graficamente [TOMASSIA 85] [BATINI 85] . Os esquemas incluídos pelo usuário são representados no sistema como conjuntos de quadros (*frames*), que serão discutidos na seção 6.3.3.1.

O editor faz também a validação semântica dos esquemas manipulados, impedindo que o usuário informe esquemas inconsistentes ou semanticamente incorretos. Para tanto, existe um módulo de tratamento de erros que procura indicar ao usuário eventuais falhas cometidas devido a uma operação inadequada do sistema, ou provocadas por atualizações realizadas contra o esquema editado.

Sempre que o usuário desejar incluir um conceito ou realizar atualizações no esquema editado, uma análise de consistência é feita para evitar inclusão de dados semanticamente inconsistentes ou transformação do esquema para estados incorretos. Os erros detectados pelo editor de esquemas são:

A) Inclusão de relacionamentos sem a existência dos objetos participantes. Um relacionamento só poderá ser incluído se as classes de objeto associadas já existirem;

B) Definição de cardinalidade mínima maior que cardinalidade máxima para atributos ou classes de objetos;

C) Definição de cardinalidade máxima igual a zero para atributos ou classes de objetos;

D) Inclusão de dois objetos ou dois relacionamentos com o mesmo nome no esquema editado;

E) Inclusão de dois atributos com o mesmo nome em um mesmo objeto ou relacionamento;

F) Definição de entidades sem atributos;

G) Definição de relacionamento possuindo atributos com o mesmo nome de atributos pertencentes às classes de objetos associadas ao relacionamento;

H) Definição de objetos não ligados a outros objetos através de relacionamentos ou hierarquias;

I) Redundância na definição de atributos entre uma entidade e uma categoria;

J) Definição de um conceito sem a inclusão de suas características. As características de cada conceito serão apresentadas na seção 6.3.3.1;

K) Exclusão de um objeto associado a um relacionamento de grau 2, sem a exclusão do relacionamento;

L) Exclusão de uma entidade sem a exclusão das categorias a ela subordinadas;

M) Especificação de superclasse incompatível com o conceito incluído. Um atributo por exemplo só pode ter como superclasse um outro atributo.

N) Especificação de grau de um relacionamento menor que 2.

Todos esses erros são indicados para o projetista e quando possível, a solução adequada também é sugerida. Outras verificações derivadas também são realizadas, como por exemplo a tentativa de excluir todos os atributos de uma entidade ou provocar homonímia indesejável através da alteração de nomes.

O módulo de validação pode ser ativado tanto automaticamente, para validação de uma operação, quanto pelo usuário, para a validação do esquema como um todo, fazendo a verificação de todos os conceitos pertencentes ao esquema, com suas correlações.

A interface gráfica utilizada apresenta menus sobrepostos que orientam o usuário nas operações realizadas pelo editor. Um exemplo dessa interface inicial é apresentado na figura 28.

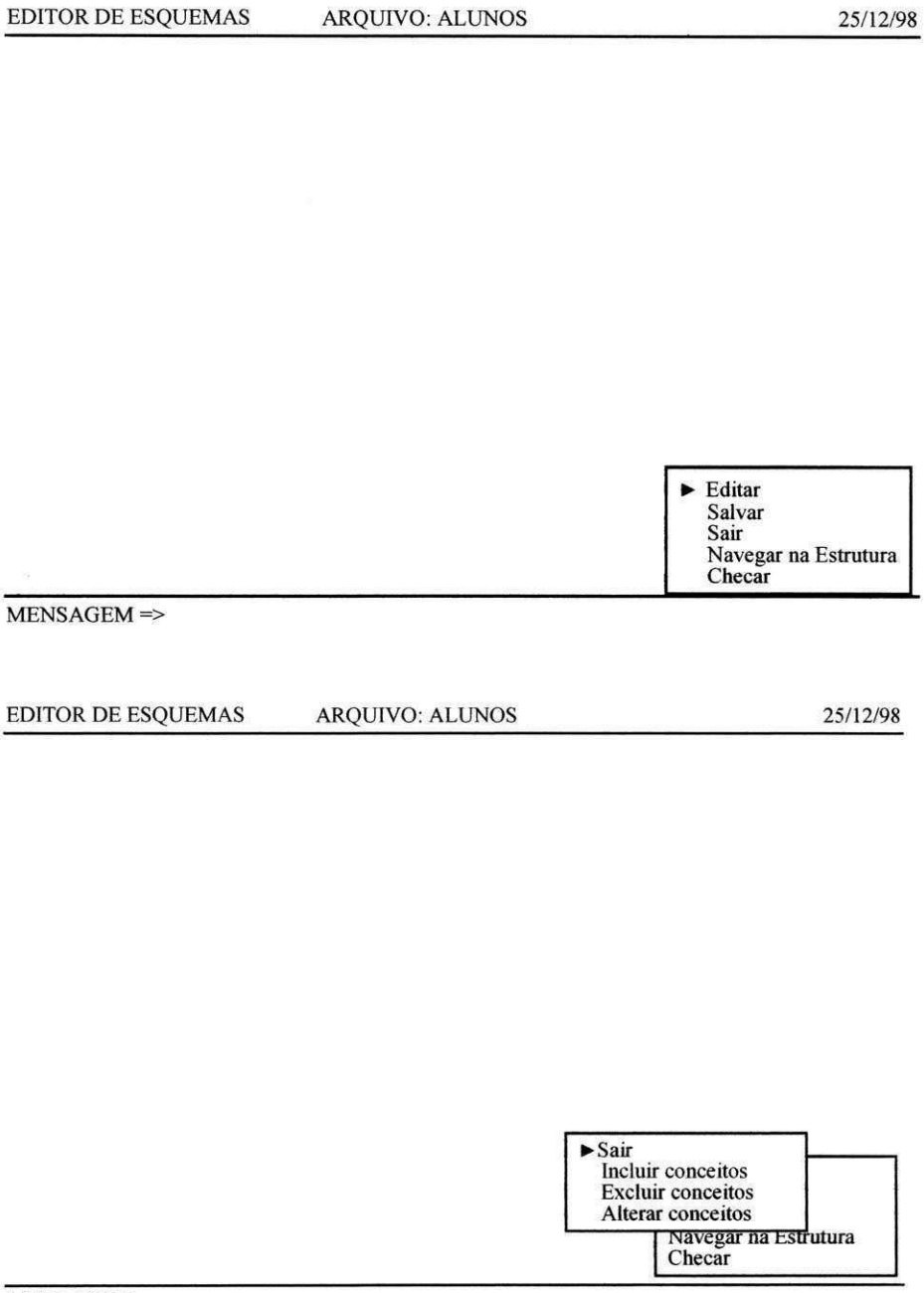

FIGURA 28: MENUS INICIAIS DO EDITOR GRÁFICO DE ESQUEMAS

Para a apresentação do esquema a tela é dividida em zonas, como mostra a figura 29. Cada zona ira conter um conceito (vértice) ou uma ligação entre conceitos (aresta).

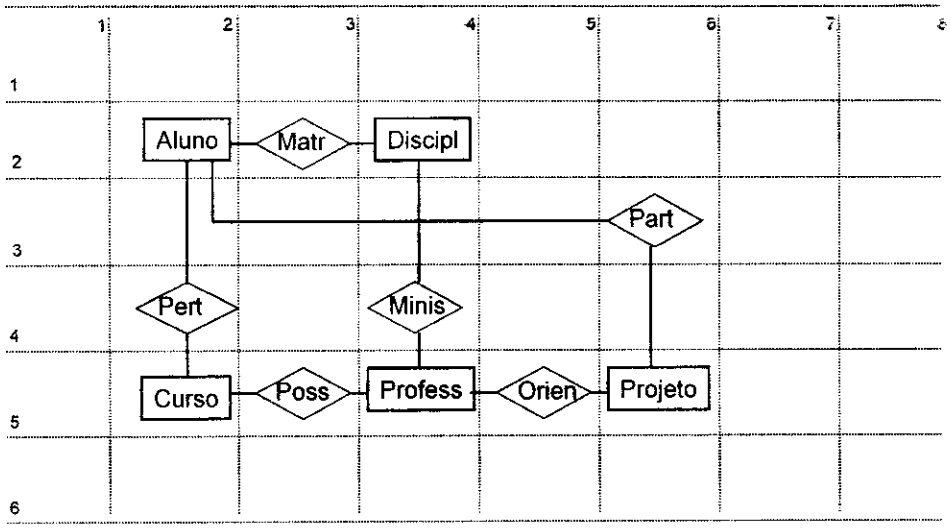

FIGURA 29: ORGANIZAÇÃO DA TELA NO EDITOR GRÁFICO

No modo navegacional o cursor estará posicionado em uma zona e poderá ser movido para zonas contíguas. Em qualquer zona do diagrama que represente um conceito, o usuário poderá solicitar informações mais detalhadas sobre esse conceito. O usuário pode por exemplo conhecer os atributos de uma entidade e após selecionar um atributo dessa entidade, conhecer as suas características. A apresentação da tela para o usuário é mostrada na figura 30.

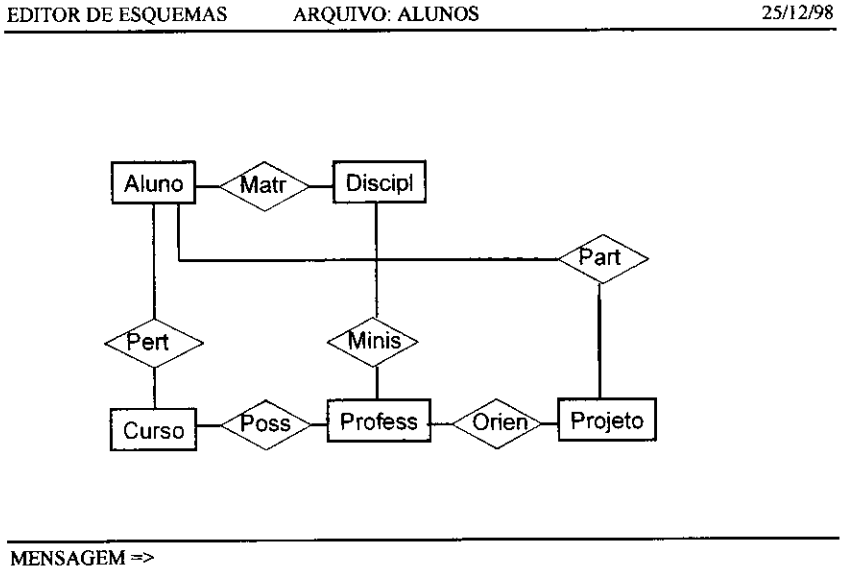

FIGURA 30: DIAGRAMA ECR APRESENTADO PELO EDITOR DE ESQUEMAS.

Para obter um esquema ECR representado graficamente, o editor necessita realizar os seguintes passos:

A) Planarização do esquema conceitual, para obter uma representação com o mínimo de cruzamentos possível;

B) Ortogonalização, a fim de obter um diagrama com o mínimo de ângulos possível;

C) Compactação, para colocar o diagrama no menor numero de zonas e

D) Desenho do diagrama na tela.

Esses passos e um algoritmo para desenhar um diagrama ER são apresentados e discutidos em [TOMASSIA 85].

6.3 - MÓDULO DE INTEGRAÇÃO

É a parte mais importante do sistema e é responsável pela integração dos esquemas. Possui uma Maquina de Inferência responsável pela seleção das regras a serem aplicadas durante o processo de integração, um Módulo de Explanação, que prove explicações ao usuário sobre as decisões tomadas pelo sistema durante o processo de integração. Uma Base de Conhecimento, composta de regras correspondentes ao conhecimento acumulado sobre a atividade de integração de esquemas e uma base de fatos que compreendem as asserções e esquemas informados pelo usuário.

Durante o processo de integração, o sistema fará perguntas ao projetista, a fim de obter informações suficientes para conseguir a integração dos esquemas. Na fase de integração intra-esquema, o projetista será perguntado sobre a existência de sinônimos no esquema analisado e será também consultado sobre os conflitos detectados.

Posteriormente o sistema tenta obter informações necessárias para a integração interesquemas. O usuário será consultado sobre sinônimos, equivalências entre conceitos,

e papeis de atributos e de entidades em relacionamentos. À medida que o sistema consegue informações suficientes, ele promove a integração de conceitos, consultando o projetista para que ele confirme ou não a integração a ser realizada.

Após a conclusão da integração dos esquemas, o projetista poderá examinar as asserções armazenadas pelo sistema na base de fatos. Além disso, poderá também editar o esquema obtido com o uso do editor gráfico de esquemas.

6.3.1 - MAQUINA DE INFERÊNCIA

É responsável pela decisão de como aplicar as regras e em que ordem. Como a *priori* não temos uma meta bem definida (esquema integrado resultante), achamos mais conveniente utilizar como estratégia de controle o Encadeamento Progressivo (*Forward Chaining*) para seleção e processamento das regras do sistema. Se as condições de uma regra são satisfeitas por um conjunto de dados da base de fatos, as ações são executadas. A execução das ações acrescenta novos fatos à base. A Base de fatos modificada ira satisfazer as condições de outras regras, e conseqüentemente, provocar a execução das ações correspondentes. O processo continua até que nenhuma regra tenha as condições satisfeitas.

A primeira tarefa da maquina de inferência é traduzir as regras armazenadas da forma original, de fácil compreensão inclusive para o projetista, para uma forma interpretável pelo Prolog, que foi a linguagem escolhida para a construção da maquina de inferência, retirando-se todas as conjunções e disjunções existentes na regra original. Os predicados Prolog responsáveis por essa tradução fazem parte da maquina de inferência do sistema.

Uma regra que esteja originalmente na forma:

(e,1) # explanação

se condição_1 e condição_2 então ação_1 e ação_2

Será armazenada na base de regras na forma:

(e,1) # explanação

se (condição_1,condição_2) então (ação_1,ação_2)

Permitindo que com um único predicado Prolog as ações sejam executadas se as condições forem satisfeitas para a regra. Na regra acima, (e,1) corresponde a identificação da regra quanto a seu tipo e numero, os conectores #, se e então são definidos como operadores no programa Prolog e explanação corresponde a explicações associadas à regra.

A próxima tarefa da maquina de inferência é selecionar e executar as regras armazenadas na Base de Regras. Isso é feito através de um conjunto de meta-regras. Meta-regras permitem agregar regras de acordo com suas funções dentro do sistema. Assim, ele é capaz de ignorar conjuntos de regras que sejam irrelevantes para o problema em um dado instante. Por exemplo, as regras para integração de atributos só se tornam executáveis quando existem na base de fatos entidades ou relacionamentos já integrados. Isso permite uma melhora significativa no desempenho do sistema.

Inicialmente são executadas todas as regras que envolvem a integração de classes de objetos, até que não existam regras desse tipo com condições satisfeitas. A seguir são examinadas todas as regras para integração de relacionamentos, que irão considerar apenas os relacionamentos envolvendo classes de objetos que sofreram integração na fase anterior. Então são verificadas as regras para integração de atributos, analisando também apenas os atributos de classes de objetos e relacionamentos que sofreram integração. Finalmente são analisadas regras associadas à garantia da consistência das integrações

feitas, como a necessidade de transportar para as novas classes de objetos todos os atributos que não foram integrados mas pertenciam a classes de objetos que tenham sido integradas.

O processo anterior é repetido sucessivamente para avaliar se alguma regra passou a ter suas condições satisfeitas, até que não existam mais regras que possam ser executadas tanto para classes de objetos quanto para os outros tipos de regra.

6.3.2 - MÓDULO DE EXPLANAÇÃO

O sistema incorpora um módulo de explanação, que permite explicar ao usuário como ou porque o sistema apresenta um determinado resultado ou faz uma pergunta. Para tanto, o sistema avalia as regras que levaram a conclusão questionada, para esclarecimento ao usuário. Cada regra possui uma explanação sob a forma de texto, que será informado ao usuário sempre que ele tenha duvidas quanto a ações tomadas pelo sistema. Essa facilidade pode servir também para o treinamento de usuários menos experientes.

6.3.3 - BASE DE CONHECIMENTO

É composta de uma Base de Fatos, inicialmente vazia, e de uma Base de Regras. Os fatos do nosso sistema são os esquemas para integração fornecidos pelo usuário, um conjunto de asserções informadas, como a indicação de sinônimos e funções de mapeamento, além dos fatos deduzidos pelo sistema, como o próprio esquema integrado. As regras são da forma SE Condições ENTÃO Ações e SE Condições ENTÃO Condições. No primeiro caso, um conjunto de ações é executado se as condições são satisfeitas.

Uma dessas ações, por exemplo, seria: Obter função de mapeamento entre objetos. No segundo caso, um conjunto de condições é gerado, podendo eventualmente satisfazer ao antecedente de outras regras.

Devido à natureza do sistema, de manipular objetos, ele pode ser considerado como um sistema híbrido. As regras não manipulam simples predicados e sim estruturas.

Para a representação do conhecimento, nosso sistema utiliza Quadros, Regras de Produção e Representação Lógica (Predicados Prolog). Uma tendência bem sucedida na construção de sistemas especialistas tem sido a integração de quadros e regras de produção para formar sistemas híbridos que combinam as vantagens das duas técnicas de representação do conhecimento [FIKES e KEHLER 85]. Esses sistemas têm mostrado que quadros podem servir como uma base poderosa para as linguagens baseadas em regras. Os quadros fornecem uma rica representação estrutural para descrever objetos referenciados pelas regras. Introduzimos ainda o uso de representação lógica para aumentar o poder de expressão e facilitar a implementação do sistema.

Pela natureza da aplicação e em se tratando de um sistema baseado em conhecimento, a máquina de inferência utilizada na implementação não utiliza incerteza, nem na premissa, nem nas regras.

6.3.3.1 - BASE DE FATOS

Os esquemas manipulados pelo sistema, tanto os esquemas componentes quanto o esquema integrado, são armazenados através de um conjunto de predicados Prolog que implementam a estrutura

de quadros [ARARIBOIA 89]. Quadros permitem uma ligação entre os elementos de sua estrutura, além de incorporar propriedades de herança hierárquica, podendo ser considerados como uma extensão das redes semânticas. Quadros permitem uma organização hierárquica dos dados e a herança de características entre os conceitos. A manipulação de quadros não é visível ao nível de usuário, que tem acesso aos esquemas

através de sua representação gráfica. Podemos incluir, excluir e ver o conteúdo de um quadro através dos predicados:

inclui_quadro(Quadro,Faceta,Operacao);

exclui_quadro(Quadro,Faceta,Operacao);

pega_faceta(Quadro,Nicho_conteudo), onde

Quadro representa o nome do quadro, podendo ser o nome de um relacionamento, atributo ou classe de objetos.

Faceta serve para informar o tipo de conteúdo do nicho (*slot*), se valor ou *default*. Uma faceta tipo valor indica que a informação do nicho é perfeitamente confiável, enquanto que o tipo *default* indica que a informação tem origem em generalizações.

Operação pode representar um simples conteúdo para o nicho, como por exemplo: nome(jose), ou um procedimento que sempre deve ser executado quando o nicho do quadro é acessado, como por exemplo: área(Valor) :- Valor is (Lado * Comprimento). As características dos conceitos são armazenadas como operações. Entretanto, nosso sistema não utiliza procedimentos associados aos conceitos, apesar da implementação permitir.

