Data de Depósito... 05 / 08 / 99

Assinatura: *Flávio de Sá Junior*

**Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação e Estatística**

**Método para Projeto de Aplicativos
centrados em BDOO utilizando Abstração
de Generalização¹**

Larissa Pinca Sarro Gomes

**Orientação:
Prof. Dr. Caetano Traina Junior**

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP – como parte dos requisitos para obtenção do título de Mestre em Ciências de Computação e Matemática Computacional

ICMC-USP, São Carlos.

Agosto de 1999

¹ Este trabalho teve apoio financeiro da CAPES.

*“Aos meus pais Vlaudir e Vera e
ao meu marido Ronaldo”*

Agradecimentos

Ao meu orientador Prof. Dr. Caetano Traina Júnior pela orientação e confiança durante todo o desenvolvimento do trabalho.

Ao Prof. Dr. João Eduardo Ferreira pelo incentivo e colaboração, fundamental para a realização deste trabalho.

À Prof. Dra. Renata P. de Mattos Fortes pela amizade e apoio, especialmente na fase final do trabalho.

À Prof. Dra. Rosely Sanches e à Prof. Dra. Agma J. M. Traina pela amizade e apoio.

À CAPES pelo apoio financeiro.

Ao meu irmão Gustavo, que sempre esteve na “torcida” por mim.

Aos amigos do Grupo de Banco de Dados e Imagens, em especial ao Mauro, Enzo, Roberto e Fábio, e à amiga Marisa.

À todos os funcionários do ICMC-USP, em especial ao “Sr. Zé” da portaria, sempre disposto a ajudar a todos.

SUMÁRIO

Lista de Figuras.....	i
Lista de Tabelas.....	iii
Resumo.....	iv
Abstract.....	v
Capítulo 1 - Introdução.....	01
Capítulo 2 - Modelo de Dados Conceituais.....	04
2.1 Considerações Iniciais.....	04
2.2 Modelos de Dados.....	04
2.2.1 Modelo Entidade Relacionamento Estendido (ME-RX).....	06
2.2.2 O Modelo de Objetos da Metodologia IDEA.....	07
2.2.3 Modelo SIRIUS.....	13
2.2.4 Modelo ITASCA.....	23
2.3 Metodologias e Métodos para o Desenvolvimento de Sistemas.....	26
2.3.1 Metodologia IDEA.....	26
2.3.2 O Método OMT.....	33
2.3.3 O Método Fusion.....	34
2.4 Comparação das Abstrações Suportadas pelos Modelos de Dados.....	36
2.5 Considerações Finais.....	37
Capítulo 3 - Opções de Mapeamento para o Modelo Relacional.....	39
3.1 Considerações Iniciais.....	39
3.2 Mapeamento do Modelo Entidade Relacionamento Estendido para o Modelo Relacional.....	40
3.3 Mapeamento das Especificações Chimera para o Modelo Relacional.....	45
3.3.1 Mapeamento do Esquema.....	46
3.3.2 Mapeamento de Regras Dedutivas.....	49
3.3.3 Mapeamento de Regras Ativas.....	49
3.4 Considerações sobre o Mapeamento.....	50

3.5 Considerações Finais.....	52
-------------------------------	----

Capítulo 4 - Método para o Mapeamento da Abstração de Generalização 53

4.1 Considerações Iniciais.....	53
4.2 O Modelo Conceitual Escolhido.....	54
4.2.1 Tipo Genérico e Tipo Especializado.....	55
4.2.2 Os Atributos dos Tipos de Objetos Envolvidos.....	55
4.2.3 As Restrições para a Generalização.....	56
4.2.4 Predicados para as Especializações.....	57
4.2.5 Regras de Inicialização de Objetos de Tipo Específico.....	58
4.2.6 Critérios da Generalização.....	58
4.2.7 Propriedades para a Generalização.....	59
4.2.8 A Estrutura Geral.....	61
4.3 Definição de Dez Opções para o Mapeamento da Generalização/Especialização.....	63
4.3.1 Opção de Mapeamento 1 (OM1).....	64
4.3.2 Opção de Mapeamento 2 (OM2).....	66
4.3.3 Opção de Mapeamento 3 (OM3).....	68
4.3.4 Opção de Mapeamento 4 (OM4).....	70
4.3.5 Opção de Mapeamento 5 (OM5).....	72
4.3.6 Opção de Mapeamento 6 (OM6).....	73
4.3.7 Opção de Mapeamento 7 (OM7).....	75
4.3.8 Opção de Mapeamento 8 (OM8).....	76
4.3.9 Opção de Mapeamento 9 (OM9).....	78
4.3.10 Opção de Mapeamento 10 (OM10).....	79
4.4 Avaliação das Dez Opções de Mapeamento.....	80
4.5 Árvore de Decisão.....	83
4.6 Comparação do Método Proposto com os Principais Métodos Encontrados na Literatura.....	85
4.7 Abrangência das Opções de Mapeamento.....	86
4.8 Considerações Finais.....	88

Capítulo 5 - Protótipo de uma Ferramenta de Suporte ao Mapeamento....	90
5.1 Considerações Iniciais.....	90
5.2 A Ferramenta E ² SIRIUS.....	91
5.2.1 Definição de Objetos e Atributos.....	92
5.2.2 As Abstrações de SIRIUS Suportadas pela Ferramenta.....	96
5.3 Editor SIRIUS Versão 1.1.....	101
5.3.1 Plataforma E ² SIRIUS Versão 1.1.....	103
5.3.2 Modificações Realizadas na Nova Versão E ² SIRIUS.....	103
5.3.3 Os Novos Formulários de E ² SIRIUS para a Versão 1.1.....	104
5.4 Considerações Finais.....	110
 Capítulo 6 - Conclusão.....	 112
6.1 Considerações Iniciais.....	112
6.2 Contribuições deste Trabalho.....	113
6.3 Sugestões para Futuras Pesquisas.....	115
 Referências Bibliográficas.....	 117

Lista de Figuras

Figura 1 – Agregação no ME-RX.....	07
Figura 2 – Representação de uma classe com suas respectivas operações.....	08
Figura 3 – Ilustrações de Restrições: de Cardinalidade e de Integridade.....	10
Figura 4 – Exemplo de restrição <i>STATIC IMMEDIATE</i>	11
Figura 5 – Generalização no Modelo de Objetos IDEA.....	12
Figura 6 – Abstrações importantes no Modelo SIRIUS.....	13
Figura 7 – Representação de Objetos.....	14
Figura 8 – Representação de Atributos.....	15
Figura 9 – Relacionamento Binário entre Objetos.....	16
Figura 10 – Representação gráfica de Objetos e tipos de Objetos.....	17
Figura 11 – Exemplos de ocorrência de Abstração de Classificação.....	19
Figura 12 – Representação de Composição (Colônia de Objetos).....	20
Figura 13 – Notação e exemplo de ocorrência de Abstração de Generalização.....	22
Figura 14 - Exemplo de uma ocorrência de Abstração de Generalização.....	23
Figura 15 – Representação de uma superclasse C com seus respectivos atributos e subclasses.....	41
Figura 16 – Exemplo de uma especialização.....	42
Figura 17 – Estrutura geral de uma OAG.....	61
Figura 18 – Exemplos de OAG's	62
Figura 19 – Mapeamento de uma OAG segundo OM1.....	65
Figura 20 – Mapeamento de uma OAG segundo OM2.....	67
Figura 21 – Mapeamento de uma OAG segundo OM3.....	69
Figura 22 – Mapeamento de uma OAG segundo OM4.....	71
Figura 23 – Mapeamento de uma OAG segundo OM5.....	73
Figura 24 – Mapeamento de uma OAG segundo OM6.....	74
Figura 25 – Mapeamento de uma OAG segundo OM8.....	77
Figura 26 – Mapeamento de uma OAG segundo OM9.....	78
Figura 27 – Mapeamento de uma OAG segundo OM10.....	80
Figura 28 – Árvore de Decisão.....	84
Figura 29 – Formulário Objeto.....	92
Figura 30 – Formulário Define/Vincula atributo.....	93

Figura 31 – Formulário para definir características de atributos.....	94
Figura 32 – Formulário para vincular atributos a um objeto.....	95
Figura 33 – Formulário para editar o valor de um atributo de instanciação.....	97
Figura 34 – Formulário Habita Colônia.....	98
Figura 35 – Formulário Generalização Ampla.....	99
Figura 36 – Formulário Especialização Restrita.....	101
Figura 37 – Módulo de Mapeamento.....	102
Figura 38 – Novo formulário Generalização Ampla.....	103
Figura 39 – Formulário Definição de Propriedades de Mapeamento com a primeira propriedade ativa.....	105
Figura 40 – Definição da primeira propriedade de mapeamento.....	106
Figura 41 – Definição da segunda propriedade de mapeamento.....	106
Figura 42 – Definição da terceira propriedade de mapeamento.....	107
Figura 43 – Definição da quarta propriedade de mapeamento.....	108
Figura 44 – Definição da quinta propriedade de mapeamento.....	109
Figura 45 – Código SQL para OAG Empresa.....	110

Lista de Tabelas

Tabela 1 – Informações relevantes de uma empresa de transporte ferroviário.....	05
Tabela 2 – Características de atributos.....	15
Tabela 3 – Comparação entre os modelos de dados.....	37
Tabela 4 – Opções de mapeamento para especialização.....	43
Tabela 5 – Características Chimera e as respectivas implementações em ORACLE.....	50
Tabela 6 – Relação das propriedades de uma OAG.....	60
Tabela 7 – Dez opções de mapeamento.....	64
Tabela 8 – Avaliação das dez opções de mapeamento.....	81

Resumo

Este trabalho apresenta um método de apoio ao processo de mapeamento da abstração de generalização para o Modelo Relacional, o qual proporciona um conjunto completo de opções de mapeamento. A definição e avaliação destas opções de mapeamento foram realizadas considerando-se alguns aspectos da estrutura conceitual da abstração de generalização. Tais aspectos estão relacionados às restrições de sobreposição e participação, e em particular, ao atributo critério, o qual define a especialização de um tipo genérico em tipos específicos. Além disso, considerou-se outros fatores, representados por propriedades, que fazem parte das características básicas de uma ocorrência da abstração de generalização. Para conduzir os analistas e projetistas na escolha de uma opção de mapeamento apropriada, é apresentada uma árvore de decisão, que foi construída de acordo com as restrições e propriedades definidas, as quais são identificadas como relevantes em cada ocorrência da abstração de generalização. Para a validação do desenvolvimento teórico das técnicas de mapeamento que foram propostas, e também, para o tratamento completamente automatizado da abstração de generalização, desde a representação em um modelo orientado a objetos até sua representação relacional, foi construída uma ferramenta.

Abstract

This dissertation presents a method to support the generalization abstraction mapping process for relational model in order to offer a complete set of options for mapping. The definition and evaluation of these options were carried out by considering some aspects of conceptual structure, of the generalization abstraction. Such aspects are related to overlapping and disjoint restrictions, and, particularly, to attribute criterion which defines the specialization of a generic type into specific types. Furthermore, other factors were considered and represent by properties which are part of the basic characteristics of an occurrence of generalization abstraction. To conduct the analysts and designs to choose an appropriate option for the mapping a decision tree is presented. It was built according to the defined restrictions and properties identified as relevant in each occurrence of generalization abstraction. For the validation of the theoretical mapping techniques proposed here and for the completely automated treatment of generalization abstraction, from the representation in an object oriented model to its relational representation, a tool was developed.

Capítulo 1 – Introdução

Na tentativa de representar as situações do mundo real através de um modelo de dados, de maneira que todas as informações relevantes sejam capturadas, uma solução encontrada foi abstrair as informações. Abstrair significa representar apenas o que é importante para caracterizar e delimitar as propriedades, regras e comportamentos da informação em questão.

Em modelos de dados, Abstração é o nome dado a uma estrutura de representação de informação, com características e propriedades particulares. Sob esse ponto de vista, os modelos de dados são constituídos de uma coleção de abstrações semanticamente integradas (Biajiz, 1996a).

O Modelo Entidade Relacionamento e o Modelo Entidade Relacionamento Estendido, ainda são muito utilizados no desenvolvimento de bases de dados para aplicações tradicionais, mas apresentam certas deficiências em aplicações mais complexas, tais como projeto e desenvolvimento de bases de dados científicas, sistemas de informações geográficas e base de dados multimídia. Essas últimas aplicações, além de exigirem estruturas mais complexas para objetos, transações de longa duração e novos tipos de dados para armazenar imagens ou itens textuais grandes, apresentam também a necessidade de se definir operações específicas para aplicações que não são padrão (Traina, 1994).

Os modelos de dados orientados a objeto (OO) foram propostos para sanar as dificuldades encontradas no desenvolvimento dessas aplicações mais complexas. A abordagem orientada a objeto oferece flexibilidade para tratar algumas dessas exigências sem estar limitada ao tipo de dado e linguagem de consulta disponível em sistemas de base de dados tradicionais. Uma característica importante das bases de dados orientadas a objeto é o poder que o desenvolvedor possui para especificar, de forma não isolada, a estrutura de objetos complexos e as operações

que podem ser aplicadas a esses objetos (Zand, 1995). Assim, os modelos OO além de proporcionarem mecanismos de abstração comportamental, proporcionam também, mecanismos de abstração estrutural equivalentes aos proporcionados pelo ME-RX.

O suporte às características de um modelo OO, semanticamente rico, através de um gerenciador de objetos, beneficia, por um lado, a corretitude das aplicações quanto a verificação dos requisitos representados, permitindo maior rapidez e facilidade de implementação. Por outro lado, pode levar a uma implementação que não atende aos requisitos de desempenho e resposta de processamento em tempo adequado.

A disponibilização de gerenciadores de objetos com bom desempenho é uma meta que está sendo perseguida. No entanto, o cenário que se tem é a utilização de um modelo de dados semanticamente rico, para o projeto das aplicações, e o mapeamento das modelagens resultantes, a fim de representá-las em gerenciadores relacionais, com menor capacidade de representação, mas que apresentam um melhor desempenho computacional.

Assim, uma vez capturadas as informações relevantes de um domínio de aplicação, utilizando para isto um modelo de dados conceitual, tais informações devem ser representadas ou mapeadas em um modelo de dados lógico, com base em registros, em particular o Modelo Relacional, a fim de serem armazenadas em um gerenciador relacional.

Uma atenção especial deve ser dada no momento do mapeamento, pois o conjunto de construtores semânticos, de um modelo de dados conceitual, proporcionam uma capacidade de representação maior, de situações do mundo real, quando comparados com os construtores semânticos do Modelo Relacional. Além disso, alguns conceitos suportados pela tecnologia orientada a objetos e a tecnologia relacional são tratados de maneira diferente. Os objetos complexos definidos na modelagem OO, por exemplo, não podem ser mapeados diretamente, uma vez que os valores dos atributos, no Modelo Relacional, devem ser representados por tipos de dados primitivos. E ainda, os Modelos Relacionais não suportam o conceito de identidade de objetos como um conceito separado do estado do objeto.

Estas e outras restrições, devem ser tratadas de forma adequada, para que se perca o mínimo possível de semântica, no momento do mapeamento de um modelo de dados conceitual, para um modelo de dados lógico com base em registros.

Considerando estes aspectos, o principal objetivo deste trabalho é definir um método para o processo de mapeamento, o qual deve restringir as escolhas dos desenvolvedores em um conjunto efetivo de operações de mapeamento. Todas as considerações sobre o mapeamento, apresentadas neste trabalho, terão enfoque na abstração de generalização, uma vez que esta

abstração é importante e suportada em todos os modelos de dados. Além disso, o processo de mapeamento de um esquema deve ser efetuado em etapas, considerando um elemento semântico de cada vez. Logo, a abstração de generalização, em geral, deve ser mapeada nos passos iniciais do processo de mapeamento, pois tal abstração estrutura o conjunto de tipos de objetos de um esquema. Dessa forma, um método para o mapeamento de outras abstrações somente pode prosseguir depois de definido o mapeamento da abstração de generalização.

Evidentemente, os aspectos de uma modelagem conceitual que não podem ser mapeados para um modelo de dados lógico com base em registros, e toda a semântica perdida na representação da estrutura estática, durante o processo de mapeamento, deve ser recuperada pelo conjunto de regras e operações de acesso, que devem ser embutidas na estrutura dinâmica representada pela linguagem de programação, tanto do aplicativo quanto do acesso à base de dados. Considerando que neste trabalho serão tratados apenas os aspectos estáticos, quanto mais completo e automático for o mapeamento, menor a necessidade de manipulação de código na construção das aplicações, e portanto, maior a rapidez e facilidade de seu desenvolvimento, além de uma segurança maior de que a aplicação resultante atende aos requisitos modelados.

Com este intuito, iniciaremos nossos estudos pelas abstrações suportadas pelos modelos de dados orientados a objeto, dentre eles o modelo SIRIUS (Biajiz, 1996a, 1996b), o modelo ITASCA (Kim, 1989, 1990) e o modelo de objetos da Metodologia IDEA (Ceri & Fraternali, 1997; Monteleone, 1997), além de uma rápida revisão no Modelo Entidade Relacionamento (Chen, 1976) e suas extensões (Smith & Smith, 1977), que serão apresentados no capítulo 2, juntamente com uma comparação entre os modelos de dados revisados, com relação as abstrações que eles suportam.

No capítulo 3 são descritas as opções de mapeamento, encontradas na literatura pesquisada, das informações capturadas na modelagem, para o Modelo Relacional. O capítulo 4 apresenta a descrição de dez opções de mapeamento, que foram definidas neste trabalho, juntamente com algumas considerações relevantes, que devem ser verificadas no momento da escolha por uma delas, e exemplos de utilização. Também é proposta uma árvore de decisão, construída com base nas dez opções de mapeamento definidas, que permite a automatização do processo de mapeamento.

O capítulo 5 apresenta o protótipo de uma ferramenta, construída com base na árvore de decisão apresentada, a qual demonstra que o processo de mapeamento, desenvolvido no capítulo 4, pode ser completamente automatizado. E finalmente, o capítulo 6 apresenta as conclusões e contribuições deste trabalho.

Capítulo 2 – Modelos de Dados Conceituais

2.1 Considerações Iniciais

Analizando os modelos orientados a objeto, da literatura pesquisada, é possível identificar vários conceitos comuns entre eles. Por exemplo, os conceitos de identificador de objetos, classes e herança múltipla. Há também muitas semelhanças nas abstrações suportadas tanto pelos modelos orientados a objeto como pelos modelos semânticos.

Para uma melhor descrição destes conceitos em alguns métodos e modelos, este capítulo está organizado da seguinte forma: na seção 2.2 são apresentados os principais conceitos de um modelo de dados semântico e de modelos orientados a objeto, e são revisadas as abstrações suportadas por esses modelos. Na seção 2.3 são revisados os conceitos de uma metodologia e de métodos utilizados na modelagem orientada a objeto. Na seção 2.4 apresenta-se uma comparação das abstrações suportadas pelos modelos de dados revisados na seção 2.2; e as considerações finais são apresentadas na seção 2.5.

2.2 Modelos de Dados

Nesta seção fazemos uma descrição dos modelos ME-RX (Modelo Entidade Relacionamento Estendido) (Elmasri & Navathe, 1994), SIRIUS (Traina, 1994; Biajiz, 1996a, 1996b), ITASCA (Kim, 1989, 1990, 1993) e do modelo de objetos da Metodologia IDEA (Ceri & Fraternali, 1997), analisando as abstrações suportadas por cada modelo.

Para ilustrar os conceitos de cada modelo escolheu-se, como estudo de caso, um sistema de suporte a uma empresa de transporte ferroviário (Monteleone, 1997), cujo domínio do problema é controlar as atividades da empresa, que presta serviços, e naturalmente, precisa de um planejamento de suas atividades para garantir um bom atendimento a seus clientes.

A Tabela 1 descreve algumas informações relevantes, requeridas para este estudo de caso.

Tabela 1 – Informações relevantes de uma empresa de transporte ferroviário

Funcionário	Para cada funcionário deve-se armazenar seu nome, endereço, código, tarefa, qualificação, descanso e ausências.
Serviço	Para cada serviço oferecido pela empresa é necessário conhecer seu código, tipo, início e término do serviço, a localização (onde o serviço será realizado, uma vez que existem serviços que são realizados fora da empresa) e um <i>flag</i> , para indicar se o serviço é noturno.
Período	Para atender da melhor maneira seus clientes, a empresa divide o ano em períodos que são: dias letivos, finais de semana e feriados. Para cada período são armazenados um código, uma descrição, a data e horário de início e término de um período considerado válido.
Oficina	Para cada oficina, onde realizam-se as manutenções nos trens da empresa, deve-se armazenar um código que a identifique, um nome, a identificação dos trens que passaram por manutenções, e a identificação dos funcionários que trabalharam em cada manutenção.
Trajetos	Para cada trajeto, percorrido por um determinado trem, devem ser armazenados o código que o identifica, e os locais de partida e chegada do trem.
Trem	Para cada trem é importante estabelecer um código, o nível de importância (p.e., um trem do centro da cidade é mais importante que um trem da periferia), o número de funcionários para proporcionar o mínimo de conforto aos usuários, o período em que o trem é afastado de circulação, e um <i>flag</i> , que indica se o trem está atrasado, ou não.

2.2.1 Modelo Entidade Relacionamento Estendido (ME-RX)

O Modelo Entidade Relacionamento (ME-R) (Chen, 1976, 1979), desde que foi proposto, teve grande aceitação e foi bastante explorado. Esse modelo utiliza quatro construtores para a representação semântica: *conjuntos de entidades*, *conjuntos de relacionamentos*, *atributos de entidades* e *atributos de relacionamentos*.

Com esses quatro construtores semânticos é possível representar muitos esquemas da base de dados para aplicações tradicionais. Entretanto, algumas bases de dados exigem maior complexidade para representar com fidelidade uma situação do mundo real. Para isto, os desenvolvedores necessitaram incorporar novos conceitos semânticos ao ME-R.

Vários autores propuseram diversas extensões ao ME-R, incluindo algumas abstrações com regras bem definidas. Teoricamente, não existe limite para o número de abstrações que podem ser estudadas e incorporadas em um modelo de dados. Porém, um modelo torna-se mais difícil de ser compreendido e utilizado quanto maior for o número de abstrações incorporadas (Traina, 1994).

Atualmente, a designação Modelo Entidade Relacionamento Estendido (ME-RX) garante que, ao menos as abstrações de generalização/especialização e a abstração de agregação, sejam suportadas. O ME-RX incorpora tais abstrações, que foram propostas inicialmente por Smith & Smith (1977), e estendidas por vários autores, entre eles, Hammer & McLeod (1981).

Uma especialização $E = \{S_1, S_2, \dots, S_n\}$ é um conjunto de subclasses que possui a mesma superclasse G , isto é, G/S_i é um relacionamento de uma superclasse/subclasse para $i = 1, 2, \dots, n$. Podemos ainda dizer que G é uma generalização do conjunto de subclasses $\{S_1, S_2, \dots, S_n\}$ (Elmasri & Navathe, 1994).

Uma especialização E é total sempre que $(\bigcup_{i=1}^n S_i) = G$, caso contrário, será uma ocorrência de especialização parcial. Ainda, E é chamada mutuamente exclusiva sempre que $S_i \cap S_j = \emptyset$ para todo $i \neq j$, caso contrário, E será chamada de especialização sobreposta.

A decisão de como definir as restrições de cada ocorrência da abstração de generalização/especialização depende da semântica do mundo real e do que se pretende representar na modelagem.

A abstração de agregação corresponde a idéia de que, elementos de modelagem podem se associar formando outros elementos que representam essa associação. Com esta abstração é possível expressar, por exemplo, relacionamentos entre relacionamentos, pois com a agregação, os conjuntos de relacionamentos são tratados como conjuntos de entidades de nível superior

associados a tal relacionamento, podem ser vistos como um conjunto de entidades de alto nível, que é tratado da mesma maneira que qualquer outro conjunto de entidades.

Segundo Traina¹ (1996), toda abstração de agregação sempre é definida a partir de um único conjunto de relacionamento.

No exemplo apresentado na Figura 1, utilizou-se a notação apresentada em Traina (1996) para representar a entidade agregação denominada *PrestaçãoServiço*, a qual envolve o conjunto de relacionamentos *executa*, e os conjuntos de entidades *funcionário* e *serviço*. Os atributos desta agregação são *local* e *data*.

Figura 1: Agregação no ME-RX.

2.2.2 O Modelo de Objetos da Metodologia IDEA

A metodologia IDEA (Ceri & Fraternali, 1997) foi desenvolvida no contexto de projeto de base de dados, mas também no amplo contexto de engenharia de software orientada a objetos. Essa metodologia suporta as três fases clássicas da engenharia de software: análise, projeto e implementação; e proporciona também a prototipação, como uma fase intermediária entre o projeto e a implementação.

Durante a fase de análise, são construídos o modelo de objetos, e os modelos dinâmico e de aplicações. O modelo de objetos possui um conjunto de conceitos e notações, usadas para expressar: as classes; os relacionamentos entre as classes; e as abstrações importantes de um domínio da aplicação. Como nas abordagens orientadas a objetos, o objetivo central está na determinação de classes, objetos, e nos relacionamentos entre os objetos.

¹ TRAINA JR., C – Sistemas de Banco de Dados- Conceitos Gerais e Motivação. Notas do Curso de Banco de Dados, ICMC-USP, 1996.

Um objeto representa um conceito individual de uma aplicação. Pelo estudo de caso, descrito anteriormente, pode-se identificar, por exemplo, **funcionário** e **oficina** como objetos. Para que um objeto seja armazenado na base de dados, ele deve estar associado a um identificador, que é gerado pelo sistema e o identifica univocamente na base. Juntamente com seu identificador, o objeto possui uma coleção de valores associados, chamados de atributos. Ao objeto são também associadas operações, com seus respectivos parâmetros, que são realizadas de acordo com os atributos do objeto.

Todo objeto possui um conjunto de restrições com uma estrutura fixa e significativa. Nesta metodologia essas restrições são chamadas de restrições embutidas, as quais são listadas a seguir:

- Atributos *not null*: são os atributos cujos valores devem ser conhecidos. Eles são identificados pelo símbolo NN à esquerda do atributo;
- *Primary Keys*: são coleções de atributos que identificam univocamente cada elemento de uma classe. Cada classe deve ter pelo menos uma chave primária e está implícito que seu valor não deve ser nulo. Elas são denotadas por um círculo negro à esquerda do atributo;
- Atributos *Unique*: são coleções de atributos que devem possuir um valor único para cada elemento de uma classe. Cada classe pode ter vários atributos únicos e são denotados pelo símbolo U e representados à esquerda de todos os atributos que formam um conjunto único. Se existirem vários conjuntos de atributos único, deve-se usar a representação: $U_1, U_2, \dots, U_N$.

As coleções de objetos que possuem uma mesma estrutura são chamadas de classes de objetos. Um exemplo de classe é o conjunto de funcionários da empresa de transporte ferroviário. Pode-se representar a classe **funcionário** com seus atributos, restrições e operações como apresentado na Figura 2.

Figura 2: Representação de uma classe com suas respectivas operações

- **modQuali**: modifica a qualificação de um funcionário;
- **modPAusências**: modifica as ausências dos funcionários;
- **modTar**: modifica a tarefa desenvolvida pelo funcionário.

A associação entre as classes é chamada de relacionamento. Um relacionamento que conecta dois objetos é chamado de relacionamento binário, enquanto que os que conectam mais do que dois objetos, por exemplo n objetos, são chamados de relacionamentos n -ários. Como no caso dos objetos, os relacionamentos podem ter atributos, mas não possuem operações associadas.

A restrição sobre o número de objetos que podem ser associados em um relacionamento é representada através de restrições de cardinalidade e denotada por meio dos seguintes símbolos:

- **min...max**: denotando o número mínimo e máximo de relacionamentos em que um objeto deve participar;
- *****: denotando zero ou mais ocorrências, sendo que zero indica uma participação opcional do objeto;
- **+**: denotando uma ou mais ocorrências, ou seja, todo objeto deve participar pelo menos uma vez do relacionamento.

Uma atenção especial deve ser dada a relacionamentos com cardinalidade máxima “um”, como é o caso do relacionamento *prestar*, ilustrado na Figura 3, envolvendo as classes *serviço* e *período* da empresa de transporte ferroviário, onde tal relacionamento significa que serviços são prestados em um determinado período. De acordo com a descrição do problema, a empresa divide o ano em períodos que são: dias letivos, finais de semana e feriados. Se por algum motivo um desses períodos for eliminado, não há mais sentido manter os serviços prestados neste período. Logo, os serviços prestados com relação a este período também devem ser eliminados.

