**UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA DE SÃO CARLOS
DEPARTAMENTO DE FÍSICA E INFORMÁTICA**

**"Representação de Modelos de Dados Orientados a Objetos através de
Parametrização de Abstrações".**

MAURO BIAJIZ

Tese apresentada ao Instituto de Física de São Carlos, Universidade de São Paulo, para obtenção do título de Doutor em Ciências "Física Aplicada-opção Física Computacional".

Orientador: Prof.Dr. Caetano Traina Junior

SÃO CARLOS

SÃO PAULO

1996

Biajiz, Mauro

Representação de Modelos de Dados Orientados a Objetos
através de Parametrização de Abstrações / Mauro Biajiz, São Carlos,
1996.

150p.

Tese(Doutorado) Instituto de Física de São Carlos, 1996.

1.Bases de dados orientadas a objetos. 2.Modelos de dados
orientados a objetos. 3.Abstrações de dados. I.Título

IFSC UNIVERSIDADE
DE SÃO PAULO
Instituto de Física de São Carlos

Av. Dr. Carlos Botelho, 1465
CEP 13560-250 - São Carlos - SP
Brasil

Fone (016) 272-6222
Fax (016) 272-2218

MEMBROS DA COMISSÃO JULGADORA DA TESE DE DOUTORADO DE **MAURO BIAJIZ** APRESENTADA AO INSTITUTO DE FÍSICA DE SÃO CARLOS, UNIVERSIDADE DE SÃO PAULO, EM 03/09/1996.

COMISSÃO JULGADORA:

Prof. Dr. Caetano Traina Junior/ICMSC-USP

Prof. Dr. Luciano da Fontoura Costa/IFSC-USP

Prof. Dr. Jan Frans Willem Slaets/IFSC-USP

Prof. Dr. Astério Kiyoshi Tanaka/IME-RJ

Prof. Dr. Fernando Fonseca de Souza/UFPe

***“Aos meus pais e
à Célia com
muito amor”***

AGRADECIMENTOS

Ao Prof. Dr. Caetano Traina Júnior pela incansável e valiosa orientação e particularmente pela sua confiança e amizade.

Aos amigos e familiares que direta ou indiretamente contribuíram emocionalmente e tecnicamente para a realização desse trabalho.

Aos componentes do Grupo de Pesquisa em Banco de Dados do ICMSC-USP, que proporcionam um ambiente digno e propício para o desenvolvimento de pesquisas.

À Wladerez A.. G. Caiado pela extrema competência e dedicação com que exerce seu trabalho junto à Secretaria de Pós-Graduação do IFSC-USP.

Aos colegas do *Departamento de Computação - UFSCar* por terem apoiado e icentivado esse trabalho.

Às professoras Dras. Agma J. M. Traina e Dra. Marina T.P. Vieira pelo apoio e amizade.

Ao *Instituto de Física de São Carlos, IFSC- USP* e ao *ICMSC-USP*, pelo apoio institucional.

Aos amigos de moradia, Ricardo, Tchelo e Marcus, que compartilharam todos esses momentos.

SUMÁRIO

LISTA DE FIGURAS	vi
-------------------------------	----

RESUMO	viii
---------------------	------

ABSTRACT	ix
-----------------------	----

Introdução

1.1.Introdução	1
1.2.Objetivos do trabalho	2
1.3.Estrutura do trabalho	3

Revisão Bibliográfica

2.1.Introdução	5
2.2.Evolução dos modelos de dados para bases de dados e seus conceitos	6
2.2.1.Modelos baseados em registros.	6
2.2.2.Modelos Semânticos	7
2.2.2.1.SDM	9
2.2.2.2.SAM*	10
2.2.2.3.IFO	12
2.2.2.4.Modelo Entidade-Relacionamento	18
2.2.3.Modelos orientados a objetos	18
2.2.3.1 Modelo de Dados ORION	22
2.2.3.2. Modelo de Dados Jasmine	24
2.2.3.3.Modelo de Dados ObjectStore	26
2.2.3.4.Modelo de Dados GemStone	26
2.2.3.5.Modelo de Dados Ontos	27
2.2.3.6.Modelo de Dados OSM	27
2.2.3.7.Modelo de Representação de Objetos (MRO*)	29
2.2.3.8.ORM	29
2.2.3.9.INFOKIT	35
2.2.3.10.Sistemas de herança com exceções	38
2.3.Conclusão	41

Desenvolvimento de um Metamodelo de Dados com Base em Abstrações

3.1.Introdução	43
3.2.Conceitos que envolvem abstrações	44
3.2.1.Conceituação	45
3.2.2.Conceituação Formal	52
3.3.Abstração de Classificação	57
3.3.1. Descrição da Classificação em Modelos Comuns	58