Os conceitos possuem as seguintes características:

A) Entidades e Categorias: Código de identificação, Nome, Texto associado, Superclasse, Atributos e Chave;

B) Relacionamentos: Código de identificação, Nome, Texto, Superclasse, Grau, Entidades participantes e suas restrições de cardinalidade e Atributos;

C) Atributos: Código, Nome, Texto, Superclasse, Código da entidade ou relacionamento do qual é membro, Cardinalidade máxima e mínima e Tipo de domínio. Os tipos de domínio de um atributo são: Nome (Pessoa, Empresa, Funcionário); Endereço (Rua, Estado, CEP); Valor (Salário, Saldo, Preço);

Identificação (Código de peca, CGC, CPF, Matrícula); Data (Nascimento, Vencimento); Código de tabela (Departamento, Estado), Tipo escalar ((verde, amarelo, azul), (masculino, feminino)); Quantidade (Saldo de estoque, numero de dependentes, idade) e Texto (outros não pré-definidos).

Nicho_conteudo representa o conteúdo de um nicho, que pode ser um dado armazenado ou um dado calculado, no caso de haver uma operação associada ao nicho. Nicho_conteudo devolve ao programa uma característica de um conceito.

Esta implementação explora a característica de herança hierárquica, além de possibilitar a associação de operações aos quadros. Embora a motivação inicial de Minsky ao introduzir o conceito de quadros tenha sido dirigir semanticamente o processo de inferência, muitos trabalhos subseqüentes baseados em quadros têm explorado a representação estrutural dos quadros, ao invés do controle da resolução [FIKES 85].

Os outros componentes da base de fatos do sistema são as asserções. Asserções são indicações feitas pelos usuários acerca dos esquemas. O usuário pode, por exemplo, informar que dois conceitos pertencentes a diferentes esquemas são sinônimos, ou também indicar que existe função de mapeamento entre domínios de um atributo de um esquema sobre o domínio de um atributo de outro esquema. Esse tipo de informação é armazenada pelo sistema através de Predicados Prolog [STERLING 86]. Se o usuário informar, por exemplo, que dois conceitos Funcionário e Empregado são sinônimos, e que existe um mapeamento entre o domínio das chaves de Funcionário e Empregado, onde cada elemento de Funcionário possui um correspondente em Empregado, o sistema armazenara as informações na forma:

sinônimo(funcionário,empregado);

f_map(empregado,funcionário,beta_equal),

As indicações de funções de mapeamento e sinônimos são feitas pelo usuário durante a fase de especificação de asserções. São armazenadas pelo sistema em uma tabela Hash a fim de otimizar o processo de busca, quando da verificação das asserções, no processo de integração dos esquemas. O predicado guarde pertencente a maquina de inferência é o responsável pela inclusão das asserções na base de fatos.

6.3.3.2 - BASE DE REGRAS

Regras de produção são utilizadas para representar o conhecimento necessário a integração de esquemas. Esse conhecimento foi extraído basicamente das metodologias para integração de esquemas pesquisadas. Durante a vida útil do sistema, o usuário poderá também acrescentar novas regras ou retirá-las, de modo a incorporar a sua experiência a base de conhecimento, incorporar conhecimento proveniente de novas metodologias desenvolvidas ou adequar o comportamento do sistema as suas necessidades mais especificas. Deste modo, buscamos apresentar um sistema bastante flexível e evolutivo, que pode incorporar conhecimento com o uso. Entretanto, a alteração da base de regras não é uma tarefa simples e manipulações inadequadas podem comprometer o desempenho do programa, ou até levar a resultados indesejáveis e incorretos.

A título de exemplo apresentamos uma das regras utilizadas pelo sistema. Essa regra refere-se à integração de classes de objetos e é executada sempre que o sistema concluir que a integração de duas classes de objetos nos esquemas componentes deve produzir uma única classe de objetos no esquema resultante.

(e,5)

$Foi informado ao sistema que existe uma equivalência total entre os domínios das duas entidades. Isso resulta na integração em uma única entidade com o esquema comum. Se não concordar, retire a informação de mapeamento informada anteriormente$.

#

se

retrieveh(asserções,f_map,f_map(Entidade_X,Entidade_Y,Resp))

e não retrieveh(asserções,integrou,integrou(Entidade_X, Entidade_Y)) e

Resp == 1 { Mapeamento de domínios iguais }

então

pega_faceta(Entidade_X,chave([Chave_1])) e

pega_faceta(Entidade_Y,chave([Chave_2])) e

transforma_nome([Entidade_X,'_',Entidade_Y],Nome_nova_entidade)

e pega_faceta(Entidade_X,texto(T)) e

guarde integrou(Entidade_X,Entidade_Y) e

[! menu([nl,'As entidades ',Entidade_X,' e ',Entidade_Y,

' serão integradas gerando a entidade ',Nome_nova_entidade,

nl],['Concorda','Discorda','Deseja saber porque'],Resp1) !] e

analisa_resposta(Resp1,(e,5)) e

guarde nova_entidade(Nome_nova_entidade) e

guarde f_map(Nome_nova_entidade,Chave_1,Chave_2,1) e

guarde integrar_atributos(Nome_nova_entidade,Entidade_X,Entidade_Y) e

inclui_quadro(Nome_nova_entidade,valor,texto(T)) e

inclui_quadro(Nome_nova_entidade,valor,superclasse(entidades_3))

e liga_quadros(Nome_nova_entidade,entidades_3)

Os termos com letras maiúsculas (Ex. Resp1) são variáveis Prolog e os termos em minúsculo (Ex. entidades_3) são valores instanciados (constantes). O texto compreendido entre dólar ($) corresponde à explanação associada a essa regra.

A base de regras foi organizada em 5 tipos principais de regras:

A) Meta-regras, que visam reduzir o universo de perguntas e melhorar o desempenho do sistema, procurando evitar que se façam perguntas desnecessárias ou incoerentes;

B) Regras para integração de classes de objetos;

C) Regras para integração de atributos;

D) Regras para integração de relacionamentos e

E) Regras para melhoramento do esquema integrado, que visam retirar ciclos ou redundâncias incorporadas durante processo de integração.

6.4 - Editor de Regras

Possibilita ao usuário mais experiente incluir novas regras no sistema ou excluir regras existentes. Na inclusão, o editor checa se a nova regra esta sintaticamente correta, não verificando entretanto a sua consistência semântica. O usuário pode criar novas regras ou alterar regras existentes com a inclusão ou exclusão de clausulas entre as evidencias ou hipóteses da regra. Poderá também alterar a explanação associada à regra.

6.5 - Metodologia de Programação Utilizada

Utilizamos as idéias de projeto modular e estruturado como metodologia para desenvolvimento do sistema. Os diversos componentes foram identificados e as estruturas de dados foram projetadas e testadas quanto à capacidade de modelar os dados do sistema. Os algoritmos principais foram desenvolvidos e cada módulo foi sendo implementado e testado progressivamente. Isso possibilitou um maior controle do domínio do problema, apesar de sua dimensão. Uma documentação das regras que foram utilizadas também foi mantida durante o desenvolvimento, permitindo um acompanhamento do comportamento do sistema de forma simples

evitando a criação de regras redundantes.

CAPÍTULO 7

CONCLUSÕES E FUTUROS TRABALHOS

O trabalho segue uma tendência atual, ao utilizar uma abordagem de sistemas baseados em conhecimento para a solução do problema da integração de esquemas [BOUZEGHOUB 85] [DOGAC 89] [LARSON 89]. Além disso, procura extrair aspectos de diversas metodologias, classificando e utilizando o conhecimento documentado, de forma a apresentar a melhor solução para a integração.

É um sistema híbrido que utiliza as vantagens de diversas formas de representação de conhecimento pesquisadas no campo da Inteligência Artificial, e evolutivo, na medida em que permite a alteração de sua base de regras, possibilitando melhorias ou mesmo reformulação do sistema, sem que se tenha que alterar a lógica do programa principal.

Procuramos ainda apresentar uma interface de trabalho a mais amigável possível, já que a tarefa de integrar esquemas é normalmente longa e cansativa, exigindo do projetista de BD um tempo considerável na análise das possibilidades de integração dos esquemas componentes.

O sistema conta atualmente com os módulos de integração e o Editor de Regras concluídos. A base de regras possui conhecimento suficiente para integrar de forma consistente dois esquemas ECR, inclusive com o tratamento de hierarquias de generalização. O módulo de explanação encontra-se parcialmente implementado, permitindo atualmente explanação apenas sobre a regra que esta sendo executada.

O código fonte do sistema compreende no momento 1.800 linhas correspondentes a 80 regras de produção. 1500 linhas em linguagem C relativas a implementação do editor de esquemas. Além de 450 linhas de código em linguagem Prolog relativas a maquina de

inferência, módulo de explanação e editor de regras. O código completo da aplicação está apresentada nos anexos a este trabalho.

Apesar de já termos definido todos os módulos do Editor Gráfico de Esquemas e identificado as atividades envolvidas na obtenção de uma representação gráfica para um esquema, o editor ainda não se encontra totalmente implementado. Além disso, o Módulo de Explanação pode ainda sofrer melhorias. Dessa maneira, concluímos que a interface final para o usuário fica como uma extensão ao nosso trabalho.

A definição das regras em uma linguagem mais próxima da linguagem natural também é desejável, para facilitar o trabalho de projetistas que desejem manipular a Base de Regras. No momento, as regras são escritas numa forma muito próxima da linguagem Prolog.

O sistema desenvolvido pode ainda ser estendido para possibilitar a integração de BDs já existentes. Para isso é necessário o tratamento de esquemas de operações e mapeamento de consultas dos esquemas componentes para o esquema integrado. Essa é uma ampliação desejável do nosso trabalho, porém é uma tarefa bastante complexa.

CAPÍTULO 8

REFERÊNCIAS BIBLIOGRÁFICAS

[AL-FEDAGHI 81] S.Al-Fedaghi e P.Scheuermann, *Mapping Considerations in the Design of Schemas for the Relational Model*, IEEE Trans. Softw. Eng., Vol. SE-7, No 1, Jan, Pags. 99-111.

[ARARIBOIA 89] G. Arariboia, *'Inteligência Artificial* - Um curso Pratico', Editora LTC, Rio de Janeiro.

[BATINI 85] C.Batini, L.Furlani e E.Nardelli, *What is a good Diagram ? A pragmatic approach*, Proc. of the 4th Intern. Conf. on the Entity Relationship Approach, IEEE Computer Society, Silver Spring, Chicago, Pag. 312-319.

[BATINI 86] C.Batini, M.Lenzerini e S.Navathe, *A Comparative Analysis of Methodologies for Databases Schema Integration*, ACM Computing Surveys, Vol. 18, No 4, Dez, Pags. 323-364.

[BATINI 84] C.Batini e M.Lenzerini, *A Methodology for Data Schema Integration in the Entity-Relationship Model*, IEEE Trans. Softw. Eng., Vol. SE-10, No 6, Nov, Pags. 650-664.

[BOUZEGHOUB 85] M.Bouzeghoub, G.Gardarin e E.Metais, *Database Design Tools: An expert system Approach*, Proc. of VLDB, Stockholm, Pags. 82-95.

[CASANOVA 83] M.A.Casanova e V.M.P.Vidal, *Towards a Sound Database View Integration Methodology* Relatório Técnico No 007, IBM Brasil - Centro Cientifico, Brasília (DF).

[CERI 85] Editado por Stefano Ceri, *Methodology and Tools for Data Bases Design*, North-Holland Publishing Company, Amsterdam.

[CHEN 76] P.P.Chen, *The Entity-Relationship Model - Towards a Unified View of Data*, ACM-TODS, Vol. 1, No 1, Mar, Pags. 9-36.

[CHOOBINEH 88] J.Choobineh, M.Mannino, J.F.Nunamaker,Jr e B.Konsynski, *An expert database design system based on analysis of forms*, IEEE Trans. Softw. Eng., Vol. 14, No 2, Feb, Pags. 242-253.

[DATE 91] C.J.Date, *Introdução a Sistemas de Bancos de Dados*, Ed. Campus, Rio de Janeiro.

[DAYAL 84] U.Dayal e H.Hwang, *View Definition and Generalization for Database Integration in a Multidatabase System*, IEEE Trans. Softw. Eng., Vol. SE-10, No 6, Nov, Pags. 628-644.

[DOGAC 89] A.Dogac, B.Yuruten e S. Spaccapietra, *A generalized expert system for database design*, IEEE Trans. Softw. Eng., Vol. 15, No 4, Apr, Pags. 479-491.

[ELMASRI 85] R.Elmasri, A.Henver e J. WeelDreyer, *The Category Concept: An Extension to the Entity-Relationship Model*, Data and Knowledge Engineering, Vol. 1, No 1, Jun, Pags. 75-116.

[FIKES 85] R.Fikes e T.Kehler, *The role of frame-based representation in reasoning*, Com. of the ACM, Vol.28, No 9, Pags. 904-920.

[HAYES-ROTH 85] F.Hayes-Roth, *Rule-based systems*, Com. of the ACM, Vol.28, No 9, Pags. 921-932.

[KORTH 89] H.F.Korth e A.Silberschatz, *Sistemas de Bancos de Dados*, Ed. McGraw-Hill, São Paulo.

[LARSON 89] J.Larson, S.Navathe e R.Elmasri, *A Theory of Atribute Equivalence in Databases with Application to Schema Integration*, IEEE Trans. Softw. Eng., Vol. 15, No 4, Apr, Pags. 449- 463.

[MAIER 83] D.Maier, *The Theory of Relational Databases*, Computer Science Press, Rockville, Maryland.

[MONGIOVI 90] G.Mongiovi, I.D.Moreira, M.C.Sampaio, *Um Sistema Baseado em Regras para Integração de Esquemas de Banco de Dados*, Anais do Premier Simpósio de Inteligência Artificial Y Robótica, SIAR'90, Lujan - Argentina, Nov.

[MOREIRA 90] I.D.Moreira, M.C.Sampaio, G.Mongiovi, *Um Sistema Inteligente para Integração de Esquemas de Banco de Dados*, Anais do 5o Simpósio Brasileiro de Banco de Dados, Rio de Janeiro (RJ), Abril, Pags. 58-72.

[NAVATHE 86] S.Navathe, R.Elmasri e J.Larson, *Integration User Views in Database Design,* IEEE-Computer, Vol. 19, No 1, Jan, Pags. 50-62.

[PECKHAM 88] J.Peckham e F.Maryanski, *Semantic Data Models*, ACM Computing Surveys, Vol. 20, No 3, Sep, Pag. 153-189.

[SMITH 77] J.M.Smith e D.C.P.Smith, *Database abstractions: Aggregation and generalization,* Acm Trans. Database Systems, Vol. 2, No 2, Mar, Pag. 105-133.

[STERLING 86] L.Sterling e E.Shapiro, *The Art of Prolog*, The MIT Press, Cambridge, Massachusetts.

[TOMASSIA 85] R.Tomassia, *New layout techiniques for entity- relationship diagrams*, Proc. of the 4th Intern. Conf. on the Entity Relationship Approach, IEEE Computer Society, Silver Spring, Chicago, Pag. 304-311.

[WATERMAN 86] D.A.Waterman, *A guide to Expert Systems*, Addison-Wesley.

[WIEDERHOLD 83] G.Wiederhold, *Database Design*, Second Edition, McGraw-Hill, New York.

[YAO 82] S.B.Yao, V.E.Waddle e B.C.Housel, *View Modeling and Integration Using the Functional Data Model*, IEEE Trans. Softw. Eng., Vol. SE-8, No 6, Nov., Pags. 544-553.

ANEXO 1

EXEMPLO DA INTEGRAÇÃO DE DOIS ESQUEMAS

Neste capítulo procuraremos acompanhar o processo de integração dos dois esquemas mostrados na figura 31 a seguir. Tentaremos principalmente demonstrar a forma como o sistema se comporta e a representação interna dos dados informados pelo projetista e manipulados pelo sistema.

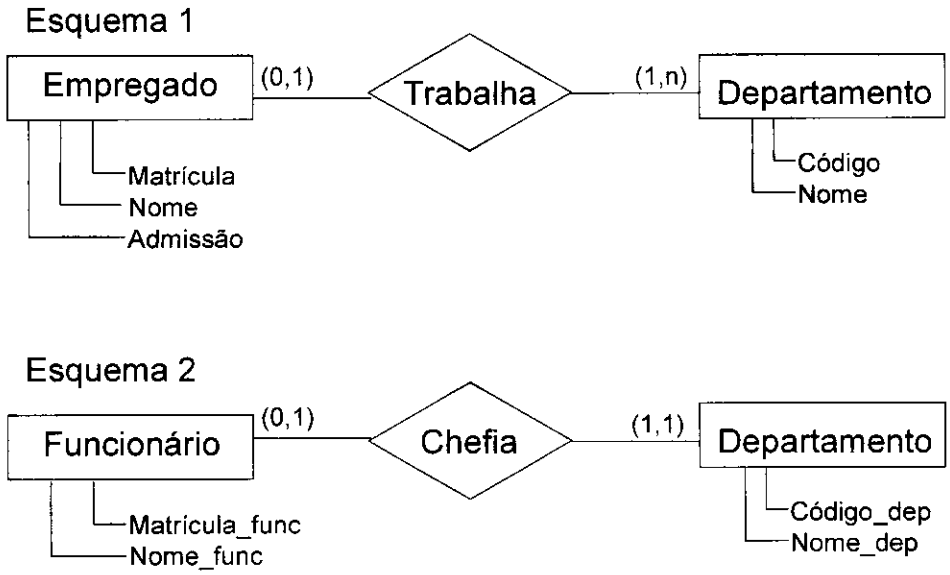

FIGURA 31: ESQUEMAS A INTEGRAR

A entrada dos dados para o sistema é feita através do editor gráfico de esquemas. A representação de alguns conceitos pertencentes ao esquema_1 na forma de quadros e a rede semântica correspondente a esses quadros são mostrados nas figuras 32 e 33 respectivamente.