A Figura 3 também ilustra a restrição de integridade referencial. No modelo de objetos IDEA, são especificadas duas estratégias para reparar violações de integridade referencial:

- **Restrict (R)**: a operação *delete* que causa a violação é abortada;
- **Cascade deletion (CD)**: a operação *delete* que causa a violação se propaga à classe que participa do relacionamento com cardinalidade máxima igual a “vários”.

Figura 3: Ilustração de restrições: de cardinalidade e de integridade

As restrições de integridade são predicados que devem ser satisfeitos para validar instâncias da base de dados. As restrições embutidas e as restrições de cardinalidade, por exemplo, possuem um formato fixo. No entanto, existem outras restrições que possuem um formato genérico, permitindo a formulação de regras de integridade arbitrárias, que dependem de um domínio de aplicação específico.

Pode-se classificar as restrições de integridade genérica em **restrições estáticas e dinâmicas**.

As restrições estáticas são avaliadas sobre um estado simples da base, isto é, avaliam o conteúdo de objetos e relacionamentos, e podem ainda serem classificadas em:

- Restrições *static deferred*: são avaliadas no *commit* da transação. Sempre que uma restrição pré-estabelecida for violada, o *commit* falha e a transação é abortada;
- Restrições *static immediate*: são avaliadas durante a transação, e por isso, as violações são detectadas no momento em que ocorrem. Seu efeito é abortar a transação imediatamente após a violação.

O exemplo da Figura 4 ilustra uma restrição *static immediate* sobre a classe funcionário:

Figura 4: Exemplo de restrição *STATIC IMMEDIATE*.

Já as restrições dinâmicas, comparam dois estados da base que são o estado inicial e o estado final, e também podem ser classificadas em:

- Restrições *dynamic deferred*: também são avaliadas no *commit* comparando o estado inicial e o estado final definidos em uma transação. Normalmente, estas restrições não são suportadas por sistemas comerciais pois é difícil ter acesso ao estado inicial de uma transação;
- Restrições *dynamic immediate*: são avaliadas durante a transação comparando o estado inicial e o estado final de operações específicas da transação.

Além de identificar uma restrição de integridade genérica como estática ou dinâmica, é importante também identificar o escopo das restrições. As restrições definidas no contexto de uma classe específica são chamadas restrições *targeted*.

Neste caso, uma restrição é um predicado que verifica o estado de um objeto *target* individual. O predicado pode também, fazer referências a outros objetos, pertencentes a classes diferentes, que são ligados por relacionamentos. Essas restrições que envolvem mais de uma classe são chamadas restrições *untargeted*.

□ Abstração de generalização

A única abstração que o modelo de objetos IDEA suporta é a generalização. Portanto, quando identifica-se outras abstrações, deve-se representá-las através de relacionamentos que caracterizam tais abstrações (Ceri & Fraternali, 1997).

A generalização pode ser complexa, contendo uma hierarquia de generalização com múltiplos níveis, na qual uma mesma entidade pode fazer o papel de superclasse, em um nível da hierarquia, e em outro nível, pode fazer o papel de subclasse.

Segundo Ceri & Fraternali (1997), quando uma superclasse possui várias subclasses, a generalização estabelece um mapeamento entre os objetos da superclasse, e os objetos das subclasses. Este mapeamento pode ser resumido em quatro propriedades:

- O mapeamento é **total** se todos os objetos da superclasse são mapeados em pelo menos um objeto da subclasse;
- O mapeamento é **parcial** se pelo menos um objeto da superclasse não é mapeado em nenhum objeto das subclasses;
- O mapeamento é **exclusivo** se nenhum objeto da superclasse for mapeado em mais de uma subclasse;
- O mapeamento é **sobreposto** se algum objeto da superclasse for mapeado em mais do que um objeto das subclasses.

A notação gráfica utilizada para representar uma generalização é um triângulo, conectando a superclasse às subclasses. Um triângulo preenchido representa uma generalização total e exclusiva, enquanto que um triângulo vazado representa uma generalização parcial e sobreposta.

Analisando o estudo de caso, apresentado no início deste capítulo, é possível identificar várias generalizações. A Figura 5 exemplifica uma dessas generalizações.

Figura 5 – Generalização no modelo de objetos IDEA

2.2.3 Modelo SIRIUS

O modelo SIRIUS foi desenvolvido com o propósito de atender às necessidades de aplicações de banco de dados não convencionais, onde os dados não apresentam uma homogeneidade estrutural (Traina, 1994; Biajiz, 1996a, 1996b). Inclui entre outros, os conceitos de: objetos, atributos, características de atributos, tipos de atributos e tipos de objetos. Os conceitos nesse modelo são organizados, semântica e sintaticamente, utilizando três abstrações fundamentais: abstração de classificação; abstração de associação; e abstração de generalização. A abstração de associação ocorre especializada em duas outras abstrações, que são as abstrações de composição e de agregação, como mostra a Figura 6:

Figura 6 – Abstrações importantes no modelo SIRIUS

□ Conceitos do Modelo

SIRIUS trata como objetos, as entidades do mundo real que podem ser identificadas pelos usuários, e utiliza o termo “objeto” diferenciadamente em três situações:

- quando o usuário reconhece um elemento do mundo real, o termo “objeto” é utilizado sem um qualificador;
- quando o sistema manipula os objetos e estes apresentam uma característica mais voltada à implementação, o termo utilizado é “objeto primário”. Pode-se considerar que os “objetos primários” não são transparentes ao usuário, mas que podem ser identificados por eles através de abstrações com outros objetos;
- o objeto que engloba os objetos primários e os objetos definidos pelo usuário é definido como “objeto conceito”.

Um objeto, que representa uma entidade do mundo real, é representado pelo seu nome, pelo nome do seu tipo e por atributos a ele associados. É importante observar que todo objeto tem exatamente um tipo.

A representação genérica de um objeto é mostrada na Figura 7_a. Na parte superior à esquerda do retângulo é colocado o nome do tipo deste objeto; a parte superior direita é utilizada para dados da abstração de composição (descrita a seguir); e a parte inferior é reservada para o nome do objeto. O tipo de um objeto pode ficar em branco apenas se o seu tipo for o próprio metatipo *tipo de objeto*, que é reconhecido e controlado pelo sistema. Na Figura 7_b, tem-se a representação de um objeto cujo nome é *mecânica* e o tipo é *curso*. Note que, neste exemplo, a parte superior direita está em branco, pois seu uso ainda não foi explicado, e na Figura 7_c tem-se um objeto *curso* onde o tipo de objeto é o metatipo *tipo de objeto*, que está omitido.

Figura 7: Representação de objetos

Associado aos objetos estão seus atributos, que têm o propósito de caracterizar a estrutura e o comportamento do objeto. Todo atributo possui um tipo e um valor, obedecendo a um domínio pré-definido, de acordo com o tipo de atributo a que pertence. Os tipos de atributos são associados aos objetos através de uma ligação monovalorada ou multivalorada. Um tipo de atributo monovalorado é aquele que possui apenas um valor para cada atributo instanciado e são representados por uma seta simples. Os tipos multivalorados possuem mais de um valor para cada instância, podendo ser organizados segundo estruturas de dados, tais como vetores, conjuntos ou listas; são representados com uma seta dupla e indica-se a estrutura por: { } para conjuntos, < > para lista e [] para vetor.

A Figura 8 mostra a representação de atributos. É importante notar que os atributos instanciados com valores são representados na parte inferior do retângulo e os não instanciados, na parte superior.

Figura 8- Representação de atributos

A cada tipo de atributo está associada uma característica e o tipo de dado que contém. A característica de um atributo define o conjunto de operações que um gerenciador de dados pode efetuar sobre esse atributo.

A Tabela 2 divide as características dos atributos em quatro classes: estática; dinâmica; interface; e estrutural. Para cada característica, são definidas uma lista com seus tipos de dados.

Tabela 2: Características de atributos

CLASSE DE CARACTERÍSTICA	CARACTERÍSTICA	SÍMBOLO	TIPO DE DADO
Estática	Texto	Tx	String, ASCII, RTF
	Número	Nu	inteiro, real
	Booleano	Bo	Booleano
Dinâmica	Regra	Rg	Trigger
	Procedimento	Pc	stored procedure, método
Interface	Visualização	Vs	DFM
	Som	So	WAV, VOC, AIFF
	Partitura	Pa	MIDI, MOD, SNG
	Imagem	Im	BMP, TIFF, J-PEG
	gráfico	Gr	WPG, CDR, WindowsMetaFile
Estruturais	estrutura de dado	Ed	vetor, lista, conjunto, dependência funcional
	tempo	Tm	absoluto, relativo
	tupla	Tp	
	atributo de atributo	Aa	
	relacionamento	Re	

□ Abstração de Agregação

Os atributos agregam-se para constituir um conjunto de dados, a fim de representar na modelagem, as informações do objeto do mundo real, ou para constituir outros atributos. A agregação de atributos, proporcionam também, a representação de relacionamentos entre os

objetos, ou seja, dois ou mais objetos são agregados através de um “objeto conceito”, formando relacionamentos. Com base nisso, a abstração de agregação caracteriza-se no modelo SIRIUS através da agregação de atributos e na agregação de objetos entre si formando relacionamentos.

Considerando a agregação de atributos, pode-se verificar que tal agregação ocorre de diversas maneiras, associando atributos simples em tuplas, listas, vetores ou conjuntos sem repetição. Pode-se dizer então, que tuplas, vetores, listas e conjuntos são estruturas de dados organizadas a partir da agregação de atributos.

Além disso, um atributo pode ter associado a si um outro atributo, o qual é denominado um atributo de atributo. Essa agregação é usada quando um determinado atributo possui características diferentes, para cada valor que lhe é atribuído, e além disso, não há interesse para o projeto particular, em que está inserido, que ele seja modelado como um objeto à parte.

A agregação entre dois objetos pode acontecer de duas maneiras. A primeira, mais imediata, consiste em referenciar em um objeto os demais objetos envolvidos. Essa maneira de indicar a agregação entre objetos não acrescenta nenhum conceito novo ao modelo, correspondendo apenas ao fato de que o valor de um atributo pode ser um objeto.

A segunda maneira institui o conceito de relacionamento, considerando a agregação entre objetos como um relacionamento. Nesse caso, os objetos envolvidos possuem uma referência para o “objeto relacionamento”. O tipo do relacionamento é caracterizado pelo tipo do atributo, e a característica desse tipo de atributo é a característica “relacionamento”. O relacionamento pode ter atributos associados, os quais são denominados atributos de relacionamento.

Na Figura 9, representa-se um relacionamento binário entre os objetos `of_zona_leste` e `SP101`, onde os tipos dos objetos são respectivamente `oficina` e `trem`.

Figura 9 – Relacionamento binário entre objetos

A notação representa a ligação dos atributos, `faz manutenção` e `recebe manutenção`, e cada um desses atributos pertencem a objetos diferentes, caracterizando uma

agregação entre eles. O círculo no centro, representa o objeto relacionamento (objeto primário²), e a chave que o envolve, indica que existe um conjunto de instâncias do atributo deste conjunto de relacionamento. Um exemplo de instância do relacionamento é o atributo data com valor 19/02/98.

Os relacionamentos ternários, quaternários, etc, também podem ser representados. Neste caso, em mais do que dois objetos existem atributos com características de relacionamento, que se associam pelo objeto primário representado pelo círculo central.

□ Abstração de Classificação

Todos os modelos de dados suportam a abstração de classificação, entretanto, este modelo trata de maneira mais abrangente tal abstração, uma vez que a classificação é tratada como um construtor semântico que permite criar hierarquias de classificação na modelagem.

Uma ocorrência de abstração de classificação dá-se através da classificação de um conjunto de objetos pelo seu tipo. Considerando que o modelo SIRIUS caracteriza um tipo como um objeto, então o processo de classificação pode repetir-se estabelecendo uma hierarquia de classificação, ou seja, um objeto é classificado pelo seu tipo, o qual pode ser classificado novamente até que atinja o tipo de objeto máximo da hierarquia.

Na Figura 10_a é representado o objeto trajeto cujo tipo é o metatipo tipo de objeto que está omitido, ou em outras palavras, ele é um tipo inicial cuja classificação é a própria abstração de classificação. Em 10_b é representado um objeto instanciável “S.Carlos/Araraquara”, cujo tipo é trajeto, e em 10_c representa-se um objeto não instanciável identificado por 10:45 e do tipo S.Carlos/Araraquara.

Figura 10 – Representação gráfica de objetos e tipos de objetos

² O termo objeto primário é utilizado para referenciar os objetos que são manipulados pelo próprio sistema, pois eles apresentam uma característica mais voltada à implementação.

Deve-se notar que trajeto é uma classificação de S.Carlos/Araraquara, que por sua vez é uma classificação de 10:45. Por outro lado, pode-se dizer que 10:45 é uma instância de S.Carlos/Araraquara que por sua vez é uma instância de trajeto.

É importante ressaltar que, assim como os objetos são classificados em tipos de objetos, atributos são classificados em tipos de atributos e colônias, que serão apresentadas no final desta seção, são classificadas em tipos de colônias.

Os atributos de classificação caracterizam a abstração de classificação pois modelam a semântica de suas ocorrências. Dessa forma, quem caracteriza a classificação são os atributos que estão associados aos objetos e obedecem a uma regra pré-definida. Assim, a maneira como um objeto é instanciado, é estabelecida através de um subconjunto de atributos de um objeto que é tipo e que são denominados atributos de classificação de Tipo. Através desses atributos, pode-se indicar quantos níveis a hierarquia terá e também, quais os atributos que pertencem em cada nível da hierarquia.

Em um diagrama, os atributos de classificação são representados por uma linha vertical acima do retângulo que representa o objeto, que nesse caso é um tipo. Os atributos do próprio objeto, denominados atributos de instância, são representados na linha abaixo do retângulo. Apenas objetos que são “tipo” possuem atributos de classificação.

Os atributo de instância podem ter um valor, serem nulos, ou indefinidos. É importante ressaltar que um atributo instanciado não pode ser repassado para suas instâncias.

Um atributo de classificação, além de poder ter um valor, tem um conjunto de propriedades que estabelecem critérios para a instanciação do objeto. O fato do atributo ter um valor, isto representa que esse valor é *default*, repassado para todas as instâncias do objeto ao qual está associado. Quanto às propriedades, os atributos de classificação indicam quantos níveis a hierarquia terá, assim como, quais atributos instanciam-se nos diversos níveis da hierarquia.

A Figura 11 mostra a notação que envolve as ocorrências da abstração de classificação. Na Figura 11_a mostra-se a representação de um tipo de objeto, o objeto trem, cujo tipo, que está em branco, é a própria abstração de classificação. Esse objeto não tem atributos de instância, apenas atributos de classificação. O número 1, precedendo o nome do atributo denominação, significa que esse atributo será instanciado em todas as instâncias de trem, ou seja, a denominação instanciada em SP101 é uma instância de trem. Os atributos precedidos pelo número 2 significa que serão instanciados nas instâncias das instâncias de trem. Podemos verificar isto pela Figura 11_c e 11_d, onde o atributo formação, instanciado em formação1 e formação2, é uma instância de SP101, que por sua vez, é uma instância de trem.

Se o número do nível onde o atributo será instanciado for colocado entre parênteses (), indica que o atributo será um identificador do objeto onde ele for instanciado.

Em SIRIUS, o conceito de identificador definido pelo usuário corresponde ao de atributo chave nos outros modelos. No entanto, não se admite que dois ou mais atributos possam ser concatenados para formar uma chave. Um único atributo deve ser suficiente para identificar univocamente o objeto dentro do seu contexto. Note-se que um atributo com característica de tupla pode ser um identificador.

Figura 11 – Exemplos de ocorrência de abstração de classificação

□ Abstração de Composição

O termo composição é utilizado para caracterizar situações em que objetos são realmente “compostos por” outros, como por exemplo, um trem é “composto por” vagões de 1a. classe, vagões da 2a. classe, mala postal, restaurante, etc.

A abstração de composição, tal como a abstração de agregação, é uma especialização da abstração de associação. Porém, esta abstração representa um significado diferente, pois cada instância da abstração de composição determina a existência de um objeto composto (**objeto abstrato**), e um conjunto de objetos que são “parte” desse objeto, chamados de **objetos detalhe**. Os objetos que são “parte” de um objeto composto são aqueles que foram definidos e reconhecidos pelo usuário.

É comum que os objetos compostos, e suas partes, formem um conjunto que tem propriedades distintas. Assim, o modelo SIRIUS aproveita esta idéia para definir o conceito de colônia de objetos. Uma colônia de objetos pode ser definida como um conjunto de objetos que compõem um objeto composto segundo um determinado aspecto. A idéia de aspecto representa o fato de um objeto poder ser composto de diferentes maneiras, disjuntas entre si.

A Figura 12_a mostra os relacionamentos implícitos que ocorrem entre os objetos envolvidos na ocorrência de uma abstração de composição: um objeto composto *constrange*³ uma colônia, na qual habitam objetos que são parte daquele objeto composto. É importante ressaltar que toda colônia tem um tipo.

Cada colônia pode representar uma composição lógica ou física de objetos. Impõem-se a restrição de que, todo objeto deve habitar fisicamente uma única colônia, e esse vínculo estabelece uma relação de dependência existencial entre o objeto e a colônia: se a colônia deixar de existir, todos os objetos que a habitam deixam de existir também. Por outro lado, um objeto pode habitar logicamente qualquer quantidade de colônias, sem causar nenhum tipo de dependência existencial.

Figura 12: Representação de composição (colônia de objetos).

A indicação do tipo de colônia que um objeto habita, é representada, no diagrama, através do nome da colônia na parte superior direita do retângulo que representa o tipo de objeto, sendo que, para todo tipo é necessário a indicação de exatamente uma colônia, na qual o objeto habita fisicamente.

³ A palavra *constrange* é utilizada no sentido de delimitar (envolver) um conjunto de elementos sob um determinado aspecto.

Por definição, existe uma colônia denominada “Global”, da qual pode haver apenas uma instância. A partir dela é definida a hierarquia de composição, a qual estabelece o contexto em que os objetos estão sendo compostos.

A Figura 12_b representa o fato de que o objeto trem constrange a colônia vagões e que o objeto 1ª.Classe constrange a colônia ambientes.

□ Abstração de Generalização

A abstração de generalização é caracterizada pelo fato de um tipo de objeto abstrair o comportamento e o significado de tipos de objetos mais específicos. Por outro lado, um tipo de objeto pode acrescentar mais detalhes a um outro tipo já existente e neste caso ocorre a especialização de um tipo de objeto.

Conceitualmente, não existe restrição para que quaisquer objetos possam ser especializados ou generalizados. Porém, quando o objeto não é um tipo, sua especialização ou generalização não é justificada, uma vez que recai na situação trivial de inserir ou eliminar atributos. Portanto, a abstração de generalização pode ser definida apenas sobre objetos que são tipo.

O modelo impõe que sejam definidos: um tipo que será especializado (supertipo); um conjunto de tipos especializados (subtipos); um critério de especialização/generalização; restrições para a especialização/generalização; um conjunto de regras de inicialização para cada tipo especializado/generalizado; e um predicado que indica a maneira pela qual os objetos do tipo genérico são especializados.

O critério que define a especialização é definido de duas maneiras: pelo valor de um único atributo, cujo domínio seja discreto e pré-definido; ou pelo resultado da avaliação de um predicado, sobre os valores de um ou mais atributos.

As restrições associadas a esta abstração são:

- **Sobreposição:** indica se um objeto do tipo genérico pode ser apenas um (disjunção) ou mais de um tipo específico (sobreposição);
- **Participação:** indica se todo objeto do tipo genérico deve necessariamente ser também de um tipo específico (participação total) ou não (participação parcial);
- **Categorização:** indica se todo objeto de um tipo específico deve necessariamente ser também do tipo genérico (categoria total ou subtipo) ou não (categoria parcial).

Na literatura esta última restrição é apresentada como um outro tipo de abstração, denominada abstração de categorização. No modelo SIRIUS, a categorização é definida como uma das restrições da abstração de generalização, a qual não será detalhada neste trabalho.

A Figura 13_a exemplifica a notação da abstração de generalização definida em SIRIUS. O supertipo é representado por um retângulo no topo da figura, e é ligado aos subtipos por um trapézio, no qual representa-se o atributo critério, o qual define a especialização, e as restrições de disjunção ou sobreposição, e participação total ou parcial.

Figura 13 - Notação e exemplo de ocorrências da abstração de generalização

O predicado e os valores de inicialização são representados no retângulo, junto com os subtipos. Nos casos em que o critério depende de um único atributo, cujo domínio é discreto e finito, o predicado é representado pelo valor do atributo critério; caso contrário, o predicado é representado por uma expressão ou um conjunto de valores.

Na Figura 13_b representa-se a especialização do objeto funcionário nos subtipos engenheiro e motorista. O trapézio, no centro, indica o atributo que estabelece o critério da especialização, denominado profissão, e as restrições de disjunção e participação parcial. Neste exemplo, o critério depende de um único atributo, cujo domínio é discreto e finito. Logo, o valor dos predicados para os subtipos engenheiro e motorista são exatamente engenheiro e motorista.

A outra maneira de representação do critério é mostrada na Figura 14, onde a especialização depende do resultado da avaliação do predicado numPolt, que significa o número de poltronas de um determinado trem. A partir do número de poltronas, os trens são divididos em trens de pequeno, médio e grande porte.

Figura 14 – Exemplo de uma ocorrência da abstração de generalização

2.2.4 Modelo ITASCA

O sistema de gerenciamento de bases de dados orientado a objetos ITASCA (Kim, 1989, 1990) foi baseado em uma série de pesquisas realizadas com protótipos, que tiveram início em 1985 em Austin, Texas - EUA. Esse sistema executa sobre sistemas UNIX e proporciona novas características, estendendo as características do sistema de gerenciamento de base de dados distribuída ORION-2.

O modelo ITASCA permite uma visão orientada a objetos que modela uniformemente qualquer entidade do mundo real como um objeto. Cada objeto tem um identificador único juntamente com um estado e comportamento. Os atributos representam o estado de um objeto enquanto que os métodos definem seu comportamento.

Os objetos, em ITASCA, podem ser representados desde os mais simples, como os tipos de dados (inteiro, caracter, etc.), até os mais complexos, que necessitam de uma estrutura de composição dos dados. Cada objeto ocupa uma porção de memória, a qual armazena sua estrutura de dados. Assim, para cada conjunto de valores, de cada objeto instanciado, existe uma porção da memória que armazena o estado do objeto em questão. O objeto denominado objeto primitivo, como um inteiro ou um caracter, não possui variável de instância, mas apenas um valor armazenado que também é um objeto. O objeto complexo, contém variáveis de instância, através das quais é possível estabelecer uma ligação com outros objetos (ITASCA, 1998).

Uma classe de objetos é definida como uma coleção de objetos que compartilham o mesmo conjunto de atributos e métodos. As subclasses derivam de classes existentes, logo, uma subclasse herda todos os atributos e métodos de sua superclasse. A hierarquia de classes torna fácil a classificação de objetos e diminui a redundância dos dados, desde que, as propriedades comuns, de diferentes classes, sejam definidas em superclasses comuns. O ITASCA suporta ainda, a herança múltipla, na qual uma subclasse pode derivar de mais de uma superclasse.

O esquema resultante, ou a definição da base de dados, é uma hierarquia de classes. Essa hierarquia é implementada através de um relacionamento entre as classes denominado **IS-A**. Como na maioria dos modelos de dados orientados a objeto, esse relacionamento caracteriza a representação da abstração de generalização e especialização entre classes.

Entretanto, este relacionamento não consegue representar a abstração de composição de objetos. Para isto, é necessário estabelecer o relacionamento **IS-PART-OF** entre as classes.

O conceito de objeto composto é definido, em ITASCA, como um objeto complexo ou uma hierarquia de agregação, e é capaz de atender as aplicações onde há necessidade de estabelecer uma soma entre as partes de um determinado projeto.

Um objeto composto tem um único objeto raiz, e este objeto faz referência a vários objetos filho através de instâncias variáveis. Os filhos também podem fazer referências a outros objetos filhos. As instâncias que constituem um objeto complexo pertencem a classes que também são organizadas em hierarquias. Esta hierarquia de classes é chamada de esquema de objetos compostos. Um esquema de objetos compostos consiste de uma única classe raiz e um conjunto de classes dependentes.

Em um objeto composto, nenhum objeto dependente pode ser referenciado por mais de um objeto. Entretanto, alguns objetos que fazem parte de uma hierarquia de composição podem ser referenciados por outros objetos, que não pertencem a tal hierarquia, através de relacionamentos.

Um outro conceito importante é o de atributo multivalorado. O modelo ITASCA propõe que todos os objetos sejam objetos-tupla. Assim, cada entrada para um objeto tupla é o valor de um atributo. Se o atributo for monovalorado então o valor é um único identificador de objetos (OID); se o valor for multivalorado, então tem-se um conjunto de identificadores de objetos (OID's).

O sistema de gerenciamento de bases de dados orientado a objetos ITASCA possui características encontradas em muitos outros sistemas de base de dados. Isto inclui, armazenamento persistente para dados e esquema, controle de concorrência, gerenciamento de transação e vários níveis de segurança. As características adicionais do gerenciador ITASCA incluem: modificação dinâmica do esquema, transações de longa duração, bases de dados privadas e compartilhadas, controle de versões distribuído, gerenciamento de transações distribuído, gerenciamento de consultas distribuído, migração de objetos, notificação de mudança distribuída e arquitetura extensível.

O gerenciador ITASCA armazena o esquema redundantemente em cada *site*, proporcionando um melhor desempenho do sistema. O esquema, armazena também o código na forma de métodos. O gerenciamento do esquema de atualização é automático para todos os *sites*. Isto inclui *sites* que foram desativados durante alguma mudança.

O gerenciador ITASCA suporta a modificação dinâmica do esquema, o que proporciona um ambiente flexível a mudanças, no sistema da base de dados. As definições de classes, estrutura de herança, especificação dos atributos e métodos podem ser modificados por usuários autorizados sem exigir a reinicialização da aplicação ou da base de dados. Os usuários autorizados podem adicionar ou remover atributos, e também, mudar o relacionamento de subclasses/superclasses. Podem ainda, adicionar ou remover partições da base de dados compartilhada. Tudo isto, pode ser realizado iterativamente, sem afetar outras partes da base de dados. É muito útil em ambientes onde a mudança na definição dos dados é relativamente freqüente.

Algumas pesquisas estão sendo realizadas com o propósito de criar uma interface entre a base de dados orientado a objeto ITASCA e as bases de dados relacionais (ITASCA, 1998; FAQ, 1998).

2.3 Metodologias e Métodos para o Desenvolvimento de Sistemas

Existem na literatura várias metodologias e métodos para proporcionar uma abordagem sistemática no desenvolvimento de softwares orientados a objeto, como Fusion (Coleman et al., 1994), OMT (Rumbaugh, 1991) e IDEA (Ceri & Fraternali, 1997).

Um destaque especial será dado a Metodologia IDEA pois, além de oferecer suporte adequado ao ciclo de vida orientado a objetos, proporciona uma orientação específica para aplicações centradas em banco de dados.

2.3.1 Metodologia IDEA

Como visto anteriormente, a metodologia IDEA suporta as fases de análise, projeto, prototipação e implementação, de sistemas de software centrado nos dados.

Uma característica importante da metodologia é a ênfase sobre as regras dedutivas e as regras ativas, que enriquecem significativamente a semântica suportada dentro de aplicações de banco de dados. Outra característica que diferencia essa metodologia é a ênfase na independência do conhecimento, cujo objetivo é obter um alto nível de independência entre as aplicações. Para isso, é preciso impor uma semântica coerente sobre todas as aplicações, o que talvez, seja a maior dificuldade para sistemas de software centrado nos dados.

Uma sugestão de IDEA é que, tanto quanto possível, o conhecimento semântico seja codificado dentro das aplicações, mas que as propriedades comuns dos dados a todas as aplicações, sejam especificadas fora das aplicações e representados no esquema, em particular dentro de objetos e regras.

□ A fase de Análise

A fase de análise é voltada para a especificação de requisitos, onde descrições informais de um problema são transformadas em descrições semi-formais, e para isto, IDEA utiliza modelos conceituais, com uma representação gráfica associada, para representar o comportamento esperado pelo sistema.

Durante a fase de análise são construídos o modelo de objetos (discutido na seção 2.2.2), o modelo dinâmico e o modelo de aplicações. IDEA constrói o modelo dinâmico utilizando uma

variação do formalismo visual statecharts de David Harel (Harel, 1987), o qual proporciona uma descrição dinâmica dos objetos em termos de estados admissíveis e transições de estados. Eventos típicos, que são representados no modelo dinâmico, são as atualizações na base de dados interna, os eventos externos, que exigem mudanças no ambiente de aplicação, e as chamadas às operações dos objetos.