3.3.2.Descrição da Classificação em Modelos de Meta-sistemas	62
3.3.3.Estrutura da Abstração de Classificação em Objetos	65
3.3.4.Interpretação	65
3.4.Abstração de Associação	68
3.4.1. Descrição da <i>Abstração de Associação</i>	69
3.4.2.Estrutura da Abstração de Associação em Objetos	71
3.4.3.Interpretação	71
3.5.Abstração de Generalização	72
3.5.1. Descrição da <i>Abstração de Generalização</i>	72
3.5.2.Estrutura da Abstração de Generalização em Objetos	75
3.5.3.Interpretação	76
3.6.Cruzamento das hierarquias de abstração	77
3.7.Conclusão	79

SIRIUS - Modelo de Dados Orientado a Objetos e Baseado em Abstrações de Dados

4.1.Introdução	81
4.2.Necessidades de Ambientes de Projeto para Engenharia	82
4.3.Conceitos	84
4.3.1.Objetos e Tipos de Objetos	84
4.3.2.Atributos e Tipos de Atributos	87
4.3.3.Características de Atributos	90
4.4.Abstração de Agregação	95
4.5.Abstração de Classificação	106
4.5.1.Classificação de Objetos em Tipos de Objetos	106
4.5.2.Atributos para a Formatação de Classificação	110
4.6.Abstração de Generalização	117
4.7.Abstração de Composição	123
4.8.Comparação entre as Abstrações	129
4.9.Conclusão	132

Contribuições e Sugestões para Futuras Pesquisas

5.1.Contribuições deste trabalho	134
5.2.Sugestões para Futuras Pesquisas	138

Referências Bibliográficas

Lista de Figuras

Capítulo 2

Figura 2.1 - Veículo modelado como hierarquia de partes	23
Figura 2.2 - Ilustração das possíveis contradições	39
Figura 2.3 - Exemplo de instanciação	40

Capítulo 3

Figura 3.1 - Ilustração do processo de modelagem	45
Figura 3.2 - Elementos de uma abstração	46
Figura 3.3 - Construtores semânticos do Modelo Entidade-Relacionamento	47
Figura 3.4 - Hierarquia dos conceitos de abstração	48
Figura 3.5 - Exemplo da hierarquia de conceitos aplicada ao ME-R	49
Figura 3.6 - Visão dos conceitos aplicados ao próprio ME-R	50
Figura 3.7 - Exemplo de ocorrências de abstração	54
Figura 3.8 - Abstração de classificação em modelos comuns	59
Figura 3.9 - Exemplo de abstração de classificação	60
Figura 3.10 - Abstração de classificação em modelos de metasistemas	61
Figura 3.11 - Exemplo da ocorrência da abstração de classificação em modelos de meta-sistemas	63
Figura 3.12 - Hierarquia da abstração de classificação em um modelo de dados para apoiar gerenciadores com característica de meta-sistemas	64
Figura 3.13 - Exemplos de ocorrência abstração de associação	68
Figura 3.14 - Exemplos de ocorrência abstração de associação	71
Figura 3.15 - Exemplo de abstração de generalização	73
Figura 3.16 - Ocorrência de um tipo de abstração de generalização	74
Figura 3.17 - Exemplo de ocorrência de propriedade de categorização	75
Figura 3.18 - Exemplo de ocorrências da abstração de classificação (a), da abstração de associação (b) e da abstração de generalização (c)	77

Figura 3.19 - Ilustração de como os objetos ocupam lugares diferentes em diferentes abstrações	78
--	----

Capítulo 4

Figura 4.1 - Abstrações utilizadas pelo SIRIUS	81
Figura 4.2 - Representação de um Objeto	85
Figura 4.3 - Notação de Atributos	89
Figura 4.4 - Símbolos utilizados nas características de atributos	91
Figura 4.5 - Exemplo de utilização do símbolo de característica	94
Figura 4.6 - Notação utilizada na agregação de atributos e exemplos	97
Figura 4.7 - Notação utilizada na agregação de objetos	99
Figura 4.8 - Exemplo de relacionamento triplo	100
Figura 4.9 - Exemplos de agregação	102
Figura 4.10 - Representação gráfica de objetos e tipos de objetos	108
Figura 4.11 - Exemplo de ocorrência de abstração de classificação e seus atributos de classificação	112
Figura 4.12 - Exemplos de atributos multivalorados na abstração de classificação	114
Figura 4.13 - Exemplo de uso de atributos “Extras”	115
Figura 4.14 - Abstrações utilizadas pelo SIRIUS	119
Figura 4.15 - Exemplo da notação da abstração de generalização	120
Figura 4.16 - Exemplos de ocorrências de abstração de generalização	121
Figura 4.17 - Representação de composição (colônia de objetos)	124
Figura 4.18 - Exemplo de representação de colônias	126
Figura 4.19 - Comparação entre as diferentes abstrações em SIRIUS	129