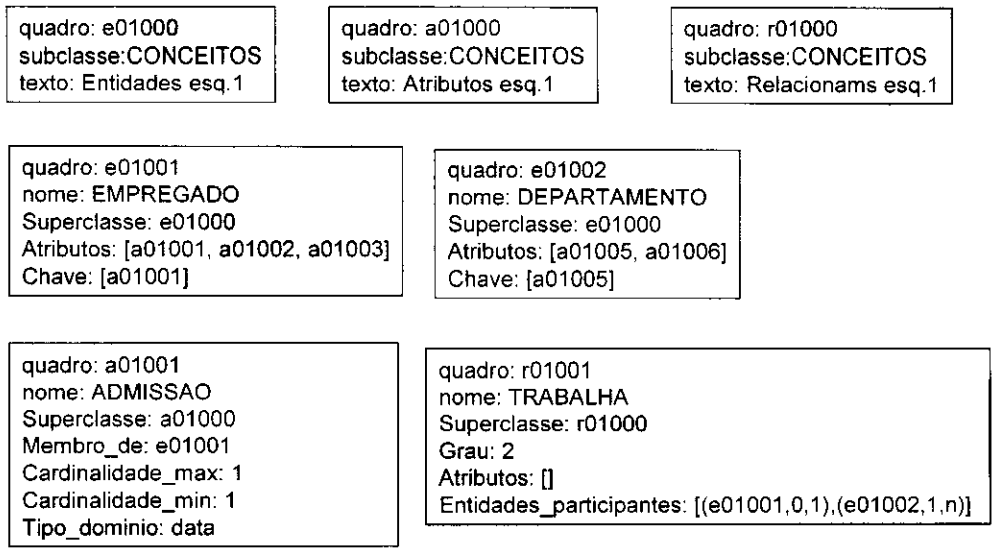

FIGURA 32: REPRESENTAÇÃO DE CONCEITOS NA FORMA DE QUADROS

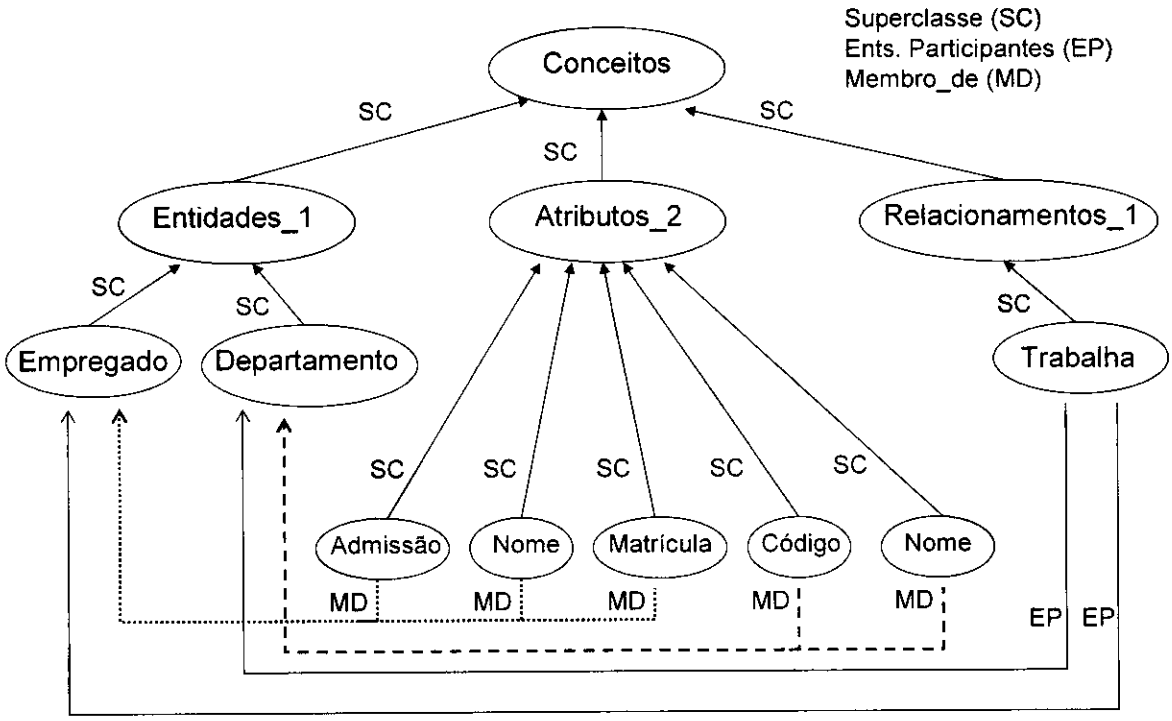

FIGURA 33: REDE SEMÂNTICA CORRESPONDENTE AOS QUADROS DO ESQUEMA 1

O sistema armazena os dados na forma:

quadro(e01001,valor,superclasse(e01000);

quadro(e01001,valor,nome(Empregado);

quadro(e01001,valor,atributos([a01001,a01002,a01003]));

quadro(e01001,valor,chave([a01001]));

quadro(r01001,valor,superclasse(r01000);

quadro(r01001,valor,nome(Trabalha);

quadro(r01001,valor,grau(2));

quadro(r01001,valor,atributos([]);

quadro(r01001,valor,entidades_participantes([(e01001,0,1),e01002,1,n)]).

Durante a fase de especificação das asserções, o usuário devera informar os sinônimos existentes, além dos mapeamentos entre os domínios dos conceitos dos dois esquemas. O sistema pergunta se existe mapeamento entre conceitos pertencentes aos dois esquemas, avaliando primeiro os conceitos sinônimos e também os que possuem tipos de domínio compatíveis, evitando perguntar se existe mapeamento entre um conceito cujo domínio seja, por exemplo, um endereço e outro cujo tipo de domínio seja uma data. Vamos considerar que o usuário indicou as asserções a seguir, referentes aos esquemas da figura 31:

"Os conceitos Empregado e Funcionário são sinônimos. Existe uma função de mapeamento entre os domínios de Empregado e Funcionário, onde todo elemento de Empregado possui um correspondente em Funcionário. Todo elemento existente em Departamento esta também presente em Depart. Nem todo Empregado que Trabalha em Departamento é um Funcionário que Chefia Depart, porém todo Funcionário que Chefia um Depart é um Empregado que Trabalha em um Departamento". As asserções são informadas pelo usuário à medida que o sistema o consulta. Para informar por exemplo o tipo de equivalência entre domínios de dois conceitos, o sistema apresenta as alternativas possíveis (Domínios iguais, contidos, superpostos ou disjuntos) para que o usuário escolha uma delas. Essas asserções serão armazenadas internamente no sistema na forma:

sinônimo(empregado,funcionário),

f_map(empregado,funcionário,1, (equivalência β *equal*)

f_map(trabalha,chefia,2, (equiv. β *contains*)

f_map(departamento,depart,1) (equiv. β *equal*).

Na fase de integração dos esquemas, os esquemas são integrados baseado nessas asserções produzindo como resultado:

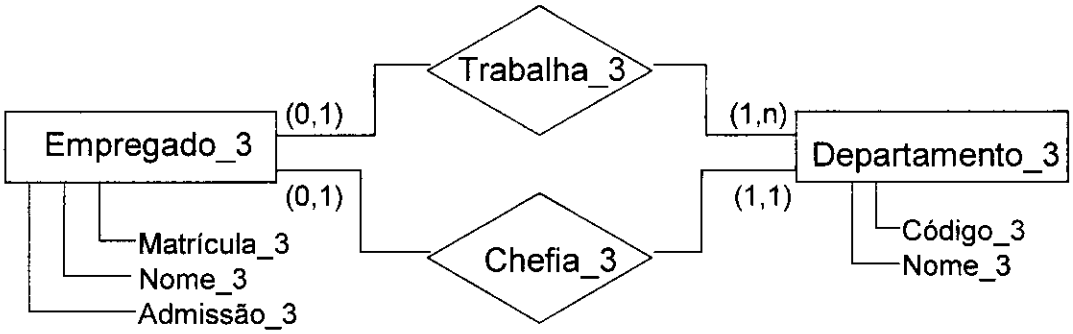

FIGURA 34: ESQUEMA INTEGRADO

Alguns dos quadros correspondentes ao esquema resultante da integração dos esquemas mostrados na figura 31 são os mostrados na Figura 35 a seguir.

quadro: e03000
subclasse:CONCEITOS
texto: Entidades esq.

quadro: a03000
subclasse:CONCEITOS
texto: Atributos esq.

quadro: r03000
subclasse:CONCEITOS
texto: Relacionams esq.

quadro: e03001
nome: EMPREGADO_3
Superclasse: e03000
Atributos: [a03001, a03002, a03003]
Chave: [a03001]

quadro: e03002
nome: DEPARTAMENTO_3
Superclasse: e03000
Atributos: [a03005, a03006]
Chave: [a03005]

quadro: a03002
nome: ADMISSAO_3
Superclasse: a03000
Membro_de: e03001
Cardinalidade_max: 1
Cardinalidade_min: 1
Tipo_dominio: data

quadro: r03001
nome: TRABALHA_3
Superclasse: r03000
Grau: 2
Atributos: []
Entidades_participantes: [(e03001,0,1),(e03002,1,1)]

FIGURA 35: QUADROS CORRESPONDENTES A CONCEITOS DA FIGURA 34

ANEXO 2

MÁQUINA DE INFERÊNCIA (PROLOG)

A aplicação foi desenvolvida no ambiente Arity-Prolog. Todo o código está baseado em Prolog Padrão e está testado e funcional.

```
* Motor de Inferencia  *
:-op(950,xfy,#).
:-op(900,fx,se).
:-op(600,xfy,':').
:-op(800,xfx,entao).
:-op(750,xfy,e).
:-op(750,xfy,ou).
:-op(350,fx,nao).
:-op(300,fx,assercao).
:-op(300,fx,guarde).
:-op(300,fx,escreva).
:-op(300,fx,retire).

trad((N # se A entao B),(N # se Al entao Bl)) :-
        retire_rotulos(A,Al),retire_rotulos(B,Bl).

retire_rotulos(A e B,(A1,B1)) :- !,
        retire_rotulos(A,Al),retire_rotulos(B,Bl).
retire rotulos(A ou B,(A1;B1) :- !,
        retire_rotulos(A,Al),retire_rotulos(B,Bl).
retire_rotulos(A,A).

consulte(Arq) :- see(Arq),
            repeat,
            read(X), armazene(X), !, seen.

armazene(end_of_file) :- !.
armazene(A) :- trad(A,B), !, assertz(B), fail.
armazene(A) :- write('Erro de gramatica' : A), nl.

reconsulte(Arq) :-
abolish((#),1), fail.
reconsulte(Arq) :- consulte(Arq).

guarde(F) :- F =.. [Chave|Algo],
retrieveh(assercoes,Chave,F), !. guarde(F) :- F =..
[Chave|Algo], modificou, !, assertz(fato(F)),
      recordh(assercoes,Chave,F).
guarde(F) :- F =.. [Chave|Algo], assertz(fato(F)),
      recordh(assercoes,Chave,F), assertz(modificou).

assercao(F) :- F =.. (Chave|Parametros],
```

```
retrieveh(assercoes,Chave,F).

inicialize :-
limpe_memoria, fail,
inicialize :-
assertz(modificou).

limpe memoria :- retract(fato(X)),
retract(X), fail. limpe_memoria :-
modificou, retract(modificou), fail.
limpe_memoria.

execute :- retract(modificou),
      integra_entidades,
      integra_relacionamentos,
      integra_atributos,

   integra_restante,
   !, execute,
execute.

integra entidades :- (e,N) # se Conds entao Acoes,
            call(Conds), call(Acoes), modificou, !,
            integra_entidades.
integra_entidades.

integra_relacionamentos :- (r,N) # se Conds entao Acoes,
                call(Conds), call(Acoes), modificou, !,
                integra_relacionamentos.
integra relacionamentos.

integra_atributos :- (a,N) # se Conds entao Acoes,
            call(Conds), call(Acoes), modificou, !,
            integra_atributos.
integra_atributos.

integra_restante :- (x,N) # se Conds entao Acoes,
           call(Conds), call(Acoes), modificou,
           !, integra_restante.
integra_restante.

liste :- fato(X), X =.. [Cabeca|Algo], Cabeca == quadro,
     write(X), nl, fail.
liste :- fato((X :- Y)), write(X), nl, fail,
liste.
liste(F) :- call(F), write(F), nl, fail.
liste(_).

menu(Pergunta,Opcs,Escolha) :-
       escreva(Pergunta),nl,
       write(Escolha uma das opcoes abaixo:'),
       nl,nl,
        escreva(1,Opcs),
        read(Escolha),nl.
```

```
/*
        read(N),
        indice(1,N,Opcs,Escolha).
*/

menu(Pergunta,Opcs,Escolha) :- write('Escolha invalida'),
        nl, menu(Pergunta,Opcs,Escolha).
escreva([]) :- !.
escreva(['?']) :- write('?'), nl, !.
escreva(['.']) :- write('.'), nl, !.
escreva([nl|X]) :- nl, escreva(X).
escreva ([X|Y]} :- write(X), write(' '), escreva (Y).
escreva(_,[]) :- !, nl.
escreva(J,[X|Y]) :- write(J:X), nl, K is J+l, escreva(K,Y).
 indice(J,J,[X|Y],X) :- !,
 indice(J,M,[_|Y],X) :- K is J+1, indice(K,M,Y,X).
 le_atomo(Atm) :- get0(X),
 le_teclas(X,[],Teclas),
            inverte(Teclas,[],ITeclas),
            name(Atm,ITeclas),nl.
 le_teclas(X,Tcs,Tcs)  :- [X] = ".", !.
 le_teclas(13,Tcs,Tcs) :- !.
 le_teclas(8,Tcs1,Tcs) :- !, apague_uma_tecla(Tcsl,Tcs2),
                get0(X), le_teclas(X,Tcs2,Tcs).
 le_teclas(X,Tcsl,Tcs) :- get0(Y),
 le_teclas(Y,[X|Tcsl],Tcs).

 apague_uma_tecla([],[]) :- !.
 apague_uma_tecla([X|Y],Y) :- write(' '),
 put(8),
 inverte([],INV,INV) :- !.
 inverte([32,32|X],ParmAcm,INV) :- !, inverte([32|X3,
 ParmAcm,INV).
 inverted[X|Y],ParmAcm,INV) :- inverte(Y,[X|ParmAcm],INV).
 nao X :- not(X).

 exec :- inicialize, nl, execute.

 membro(X,[X|Xs]).
 membro(X,[Y|Ys]) :-
 membro(X,Ys).

 transforma_nome(Atomo,Atomo_sai) :- concat(Atomo,Str),
                        atom_string(Atomo_sai,Str).

 /* conta_entidade, conta_relacionamento e conta_atributo
   acumulam o contador de conceitos incluidos no esquema resultante para
   identificacao. O acumulador comeca com 1.          */

 conta_entidade(X) :- retract(acum_entidade(Z)),X is Z + 1,
              assertz(acum_entidade(X)).
 conta entidade(X) :- not call(acum_entidade(_)), X is 1,
               assertz(acum_entidade(X)).
 conta relacionamento(X) :- retract(acum_relacionamento(Z)),X is
```

```
            Z + 1, assertz(acum_relacionamento(X)).
conta_relacionamento(X) :- not call(acum_relacionamento(_)), X
            is 1, assertz(acum_relacionamento(X)).
Conta_atributo(X) :- retract(acum_atributo(Z)),X is Z + 1,
           assertz(acum_atributo(X)).
conta_atributo(X) :- not call(acum_atributo(_)), X
           is 1, assertz(acum_atributo(X)).
resposta(Resp) :- obtem_sim_ou_nao(Resp).

obtem_sim_ou_nao(Resp) :- repeat,
                write('Responda sim ou nao ==> '),read(Resp),
          (Resp == sim; Resp == nao) ,!.

sinonimos(A,B) :- sinonimo(A,B), !.
sinonimos(A,B) :-  sinonimo(B,A).
sinonimos(A,C) :- sinonimo(A,B), sinonimos(B,C),
sinonimos(A,C) :- sinonimo(C,B), sinonimos(B,A).
```

Tratamento de Frames

```
/* 0 predicado ponha coloca uma Regra no Frame. 0 funtor
  da cabeca e o nicho e 0s argumentos formam o conteudo.
  A cabeca, portanto, fornece tanto o nicho quanto o
  conteudo.
  0 corpo da regra, quando existir, permite calcular o
  conteudo.
  A faceta e dada pelo segundo argumento de inclui quadro
  */
/************************

inclui_quadro anterior
inclui_quadro(Frame,Faceta,Regra) :- atom(Faceta),
                decapita(Regra,Nicho_Conteudo,Corpo),
                Frame =.. [Fn|Args],
                Cbl = .. [quadro,Fn,Faceta,Nicho_Conteudo|Args],
                 \+(clause(Cbl,Corpo)), assertz((Cbl :- Corpo)).
 ************************/

inclui_quadro(Frame,Faceta,Regra) :- atom(Faceta),
                decapita(Regra,Nicho_Conteudo,Corpo),
                Frame =.. [Fn|Args],
                Cbl =..
                [quadro,Fn,Faceta,Nicho_Conteudo|Args],
                 \+(clause(Cbl,Corpo)), assertz((Cbl :-
                 Corpo)), assertz(fato((Cbl :- Corpo))).

/* Remova serve para remover a regra colocada com ponha */

exclui_quadro(Frame,Faceta,Regra) :- atom(Faceta),
                 decapita(Regra,Nicho_Conteudo,Corpo),
                 Frame =.. [Fn|Args],
                 Cb1 =.. [quadro,Fn,Faceta,Nicho_Conteudo|Args],
```

```
                retract((Cb1 :- Corpo)),retract(fato((Cbl :- Corpo))), !.

exclui_quadro(Objeto,Faceta,Conteudo).

/* liga quadros une dois frames por um elo do tipo ako,  */

liga_quadros(Frame1,Frame2) :- Frame1 =.. [Fn1|Args1],
            Frame2 =.. [Fn2|Args2],
            L =.. [quadro,Fnl,Argsl,ligado_com,Fn2,Args2],
            assertz(L),assertz(fato(L)).

decapita((C :- B),C,B) :- !.
decapita(C,C,true).

/*Q funtor \+ significa nao (not). */

verifique(P) :- \+(\+(clause(P,B))).

/* pega_faceta recupera o conteudo de um nicho. 0
  primeiro argumento e o Frame. 0 segundo
  argumento e uma estrutura cujo funtor e o nicho
  e cujo argumento e o conteudo     */

pega_faceta(Frame,Nicho_Conteudo) :- Frame =,. [Fn|Args],
      corrente(quadro,Fn,Args,Nicho_Conteudo,Consulta),call(Consulta).

/* Em prime!to lugar, verifies se o Nicho_Conteudo pode ser
  obtido na faceta valor. Se puder, o problema esta resolvido     */

corrente(quadro,Fn,Args,Nicho_Conteudo,Consulta) :-
            Consulta =.. [quadro,Fn,valor,Nicho_Conteudo|Args],
            verifique(Consulta), !.

/* Se a faceta valor nao existe, procure na faceta default  */

corrente(quadro',Fn,Args.,Nicho_Conteudo,Consulta) :-
            Consulta =.. [quadro,Fn,default,Nicho_Conteudo|Args],
            verifique(Consulta), !.

/* Se a faceta default falhar, siga o elo do tipo ako  */

corrente(quadro,Fn,Args,Nicho_Conteudo,Consults) :-
            L =..[quadro,Fn,Args,ligado_com,Fnl,Argsl],call(L),
            corrente(quadro,Fnl,Argsl,Nicho_Conteudo,Consulta).
```

ANEXO 3

SISTEMA BASEADO EM CONHECIMENTO PARA INTEGRAÇÃO (REGRAS DE PRODUÇÃO)

Integridade de Esquemas

```
(x,01) #
se
  assercao nova_entidade(Ent) e
  nao assercao colocou_atributos(Ent)
entao
  guarde colocou_atributos(Ent) e
  setof(X,(assercao
  novo_atributo(X),pega_faceta(X,membro_de(Ent))),Y) e
  inclui_quadro(Ent,valor,atributos(Y)).
```