O modelo de aplicações trata-se de uma especificação simples para descrever, a interação de uma aplicação, com os objetos da base de dados. As aplicações são identificadas por um nome, uma descrição, e um conjunto de objetos da base de dados, que podem ser lidos ou escritos juntamente com suas pré e pós-condições e com os eventos que podem ser produzidos.

□ A fase de Projeto

Na transição da fase de análise para a fase de projeto há um aumento no formalismo do modelo, da linguagem e das técnicas empregadas. O processo de projeto visa o suporte automatizado de alto nível, e por isso, necessita de fundamentos sólidos de linguagem formal. Dessa forma, durante o projeto, o modelo de objetos e dinâmico são gradualmente transformados em uma descrição textual escrita em Chimera, a linguagem conceitual da metodologia IDEA.

O processo de projeto inicia-se pelo projeto do esquema, seguido do projeto de regras dedutivas e regras ativas. O motivo pelo qual esta abordagem foi escolhida é dar prioridade ao conhecimento comportamental e estrutural associado às classes e objetos, seguido do conhecimento declarativo, o qual é definido por meio de regras dedutivas, e finalmente o conhecimento procedimental definido por meio de regras ativas.

1 - Projeto do Esquema

O projeto do esquema é uma seqüência de transformações que considera progressivamente as características do modelo de objetos e representa-as em Chimera. São considerados tipos, classes, generalizações, operações e relacionamentos.

O projeto de tipos, classes e hierarquia de generalizações constituem o primeiro núcleo do projeto do esquema. A seguir são modelados os relacionamentos por meio dos atributos da Chimera e, finalmente, as operações são projetadas para representar os componentes comportamentais do modelo de objetos.

A principal atividade no projeto do esquema é a transformação dos relacionamentos do modelo de objetos em atributos objeto-valorados que servem como referência entre os objetos. Esta transformação é exigida porque o modelo Chimera, assim como os sistemas de base de dados comerciais, não suportam relacionamentos explicitamente. Já nas bases de dados relacionais, os relacionamentos são representados adicionando atributos nas relações, onde os atributos são classificados sobre o mesmo domínio, ou introduzindo novas relações. Em qualquer dos dois mapeamentos é possível reconstruir a semântica do relacionamento através das operações *join*, conectando as tabelas de interesse.

O projeto do relacionamento, na metodologia IDEA, satisfaz as exigências dos modelos orientados a objeto, e podem ser considerados como um primeiro passo em direção a uma implementação relacional.

□ Projeto de Tipos, Classes e Hierarquia de Generalização

Durante o projeto de tipos, o modelo de objetos deve ser analisado no sentido de encontrar atributos ou parâmetros que são amplamente utilizados, justificando a criação de um tipo específico.

No projeto de classes, a especificação informal do tipo dos atributos, coletadas durante a fase de análise, é transformada em definições de tipo não ambíguas da linguagem Chimera.

O projeto de generalização tem como propósito avaliar as generalizações do modelo de objetos, com o intuito de verificar se os motivos que levaram a sua definição inicial, durante a análise, ainda são válidos.

A avaliação das generalizações pode ser iniciada verificando-se se toda subclasse serve para pelo menos um dos seguintes propósitos:

- a subclasse possui informações adicionais relevantes que devem ser armazenadas e que não são suportadas por suas superclasses;
- algumas operações, restrições ou *triggers* são aplicados diretamente aos objetos das subclasses.

Para verificar a consistência de hierarquias de generalização, as seguintes diretrizes podem ser aplicadas:

- (1) A hierarquia de generalização deve ser acíclica.
- (2) Quando uma classe herda atributos de múltiplas classes, estas devem ter um ancestral comum.

- (3) Quando um atributo é especializado em uma subclasse, o tipo deste atributo na subclasse deve ser um subtipo do atributo correspondente na superclasse.
- (4) Quando um parâmetro de entrada de uma operação é especializado em uma subclasse, o tipo deste parâmetro na subclasse deve ser um supertipo do atributo correspondente na superclasse.
- (5) Quando um parâmetro de saída de uma operação é especializado em uma subclasse, o tipo deste parâmetro na subclasse deve ser um supertipo do atributo correspondente na superclasse.

A regra 2 é importante pois proporciona, a cada hierarquia de generalização, a interpretação de um subconjunto de relacionamentos entre extensões de classes. Devido a esta regra, não há necessidade na Chimera de gerar um novo OID quando uma instância de um objeto é especializado: OID's são reusados em subclasses e indicam o fato que o mesmo objeto do mundo real pode ocupar qualquer subclasse. Os objetos de classes que não são relacionados por herança devem necessariamente corresponder a objetos distintos do mundo real.

As hierarquias totais e exclusivas introduzem uma partição de objetos da superclasse em objetos das subclasses. Já as hierarquias com sobreposição podem causar herança múltipla uma vez que queremos caracterizar as instâncias pertencentes a uma ou mais subclasses, e as hierarquias com participação parcial introduzem a necessidade de caracterizar precisamente os objetos que não são especializados. Portanto, segundo a metodologia IDEA, a existência de hierarquia de sobreposição com participação parcial constitui um argumento fraco para a manutenção de uma hierarquia de generalização no esquema.

□ Projeto do relacionamento

Em modelos orientados a objeto um relacionamento é normalmente representado por meio de atributos objeto-valorado, ou seja, uma classe possui um ponteiro para uma outra classe.

Nesta fase, os atributos devem ser analisados com o objetivo de identificar aqueles que são armazenados em uma classe, e os que são derivados de outras classes.

Existem várias alternativas para o mapeamento de relacionamentos. A abordagem utilizada por esta metodologia é a mesma do mapeamento do ME-RX para o Modelo Relacional, apresentada por Elmasri & Navathe (1994).

Em muitos casos é suficiente armazenar apenas um ponteiro para cada relacionamento, em outros, é conveniente armazenar pares de ponteiros em ambas as classes. Nesta última opção,

é importante verificar se não há redundância, e deve-se pagar pelo *overhead*. Em alguns outros casos, uma classe auxiliar deve ser introduzida.

□ Projeto de Operações

As operações em Chimera combinam expressões declarativas, utilizadas para representar pré-condições, e expressões procedimentais que são executadas somente quando as pré-condições estabelecidas forem satisfeitas. Pode-se dizer que o projeto das operações permite a representação da semântica das operações em um caminho formal, ainda que de fácil entendimento.

Para ilustrar a definição de operações, selecionou-se duas operações da classe *funcionário* já citadas anteriormente:

1. **modQuali**: modifica a qualificação de um funcionário;

```
define operation modQual(in Q: tipoQuali) for funcionário
condition true
action
    modify(funcionário.qualificação, Self, Q)
end;
```

2. **modPAusências**: modifica o período de descanso;

```
define operation modPAusências(in D: data) for funcionário
condition
    D in Self.ausências
action
    modify(funcionário.ausências, Self, Self.ausências - D)
end;
```

```
define operation modPAusências(in D: data) for funcionário
condition
    D not in Self.ausências
action
    modify(funcionário.ausências, Self, Self.ausências + D)
end;
```

O modelo de objetos, possui ainda, uma grande quantidade de informações que não foram representadas em Chimera. Em particular pode-se destacar:

- restrições de integridade genérica coletadas durante a fase de análise;
- restrições de integridade referencial coletadas durante a fase de análise e projeto do esquema;

- indicações de que certas subclasses são totais e/ou exclusivas coletadas durante a fase de análise;
- restrições de relacionamento inverso, coletadas durante o projeto do relacionamento.

Estas características do modelo de objetos serão convertidas em especificações Chimera no projeto de regras dedutivas.

2 - Projeto de Regras Dedutivas

O projeto de regras dedutivas concentra-se na geração de regras para a representação do conhecimento. Com base no seu uso, as regras dedutivas podem ser classificadas em duas categorias:

- **Regras para derivação dos dados:** especifica atributos derivados (são aqueles cujo valor pode ser derivado de outros atributos), visões e classes derivadas.
- **Restrições de integridade:** são classificadas em formato fixo e formato genérico. As restrições com formato fixo incluem atributos *notnull*, chave, *unique*, atributo inverso e restrições de integridade referencial. As restrições genéricas são classificadas em restrições *target*, *untarget* e restrições dinâmicas.

Muitas das restrições de formato fixo, definidas em Chimera, são suportadas diretamente por sistemas de base de dados convencionais; por exemplo, as restrições de chave, *notnull* e *unique*.

Os atributos inverso, a integridade referencial e as subclasses exclusivas, descritas durante a fase de análise, não são diretamente suportados pelos sistemas de base de dados convencionais, e podem ser facilmente transformados em expressões simples, suportadas pela linguagem Chimera. Por exemplo, no caso de atributos inverso, o desenvolvedor deve escolher o atributo que é derivado e projetar a regra de derivação.

Independente de sua aplicação, uma propriedade fundamental das regras dedutivas é que elas definem características declarativamente, baseada em seu significado. Elas proporcionam uma descrição do conhecimento e não exigem nenhum detalhe procedimental, sendo completamente independente de como serão implementadas.

No entanto, a declaratividade é também uma limitação das regras dedutivas. Elas não podem expressar características procedurais, que algumas vezes são necessárias, em particular, para expressar mecanismos para reparar violações de restrições de integridade. Por este motivo,

IDEA proporciona duas alternativas para descrever restrições de integridade: por meio de regras dedutivas ou por regras ativas.

Quando a declaratividade é desejada, as regras dedutivas são suficientes. Por outro lado, as regras dedutivas devem ser transformadas em ativas.

3 - Projeto de Regras Ativas

Esta fase destina-se a especificação de regras ativas e, para isto, deve-se considerar as restrições de integridade, especificações do processo de trabalho e o comportamento dinâmico dos objetos.

As regras ativas proporcionam um tratamento especial para várias situações, baseadas não apenas no estado atual da base de dados, mas também, na dinâmica que levou a base ao estado atual. Esta característica é fundamental para monitorar as mudanças na base de dados, a fim de detectar violações e então estabelecer uma estratégia reparadora, sem travar a transação.

Na metodologia IDEA, as regras reparadoras podem ser facilmente derivadas pela combinação da especificação declarativa das restrições de integridade, produzidas pelo projeto de regras dedutivas, e a indicação de uma ação reparadora, coletada durante a análise e expressas por meio de *triggers*.

□ A fase de Prototipação e Implementação

Terminada a fase de projeto, IDEA proporciona softwares de prototipação rápida para que a adequabilidade e conformidade com os requisitos sejam avaliadas pelos desenvolvedores e usuários, com o objetivo de melhorar a qualidade do projeto.

As regras dedutivas e as regras ativas serão o principal alvo da prototipação. O desenvolvedor deverá se concentrar no comportamento coletivo de regras complexas. Em sistemas baseado em regras o maior problema não é o desenvolvimento correto de regras individuais, mas sim a construção de coleções de regras, todas com um bom desempenho e que possam ser mantidas mesmo depois de mudanças nos requisitos.

A implementação é o processo de mapeamento das especificações Chimera em esquemas, objetos e regras de acordo com uma plataforma de base de dados. O processo é influenciado por características do ambiente de programação escolhido.

2.3.2 O Método OMT

O OMT - Object Modeling Technique (Rumbaugh, 1991) foi desenvolvido no início dos anos noventa. A partir desse modelo, houve uma evolução mais concreta dos modelos orientados a objeto.

A metodologia empregada em OMT segue os seguintes passos: a *análise*, onde são construídos o modelo de objetos, o modelo dinâmico e o modelo funcional; o *projeto* dividido em *projeto do sistema* e *projeto de objetos*; e a *implementação*.

□ A fase de Análise

O modelo de objetos é o modelo mais importante da fase de análise pois descreve a estrutura estática dos objetos, isto é, seus relacionamentos com outros objetos e seus atributos, fornecendo uma estrutura essencial para a construção do modelo dinâmico e funcional.

O modelo de objetos utiliza uma notação gráfica, desenvolvida para o OMT, a qual foi utilizada por outros modelos orientados a objetos (Booch, 1994; Coleman et al., 1994), depois de passar por algumas modificações.

As duas abstrações suportadas pelo modelo de objetos do OMT são a abstração de agregação e a abstração de generalização. A agregação é a “parte do todo” de um relacionamento, no qual os objetos componentes são associados para representar a totalidade. A generalização é uma conexão entre uma classe e, uma ou mais versões refinadas ou especializadas. A classe que será especializada é chamada superclasse, e cada versão de uma classe especializada é uma subclasse.

A generalização e a especialização podem ser entendidas como dois pontos de vista distintos de um mesmo relacionamento. A generalização refere-se ao fato de que superclasses generalizam as subclasses e na especialização, as superclasses são refinadas ou especializadas em subclasses.

Assim como acontece em outros processos de desenvolvimento, a análise do OMT é iterativa, ou seja, o modelo funcional, que é o último modelo a ser construído, pode revelar objetos e métodos extras, os quais precisam ser incorporados aos outros dois modelos, o modelo de dados e o modelo dinâmico.

A fase de análise estará completa quando todos os modelos forem consistentes e capturarem todos os requisitos exigidos pelo sistema.

□ A fase de Projeto e Implementação

A fase de projeto parte dos modelos gerados pela análise para construir o projeto do sistema e o projeto de objetos.

O projeto do sistema divide o sistema em subsistemas, aloca processos aos processadores e procura definir a arquitetura geral, sugerindo várias abordagens, mas não restringindo o projetista a utilizar alguma delas.

O projeto de objetos determina os métodos que serão incluídos em cada classe e as representações de dados. Os relacionamentos devem ser codificados de forma conveniente para seu uso. Finalmente, o projeto é particionado em módulos apropriados à linguagem de programação escolhida.

A fase de implementação converte em código o projeto dos objetos. São fornecidas orientações sobre como os diferentes aspectos (como a implementação dos relacionamentos) podem ser codificados em várias linguagens orientadas a objetos. Analisa também aspectos como o estilo de programação e a forma de projetar classes visando a reutilização.

2.3.3 O Método FUSION

O método Fusion (Coleman et al., 1994) foi desenvolvido para proporcionar uma abordagem sistemática no desenvolvimento de softwares orientado a objetos, de forma a integrar e estender abordagens existentes, tais como OMT (Rumbaugh, 1991) e Booch (Booch, 1994).

O processo de desenvolvimento do método Fusion é dividido em três fases distintas: análise, projeto e implementação.

Partindo de um documento de requisitos, a *fase de análise* produz dois modelos: o modelo de objetos e o modelo de interface, que identificam diferentes aspectos do sistema.

□ A fase de Análise

O modelo de objetos tem por objetivo representar os conceitos existentes no domínio do problema e suas relações. A definição da notação gráfica foi baseada no modelo de objetos do OMT (Rumbaugh, 1991).

Com esse modelo pode-se representar classes, atributos e relacionamentos entre classes, incluindo agregação e especialização/generalização. Os relacionamentos, assim como os objetos, podem ter atributos e dependendo do número n de classes envolvidas em um relacionamento dizemos que o relacionamento é n -ário.

A generalização é um tipo especial de relacionamento onde uma classe, chamada supertipo, representa as propriedades comuns a várias classes, chamadas de subclasses. A especialização é o processo inverso da generalização onde as subclasses herdam os atributos e os relacionamentos da superclasse.

É possível representar também a especialização múltipla onde uma nova subclasse pode ser definida como uma especialização de mais de um superclasse. Com isso, a subclasse herda atributos e relacionamentos de todas as suas superclasses.

A agregação, outro tipo especial de relacionamento, é formada com classes componentes contidas dentro de uma classe agregada. O objeto agregado é representado por um retângulo e é tratado como qualquer outra classe, podendo ter atributos. Um componente de um agregado pode participar de qualquer relacionamento.

Antes de passar para a próxima fase, a fase de projeto, deve-se checar os modelos da análise, verificando se o modelo é consistente e completo. Um modelo é dito consistente quando não existem contradições implícita ou explicitamente entre as partes do modelo. Um modelo é dito completo se todas as abstrações significativas do domínio foram captadas e se o modelo de análise é condizente com as expectativas do usuário descritas no documento de requisitos (Coleman et al., 1994).

□ A fase de Projeto e Implementação

A *fase de projeto* gera um conjunto de modelos que representam o comportamento do sistema na forma de uma coleção de objetos que interagem. A *fase de implementação* mostra como mapear os modelos do projeto em uma linguagem de implementação.

Dessa forma, o processo Fusion é descrito como uma sequência de etapas estabelecendo uma ordem a ser seguida, auxiliando a tomada de decisões do desenvolvedor. Os resultados de cada etapa são explicitamente enviados para uma etapa subsequente e juntamente com cada etapa, o Fusion também estabelece regras para verificar a consistência e integridade dos modelos desenvolvidos.

2.4 Comparação das Abstrações Suportadas pelos Modelos de Dados

A partir dos estudos sobre os modelos de dados, descritos neste capítulo, pôde-se comparar as abstrações que estes modelos suportam. É importante ressaltar que cada modelo de dados define suas próprias abstrações e propriedades particulares, e a maneira que esses modelos nomeiam tais abstrações, nem sempre é uniforme (Biajiz, 1996a).

Isto pôde ser verificado com a abstração de agregação, definida nos modelos ME-RX, SIRIUS, OMT e Fusion. No modelo ME-RX, toda agregação é gerada a partir de um único conjunto de relacionamento, que em geral, envolve dois conjuntos de entidades. Em SIRIUS, esta abstração é tratada de maneira mais abrangente, uma vez que pode ser representada através de tuplas, relacionamentos e atributos de atributos.

O modelo de objetos do OMT e Fusion tratam da mesma forma a agregação, uma vez que o Fusion teve forte influência do método OMT. No modelo de objetos destes métodos, a agregação é caracterizada por classes componentes, as quais estão contidas em uma classe agregada. As classes componentes, não precisam, necessariamente, estarem ligadas por um relacionamento.

A abstração de composição é caracterizada nos modelos SIRIUS e ITASCA pelo relacionamento *is-part-of*, determinando a existência de um objeto composto, e um conjunto de objetos que são parte desse objeto composto. No entanto, cada um dos modelos definem suas próprias particularidades.

Com relação a abstração de generalização/especialização, verificou-se que todos os modelos suportam tal abstração, de maneira homogênea entre eles, sendo caracterizada pelo relacionamento *is-a* entre as classes, embora possuam notações diferentes.

No entanto, o modelo SIRIUS apresenta um conjunto de características que definem uma estrutura básica, para cada ocorrência da abstração de generalização, que é mais completa do que a encontrada nos outros modelos. Dentre essas características pode-se destacar o conjunto de atributos denominado critério, que definem a especialização de um tipo genérico para um, ou mais, tipos específicos.

A Tabela 3 apresentada a seguir, mostra uma comparação entre os modelos de dados e as abstrações que suportam. Com esta tabela foi possível verificar que o modelo de dados SIRIUS é um modelo com alto poder de representação semântica, pois suporta as abstrações de generalização/ especialização, agregação/desagregação, composição/decomposição e classificação/instanciação.

Com a comparação entre os modelos de dados, também foi possível verificar que quanto mais abstrações um modelo de dados suportar, mais detalhes de projeto podem ser representados através da modelagem. E que, a abstração de generalização é importante e tratada em todos os modelos, sendo que quanto mais características semânticas essa abstração suportar, maior será a representação de detalhes definidos através da generalização.

Tabela 3: Comparação entre os modelos de dados

	ABSTRAÇÕES			
	Generalização/ Especialização	Agregação/ Desagregação	Composição/ Decomposição	Classificação/ Instanciação
ME-RX	✓	✓		
IDEA	✓			
SIRIUS	✓	✓	✓	✓
ITASCA	✓		✓	
OMT	✓	✓		
Fusion	✓	✓		

2.5 Considerações Finais

Neste capítulo foram discutidos vários modelos de dados, alguns deles fazem parte do processo de desenvolvimento de métodos e metodologias. É importante destacar que a UML (Unified Modeling Language) (UML, 1998) representa uma coleção das melhores experiências de engenharia utilizadas com sucesso na modelagem de sistemas grandes e complexos. Os vários diagramas que formam a notação da UML, incorporam diferentes elementos, propostos por vários autores, alguns deles descritos neste capítulo. Por este motivo, os diagramas da UML, em particular o diagrama de classes, não foram descritos, juntamente com a metodologia e métodos revisados.

Pôde-se verificar com esta revisão, que as abstrações suportadas pelos modelos de dados proporcionam o desenvolvimento de modelagens mais completas e consistentes. Uma atenção

especial deve ser dada no momento de representar as informações capturadas pelos modelos de dados em um gerenciador relacional.

Para isto, existem na literatura várias abordagens, para auxiliar o desenvolvedor na fase de projeto lógico (Sakai, 1983; Batini & Ceri, 1985; Batini et al., 1992; Spiegler & Widder, 1993). Algumas destas abordagens foram descritas rapidamente e serão detalhadas no próximo capítulo.

Capítulo 3 – Opções de Mapeamento para o Modelo Relacional

3.1 - Considerações Iniciais

O problema do mapeamento de um modelo de dados orientado a objetos para o Modelo Relacional reduz-se ao problema de decidir uma implementação eficiente para as características que não são diretamente suportadas pelo sistema relacional, tais como: identidade do objeto, hierarquias de herança, e construtores de tipos complexos; sempre preservando as propriedades semânticas dos conceitos mapeados.

Um bom mapeamento implica diretamente uma implementação cujo desempenho é previsível, mantendo um compromisso entre tal desempenho e a flexibilidade, coerência de desenvolvimento e preservação das propriedades semânticas. Pode-se dizer que a implementação está completa quando captura todas as abstrações determinadas no projeto e que está correta quando representa fielmente a semântica de todas as abstrações determinadas no projeto (Ceri & Fraternali, 1997).

Vários trabalhos apresentam uma abordagem sistemática para o projeto lógico de banco de dados relacionais considerando o ME-RX como modelo conceitual (Sakai, 1980; Teorey et. al, 1986; Elmasri & Navathe, 1994), ou considerando modelos orientados a objetos (Eick, 1991; Ceri & Fraternali, 1997).

Dentre estes trabalhos, Elmasri e Navathe propõem uma abordagem de mapeamento considerando o ME-RX como modelo conceitual, descrita na seção 3.2 deste capítulo, que trata amplamente todos os aspectos estudados nas outras propostas. Nesta abordagem serão considerados apenas os aspectos estruturais, enquanto que na abordagem de mapeamento da metodologia IDEA (Ceri & Fraternali, 1997) para modelos orientados a objetos, apresentada na seção 3.3, serão considerados também, os aspectos comportamentais.

3.2 - Mapeamento do Modelo Entidade Relacionamento Estendido para o Modelo Relacional

Nesta seção descreve-se o mapeamento do Modelo Entidade Relacionamento Estendido para o Modelo Relacional segundo a abordagem de Elmasri & Navathe (1994). Nesta abordagem, foram incorporados novos passos aos sete passos clássicos de mapeamento, considerando o ME-R. Estes novos passos tem como objetivo contemplar o mapeamento das extensões do ME-R, caracterizando o ME-RX, o qual trata das abstrações de generalização e agregação.

Passo 1: Mapear todos os conjuntos de entidades regulares (que não são fracas, nem especializadas, nem agregadas) do diagrama ME-RX. Para cada conjunto de entidade (CE) E cria-se uma relação R que inclui todos os atributos de E. Caso exista atributo composto, deve-se incluir todos os atributos elementares que compõem o atributo composto. Escolhe-se um dos atributos chave de E como chave primária para a relação R. Se a chave escolhida de E for composta, o conjunto de atributos elementares que a compõe, é juntamente, a chave primária de R.

Passo 2: Para cada conjunto de entidade fraca F cria-se uma relação R formada por todos os atributos do conjunto de entidade fraca, mais os atributos que são chave das relações que mapeiam os CE envolvidos pelo conjunto de relacionamento (CR) total que torna esse CE fraca. A chave da relação que mapeia o conjunto de entidade fraca é a chave desse CE concatenada com a chave das relações que mapeiam os demais CE's envolvidos no CR total que torna esse conjunto de entidade fraca.

É importante notar que, um conjunto de entidade fraca só pode ser mapeado quando os demais CE envolvidos no CR total, que torna esse conjunto de entidade fraco, já estiverem mapeados.

Passo 3: Há várias opções para mapear as subclasses que formam a especialização, ou ainda, que são generalizadas em uma superclasse. Discute-se abaixo todas as possíveis opções de mapeamento juntamente com as condições em que cada opção deve ser utilizada. Para isto, considera-se uma superclasse C, onde os atributos de C são $\{k, a_1, a_2, \dots, a_n\}$ sendo k a chave primária, juntamente com m subclasses $\{S_1, S_2, \dots, S_m\}$, mostradas na Figura 15, segundo uma das notações possíveis para a representação da generalização/especialização.

Figura 15: Representação de uma superclasse C com seus respectivos atributos e subclasses

Opção 3A: Cria-se uma relação $L = \{k, a_1, a_2, \dots, a_n\}$ para C onde k é a chave primária da relação e cria-se uma relação $L_i = \{k\} \cup \{\text{atributos de } S_i\}$ para cada subclasse S_i , $1 \leq i \leq m$. Esta opção é usada para qualquer restrição sobre a abstração de generalização/especialização: mutuamente exclusiva ou sobreposta, total ou parcial. Uma operação *equijoin* sobre a chave primária entre algum L_i e L produz todos os atributos específicos e herdados das entidades em L_i .

Opção 3B: Cria-se uma relação $L_i = \{\text{atributos de } S_i\} \cup \{k, a_1, a_2, \dots, a_n\}$ para cada subclasse S_i , $1 \leq i \leq m$, sendo k a chave primária de cada L_i . Esta opção é viável apenas com as restrições de exclusão mútua e participação total. Se a especialização não é total, uma entidade que não pertence a nenhuma subclasse S_i é perdida. Se a especialização não for exclusão mútua, uma entidade irá pertencer a mais do que uma subclasse e terá os atributos herdados da superclasse C armazenados redundantemente em mais de uma L_i .

Sempre que for necessário pesquisar um atributo qualquer da superclasse, será preciso pesquisar em todas as subclasses; e para isto, aplica-se a operação *outer union* nas L_i relações.

Opção 3C: Cria-se uma relação $L = \{k, a_1, a_2, \dots, a_n\} \cup \{\text{atributos de } S_1\} \cup \{\text{atributos de } S_2\} \cup \{\text{atributos de } S_m\} \cup \{t\}$. Esta opção é apropriada para a especialização onde as subclasses são **disjuntas** e t é um atributo que indica a subclasse a qual cada tupla pertence, portanto o domínio de t deve ser $\{1, 2, \dots, m\}$. Se a especialização é parcial, t pode ter valor nulo nas tuplas que não pertencem a nenhuma subclasse. No caso da especialização ser definida por um atributo, tal atributo exerce o mesmo papel de t e t torna-se desnecessário.

No exemplo da Figura 16, o atributo *qualificação* determina se um funcionário é motorista ou engenheiro, segundo uma das notações possíveis para representar a generalização/especialização. As restrições definidas para esta especialização são: participação parcial e disjunção.

Figura 16: Exemplo de uma especialização

Mapeando esse esquema de acordo com a opção 3C:

Funcionário

<u>CodFunc</u>	Nome	Endereço	T	D	A	Quali	Sal	NumHab	ChefMot	NumCREA
----------------	------	----------	---	---	---	-------	-----	--------	---------	---------

- T: tarefa;
- D: descanso;
- A: ausências;
- Quali: qualificação;
- Sal: salário;
- NumHab: número de habilitação;
- ChefMot: chefe dos Motoristas;
- NumCREA: número do CREA;

Opção 4C: Cria-se uma relação $L = \{k, a_1, a_2, \dots, a_n\} \cup \{\text{atributos de } S_1\} \cup \{\text{atributos de } S_2\} \cup \{\text{atributos de } S_m\} \cup \{t_1, t_2, \dots, t_m\}$. Esta opção é apropriada para a especialização cujas subclasses são *sobrepostas* e cada t_i , $1 \leq i \leq m$ é um atributo *booleano*, indicando se a tupla pertence a uma subclasse S_i . Cada campo t_i tem como domínio $\{\text{sim}, \text{não}\}$, onde o valor “sim” indica que a tupla é um membro da subclasse S_i .

Nas opções 3C e 3D cria-se uma única relação para representar a superclasse C e todas as suas subclasses. Uma entidade que não pertence a nenhuma subclasse terá valores nulos para todos os atributos específicos das subclasses. Se existir poucos atributos específicos, estes mapeamentos são preferíveis às opções 3A e 3B pois não há necessidade de realizar as operações *equijoin* e *outer union*, resultando em uma maior eficiência com relação às consultas.