RESUMO

Este trabalho desenvolve um modelo conceitual de dados para ser usado como “ferramenta” para a construção de outros modelos de dados, obedecendo a um determinado padrão. Como resultando foi obtido um metamodelo que satisfaz esse objetivo. Para isso, identificaram-se as abstrações de dados fundamentais utilizadas em modelos de dados: as *abstrações de classificação, de associação e de generalização*, construindo-se com as mesmas um Metamodelo que permite a representação de outros modelos de dados.

O metamodelo é construído reconhecendo-se que os construtores semânticos de um determinado modelo de dados são construídos pela parametrização das abstrações utilizadas nesse modelo. Considera-se uma parametrização como a identificação das restrições que envolvem uma abstração e a atribuição de valores a essas restrições, segundo as possibilidades de cada restrição. Com essa parametrização foi estabelecido um arcabouço conceitual que define uma estrutura precisa para cada construtor do metamodelo.

A parametrização foi exemplificada aplicando-a ao Modelo Entidade-Relacionamento original, e foi depois utilizada na construção de um modelo orientado a objetos construído para atender a necessidades bem especificadas definidas à priori (atender às necessidades de aplicações de projeto e engenharia e aplicações científicas) que foi denominado SIRIUS.

ABSTRACT

A set of conceptual constructions is developed to be used as a “tool” to support the representation or construction of other data models. The result is itself a Data Model with the property of being a Meta model, which can be used as a reference standard for construction of other data models. The Meta model is based on the identification of the fundamental *data abstractions* used in the representation of actual systems. It is shown that such fundamental abstractions are the classification, the association and the generalization ones.

The Meta model permits the definition of the semantic constructors of a determined model through the parametrization of the abstractions used in each constructor of the model.

An abstraction is parametrized by collecting the set of restrictions occurring over the abstraction, and by setting each restriction with a particular value, creating an instance of the abstraction. In this way, the same abstraction can generate several distinct abstraction instances. Several abstraction instances of distinct Abstractions can be associated to build each semantic constructor of the target data model.

This process is exemplified using the Entity-Relationship Model as the target. The use of the same process to construct a new data model, aiming to support the requisites of one application domain, is illustrated by constructing a data model to support the requisites of the development of computer aided engineering design and scientific applications. This new data model is named SIRIUS, and incorporates the fundamental abstractions instantiated into semantic constructors tailored to the needs of those domains, as described in the related bibliographies.

Capítulo 1

Introdução

1.1.Introdução

A pesquisa atual em modelos de dados para bases de dados tem como um dos seus objetivos, a busca de um padrão para modelos de dados orientados a objetos e para sistemas gerenciadores de bases de dados orientados a objetos. Embora esteja sendo empreendido grande esforço neste sentido, ainda não há um padrão “universalmente” aceito para esses modelos e sistemas. Deve-se ressaltar que é de extrema importância alcançar-se um tal padrão, tanto por motivos “acadêmicos” quanto “comerciais”.

No início desse processo, duas linhas de trabalho se delinearam: uma linha chamada evolucionista [Atkinson-89] [Darwen-95], e uma outra denominada revolucionista [Stonebraker-90]. A primeira teoria “considera” que os modelos de dados orientados a objetos devem evoluir a partir dos conceitos do modelo de dados relacional, ou seja, apenas novos conceitos deveriam ser incorporados nos novos modelos, aproveitando-se assim todo o desenvolvimento já alcançado por muitos anos de pesquisas na área. Já a segunda “considera” que os modelos de dados orientados a objetos devem constituir uma teoria com conceitos novos, ou seja, não derivados a partir dos modelos relacionais. Outros trabalhos foram sendo escritos e uma aceitação sobre os conceitos básicos para um modelo orientado a objetos começou a surgir [Kim-90a].

Apontando nesse sentido já existem alguns padrões que começam a ser

alcançados e aceitos. Uma dessas propostas é constituída pelo padrão *ODMG* baseado no trabalho do grupo *OMG* (Object Management Group) [Cattell-95][Betz-95a] e, de alguns padrões de linguagens que permitem a interoperabilidade de objetos [Betz-94] entre vários sistemas, tais como o padrão *CORBA* [Betz-95b], o padrão *OpenDoc* [Rush-95] e o difundido padrão *OLE* [Brockschmidt-95] [Williams-95].