Integração de Entidades

```
(e,01) #
Se
   assercao integrar(Entidade_X) e
   entidade(Entidade_X)
entao
   retract(entidade(Entidade_X)).

(e,02) #
se
  entidade(Entidade_X) e
  entidade(Entidade_Y) e
  Entidade_X \== Entidade_Y e
  pega_faceta(Entidade_X,superclasse(entidades_1}) e
  pega_faceta(Entidade_Y,superclasse(entidades_2)) e
  nao assercao perguntou(Entidade_X,Entidade_Y) e
  nao assercao integrar(Entidade_X) e
  nao assercao integrar(Entidade_Y) e
  pega_faceta(Entidade_X,nome(Nome_X)) e
  pega_faceta(Entidade_Y,nome(Nome_Y)) e
  Nome_X == Nome_Y
entao
  pega_faceta(Entidade_X,chave([Chave_1])) e
  pega faceta(Entidade_Y,chave([Chave_2])) e
  pega_faceta(Chave_1,nome(Nome_1)) e
  pega faceta(Chave_2,nome(Nome_2)) e
  guarde perguntou(Entidade_X,Entidade_Y) e
  [! cls,escreva [nl,Nome_X,' no esquema 1 e ',Nome_Y,
        'no esquema 2 sao homonimos ',nl,
        'Existe funcao de mapeamento entre ',Nome_1,
        de ',
        Nome_X,nl,'                    e ',
```

```
        Nome_2,' de ', Nome_Y,' ?',nl] !] e
  resposta(Resp1) e
  Resp1 == sim e
  [! menu([nl,'Como se classificam os dominios de  A ->
  ',Nome_1,nl, B -)  ,Nome_2,' ?',nl],
  ['Os dominios de A e B sao exataaente iguais',
  'O dominio de A contem tctalmente o de  B',
  'O dominio de  A esta totalmente contido no de  B',
  'Os dominios de A e B possuem uma intersecao',
  'Apesar dos dominios disjuntos, A e B possuem o mesmo
  papel'],Resp2) !] e
  guarde f_map(Entidade_X,Entidade_Y,Resp2) e
  guarde integrar(Entidade_X) e
  guarde integrar(Entidade_Y).

(e,03) #
se
  entidade(Entidade_X) e
  entidade(Entidade_Y) e
  Entidade_X \== Entidade_Y e
  pega_faceta(Entidade_X,superclasse(entidades_1)) e
  pega_faceta(Entidade_Y,superclasse(entidades_2)) e
  nao assercao perguntcu(Entidade_X,Entidade_Y) e
  nao assercao integrar(Entidade_X) e
  nao assercao integrar(Entidade_Y) e
  sinonimos(Entidade_X,Entidade_Y)
entao
  pega_faceta(Entidade_X,nome(Nome_X)) e
  pega_faceta(Entidade_Y,nome(Nome_Y)) e
  pega_faceta(Entidade_X,chave([Chave_1])) e
  pega faceta(Entidade_Y,chave([Chave_2])) e
  pega_faceta(Chave_1,nome(Nome_1)} e
  pega_faceta(Chave_2,nome(Nome_2)) e
  guarde perguntou(Entidade_X,Entidade_Y) e
  [! cls,escreva  [nl,Nome_X,' e  ',Nome_Y,'  sao
        sinonimos ',nl,'Existe  funcao  de  mapeamento  entre
        ,Nome_1,' de ',Nome__X,nl,'                  e ',
        Nome_2,' de ', Nome_Y/ ?',nl] !] e
  resposta(Resp1) e
  Resp1 == sim e
  [! menu([nl,'Como se classificam os dominios de A ->',Nome_1,nl,
                          B -) ',Nome_2,' ?',nl],
  ['Os dominios de A e  B  sao exatamente iguais',
  '0 dominio de A contem totalmente o de B',
   '0 dominio de A esta totalmente contido no de B',
  'Os dominios de A e B possuem uma intersecao',
  'Apesar dos dominios disjuntos, A e B possuem o mesmo papel'],Resp2) !] e
  guarde f_map(Entidade_X,Entidade_Y,Resp2) e
  guarde integrar(Entidade_X) e
  guarde integrar(Entidade_Y).

(e,04) #
se
  entidade(Entidade_X) e
```

```
 entidade(Entidade_Y) e
 Entidade_X \== Entidade_Y e
 pega_faceta(Entidade_X,superclasse(entidades_1)) e
 pega faceta(Entidade_Y,superclasse(entidades_2)) e
 nao assercao perguntou(Entidade_X,Entidade_Y) e
 nao assercao integrar(Entidade_X) e
 nao assercao integrar(Entidade_Y) e
 pega_faceta(Entidade_X,chave([Chave 1])) e
 pega faceta(Entidade_Y,chave([Chave_2])) e
 pega_faceta(Chave_1,nome(Nome_1)) e
 pega_faceta(Chave_2,nome(Nome_2)) e
 pega_faceta(Chave_1,dominio(Tipo_dominio_1) e
 pega faceta(Chave_2,dominio(Tipo_dominio_2)) e
 Tipo_dominio_1 == Tipo_dominio_2 entao
 pega_faceta(Entidade_X,nome(Nome_X)) e
 pega faceta(Entidade_Y,nome(Nome_Y)) e
 guarde perguntou(Entidade_X,Entidade_Y) e
  [! cls,escreva [nl,'As chaves de ',Nome_X,' e ',Nome_Y,
        'possuem o mesmo tipo de dominio',nl,
       'Existe funcao de mapeamento entre ',Nome_1,'
       de ',
       Nome_X,nl,'                    e ',
       Nome_2,' de ', Nome_Y,' ?',nl,'==) '] !] e
  read(Resp1) e
  Resp1 == sim e
  [! menu([nl,'Como se classificam os dominios de  A ->
  ',Nome_1,nl,
                              B -) ',Nome_2,'
                              ?',nl],
  ['Os dominios de A e B sao exatamente iguais',
  'O dominio de  A contem totalmente o de B',
  'O dominio de A esta totalmente contido no de B',
  'Os dominios de A e B possuem uma intersecao',
  'Apesar dos dominios disjuntos, A e B possuem o mesmo
  papel'],Resp2) !] e
  guarde f_map(Entidade_X,Entidade_Y,Resp2) e
  guarde integrar(Entidade_X) e
  guarde integrar(Entidade_Y).

(e,05) #
se
  entidade(Entidade_X) e
  entidade(Entidade_Y) e
  EntidadeX \== Entidade_Y e
  pega_faceta(Entidade_X,superclasse(entidades_1)) e
  pega_faceta(Entidade_Y,superclasse(entidades_2)) e
  nao assercao perguntou(Entidade_X,Entidade_Y) e
  nao assercao integrar(Entidade_X) e
  nao assercao
  integrar(Entidade_Y)
entao
  pega_faceta(Entidade_X,nome(Nome_X)) e
  pega faceta(Entidade_Y,nome(Nome_Y)) e
  pega faceta(Entidade_X,chave([Chave_1])) e
```

```
  pega faceta(Entidade_Y,chave([Chave_2])) e
  pega_faceta(Chave_1,nome(Nome_1)) e
  pega_faceta(Chave_2,nome(Nome_2)) e
  guarde perguntou(Entidade_X,Entidade_Y) e
  [! cls, escreva nl,'Nao ha indicios de equivalencia
        entre ', Nome_X,' e ',Nome_Y,nl,
        'Existe funcao de mapeamento entre ',Nome_1,'
        de ',
        Nome_X,nl,'                    e ',
        Nome_2,' de ', Nome_Y,' ? ,nl,'==> '3 !] e
  read(Resp1) e
  Resp1 == sim e
        [! menu([nl,'Como se classificam os dominios de A ->
        '.Nome_1,nl, B -> ',Nome_2,' ?',nl],
        ['Os dominios de A e  B  sao exatamente iguais',
        '0 dominio de A contem totalmente o de B',
        '0 dominio de A esta totalmente contido no de  B',
        '0s dominios de A e B possuem uma intersecao',
        'Apesar dos dominios disjuntos, A e B possuem o mesmo
         papel'],Resp2) !] e
  guarde f_map(Entidade_X,Entidade_Y,Resp2) e
  guarde integrar(Entidade_X) e
  guarde integrar(Entidade_Y).

(e,06) #
se
  assercao f_map(Entidade_X,Entidade_Y,Resp) e
  nao assercao integrou(Entidade_X,Entidade_Y) e
  Resp ==1
entao
  pega_faceta(Entidade_X,nome(Nome_X)) e
  pega_faceta(Entidade_Y,nome(Nome_Y)) e
  pega_faceta(Entidade_X,chave([Chave_1])) e
  pega_faceta(Entidade_Y,chave([Chave_2])) e
  conta_entidade(Cont) e
  int_text(Cont,Num_ent) e
  transforma_nome(['e0300',Num_ent],Id_nova_entidade) e
  transforma_nome([Nome_X,'_',Nome_Y],Nome_nova_entidade) e
  pega_faceta(Entidade_X,texto(T)) e
  guarde integrou(Entidade_X,Entidade_Y) e
  [ ! cls, menu([nl,'As entidades ',Nome_X,' e ',Nome_Y,
       ' serao integradas',nl,'Gerando a entidade
       ',Nome_ncva_entidade,nl,
       'com identificacao ',Id_nova_entidade,nl],
       ['Concorda','Discorda','Deseja saber porque'],Resp1) !] e
  Resp1 == 1 e
  guarde nova_entidade(Id_nova_entidade) e
  guarde f_map(Id_nova_entidade,Chave_1,Chave_2,1) e
  guarde integrar_atributos(Id_nova_entidade,Entidade_X,Entidade_Y) e
  inclui_quadro(Id_nova_entidade,valor,nome(Nome_nova_entidade)) e
  inclui_quadro(Id_nova_entidade,valor,texto(T)) e
  inclui_quadro(Id_nova_entidade,valor,superclasse(entidades_3)) e
  liga_quadros(Id_nova_entidade,entidades_3).
```

```
(e,07) #
se
  assercao f_map(Entidade_X,Entidade_Y,Resp) e
  nao assercao integrou(Entidade_X,Entidade_Y) e
  Resp == 2
entao
  pega_faceta(Entidade_X,nome(Nome_X)) e
  pega_faceta(Entidade_Y,nome(Nome_Y)) e
  pega_faceta(Entidade_X,chave([Chave_1])) e
  pega_faceta(Entidade_Y,chave([Chave_2])) e
  guarde integrou(Entidade_X,Entidade_Y) e
  conta_entidade(Cont1) e
  int text(Cont1,Num_ent1) e
  transforma_nome(['e0300',Num_ent1],Id_nova_entidade_X) e
  conta_entidade(Cont2) e
  int_text(Cont2,Num_ent2) e
  transforma_nome(['e0300',Num ent2],Id_nova_entidade_Y) e
  transforma_nome([Nome_X,'_3'],Nome_nova_entidade_X) e
  transforma_nome([Nome_Y,'_3'],Nome_nova_entidade_Y) e
  [! els, menu([nl,'As entidades ',Nome_X,' e ',Nome_Y,
       'serao integradas',nl,'gerando as entidades: ',nl,
       Nome_nova_entidade_X,' com identificacao
       ',Id_nova_entidade_X, ' e ',nl,Nome_nova_entidade_Y,'
       com identificacao ", Id_nova_entidade_Y,' subordinada
       a ',Nome_nova_entidade_X,nl],
       ['Concorda','Discorda','Deseja saber porque'],Respl) !] e
  Respl == 1 e
  guarde nova_entidade(Id_nova_entidade_X) e
  guarde nova_entidade(Id_nova_entidade_Y) e
  guarde f_map(Id_nova_entidade_X,Id,nova_entidade_Y,Chave_1,Chave_2,2) e
  guarde integrar_atributos(Id_nova_entidade_X,Id_nova_entidade_Y,
  Entidade_X,Entidade_Y) e
  pega_faceta(Entidade_X,texto(T)) e
  liga_quadros(Id_nova_entidade_X,entidades_3) e
  liga_quadros(Id_nova_entidade_Y,Id_nova_entidade_X) e
  inclui_quadro(Id_nova_entidade_X,valor,nome(Nome_nova_entidade_X)) e
  inclui_quadro(Id_nova_entidade_Y,valor,nome(Nome_nova_entidade_Y)) e
  inclui_quadro(Id_nova_entidade_X,valor,texto(T)) e
  inclui_quadro(Id_nova_entidade_X,valor,superclasse(entidades_3)) e
  inclui_quadro(Id_nova_entidade_Y,valor,superclasse(Id_nova_entidade_X)).

(e,08) #
se
  assercao f_map(Entidade_X,Entidade_Y,Resp) e
  nao assercao integrou(Entidade_X,Entidade_Y) e
  Resp == 3
entao
  pega_faceta(Entidade_X,nome(Nome_X)) e
  pega_faceta(Entidade_Y,nome(Nome_Y)) e
  pega_faceta(Entidade_X,chave([Chave_1])) e
  pega_faceta(Entidade_Y,chave([Chave_2])) e
  guarde integrou(Entidade_X,Entidade_Y) e
  conta_entidade(Cont1) e
  int text(Cont1,Num ent1) e
```

```
 transforma_nome(['e0300',Num ent1],Id_nova_entidade_X) e
 conta_entidade(Cont2) e
 int_text(Cont2,Nument2) e
 transforma_nome(['e0300',Num_ent2],Id_nova_entidade_Y) e
 transforma_nome([Nome_X,'_3'],Nome_nova_entidade_X) e
 transforma_nome([Nome_Y,'_3'],Nome_nova entidade_Y) e
 [! cls, menu([nl,'As entidades ',Nome_X,' e ',Nome_Y,
     serao integradas',nl,'gerando as entidades: ',nl,
     Nome_nova_entidade_X,' com identificacao
     ',Id_nova_entidade_X, 'subordinada a',nl,Nome_nova_entidade_Y,
     ' com identificacao ',Id_nova entidade_Y,nl],
     ['Concorda','Discorda','Deseja saber porque'],Resp1) !] e
 Resp1 == 1 e
 guarde nova_entidade(Id_nova_entidade_X) e
 guarde nova_entidade(Id_nova_entidade_Y) e
 guarde f_map(Id_nova_entidade_Y,Id_nova_entidade_X,Chave_2,Chave_1,2)  e
 guarde integrar_atributos(Id_nova_entidade_Y,Id_nova_entidade_X,
      Entidade_Y,Entidade_X) e
 pega_faceta(Entidade_X,texto(T)) e
 liga_quadros(Id_nova_entidade_X,Id_nova_entidade_Y) e
 liga_quadros(Id_nova_entidade_Y,entidades_3) e
 inclui_quadro(Id_nova_entidade_X,valor,nome(Nome_nova_entidade_X)) e
 inclui_quadro(Id_nova_entidade_Y,valor,nome(Nome_nova_entidade_Y)) e
 inclui_quadro(Id_nova_entidade_X,valor,texto(T)) e
 inclui_quadro(Id_nova_entidade_X,valor,superclasset(Id_nova_entidade_Y)) e
 inclui_quadro(Id_nova_entidade_Y,valor,superclasse(entidades_3)).

(e,09) #
se
 assercao f_map(Entidade_X,Entidade_Y,Resp) e
 nao assercao integrou(Entidade_X,Entidade_Y) e
 Resp == 4
entao
  pega_faceta(Entidade_X,nome(Nome_X)) e
  pega_faceta(Entidade_Y,nome(Nome_Y)) e
  pega_faceta(Entidade_X,chave([Chave_1])) e
  pega_faceta(Entidade_Y,chave([Chave_2])) e
  guarde integrou(Entidade_X,Entidade_Y) e
  conta_entidade(Cont1) e
  int_text(Cont1,Num_ent1) e
  transforma_nome(['e0300',Num_ent1],Id_nova_entidade_X) e
  conta_entidade(Cont2) e
  int_text(Cont2,Num_ent2) e
  transforma_nome(['e0300',Num_ent2],Id_nova_entidade_Y) e
  conta_entidade(Cont3) e
  int_text(Cont3,Num_ent3) e
  transforma_nome(['e0300',Num_ent3],Id_nova_entidade_X_Y) e
  transforma_nome([Nome_X,'_3'],Nome_nova entidade_X) e
  transforma nome([Nome_Y,'_3'],Nome_nova_entidade_Y) e
  transforma_nome([Nome_X,' ',Nome_Y],Nome_nova_entidade_X Y) e
    [! cls, menu([nl,'As entidades ',Nome_X,' e ',Nome_Y,
    ' serao integradas',nl,'gerando as entidades: \nl,
    Nome_nova_entidade_X_Y,' com
    identificacao',Id_nova_entidade_X_Y,
```

```
    nl,'com as entidades subordinadas:',nl,
    Nome_nova_entidade_X,' com identificacao ',
    Id_nova_entidade_X,' e ',nl,Nome_nova_entidade_Y,
    ' com identificacao ',Id_nova_entidade_Y,nl],
    ['Concorda','Discorda','Deseja saber porque'],Resp1) !] e
  Resp1 == 1 e
  guarde nova_entidade(Id_nova_entidade_X) e
  guarde nova_entidade(Id_nova_entidade_Y) e
  guarde nova_entidade(Id_nova_entidade_X_Y) e
  guarde f_map(Id_nova_entidade_X_Y,Id_nova entidade_X,
  Id_nova_entidade_Y,Chave_1,Chave_2,5) e
  guarde integrar_atributcs(Id_nova_entidade_X_Y,Id_nova_entidade_X,
        Id_nova_entidade_Y,Entidade_X,Entidade Y) e
  pega_faceta(Entidade_X,texto(T_X)) e
  pega_faceta(Entidade_Y,texto(T_Y)) e
  liga_quadros(Id_nova_entidade_X_Y,entidades_3) e
  liga_quadros(Id_nova_entidade_X,Id_nova_entidade_X_Y) e
  liga_quadros(Id_nova_entidade_Y,Id_nova_entidade_X_Y) e
  inclui_quadro(Id_nova_entidadeXY,valor,nome(Nome_nova_entidade_X_Y)) e
  inclui_quadro(Id_nova_entidade_X,valor,nome(Nome_nova_entidade_X)) e
  inclui_quadro(Id_nova_entidadeY,valor,nome(Nome_nova_entidade_Y)) e
  inclui_quadro(Id_nova_entidade_X,valor,texto(T_X)) e
  inclui_quadro(Id_nova_entidadeY,valor,texto(T_Y)) e
  inclui quadro(Id_nova_entidade_X,valor,superclasse(Id_nova_entidade_X_Y)) e
  inclui_quadro(Id_nova_entidade_Y,valor,superclasse(Id_nova_entidade_X_Y)) e
  inclui_quadro(Id_nova_entidade_X_Y,valor,superclasse(entidades 3)) e
  atom_string(Nome_nova_entidade_X_Y,T_XY) e
  inclui_quadro(Id_nova_entidade_X_Y,valor,texto(T_XY)).

(e,10) #
se
  assercao f_map(Entidade_X,Entidade_Y,Resp) e
  nao assercao integrou(Entidade_X,Entidade_Y) e
  Resp == 5
entao
  pega_faceta(Entidade_X,nome(Nome_X)) e
  pega_faceta(Entidade_Y,nome(Nome_Y)) e
  pega_faceta(Entidade_X,chave([Chave_1])) e
  pega_faceta(Entidade_Y,chave([Chave_2]))e
  guarde integrou(Entidade_X,Entidade_Y) e
  conta_entidade(Cont1) e
  int_text(Cont1,Num_ent1) e
  transforma_nome(['e0300',Num__ent1],Id_nova_entidade_X) e
  conta_entidade(Cont2) e
  int_text(Cont2,Num_ent2) e
  transforma_nome(['e0300',Num_ent2],Id_nova_entidade_Y) e
  conta_entidade(Cont3) e
  int_text(Cont3,Num_ent3) e
  transforma_nome(['e0300',Num_ent3],Id_nova_entidade_X_Y) e
  transforma_nome([Nome_X,'_3'],Nome_nova_entidade_X) e
  transforma_nome([Nome_Y,'_3'],Nome_nova_entidade_Y) e
  transforma_nome([Nome_X,'_',Nome_Y],Nome_nova_entidade_X_Y) e
      [! cls, menu([nl,'As entidades ',Nome_X,' e ',Nome_Y,
      ' serao integradas',nl,'gerando as entidades: ',nl,
```

```
      Nome_nova_entidade_X_Y,' com identificacao
      ',Id_nova_entidade_X_Y, nl,'com as entidades subordinadas:',nl,
      Nome_nova_entidade_X,' com identificacao ',
      Id_nova_entidade_X,' e ',nl,Nome_nova_entidade_Y,
      ' com identificacao  ,Id_nova_entidade_Y,nl],
      ['Concorda','Discorda','Deseja saber porque'],Resp1) !] e
 Resp1 == 1 e
 guarde nova_entidade(Id_nova_entidade_X) e
 guarde nova_entidade(Id_nova_entidade_Y) e
 guarde nova_entidade(Id_nova_entidade_X_Y) e
 guarde f_map(Id_nova_entidade_X_Y,Id nova_entidade_X,
      Id_nova_entidade_Y,Chave_1,Chave_2,5) e
 guarde integrar_atributos(Id_nova_entidade_X_Y,Id_nova_entidade_X,
      Id_nova_entidade_Y,Entidade_X,Entidade_Y) e
  pega_faceta(Entidade_X,texto(T X)) e
  pega_faceta(Entidade_Y,texto(T_Y)) e
  liga_quadros(Id_nova__entidade_X_Y,entidades_3} e
  liga_quadros(Id_nova_entidade_X,Id_nova entidade_X_Y) e
  liga_quadros(Id_nova_entidade_Y,Id_nova entidade_X_Y) e
  Inclui_quadro(Id_nova_entidade_X_Y,valor,nome(Nome_nova_entidade_X_Y)) e
  inclui_quadro(Id_nova_entidade_X,valor,nome(Nome_nova_entidade_X)) e
  inclui_quadro(Id_nova_entidade_Y,valor,nome(Nome_nova_entidade_Y)) e
  inclui_quadro(Id_nova_entidade_X,valor,texto(T_X)) e
  inclui_quadro(Id_nova entidade_Y,valor,texto(T_Y)) e
  inclui_quadro(Id_nova_entidade_X,valor,superclasses(Id_nova_entidade_X_Y)) e
  inclui_quadro(Id_nova_entidade_Y,valor,superclasse(Id_nova_entidade_X_Y)) e
  inclui_quadro(Id_nova_entidade_X_Y,valor,superclasse(entidades_3)) e
  atom_string(Nome_nova_entidade X_Y,T_XY) e
  inclui_quadro(Id_nova_entidade_X_Y,valor,texto(T_XY)).

(e,11) #
se
  entidade(Entidade_X) e
  pega_faceta(Entidade_X,superclasse (entidades_1)) e
  nao assercao integrar(Entidade_X)
entao
  pega_faceta(Entidade_X,nome(Nome_X)) e
  pega_faceta(Entidade_X,chave([Chave_1])) e
  conta_entidade(Cont) e
  int_text(Cont,Num_ent) e
  transforma nome(['e0300',Num_ent],Id_nova_entidade) e
  transforma_nome([Nome_X,'_','3'],Nome_nova_entidade) e
  pega_faceta(Entidade_X,texto(T)) e
  liga_quadros(Id_nova_entidade,entidades_3) e
  inclui_quadro(Id_nova_entidade,valor,nome(Nome_nova_entidade}) e
  inclui_quadro(Id_nova_entidade,valor,texto(T)) e
  inclui_quadro(Id_nova_entidade,valor,supereclasse(entidades_3)) e
  guarde integrar(Entidade_X) e
  guarde nova_entidade(Id_nova_entidade) e
  guarde transportou(Id_nova_entidade,Entidade_X).

(e,12) #
se
  entidade(Entidade_Y) e
```

```
  pega_faceta(Entidade_Y,superclasse(entidades_2)) e
  nao assercao integrar(Entidade_Y)
entao
  pega_faceta(Entidade_Y,nome(Nome Y)) e
  pega_faceta(Entidade_Y,chave([Chave_1])) e
  conta_entidade(Cont) e
  int_text(Cont,Num_ent) e
  transforma_nome(['e0300',Num_ent],Id_nova_entidade) e
  transforma nome([Nome_Y,'_','3'],Nome_nova_entidade) e
  pega_faceta(Entidade_Y,texto(T)) e
  liga_quadros(Id_nova_entidade,entidades_3) e
  inclui_quadro(Id_nova_entidade,valor,nome(Nome_nova_entidade)) e
  inclui_quadro(Id_nova_entidade,valor,texto(T)) e
  inclui_quadro(Id_nova_entidade,valor,superclasse(entidades_3)) e
  guarde integrar(Entidade_Y) e
  guarde nova_entidade(Id_nova_entidade) e
  guarde transportou(Id_nova_entidade,Entidade_Y).
```