Quando temos uma especialização em vários níveis, não é necessário adotar a mesma opção de mapeamento para todos os níveis. É possível utilizar uma opção diferente para cada nível (Elmasri & Navathe, 1994).

A Tabela 4 resume as quatro opções de mapeamento apresentadas acima, juntamente com as restrições onde cada opção pode ser utilizada.

Tabela 4 : Opções de mapeamento para a especialização

Opção 1	$L = \{k, a_1, a_2, \dots, a_n\}$ $L_i = \{k\} \cup \{atr S_i\} \quad 1 \leq i \leq m$	- sobreposição/exclusão mútua - participação total/parcial
Opção 2	$L_i = \{atr S_i\} \cup \{k, a_1, a_2, \dots, a_n\}$ $1 \leq i \leq m$	- exclusão mútua - participação total
Opção 3	$L = \{k, a_1, a_2, \dots, a_n\} \cup$ $\{atr S_1\} \cup \dots \cup \{atr S_m\} \cup \{t\}$	- exclusão mútua
Opção 4	$L = \{k, a_1, a_2, \dots, a_n\} \cup$ $\{atr S_1\} \cup \dots \cup \{atr S_m\} \cup \{t_1, \dots, t_m\}$	- sobreposição

Passo 4: No mapeamento de conjuntos de entidades agregação deve-se levar em consideração dois casos:

1. Quando existe apenas uma entidade agregação para cada relacionamento gerador da agregação: mapear os conjuntos de entidades (CE) envolvidos pelo conjunto de relacionamento (CR) gerador conforme os passos 1 e 2. Os atributos do CR gerador deverão ser repassados para o CE agregação. Criar uma relação para o CE agregação incluindo os atributos chaves dos CE's envolvidos como chave candidata.
2. Quando existe mais de uma entidade agregação para cada relacionamento gerador da agregação: deve-se verificar se é possível repassar os atributos do CR gerador para o CE agregação. Define-se uma relação para mapear o CE agregação que será composta por sua chave (seus identificadores unidos às chaves dos CE's envolvidos) e seus

atributos, repassados ou não. Caso o relacionamento tenha atributos que não possam ser repassados ao CE agregação, então este conjunto de relacionamentos deverá ser mapeado normalmente, seguindo os próximos passos.

Passo 5: O mapeamento de conjuntos de relacionamentos (CR) nem sempre resultam em uma nova relação, como é o caso dos CR binário 1:1. Deve-se identificar os CE's S e T que participam do relacionamento. Um dos CE's deve ser escolhido, por exemplo S, e acrescenta-se a ele os atributos do CR e ainda os atributos chave da relação que mapeia o outro CE, que neste caso corresponde a T. Deve-se ressaltar que os atributos chave do conjunto de entidade T são incluídos como atributos não chave no conjunto de entidade S.

Passo 6: Os conjuntos de relacionamento binário 1:N também não são representados como novas relações. Primeiro identifica-se o CE que participa da relação com cardinalidade N que será chamada de S e o outro CE chamada de T. Os atributos do CR são acrescentados no conjunto de entidade S, ou seja, o conjunto de entidade com cardinalidade N. Os atributos chave da relação que mapeia o conjunto de entidade que participa com cardinalidade 1, representado por T, são também acrescentados no conjunto de entidade S como atributos não chave.

Passo 7: Para cada relacionamento binário M:N cria-se uma nova relação. Os atributos da relação são os atributos do CR juntamente com os atributos chave das relações que mapeiam os CE's envolvidos. A chave da relação é a concatenação dos atributos chave das relações que mapeiam os CE's envolvidos.

Passo 8: Para conjuntos de relacionamentos n-ário, $n > 2$ sempre considera-se que possuam cardinalidade **vários: vários: vários**. Para cada CR será criada uma nova relação cujos atributos próprios são os do CR (se existirem) e cuja chave é formada pelos atributos chave das relações que mapeiam os CE's envolvidos. Os CR's de ordem maior que três são tratados da mesma maneira que os ternários. Seu mapeamento cria uma relação para cada CR e esta relação possui: os atributos do CR como atributos próprios e, como chave, os atributos concatenados de cada relação que mapeia os CE's envolvidos.

Passo 9: Existem duas maneiras de mapear atributos multivalorados. A primeira maneira não leva-se em conta conhecimento adicional sobre o atributo que está sendo mapeado. Para cada

atributo multivalorado cria-se uma nova relação que tem como chave os atributos chave da relação que mapeia o CE ou CR ao qual pertencia o atributo, juntamente com o atributo multivalorado tomado como um atributo monovalorado.

Isto pode ser verificado no mapeamento da relação serviço, descrito abaixo, que armazena informações a respeito dos serviços de manutenção realizado em um trem. Nesta relação o atributo localização é multivalorado e por isso, a relação serviço foi quebrada em serviço e servLocalização :

Serviço					
<u>CodServ</u>	TipoServ	localização	flagNoturno	horaIni	horaTerm

Excluindo o atributo multi-valorado:

Serviço					
<u>CodServ</u>	TipoServ	localização	FlagNoturno	horaIni	horaTerm

ServLocalização	
<u>Número</u>	<u>localização</u>

A segunda forma, de mapear atributos multi-valorados, leva-se em conta o conhecimento adicional sobre o atributo que está sendo mapeado. Em alguns casos é possível determinar a quantidade de ocorrências de valores nos atributos. Quando isso acontece e essa quantidade é pequena, pode-se instanciar essa quantidade de atributos como monovalorados na mesma relação que mapeia o CE ou CR ao qual o atributo multivalorado está associado.

No entanto, esta segunda forma de mapeamento impede a automação do procedimento de mapeamento do ME-RX para o Modelo Relacional (Elmasri & Navathe, 1994).

3.3 - Mapeamento das Especificações Chimera para o Modelo Relacional

O processo de mapeamento das especificações conceituais em esquemas, objetos e regras é influenciado pelas características apresentadas pela plataforma de base de dados escolhida.

A metodologia IDEA determina que é necessário mapear o esquema, operações, restrições de integridade, regras dedutivas e regras ativas. Tal metodologia representa essas informações utilizando a linguagem Chimera.

Nas próximas seções descreve-se o mapeamento das especificações Chimera, definidas na fase de projeto, em Oracle. O Oracle é um servidor de base de dados relacional muito difundido

no mercado. É compatível em vários sistemas operacionais e vários hardwares; suporta processamento paralelo, cliente-servidor e distribuição de dados. Proporciona várias soluções para interoperabilidade com outros produtos relacionais, e serviços avançados para replicação dos dados. (ORACLE, 1998).

3.3.1 - Mapeamento do Esquema

O mapeamento do esquema é o processo de transformar o modelo de objetos Chimera em tabelas relacionais. Durante esse processo é preciso preservar, tanto quanto possível, a semântica de orientação a objetos. Para isto, constrói-se uma coleção de tabelas semanticamente relacionadas, definindo-se restrições de integridade referencial entre elas, que serão possivelmente implementadas com *triggers*. Dessa forma, apesar dos objetos estarem dispersos em um grande número de tabelas, é possível reproduzi-los, através de conexões entre as tuplas relevantes, utilizando para isto, as operações relacionais adequadas.

Como tais operações são computacionalmente custosas, esse é um aspecto do processo de mapeamento que pode afetar o desempenho do sistema.

□ Mapeamento da identidade do objeto, referências ao objeto e herança

A identidade do objeto é automaticamente suportada por bases de dados orientadas a objeto, mas não por bases de dados relacionais. Mas este é um conceito muito importante, especialmente para proporcionar integridade referencial e *links* semânticos entre as tabelas.

Assim, é preciso simular o identificador de objetos em Oracle. Para cada classe no esquema é criado um atributo extra do tipo inteiro, cujo propósito é armazenar valores numéricos representando o identificador do objeto. Este valor é único para todos os objetos conceituais na base e é gerado sempre que um novo objeto é criado na classe raiz de uma hierarquia de generalização ou em uma classe que não faz parte de nenhuma hierarquia.

Para ilustrar esta idéia, considera-se os objetos da classe Chimera trem com o esquema:

```
define object class trem
attributes
    codTrem: integer,
    nívelImport: integer derived notnull,
    funcMin: integer notnull,
```

```
funcAdeq: integer notnull,  
perAfastado: set-of(data),  
flagAtraso: boolean notnull,  
possui: trajeto notnull  
end;
```

Este esquema é representado em Oracle da seguinte maneira:

```
CREATE TABLE trem  
(self INTEGER PRIMARY KEY,  
codTrem INTEGER UNIQUE NOT NULL,  
funcMin INTEGER NOT NULL,  
funcAdeq INTEGER NOT NULL,  
flagAtraso INTEGER NOTNULL,  
possui trajeto.self%TYPE;  
CREATE TABLE tremPerAfastado  
(self INTEGER PRIMARY KEY,  
perAfastado char,  
PRIMARY KEY (self, perAfastado),  
FOREIGN KEY (self)  
REFERENCES trem ON DELETE CASCADE);  
end;
```

As restrições de integridade de formato fixo Chimera, como a restrição da chave primária *codTrem*, são transformadas em restrições de unicidade do Oracle. O campo *self* é declarado como chave primária da tabela e todos os tipos de valores são transformados em *strings* ou inteiros do Oracle, dependendo do tipo original.

A classe *trem* (em Chimera) possui um atributo multivalorado chamado *perAfastado*. Por esse motivo, a implementação dessa classe consiste de uma tabela, contendo todos os atributos de *trem*, e outra tabela, *tremPerAfastado*, contendo os valores dos atributos multivalorados.

Essa classe possui ainda, um atributo derivado, multivalorado e não recursivo, chamado *nívelImport*, que será implementado utilizando a definição de visões, que serão rapidamente discutidas na seção 3.3.2.

As referências entre as classes são tratadas incluindo o atributo *self*, dos objetos referenciados, conforme verifica-se no atributo *possui*, o qual referencia o atributo *self* (identificador) da tabela *trajeto*.

O mapeamento de hierarquias de generalização na metodologia IDEA é tratado adotando-se o mapeamento vertical, onde cada classe da hierarquia é mapeada em uma tabela específica, contendo apenas atributos que são localmente definidos naquela classe. Nesta solução, qualquer classe *C1* que herda atributos de uma classe *C2* deve incluir um atributo *self* cujos valores, ao

contrário de serem criados novamente em cada inserção, são considerados a partir dos valores armazenados em C2.

Para facilitar a implementação de uma remoção Chimera na presença de herança, os *triggers* de integridade referencial podem garantir que a remoção de um objeto da superclasse resulte na remoção de todos os objetos correspondentes nas subclasses.

Para o exemplo de generalização apresentada no capítulo 2, onde funcionário foi especializado em motorista e engenheiro, pode-se escrever o *trigger* `propagueDelMotorista`, o qual remove as tuplas da relação motorista sempre que houver a remoção de uma tupla correspondente na relação funcionário.

```
CREATE TRIGGER propagueDelMotorista
AFTER DELETE ON funcionário
BEGIN
DELETE motorista WHERE self
      NOT IN (SELECT self FROM funcionário);
END;
```

Uma outra alternativa é colocar a restrição declarativa `FOREIGN KEY ON DELETE CASCADE` na tabela motorista, ligando o atributo *self*, da tabela motorista, ao atributo correspondente na tabela funcionário.

□ Mapeamento de Operações

As operações Chimera são sequências de consultas e/ou atualizações, e são executadas sobre conjuntos de objetos, podendo ter variáveis de entrada e de saída.

A melhor opção no mapeamento das operações são as *stored procedures* do Oracle, pois proporcionam uma representação bem próxima das operações Chimera, uma vez que permitem que uma parte do código PL/SQL seja colocado dentro de uma unidade que pode ter parâmetros de entrada e saída. Toda *procedure* que implementa uma operação Chimera tem pelo menos um parâmetro de entrada, o qual corresponde ao identificador do objeto sobre o qual a operação é executada.

3.3.2 - Mapeamento de Regras Dedutivas

Como Oracle é baseado na linguagem SQL, o suporte a restrições de integridade é feito de maneira muito restrita, permitindo a derivação dos dados por meio de visões apenas quando a derivação é não recursiva.

As regras dedutivas recursivas podem ser implementadas primeiro transformando-as em regras ativas, e então, mapeando essas regras ativas resultantes por meio de técnicas que serão discutidas na próxima seção.

As visões não recursivas, as classes derivadas e os atributos derivados, escritos em Chimera, podem ser implementados por meio de visões relacionais. As visões, em Oracle, podem ser definidas em termos de tabelas existentes ou por meio de outras visões.

3.3.3 - Mapeamento de Regras Ativas

Embora suportado por muitos sistemas, como Oracle, Illustra, Db2 e ODE, o mapeamento de regras ativas é muito difícil e problemático (Ceri & Fraternali, 1997). Um dos problemas é a natureza operacional intrínseca da semântica das regras ativas suportadas no modelo conceitual Chimera, que é um tanto diferente dos sistemas propostos.

No caso do mapeamento para Oracle, um *trigger* Chimera é transformado em um *trigger* Oracle correspondente. Em geral, a transformação de *triggers* Chimera em *triggers* Oracle não se mantém, uma vez que existem algumas restrições impostas no sistema de *triggers* Oracle.

Em função dessas dificuldades, o desenvolvedor deve estar atento no momento do mapeamento de *triggers*, a fim de encontrar a solução mais apropriada. Uma das soluções consiste em proporcionar uma implementação em Oracle, da semântica de processamento de *triggers*, mapeando cada *trigger* em uma *stored procedure*. Desse modo, cada *trigger* em Chimera é mapeado, para exatamente, um procedimento em Oracle. Para isto, é preciso adicionar estruturas de dados auxiliares e *triggers*, com o intuito de que essa *stored procedure* seja corretamente executada.

Esta abordagem é chamada de *meta-triggering*, uma vez que, muita semântica está sendo representada pelos *meta-triggers*, com a propagação de eventos, resolução de conflitos e chamada a procedimentos. A vantagem dessa solução, é que, as *stored procedures* “lógicas”

(escritas em Oracle) preservam a semântica de *triggers* “conceituais” (escritos em Chimera). A desvantagem é que o disparo é indireto, afetando o desempenho do sistema.

A segunda solução consiste em proporcionar o mapeamento de muitos *triggers* Chimera para *triggers* Oracle. Os *triggers* Chimera são desmembrados em *triggers* com eventos simples e, a partir disso, os *triggers* de mesmo tipo, sobre o mesmo evento, que são executados em uma sequência fixa, são reagrupados em um *trigger* Oracle. Esta abordagem é chamada *macro-triggering*.

A Tabela 5 mostra uma comparação das características Chimera e suas respectivas representações em Oracle.

Tabela 5: Características Chimera e as respectivas implementações em Oracle

Características Chimera	Implementação Oracle
identificador do objeto	atributo do tipo inteiro - self
hierarquias de generalização	Mapeamento vertical
operações	<i>stored procedure</i>
restrições de integridade de formato fixo	cláusulas: - <i>primary key</i> - <i>unique</i> - <i>not null</i>
- atributos derivados - classes derivadas - visões não recursivas	Visões
regras dedutivas não recursivas	Visões
regras ativas	- <i>meta-trigger</i> - <i>macro-trigger</i>

3.4 – Considerações sobre o Mapeamento

A maneira como alguns conceitos são tratados pela abordagem relacional e a de objetos difere, e para que as propriedades que se busca garantir num projeto orientado a objeto possam ser preservadas numa implementação apoiada no Modelo Relacional, é necessário que se adotem diretrizes de projeto e implementação adequadas. Almeja-se que a integração possa explorar e preservar plenamente as vantagens de ambos os “mundos” (abordagem relacional e OO).

Na metodologia IDEA, todas as decisões relevantes para o mapeamento são tratadas durante a fase de projeto, conforme descrito no capítulo 2, onde são definidas as especificações Chimera. Com isso, o mapeamento de Chimera para um gerenciador relacional depende apenas das características apresentadas pelo sistema gerenciador de base de dados escolhido.

É importante ressaltar que, na fase de projeto, a abordagem utilizada para o mapeamento do relacionamento é a mesma que foi descrita em detalhes neste capítulo (seção 3.2), com a ressalva de que, como o mapeamento é feito considerando-se um modelo OO, utiliza-se atributos objeto-valorados que servem como referência entre os objetos.

Outra consideração importante é com o mapeamento da abstração de generalização/especialização, pois IDEA define algumas diretrizes para o mapeamento desta abstração. Nestas diretrizes, verificadas na fase de projeto das hierarquias de generalização, as restrições de participação e sobreposição são reavaliadas com o objetivo de priorizar as hierarquias com restrição de participação total e disjunção, e eliminar as demais hierarquias (participação parcial e sobreponível).

Estas prioridades são estabelecidas no momento do mapeamento para que as regras obtidas não sejam complexas, inviabilizando seu mapeamento para plataformas comerciais (Ceri & Fraternali, 1997). Uma justificativa fornecida pelos autores da metodologia é que, apesar destas restrições impostas, IDEA atende a maioria dos casos com sucesso.

No mapeamento do modelo conceitual ME-RX para o Modelo Relacional, proposto por Elmasri & Navathe (1994), foram definidos vários passos, considerando-se entre outros, o mapeamento de relacionamentos e das abstrações de generalização e agregação. Para a generalização foram definidas quatro opções de mapeamento, considerando para isto, as restrições de participação (total/parcial) e de sobreposição (disjunta/sobreponível). Note-se que estas quatro opções proporcionam ao desenvolvedor uma maior flexibilidade na escolha do mapeamento adequado para hierarquias, do que a abordagem proposta pela metodologia IDEA.

Pôde-se concluir que a metodologia IDEA proporciona suporte adequado ao ciclo de vida orientado a objetos além de uma orientação específica para aplicações centradas em banco dados. No entanto, tal metodologia restringe algumas opções dos usuários. Este fato pôde ser verificado na fase de análise, onde o modelo de objetos suporta apenas a abstração de generalização, e na fase do projeto de hierarquias, onde é dada prioridade às hierarquias com restrições de disjunção e participação total. Estas restrições, no entanto, não atrapalham o desenvolvimento da maioria das aplicações.

Considerando os mapeamentos estudados, observa-se que as abordagens de mapeamento para a abstração de generalização são tratadas a partir das restrições de disjunção e participação total (Ceri & Fraternali, 1997), ou mais genericamente, considerando as restrições de disjunção/sobreposição e a participação total/parcial (Elmasri & Navathe, 1994). Neste contexto, no próximo capítulo será apresentada uma proposta de mapeamento que estende as propostas apresentadas neste capítulo, com o objetivo de proporcionar ao desenvolvedor, um conjunto completo de opções de mapeamento.

3.5 – Considerações Finais

Normalmente, o mapeamento de uma modelagem orientada a objeto para uma modelagem relacional deve seguir uma seqüência de passos, onde cada construtor semântico é “traduzido” para um conjunto de relações e um conjunto de regras de manipulação, em geral implementadas por *triggers* ou *stored procedures* no gerenciador de dados. Cada um dos construtores semânticos possui diversas opções de mapeamento, e cada uma das opções, em geral, depende de duas classes de fatores: aqueles que preservam as propriedades do modelo orientado a objeto; e aqueles que dependem de decisões de implementação, cujos fatores de decisão normalmente não são capturados pelos modelos de objetos, tais como, volume de dados, tipo de consulta a ser efetuado, e frequência de acesso aos dados.

Considerando o conjunto de construtores semânticos de um modelo OO semanticamente rico, e as diversas possibilidades de representação de cada construtor semântico, o capítulo 4 é centrado no tratamento e mapeamento da abstração de generalização/especialização. Neste mapeamento que será proposto, considerada-se entre outros, as restrições de disjunção/sobreposição e participação total/parcial, o volume dos dados, as consultas a serem efetuadas, e a frequência de acesso aos dados, das abstrações de generalização.

Capítulo 4 - Método para o Mapeamento da Abstração de Generalização

4.1 Considerações Iniciais

A escolha da abstração de generalização, para este trabalho, deve-se ao fato de que entre os modelos de dados a generalização é importante e está presente em todos eles. Além disso, tal abstração possibilita uma estrutura do conjunto de tipos de objetos de um esquema, e por isso, em geral, deve ser sempre considerada nos passos iniciais do processo de mapeamento do esquema.

A maioria das ocorrências dos demais construtores semânticos, que constituem um esquema de dados, somente podem ser mapeados após o mapeamento das OAGs. Dessa forma, um método para o mapeamento de outras abstrações poderá ser estudado depois de definido o mapeamento da abstração de generalização.

Neste capítulo apresenta-se um método para o mapeamento das OAGs, de um esquema apoiado num modelo conceitual semanticamente rico, para um esquema baseado no Modelo Relacional, através de um conjunto de dez opções de mapeamento.

Na seção 4.2 é apresentado um modelo de dados orientado a objetos, o qual suporta um conjunto de características básicas que definem uma ocorrência da abstração de generalização. Na seção 4.3 é apresentada a descrição das dez opções de mapeamento propostas, juntamente com as considerações relevantes que devem ser verificadas no momento da escolha por uma delas, e exemplos de utilização. A avaliação destas opções é apresentada na seção 4.4, e na seção 4.5 define-se uma árvore de decisão, a qual permite a automatização do processo de mapeamento. Na seção 4.6 compara-se o método proposto com os principais métodos

encontrados na literatura, e na seção 4.7, discute-se a abrangência das opções de mapeamento propostas. Finalmente, na seção 4.8 apresenta-se as considerações finais.

4.2 O Modelo Conceitual Escolhido

O conjunto de construtores semânticos de um modelo de dados proporcionam uma capacidade de representação maior, de situações do mundo real, quando comparados com os construtores do Modelo Relacional.

Com o objetivo de cobrir amplamente as situações representadas pelos modelos de dados, adotou-se neste trabalho um modelo conceitual rico, de maneira a tornar o processo de mapeamento resultante abrangente, para que possa ser utilizado em muitos modelos de dados conceituais.

Para isso, define-se um modelo de dados orientado a objeto especificamente construído para suportar a abstração de generalização, e ser adequado para este trabalho. Este modelo, que será referenciado como Modelo Canônico para a abstração de generalização, não tem a preocupação de ser completo, uma vez que os demais conceitos que compõem um modelo de dados não serão discutidos. Considera-se implicitamente, que os conceitos e construtores semânticos, comuns aos modelos orientados a objetos, são suportados. A definição que será feita nesta seção leva em conta as características diretamente vinculadas a abstração de generalização.

Basicamente, o conjunto de características que definem uma OAG, que deseja-se tratar no momento do mapeamento, foi extraído do Modelo SIRIUS (Araujo, 1998a). Esse conjunto de características define uma estrutura básica para cada OAG, que é constituída dos seguintes elementos:

- a) um tipo (único) genérico de objeto - G ;
- b) um conjunto de tipos de objetos especializados $E = \{E_1, E_2, \dots, E_{ne}\}$;
- c) um conjunto de atributos do tipo genérico $XG = \{G.Atributo_i\} = K \cup AG$, onde $K = \{G.AtributoChave_j\}$, e $AG = \{G.AtributoNãoChave_k\}$;
- d) um conjunto de atributos de cada tipo especializado $AE_e = \{E_e.Atributo_i\}$, $1 \leq e \leq ne$;
- e) um conjunto de restrições para a generalização $R = \{rS, rP\}$;
- f) um conjunto de propriedades para a generalização $P = \{p_1 \dots p_p\}$;
- g) um predicado para cada especialização baseado no conjunto de atributos do tipo genérico $P_e = p_e(G.Atributo_i, \dots)$, $1 \leq e \leq ne$;

- h) um conjunto de regras de inicialização de valores para cada tipo de objeto especializado $I_e = \{E_e.Atributo_i = v_i(G.Atributo_j, \dots)\}, 1 \leq e \leq n_e$; e
- i) um conjunto de atributos denominado critério da OAG, composto por todos os atributos do tipo genérico dessa ocorrência que participam do predicado de algum tipo específico, isto é $C = \{G.Atributo_i\}, \mid \exists p_e(G.Atributo_j, \dots) \ 1 \leq e \leq n_e$ onde $G.Atributo_i$ é antecedente de ao menos um predicado $p_e()$.

Nas próximas subseções serão descritas com mais detalhes, algumas das características citadas acima, que definem uma estrutura básica para as OAGs.

4.2.1 Tipo Genérico e Tipos Especializados

Note-se que como em SIRIUS, adota-se aqui os termos tipo e classe de objetos como tendo significados distintos: o termo tipo de objeto determina o conjunto de regras e propriedades que define como são as instâncias de um objeto que tenha esse tipo; já o termo classe determina o conjunto de objetos de um mesmo tipo. Assim, ao contrário dos termos: classe; superclasse e subclasse, são usados os termos:

- tipo - qualquer tipo de objeto, sem levar em conta sua posição em alguma OAG;
- supertipo - o tipo de objeto G que é o tipo genérico em uma OAG; e
- subtipo - um dos tipos de objeto especializados E_i em uma OAG;

Uma OAG tem exatamente um tipo genérico, e $n_e \geq 1$ tipos de objetos especializados.

4.2.2 Os Atributos dos Tipos de Objetos envolvidos

Os tipos genérico e especializados são caracterizados através do conjunto de atributos que os compõem. Neste trabalho considera-se que, se o modelo de dados conceitual suportar a representação de relacionamentos além da representação de atributos, os relacionamentos são considerados casos particulares de atributos, a exemplo da maneira como isso é tratado em SIRIUS (Biajiz, 1996a). Note-se que a intersecção do conjunto de atributos de qualquer tipo especializado com o conjunto de atributos do tipo genérico é sempre vazia, isto é, $G \cap E_e = \emptyset, \forall e, 1 \leq e \leq n_e$.

Todo tipo genérico tem um conjunto K , não vazio, de atributos que identificam univocamente cada objeto desse tipo, denominado chave do objeto. Caso exista mais de um atributo no conjunto de chaves, considera-se que cada atributo isoladamente identifica univocamente cada objeto. Isso significa que se uma chave for a concatenação de mais do que um atributo, a tupla resultante deve fazer parte do conjunto K , mas não cada atributo componente. Os subtipos podem ou não ter chaves especializadas, mas é importante notar que, em qualquer caso, todos os subtipos herdam a chave do supertipo. Além disso, uma chave não pode ser alterada e a identificação do objeto mantida.

Devido a essas restrições, adota-se o conceito de identificador de objetos – Oid, dos modelos de dados OO. No mapeamento para o Modelo Relacional, o conjunto de chaves de um tipo é analisado. Caso exista ao menos um atributo que atenda essas restrições, e além disso, possa identificar todos os objetos da classe desse tipo, então esse atributo pode ser usado diretamente no Modelo Relacional. Caso não exista, ou a restrição de não se permitir a alteração no valor da chave não seja segura, então deve-se criar um atributo especial no esquema mapeado, associado à relação que mapeia o supertipo, controlado pelo sistema, que tenha a função de atuar como um Oid.

4.2.3 As restrições para a generalização

Neste trabalho, considera-se que toda OAG deve ter definida as seguintes restrições:

- **Sobreposição:** indica se um objeto do tipo genérico pode ser de no máximo um tipo específico ($rS=falso$, disjunção), ou pode ser de mais de um tipo específico ($rS=verdadeiro$, sobreposição);
- **Participação:** indica se todo objeto do tipo genérico deve necessariamente ser também de um tipo específico ($rP=falso$, participação total) ou não ($rP=verdadeiro$, participação parcial).

Além dessas duas restrições, é importante ressaltar que todo objeto que é de um determinado tipo específico tem que ser necessariamente do tipo genérico.

Considera-se ainda, que toda OAG tem exatamente um tipo genérico. Isso não significa que um tipo qualquer não possa ser a especialização de mais do que um tipo genérico, pois um mesmo tipo de objeto pode ser subtipo em mais de uma OAG. No entanto, para garantir a

consistência das operações de mapeamento propostas, devem ser asseguradas as duas restrições globais seguintes:

- uma coleção de OAGs não podem formar ciclos, ou seja, um tipo de objeto E_1 que é direta ou indiretamente subtipo de um objeto E_2 não pode ser supertipo, direta ou indiretamente, do mesmo objeto E_2 ;
- todo objeto deve possuir um único tipo natural. Entende-se que tipo natural é o tipo de objeto que não participa como subtipo de nenhuma OAG. Caso um objeto participe de mais do que uma OAG, então o tipo natural do supertipo de cada uma dessas ocorrências deve ser o mesmo.