1.2. Objetivos do trabalho

Neste trabalho buscou-se a elaboração de um modelo conceitual de dados que pudesse ser usado como “ferramenta” para a construção de outros modelos de dados, os quais obedeceriam a um determinado padrão. Como resultado foi obtido um metamodelo que satisfaz o objetivo acima.

Inicialmente foram identificadas as abstrações de dados utilizadas em modelagens. Elas foram classificadas em *abstrações de dados obrigatórias*, correspondendo àquelas que um modelo deve necessariamente suportar e, as *abstrações de dados não obrigatórias*, correspondendo àquelas que um modelo suporta ou não, conforme sua concepção. Como obrigatórias foram identificadas as *abstrações de classificação, de associação e de generalização*. A abstração de associação foi considerada como sendo uma generalização de diversas outras abstrações, que assumem diversos significados ou definições nos diversos modelos, mas que conceitualmente se referem ao fato de que informações diversas, já presentes no modelo, associam-se entre si: as *abstrações de agregação, de composição, de sumarização, relacionamentos, referências cruzadas, etc.*

Para obter esse “padrão” foi idealizada uma parametrização dessas *abstrações de dados* utilizadas em modelagens, sendo os construtores semânticos do metamodelo construídos com base nessa parametrização. Considera-se uma parametrização como a identificação das restrições que envolvem uma abstração e a atribuição de valores a essas restrições, segundo as possibilidades de cada uma. Com essa parametrização construída foi estabelecido um arcabouço conceitual que define uma estrutura bem mais precisa para cada construtor do metamodelo.

A parametrização foi exemplificada através de sua aplicação ao Modelo Entidade-Relacionamento original [Chen-76], e depois utilizada na construção de um

modelo orientado a objetos denominado SIRIUS [Biajiz-96a], que foi construído para atender a necessidades bem especificadas definidas à priori (atender às necessidades de aplicações de projeto e engenharia e aplicações científicas) .

1.3.Estrutura do trabalho

No *capítulo 2* é feita uma revisão bibliográfica compreendendo o levantamento de modelos de dados encontrados na literatura, muitos dos quais contribuíram significativamente para o desenvolvimento do paradigma de modelagem. Os modelos foram classificados em três divisões: *modelos baseados em registros*, *modelos semânticos* e *modelos orientados a objetos*. Como modelos baseados em registros são apresentados os *modelos hierárquicos*, *modelo em rede* e o *modelo relacional*. Como modelos semânticos são descritos os modelos *SDM*, *SAM**, *IFO* e *ME-R*. Como modelos orientados a objetos são apresentados o *ORION*, *Jasmine*, *ObjectStore*, *GemStone*, *Ontos*, *MRO**, *OSM*, *ORM* e o modelo do projeto *INFOKIT*. Além desses modelos é apresentada sucintamente uma técnica de *resolução de exceções em sistemas de herança*, conforme a visão de pesquisadores da área de inteligência artificial.

No *capítulo 3* são apresentados os conceitos que envolvem as *abstrações de dados*. Devido à pluralidade de nomes e características que envolvem de uma maneira não uniforme as abstrações de dados, os conceitos foram definidos ou redefinidos em alguns casos, buscando-se uniformidade e clareza. São discutidas as *abstrações de classificação*, *agregação*, *generalização* e *composição*, sendo as abstrações de agregação e de composição definidas como especializações da abstração de associação. Apresenta-se um formalismo básico para cada abstração, como os componentes de um metamodelo para a construção e definição de modelos de dados. Nesse capítulo ressalta-se a *abstração de classificação*, a qual foi apresentada admitindo-se que esta possa constituir-se em múltiplos níveis de abstração, ao contrário da forma como é tradicionalmente definida nos modelos de dados, em que a abstração de classificação sempre ocorre em 2 ou no máximo 3 níveis (quando o modelo admite o nível de meta-modelo).

No *capítulo 4* é apresentado um novo modelo de dados denominado SIRIUS,

criado com base na conceituação apresentada no *capítulo 3*, e procurando seguir o paradigma de orientação a objetos, e atender aos requisitos de sistemas de apoio a projetos de engenharia e aplicações científicas. Como resultado obteve-se um modelo de dados com seus elementos de representação (construtores semânticos) bem definidos com relação ao tipo de abstração de dados que suporta. SIRIUS baseia-se nas abstrações de classificação, agregação, generalização e composição, sendo que a abstração de classificação é particularmente interessante, uma vez que oferece a possibilidade de classificação em vários níveis, conferindo forte capacidade semântica de modelagem.

No *capítulo 5* são apresentadas as conclusões deste trabalho, bem como idéias e propostas para a continuidade da pesquisa iniciada.