Integração de Relacionamentos

```
(r,01) #
se
  assercao integrou(Relacionamento_X) e
  relacionamento(Relacionamento_X)
entao
  retract(relacionamento(Relacionamento_X)).

(r,02) #
se
  relacionamento(Relacionamento_X) e
  relacionamento(Relacionamento_Y) e
  Relacionamento_X  \== Relacionamento_Y e
  pega_faceta(Relacionamento_X,superclasse(relacionamentos_1)) e
  pegafaceta(Relacionamento_Y,superclasse(relacionamentos_2)) e
  nao assercao integrou(Relacionamento_X) e
  nao assercao integrou(Relacionamento_Y) e
  pega_faceta(Relacionamento_X,entidades_participantes(Z)) e
  pega_faceta(Relacionamento_Y,entidades_participantes(W)) e
  assercao integrou(Entidade_A,Entidade_C) e
  assercao  integrou(Entidade_B,Entidade_D)  e
  Entidade_A  \== Entidade_B e
  Entidade_C \== Entidade_D e
  membro(Entidade_A,Z) e
  membro(Entidade_B,Z) e
  membro(Entidade_C,W) e
  membro(Entidade_D,W)
entao
  guarde ntegrou(Relacionamento_X) e
  guarde integrou(Relacionamento_Y) e
  guarde integrar((Relacionamento_X,Z),(Relacionamento_Y,W)).

(r,03) #
  assercao integrar((Relacionamento_X,Z),(Relacionamento_Y,W)) e
  nao assercao perguntou(Relacionamento_X,Relacionamento_Y) e
```

```
 pega_faceta(Relacionamento_X,grau(Grau_X)) e
 pega_faceta(Relacionamento_Y,grau(Grau_Y)) e
 sinonimo(Relacionamento_X,Relacionamento_Y) e
 Grau_X =:= Grau_Y
entao
 guarde perguntou(Relacionamento_X,Relacionamento_Y) e
 [! cls, escreva [Relacionamento_X,' e ',Relacionamento_Y, nl,
       'Sao sinonimos e possuem o mesmo grau ',nl,nl,
       Relacionamento_X,'do esquema 1 tem o mesmo papel que
       ', Relacionamento_Y,' do esquema 2 ?',nl,nl] !]e
 resposta(Resp) e
 guarde
 papel((Relacionamento_X,Z),(Relacionamento_Y,W),Grau_X,Resp).

(r,04) #
 assercao integrar((Relacionamento_X,Z),(Relacionamento_Y,W)) e
 nao assercao perguntou(Relacionamento_X,Relacionamento_Y) e
 pega_faceta(Relacionamento_X,grau(Grau_X)) e
 pega_faceta(Relacionamento_Y,grau(Grau_Y)) e
 pega_faceta(Relacionamento_X,nome(Nome_X)) e
 pega_faceta(Relacionamento_Y,nome(Nome_Y)) e
 Nome_X == Nome_Y e
 Grau_X =:= Grau_Y
entao
 guarde perguntou(Relacionamento_X,Relacionamento_Y) e
 [! cls, escreva ['Osrelacicnamentos sao homonimos e pessuem o
       mesmo grau', nl, nl,
       Relacionamento_X,' do esquema 1 tem o mesmo papel
       que ', Relacionamento_Y,' do esquema 2 ?',nl,nl] !] e
 resposta(Resp) e
 guarde papel((Relacionamento_X,Z),(Relacionamento_Y,W),Grau_X,Resp).

(r,05) #
se
 assercao integrar((Relacionamento_X,Z),(Relacionamento_Y,W)) e
 nao assercao perguntou(Relacionamento_X,Relacionamento_Y) e
 pega_faceta(Relacionamento_X,grau(Grau X)) e
 pega facet a(RelacionamentoY,grau(Grau_Y)) e
 Grau_X =:= Grau_Y
entao
 guarde perguntou(Relacionamento_X,Relacionamento_Y) e
 [! cls, escreva [Relacionamento_X,' e ',Relacionamento_Y,nl,
       'Possuem o mesmo grau e',nl,'associam entidades
       integradas ',nl,
       nl, Relacionamento_X,' no esquema 1 tem o mesmo papel
       que ', Relacionamento_Y,' no esquema 2 ?\nl,nl] !] e
 resposta(Resp) e
 guarde papel((Relacionamento_X,Z),(Relacionamento_Y,W),Grau_X,Resp).

(r,06) #
se
 assercao papel((Relacionamento_X,Z),(Relacionamento_Y,W),Grau,Resp) e
 Resp == Sim e
 nao assercao incorporar_entidade5(_,(Relacionamento_X,Z),(Relacionamento_Y,W))
```

```
entao
  transforma_nome([Relacionamento_X,Relacionamento_Y],
              Nome_novo_relacionamento} e
  guarde novo relacionamento(Nome_novo_relacionamento) e
  inclui_quadro(Nome_novo_relacionamento,valor,
          superclasse(relacionamentos_3)) e
  inclui_quadro(Nome_novo_relacionamento,valor,grau(0)) e
  inclui_quadro(Nome_novo_relacionamento,valor,entidades_participantes([])) e
  liga_quadros(Nome_novo_relacionamento,relacionamentos_3) e
  guarde incorporar_entidades(Nome_novo_relacionamento,(Relacionamento_X,Z),
         (Relacionamento_Y,W)).

(r,07) #
se
  assercao incorporar_entidades(Nome_novo_relacionamento,
         (Relacionamento_X,Z),(Relacionamento_Y,W)} e
   assercao integrar_atributos(Nome_nova_entidade,Entidade_X,Entidade_Y) e

 membro(Entidade_X,Z) e
 membro(Entidade_Y,W) e
 nao assercao ajustou_grau(Nome_novo_relacionamento,Entidade_X,Entidade_Y)
entao
 guarde ajustou_grau(Nome_novo_relacionamento,Entidade_X,Entidade_Y) e
 pega_faceta(Nome_novo_relacionamento,grau(G)) e
 pega_faceta(Nome_novo_relacionamento,entidades_participantes(K)) e
 G1 is G + 1 e
 exclui_quadro(Nome_novo_relacionamento,valor,grau(G)) e
 exclui_quadro(Nome_novo_relacionamento,valor,entidades_participantes(K)) e
 inclui_quadro(Nome_novo_relacionamento,valor,grau(G1) e
 inclui_quadro(Nome_novo_relacionamento,valor,
      entidades_participantes([Nome_nova_entidade|K])).

(r,08) #
se
  assercao incorporar_entidades(Nome_novo_relacionamento,
         (Relacionamento_X,Z),(Relacionamento_Y,W)) e
  assercao integrar_atributos(Nome_nova_entidade_X,
        Nome_nova entidade_Y,Entidade_X,Entidade_Y) e
  nao assercao ajustou_grau(Nome_novo_relacionamento,Entidade_X,Entidade_Y) e
  ((membro(Entidade_X,Z) e membro(Entidade_Y,W)) ou
  (membro(Entidade_X,W) e
  membro(Entidade_Y,Z)))
entao
  guarde ajustou_grau(Nome_novo_relacionamento,Entidade_X,Entidade_Y) e
  pega_faceta(Nome_novo_relacionamento,grau(G)) e
  pega_faceta(Nome_novo_relacionamento,entidades_participantes(K)) e
  G1 is G + 1 e
  exclui_quadro(Nome_novo_relacionamento,valor,grau(G)) e
  exclui_quadro(Nome_novo_relacionamento,valor,entidades_participantes(K)) e
  inclui _quadro(Nome_novo_relacionamento, valor,grau(GU) e
  inclui_quadro(Nome_novo_relacionamento,valor,
      entidades_participantes([Nome_nova_entidade_X|K])).

(r,09) #
```

```
se
  assercao incorporar_entidades(Nome_novo_relacionamento,
        (Relacionamento_X,Z),(Relacionamento_Y,W)) e
  assercao integrar_atributos(Nome nova_entidade_X_Y,Nome_nova_entidade_X,
        Nome_nova_entidade_Y,Entidade_X,Entidade_Y) e
  membro(Entidade_X,Z) e
  membro(Entidade_Y,W) e
  nao assercao ajustou_grau(Nome_novo_relacionamento,Entidade_X,Entidade_Y)
entao
  guarde ajustou_grau(Nome_novo_relacionamento,Entidade_X,Entidade_Y) e
  pega_faceta(Nome_novo_relacionamento,grau(G)) e
  pega_faceta(Nome_novo_relacionamento,entidades_participantes(K)) e
  G1 is G + 1 e
  Exclui_quadro(Nomenovo_relacionamento,valor,grau(G)) e
  exclui_quadro(Nome_novo_relacionamento,valor,entidades_participantes(K)) e
  inclui_quadro(Nome_novo_relacionamento,valor,grau(G1)) e
   inclui_quadro(Nome_novo_relacionamento,valor,
       entidades_participantes([Nome_nova_entidade_X_Y|K])).

(r,10) #
se
   assercao papel((Relacionamento__X,Z),(Relacionamento_Y,W),Grau,Resp) e
   Resp \== sim e
   nao assercao verificou_dominios((Relacionamento_X,Z), (Relacionamento_Y,W),_)
entao
   [! cls, menu([nl,'Como se classificam os dominios de  A ->
      ',Relacionamento_X,nl,B -> ',Relacionamento_Y,' ?',nl],
   ['0 dominio de A coritem o dominio de  B',
   '0 dominio de B contem totalmente o de  A',
   '0s dominios de A e  B  sao disjuntos',
   '0s dominios de A e B possuem uma intersecao'],Resp2) !) e
   guarde verificou_dominios((Relacionamento_X,Z),
       (Relacionamento_Y,W),Resp2).

(r,11) #
se
   assercao verificou_dominios((Relacionamento_X,Z), (Relacionamento_Y,W),Resp) e
   Resp == 1 e
   nao assercao integrou(Relacionamento_X,Relacionamento_Y)
entao
   transforma_nome([Relacionamento_X,'_3'],Nome_novo_relacionamento_X) e
   transforma_nome([Relacionamento_Y,'_3'],Nome_novo_relacionamento_Y) e
   guarde integrou(Relacionamento_X,Relacionamento_Y) e
   guarde novo_relacionamento(Nome_novo_relacionamento_X) e
   guarde novo_relacionamento(Nome novo_relacionamento_Y) e
   liga_quadros(Nome_novo_relacionamento_Y,Nome_novo_relacionamento_X) e
   liga_quadros(Nome_novo_relacionamento_X,relacionamentos_3) e
   inclui_quadro(Nome_novo_relacionamento_X,valor,
        superclasse(relacionamentos_3)) e
   inclui_quadro(Nome_novo_relacionamento_X,valor,grau(0)) e
   inclui_quadro(Nome_novo_relacionamento_X,valor,entidades_participantes([])) e
   inclui_quadro(Nome_novo_relacionamento_Y,valor,
        superclasse(Nome_novo_relacionamento_X) e
   inclui_quadro(Nome_novo_relacionamento_Y,valor,grau(01) e
```

```
 inclui_quadro(Nome_novo_relacionamento_Y,valor,
      entidades_participantes([])) e
 guarde incorporar_entidades(Nome_novo_relacionamento_X,
      (Relacionamento_X,Z)) e
 guarde incorporar_entidades(Nome_novo_relacionamento_Y,(Relacionamento_Y,W)).

(r,12) #
se
 assercao verificou_dominios((Relacionamento_X,Z),(Relacionamento_Y,W),Resp) e
 Resp == 2 e
 nao assercao integrou(Relacionamento_X,Relacionamento_Y)
entao
 transforma_nome([Relacionamento_X,'_3'],Nome_novo_relacionamento_X) e
 transforma_nome([Relacionamento_Y,'_3'],Nome_novo_relacionamento_Y) e
 guarde integrou(Relacionamento_X,Relacionamento_Y) e
 guarde novo_relacionamento(Nome_novo_relacionamento_X) e
 guarde novo_relacionamento(Nome_novo_relacionamento_Y) e
 liga_quadros(Nome_novo_relacionamento_X,Nome_novo_relacionamento_Y) e
 liga_quadros(Nome_novo_relacionamento_Y,relacionamentos_3) e
 inclui_quadro(Nome_novo_relacionamento_X,valor,
    superclasse(Nome_novo_relacionamento_Y))
 inclui_quadro(Nome_novo_relacionamento_X,valor,grau(0)) e
 inclui_quadro(Nome_novo_relacionamento_X,valor,
    entidades_participantes([])) e
 inclui_quadro(Nome_novo_relacionamento_Y,valor,
    superclasse(relacionamentos_3)) e
 inclui_quadro(Nome_novo_relacionamento_Y,valor,grau(O)) e
 inclui_quadro(Nome_novo_relacionamento_Y,valor,entidades_participantes([])) e
 guarde incorporar_entidades(Nome_novo_relacionamento_X,(Relacionamento_X,Z) e
 guarde incorporar_entidades(Nome_novo_relacionamento_Y,(Relacionamento_Y,W)).

(r,13) #
se
 assercao verificou_dominios((Relacionamento_X,Z),(Relacionamento_Y,W),Resp) e
 Resp == 3 e
 nao assercao integrou(Relacionamento_X,Relacionamento_Y)
entao
 transforma_nome(Relacionamento_X,'_3'],Nome_novo_relacionamento_X) e
 transforma_nome(Relacionamento_Y,'_3'],Nome_novo_relacionamento Y) e
 guarde integrou(Relacionamento_X,Relacionamento_Y) e
 guarde novo_relacionamento(Nome_novo_relacionamento X) e
 guarde novo_relacionamento(Nome_novo_relacionamento_Y) e
 liga_quadros(Nome_novo_relacionamento_X,relacionamentos_3) e
 liga_quadros(Nome_novo_relacionamento_Y,relacionamentos_3) e
 inclui_quadro(Nome_novo_relacionamento_X,valor,
    superclasse(relacionamentos_3)) e
 inclui_quadro(Nome_novo_relacionamento_X,valor,grau(0)) e
 inclui_quadro(Nome_novo relacionamento.X,valor,
    entidades_participantes([])) e
 inclui_quadro(Nome_novo_relacionamento_Y,valor,
    superclasse(relacionamentos 3)) e
 inclui_quadro(Nome novo_relacionamento_Y,valor,grau(O)) e
 inclui quadro(Nome_novo_relacionamento_Y,valor,entidades_participantes([])) e
 guarde incorporar entidades(Nome_novo_relacionamento_X,Relacionamento_X,Z)) e
```

```
 guarde incorporar_entidades(Nome_nGvo_relacionamento_Y,(Relacionamento_Y,W)).

(r,14) #
se
 assercao incorporar entidades(Nome_novo_relacionamento,
       (Relacionamento_X,Z),(Relacionamento_Y,W)) e
 assercao transportou(Nome_nova_entidade,Entidade_X) e
 nao assercao ajustou_grau(Nome_novo_relacionamento,Entidade_X) e
 membro(Entidade_X,Z)
entao
 guarde ajustou grau(Nome_novo_relacionamento,Entidade_X) e
 pega_faceta(Nome_novo_reiacionamento,grau(G)) e
 pega_faceta(Nome_novo_relacionamento,entidades_participantes(K)) e
 G1 is G + 1 e
 Exclui_quadro(Nome_novo_relacionamento,valor,grau(G)) e
 exclui_quadro(Nome_novo_relacionamento,valor,entidades_participantes(K)) e
 inclui_quadro(Nome novo_relacionamentc,valor,grau(G1)) e
 inclui_quadro(Nome_novo__relacionamento, valor,
     entidades_participantes([Nome_nova_entidade!K])).

(r,15) #
se
 assercao incorporar_entidades(Nome_novo_relacionamento,
       (Relacionamento_X,Z},(Relacionamento_Y,W)) e
 assercao transportou(Nome_nova_entidade,Entidade_Y) e
 nao assercao ajustou_grau(Nome novo_relacionamento,Entidade_Y) e
 membro(Entidade_Y,W)
entao
 guarde ajustou_grau(Nome_novo_relacionamento,Entidade_Y) e
 pega_faceta(Nome_novo_relacionamento,grau(G)) e
 pega_faceta(Nome_novo_relacionamento,entidades_participantes(K)) e
 G1 is G + 1 e
 exelui_quadro(Nome_novo_relacionamento,valor,grau(G)) e
 exclui_quadro(Nome_novo_relacionamento,valor,entidades_participantes(K)) e
 inclui_quadro(Nome_novo relacionamento,valor,grau(G1)) e
 inclui_quadro(Nome_novo_relacionamento,valor,
     entidades_participantes([Nome_nova_entidade!K])),

(r,16) #
se
 assercao incorporar_entidades(Nome_novo_relacionamento,(Relacionamento,Z)) e
 assercao transportou(Nome nova_entidade,Entidade) e
 nao assercao ajustou_grau(Entidade,Nome novo relacionamento) e
 membro(Entidade,Z) entao
 guarde ajustou_grau(Entidade,Nome_novo relacionamento) e
 pega_faceta(Nome_novo_relacionamento,grau(G}) e
 pega_faceta(Nome_novo_relacionamento,entidades_participantes(K)) e
 G1 is G + 1 e
 exclui_quadro(Nome novorelacionamento,valor,grau(G)) e
 exclui_quadro(Nome novo_relacionamento,valor,entidades_participantes(K)) e
 inclui_quadro(Nome_novo_relacionamento,valor,grau(G1)) e
 inclui_quadro(Nome_novo_relacionamento,valor,
     entidades_participantes([Nome_nova_entidade!K])).
```

```
(r,17) #
se
 relacionamento(Relacionamento_X) e
 pega_faceta(Relacionamento_X,superclasse(relacionamentos_1)) e
 nao assercao integrou(Relacionamento_X)
entao
 guarde integrou(Relacionamento_X) e
 transforma_nome([Relacionamento_X,'_3'],Nome_novo_relacionamento_X) e
 pega_faceta(Relacionamento_X,texto(T)) e
 liga_quadros(Nome_novo_relacionamento_X,relacionamentos_3) e
 inclui_quadro(Nome_novo_relacionamento_X,valor,texto(T)) e
 inclui_quadro(Nome_novo_relacionamento_X,valor,superclasse(relacionamentos_3)) e
 inclui_quadro(Nome_ novo_relacionamento_X,valor,grau(0)) e
 inclui_quadro(Nome_novo_relacionamento_X,valor,
    entidades_participantes([])) e
 guarde novo_relacionamento(Nome_novo_relacionamento_X} e
 guarde transportou(Nome_novo_relacionamento_X,Relacionamento_X) e
 pega_faceta(Relacionamento_X,entidades_participantes(Z)) e
 guarde incorporar_entidades(Nome_novo_relacionamento_X,(Relacionamento_X,Z).

(r,18) #
se
 relacionamento(Relacionamento_Y) e
 pega_faceta(Relacionamento_Y,superclasse(relacionamentos_2)) e
 nao assercao integrou(Relacionamento_Y)
entao
 guarde integrou(Relacionamento_Y) e
 transforma_nome([Relacionamento_Y,'_3'],Nome_novo_relacionamento_Y) e
 pega_faceta(Relacionamento_Y,texto(T)) e
 liga_quadros(Nome_novo_relacionamento_Y,relacionamentos_3) e
 inclui_quadro(Nome_novo_reiacionamento_Y,valor,texto(T)) e
 inclui_quadro(Nome_novo_relaciDriamento_Y, valor,
    superclasse(relacionamentos_3)) e
 inclui_quadro(Nome_novo_relacionamento_Y,valor,grau(O)) e
 inclui_quadro(Nome_novo_relacionamento_Y,valor,
    entidades_ participantes([])) e
 guarde novo_relacionamento(Nome_novo_relacionamento_Y) e
 guarde transportou(Nome_novo_relacionamento_Y,Relacionamento_Y) e
 pega_faceta(Relacionamento_Y,entidades_participantes(Z)) e
 guarde incorporar entidades(Nome_novo_relacionamento_Y,(Relacionamento_Y,Z)).

(r,19) #
se
 assercao incorporar_entidades(Nome_novo_relacionamento,
    (Relacionamento_X,Z)) e
 assercao integrar_atributos(Nome_nova_entidade,Entidade_X,Entidade_Y) e
 nao assercao ajustou_grau(Nome_novo_relacionamento,Entidade_X,Entidade Y) e
 (membro(Entidade_X,Z) ou membro(Entidade_Y,Z))
entao
 guarde ajustou_grau(Nome_novo_relacionamento,Entidade_X,Entidade_Y) e
 pega_faceta(Nome_novo_relacionamento,grau(G)) e
 pega_faceta(Nome_novo_relacionamento,entidades_participantes(K)) e
 G1 is G + 1 e
 exclui_quadro(Nome_novo_relacionamento,valor,grau(G)) e
```

```
 exclui_quadro(Nome_novo_relacionamento,valor,
     entidades_participantes(K)) e
 inclui quadro(Nome_novo_relacionamento,valor,grau(Gl), e
 inciui_quadro(Nome_novo_relacionamento,valor,
     entidades_participantes([Nome_nova_entidade[K])).

(r,20) #
se
 assercao incorporar_entidades(Nome_novo_relacionamento,(Relacionamento_X,Z)) e
 assercao integrar_atributos(Nome_nova_entidade_X,
      Nome_nova_entidade_Y,Entidade_X,Entidade_Y) e
 nao assercao ajustou_grau(Nome_novo_relacionamento,Entidade_X,Entidade_Y) e
 (membro(Entidade_X,Z) ou  membro(Entidade_Y,Z))
entao
 guarde ajustou_grau(Nome_novo_relacionamento,Entidade_X,Entidade_Y) e
 pega_faceta(Nome_novo_relacionamento,grau(G)) e
 pega_faceta(Nome_novo_relacionamento,entidades_participantes(K)) e
 G1 is G + 1 e
 exclui_quadro(Nome_novo_relacionamento,valor,grau(G)) e
 exclui_quadro(Nome_novo_relacionamento,valor,entidades_participantes(K)) e
 inclui_quadro(Nome_novo_relacionamento,valor,grau(G1)) e
 inclui_quadro(Nome_novo_relacionamento,valor,
     entidades_participantes([Nome_nova_entidade_X|K])).

(r,21) #
se
 assercao incorporar entidades(Nome_novo_relacionamento,(Relacionamento_X,Z)) e
 assercao integrar_atributos(Nome_nova entidade_X_Y,Nome_nova_entidade_X,
      Nome_nova_entidade_Y,Entidade_X,Entidade_Y) e
 nao assercao ajustou_grau(Nome_novo_relacionamento,Entidade_X,Entidade_Y) e
 (membro(Entidade_X,Z) ou membro(Entidade_Y,Z))
entao
 guarde ajustou_grau(Nome_novo_relacionamento,Entidade_X,Entidade_Y) e
 pega_faceta(Nome_novo_relacionamento,grau(G)) e
 pega_faceta(Nome_novo_relacionamento,entidades_participantes(K)) e
 G1 is G + 1 e
 exclui_quadro(Nome_novo_relacionamento,valor,grau(G)) e
 exclui_quadro(Nome_novo_relacionamento,valor,
    entidades_participantes(k)) e
 inclui_quadro(Nome_novo_relacionamento,valor,grau(G1)) e
 inclui_quadro(Nome_novo_relacionamento,valor,
     entidades_participantes([Nome_nova_entidade_X_Y|K])).
```