4.2.4 Predicados para as especializações

Um predicado é uma função $p_e(G.Atributo_i, \dots)$ aplicada sobre um subconjunto dos atributos do tipo genérico, que resulta num valor lógico verdadeiro ou falso. Essa função é avaliada para cada objeto de tipo genérico, e verifica os valores de um ou mais atributos desse objeto. Se o resultado for verdadeiro, significa que esse objeto é também do subtipo correspondente ao predicado que fora avaliado.

Existem na literatura alguns modelos que consideram a possibilidade de definição de uma OAG de duas maneiras: OAG definida pelo usuário; ou definida por predicado. Neste trabalho considera-se apenas aquelas definidas por predicados, por dois motivos:

1. ocorrências definidas pelo usuário não podem ser especializadas pelo sistema automaticamente. Embora isso não afete o processo de automação do processo de mapeamento em si, se houver uma maneira de contornar essa restrição sem afetar negativamente outras características do sistema, então essa possibilidade de definição de uma OAG é irrelevante;
2. não existe nenhuma diferença, tanto do ponto de vista do usuário, quanto do ponto de vista da implementação do aplicativo ou da implementação da base de dados, entre o usuário indicar diretamente que um determinado objeto é de um determinado subtipo, e o usuário indicar o valor de um atributo que o sistema utilizará como critério para decidir o subtipo desse objeto.

4.2.5 Regras de inicialização de objetos de tipo específico

Um objetivo deste trabalho é definir a abstração de generalização o mais completamente possível, de maneira a permitir que uma aplicação “entenda” o máximo possível do significado embutido nas estruturas embasadas nela, para poder automatizar ao máximo as operações que as envolve.

Os predicados das especializações permitem definir a seguinte operação: dado um objeto de tipo genérico, ele é também de um tipo específico ? e se for, de qual ou quais tipos específicos? Ou seja, conhecendo-se os valores de seus atributos, um predicado permite identificar seu(s) tipo(s) específico(s). Para que um sistema seja completo, é necessário também que ele seja capaz de resolver a questão inversa, ou seja, dado um objeto o qual sabe-se a priori que é de um tipo específico, quais devem ser os valores de seus atributos para que ele possa consistentemente ser desse tipo específico?

Em particular, quando um objeto é criado como sendo de um tipo específico, é necessário definir quais são as condições fundamentais de inicialização dos seus atributos para que ele efetivamente seja posteriormente reconhecido como sendo desse tipo pelo predicado dessa especialização. Uma regra de inicialização é portanto um conjunto de operações de inicialização de atributos, baseada em valores constantes, ou em valores dos atributos do tipo genérico.

4.2.6 Critérios da Generalização

Um critério C é o conjunto de todos os atributos do tipo genérico que são antecedentes de ao menos um predicado de uma determinada OAG. O domínio de um atributo contido no critério define dados importantes para caracterizar a OAG. Por exemplo, um critério composto por um único atributo monovalorado, discreto e finito, na qual todos os predicados correspondem a uma única comparação de igualdade, caracteriza uma OAG que será sempre não sobreponível.

Em geral, o critério pode ser representado por:

- um conjunto de atributos monovalorados;
- um conjunto de atributos, onde ao menos um deles é multivalorado;
- uma expressão sobre um ou mais atributos.

4.2.7 Propriedades para a Generalização

As restrições de uma OAG são condições lógicas que devem ser sempre seguidas para garantir a consistência dos dados. No entanto, na implementação de uma estrutura de dados que representa uma abstração de generalização, ocorrem fatores, próprios da aplicação e não da estrutura conceitual da abstração, que podem ser importantes na escolha de uma maneira particular de implementar uma OAG. Neste trabalho, consideramos como propriedades os seis seguintes fatores:

p_1 – Frequência de acesso ao conjunto de atributos que definem o critério – nesta propriedade, verifica-se a frequência com que, o conjunto de atributos que definem o critério, são acessados. Em toda referência feita ao critério, é importante considerar as três maneiras em que é possível representá-lo. Por isso, esta propriedade subdivide-se em três outras, que são:

- **p_{11}** – acesso ao critério é realizado sobre um conjunto de atributos monovalorados;
- **p_{12}** – acesso ao critério é realizado sobre um conjunto de atributos onde, ao menos um deles é multivalorado;
- **p_{13}** – acesso ao critério é realizado por uma expressão sobre um ou mais atributos;

p_2 – Frequência de acesso a informações conjuntas de vários subtipos – esta propriedade considera que sempre, ou com muita frequência, verifica-se o acesso às informações dos atributos de vários subtipos ao mesmo tempo. Caso essa propriedade seja verdadeira, é ineficiente manter uma tabela para cada subtipo, pois seria necessário o uso freqüente de operações de junção de tabelas. A melhor opção neste caso, seria manter as informações de todos os subtipos em uma única tabela, tornando dessa forma, o acesso às informações mais eficiente.

p_3 - Frequência de acesso a informações conjuntas do subtipo e do supertipo - esta propriedade leva em conta a previsão de que sempre, ou com muita frequência, o acesso a informações nos atributos correspondentes ao supertipo estarão conjugadas a informações do(s) respectivo(s) subtipo(s), ou não. Se isso é verdade, manter ambas as partes da informação separadas seria ineficiente, pois irá requerer o uso freqüente de operações de junção de tabelas. Caso a informação seja preferencialmente acessada pelo supertipo, ou pelo subtipo, então mantê-las em tabelas separadas, menores, pode agilizar o acesso.

p_4 - Supertipo/Subtipo envolvido com outros construtores semânticos (relacionamento, agregação, ...) – neste caso, pode ser mais apropriado armazenar as

informações do supertipo e dos subtipos em tabelas separadas. Diminuindo dessa maneira, o acesso à parte da informação compartilhada com ocorrências de outras abstrações. A existência de mais de um elemento semântico compartilhando a mesma informação também indica um ponto da modelagem sensível a problemas de normalização no esquema mapeado. Portanto, a separação das informações contribui para diminuir problemas de acesso e de inconsistência de atualizações potenciais.

p_5 – Existem muitos atributos no supertipo – o mais apropriado seria definir uma tabela para armazenar apenas as informações do supertipo. Considerando a opção em que define-se uma única tabela com informações do supertipo e do subtipo, o resultado seria uma tabela com uma quantidade consideravelmente grande de informações para serem armazenadas e mantidas. Essa situação seria ainda pior, caso não existisse uma tabela para mapear as informações do supertipo, e estas serem replicadas nas tabelas que mapeiam os subtipos.

p_6 – Existem muitos atributos em cada subtipo – o mais apropriado seria definir uma tabela para mapear cada subtipo. Análogo a p_5 , não é apropriado definir uma única tabela com informações do supertipo e do subtipo. No caso de não existir uma tabela que mapeia as informações do supertipo e tais informações forem repassadas para cada subtipo, também corre-se o risco de se trabalhar com tabelas maiores do que o necessário.

A Tabela 6 relaciona cada uma das propriedades, definidas acima, com as possíveis situações identificadas a partir de tais propriedades, as quais servirão de base para a avaliação mostrada na Tabela 8 da seção 4.4.

p_1	E
p_{11}	F
p_{12}	G
p_{13}	H
p_2	I, J
p_3	K, L
p_4	M, N
p_5	O
p_6	P

Tabela 6: Relação das propriedades de uma OAG

4.2.8 A Estrutura geral

Para a descrição das opções de mapeamento, apresentada na seção 4.3, utiliza-se a notação gráfica ilustrada na Figura 17. O supertipo da ocorrência é sempre representado por um retângulo acima dos demais componentes, e indica o tipo genérico G , e seus atributos K e AG . O trapézio no centro da rede representa a OAG em si, onde são indicadas as restrições R , as propriedades P e o critério da ocorrência C . Os n_e subtipos são representados abaixo do trapézio, juntamente com seus atributos específicos AE_e , as regras de inicialização I_e e o predicado P_e desse subtipo, onde $1 \leq e \leq n_e$. A Figura 18 exemplifica esses conceitos.

Figura 17: Estrutura geral de uma OAG

Na Figura 18_a representa-se uma OAG onde o supertipo Funcionário pode ser especializado nos subtipos Motorista e Engenheiro, de acordo com as restrições de disjunção ($rS=falso$) e participação parcial ($rP=verdadeiro$). O atributo profissão, representado no trapézio central juntamente com as restrições, é o atributo que estabelece o critério desta OAG.

Neste caso, o critério é determinado pelo valor de um único atributo cujo domínio é $dom(profissão)=\{Motorista, Engenheiro\}$. Logo, se a avaliação do predicado 'profissão=motorista' for verdadeira, isto indica que um objeto Funcionário foi especializado no subtipo Motorista. Por outro lado, se a avaliação do predicado 'profissão=engenheiro' for verdadeira, então o Funcionário foi especializado para o subtipo Engenheiro. Caso as duas avaliações resultem em falso, pode-se concluir que o Funcionário

que está sendo avaliado não foi especializado em nenhum subtipo, uma vez que a restrição de participação é parcial.

Figura 18: Exemplos de OAGs

É importante observar ainda, que na parte inferior direita dos retângulos que representam os subtipos, define-se os valores de inicialização dos objetos. No exemplo da Figura 18_a, os valores de inicialização são MO e EN, para os subtipos Motorista e Engenheiro, respectivamente. Suponha que deseja-se verificar por que um determinado objeto pertence ao subtipo Motorista. Com o valor de inicialização MO, os objetos do subtipo Motorista podem efetivamente serem reconhecidos como sendo deste subtipo, pela avaliação do predicado definido nesta OAG.

Na Figura 18_b representa-se uma OAG onde o supertipo Pessoa pode ser especializado nos subtipos Contribuinte e Aposentado de acordo com as restrições de disjunção ($rS=falso$) e participação total ($rP=falso$). Neste exemplo, o critério que define a especialização depende de mais de um atributo: idade e contribuição, e a operação aplicada ao conjunto de atributos critério (o teste do critério) é diferente da igualdade.

Assim, se a avaliação do predicado 'idade < 60 e 0 < contribuição < 35' for verdadeira, isto indica que o objeto Pessoa foi especializado na subclasse Contribuinte. E se a avaliação do predicado 'idade ≥ 60 e contribuição ≥ 35' for verdadeira, indica que o objeto Pessoa foi especializado na subclasse Aposentado.

Os atributos do supertipo e dos subtipos foram omitidos nestes exemplos, mas serão sempre listados associados aos retângulos que os representam.

4.3 Definição de Dez Opções para o Mapeamento da Generalização/Especialização

O processo de mapeamento deve ser dividido em duas etapas. Na primeira etapa deve-se efetuar o mapeamento estrutural dos aspectos estáticos do esquema original para um conjunto de tabelas e restrições de integridade do Modelo Relacional canônico. Em uma segunda etapa, deve-se efetuar o mapeamento dos aspectos dinâmicos do esquema original, em regras de acesso à base de dados, e procedimentos na implementação do código do aplicativo. Este trabalho está centrado no tratamento dos aspectos estáticos do processo de mapeamento.

Para isto, nas subseções seguintes, tal processo é descrito através de um conjunto de dez “opções de mapeamento”, onde cada uma delas, é voltada para atender as necessidades de uma situação específica, em uma OAG. Nessa descrição, cada opção é caracterizada por um esquema de relações, resultante do processo de mapeamento de uma OAG.

Em cada procedimento, apresenta-se também um exemplo, seu respectivo mapeamento para Modelo Relacional e uma descrição de situações adequadas para o uso de cada opção em particular.

A Tabela 7 classifica as dez opções de mapeamento em três grupos, conforme os resultados de mapeamento:

- $G + \{E_i\}$: o mapeamento resulta em uma tabela para tratar o tipo genérico e uma para cada tipo específico envolvido na OAG;
- $\{E_i\}$: o mapeamento resulta em uma tabela para cada tipo específico envolvido em uma OAG, e as informações do tipo genérico serão replicadas em cada tipo específico;
- G : o mapeamento resulta em uma única tabela que contém informações do tipo genérico e de todos os tipos específicos.

Tabela 7: Dez Opções para o Mapeamento

Opções de Mapeamento	Descrição	Grupo
1	$G = \{\underline{K} \cup AG\}$ $E_e = \{\underline{K} \cup AE_e\} \quad 1 \leq e \leq n_e$	$G + \{E_i\}$
2	$G = \{\underline{K} \cup C \cup AG\}$ $E_e = \{\underline{K} \cup AE_e\} \quad 1 \leq e \leq n_e$	
3	$G = \{\underline{K} \cup AG\}$ $E_e = \{\underline{K} \cup AE_e\} \quad 1 \leq e \leq n_e$ $D = \{\underline{K} \cup C\}$	
4	$E_e = \{\underline{K} \cup AG \cup AE_e\} \quad 1 \leq e \leq n_e$	$\{E_i\}$
5	$E_e = \{\underline{K} \cup C \cup AG \cup AE_e\} \quad 1 \leq e \leq n_e$	
6	$E_e = \{\underline{K} \cup AG \cup AE_e\} \quad 1 \leq e \leq n_e$ $D = \{\underline{K} \cup C\}$	
7	$E_e = \{\underline{K} \cup AG \cup AE_e\} \quad 1 \leq e \leq n_e$ $D = \{\underline{K} \cup C\}$	
8	$G = \{\underline{K} \cup AG \cup \{AE_1 \cup AE_2 \cup \dots \cup AE_{n_e}\}\}$	G
9	$G = \{\underline{K} \cup C \cup AG \cup \{AE_1 \cup AE_2 \cup \dots \cup AE_{n_e}\}\}$	
10	$G = \{\underline{K} \cup AG \cup \{AE_1 \cup AE_2 \cup \dots \cup AE_{n_e}\} \cup \{s_1, s_2, \dots, s_{n_e}\}\}$	

4.3.1 Opção de Mapeamento 1 (OM1)

O esquema de relações gerado pela opção de mapeamento 1 é descrito da seguinte maneira:

$$G = \{\underline{K} \cup AG\}$$

$$E_e = \{\underline{K} \cup AE_e\} \quad 1 \leq e \leq n_e$$

Nesta opção cria-se uma relação G para o tipo genérico e n_e relações (E_e) para cada tipo específico, onde n_e é o número de subtipos. A relação G possui o conjunto de atributos identificadores (K), como chave primária, e o conjunto de atributos do tipo genérico (AG). Cada relação E_e possui o conjunto de atributos identificadores (K), herdados do tipo genérico, como chave primária, e o conjunto de atributos específicos do subtipo (AE_e).

Para ilustrar esta OM, considere-se o exemplo da Figura 19_a onde Pessoa pode ser especializada como Funcionário, Professor e Aluno, em uma OAG sobreponível ($rS=\text{verdadeiro}$) e de participação total ($rP=\text{falso}$). O atributo critério desta OAG é representado pelo atributo vínculo cujo domínio é $\text{dom}(\text{vínculo}) = \{\text{Funcionário}, \text{Professor}, \text{Aluno}\}$.

Dentre os atributos do supertipo, aquele que aparece sublinhado (IdPessoa) é o atributo identificador que será herdado pelos objetos dos subtipos.

O mapeamento deste exemplo, usando a OM1, está ilustrado na Figura 19_b, e tem como resultado uma tabela para representar o supertipo e uma tabela para cada subtipo. Note-se que não existe uma estrutura sintática específica para mapear o conjunto de atributos critério: a identificação de uma pessoa como funcionário, aluno ou professor dá-se pelo fato de seu identificador ser encontrado em uma relação específica.

(a)

(b)

G = Pessoa

<u>IdPessoa</u>	Nome	Endereço	Nascimento
-----------------	------	----------	------------

E₁ = Funcionário

<u>IdPessoa</u>	DtContratação	Função
-----------------	---------------	--------

E₂ = Professor

<u>IdPessoa</u>	AreaPesquisa	NroSala	Status
-----------------	--------------	---------	--------

E₃ = Aluno

<u>IdPessoa</u>	NroAluno	Curso
-----------------	----------	-------

Figura 19: Mapeamento de uma OAG segundo OM1

Esta opção é apropriada para as situações em que o subtipo, que deve ser acessado para responder a uma determinada consulta, é conhecido a priori. Ou seja, quando for preciso acessar informações específicas de uma determinada pessoa, e já sabe-se que esta pessoa é um funcionário, professor ou aluno. Dessa forma, a consulta já é direcionada à tabela específica, que contém as informações que deseja-se acessar. Essa situação típica é exemplificada com as informações de uma escola, pois as operações são sempre dirigidas especificamente para cada tipo de pessoa (gerar lista de notas, matricular alunos, atribuir carga didática para professor,

efetuar pagamento, etc.), ou genericamente para qualquer pessoa, sem importar seu subtipo (enviar mala-direta).

4.3.2 Opção de Mapeamento 2 (OM2)

O esquema de relações gerado pela opção de mapeamento 2 é descrito da seguinte maneira:

$$G = \{\underline{K} \cup C \cup AG\}$$

$$E_e = \{\underline{K} \cup AE_e\} \quad 1 \leq e \leq ne$$

Nesta opção cria-se uma relação G para o tipo genérico e ne relações E_e para cada tipo específico, onde ne é o número de subtipos. A relação G possui o conjunto de atributos identificadores (K), como chave primária, o conjunto de atributos do tipo genérico (AG), e o atributo C para mapear o valor do critério. Cada relação E_e possui o conjunto de atributos identificadores (K) herdado do tipo genérico como chave primária, mais o conjunto de atributos específicos do subtipo (AE).

A Figura 20_a mostra a OAG de uma Empresa que pode ser especializada de acordo com o atributo critério NroFunc (número de funcionários) a partir da avaliação dos predicados:

- (1) **PredicadoPequena:** $1 \leq \text{nroFunc} \leq 5$;
- (2) **PredicadoMédia:** $5 < \text{nroFunc} \leq 50$;
- (3) **PredicadoGrande:** $50 < \text{nroFunc}$.

Como pode-se observar, o intervalo de valores que determina o tipo específico, para o qual um tipo genérico foi especializado, é armazenado nos predicados e representado na parte inferior dos retângulos que representam os subtipos. O atributo IdEmpresa é o identificador do supertipo e será repassado para os subtipos no momento do mapeamento.

Como a restrição de participação para esta OAG é total ($rP=falso$) e disjunta ($rS=falso$), todo tipo genérico deve necessariamente ser especializado para um único tipo específico.

A Figura 20_b, ilustra o mapeamento, do exemplo da Figura 20_a, usando a opção de mapeamento OM2. Obteve-se como resultado, uma tabela para mapear o supertipo e uma para cada subtipo como na OM1. A diferença entre essas duas opções, é que em OM2 existe uma estrutura sintática específica para mapear o atributo critério (NroFunc), que neste caso é

importante, pois tal atributo armazena uma informação relevante que poderá ser acessada posteriormente por alguma consulta.

Além disso, considere-se que para esta aplicação não sabe-se a priori o subtipo para o qual um tipo genérico foi especializado. Dessa forma, quando for preciso recuperar informações específicas de uma determinada empresa, é preciso avaliar o atributo *NroFunc* de acordo com os predicados (1), (2) e (3), para identificar o subtipo para o qual a empresa foi especializada. Portanto, o conjunto de atributos critério deve ser representado em alguma estrutura sintática. Nesta OM2, ele foi representado junto com os atributos do tipo genérico, na relação *G*.

(a)

(b)

$G = \text{Empresa}$

<u>IdEmpresa</u>	Nome	Endereço	CGC	NroFunc
------------------	------	----------	-----	---------

$E_1 = \text{Pequena}$

<u>IdEmpresa</u>	NomeProprietário	FériasColetivas
------------------	------------------	-----------------

$E_2 = \text{Média}$

<u>IdEmpresa</u>	QtdDeptos	MédiaSalário
------------------	-----------	--------------

$E_3 = \text{Grande}$

<u>IdEmpresa</u>	LocalMatriz	ValeTransporte	CestaBásica
------------------	-------------	----------------	-------------

Figura 20: Mapeamento de uma OAG segundo OM2

É importante ressaltar que esta OM é apropriada somente para OAGs em que verifica-se a restrição de disjunção, uma vez que o atributo critério *C* é representado juntamente com os atributos do tipo genérico (AG). Com relação a restrição de participação, esta OM é válida tanto para participação total quanto para participação parcial. No primeiro caso, o valor do atributo

critério nunca será nulo, pois todo tipo genérico será especializado para algum tipo específico. No segundo caso, o atributo critério poderá ter valor nulo para algumas instâncias.

Considerando-se as três maneiras em que o atributo critério pode ser representado, que são: um conjunto de atributos monovalorados; um conjunto de atributos onde ao menos um deles é multivalorado; e, uma expressão sobre um ou mais atributos, a OM2 suporta a primeira e última representação do critério. Esta última, foi ilustrada no exemplo da Figura 20.

4.3.3 Opção de Mapeamento 3 (OM3)

O esquema de relações gerado pela opção de mapeamento 3 é descrito da seguinte maneira:

$$\begin{aligned} G &= \{ \underline{K} \cup AG \} \\ E_e &= \{ \underline{K} \cup AE_e \} & 1 \leq e \leq ne \\ D &= \{ \underline{K} \cup \underline{C} \} \end{aligned}$$

Nesta opção cria-se uma relação G para o tipo genérico, ne relações E_e para cada tipo específico, onde ne é o número de subtipos, e uma relação D que mapeia o atributo critério. A relação G possui o conjunto de atributos identificadores (K) como chave primária, e o conjunto de atributos do tipo genérico (AG). Cada relação E_e possui o conjunto de atributos identificadores (K) herdado do tipo genérico como chave primária, mais o conjunto de atributos específicos do subtipo (AE_e). A relação D possui o conjunto de atributos identificadores (K), também herdados do tipo genérico, e o atributo critério que define a especialização, ambos como chave primária.

Para ilustrar esta opção de mapeamento, considere uma empresa onde os funcionários podem trabalhar como analista de sistemas ou como programador. O funcionário que se destacar em sua função receberá um cargo adicional: o de supervisor de seu departamento.

A Figura 21_a representa este exemplo através de uma OAG onde um Funcionário pode ser especializado de acordo com o atributo critério atribuições em Supervisor, Analista e Programador. As restrições definidas para esta OAG é de participação parcial (rP =verdadeiro) e de sobreposição (rS =verdadeiro).

(a)

(b)

G = Funcionário

<u>IdFunc</u>	FNome	LNome	Nascimento
---------------	-------	-------	------------

E₁ = Supervisor

<u>IdFunc</u>	SalárioAdic	LocalDeptoSuperv
---------------	-------------	------------------

E₂ = Analista

<u>IdFunc</u>	TpExperiência	Formação
---------------	---------------	----------

E₃ = Programador

<u>IdFunc</u>	LP_Preferência	BD_Preferência
---------------	----------------	----------------

D = D Funcionário

<u>IdFunc</u>	Atribuições
---------------	-------------

Figura 21: Mapeamento de uma OAG segundo OM3

Neste exemplo, o critério é determinado pelo valor de um único atributo multivalorado cujo domínio é $\text{dom}(\text{atribuições}) = \{\text{programação, lay-out, interface, documentação, modelador, DBA, rede, suporte, coordenação, gerência}\}$. Para verificar o subtipo para o qual um determinado objeto do tipo genérico foi especializado define-se:

- (1) **PredicadoProgramador** $\in \{\text{programação, lay-out, documentação}\}$;
- (2) **PredicadoAnalista** $\in \{\text{interface, modelador, rede, suporte}\}$;
- (3) **PredicadoSupervisor** $\in \{\text{DBA, coordenação, gerência}\}$.

Assim como a OM2, esta opção de mapeamento proporciona uma estrutura sintática para mapear o conjunto de atributos critério. Mas na OM2, tal atributo era representado na tabela que mapeia o supertipo, juntamente com AG, ficando restrito às OAGs onde verifica-se a disjunção.

No mapeamento resultante segundo a OM3, na Figura 21_b, define-se uma relação D_Funcionário para mapear o atributo critério, e como K e C são chaves da relação, esta OM pode ser utilizada em OAGs onde verifica-se a sobreposição.

Com a tabela D é possível verificar todas as atribuições de um funcionário. Considerando-se que, em uma consulta a tabela D_Funcionário, obteve-se a informação de que as atribuições de um determinado Funcionário é DBA, programação e documentação. Através da avaliação dos predicados (1), (2) e (3) conclui-se que tal Funcionário é Supervisor e Programador, podendo verificar suas informações específicas nas tabelas correspondentes.

Portanto, verifica-se a importância da tabela D_Funcionário por dois motivos:

1. é possível consultar todas as atribuições de um funcionário;
2. é possível determinar a(s) tabela(s) específicas a que um determinado funcionário foi especializado.

Assim, pode-se concluir que esta OM pode ser utilizada para qualquer uma das três maneiras em que o atributo critério pode ser representado. Principalmente, nos casos em que o critério pode ser representado por um conjunto de atributos, onde ao menos um deles é multivalorado.

4.3.4 Opção de Mapeamento 4 (OM4)

O esquema de relações gerado pela opção de mapeamento 4 é descrito da seguinte maneira:

$$E_e = \{ \underline{K} \cup AG \cup AE_e \} \quad 1 \leq e \leq ne$$

Nesta opção de mapeamento cria-se uma relação E_e para cada tipo específico, com $1 \leq e \leq ne$, onde ne é o número de subtipos. Cada relação E_e possui o conjunto de atributos identificadores (K) como chave primária, o conjunto de atributos do tipo genérico (AG), e o conjunto de atributos específicos de cada subtipo (AE_i).

A ilustração desta OM é exemplificada na Figura 22_a. Nesta figura representa-se uma OAG onde um tipo genérico Veículo é especializado em Carro, Caminhão e Ônibus, de acordo com o atributo critério transporte cujo domínio é $\text{dom}(\text{transporte}) = \{\text{particular}, \text{carga e coletivo}\}$. Os valores de inicialização dos subtipos Carro, Caminhão e Ônibus são PA (particular), CA (carga) e CO (coletivo), respectivamente.

(a)

(b)

$E_1 = \text{Carro}$

<u>Placa</u>	QtidadePortas	PotenciaMotor
--------------	---------------	---------------

$E_2 = \text{Caminhão}$

<u>Placa</u>	NroEixos	CargaMax
--------------	----------	----------

$E_3 = \text{Ônibus}$

<u>Placa</u>	NroPassageiros	NomeEmpresa
--------------	----------------	-------------

Figura 22: Mapeamento de uma OAG segundo OM4

A Figura 22_b mostra o mapeamento deste exemplo segundo a OM4. Com esta OM cria-se uma tabela para representar cada tipo específico (Carro, Caminhão e Ônibus). Nota-se que não existe uma tabela que mapeia os atributos do tipo genérico (Veículo), ideal para este exemplo, pois não existem atributos genéricos. Caso existissem, estes atributos teriam que ser repassados para cada tipo específico. Nota-se também, que não existe uma estrutura sintática para mapear o atributo critério (transporte), portanto, esta OM é apropriada para situações em que não é relevante manter armazenada a informação do atributo critério. Consequentemente, a busca pelas informações já é direcionada à tabela de interesse.

O atributo Placa é o atributo identificador que deverá ser repassado para cada subtipo E_e .

Conforme definido na Figura 22_a, a restrição de participação é total ($rP=falso$). Com isso, é possível garantir que nenhuma informação será perdida no momento do mapeamento, uma vez que não existe uma tabela para mapear o tipo genérico. Vale ressaltar que, com a definição da restrição de disjunção ($rS=falso$) não seria preciso preocupar-se em manter a consistência das informações do tipo genérico entre as E_e 's, caso existissem, uma vez que tais informações seriam repassadas para um único tipo específico, ou seja, uma única E_e .

4.3.5 Opção de Mapeamento 5 (OM5)

O esquema de relação da opção de mapeamento 5 é descrito da seguinte maneira:

$$E_e = \{ \underline{K} \cup C \cup AG \cup AE_e \} \quad 1 \leq e \leq ne$$

Na OM5 cria-se uma relação E_e para cada tipo específico, com $1 \leq e \leq ne$, onde ne é o número de subtipos. Esta relação possui o conjunto de atributos identificadores (K) como chave primária, o conjunto de atributos genéricos do supertipo (AG), o conjunto de atributos específicos de cada subtipo (AE_i) e o atributo que mapeia o critério (C).

No exemplo da Figura 23_a, considera-se a especialização de Veículo em um outro contexto. Neste exemplo, a especialização dá-se pelo atributo critério capacidade de carga. Dessa forma, um Veículo será especializado de acordo com o atributo critério capacidade de carga a partir da avaliação dos predicados:

(1) PredicadoPasseio: $0 < \text{carga} < 400\text{Kg}$

(2) PredicadoUtilitário: $400\text{Kg} \leq \text{carga} \leq 1000\text{Kg}$

As restrições definidas nesta OAG são as restrições de disjunção ($rS=falso$) e participação total ($rP=falso$). Assim como a OM4, deve-se verificar a restrição de participação total para utilizar esta opção para que as informações do supertipo não sejam perdidas.