No *capítulo 6* são colocadas as referências bibliográficas.

Capítulo 2

Revisão Bibliográfica

2.1.Introdução

O trabalho de revisão bibliográfica teve como objetivos levantar características que envolvem modelagens de dados para bases de dados, modelos de dados utilizados para essas modelagens, e variações que envolvem as abstrações como semântica de modelagem. As características semânticas das abstrações descritas aqui foram “parametrizadas” e “tratadas” no capítulo 3, e suas aplicações em modelos de dados foram utilizadas no capítulo 4.

Na seção 2.2. é descrita a evolução que sofreram os modelos de dados na área de bases de dados. Procurou-se ilustrar modelos representativos dentro de suas características, através de uma descrição sumária dos mesmos. Na seção 2.2.3.8 é apresentado um modelo que propõe um núcleo básico de conceitos de modelagens agrupados de forma a servir como uma base para ferramentas CASE que podem utilizar diversas metodologias diferentes. Na seção 2.3.3.9 é apresentado um metamodelo como um formalismo para tradução de esquemas entre modelos de dados diferentes. Ambas as seções estão relacionadas com o formalismo apresentado no capítulo 3. Já na seção 2.2.3.10 é ilustrado como é feito o tratamento de sistemas de herança por pesquisadores da área de inteligência artificial. O sistema de herança é intimamente ligado aos relacionamentos de classificação/instanciação, generalização/especialização apresentados nos capítulos 3 e 4.

Dessa forma, procurou-se coletar uma “amostra” significativa de importantes trabalhos encontrados na literatura, os quais serviram de base para o desenvolvimento desse trabalho.

2.2.Evolução dos modelos de dados para bases de dados e seus conceitos

*Base de dados*¹ é o nome dado a uma coleção de dados armazenados adequadamente para serem eficientemente manipulados. Essa coleção de dados é a junção de dados necessários para uma ou mais aplicações e seus objetivos. Para que esses dados sejam dispostos lógica e fisicamente de forma adequada para armazenamento e recuperação, um “esquema” segundo um modelo de dados específico deve ser elaborado. Note que um esquema é a descrição da base, enquanto a base compreende os dados armazenados. Dessa forma, o esquema da base é a estrutura em que os programas de aplicação se baseiam para acessar e manipular a base de dados, sendo esse organizado segundo um modelo de dados.

Um *modelo de dados* é um conjunto de conceitos que pode ser usado para descrever a estrutura de uma base de dados e as operações sobre essa base de dados. Por estrutura, entenda-se: os tipos de dados, os relacionamentos, e as restrições que definem o esquema da base. Por operações, entenda-se: as operações que permitem recuperar, inserir, remover e modificar dados.

Nas próximas seções são descritos os diferentes aspectos que envolvem o paradigma de modelagem descrevendo a evolução dos modelos de dados [Mcleod-91][Navathe-92] [Sheck-90] e classificando-os em modelos baseados em registros, modelos semânticos e modelos orientados a objetos.

2.2.1.Modelos baseados em registros

Os primeiros modelos de dados idealizados para bases de dados foram os

1

O termo banco de dados é utilizado como sinônimo de bases de dados por pesquisadores da área. Nesse trabalho optou-se pela utilização do termo bases de dados.

modelos baseados em registros, os quais organizam a visão sobre como os dados estão esquematizados, em tipos de registros. O esquema da base de dados, segundo essa filosofia, consiste de um número de tipos de registros relacionados.

Os modelos que representam essa linha são o *modelo hierárquico* [Tsichritzis-76], o *modelo em rede* [Taylor-76] e o *modelo relacional* [Codd-70] [Codd-79]. Descrevendo resumidamente, o modelo hierárquico organiza os tipos de registros em uma estrutura de árvore, o modelo em rede organiza o esquema colocando os tipos de registros como nós de um grafo, e por fim, o modelo relacional organiza os dados com uma visão orientada a conjuntos. Entre os três modelos, o relacional é o mais utilizado por projetistas de bases de dados.

A estrutura de dados baseada em registros foi adequadamente desenvolvida para o suporte da álgebra das relações, sendo o modelo relacional totalmente formalizado segundo essa teoria, a qual pode ser encontrada detalhadamente em [Maier-83].

Apesar do grande sucesso na utilização comercial desses modelos, a visão que se obtém através dos esquemas de bases de dados desenvolvidos a partir deles é muito próxima à implementação.

2.2.2. Modelos Semânticos

A idéia de criar modelos que representassem a modelagem de uma forma mais lógica, com maior distância da implementação, e que atribuisse mais semântica às modelagens, direcionou as pesquisas para uma nova classe de modelos, os quais foram denominados *modelos semânticos* [Abiteboul-87][Batory-88][Feldman-86] [Hanna-95] [Hull-87][Peckham-88].