Integração de Atributos

```
(a,01) #
se
 assercao integrou(Atributo) e
 atributo(Atributo)
entao
 retract(atributo(Atributo)).

(a,02) #
se
```

```
 assercao transportou(Atributo) e
 atributo(Atributo)
entao
 retract(atributo(Atributo)).

(a,03) #
se
 nao assercao sabe_estrategia
entao
 cls e
 escreva ['Informe a estrategia para integracao de atributos',nl,nl,'==> '] e
 read(Estrategia) e
 guarde Estrategia e
 guarde sabe_estrategia.

(a,04) #
se
 assercao integrar_atributos(Nome_nova_entidade,Entidade_X,Entidade_Y) e
 atributo(Atributo_1) e
 pega_faceta(Entidade_X,atributos(Atributos_X)) e
 membro(Atributo_1,Atributos_X) e
 pega_faceta(Entidade_X,chave(Chave 1)) e
 nao membro(Atributo_1,Chave_1) e
 atributo(Atributo_2) e
 pega_faceta(Entidade_Y,atributos(Atributos_Y)) e
 membro(Atributo_2,Atributos_Y) e
 nao assercao perguntou(Atributo_l,Atributo_2) e
 nao assercao integrar(Atributo_1) e
 nao assercao integrar(Atributo_2) e
 pega_faceta(Entidade_Y,chave(Chave_2)) e
 nao membro(Atributo_2,Chave_2)
entao
 guarde perguntou(Atributo_1,Atributo_2) e
 escreva [nl,'Existe funcao de mapeamento entre ',Atributo_1,
       nl,'                    e ',Atributo_2,'
       ?',nl,'==> '] e
 read(Resp1) e
  ! e
 Resp1 == sim e
 guarde f_map(Nome_nova_entidade,Atributo_1,Atributo_2,l) e
 guarde integrar(Atributo_1) e
 guarde integrar(Atributo_2).

(a,05) #
se
 assercao integrar_atributos(Nome_nova_entidade_X,Nome_nova_entidade_Y
      , Entidade_X,Entidade_Y) e
 atributo(Atributo_1) e
 pega_faceta(Entidade_X,atributos(Atributos_X)) e
 membro(Atributo_1,Atributos_X) e
 pega_faceta(Entidade_X,chave(Chave_1) e
 nao membro(Atributo_1,Chave_1) e
 atributo(Atributo_2) e
 pega_faceta(Entidade_Y,atributos(Atributos_Y)) e
```

```
  membro(Atributo_2,Atributos_Y) e
  nao assercao perguntou(Atributo_1,Atributo_2) e
  nao assercao integrar(Atributo_1) e
  nao assercao integrar(Atributo_2) e
  pega_faceta(Entidade_Y,chave(Chave_2)) e
  nao membro(Atributo_2,Chave_2)
entao
  guarde perguntou(Atributo_1,Atributo_2) e
  escreva [nl,'Exists funcao de mapeamento entre ',Atributo_1,
       nl,'        ,              e ',Atributo_2,'
       ?',nl,'==> '] e
  read(Respl) e
  ' e
  Respl == sim e
  menu([nl,'Como se classificam os dominios de  A -> \Atributo_1,nl,
                        B -> ',Atributo[2,'?',nl],
  ['Os dominios de A e B sao exatamente iguais',
  'O dominio de A contem totalmente o de B'],Resp2) e
  guarde integrar(Atributo_1) e
  guarde integrar(Atributo_2) e
  guarde f_map(Nome_nova_entidade_X,Nome_nova
          entidade_Y,Atributo_1, Atributo_2,Resp2).

(a,06) #
se
  assercao integrar atributos(Nome_nova_entidade_X_Y,
  Nome_nova_entidade_X, Nome_nova_entidade_Y,Entidade_X,Entidade_Y) e
  atributo(Atributo_1) e
  pega_faceta(Entidade_X,atributos(Atributos_X)) e
  membro(Atributo_1,Atributos_X) e
  pega faceta(Entidade_X,chave(Chave_1)) e
  nao membro(Atributo_1,Chave_1) e
  atributo(Atributo_2) e
  pega_faceta(Entidade_Y,atributos(Atributos_Y))- e
  membro(Atributo_2,Atributos Y) e
  nao assercao perguntou(Atributo_1,Atributo_2) e
  nao assercao integrar(Atributo_1) e
  nao assercao integrar(Atributo_2) e
  pega_faceta(Entidade_Y,chave(Chave_2)) e
  nao membro(Atributo_2,Chave_2)
entao
  guarde perguntou(Atributo_1,Atributo_2) e
  escreva [nl,'Existe funcao de mapeamento entre ',Atributo_1,
       nl,'                  e ',Atributo_2,'?',nl,'==> '] e
  read(Respl) e
  ! e
  Resp1 == sim e
  menu([nl,'Como se classificam 05 dominios de  A ->',Atributo_1,nl,
                          B -> ',Atributo_2,' ?',nl],
  ['Os dominies de A e B  sao exatamente iguais', 'O
  dominio de A contem totalmente o de B', 'O
  dominio de A esta totalmente contido no de B', 'Os
  dominios de A e B possuem uma intersecao'],Resp2) e
  guarde integrar(Atributo_1) e
```

```
  guarde integrar(Atributo_2) e
  guarde f_map(Nome_nova_entidade_X_Y,Nome_nova_entidade_X,
          Nome_nova_entidade_Y,Atributo_1,Atributo_2,Resp2).

(a,07) #
se
  assercao f_map(Nome_nova_entidade,Atributo_1,Atributo_2,Resp2) e
  nao assercao integrou(Atributo_1) e
  nao assercao integrou(Atributo_2) e
  Resp2 ==1
entao
  Transforma_nome([Atributo_1,'_',Atributo_2],Nome_novo_atributo) e
  guarde novo_atributo(Nome_novo_atributo) e
  liga_quadros(Nome_novo_atributo,atributos_3) e
  inclui_quadro(Nome_novo_atributo,valor,membro_de(Nome_nova_entidade)) e
  inclui_quadro(Nome_novo_atributo,valor,superclasse(atributos_3)) e
  pega_faceta(Atributo_1,texto(T)) e
  inclui_quadro(Nome_novo_atributo,valor,texto(T)) e
  pega_faceta(Atributo_1,cardinalidade_max(Cmax)) e
  inclui_quadro(Nome_novo_atributo,valor,cardinalidade_max(C_max)) e
  pega_faceta(Atributo_1,cardinalidade_min(C_min)) e
  inclui_quadro(Nome_novo_atributo,valor,cardinalidade_min(C_min)) e
  pega_faceta(Atributo_1,dominio(Tipo_dominio_D)) e
  inclui_quadro(Nome_novo_atributo,valor,dominio(Tipo_dominio_D)) e
  guarde integrou(Atributo_1) e
  guarde integrou(Atributo_2).

(a,08) #
se
  assercao f_map(Nome_nova_entidade_X,Nome_nova
           entidade_Y,Atributo_1, Atributo_2,Resp2) e
  nao assercao integrou(Atributo_1) e
  nao assercao integrou(Atributo_2) e
  (Resp2 == 1 ou (Resp2 == 2 , assercao estrategia_1) ou
   (Resp2 == 2 , assercao estrategia_4))
entao
  transforma_nome([Atributo_1,'_',Atributo 2],Nome_novo_atributo) e
  guarde novo_atributo(Nome_novo_atributo) e
  liga_quadros(Nome novo_atributo,atributos_3) e
  inclui_quadro(Nome_novo_atributo,valor,membro_de(Nome_nova_entidade_X)) e
  inclui_quadro(Nome_novo_atributo,valor,superclasse(atributos_3)) e
  pega_faceta(Atributo_1,texto(T)) e
  inclui_quadro(Nome_novo_atributo,valor,texto(T)) e
  pega_faceta(Atributo_1,cardinalidade_max(C_max)) e
  inclui_quadro(Nome_novo_atributo,valor,cardinalidade_max(C_max)) e
  pega_faceta(Atributo_1,cardinalidade_min(Cmin)) e
  inclui_quadro(Nome_novo_atributo,valor,cardinalidade_min(C_min)) e
  pega_faceta(Atributo_1,dominio(Tipo_dominio_D)) e
  inclui_quadro(Nome_novo_atributo,valor,dominio(Tipo_dominio_D)) e
  guarde integrou(Atributo_1) e
  guarde integrou(Atributo_2).

(a,09) #
se
```

```
  assercao f_map(Nome_nova_entidade_X,Nome_nova_entidade_Y,Atributo_1,
  Atributo_2,Resp2) e
  nao assercao integrou(Atributo_1) e
  nao assercao integrou(Atributo_2) e
  Resp2 == 2 e
  assercao estrategia_2
entao
  transforma_nome([Atributo_1,'_3'],Nome_novo_atributo_1) e
  transforma nome([Atributo_2,'_3'],Nome_novo_atributo_2) e
  guarde novo_atributo(Nome_novo_atributo 1) e
  guarde novo_atributo(Nome_novo_atributo 2) e
  liga_quadros(Nome novo_atributo_1,atributos_3) e
  inclui_quadro(Nome_novo_atributo_1,valor,membro_de(Nome_nova_entidade_X)) e
  inclui_quadro(Nome_novo_atributo_1,valor,superclasse(atributos_3)) e
  pega_faceta(Atributo_1,texto(T)) e
  inclui_quadro(Nome_novo_atributo_1,valor,texto(T)) e
  pega_faceta(Atributo_1,cardinalidade_max(C_max)) e
  inclui_quadro(Nome_novo_atributo_1,valor,cardinalidade_max(C_max)) e
  pega_faceta(Atributo_1,cardinalidade_min(C_min)) e
  inclui_quadro(Nome_novo_atributo_1,valor,cardinalidade_min(C_min)) e
  pega faceta(Atributo_1,dominio(Tipo_dominio_D)} e
  inclui_quadro(Nome_novo_atributo_1,valor,dominio(Tipo_dominio_D)) e
  liga_quadros(Nome_novo_atributo_2,atributos_3) e
  inclui_quadro(Nome_novo_atributo_2,valor,membro_de(Nome_nova_entidade_Y)) e
  inclui_quadro(Nome_novo_atributo_2,valor,superclasse(atributos_3)) e
  pega_faceta(Atributo_1,texto(T)) e
  inclui quadro(Nome_novo_atributo_2,valor,texto(T)) e
  pega_faceta(Atributo_I,cardinalidade_max(C_max)) e
  inclui_quadro(Nome_novc_atributo_2,valor,cardinalidade_max(C_max)) e
  pega_faceta(Atributo_1,cardinalidade_min(C_min)) e
  inclui_quadro(Nome_novo_atributo_2,valor,cardinalidade_min(C_min)) e
  pega_faceta(Atributo_1,dominio(Tipo dominio D)) e
  inclui_quadro(Nome_novo_atributo_2,valor,dominio(Tipo_dominio_D)) e
  guarde integrou(Atributo_1) e
  guarde integrou(Atributo_2).

(a,10) #
se
  assercao f_map(Nome_nova_entidade_X,Nome_nova_entidade_Y,Atributo_1,
          Atributo_2,Resp2) e
  nao assercao integrou(Atributo_1) e
  nao assercao integrou(Atributo_2) e
  Resp2 == 2 e
  assercao estrategia_3
entao
  transforma_nome([Atributo_1,'_3'],Nome_novo_atributo_1) e
  transforma_nome([Atributo'_2,'_3'],Nome_novo_atributo_2) e
  guarde novo_atributo(Nome_novo_atributo_1) e
  guarde novo_atributo(Nome_novo_atributo_2) e
  liga_quadros(Nome_novo_atributo_1,atributos_3) e
  inclui_quadro(Nome_novo_atributo_1,valor,membro_de(Nome_nova_entidade_X)) e
  inclui_quadro(Nome_novo_atributo_1,valor,superclasse(atributos_3)) e
  pega_faceta(Atributo_1,texto(T)) e
  inclui_quadro(Nome_novo atributo_1,valor,texto(T)) e
```

```
  pega_faceta(Atributo_1,cardinalidade_max(C_max)) e
  inclui_quadro(Nome_novo_atributo_1,valor,cardinalidade_max(C_max)) e
  pega_faceta(Atributo_1,cardinalidade_min(C_min)) e
  inclui_quadro(Nome_novo_atributo_1,valor,cardinalidade_min(C_min)) e
  pega_faceta(Atributo_1,dominio(Tipo_dominio_D)) e
  inclui_quadro(Nome_novo_atributo_1,valor,dominio(Tipo_dominio_D)) e
  liga_quadros(Nome novo atributo_2,Nome_novo_atributo_1) e
  inclui_quadro(Nome_novo_atributo_2,valor,membro_de(Nome_nova_entidade_Y)) e
  inclui_quadro(Nome_novo_atributo_2,valor,superclasse(Nome_novo_atributo_1)) e
  pega_faceta(Atributo_1,texto(T)) e
  inclui_quadro(Nome_novo_atributo_2,valor,texto(T)) e
  pega_faceta(Atributo_1,cardinalidade_max(C_max)) e
  inclui_quadro(Nome_novo_atributo_2,valor,cardinalidade_max(C_max)) e
  pega_faceta(Atributo_1,cardinalidade_min(C_min)) e
  inclui_quadro(Nome_novo_atributo_2,valor,cardinalidade_min(C_min)) e
  pega_faceta(Atributo_1,dominio(Tipo_dominio_D)) e
  inclui_quadro(Nome_novo_atributo_2,valor,dominio(Tipo_dominio_D)) e
  guarde integrou(Atributo_1) e
  guarde integrou(Atributo_2).

(a,II) #
se
  assercao f_map(Nome_nova_entidade_X_Y,Nome_nova_entidade_X,
       Nome_nova_entidade_Y,Atributo_1,Atributo_2,Resp2) e
  nao assercao integrou(Atributo_1) e
  nao assercao integrou(Atributo 2) e
  (Resp2 == 1 ou (Resp2 == 2 ,assercao estrategia_1)
entao
  transforma nome([Atributo_1,'_',Atributo2],Nome_novo_atributo) e
  guarde novo_atributo(Nome_novo_atributo) e
  liga_quadros(Nome novo_atributo,atributos_3) e
  inclui_quadro(Nome_novo_atributo,valor,membro_de(Nome_nova_entidade_X_Y)) e
  inclui_quadro(Nome_novo_atributo,valor,superclasse(atributos_3)) e
  pega_faceta(Atributo_1,texto(T)) e
  inclui_quadro(Nome_novo_atributo,valor,texto(T)) e
  pega_faceta(Atributo_1,cardinalidade max(C_max)) e
  inclui_quadro(Nome_novo_atributo,valor,cardinalidade_max(C_max)) e
  pega_faceta(Atributo_1,cardinalidade_min(C_min)) e
  inclui_quadro(Nome_novo_atributo,valor,cardinalidade_min(C_min)) e
  pega_faceta(Atributo_1,dominio(Tipo_dominio_D)) e
  inclui_quadro(Nome_novo_atributo,valor,dominio(Tipo_dominio_D)) e
  guarde integrou(Atributo_1) e
  guarde integrou(Atributo_2).
(a,12) #
se
  assercao f_map(Nome_nova_entidade_X_Y,Nome_nova_entidade_X,
          Nome_nova_entidade_Y,Atributo_1,Atributo_2,Resp2) e
  nao assercao integrou(Atributo_1) e
  nao assercao integrou(Atributo_2) e
  Resp2 == 3 e
  assercao estrategia_1
entao
  transforma_nome([Atributo_2,'_',Atributo_2],Nome_novo_atributo) e
  guarde novo_atributo<Nome_novo_atributo) e
```

```
 liga_quadros(Nome_novo_atributo,atributos_3) e
 inclui_quadro(Nome_novo_atributo,valor,membro_de(Nome_nova_entidade_X_Y)) e
 inclui_quadro(Nome_novo_atributo,valor,superclasse(atributos_3)) e
 pega_faceta(Atributo_2,texto(T)) e
 inclui_quadro(Nome_novo_atributo,valor,texto(T)) e
 pega_faceta(Atributo_2,cardinalidade_max(C_max)) e
 inclui_quadro(Nome_novo_atributo,valor,cardinalidade_max(C_max)) e
 pega_faceta(Atributo_2,cardinalidade_min(C_min)) e
 inclui_quadro(Nome_novo_atributo,valor,cardinalidade_min(C_min)) e
 pega_faceta(Atributo_2,dominio(Tipo_dominio_D)) e
 inclui_quadro(Nome novo_atributo,valor,dominio(Tipodominio_D)) e
 guarde integrou(Atributo_2) e
 guarde integrou(Atributo_2).

(a,13) #
se
 assercao f_map(Nome_nova_entidade_X_Y,Nome_nova_entidade_X,
          Nome_nova_entidade_Y,Atributo_1,Atributo_2,Resp2) e
 nao assercao integrou(Atributo_1 e
 nao assercao integrou(Atributo_2) e
 ((Resp2 == 4 , assercao estrategia_1) ou (Resp2 == 4 ,
 assercao estrategia_4))
entao
 transforma_nome([Atributo_1,'_',Atributo_2],Nome_novo_atributo) e
 guarde novo_atributo(Nome_novo_atributo) e
 guarde novo_atributo(Nome_novo_atributo_1) e
 guarde novo_atributo(Nome_novo_atributo_2) e
 liga_quadros(Nome_novo_atributo,atributos_3) e
 inclui_quadro(Nome_novo_atributo,valor,membro_de(Nome_nova_entidade_X_Y)) e
 inclui_quadro(Nome_novo_atributo,valor,superclasse(atributos_3)) e
 pega_faceta(Atributo_1,texto(T)) e
 inclui_quadro(Nome_novo_atributo,valor,texto(T)) e
 pega_faceta(Atributo_1,cardinalidade_max(C_max)) e
  inclui_quadro(Nome_novo_atributo,valor,cardinalidade_max(C_max)) e
 pega_faceta(Atributo_1,cardinalidade_min(C_min))l e
  inclui_quadro(Nome_novo_atributo,valor,cardinalidade_min(C_min)) e