O mapeamento deste exemplo, segundo a OM5, é mostrado na Figura 23_b.

As mesmas considerações feitas para a OM4 também valem para esta OM. A diferença entre elas é a inclusão, na OM5, de uma estrutura sintática (C) para mapear o conjunto de atributos critério. Como C pertence às relações E_e , o objetivo de defini-lo não recai no uso da avaliação do critério para determinar o(s) subtipos(s) para o qual um determinado tipo genérico foi especializado, mas sim na verificação da informação de tal atributo, neste caso representado por capacidade de carga.

Assim, esta OM deve ser utilizada quando o critério é representado por uma expressão sobre um ou mais atributos.

(a)

(b)

$E_1 = \text{Passeio}$

<u>Placa</u>	CapacidadeCarga	NroPortas	ArCondicionado	Bagageiro
--------------	-----------------	-----------	----------------	-----------

$E_2 = \text{Utilitário}$

<u>Placa</u>	CapacidadeCarga	PrazoEntrega	CargaPercível
--------------	-----------------	--------------	---------------

Figura 23: Mapeamento de uma OAG segundo OM5

4.3.6 Opção de Mapeamento 6

O esquema de relação da opção de mapeamento 6 é descrito da seguinte maneira:

$$E_e = \{\underline{K} \cup AG \cup A E_e\} \quad 1 \leq e \leq ne$$

$$D = \{\underline{K} \cup \underline{C}\}$$

Nesta opção de mapeamento cria-se uma relação E_e para cada tipo específico, com $1 \leq e \leq ne$, onde ne é o número de subtipos, e uma relação D (Domínio) que mapeia o atributo critério. Cada relação E_e possui o conjunto de atributos identificadores (K) como chave primária, o conjunto de atributos genéricos do supertipo (AG) e o conjunto de atributos específicos de cada subtipo (AE_e). A relação D possui o conjunto de atributos identificadores (K) e o atributo critério C que define a especialização, ambos como chave primária. A chave primária das relações E_e 's é propagada para a relação D .

Considera-se uma aplicação de um centro médico (CM) onde deseja-se armazenar informações a respeito de laboratórios de análises clínicas. No CM existem laboratórios que

atendem pacientes particular, outros atendem pacientes particular e por algum convênio, e os que atendem apenas por algum convênio. Para cada convênio existe um andar reservado no CM para que os laboratórios conveniados atendam os pacientes neste andar determinado. Para os laboratórios que atendem pacientes particular, um outro andar específico do CM é determinado.

A Figura 24_a ilustra esta situação através de uma OAG onde um Laboratório é especializado de acordo com o tipo de atendimento que realiza, representado pelo atributo critério TipoAtend, em Particular e Conveniado. As restrições definidas para esta OAG são de participação total (rP=falso) e sobreposição (rS=verdadeiro).

(a)

(b)

$E_1 = \text{Particular}$

<u>Nome</u>	HorárioAtend	LocalLab
-------------	--------------	----------

$E_2 = \text{Conveniado}$

<u>Nome</u>	NomeConvênio	TelConvênio	LocalAtend
-------------	--------------	-------------	------------

$D = D_Laboratório$

<u>Nome</u>	<u>TipoAtend</u>
-------------	------------------

Figura 24: Mapeamento de uma OAG segundo OM6

O mapeamento desta OAG segundo OM6 é mostrado na Figura 24_b, onde além das tabelas Particular e Conveniado que mapeiam os subtipos, existe também a tabela D_Laboratório que mapeia o atributo critério. Vale ressaltar que os atributos K e C, desta última tabela são chaves da tabela D_Laboratório; consequentemente, esta OM é apropriada para os casos em que verifica-se a sobreposição.

Assim, com a tabela *D_Laboratório* é possível recuperar, por exemplo, todos os laboratórios que atendem particular e por algum convênio também, através da busca pelo nome do laboratório.

Uma outra situação em que a tabela *D_Laboratório* é eficiente recai no caso em que um paciente deseja marcar uma consulta em um determinado laboratório, mas não sabe se tal laboratório é credenciado em algum convênio. Através de uma busca pelo nome do laboratório na tabela *D_Laboratório* é possível verificar se este laboratório atende por algum convênio ou não. Caso a resposta for positiva, é possível consultar a tabela *Conveniado* para verificar informações mais específicas a respeito do atendimento.

Se não existisse a tabela *D_Laboratório*, esta pesquisa deveria ser feita na tabela *Conveniado* o que tornaria o acesso mais lento, uma vez que esta tabela tem mais atributos do que a tabela *D_Laboratório*.

É importante ressaltar que, nos casos em que ocorre a sobreposição e existem atributos genéricos no supertipo, estas informações genéricas seriam replicadas nas tabelas que mapeiam os subtipos, e isto pode gerar inconsistências.

Portanto, esta OM é ideal para os casos de sobreposição e que não existem atributos genéricos no supertipo.

Com relação às formas de representação do atributo critério, assim como na OM3, a OM6 pode ser utilizada para qualquer uma das três maneiras de representação do critério.

4.3.7 Opção de Mapeamento 7 (OM7)

O esquema de relação da opção de mapeamento 7 é descrito da seguinte maneira:

$$E_e = \{\underline{K} \cup AG \cup AE_e\} \quad 1 \leq e \leq ne$$

$$D = \{\underline{K} \cup C\}$$

Deve-se observar que a principal diferença entre esta OM7 e a OM6 é que, na tabela *D* resultante, o critério (*C*) não é definido como chave, juntamente com *K*.

Nesta opção de mapeamento cria-se uma relação E_e para cada tipo específico, com $1 \leq e \leq ne$, onde ne é o número de subtipos, e uma relação *D* (Domínio) que mapeia o atributo critério. Cada relação E_e possui o conjunto de atributos identificadores (*K*) como chave primária, o conjunto de atributos genéricos do supertipo (*AG*) e o conjunto de atributos específicos de cada subtipo (AE_e). A relação *D* possui o conjunto de atributos identificadores (*K*) como chave primária e o

atributo critério (C) que define a especialização. A chave primária das relações E_e 's é propagada para a relação D.

Esta OM é apropriada para situações em que ocorre a restrição de disjunção pois o conjunto de atributos critério não faz parte da chave na relação D, como ocorria na OM6. E ainda, o critério não pode ser representado por um conjunto de atributos onde um deles é multivalorado. As considerações feitas na opção de mapeamento 6 também valem para esta opção, com a ressalva de que deve-se verificar a restrição de disjunção para utilizá-lo.

A tabela domínio, D, pode ser utilizada para verificar o subtipo para o qual um determinado tipo genérico foi especializado, identificando a tabela onde a busca por informações específicas deve ser realizada.

Assim como em OM5, o atributo critério pode ser representado por uma expressão sobre um ou mais atributos, com a diferença que nesta OM, C é representado em uma relação D independente de cada subtipo E_e .

4.3.8 Opção de Mapeamento 8 (OM8)

O esquema de relação da opção de mapeamento 8 é descrito da seguinte maneira:

$$G = \{\underline{K} \cup AG \cup \{AE_1 \cup AE_2 \cup \dots \cup AE_{ne}\}\}$$

Nesta opção de mapeamento cria-se uma única relação G que possui como chave primária o conjunto de atributos identificadores do tipo genérico (K). O conjunto de atributos genéricos (AG) e o conjunto de atributos específicos de cada subtipo (AE_e), com $1 \leq e \leq ne$, também pertencem à relação G.

Nesta opção de mapeamento não existe uma estrutura sintática para mapear o conjunto de atributos critério. Por isso, esta OM é apropriada para situações em que, a determinação do critério pode ser obtida por uma avaliação simples sobre os atributos dos tipos específicos A_e . Por exemplo, é possível verificar se um objeto é de um determinado subtipo, ou não, verificando se os valores dos atributos A_{ei} dessa especialização, são nulos, ou não.

Para adotar esta opção de mapeamento é apropriado que a participação seja total, e que, os objetos do tipo genérico sejam especializados para mais de um tipo específico. Caso contrário, seria adequado se existissem poucos atributos em cada tipo específico, para que a tabela resultante não tenha muitos atributos com valores nulos.

A Figura 25_a ilustra uma OAG onde Trem é especializado de acordo com o atributo critério Transporte em Carga e Passageiro, com as restrições de participação total ($rP=falso$) e sobreposição ($rS=verdadeiro$). Para este exemplo, considera-se que os trens transportam carga e/ou passageiros.

A Figura 25_b ilustra o mapeamento, do exemplo da Figura 25_a, segundo OM8. Para descobrir o tipo de Transporte de um determinado Trem, deve-se verificar os valores dos atributos TonelMax e NroPassageiros. Por exemplo, se o valor do atributo TonelMax for não nulo, para um determinado objeto Trem, pode-se concluir que tal objeto foi especializado para o subtipo Carga. A mesma verificação pode ser feita para o atributo NroPassageiros, para decidir se um determinado objeto Trem foi especializado para o subtipo Passageiro.

(a)

(b)

G = Trem

<u>IdTrem</u>	NroVagões	Trajeto	UltimaManutenção	TonelMáx	NroPassageiros
---------------	-----------	---------	------------------	----------	----------------

Figura 25: Mapeamento de uma OAG segundo OM8

Neste exemplo, nota-se ainda, que cada tipo específico possui um único atributo. Nas situações em que existem muitos atributos nos tipos específicos, sempre que ocorrer uma consulta ao supertipo ou a um único subtipo, muitas informações que não interessam à consulta, serão trazidas para memória.

Por outro lado, se a frequência de acesso a informações conjuntas dos subtipos, ou do supertipo e dos subtipos, for alta, esta opção é preferível aquelas que tratam o supertipo e os subtipos em tabelas separadas. Estes são alguns pontos que devem ser avaliados, pelo desenvolvedor, no momento da escolha por esta opção de mapeamento.

4.3.9 Opção de Mapeamento 9 (OM9)

O esquema de relação da opção de mapeamento 9 é descrito da seguinte maneira:

$$G = \{\underline{K} \cup C \cup AG \cup \{AE_1 \cup AE_2 \cup \dots \cup AE_{ne}\}\}$$

Na OM9 cria-se uma única relação G que possui como chave primária o conjunto de atributos identificadores do tipo genérico (K). O conjunto de atributos genéricos (AG), o conjunto de atributos específicos de cada subtipo (AE_e), com $1 \leq e \leq ne$, e o atributo critério (C), também pertencem à relação G .

Para ilustrar esta opção de mapeamento considera-se o exemplo da Figura 26_a, onde os vagões dos trens de carga, ilustrado no exemplo anterior, são especializados em:

- gondola, representando os vagões abertos, utilizados, por exemplo, para o transporte de minério de ferro;
- fechado, utilizados para o transporte de mercadorias perecíveis;
- tanque, para o transporte de líquidos;
- animais, utilizados principalmente para o transporte de gado;

O atributo critério, *Tipo*, é o atributo que define a especialização do tipo genérico *Carga*, de acordo com as restrições de participação total ($rP=falso$) e disjunção ($rS=falso$).

(a)

(b)

$G = Trem$

<u>IdTrem</u>	TonelMax	Comprimento	Tara	Tipo	Volume	Altura	Unidade	Volume
---------------	----------	-------------	------	------	--------	--------	---------	--------

Figura 26: Mapeamento de uma OAG segundo OM9

Caso o exemplo da Figura 26_a fosse mapeado utilizando a opção de mapeamento 8, o critério não poderia ser determinado por uma avaliação simples sobre os atributos dos tipos específicos. Isto ocorre porque existem atributos que são compartilhados por mais de um tipo de vagão. Por exemplo, o atributo volume é compartilhado, entre os subtipos, Gondola e Tanque. Dessa forma, não é possível determinar o subtipo para o qual um tipo genérico foi especializado, apenas por uma verificação dos atributos dos subtipos, que possuem valores não nulos.

Assim, fica evidente que a OM9 é a opção adequada para o mapeamento do exemplo ilustrado na Figura 26_a, uma vez que ela fornece uma estrutura sintática (C) para armazenar informações sobre o critério. A Figura 26_b mostra tal mapeamento, onde o critério está representado pelo atributo Tipo, cujo domínio é $\text{dom(Tipo)} = \{\text{Gondola, Fechado, Tanque, Animais}\}$. Nota-se que, como C pertence a relação G, esta OM é utilizada somente quando verifica-se a restrição de disjunção.

Além da representação do critério por um atributo monovalorado, conforme ilustrado no mapeamento da Figura 26, a OM9 pode ser utilizada também, quando o critério é representado por uma expressão sobre um ou mais atributos.

4.3.10 Opção de Mapeamento 10 (OM10)

O esquema de relação da opção de mapeamento 10 é descrito da seguinte maneira:

$$G = \{\underline{K} \cup AG \cup \{AE_1 \cup AE_2 \cup \dots \cup AE_{ne}\} \cup \{s_1, s_2, \dots, s_{ne}\}\}$$

Na OM10 cria-se uma única relação G que possui como chave primária o conjunto de atributos identificadores do tipo genérico (K). O conjunto de atributos genéricos (AG), o conjunto de atributos específicos de cada subtipo (AE_e), com $1 \leq e \leq ne$, e o conjunto de atributos critério ($\{s_1, s_2, \dots, s_m\}$), também pertencem à relação G.

Nesta opção de mapeamento o critério é representado pelo conjunto de atributos $\{s_1, s_2, \dots, s_{ne}\}$ que são atributos *booleanos* com domínio {verdadeiro, falso}. Caso o valor de “ s_{ne} =verdadeiro” para um determinado objeto, isto indica que tal objeto foi especializado para o subtipo E_{ne} ; caso contrário “ s_{ne} = falso”.

Esta OM é ideal para situações em que ocorre a restrição de sobreposição e a participação for total, o que resultará em poucos atributos nulos.

Considere o exemplo da Figura 27_a onde uma Conta pode ser especializada em Corrente (conta corrente) e Poupança de acordo com o atributo critério Movimentação, em

uma OAG sobreponível ($rS=\text{verdadeiro}$) e de participação total ($rP=\text{falso}$). O atributo *NroConta* (número da conta) é o atributo chave, e o atributo *saldo* é o único atributo genérico desta OAG. Como atributos específicos tem-se *LimChEspecial* (limite do cheque especial) e *SaldoMínimo* que pertencem ao subtipo *Corrente* e o atributo *TaxaJuros* que pertence ao subtipo *Poupança*.

O mapeamento deste exemplo, segundo OM10, é mostrado na Figura 27_b, onde o atributo critério é representado por S_{Corrente} e $S_{\text{Poupança}}$. Portanto, esta OM é indicada para situações em que não há necessidade de se armazenar a informação do atributo critério, uma vez que este atributo é representado por atributos *booleanos* com domínio {verdadeiro, falso}. Assim, o conjunto de atributos critério é utilizado para a verificação do subtipo para o qual um objeto foi especializado.

(a)

(b)

G = Conta

<u>NroConta</u>	Saldo	S_{Corrente}	LimChEspecial	SaldoMínimo	$S_{\text{Poupança}}$	TaxaJuros
-----------------	-------	-----------------------	---------------	-------------	-----------------------	-----------

Figura 27: Mapeamento de uma OAG segundo OM10

4.4 Avaliação das dez opções de mapeamento

A Tabela 8 mostra a avaliação das dez OM's propostas com relação às propriedades da generalização definidas na seção 4.2.7, e às restrições de sobreposição e participação. Para esta avaliação foram definidos os parâmetros: ideal, adequado, inadequado e não se aplica.

Estas propriedades foram discutidas nos exemplos apresentados, para ilustrar cada uma das opções de mapeamento propostas. Em particular, as propriedades de A a D, comumente discutidas na literatura, e tratadas nos capítulos anteriores, não receberão destaque nesta seção.

Tabela 8: Avaliação das dez opções de mapeamento

Propriedades e Restrições de uma OAG	Opções de Mapeamento									
	OM1	OM2	OM3	OM4	OM5	OM6	OM7	OM8	OM9	OM10
A – Restrição de sobreposição (S)	●	×	●	*●	×	*●	×	●	×	●
B – Restrição de disjunção (D)	●	●	●	●	●	●	●	○	●	○
C – Participação total (T)	●	●	●	●	●	●	●	●	●	●
D – Participação parcial (P)	●	●	●	×	×	×	×	○	○	○
E – Frequência de acesso ao conjunto de atributos que definem o critério (FAC)	×	●	●	×	●	●	●	×	●	●
F – O acesso ao critério é realizado sobre um conj. atributos monovalorados (ACMN)	×	●	●	×	×	●	●	×	●	●
G – O acesso ao critério é realizado por uma expressão sobre um ou mais atributos	×	●	●	×	●	●	●	×	●	×
H – O acesso ao critério é realizado sobre um conj. atributos, ao menos um, multivalorado (ACMT)	×	×	●	×	×	●	×	×	×	×
I – Frequência de acesso a informações conjuntas de vários subtipos (FAVS)	○	○	○	○	○	○	○	●	●	●
J – Frequência de acesso a informações de um único subtipo (FAUS)	●	●	●	●	●	●	●	●	●	●
K – Frequência de acesso a informações conjuntas do subtipo e do supertipo	○	○	○	●	●	●	●	●	●	●
L – Frequência de acesso a informações do subtipo ou do supertipo	●	●	●	●	●	●	●	●	●	●
M – Supertipo envolvido com outros construtores semânticos	●	●	●	○	○	○	○	●	●	●
N – Subtipo envolvido com outros construtores semânticos	●	●	●	●	●	●	●	○	○	○
O – Existem muitos atributos no supertipo (MAS)	●	●	●	○	○	○	○	○	○	○
P – Existem muitos atributos em cada subtipo	●	●	●	●	●	●	●	○	○	○
<div> <div>●</div> Ideal <div>●</div> Adequado <div>○</div> Inadequado <div>×</div> Não se aplica </div>										

* - considerando-se que não existem atributos genéricos no supertipo.

Com a Tabela 8 pôde-se observar que as propriedades que tratam a participação total (C), frequência de acesso a informações a informações de um único subtipo (J), e frequência de acesso a informações do supertipo ou do subtipo (L), foram avaliadas como ideais ou adequadas, para todas as opções de mapeamento.

Na propriedade que trata a frequência de acesso ao conjunto de atributos que definem o critério (E), observa-se que nas OMs 1, 4 e 8, tal propriedade não se aplica, pois não existe uma estrutura sintática para mapear o critério.

Na propriedade F, que verifica se o acesso ao critério é realizado sobre um conjunto de atributos monovalorados, as OMs 2, 3, 6, 7, 9 e 10 foram consideradas ideais. Uma observação deve ser feita com relação a OM5, onde considera-se que a propriedade F não se aplica, apesar

de existir uma estrutura sintática C , para mapear o critério. Isto ocorre, pois C , é mapeado em cada relação E_e , e por isto, perde o sentido defini-lo, já que nesta propriedade o teste do critério é realizado para determinar a tabela específica, a que, um determinado objeto pertence.

Com relação ao acesso ao critério, realizado por uma expressão sobre um ou mais atributos, tratado na propriedade G , as OMs 2, 3, 5, 6, 7 e 9 são consideradas ideais, pois neste caso é importante armazenar o valor exato do conjunto de atributos critério, para cada subtipo, e isto pode ser feito utilizando-se a estrutura sintática C , definida em cada uma destas opções de mapeamento. Comparando com o teste do critério anterior, a diferença recai na inclusão de OM5 como ideal, e na avaliação de OM10 como 'não se aplica'. Esta última avaliação deve-se ao fato que, o domínio do conjunto de atributos critério, de OM10, é: {verdadeiro, falso}.

A última propriedade que avalia o critério, a propriedade H , trata do acesso ao critério realizado sobre um conjunto de atributos, onde ao menos um deles é multivalorado. Apenas as OMs 3 e 6 são avaliadas como ideais, e para as demais, esta propriedade não se aplica. Isto porque, nas OMs 3 e 6 existe uma tabela domínio (D), onde é definido uma estrutura sintática para mapear o critério, representada por C , e C é chave desta tabela juntamente com K (chave primária). Assim, com esta tabela D , é possível mapear os atributos multivalorados, que compõem o critério.

Na avaliação das opções de mapeamento, com relação às propriedades I a P , ficou evidente os três grupos em que tais opções se dividem, que são: $G + \{E_i\}$, $\{E_i\}$ e G . Nas opções 1, 2 e 3, que pertencem ao grupo $G + \{E_i\}$, as propriedades de I a P foram igualmente avaliadas, entre as opções de mapeamento deste grupo, de acordo com os parâmetros definidos (ideal, adequado, inadequado e 'não se aplica'). O mesmo ocorre, com as opções 4, 5, 6 e 7, que pertencem ao grupo $\{E_i\}$, e com as opções 8, 9 e 10, do grupo G .

A propriedade I , que trata a frequência de acesso a informações conjuntas de vários subtipos, é considerada inadequada para as OMs de 1 a 7 e ideal para as OMs de 8 a 10, pois nestas últimas opções, todos os subtipos são mapeados para uma única tabela, juntamente com as informações do supertipo.

Na propriedade que trata a frequência de acesso a informações conjuntas do subtipo e do supertipo (K), foram consideradas inadequadas as OMs de 1 a 3, pois neste caso existe uma tabela para mapear o supertipo e uma para cada subtipo; as demais OMs foram consideradas ideais.

As propriedades M e N , avaliam o supertipo e subtipo envolvido com outros construtores semânticos. Considera-se mais apropriado armazenar as informações do supertipo e dos subtipos em tabelas separadas, por isto, os OMs 1, 2 e 3 foram avaliados como ideais.

Com relação as propriedades *O* e *P*, que avaliam respectivamente, a existência de muitos atributos no supertipo e a existência de muitos atributos em cada subtipo, as OMs 1 a 3 são consideradas ideais, pois nestas opções define-se uma tabela para mapear os atributos do supertipo e uma tabela para cada subtipo. As OMs de 4 a 7 foram avaliadas como inadequadas, com relação a propriedade *O*, pois neste caso, os atributos do genérico são repassados para cada subtipo havendo a necessidade de controlar as potenciais inconsistências. Ainda com relação a propriedade *O*, as OMs 8, 9 e 10 também foram avaliadas como inadequadas. Na verdade, nestas últimas opções, o desenvolvedor deverá avaliar a quantidade de subtipos também, pois se existirem muitos atributos no supertipo e muitos atributos nos subtipos, a escolha por estas opções fica inviável, uma vez que o resultado do mapeamento serão tabelas com muitos atributos.

Para a propriedade *P*, as OMs de 4 a 7 são consideradas adequadas, pois existe uma tabela para mapear cada subtipo, e as OMs de 8 a 10 são inadequadas por resultar em tabelas com muitos atributos.

4.5 *Árvore de Decisão*

Considerando-se as dez opções de mapeamento, as propriedades e restrições avaliadas na Tabela 8, definiu-se uma árvore de decisão, com o objetivo de automatizar o processo de mapeamento, auxiliando o desenvolvedor na escolha da OM adequada para o problema que deseja mapear.

Esta árvore é mostrada na Figura 28, e é a base conceitual sobre a qual se apóia a ferramenta de auxílio ao processo de mapeamento da abstração de generalização, de um Modelo Canônico para o Modelo Relacional, que será apresentada no capítulo 5. É importante ressaltar que, dependendo das características particulares de cada OAG, é possível encontrar uma OM adequada, mesmo que, nem todas as propriedades listadas na Tabela 8 tenham sido avaliadas.

Para árvore de decisão ilustrada na Figura 28, foram definidas algumas das propriedades, dentre aquelas mostradas na Tabela 8, e a sequência em que cada propriedade deve ser avaliada em um processo de mapeamento.

4.6 Comparação do método proposto com os principais métodos encontrados na literatura

Conforme citado anteriormente, vários autores (Teorey et al., 1986; Elmasri & Navathe, 1994; Silbershatz et al., 1997; Ceri & Fraternali, 1997) tratam o mapeamento da abstração de generalização. Todos estes autores propõem a definição de uma tabela para mapear o supertipo e uma para cada subtipo, onde os subtipos herdam a chave primária do supertipo.

Silbershatz destaca ainda que, se a generalização for exclusiva e total, pode-se definir uma tabela para cada subtipo, cada qual com seus próprios atributos, mais os atributos genéricos herdados do supertipo.

Teorey aborda a definição de uma única tabela que possui todos os atributos genéricos e os atributos de todos os subtipos, considerando o tratamento de valores nulos na tabela.

Elmasri e Navathe tratam amplamente todas estas propostas, inclusive dando um tratamento ao ‘atributo tipo’ que define a especialização. Este atributo, quando pertence à tabela que mapeia o genérico, pode indicar o subtipo para o qual um determinado objeto foi especializado. E ainda, é possível verificar se um objeto foi especializado para mais de um subtipo.

As dez opções de mapeamento para generalização, definidas neste trabalho, estendem as propostas encontradas na literatura, proporcionando ao desenvolvedor um conjunto mais detalhado de possibilidades de mapeamento, onde um deles deve ser adequado ao problema que deseja-se mapear. É importante destacar que, estas dez opções de mapeamento, cobrem todos os casos encontrados na literatura, além de apresentar cinco novos casos que, tanto quanto é de nosso conhecimento, não foram estudados anteriormente em nenhum outro trabalho. Os cinco novos casos, explorados neste trabalho, corresponde às opções de mapeamento 2, 3, 5, 6 e 7.

Na definição destas novas opções, um destaque especial foi dado ao conjunto de atributos denominado critério, de uma OAG. Pode-se considerar, que a idéia de um conjunto de atributos atuando como critério, da abstração de generalização, é um conceito novo, que refina o conceito de ‘atributo tipo’, definido por Elmasri e Navathe.

O conjunto de atributos que compõem o critério, indicam os subtipos para o qual um objeto foi especializado, otimizando dessa forma, as consultas à subtipos específicos, e ‘generaliza’ o tratamento da abstração de generalização. Além disso, este trabalho aproveita o domínio dos atributos que compõem o critério, e define parâmetros, importantes para caracterizar uma OAG. A representação implícita ou explícita destes atributos, definem em que

situações é preciso armazená-los na base, e por este motivo, este assunto foi amplamente explorado neste trabalho.

4.7 Abrangência das Opções de Mapeamento

Dentre os elementos estruturais definidos na seção 4.2, para a descrição das OAGs, pode-se destacar: o conjunto de atributos critério; as restrições e propriedades da generalização; os predicados; e as regras de inicialização de valores, para cada tipo de objeto especializado. Todos esses elementos são sintaticamente representados, ou apoiam-se, nos atributos definidos nos diversos tipos (subtipos e supertipos) criados nas hierarquias de generalização. Até este ponto do trabalho, considerou-se que esses atributos são os usuais, disponíveis nos gerenciadores de dados relacionais e linguagens de programação atuais, como por exemplo, os atributos do tipo número (integer, float, etc), valores booleanos e pequenas cadeias de caracteres (char, varchar, etc).

Considerando esses tipos de dados para os atributos, as dez opções de mapeamento apresentadas atendem a todos os casos tradicionais de representação de situações da abstração de generalização. Isto pode ser verificado uma vez que, qualquer expressão sobre os atributos de um supertipo que resulte em um valor booleano, verdadeiro ou falso, pode ser utilizada como predicado de um subtipo, e ao mesmo tempo, qualquer regra de inicialização pode ser descrita através de uma combinação de valores dos atributos do supertipo, com dados de inicialização fornecidos pela aplicação, para a criação de cada objeto de um determinado subtipo.

Teoricamente, qualquer tipo de objeto deve ser representado em um mapeamento para o Modelo Relacional através de um conjunto de relações. Considerando-se o supertipo e cada um dos subtipos de uma OAG, existem três possibilidades de representação: uma relação para todos os tipos (grupo de opções correspondente a mapear toda a OAG em apenas uma relação G); uma relação para cada subtipo, incluindo as informações genéricas (grupo de opções correspondente a mapear cada subtipo em uma relação Ei); e uma relação para o supertipo e mais uma relação para cada subtipo, (grupo de opções correspondente a mapear cada subtipo em uma relação G mais um conjunto de relações Ei).

O conceito de atributos selecionados como critério, para cada OAG, permite definir todas as opções de como podem ser suportadas as chaves das relações nos três grupos. Como todas as possibilidades (significativas) estão contempladas nas dez opções desenvolvidas, pode-se dizer que todas as necessidades de representação de OAGs, baseadas em atributos tradicionais, estão contempladas no processo de mapeamento desenvolvido.