Os conceitos básicos dos modelos semânticos são:

- *tipos* - representação direta de tipos de entidades. Esses tipos podem ser divididos em *atômicos* ou *não atômicos*. Os atômicos são os tipos básicos como inteiros, reais, strings, etc. Os não atômicos são tipos construídos a partir dos tipos atômicos, os quais são conjuntos, listas, vetores, tuplas, etc. Os tipos são ainda divididos

em: tipos *imprimíveis* (“*printable*”)², os quais correspondem àqueles que possuem uma forma natural e concreta perante o usuário; *tipos abstratos*, os quais correspondem àqueles que não têm uma forma natural e concreta perante o usuário, como por exemplo o tipo Pessoa, o qual não possui uma forma direta de representação.

- *tipo construído agregação* - representa uma agregação de tipos de entidades em uma entidade que os compõe. Tipos diferentes de agregação aparecem nos diversos modelos, cada qual com uma possível e provável semântica particular, com regras e restrições específicas.
- *tipo construído agrupamento (ou associação)* - representa um conjunto de entidades de um mesmo tipo.
- *atributos* - aparecem como característica de relacionamentos, ou simplesmente como característica de tipos. Os atributos podem ser monovalorados ou multivalorados. Um atributo é monovalorado quando ele suporta apenas um valor em uma mesma ocorrência, e é multivalorado quando ele suporta mais de um valor em uma mesma ocorrência.

Os conceitos básicos listados acima são estruturados semanticamente através de vários tipos de abstrações de dados. Deve-se notar que nos diversos modelos, nem sempre as abstrações aparecem com o mesmo nome, e nem sempre são todas suportadas por um mesmo modelo. Abaixo são listadas as abstrações utilizadas pelos modelos semânticos.

- *Agregação* [Smith-77a]: corresponde ao ato de agregar atributos a entidades, ou seja, uma entidade é caracterizada através de atributos que são ligados a ela.
- *Identificação*: corresponde ao ato de fazer com que entidades sejam univocamente identificadas através de algum identificador.
- *Classificação e Instanciação*: classificação envolve classificar entidades similares em uma única entidade, e instanciação envolve instanciar entidades a partir de suas classes.

- *Generalização/especialização* [Smith-77b]: as instâncias de um tipo de entidade podem ser um subconjunto de instâncias de outro tipo de entidade. Por exemplo, um tipo de entidade *Estudante* pode representar uma especialização de um tipo de entidade *Pessoa*. O tipo *Estudante* é denominado subtipo do tipo *Pessoa*, o qual é denominado supertipo. O relacionamento entre as duas entidades é chamado relacionamento “is-a”. Os relacionamentos “is-a” estabelecem *herança de atributos*, na qual um subtipo herda automaticamente os atributos de suas superclasses.

Inicialmente, os modelos semânticos foram introduzidos como ferramentas para a elaboração de projetos de esquemas. Algumas vantagens dos modelos semânticos sobre os baseados em registros são o aumento da separação dos componentes físicos e conceituais, a diminuição da sobrecarga semântica sobre tipos de relacionamento e a disponibilidade de mecanismos de abstração.

Abaixo são apresentados resumidamente alguns modelos semânticos enfocando-se os construtores que esses modelos suportam e quais abstrações eles utilizam. Estes modelos foram escolhidos pela importância que representam na evolução da teoria de modelagem, uma vez que existem muitos modelos citados na literatura.

2.2.2.1.SDM

O modelo *SDM* (Semantic Data Model) [McLeod-81] é um modelo rico em construtores semânticos, sendo sempre apresentado como referência por outros modelos semânticos. O modelo suporta *componentes derivados de esquema e agrupamento de tipos*, sendo que os agrupamentos são definidos por expressões, podendo ser enumerados e controlados pelo usuário. O modelo suporta *atributos primitivos*, e através deles simula a *abstração de agregação*. Suporta *relacionamentos de subtipos* que são divididos em quatro categorias: i) aqueles definidos por atributos;

ii) aqueles definidos por operações de conjuntos sobre tipos existentes;

iii) aqueles que servem como extensão de alguns atributos;

iv) aqueles especificados pelo usuário.

O conceito de metatipo é suportado a fim de proporcionar facilidades para a representação de tipos como elementos individuais ou como conjuntos definidos pelo

usuário.