 pega_faceta(Atributo_1,dominio(Tipo_dominio_D)) e
  inclui_quadro(Nome_novo_atributo,valor,dominio(Tipo_dominio_D)) e
 transforma_nome([Atributo_1,'_3'],Nome_novo_atributo_1) e
 transforma_nome([Atributo_2,'_3'],Nome_novo_atributo 2) e
 liga_quadros(Nome_novo_atributo_1,atributos_3) e
 inclui_quadro(Nome_novo_atributo_1,valor,membro_de(Nome_nova_entidade_X)) e
 inclui_quadro(Nome_novo_atributo_1,valor,superclasse(atributos_3)) e
 pega_faceta(Atributo_1,texto(T)) e
 inclui_quadro(Nome_novo_atributo_1,valor,texto(T)) e
 pega_faceta(Atributo_1,cardinalidade_max(C_max)) e
 inclui_quadro(Nome_novo_atributo_1,valor,cardinalidade_max(C_max}) e
 pega_faceta(Atributo_1,cardinalidade_min(C_min)) e
 inclui_quadro(Nome_novo_atributo_1,valor,cardinalidade_min(C_min)) e
 pega_faceta(Atributo_1,dominio(Tipo_dominio_D)) e
 inclui_quadro(Nome_novo_atributo_1,valor,dominio(Tipo_dominio_D)) e
 liga_quadros(Nome_novo_atributo_2,atributos_3) e
 inclui_quadro(Nome_novo_atributo_2,valor,membro_de(Nome_nova_entidade_Y)) e
 inclui_quadro(Nome_novo_atributo_2,valor,superclasse(atributos_3)) e
```

```
  pega_faceta(Atributo_1,texto(T)) e
  inclui_quadro(Nome_novo_atributo_2,valor,texto(T)) e
  pega_faceta(Atributo_1,cardinalidade_max(C_max)) e
  inclui_quadro(Nome_novo_atributo_2,valor,cardinalidade_max(C_max)) e
  pega_faceta(Atributo_1,cardinalidade_min(C_min)) e
  inclui_quadro(Nome_novo_atributo_2,valor,cardinalidade_min(C_min)i e
  pega_faceta(Atributo_1,dominio(Tipo_dominio_D)) e
  inclui_quadro(Nome_novo_atributo_2,valor,dominio(Tipo_dominio_D)) e
  guarde integrou(Atributo_1) e
  guarde integrou(Atributo 2).

(a,14) #
se
  assercao f_map(Nome_nova_entidade_X_Y,Nome_nova_entidade_X,
     Nome_nova_entidade_Y,Atributo_1,Atributo_2,Resp2) e
  nao assercao integrou(Atributo_1) e
  nao assercao integrou(Atributo 2) e
  Resp2 \== 1 e
  assercao estrategia_2
entao
  transforma_nome([Atributo_1,'_3'],Nome_novo_atributo_1) e
  transforma_nome([Atributo_2,'_3'],Nome_novo_atributo_2) e
  guarde novo_atributo(Nome novo_atributo_1) e
  guarde novo_atributo(Nome_novo_atributo_2) e
  liga_quadros(Nome_novo_atributo_1,atributos_3) e
  inclui_quadro(Nome_novo_atributo_1,valor,membro_de(Nome_nova_entidade_X)) e
  inclui_quadro(Nome_novo_atributo_1,valor,superclasse(atributos3)) e
  pega_faceta(Atributo_1,texto(T)) e
  inclui_quadro(Nome_novo_atributo_1,valor,texto(T)) e
  pega_faceta(Atributo_1,cardinalidade_max(C_max)) e
  inclui_quadro(Nome_novo_atributo_1,valor,cardina3idade_max(C_max)) e
  pega_faceta(Atributo_1,cardinalidade_min(C_min)) e
  inclui_quadro(Nome_novo_atributo_1,valor,cardinalidade_min(C_min)) e
  pega_faceta(Atributo_1,dominio(Tipo dominio D)) e
  inclui_quadro(Nome_novo_atributo_1,valor,dominio(Tipo_dominio_D)) e
  liga_quadros(Nome novo atributo_2,atributos_3) e
  inclui_quadro(Nome_novo_atributo_2,valor,membro_de(Nome_nova_entidade_Y)) e
  inclui_quadro(Nome_novoatributo_2,valor,superclasse(atributos_3)) e
  pega_faceta(Atributo_1,texto(T)) e
  inclui_quadro(Nome_novo_atributo_2,valor,texto(T)) e
  pega_faceta(Atributo_1,cardinalidade_max(C_max)) e
  inclui_quadro(Nome_novo_atributo_2,valor,cardinalidade_max(C_max)) e
  pega_faceta(Atributo_1,cardinalidade_min(C_min)) e
  inclui_quadro(Nome_novo_atributo_2,valor,cardinalidade_min(C_min)) e
  pega_faceta(Atributo_1,dominio(Tipo_dominio_D)) e
  inclui_quadro(Nome_novo_atributo_2,valor,dominio(Tipo_dominio_D)) e
  guarde integrou(Atributo_1) e
  guarde integrou(Atributo_2).

(a,15) #
se
  assercao
     f_map(Nome_nova_entidade_X_Y,Nome_nova_entidade_X,
     Nome_nova_entidade_Y,Atributo_1,Atributo_2,Resp2) e
```

```
  nao assercao integrou(Atributo_1) e
  nao assercao integrou(Atributo_2) e
  Resp2 \== 1 e
  assercao estrategia_3
entao
  transforma nome([Atributo_1,'_',Atributo_2],Nome_novo_atributo) e
  guarde novo_atributo(Nome_novo_atributo) e
  guarde novo_atributo(Nome_novo_atributo_1) e
  guarde novo_atributo(Nome_novo_atributo_2) e
  liga quadros(Nome_novo_atributo,atributos_3) e
  inclui_quadro(Nome_novo_atributo,valor,membro_de(Nome_nova_entidade_X_Y)) e
  inclui_quadro(Nome_novo_atributo,valor,superclasse(atributos_3)) e
  pega_faceta(Atributo_1,texto(T)) e
  inclui_quadro(Nome_novo_atributo,valor,texto(T)) e
  pega_faceta(Atributo_1,cardinalidade_max(C_max)l e
  inclui_quadro(Nome_novo_atributo,valor,cardinalidade_max(C_max)) e
  pega_faceta(Atributo_1,cardinalidade_min(C_min)) e
  inclui_quadro(Nome_novo_atributo,valor,cardinalidade_min(C_min)) e
  pega_faceta(Atributo_1,dominio(Tipo_dominio_D)) e
  inclui_quadro(Nome_novo_atributo,valor,dominio(Tipo_dominio_D)) e
  transforma_nome([Atributo_1,'_3'],Nome_novo_atributo_1) e
  transforma_nome([Atributo_2,'_3'],Nome novo_atributo_2) e
  inclui_quadro(Nome_novo_atributo_1,valor,membro_de(Nome_nova_entidade_X)) e
  pega_faceta(Atributo_1,texto(T)) e
  inclui_quadro(Nome_novo_atributo_1,valor,texto(T)) e
  pega_faceta(Atributo_1,cardinalidade max(C_max)) e
  inclui_quadro(Nome_novo_atributo_1,valor,cardinalidade_max(C_max)) e
  pega faceta(Atributo_1,cardinalidade_min(C_min)) e
  inclui_quadro(Nome_novo_atributo_1,valor,cardinalidade_min(C_min)) e
  pega faceta(Atributo_1,dominio(Tipo_dominio_D)) e
  inclui_quadro(Nome_novo_atributo_1,valor,dominio(Tipo_dominio_D)) e
  inclui_quadro(Nome_novo_atributo_2,valor,membro_de(Nome_nova_entidade_Y)) e
  pega_faceta(Atributo_1,texto(T)) e
  inclui_quadro(Nome_novo_atributo_2,valor,texto(T)) e
  pega_faceta(Atributo_1,cardinalidade_max(C_max)l e
  inclui_quadro(Nome_novo_atributo_2,valor,cardinalidade_max(C_max)) e
  pega_faceta(Atributol,cardinalidade_min(C_min)) e
  inclui_quadro(Nome_novo_atributo_2,valor,cardinalidade_min(C_min)) e
  pega_faceta(Atributo_1,dominio(Tipo_dominio_D)) e
  inclui_quadro(Nome_novo_atributo_2,valor,dominio(Tipo_dominio_D)) e
  liga_quadros(Nome_novo_atributo_1,Nome_novo_atributo) e
  liga_quadros(Nome_novo_atributo_2,Nome_novo_atributo) e
  inclui_quadro(Nome_novo_atributo_1,valor,superclasse(Nome_novo_atributo)) e
  inclui_quadro(Nome_novo_atributo_2,valor,superclasse(Nome_novo_atributo)) e
  guarde integrou(Atributo_1) e
  guarde integrou(Atributo_2).

(a,16) #
se
  assercao
     f_map(Nome_nova_entidade_X_Y,Nome_nova_entidade_X,
     Nome_nova_entidade_Y,Atributo_1,Atributo_2,Resp2) e
  nao assercao integrou(Atributo_1) e
  nao assercao integrou(Atributo_2) e
```

```
  Resp2 == 2 e
  assercao estrategia_4
entao
  transforma_nome([Atributo_1,'_',Atributo_2],Nome_novo_atributo) e
  guarde novo_atributo(Nome_novo_atributD) e
  guarde novo atributo(Nome novo_atributo_i) e
  liga_quadros(Nome_novo_atributo,atributcs_3) e
  inclui_quadro(Nome_novo_atributo,valor,membro_de(Nome_nova_entidade X_Y)) e
  inclui_quadro(Nome_novo_atributo,valor,superclasse(atributos_3)) e
  pega_faceta(Atributo_1,texto(T)) e
  inclui_quadro(Nome_novo_atributo,valor,texto(T)) e
  pega_faceta(Atributo_1,cardinalidade_max(C_max)) e
  inclui_quadro(Nome_novo_atributo,valor,cardinalidade_max(C_max)) e
  pega_faceta(Atributo_1,cardinalidade min(C_min)) e
  inclui_quadro(Nome_novo_atributo,valor,cardinalidade_min(C_min)) e
  pega_faceta(Atributo_1,dominio(Tipo dominio_D)) e
  inclui_quadro(Nome_novo_atributo,valor,dominio(Tipo_dominio))) e
  transforma_nome([Atributo_1,'_3'],Nome_novo_atributo_1) e
  liga_quadros(Nome_novo_atributo_1,atributos_3) e
  inclui_quadro(Nome_novo_atributo_1,valor,membro_de(Nome_nova_entidade_X)) e
  inclui_quadro(Nome_novo_atributo_1,valor,superclasse(atributos_3)) e
  pega_faceta(Atributo_1,texto(T)) e
  inclui quadro(Nome_novo_atributo_1,valor,texto(T)) e
  pega_faceta(Atributo_1,cardinalidade_max(C_max)) e
  inclui_quadro(Nome_novo_atributo_1,valor,cardinalidade_max(C_max)) e
  pega_faceta(Atributo_1,cardinalidade_min(C_min)) e
  inclui_quadro(Nome novo_atributo_1,valor,cardinalidade_min(C_min)) e
  pega_faceta(Atributo_1,dominio(Tipo_dominio_D)) e
  inclui_quadro(Nome_novo_atributo_1,valor,dominio(Tipo_dominio_D)) e
  guarde integrou(Atributo_1) e
  guarde integrou(Atributo_2).

(a,17) #
se
   assercao f_map(Nome_nova_entidade_X_Y,Nome_nova_entidade_X,
      Nome_nova_entidade_Y,Atributo_1,Atributo_2,Resp2) e
   nao assercao integrou(Atributo_1) e
   nao assercao integrou(Atributo_2) e
  Resp2 == 3 e
  assercao estrategia_4
entao
  transforma_nome([Atributo_1,'_',Atributo_2],Nome_novo_atributo) e
  guarde novo_atributo(Nome_novo_atributo) e
  guarde novo_atributo(Nome novo_atributo_2) e
  liga_quadros(Nome_novo_atributo,atributos_3) e
  inclui_quadro(Nome_novo_atributo,valor,membro_de(Nome_nova_entidade_X_Y)) e
  inclui_quadro(Nome_novo_atributo,valor,superclasse(atributos_3)) e
  pega_faceta(Atributo_1,texto(T)) e
  inclui_quadro(Nome_novo_atributo,valor,te)ito(T)) e
  pega_faceta(Atributo_1,cardinalidade_max(C_max)) e
  inclui_quadro(Nome_novo_atributo,valor,cardinalidade_max(C_max)) e
  pega_faceta(Atributo_1,cardinalidade min(C_min)) e
  inclui_quadro(Nome_novo_atributo,valor,cardinalidade_min(C_min)) e
  pega_faceta(Atributo_1,dominio(Tipo_dominio_D)) e
```

```
 inclui_quadro(Nome_novo_atributo,Valor,dominio(Tipo_dominio_D)) e
 transforma_nome([Atributo_2,'_3'],Nome_novo_atributo_2) e
 liga_quadros(Nome_novo_atributo_2,atributos_3) e
 inclui_quadro(Nome_novo_atributo_2,valor,membro_de(Nome_nova_entidade_Y)) e
 inclui_quadro(Nome_novo_atributo_2,valor,superclasse(atributos_3)) e
 pega_faceta(Atributo_2,texto(T)) e
 inclui_quadro(Ncmenovo_atributo_2,valor,texto(T)) e
 pega_faceta(Atributo_2,cardinalidade_max(C_max)) e
 inclui_quadro(Nome_novo_atributo_2,valor,cardinalidade_max(C_max)) e
 pega_faceta(Atributo_2,cardinalidade_min(C_min)) e
 inclui quadro(Nome novo_atributo_2,valor,cardinalidade_min(C_min)) e
 pega_faceta(Atributo_2,dominio(Tipodominio_D)) e
 inclui_quadro(Nome_novoatributo_2,valor,dominio(Tipodominio_D)) e
 guarde integrou(Atributo_1) e
 guarde integrou(Atributo_2).

(a,18) #
se
 ((assercao integrar_atributos(Nome_nova_entidade_X,Entidade_X,Entidade_Y) ou
   assercao integrar_atributos(Nome_nova_entidade_X,Nome_nova_entidade_Y,
   Entidade_X,Entidade Y) ou assercao integraratributos(Nome_nova_entidade_X_Y,
     Nome_nova_entidade_X,Nome_nova_entidade_Y,Entidade_X,Entidade_Y) )) e
 atributo(Atributo) e
 pega_faceta(Entidade_X,atributos(Atributos_X)) e
 membro(Atributo,Atributos_X) e
 pega_faceta(Entidade_X,chave(Chave_1)) e
 nao membro(Atributo,Chave_1) e
 nao assercao integrou(Atributo)
entao
 transforma_nome([Atributo,'_3'],Nome_novo_atributo) e
 guarde novo_atributo(Nome_novo_atributo) e
 1iga_quadros(Nome_novo_atributo,atributos_3) e
 inclui_quadro(Nome_novo_atributo,valor,membro_de(Nome_nova_entidade_X)} e
 inclui_quadro(Nome_novo_atributo,valor,superclasse(atributos_3)) e
 pega_faceta(Atributo,texto(T)) e
 inclui_quadro(Nome_novo_atributo,valor,texto(T)) e
 !
 pega_faceta(Atributo,cardinalidade_max(C_max)) e
 inclui_quadro(Nome_novo_atributo,valor,cardinalidade_max(C_max)) e
 pega_faceta(Atributo,cardinalidade_min(C_min)) e
 inclui_quadro(Nome_novo_atributo,valor,cardinalidade_min(C_min)) e
 pega faceta(Atributo,dominio(Tipo dominio_D)) e
 inclui_quadro(Nome_novo_atributo,valor,dominio(Tipo_dominio_D)) e
 guarde integrou(Atributo).

(a,l9) #
se
 ((assercao integrar atributos(Nome_nova_entidade_Y,Entidade_X,Entidade Y) ou
   assercao integrar atributos(Nome_nova_entidade_X,
   Nome_nova_entidade_Y,Entidade_X,Entidade_Y) ou
 assercao integrar_atributos(Nome_nova_entidade_X_Y,
       Nome_nova_entidade_X,Nome_nova_entidade_Y,Entidade_X,Entidade_Y) )) e
 atributo(Atributo) e
 pega_faceta(Entidade_Y,atributos(Atributos_Y)) e
```

```
  membro(Atributo,Atributos_Y) e
  pega_faceta(Entidade_Y,chave(Chave_1)) e
  nao membro(Atributo,Chave_1) e
  nao assercao integrou(Atributo)
entao
  transforma_nome([Atributo,'_3'],Nome_novo_atributo) e
  guarde novo_atributo(Nome_novo_atributo) e
  liga_quadros(Nome_novo_atributo,atributos_3) e
  inclui_quadro(Nome_novo_atributo,valor,membro_de(Nome_nova_entidade_Y)) e
  inclui_quadro(Nome_novo_atributo,valor,superclasse(atributos_3)) e
  pega_faceta(Atributo,texto(T)) e
  inclui_quadro(Nome_novo_atributo,valor,texto(T)) e
  pega_faceta(Atributo,cardinalidade_max(C_max)) e
  inclui_quadro(Nome_novo_atributo,valor,cardinalidade_max(C_max)) e
  pega_faceta(Atributo,cardinalidade_min(C_min)) e
  inclui_quadro(Nome_novo_atributo,valor,cardinalidade_min(C_min)) e
  pega_faceta(Atributo,dominio(Tipo_dominio_D)) e
  inclui_quadro(Nome_novo_atributo,valor,dominio(Tipo_dominio_D)) e
  guarde integrou(Atributo).

(a,20) #
se
  assercao transportou(Nome_nova_entidade,Entidade) e
  atributo(Atributo) e
  pega_faceta(Entidade,atributos(Atributos)) e
  membro(Atributo,Atributos) e
  nao assercao transportou(Atributo)
entao
  guarde transportou(Atributo) e
  transforma_nome([Atributo,'_3'],Nome_novo_atributo) e
  guarde novo atributo(Nome_novo_atributo) e
  liga_quadros(Nome_novo_atributo,atributos_3) e
  inclui_quadro(Nome_novo_atributo,valor,superclasse(atributos_3)) e
  inclui_quadro(Nome_novo_atributo,valor,membro_de(Nome_nova_entidade)).
```