O caso de atributos que representam tempo, deve ser analisado de maneira separada, dependendo de como o gerenciador de dados relacional trata tais atributos. Segundo Jensen & Snodgrass (1994) e Jensen et al. (1993), do ponto de vista do suporte à dimensão tempo, uma base de dados pode ser não temporizada, sequencial (Snapshot), histórica ou temporal, dependendo de como o tempo é tratado, em relação ao tempo considerado pela aplicação (tempo válido), ou pelo gerenciador (tempo físico).

Uma base não temporizada não suporta nenhum tipo de tempo. Uma base de dados que suporta o tempo físico significa que o gerenciador registra o momento em que todas as operações são solicitadas pela aplicação, e pode apresentar o estado da base em qualquer tempo anterior (representado pelo comando do SQL estendido *as of*). Assim, operações de remoção de um objeto, por exemplo, não apagam o objeto da base, de fato, mas indicam que seu tempo de validade está expirado devido a execução do comando *delete*. Uma consulta envolvendo um comando *as of* <determinada data>, retorna todos os objetos cujo tempo de vida era válido na data indicada. As bases de dados que suportam o tempo físico são chamadas de sequenciais ou temporais.

Uma base de dados que suporta o tempo válido, significa que o gerenciador permite à aplicação, determinar quando a operação solicitada se torna válida, independente de quando a operação é realmente (fisicamente) solicitada. O operador que representa tempo válido, em uma linguagem SQL estendida, é *when*. As bases de dados que suportam o tempo válido são chamadas históricas ou temporais (Tansel et al., 1993).

Uma base sequencial suporta apenas o tempo físico, e pode ser entendida como uma base de dados em que, o estado da base, pode ser “refeito” para qualquer situação anterior. Uma base histórica suporta apenas o tempo válido, e pode ser entendida como uma base que armazena dados quando, do ponto de vista da aplicação, refere-se a um dado válido. Uma base de dados temporal suporta ambos os tipos de tempo, e permite consultar qualquer um deles, em qualquer combinação. Por exemplo, se a inserção de um objeto for feita hoje, retroativo a dez dias, qualquer consulta que inclua o operador *when*, indicando dois dias atrás, irá incluir o novo objeto. No entanto, se na mesma consulta for definido o operador *as of*, uma pesquisa na data do dia anterior, não irá encontrar o objeto em questão, pois no dia anterior, do ponto de vista do gerenciador, o objeto não havia sido fisicamente incluído (Ozsouoglu & Snodgrass, 1995).

Em todos os casos de bases de dados que suportam tempo, as diversas operações sobre os objetos (inclusão, remoção e atualização) podem afetar os atributos que fazem parte do critério, pois é possível que os atributos “troquem de subtipo”, navegando dinamicamente na hierarquia de subtipos. Embora a propriedade “um objeto pode evoluir para um novo subtipo”, não tenha

sido incluída como uma propriedade a ser considerada na decisão por uma opção de mapeamento adequada, essa possibilidade é permitida em várias opções de mapeamento. Isto é possível, devido a manutenção do identificador de objetos, através de sua chave comum em toda a hierarquia de generalização, e devido a existência dos atributos critério.

Considerando-se que a identificação de um objeto em um subtipo em particular é feita através da avaliação dos valores dos atributos do critério, se uma base de dados que suporta tempo for utilizada (sequencial, histórica ou temporal), a evolução dos valores dos atributos do critério permite o rastreio dos subtipos assumidos pelo objeto. Isso garante que a evolução de tipos de um objeto corresponda a evolução do objeto representado pela chave comum, inserido, removido e atualizado nas diversas relações que mapeiam a hierarquia de generalização do qual esse objeto faz parte. Portanto, pode-se dizer que o processo de mapeamento proposto neste trabalho é abrangente também para suportar bases de dados sequenciais, históricas e temporais.

Por outro lado, não existem restrições sobre como devem ser os atributos que fazem parte tanto do critério, como dos predicados e das regras de inicialização, incluindo-se a possibilidade de que eles possam, por sua vez, ser atributos cujo tipo de dados seja “tempo”. Assim, o suporte a regras de comportamento e evolução dos objetos pode ser incluído nas regras que mapeiam o comportamento das relações, sejam estas, ou não, baseadas em eventos temporais, possibilitando que o mapeamento proposto possa ser feito em bases de dados ativas e reativas.

4.8 Considerações Finais

Com base em exemplos, extraídos de aplicações reais, foram apresentadas as características particulares de dez opções de mapeamento, onde cada opção é caracterizada por um esquema de relações, resultante do processo de mapeamento. Destas opções, cinco delas nunca haviam sido tratadas, do ponto de vista da análise teórica, embora sejam necessárias e, possivelmente utilizadas na prática por profissionais que precisam de tais resultados.

Para a definição destas novas opções de mapeamento, que estendem e unificam os trabalhos existentes na literatura até o momento, foi importante a definição do conceito de critério, e as maneiras de representá-lo. Pode-se afirmar que, as opções apresentadas, cobrem todas as situações potenciais de mapeamento, pois tratam amplamente da utilização dos três grupos de relações, resultantes do mapeamento, G , $\{E_i\}$, e $G + \{E_i\}$, e em cada caso, são tratadas todas as maneiras de representar o critério.

À partir da definição de um conjunto de características básicas, que representam toda a semântica da abstração de generalização, incluindo as propriedades relevantes para o processo de decisão do mapeamento, apresentou-se uma árvore de decisão, onde definiu-se como e em que situações, cada propriedade deve ser considerada.

Assim, tais definições resultaram em um processo de mapeamento que, pela primeira vez na literatura, permite automatizar totalmente o processo de mapeamento da abstração de generalização, baseando-se em parâmetros facilmente e precisamente qualificados e quantificados.

No próximo capítulo apresenta-se a ferramenta E² SIRIUS versão 1.1, desenvolvida para suportar o mapeamento da abstração de generalização.

Capítulo 5—Protótipo de uma Ferramenta de Suporte ao Mapeamento

5.1 Considerações Iniciais

Para que uma ferramenta apoie o mapeamento de ocorrências da abstração de generalização (OAG), para gerenciadores relacionais, ela deve conduzir o desenvolvedor a definir algumas informações relevantes, para que o mapeamento resultante seja executado adequadamente.

No Capítulo 4 estabeleceu-se um conjunto de características que um modelo de dados canônico, orientado para o suporte da abstração de generalização, deve suportar. Apoiado nesse modelo canônico, foram estabelecidas dez opções de mapeamento para a abstração em questão. Um dos objetivos que foram considerados para o desenvolvimento dessas opções foi que, o processo de mapeamento deveria ser definido até o ponto de ser possível sua execução automática. Evidentemente, a possibilidade de automatizar esse processo depende do modelo de dados original ser capaz de representar todas as informações necessárias para os diversos passos e decisões, que devem ser tomadas durante o processo de mapeamento. Para comprovar tanto a validade do processo de mapeamento desenvolvido, quanto que este objetivo foi efetivamente atingido, foi desenvolvido o protótipo de uma ferramenta automática de mapeamento das OAGs de um esquema orientado a objetos, para uma base de dados relacional.

Esse protótipo deve partir de um esquema original, de uma aplicação, representado em um modelo conceitual, e para isso, é necessário que exista um recurso computacional a ser utilizado previamente à este protótipo. Como o modelo canônico é semelhante ao modelo SIRIUS, para o qual já existia um editor de esquemas denominado E²SIRIUS (Araujo, 1998a), foi possível verificar que todas as características, definidas no modelo canônico, são encontradas também no modelo SIRIUS. Algumas extensões foram realizadas em E²SIRIUS para que essa

ferramenta pudesse suportar os demais recursos do modelo canônico, necessários ao processo de mapeamento, e a seguir, foi criado um novo módulo, conceitualmente acoplado ao editor, que é responsável pelo processo de mapeamento automático das OAGs existentes em esquemas efetuados com esse editor.

Para apresentar o protótipo da ferramenta de mapeamento construída, este capítulo foi organizado da seguinte forma: na seção 5.2 descreve-se os recursos já existentes na ferramenta E²SIRIUS (Araújo, 1998a), importantes para este trabalho; na seção 5.3 descreve-se as extensões feitas ao editor, e apresenta-se o novo módulo de mapeamento acoplado a essa ferramenta. A implementação deste novo módulo permite que o desenvolvedor navegue na hierarquia de generalização, defina informações relevantes, resultando em uma opção de mapeamento apropriada, e verifique, de acordo com tal opção, o respectivo código SQL para a criação das tabelas em um gerenciador relacional.

5.2 A Ferramenta E² SIRIUS

O Editor de Esquemas E² SIRIUS (Araújo, 1998a) foi desenvolvido para sustentar a edição de modelagens no gerenciador de objetos SIRIUS/GO (Traina, 1991; 1996) com o propósito de apoiar o desenvolvedor durante a fase de análise.

Considerando-se que o gerenciador SIRIUS/GO ainda está sendo desenvolvido pelo grupo GBDI (Grupo de Banco de Dados e Imagens) do ICMC-USP, e não encontra-se suficientemente avançado para suportar todos os conceitos de SIRIUS, a atual versão do editor de esquemas utiliza um gerenciador relacional para armazenar as informações dos esquemas editadas por seus desenvolvedores.

A ferramenta E² SIRIUS suporta a edição de todas as abstrações do modelo SIRIUS: classificação/instanciação, agregação/separação, generalização/especialização e composição/decomposição. O protótipo da ferramenta de suporte ao mapeamento, acoplado a uma versão de E² SIRIUS, especialmente estendida para incluir todos os recursos necessários ao suporte da abstração de generalização/especialização, tal como previstos pelo modelo canônico, descrito no capítulo anterior. Essa nova versão do editor, que está referenciada como versão 1.1, tem as extensões descritas neste capítulo. Os formulários de E² SIRIUS, importantes para o desenvolvimento deste novo módulo, serão descritos nas próximas seções.

5.2.1 Definição de Objetos e Atributos

O primeiro passo para editar uma modelagem na ferramenta E² SIRIUS é definir os objetos e atributos que compõem tal modelagem. Para a definição de um novo objeto utiliza-se o botão +objeto, do formulário objeto, conforme é mostrado na Figura 29. No centro deste formulário representa-se um objeto com as seguintes informações:

- tipo do objeto: por *default*, todo objeto é do tipo 'meta tipo'. Isso é feito porque a ferramenta é basicamente um editor de esquemas, portanto os objetos definidos são normalmente tipos;
- tipo da colônia: onde as instâncias do objeto habitam, e por *default*, todo objeto habita a colônia global. A colônia global é a única que o sistema garante sempre existir. No entanto, para uma boa prática de programação, é interessante que cada esquema seja armazenado em uma ou mais colônia, especialmente criadas para esse fim;
- nome do objeto: que será definido pelo desenvolvedor;

Objeto

Abstrato Detalhe Relacionamento

Abstrato

NOME	CAPACT	NIVEL	IDENT	DEFAULT	ORGANIZACAO
CodFunc	Texto	1	S	N	Monovalorado
Endereco	Texto	1	N	N	Monovalorado
FuncAtribuicoes	Texto	1	N	N	Conjunto
Nome	Texto	1	N	N	Monovalorado
RG	Número	1	N	N	Monovalorado

Detalhe

Objeto

Tipo do Objeto: Meta Tipo Obj

Tipo da Colônia: Global

Nome do Objeto: Funcionario

Defina Colônia

+ Objeto

+ Atributo

+ Objeto

+ Atributo

Topo

Relacionamento

Abstrato

Classificar

Generalizar

Condição

Detalhe

Instanciar

Especializar

Habitar

Objeto

ATRIBUTO	SUB ATRIBUTO	CAPACT	VALOR
AnalistaResponsavel		Texto	José

AL Instanciar

Edita Valor

Del Valor

Del At. Edit

Fechar

Figura 29: Formulário Objeto

Os atributos de classificação, associados ao objeto, são representados na parte superior do formulário objeto, e os atributos de instanciação, na parte inferior.

A Figura 29 mostra a representação do objeto funcionário, com alguns atributos de classificação e de instanciação. Os detalhes sobre estes atributos serão apresentados nas próximas seções, na descrição do formulário vínculo.

Com o botão +atributo é possível definir e/ou vincular atributos a um objeto, utilizando para isto, o formulário mostrado na Figura 30. Neste formulário é possível verificar todos os atributos que já foram definidos pelo desenvolvedor, cada qual com suas respectivas características.

O formulário 'Define/Vincula Atributos' apresenta uma tabela com duas colunas: 'NOME ATRIBUTO' e 'CARACTERÍSTICA'. Abaixo da tabela, há três botões: 'Define objeto', 'Vincula' e 'Fechar'.

NOME ATRIBUTO	CARACTERÍSTICA
Endereco	Texto
Especializacao	Texto
FNome	Texto
IdFunc	Número
LNome	Texto
Licenciatura/Bach	Texto
LinguaFalada	Texto
Nascimento	Texto
Nome FJ	Texto

Figura 30: Formulário define/vincula atributo

Para definir um novo atributo é preciso definir o nome e a característica deste atributo. A característica de um atributo determina seu tipo de dado. Assim, de acordo com a característica escolhida, o desenvolvedor deverá definir algumas informações relevantes. Por exemplo, se a característica do atributo for número, então o desenvolvedor deverá escolher entre os tipos: *short int*, *unsigned int*, *int*, *unsigned long*, *long*, *long double*, *float* ou *double*.

A Figura 31 mostra um exemplo onde, para o atributo idade, definiu-se a característica número que será do tipo *int* (inteiro).

Neste mesmo formulário é possível definir um atributo com as seguintes características: atributo de atributo, atributo booleano, estrutura de dados, gráfico, imagem, partitura, procedimento, regra, relacionamento, som, tempo, texto, visualização, número e tupla.

Figura 31: Formulário para definir características de atributos

Após a definição de um novo atributo é preciso vincular este atributo a um objeto. Para vincular um atributo a um objeto, algumas informações devem ser definidas no formulário vínculo, conforme mostra a Figura 32. Dentre estas informações destaca-se:

- nome do atributo que será vinculado;
- se o atributo é de classificação (exibido na parte superior do objeto), ou de instanciação (exibido na parte inferior do objeto);
- o tipo de organização (monovalorado, lista, conjunto ou vetor);

Nos casos em que o atributo for monovalorado, a norma (ou comprimento) fica desabilitada com valor *default* igual a 1 (um). Nos casos de atributos multivalorados, as opções disponíveis são: lista, conjunto e vetor. Nestes casos, é preciso definir um valor para a norma, e ainda, se o atributo multivalorado foi organizado em um vetor, é preciso definir um atributo que fornece a norma. Os valores de atributos disponíveis, neste caso, são aqueles que possuem a característica número, cujo tipo de dado é um valor inteiro.

Quando o atributo for vinculado como sendo de classificação, conforme o exemplo da Figura 32, fica habilitado ao desenvolvedor as opções:

- Identificador: especifica se o atributo é identificador do objeto;
- Fixo: o atributo sempre tem ao menos um valor fixo em todas as instâncias do objeto, mesmo que esse valor seja “nulo”;

- *Default*: o atributo possui um valor pré definido que pode ser repassado, ou não, para suas instâncias;
- Atributo indefinido: o valor do atributo pode ser indefinido;
- Valor nulo: o valor do atributo pode ser nulo;
- Nível: indica o nível de classificação do atributo, no tipo de objeto ao qual este atributo está sendo vinculado.

A imagem mostra uma janela de software intitulada "Vínculo". No topo, há um campo de texto rotulado "Nome do atributo" com o valor "LocalDeptoSuperv" e um ícone de lupa. Abaixo, há duas seções de opções de rádio. A primeira seção, rotulada "Classif./Instanc.", contém "Classificação" (selecionada) e "Instanciação". A segunda seção, rotulada "Organização", contém "Monovalorado" (selecionada), "Lista", "Conjunto" e "Valor". À direita dessas seções, há uma série de caixas de seleção: "Identificador", "Atrib. indefinido", "Fixo", "Valor nulo" e "Default". À direita dessas, há um campo rotulado "Nível" com o valor "1". Abaixo das opções de rádio, há um campo rotulado "Nome" com o valor "0". Abaixo disso, há um campo rotulado "Atributo que fornece uma nome" com um ícone de lupa. No canto inferior direito, há dois botões: "Salvar" e "Fechar".

Figura 32: Formulário para vincular atributos a um objeto

O conceito de identificador de objetos, em modelos orientados a objetos, corresponde ao conceito de 'atributo chave', em modelos relacionais. Em SIRIUS, não é permitido que dois ou mais atributos sejam concatenados para criar uma chave. Um atributo sozinho deve ser suficiente para identificar univocamente o objeto dentro de seu contexto (nada impedindo que esse atributo seja da característica tupla).

Várias outras restrições, também relacionadas à definição de um atributo como identificador, são tratadas na ferramenta através da habilitação e desabilitação de alguns componentes. Por exemplo, caso um atributo seja escolhido como identificador, ficam desabilitadas as opções: fixo, *default*, atributo indefinido e valor nulo. A opção nível permanece habilitada, para que o desenvolvedor forneça o nível de classificação do atributo que está sendo vinculado.

Caso o atributo, a ser vinculado a um objeto, seja de instanciação, ficam desabilitadas as opções de identificador e nível.

O botão salvar, do formulário vínculo, armazena na base informações do atributo que foi vinculado ao objeto especificado.

Com relação a remoção de um atributo, tal operação poderá ser efetuada, somente se o atributo que deseja-se remover não estiver vinculado a nenhum outro objeto da base. Na ferramenta, o botão -atributo, do formulário objeto, cumpre esta função.

5.2.2 As abstrações de SIRIUS suportadas pela ferramenta

Na ferramenta E² SIRIUS, todos os construtores semânticos do modelo estão centralizados no objeto. Para cada abstração, existe um módulo que trata suas particularidades, mas ao mesmo tempo, está interligado com outros módulos, uma vez que as abstrações não ocorrem independentes umas das outras.

□ Abstração de Classificação/Instanciação

A abstração de classificação/instanciação é tratada no formulário objeto, conforme discutido na seção anterior. Com os botões classificar e instanciar, desse formulário, o desenvolvedor poderá navegar através da hierarquia de classificação, desde o elemento mais alto da hierarquia até a última instância do objeto. Note-se que, embora esta ferramenta tenha sido desenvolvida para ser um Editor de Esquemas, como SIRIUS trata igualmente todos os objetos, sejam eles tipos ou não, a mesma ferramenta pode ser utilizada exatamente da mesma maneira para editar tanto tipos de objetos, quanto objetos que não são tipos.

Os valores dos atributos agregados ao objeto, ou entre si, são definidos através dos botões 'edita valor'. No formulário objeto existe o botão edita valor, para editar o valor de um atributo de classificação, associado ao objeto corrente, e outro, para editar o valor de um atributo de instanciação.

Conforme verifica-se na Figura 29, o botão edita valor, ao lado do componente que exibe informações dos atributos de classificação, permite ao desenvolvedor definir e/ou alterar um valor *default* para um determinado atributo de classificação.

Para editar o valor de um atributo de instanciação, o desenvolvedor poderá utilizar o botão edita valor, posicionado ao lado do componente que exibe informações dos atributos de instanciação, juntamente com os botões def.valor e def.atr.extra, no formulário objeto. A Figura 33 mostra a definição de um valor, para o atributo de instanciação AnalistaResponsável. Nota-se, nesta Figura, que as opções, nome do subatributo, e atributo que fornece a norma, estão desabilitadas. A primeira opção, fica habilitada

para atributos com característica de tupla, ou atributo de atributo, e a segunda, será habilitada apenas quando o atributo for multivalorado.

Ainda no formulário objeto, o botão `def.valor` permite definir e/ou alterar o valor de atributos fixos, monovalorados, e que pertençam ao primeiro nível da hierarquia de classificação/instanciação, ou seja, estão vinculados ao objeto “pai” da hierarquia. E o botão `def.atr.extra`, utilizado para definir um atributo extra, ou seja, vincular um novo atributo que não tinha sido previsto, ao objeto corrente.

Figura 33: Formulário para editar o valor de um atributo de instanciação

□ Abstração de Composição/Decomposição

Cada instância da abstração de composição determina a existência de um objeto composto, e um conjunto de objetos que são ‘parte’ do objeto composto. Os objetos que são parte de um objeto composto, podem ser agrupados segundo algumas características comuns.

Com base nesta idéia, definiu-se no modelo SIRIUS o conceito de colônia. Ao conjunto de objetos que compõem um objeto composto, segundo um determinado aspecto, dá-se o nome de colônia (Biajiz, 1996).

No editor de esquemas SIRIUS, os objetos devem ser criados em colônias que estejam abertas. Sendo assim, foi definido um formulário, colônias abertas, que cumpre este objetivo, exibindo todas as colônias abertas ao desenvolvedor, e permanece sempre ativo durante a execução da ferramenta.

Uma colônia é identificada por seu tipo e pelo nome do objeto que a restringe, pois cada objeto restringe no máximo uma colônia de um dado tipo.

Através do botão `def.colônia`, no formulário objeto, é possível definir um novo tipo de colônia e o objeto que esta colônia restringe.

NOME DO OBJETO	TIPO COLÔNIA QUE HABITA	DESEJO QUE CONSTRINJA
Analista	Esquema	
Empresa	Global	
Funcionario	Esquema	
Grande	Global	
Media	Global	
Pequena	Global	
Programador	Global	
Supervisor	Global	

Figura 34: Formulário habita colônia

Para abrir uma colônia para, por exemplo, criar um novo objeto nesta colônia, o desenvolvedor deve escolher o botão *constringir*. Este botão irá criar uma colônia, caso ela ainda não exista, a qual permanece aberta, sendo listada no formulário colônias abertas.

Outro botão que trata a abstração de composição/decomposição é o botão *habita*, também do formulário objeto, ilustrado na Figura 34. Este botão exibe um formulário que tem por objetivo, informar ao desenvolvedor, as colônias que os objetos habitam. Este formulário é composto por três colunas com informações sobre: nome do objeto, tipo de colônia que habita e o objeto que constrinje tal colônia.

Esta última informação é muito importante para localizar a colônia exata em que o objeto habita. Isto porque, o modelo permite que exista mais de uma colônia com o mesmo tipo, mas cada uma é constrita por objetos distintos.

❑ Abstração de Agregação/Separação

A abstração de agregação/separação foi implementada na ferramenta através da associação de atributos aos objetos. O tratamento do relacionamento caracterizado pela

agregação de dois ou mais objetos através de um objeto primário, será implementado quando o módulo agregação/separação for tratado.

Considerando a agregação de atributos, verifica-se que tal agregação ocorre de diversas maneiras, associando atributos simples em tuplas, listas, vetores ou conjuntos sem repetição. Estas estruturas modelam diretamente um atributo multivalorado, e foram tratadas na ferramenta através do formulário vínculo, conforme discutido anteriormente.

□ Abstração de Generalização/Especialização

No editor SIRIUS foram criados dois formulários para tratar a abstração de generalização/especialização: o formulário generalização ampla; e, o formulário especialização restrita.

Generalização Ampla

Objeto Generalização

Tipo do Objeto: Meta Tipo Obj Tipo da Colônia: Global

Nome do Objeto: Empresa

Opção de Especialização: Generalização Restrita Entero: NroFunc

Restrições:

Subordinação: ☒ Direção ☐ Subordinação

Participação: ☒ Total ☐ Parcial

Objetos Específicos:

OBJETO ESPECÍFICO	PREDICADO	INICIALIZAÇÃO
Pequena	1<=NroFunc<=5	
Media	5<=NroFunc<=50	
Grande	50<NroFunc	

Fundo

Figura 35: Formulário generalização ampla

O formulário generalização ampla, ilustrado na Figura 35, é um formulário de consulta, onde o desenvolvedor define valores para os campos nome do objeto, e critério, para que as informações de especialização sejam recuperadas da base.

No componente opção de especialização, o desenvolvedor poderá escolher a maneira como o tipo de objeto foi especializado: através da generalização restrita ou categorização. Embora SIRIUS considere a abstração de categorização como uma especialização da abstração de generalização, neste trabalho estaremos considerando a abstração de generalização tal como é tradicionalmente definida, a qual em SIRIUS é denominada Abstração de Generalização Restrita.

É importante que seja definido um valor para o critério, uma vez que um mesmo objeto pode ser especializado à partir de critérios diferentes, resultando em mais de uma hierarquia de especialização. Assim como os atributos identificadores, em SIRIUS, um critério deve ser representado por apenas um atributo. Porém, nada impede que esse atributo tenha a característica tupla.

Tendo definido estas informações relevantes, serão recuperadas da base:

- restrições de sobreposição (disjunção ou sobreposição);
- restrições de participação (total ou parcial);
- lista de objetos específicos com informações sobre o nome do objeto específico, o predicado, e um valor para a inicialização do objeto.

No formulário especialização restrita, ilustrado na Figura 36, o desenvolvedor poderá definir os tipos de objetos específicos de um determinado tipo genérico. Para isto, é preciso definir as restrições para a especialização, o critério que define a especialização, e informações sobre o tipo de objeto específico. Dentre estas informações destaca-se o tipo de objeto, tipo de colônia e nome do objeto.

Na Figura 36, o tipo genérico empresa foi especializado no tipo específico pequena, de acordo com o atributo critério `NroFunc` (número de funcionários). Para que, um objeto pertença ao tipo específico pequena, é preciso avaliar o predicado $1 \leq \text{NroFunc} \leq 5$. Caso o resultado for verdadeiro, pode-se afirmar que o objeto avaliado pertence ao subtipo pequena.

Na próxima seção serão apresentadas as extensões realizadas na ferramenta E² SIRIUS, e o novo módulo de mapeamento acoplado a essa ferramenta. Maiores detalhes sobre os formulários descritos nesta seção podem ser encontrados em (Araújo, 1999).

Especialização Restrita

Objeto generalização

Tipo do Objeto: Meta Tipo Obj Tipo da Coluna: Global

Nome do Objeto: Empresa

Restrições

Superordenação: ☒ Disjunção ☐ Subordinação

Participação: ☒ Total ☐ Parcial

Critério

NroFunc

Objeto específico

Tipo do Objeto: Empresa Tipo da Coluna: Global

Nome do Objeto: Pequena

Predicador

1<=NroFunc<=5

Inicialização

= Obj. Espec.

Salvar

Fechar

Figura 36: Formulário especialização restrita

5.3 Editor SIRIUS Versão 1.1

O módulo de mapeamento de ocorrências da abstração de generalização (OAG), implementado em E² SIRIUS versão 1.1, utiliza informações da estrutura e das propriedades da OAG que deseja-se mapear, conforme mostra a Figura 37. Tais informações podem ser armazenadas na base de dados para futuras consultas.

Com base nestas informações, o módulo de mapeamento utiliza a estrutura da árvore de decisão, que também fica armazenada na base, para encontrar a opção de mapeamento apropriada. A partir disso, o *script* SQL é gerado, com as restrições de domínio, chave primária e chave estrangeira, a fim de que, tal *script* seja executado em uma outra base de dados relacional.

Figura37: Módulo de Mapeamento

A estrutura de dados adequada, para a implementação deste módulo, foi definida a partir da própria estrutura da árvore de decisão, apresentada anteriormente. Conforme observou-se no Capítulo 4, existe um nó que é fonte (raiz) dessa árvore, que possui como valores as dez opções de mapeamento definidas, e ainda, duas subárvores (esquerda e direita). Os outros nós têm grau de entrada igual a um, e não têm ciclos, podendo-se afirmar que entre dois nós existe um único caminho. Observa-se ainda, que trata-se de uma árvore binária desbalanceada, onde cada nó da árvore de decisão, possui zero, um, ou dois filhos. Um nó que possui as subárvores, esquerda e direita, vazias, é chamado de nó folha. Na árvore de decisão, cada nó folha possui uma única opção de mapeamento, devido ao caminho percorrido em função das escolhas do desenvolvedor. Nota-se no entanto, que há vários nós (folhas ou não) na árvore, que possuem como resultado as mesmas opções de mapeamento. É possível diferenciá-las através da localização dos nós na árvore, com relação a raiz, uma vez que cada nó, em uma árvore binária, é determinado unicamente em função de sua localização (Collins, 1992).

Baseando-se nestas características, e somando-se ao fato de que, a árvore de decisão possui uma estrutura fixa (nunca muda), optou-se por implementá-la como uma matriz de vetores estática.

5.3.1 Plataforma de E² SIRIUS versão 1.1

O protótipo da ferramenta foi implementada em plataforma Windows 95 e utilizou-se a ferramenta C++Builder versão 3.0 Cliente/Servidor da Borland (BORLAND, 1999), para o desenvolvimento da interface. Esta versão do C++Builder permite acessar vários SGBD's como: Oracle, Sybase, Informix, DB2, SQLServer e Interbase (INTERBASE, 1999). Este último fora escolhido para o desenvolvimento de E² SIRIUS e mantido na nova versão da ferramenta, estando prevista a sua extensão para outros SGBD's.