A riqueza de construtores que SDM oferece para a especificação de esquemas, proporciona a liberdade entre se utilizar um pequeno ou grande número de construtores para estruturar dados primitivos. Em modelos com um pequeno número de construtores, a representação de alguns conjuntos de dados requer a combinação artificial desses construtores; em um modelo com muitos construtores, tal como SDM, o projetista tem a possibilidade de escolha entre uma variedade de caminhos para representar o mesmo dado.

2.2.2.2.SAM*

O modelo SAM* (Semantic Association Model) [Su-86] identifica sete tipos de associações, tornando-se essencial na discussão sobre modelos semânticos.

SAM* possui dois tipos de elementos, os atômicos e os não atômicos.

- *Tipos Atômicos* → são os elementos que não podem ser decompostos, sendo objetos físicos observáveis, e cujos tipos são inteiros, reais, strings, tipos lógicos (booleans), etc.
- *Tipos Não atômicos* → são objetos físicos ou eventos cujos significados são descritos (ou definidos) em termos de outros conceitos atômicos ou não atômicos (ex.: vetores, arrays, conjuntos ordenados e não ordenados, textos, etc).

Baseando-se nesses dois tipos de elementos, o modelo fornece um rico conjunto de construtores, buscando exaurir os tipos de relacionamento possíveis que podem surgir em aplicações comerciais e estatísticas. Assim, estabelece formas especiais de associação, distinguindo diferentes usos de alguns construtores fundamentais dos modelos semânticos. Uma associação consiste no agrupamento de conceitos atômicos ou não atômicos. Diferentes tipos de associação podem ser distinguidos de acordo com diferentes propriedades, características operacionais e restrições semânticas. Sete classes de associação (representada por construtores de tipo) são utilizados por SAM*.

- *Associação de pertinência* (“*membership*”) - é um agrupamento de um conjunto de conceitos atômicos homogêneos. Os elementos do conjunto têm o mesmo tipo de dados e formam um domínio. Uma restrição de pertinência define quais elementos pertencem à associação. Este tipo de associação modela a semântica

de “é um membro de” em linguagens naturais. Por exemplo, um tipo Cidade pode ter membros como São Paulo, São Carlos, etc representados no esquema.

- *Associação de agregação* - fornece uma descrição de uma entidade e seus atributos; é o agrupamento de um conjunto de atributos ou características para definir um outro conceito.
- *Associação de interação* - tipos de entidades independentes definidas por associação de agregação, agrupadas para descrever um conjunto de eventos ou fatos que são o resultado de algumas interações (ações ou relacionamentos) entre as ocorrências desse tipo de entidade. Neste tipo de associação, todos os tipos de conceitos componentes formam uma chave composta cujos valores identificam univocamente os eventos ou fatos modelados pela associação de interação.
- *Associação de generalização* - são usadas para modelar relacionamentos “is-a”. Cada ocorrência de um tipo componente é um exemplo ou uma classe do conceito geral definido.
- *Associação Produto-cruzado* - é um agrupamento de tipos de conceitos cujas ocorrências são os resultados do produto cruzado tomado das ocorrências de seus tipos de conceitos componentes. Por exemplo, categorias na qual se têm Média-de-idade, Raça e Sexo, poderiam ser usadas para aplicações estatísticas como, por exemplo, características populacionais.
- *Associação de composição* - é um agrupamento de tipos similares ou não, onde cada qual é parte do todo modelado pela composição. Ela modela o relacionamento “é parte de”. Por exemplo, um tipo Comboio pode ser modelado como uma tripla com três coordenadas para conjuntos de tipos Carros, Caminhões e Navios.
- *Associação de sumarização* - é usada para especificar atributos para ambas *Associações de Produto-cruzado* e *Associações de composição*. Estes atributos tipicamente possuem valores estatísticos.

Apesar de as sete classes de associação terem sobrecarga semântica, elas se distinguem por sua semântica de atualização e restrições. Com o número de construtores semânticos que possui, SAM* possibilita uma semântica muito clara para a modelagem de esquemas de bases de dados, além de ter se tornado um modelo essencial para a discussão de modelos semânticos.