ANEXO 4

APLICAÇÃO C PARA INTERFACE GRÁFICA DE INTEGRAÇÃO (CÓDIGO FONTE COMPLETO DISPONÍVEL NO COPIN)

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
#include <graph.h>
#include <bios.h>
#include <math.h>
#include <string.h>
#include "defs.h"
#include "cont_mnu.h"
#include "lista2.c"
#include "altera.c"
#include "editor.c"

struct videoconfig vc;

char seta[] = { 16 };
char *buffer1;
char *buffer2;
char *buffer3;
char cabeca[] = {"SISTEMA    EDITOR    ARQUIVO :  <N0ME> DIA/MES/ANO"};
char erro[] = ("ERRO / AJUDA "};

main()
{
 int choice, escolha, crow, ccol, num,
 max; int tmode, vmode;

  _getvideoconfig(&vc);
  crow = (vc.numtextrows / 2) + 8;  /* 15 */
  ccol = (vc.numtextcols / 2) + 26;  /* 54 */

 /* SELECIONA MODO DE VIDEO */
 _setvideomode(_HRESBW);
 _clearscreen(_GCLEARSCREEN);
 _settextposition(0,0);
 _outtext(cabeca);
 _moveto(0,9);
 _lineto(639,9);
 _moveto(0,188);
 _lineto(639,188);
 _settextposition(25,0);
  _outtext(erro);

  /* SELECIONA E DESVIA PARA 0 MENU ESCOLHIDO */
```

```
  for (;;) {
    choice = menu(crow, ccol,mnuMain);
    switch (choice) {
     case EDITAR :
      editar(crow,ccol);
      continue;
     case SALVAR : salvar();
      continue;
     case SAIR : sair();
      continue;
    case NAVEGAR : navegar ();
      continue: case CHECAR :
      checar (); continue; }

  _bios_keybrd(_KEYBRD_READ);
  _setvideomode(tmode); } }

void editar(row_ant, col_ant)
int row_ant, col_ant:
{
 int escolha, row, col;
 bufferl = (char *) malloc ((unsigned int)
      _imagesize (col_ant-2,row_ant-5,639,199));
 _getimage(col_ant-2, row_ant-5, 639, 199, buffer1);

 for (;:) {
  escolha = menu(row_ant-1, col ant-4, mnuEdicao);
  switch (escolha) {
       case INCLUIR :
           editor( 8, 2, 15, 25); incluir(row ant-
           2,col_ant-4); continue; case EXCLUIR :
           excluir(row_ant-2,col_ant-4); continue;
       case ALTERAR :
           alterar(row_ant-2,col_ant-4); continue;
       case SAIR_ED :
          _putimage(col_ant-2,row_ant-5,bufferl,_GPSET);
          free(bufferl); return:
    }
    bios_keybrd( KEYBRD_READ);
  }
  while (!kbhit());
}

void salvar () {}
void sair () {
    _setvideomode(_DEFAULTMODE);
    exit(0);
}

void navegar() {}
void checar() {}

void incluir(row_ant, col_ant)
int row_ant, col_ant;
```

```
{
 int escolha;
 buffer2 = (char *) malloc ((unsigned int)
        _imagesize (col_ant-2,row_ant-5,639,l99));
 _getimage(col_ant-2, row_ant-5, 639, 199, buffer2);

 for (;;) {
   escolha = menu(row_ant-2, col_ant-4, mnuIncExc);
   switch (escolha) (
       case ENTIDADE :
           inclui_entidade('n'); continue;
       case RELACIONAMENTO :
           inclui_relacionamento(); continue;
       case CATEGORIA :
           inclui_entidade('s'); continue;
       case SAIR_IE :
           _putimage(col_ant-2,row_ant-5,buffer2,_GPSET);

/* por enquanto */ free(buffer2);
           return;
    }
    _bios_keybrd(_KEYBRD_READ);
  }
  while (!kbhit());
}

void excluir(row_ant, col_ant)
int row_ant, col_ant;
{
 int escolha;
 buffer2 = (char *) malloc ((unsigned int)
        _imagesize (col_ant-2,row_ant-5,639,199));
 _getimage(col_ant-2, row_ant-5, 639, 199, buffer2);

 for (;;) {
   escolha = menu(row ant-2, col_ant-4, mnuIncExc);
   switch (escolha) (
       case ENTIDADE :
           exclui_entidade('n'); continue;
       case RELACIONAMENTO :
           exclui_relacionamento(); continue;
       case CATEGORIA :
           exclui entidade('s'); continue;
       case SAIR_IE :
           _putimage(col_ant-2,row_ant-5,buffer2,_GPSET);
           free(buffer2);
           return; }
           _bios_keybrd(_KEYBRD_READ);
  }
  while (!kbhit());
}

void alterar(row_ant, col_ant)
int row ant, col_ant;
```

```
{
int escolha;
buffer2 = (char *) malloc ((unsigned int)
        _imagesize (col_ant-2,row_ant-5,639,199));
  _getimage(col_ant-2, row ant-5, 639, 199, buffer2);
  for (;;) {
    escolha = menu(row_ant-2, col_ant-4, mnuAlt);
    switch (escolha) (
        case ALT_ENT :
            altera_ent(row ant-2,col_ant-4)! continue;
        case ALT_REL :
            altera_rel(row_ant-2,col_ant-4); continue;
        case ALT_CAT :
            altera cat(row ant-2,col_ant-4); continue;
        case ALT_ATR :
            altera_atr(row_ant-2,col_ant-4); continue;
        case SAIR_A :
           _putimage(col_ant-2,row_ant-5,buffer2,_GPSET);
           /* por enquanto */ free(buffer2);
           return;
     }
     _bios_keybrd(_KEYBRD_READ);
  }
  while (!kbhit());
}

void altera_ent(row_ant, col_ant)
int row_ant, col_ant;
{
  int escolha;
  buffer3 = (char *) malloc ((unsigned int)
        _imagesize (col_ant-2,row_ant-5,639,199));
  _getimage(col_ant-2, row_ant-5, 639, 199, buffer3);
  for (;;) {
    escolha = menu(row_ant-2, col_ant-4, anuAltEntCat);
    switch (escolha) { case ATRIB :
                        altera_entidade('n', 'n'); continue;
                    case CHAVE :
                        altera_entidade('n', 's'); continue;
                    case SA1R_AEC :
    _putimage(col_ant-2,row ant-5,buffer3,_GPSET);
    free(buffer3);
    return;
     }
     _bios_keybrd(_KEYBRD_READ);
  }
  while (!kbhit());
}
void altera_rel(row_ant, col_ant)
int row_ant, col_ant;
{
  int escolha;
  buffer3 = (char *) malloc ((unsigned int)
        _imagesize (col_ant-2,row_ant-5,639,199));
```

```
  _getimage(col_ant-2, row_ant-5, 639, 199, buffer3);
  for (;;) {
    escolha = menu(row_ant-2, col_ant-4, mnuAltRel);
    switch (escolha) { case GRAU :
            altera_relacionamento('g');
            continue;
        case ATRIBUTO :
            altera_relacionamento('a');
            continue;
        case ENT_PART :
            altera_relacionamento('e');
            continue;
        case SAIR_AR :
           _putimage(col_ant-2,row_ant-5,buffer3,_GPSET);
           free(buffer3):
           return; }
    _bios_keybrd(_KEYBRD_READ);
   }
   while (!kbhit());
}
void altera_cat(row_ant, col_ant)
int row_ant, col ant;
{
  int escolha;
  buffer3 = (char *) malice ((unsigned int)
        _imagesize (col_ant-2,row_ant-5,639,199));
  _getimage(col_ant-2, row_ant-5, 639, 199, buffer3);
  for (:;) {
    escolha = menu(row_ant-2, col_ant-4, mnuAltEntCat);
    switch (escolha) { case ATRIB .:
            altera_entidade('s', 'n');
            continue;
        case CHAVE :
            altera entidade('s', 's');
            continue;
        case SAIR_AEC :
            _putimage(col_ant-2,row_ant-5,buffer3,_GPSET);
            free(buffer3);
            return; }
    _bios_keybrd(_KEYBRD_READ);
   }
   while (!kbhit());
}
void altera_atr(row ant, col_ant)
int row_ant, col_ant;
{
  int escolha;
  buffer3 = (char *) malloc ((unsigned int)
         _imagesize (col_ant-2,row_ant-5,639,199));
  _getimage(col_ant-2, row_ant-5, 639, 199, buffer3);
  for (;;) {
    escolha = menu(row_ant-2, col_ant-4, mnuAltAtr);
    switch (escolha) {
        case MEMBRQ :
```

```
            altera_atributo('n', 'm'):
            continue;
         case CARD_MIN :
            altera_atributo('n , 'i');
            continue;
         case CARD_MAX :
            altera_atributo('n , 'x');
            continue;
         case T_DOMINIO :
            altera_atributo('n', 't');
            continue;
         case DOMINIO :
            altera__atributo('n', 'd');
            continue;
         case SAIR_AA :
            _putimage(col_ant-2,row_ant-5,buffers,_GPSET);
            free(buffer3);
            return; }
    _bios_keybrd(_KEYBRD_READ);
   }
   while (!kbhit());
 }

/* COLOCA MENU NA TELA */
int menu(row, col, items) int
row, col; char *items[];
  int i, num, max = 2, prev, curr = 0, choice;
  int litem[25];  /* talvez aumentar este limite */
  long bcolor;
  cursor(TCURSOROFF);
  for (num = 0; items[num]; num++)  {
    litem[num] = strlen(items[num]);
    max = diteaCnum] > max} ? litem[num] : max;
  }
  max += 2;
  row -= num / 2; col -= max / 2;
   rectangle(_GBORDER, 8*col-l, 8*row-2,
   8*col+8*max,8*row+8*num};
   col++;
   row++;
  for (i = 0; i < num; ++i)  ( if (i == curr)  {
     _settextposition(row, col):
     _outtext(" "); } else  {
       _settextposition(row, col);
       _outtext(seta);
      }
     itemize(row+i, col, items[i], max - litem[i]);
   }
   for (;;)  (
     switch ((_bios_keybrd(_KEYBRD_READ) & OxffOO) >> 8)  (
        case UP    :
            prev = curr;
            curr = (curr > 0) ? (--curr % num) : num-1;
            break;
```

```
        case  DOWN  ;
            prev = curr;
            curr = (curr < num) ? \++curr % num) : 0;
            break;
        case  ENTER  :
            _settextposition(row+curr, col);
            _outtext(" ");
            return(curr); default;
            continue;
     }
      _settextposition(row+curr, col);
      _outtext(seta);
     itemize(row+curr, col, items[curr], max - litem[curr]);
     _settextposition(row,w+prev, col); _outtext(" ");
     itemize(row+prev, col, items[prev], max - litem[prev]):
   }
}

void itemize(row, col, str, len) int
row, col, len;
char str[]; {
  char temp[80];
  _settextposition(row, ++col);
  _outtext(str); memset(temp, ' ', len--); temp[len] = NULL;
  _outtext(temp);
}
unsigned  cursor(value)
unsigned value;

{
 union REGS inregs, outregs;
 int ret;
 inregs.h.ah = 3;
 inregs.h.bh = 0;
 int86(0x10, &inregs, &outregs);
 ret = outregs.x.cx; inregs.h.ah = 1;
 inregs.x.cx = value;
 int86(0x10, &inregs, &outregs);
 return(ret);
```