5.3.2 Modificações realizadas na nova versão de E² SIRIUS

Para a automatização do processo de mapeamento, alguns formulários foram modificados, e outros, definidos.

Generalização Ampla

Objeto generalização

Tipo do Objeto: Meta Tipo Obj Tipo da Colônia: Global

Nome do Objeto: Empresa

Restrições

Sobreposição: ☒ Disjunção ☐ Sobreposição

Particionamento: ☒ Total ☐ Parcial

Opção de Especialização: Generalização Restrita

Critério: NroFunc

Objetos Especializados

OBJETO ESPECIALIZADO	PREDICADO	INICIALIZAÇÃO
Pequena	1<=NroFunc<=5	
Media	5<=NroFunc<=50	
Grande	50<NroFunc	

Propriedades de Mapeamento

Definir

Visualizar

Fechar

Figura 38: Novo formulário Generalização Ampla

O formulário que trata a generalização ampla, descrito na seção anterior, foi modificado conforme mostra a Figura 38, com a inclusão de três novos botões: topo, definir e visualizar.

O botão topo permite que o desenvolvedor alcance o topo de uma hierarquia, seja qual for o nível atual do objeto. Para a definição deste botão, considerou-se que a hierarquia de generalização pode ter vários níveis, e que, um mesmo subtipo pode ter sido especializado à partir de supertipos diferentes, desde que possuam o mesmo tipo natural.

Com o botão definir, o desenvolvedor poderá selecionar as propriedades relevantes, de uma OAG particular, para o mapeamento. O botão visualizar permite que as propriedades definidas para uma determinada OAG, e já armazenadas na base, possam ser consultadas.

5.3.3 Os novos formulários de E² SIRIUS para a versão 1.1

Conforme dito anteriormente, para a escolha de uma maneira particular de se implementar uma OAG é importante que sejam definidos alguns fatores, próprios da aplicação, que estão representados através das propriedades de mapeamento, e de condições lógicas, que garantem a consistência dos dados, representados pelas restrições de participação e sobreposição.

Na ferramenta, a definição de tais restrições pode ser feita no formulário especialização restrita, e a definição das propriedades, no formulário definição de propriedades de mapeamento, mostrado na Figura 39. Este último formulário é exibido através do botão definir, do formulário generalização ampla (Figura 38).

Conforme o desenvolvedor define as propriedades referentes ao mapeamento de sua modelagem, as opções vão sendo selecionadas, de acordo com a árvore de decisão, e exibidas no componente opções de mapeamento (Figura 39). É importante ressaltar que, no momento em que o formulário é exibido, antes mesmo do usuário definir qualquer propriedade, algumas opções de mapeamento já foram excluídas em decorrência da definição das restrições de sobreposição (sobreposição/disjunção) e participação (total/parcial) definidas no formulário especialização restrita.

Para a OAG empresa, mostrada na Figura 38, as restrições definidas foram de disjunção, e de participação total. Logo, as opções de mapeamento possíveis, listadas no componente opções de mapeamento, no momento em que o formulário definição de propriedades de mapeamento é exibido, são as opções 1, 2, 3, 4, 5, 6, 7 e 9, conforme mostra a Figura 39.

Figura 39: Formulário Definição de Propriedades de Mapeamento com a Primeira Propriedade Ativa

A primeira propriedade a ser definida pelo usuário é a propriedade que trata do acesso ao conjunto de atributos critério, representada pelo componente acesso ao critério, que pode ser freqüente ou nenhum. Nota-se na Figura 40 que, com a escolha do acesso ao critério como freqüente, para a OAG empresa, as opções de mapeamento possíveis, listadas no componente opções de mapeamento, alteraram para as opções 2, 3, 5, 6, 7 e 8.

O ideal é que o desenvolvedor obtenha uma única opção de mapeamento para a OAG que deseja mapear. Para isto, será preciso definir mais algumas propriedades. Com o botão +Propriedades, o desenvolvedor poderá definir outras propriedades relevantes, conforme foi definido na árvore de decisão. Assim, com o clique no botão +Propriedades, o formulário definição de propriedades de mapeamento exibe o componente atributos no supertipo, conforme mostra a Figura 41. Neste componente é possível escolher entre muitos atributos no supertipo, ou poucos/nenhum.

Figura 40: Definição da Primeira Propriedade de Mapeamento

Figura 41: Definição da Segunda Propriedade de Mapeamento

A terceira propriedade que estará ativa em E²SIRIUS versão 1.1, mostrada na Figura 42, trata do acesso a informações conjuntas de vários subtipos, onde é possível escolher entre freqüente ou nenhum, no componente acesso a informações conjuntas de vários subtipos.

Nota-se que algumas propriedades não excluem nenhuma opção de mapeamento. Isto pode ocorrer, por exemplo, quando uma propriedade está sendo verificada para opções que pertencem a um mesmo grupo de mapeamento, como é o caso das opções 2 e 3 que pertencem ao grupo $G + \{E_i\}$, e nenhuma delas é excluída com a avaliação da propriedade que trata o acesso a informações conjuntas de vários subtipos.

A próxima propriedade, trata dos atributos que compõem o critério. Nota-se que esta propriedade estará ativa, apenas se o desenvolvedor selecionou freqüente, no componente acesso ao critério. Caso contrário, a opção adequada ao mapeamento será sempre definida pelas informações selecionadas nas três primeiras propriedades, representadas nos componentes: acesso ao critério, atributos no supertipo e acesso a informações conjuntas de vários subtipos. Esta observação pode ser verificada na árvore de decisão.

Figura 42: Definição da Terceira Propriedade de Mapeamento

Na Figura 43, a informação selecionada no componente multivalorado foi nenhum, significando que nenhum dos atributos que compõem o critério são multivalorados. Esta escolha resulta em uma única opção de mapeamento, que neste caso é a opção 2. Assim, a OAG empresa, que pode ser especializada nos subtipos pequena, média e grande, será mapeada de acordo com a opção de mapeamento 2, em decorrência das propriedades definidas, conforme foi mostrado nas figuras 39 a 43.

Figura 43: Definição da Quarta Propriedade de Mapeamento

Caso a escolha no componente multivalorado fosse nenhum, e ainda existisse mais de uma opção, no componente opções de mapeamento, uma última propriedade seria exibida. Esta propriedade pode ser verificada na Figura 44, onde as informações neste formulário, foram definidas para uma OAG, diferente de empresa. Neste caso, a propriedade do componente monovalorado deve ser preenchida para que se obtenha uma única opção de mapeamento. Neste último componente, duas situações foram representadas. A primeira, trata a situação em que o critério é representado por um conjunto de atributos monovalorados, e na segunda, o critério é representado por uma expressão sobre um ou mais atributos.

A janela "Definição de propriedades para o mapeamento" apresenta as seguintes opções:

- Acesso ao critério:**
 - ☒ frequente
 - ☐ nenhum
- Atributos no supertipo:**
 - ☐ muitos
 - ☒ poucos ou nenhum
- Acesso a informações conjuntas de vários subtipos:**
 - ☐ frequente
 - ☒ nenhum
- Atributos que compõem o critério:**
 - Multivalorado:**
 - ☐ pelo menos um
 - ☒ nenhum
 - Monovalorado:**
 - ☐ conjunto de atributos
 - ☒ expressão sobre um ou mais atributos

Botões de ação:

- Propriedades:** Botão com ícone de cruz verde.
- SQL:** Botão com ícone de documento e seta vermelha.
- Salvar:** Botão com ícone de disquete.
- Cancelar:** Botão com ícone de lixeira.
- Fechar:** Botão com ícone de porta.

À direita, há uma seção "Opções Mapeamento:" com uma barra deslizante vertical cujo valor atual é 5.

Figura 44: Definição da Quinta Propriedade de Mapeamento

No formulário definição de propriedades de mapeamento, o desenvolvedor poderá ainda, salvar as propriedades definidas, ou cancelar, utilizando para isto, os botões salvar ou cancelar, respectivamente. E finalmente, o botão SQL, exibirá o formulário Código SQL, mostrado na Figura 45.

Considerando novamente a OAG empresa que, de acordo com as informações definidas pelo desenvolvedor, a opção 2 é a opção mais adequada para o mapeamento, conforme mostra a Figura 43. Com o botão SQL o desenvolvedor terá acesso a um *script* SQL, com restrições de domínio, chave primária e integridade referencial entre as tabelas que mapeiam o supertipo e os subtipos, conforme mostra a Figura 45. Com o *script* resultante é possível representar esta OAG em um gerenciador relacional.

Todas as informações que o usuário fornece, com relação a cada ocorrência, são armazenadas na base de dados, de maneira que a geração dos *scripts* de mapeamento podem ser solicitados logo após a definição das informações de cada OAG, ou posteriormente, pela recuperação de tais informações da base de dados. Caso ocorra alguma alteração no projeto, como por exemplo, a mudança no nome de um objeto ou de atributos, elas podem ser repassadas ao esquema, e um novo mapeamento pode ser requerido, sem que o usuário precise definir novamente, as informações relevantes, para o mapeamento da OAG em questão.

Figura 45: Código SQL para a OAG empresa

5.4 Considerações Finais

A implementação do módulo de mapeamento proporcionou um ambiente gráfico simples, seguindo a interface da ferramenta Editor de Esquemas SIRIUS, E² SIRIUS (Araujo, 1998a), e teve como propósito validar a árvore de decisão, definida no Capítulo 4, além de expandir a ferramenta E² SIRIUS com o tratamento do módulo referente a abstração de generalização.

O Editor foi utilizado para construir todos os exemplos mostrados no Capítulo 4, e foram solicitados os mapeamentos correspondentes, validando a correção da execução de E² SIRIUS versão 1.1, para todas as opções de mapeamento desenvolvidas neste trabalho.

A inclusão do processo de mapeamento como um módulo do E² SIRIUS proporcionou diversas vantagens. Entre estas vantagens, pode-se citar o fato de que, o desenvolvedor pode utilizar os demais recursos do Editor juntamente com o mapeamento, sem necessidade de esforço adicional. E ainda, quando os outros recursos previstos para o Editor estiverem disponíveis, tal como a disponibilidade de suporte a comandos de macro, este módulo também poderá aproveitá-las igualmente. Outra vantagem é a manutenção da consistência das informações. Um exemplo disso, citado anteriormente, é o fato que quando um atributo ou objeto tem seu nome alterado, essa alteração é global e consistentemente tratada em todos os módulos.

Capítulo 6 – Conclusão

6.1 Considerações Iniciais

Este trabalho teve como objetivos propor um método ao processo de mapeamento da abstração de generalização para o Modelo Relacional e a construção do protótipo de uma ferramenta que apóia o método proposto, validando o desenvolvimento teórico das técnicas de mapeamento que foram definidas.

Para isto, foram estudados vários modelos de dados, tanto os modelos semânticos, como os orientados a objetos, com o intuito de comparar os construtores semânticos destes modelos, sempre com enfoque na abstração de generalização. Após o estudo dos modelos de dados, encontrados na literatura pesquisada, foi possível verificar um conjunto de características que especificam, com alto grau de refinamento, a estrutura básica de cada ocorrência da abstração de generalização. Foram estudadas também, as diversas técnicas de mapeamento desta abstração, para o Modelo Relacional, propostas por vários autores.

A partir da constatação de que, o conjunto de construtores semânticos de um modelo de dados proporcionam uma capacidade de representação maior, de situações do mundo real, quando comparados com os construtores do Modelo Relacional, iniciou-se o desenvolvimento de dez opções de mapeamento. O desenvolvimento destas opções teve como objetivos tornar o processo de mapeamento abrangente, para que tais opções pudessem ser utilizadas em muitos modelos de dados conceituais, garantindo o máximo possível, a preservação das propriedades semânticas das modelagens originais, e a manutenção da consistência da informação envolvida. Estas opções de mapeamento propostas estendem as opções encontradas na literatura, uma vez que tratam amplamente as características que definem uma ocorrência da abstração de generalização, e permitem pela primeira vez, tanto quanto é de nosso conhecimento, o

tratamento completamente automatizado de uma ocorrência da abstração de generalização, desde a representação em um modelo orientado a objetos, até a representação relacional.

Dentre o conjunto de características que definem uma OAG pode-se destacar o conjunto de restrições (participação e sobreposição), um conjunto de propriedades relevantes para o processo de mapeamento, e um conjunto de atributos denominado critério. À partir da avaliação destas características, em vários exemplos práticos, procurou-se evidenciar as vantagens e desvantagens do uso de cada uma das dez opções de mapeamento, desenvolvidas como parte da técnica de mapeamento, a fim de auxiliar o desenvolvedor na escolha pela opção apropriada.

A proposta de uma árvore de decisão, construída de acordo com o conjunto de propriedades e restrições definidas, teve como objetivo, servir de base para a construção do protótipo de uma ferramenta, a qual tornou possível o mapeamento automático das ocorrências da abstração de generalização. Para isto, estendeu-se o Editor de Esquemas E²SIRIUS, para o processo de mapeamento, incluindo-se recursos que permitem ao projetista de aplicações, a definição de todos os dados que descrevem cada ocorrência da abstração de generalização, e ainda, a inclusão de um novo módulo, que efetivamente executa o processo de mapeamento.

6.2 Contribuições deste Trabalho

O método para mapeamento das ocorrências da abstração de generalização foi desenvolvido com o objetivo de proporcionar ao desenvolvedor, um conjunto completo de opções de mapeamento, esgotando-se todas as opções possíveis para os casos normais de mapeamento.

Para o desenvolvimento de tal método definiu-se um conjunto de características básicas, que devem ser identificadas em uma OAG. Tais características orientam o desenvolvedor na representação de todas as informações relevantes, da abstração de generalização, durante as atividades de modelagem.

Dentre estas características básicas, uma delas foi importante para a definição das dez opções de mapeamento que compõem o método proposto: o conjunto de atributos denominado critério, que definem a especialização de um tipo genérico em tipos específicos. Mais propriamente, na maneira em que o critério é representado, que pode ser por:

- um conjunto de atributos monovalorados;
- um conjunto de atributos onde ao menos um deles é multivalorado;
- uma expressão sobre um ou mais atributos.

Considerando estas três maneiras de representar o critério, foi possível estender e unificar os trabalhos existentes na literatura, com a definição de dez opções de mapeamento, sendo que cinco destas opções, tanto quanto é de nosso conhecimento, não haviam sido tratadas em nenhum outro trabalho anterior. Em nossa proposta, o critério pode ser tratado: na tabela que mapeia o tipo genérico; nas tabelas que mapeiam cada tipo específico; ou em uma tabela separada, definida exclusivamente para o tratamento do critério.

É importante destacar que, com estas opções de mapeamento é possível tratar não apenas a representação de OAGs cujos critérios são baseados em atributos estáticos dos objetos especializados, mas também, de atributos vinculados a eventos que provêem a dinâmica de uma base de dados. Assim, as técnicas desenvolvidas podem ser aplicadas também às bases de dados temporais e históricas, bem como às bases de dados ativas e reativas. Algumas das opções de mapeamento permitem a evolução dos objetos especializados, incluindo a navegação baseada em eventos da estrutura de subtipos gerada pelas hierarquias da abstração de generalização.

A avaliação das opções de mapeamento foi realizada considerando-se alguns aspectos da estrutura conceitual, da abstração de generalização, representadas pelas restrições de sobreposição e participação, e por outros fatores, representados por propriedades, que fazem parte das características básicas de uma OAG, e que consideram o volume dos dados a serem armazenados e o tipo de acesso aos dados. Este resultado é muito útil para que o desenvolvedor possa avaliar, o problema que deseja mapear, com relação às dez opções de mapeamento.

Para decidir entre uma, das dez opções, o desenvolvedor deverá utilizar a árvore de decisão, que resultará em uma única opção, de acordo com as características particulares de cada OAG definidas em cada nível da árvore. Esta árvore de decisão foi construída considerando-se as restrições, e as propriedades relevantes em cada OAG, utilizadas para a avaliação das opções de mapeamento.

A implementação do módulo referente a abstração de generalização, deu origem ao protótipo da ferramenta E²SIRIUS versão 1.1, proporcionando a automatização do processo de mapeamento das OAGs, validando as propostas deste trabalho, e integrando-as com uma ferramenta de edição de esquemas, que suporta todas as características estruturais que devem ser verificadas para o uso das opções de mapeamento.

Dessa maneira, podem ser consideradas contribuições inéditas deste trabalho:

1. A estruturação detalhada de uma OAG como sendo constituída pelos seguintes elementos:
 - a) um tipo (único) genérico de objeto;
 - b) um conjunto de tipos de objetos especializados;

- c) um conjunto de atributos do tipo genérico;
 - d) um conjunto de atributos dos tipos especializados;
 - e) um conjunto de restrições para a generalização;
 - f) um conjunto de propriedades para a generalização;
 - g) um predicado para cada especialização;
 - h) um conjunto de regras de inicialização para cada tipo de objeto especializado;
 - i) um conjunto de atributos critério.
2. O desenvolvimento de cinco novas opções de mapeamento de OAGs para o Modelo Relacional, inéditas na literatura.
 3. A caracterização do conjunto completo de dez opções de mapeamento, capaz de cobrir todas as necessidades de mapeamento de aplicações tradicionais, bases históricas e temporais, e bases de dados ativas e reativas.
 4. A elaboração de uma árvore de decisão, a qual permite a escolha algorítmica de uma das dez opções de mapeamento, resultando na opção mais apropriada para cada OAG.
 5. O desenvolvimento de um processo de mapeamento completamente automático, para o mapeamento de OAGs, em uma modelagem de dados efetuada em um modelo orientado a objetos, para uma modelagem efetuada no Modelo Relacional.

Além disso, a validade e a aplicabilidade prática dessas contribuições foi demonstrada através de uma ferramenta, implementada como um módulo de mapeamento automático, associado a um Editor de Esquemas.

6.3 Sugestões para Futuras Pesquisas

Com relação ao método, proposto neste trabalho, procurou-se tratar amplamente, os aspectos estáticos do processo de mapeamento das ocorrências da abstração de generalização. O próximo passo é tratar os aspectos dinâmicos, definindo regras para controlar a consistência da hierarquia de generalização, para entre outros, permitir as possíveis formas de evolução de objetos na hierarquia, baseado em eventos. Muitos desses eventos podem ser associados a operações padrão, de maneira que processos inteiros, normalmente existentes em aplicações típicas, possam ter suporte automatizado por parte do gerenciador de dados.

Um outro trabalho importante que poderia ser desenvolvido, é a formalização das dez opções de mapeamento, que foram definidas, em uma álgebra relacional semelhante a utilizada em (Biajiz, 1996a).

Uma outra linha de pesquisa interessante corresponde ao estudo do mapeamento de outras abstrações, suportadas em modelos orientados a objetos, para o Modelo Relacional. Considerando-se apenas o modelo SIRIUS, verifica-se que um trabalho, equivalente a este, deve ser desenvolvido para as abstrações de composição, agregação e classificação, suportadas pelo modelo. Além disso, deve-se avaliar, como os diversos processos de mapeamento podem ser integrados, com o tratamento da interação entre mais de uma abstração.

Em particular, no estudo do inter-relacionamento das abstrações de generalização e de composição. Enquanto a abstração de generalização fornece suporte para o conceito de herança, através da recuperação de atributos entre os objetos, baseando-se em seus tipos mais genéricos ou mais específicos, a abstração de composição fornece o suporte conceitual para a criação e remoção de objetos, incluindo-se conceitos como tempo de vida, local, e domínio de validade do identificador dos objetos. Assim, um estudo completo, integrando ambas as abstrações, poderá fornecer um conjunto completo de todas as operações que envolvem o identificador dos objetos.

Finalmente, a árvore de decisão proposta neste trabalho poderia ser generalizada, com a adoção de outras estruturas de dados, para se chegar a uma opção de mapeamento adequada. Por exemplo, pode-se explorar recursos das máquinas de estados finitos para modelar as opções de mapeamento. O estado final seria a representação das opções de mapeamento, e os possíveis caminhos, previstos na máquina de estados, deveriam corresponder as escolhas de utilização ou não, das propriedades e restrições definidas para cada OAG.

Referências Bibliográficas

- ARAUJO, M.R.B. *Representação de Construtores Semânticos em SIRIUS e Suporte através de um Editor de Esquemas*. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação – ICMC-USP, São Carlos – SP, Junho, 1998.
- BAROODY, A.J.; DEWITT, D.J. An Object-Oriented Approach to Database System Implementation - *ACM Transactions on Database Systems*, vol.6, nº 4, p.577-601, December 1981.
- BATINI, C.; CERI, S.; NAVATHE, S.B. *Conceptual Database Design: An Entity-Relationship Approach*. Benjamin/Cummings, 1992.
- BATINI, C.; CERI, S. Database Design: Methodologies, Tools and Environments, *ACM SIGMOD Conference*, Austin – Texas, p.148-150, May 1985.
- BERTINO, E.; LORENZO, M. Object-Oriented Database Systems. *Internacional Computer Science Series*, Addison-Wesley, 1.993.
- BIAJIZ, M. *Representação de Modelos de Dados Orientados a Objetos através de Parametrizações de Abstrações*. Tese (Doutorado), IFSC - USP, São Carlos - SP, Setembro, 1996.
- BIAJIZ, M.; TRAINA, C. Jr. "SIRIUS - Modelo de Dados Orientado a Objetos e Baseado em Abstrações de Dados. *Anais do 11º. Simpósio Brasileiro de Banco de Dados*, São Carlos - S.P., 1996.
- BOOCH, G. *Object-Oriented Analysis and Design with Applications*. Benjamin/Cummings, 1994.
- BORLAND HOME, Inprise Corp. (online). (04/01/1999). Disponível na Internet: <<http://www.borland.com/bcppbuilder>>.
- CERI S.; FRATERALI P. *Database Applications with Objects and Rules*. Addison-Wesley, 1997.
- CHEN, P.P. The Entity-Relationship Model - Toward a Unified View of Data. *ACM Transactions on Database Systems*, vol.1, nº1, p. 9-36, March 1976.
- CHEN P.P. Entity-Relationship Approach to Systems Analysis and Design. *ER Conference, North-Holland*, 1980. p. 13-14.
- CODD, E.F. Extending the database relational model to capture more meaning. *ACM Transaction Database System*, vol.4, nº.4, p. 397-434, 1979.

- COLEMAN, D.; ARNOLD, P.; BODOFF, S.; DOLLIN, C.; GILCHIRST, H.; HAYES, F.; JEREMAES, P. *Object-oriented development : The Fusion Method*, New Jersey, USA, Prentice Hall, 313p., 1994.
- COLLINS, W.J. *Data Structures: An Object-Oriented Approach*. Addison-Wesley, 1992.
- EICK, C. A Methodology for the Design and Transformation of Conceptual Schemas. *VLDB*, 1991. p.25-34.
- ELMASRI, R.; NAVATHE, S.B. *Fundamentals of Database Systems*, Addison-Wesley, 1994, 2ª Edição.
- OBJECT-ORIENTATION FAQ. (online). (20/03/1998). Disponível na Internet: <<http://attila.ce.unipr.it/docs/OOP/OOFAQ/oo-faq-S8.12.1.4.html>>.
- HAREL, D. Statecharts: a visual formalism to complex systems. *Science of Computer Programming*, 1987. nº 8, p.231-274.
- HAMMER M.; MCLEOD D. Database Description with SDM: A Semantic Database Model. *ACM Transaction on Database Systems*, vol. 6, nº 3, p. 352-386, September 1981.
- HULL R.; KING, R. Semantic Database Modeling: Survey, Application and Research Issues. *ACM Computing Surveys*, vol.19, nº 3, p.201-260, September 1987.
- INTERBASE SERVER VERSION 4.2.1. Borland International, Inc. (online). (10/01/1999). Disponível na Internet: <<http://www.interbase.com>>.
- ITASCA (SMBDOO), Applications Integration Framework. (online). (20/03/1998). Disponível na Internet: <<http://w3.iprolink.ch/ibexcom/products.html>>.
- JACKSON, R.B.; EMBLEY, D.W.; WOODFIELD, S.N. Developing Formal Object-Oriented Requirements Specifications: A Model, Tool and Technique. *Information Systems*, vol.20, nº 4, p.273-289, 1995.
- JENSEN, C.S.; SNODGRASS, R.T. Temporal Specialization and Generalization. *IEEE Transactions on Knowledge and Data Engineering*, vol.6, nº 6, p.954-974, 1994.
- JENSEN, C.S.; SOO, M.D.; SNODGRASS, R.T. Unification of Temporal Data Models. *International Conference on Data Engineering (ICDE)*, p.262-271, 1993.
- KIM, W.; LOCHOVSKY, F. *Object Oriented Concepts, Databases and Applications*, Addison-Wesley, 1989.
- KIM, W. *Introduction to Object Oriented Database*, Cambridge MA: The MIT Press, 1990.
- KIM, W.; SUH, Y.; WHINSTON, A.B. An IBIS and Object-Oriented Approach to Scientific Research Data Management, *J.Systems Software*, vol.23, p.183-197, 1993.

- MAIER, D. The Theory of Relational Databases. *Oregon Graduate Center, Computer Science Press*, 1983.
- MONTELEONE, G. Applications Case Studies - IDEA Project. (online) (15/03/1998). Disponível na Internet: <http://www.txt.it.5858/case_studies/case_studies.html>.
- NAVATHE, S.B. Evolution of Data Modeling for Databases, *Communications of the ACM*, vol.35, nº 9, p.112-123, September 1992.
- ORACLE Application Server, Oracle Corporation. (online). (12/03/1998). Disponível na Internet: <<http://www.oracle.com>>.
- OZSOUGLU, G.; SNODGRASS, R.T. Temporal and Real-Time Databases: A Survey. *IEEE Transaction on Knowledge and Data Engineering*, vol.7, nº 4, August 1995.
- RUMBAUGH, J. et. al. *Object-Oriented Modeling and Design*, Prentice Hall, 1991.
- SAKAI, H. Entity-Relationship Approach to Conceptual Schema Design. *SIGMOD*, California, May 1980. p.1-8.
- SAKAI, H. Entity-Relationship approach to Logical Database Design. *ER'93*, Anaheim, California – USA, p.155-188, 1983.
- SILBERSCHATZ, A.; Korth, H.F.; Sudarshan, S. *Database System Concepts*. McGraw-Hill, 1997.
- SMITH, J.; SMITH, D. Database Abstractions: Agregation and Generalization, *TODS*, vol.2, nº2, June 1977.
- SPIEGLER, I.; WIDDER, D. Physical Database Design: A Decision Support Model. *Database Advances*, p.5-11, August 1993.
- TANSEL A.U. et. al. *Temporal Databases: Theory, Design and Implementation*. Benjamin/Cummings, 1993.
- TEORY, T. J.; YANG, D.; FRY, J.P. A Logical Design Methodology for Relational Databases using the Extended Entity-Relationship Model. *ACM Computing Surveys*, vol.18, nº2, p.197-222, June 1986.
- TRAINA Jr., C. *GEO – Um Sistema de Gerenciamento de Bases de Dados Orientado a Objetos – Estado Atual de Desenvolvimento e Implementação*. *Anais do 6º Simpósio Brasileiro de Banco de Dados*, Manaus – AM, 1991.
- TRAINA JR., C.; SLAETS, J.F.W. Um Modelo de Representação de Objetos. *Notas do ICMC-USP*, nº 104, São Carlos – SP, Fevereiro 1992.
- TRAINA JR., C.; TRAINA, A.J.M.; Biajiz, M. O Papel da Abstração de Instanciação em um Meta-Modelo de Abstrações para BDOO. *Anais do IX Simpósio Brasileiro de Banco de Dados*, São Carlos -S.P., p.173-187, 1994.

- TRAINA JR., C.; Traina, A.J.M. Algoritmo para Criação e Gerenciamento de Identificadores de Objetos em um Gerenciador de Objetos. JAIO – 25ª *Jornadas Argentinas de Informática e Investigación Operativa*, p.2.41-2.52, Septiembre 1996.
- Unified Modeling Language, Rational Software Corporation, Notation Guide Version 1.0. (online). (12/02/1998). Disponível na Internet: <<http://www.rational.com/UML>>.
- ZAND, M.; COLLINS, V.; CAVINESS, D. A Survey of Current Object-Oriented Databases, *Database Advances*, vol.26, nº 34, p.14-29, February 1995.
- ZHAO, L.; ROBERTS, S.A. An Object-Oriented Data Model for Database Modelling Implementation and Access - *The Computer Journal*, vol.31, nº 2, p.116-124, 1988.