2.2.2.3.IFO

O modelo IFO [Abiteboul-87] é um modelo de bases de dados matematicamente definido, que considera conceitos de outros modelos como o relacional, o ME-R [Chen-76], o FDM [Shipman-81], e os aspectos estruturados do SDM. Foi desenvolvido como proposta para fornecer uma estrutura teórica para o estudo de aspectos estruturais de modelos semânticos. O modelo incorpora atributos e construtores de tipos para agregação e agrupamento, e distingue duas classes de relacionamentos “is-a”. Seus conceitos básicos são:

- *Tipos* - elementos atômicos do modelo.
- *Tipos Imprimíveis (“printable”)* - elementos com tipos pré-definidos, tais como strings, e inteiros, os quais servem como base para entrada e saída de dados.
- *Tipos Abstratos* - elementos do mundo real que não têm uma estrutura específica de representação, como por exemplo, um tipo Pessoa.
- *Tipos Livres (“free”)* - são entidades obtidas via relacionamentos “is-a”. Por exemplo, num relacionamento “is-a” entre os tipos Pessoa e Estudante, o tipo Estudante é classificado como sendo um tipo livre.
- *Tipos Conjuntos* - conjunto finito de tipos abstratos, imprimíveis ou livres.
- *Produto Cartesiano* - conjunto finito ordenado de tipos, obtido a partir de outros conjuntos.
- *Fragmentos* - são blocos de esquema do IFO, ou seja, o esquema é dividido em blocos lógicos que são denominados fragmentos. Fragmentos são usados como mecanismos de abstração para representar um tipo de objeto com sua estrutura interna e seus atributos.
- *Relacionamentos “is-a”* - indicam relacionamentos entre entidades formando uma hierarquia. Nessa hierarquia os elementos inferiores são chamados subtipos e os seus superiores de supertipos. Cada função definida sobre o supertipo é automaticamente definida no subtipo, ou seja, as funções sobre o supertipo são herdadas pelos seus subtipos.

O modelo IFO é bastante difundido pela riqueza de elementos que suporta associado à estrutura formal do mesmo. Sua estrutura formal é tomada como referência e está descrita abaixo.

O formalismo dos modelos semânticos é construído, na maioria dos modelos, em torno da teoria dos conjuntos e da teoria dos grafos. O IFO utiliza a teoria dos conjuntos para definir formalmente seus elementos básicos, e usa a teoria dos grafos para formalizar o esquema de bases de dados por ele modeladas.

Definição 1: Seja P um conjunto infinito de *tipos visíveis*, e seja A um conjunto infinito de *tipos abstratos*, onde $P \cap A = \emptyset$. Para cada tipo $\Gamma \in A$ ou P , considere $dom(\Gamma)$ como sendo o conjunto de símbolos chamado o *domínio* de Γ . Assume-se que para cada par de tipos Γ_1 e Γ_2 , que $dom(\Gamma_i)$ é infinito enumerável, e que $dom(\Gamma_1) \cap dom(\Gamma_2) = \emptyset$.

Elementos pertencentes ao $dom(\Gamma)$ para algum Γ , são chamados *objetos atômicos*.

Definição 2: Para cada $\tau \in P \cup A$ e $O \in dom(\tau)$, O é um *objeto (atômico)*. Se $n > 0$ e $O_1, \dots, O_n$ são objetos, então $[O_1, \dots, O_n]$ é um *objeto (tupla)*. E se $m \geq 0$ e $O_1, \dots, O_m$ são objetos, então $\{O_1, \dots, O_m\}$ é um *objeto (conjunto)*.

Começando com tipos visíveis e tipos abstratos, tipos mais "complexos" são construídos usando *construtores de tuplas* e *conjuntos*. Neste processo, um tipo particular chamado *tipo livre*, é usado também. Intuitivamente, o tipo *livre* é necessário, porque o tipo de um objeto não é sempre uma propriedade local; ele pode resultar do uso de um relacionamento "is-a".

Definição 3: Seja livre um símbolo não pertencente a $A \cup P$. Cada tipo pertencente a $A \cup P \cup \{\text{livre}\}$, é chamado um tipo atômico. O $dom(\text{livre}) = \{O \mid O \text{ é um objeto}\}$.

Definição 4: Um tipo é uma *árvore direcionada* $R = (V, E)$ tal que:

1. V é a união disjunta de cinco conjuntos V_p (vértices visíveis), V_A (vértices abstratos), $V_\bullet$ (vértices- $\bullet$), $V_\otimes$ (vértices- $\otimes$) e V_F (vértices livres);
2. vértices visíveis, abstratos e livres são folhas da árvore;
3. vértices- $\bullet$ tem um filho;
4. vértices- $\otimes$ tem um ou mais filhos, os quais são vistos como sendo ordenados.

Para cada tipo R , a definição dos objetos do tipo R é direta.

Definição 5: Seja R um tipo. O conjunto de objetos do tipo R , denotado $dom(R)$, é definido como:

1. Se R é um tipo atômico (visíveis, abstratos, ou livres), então $dom(R)$ é definido como na definição 1 acima;
2. Se a raiz r de R é um vértice- $\otimes$, e R_i é a subárvore filha de r , então $dom(R) = \{O_1, \dots, O_m \mid m \geq 0, \text{ e } \forall i, O_i \in dom(R_i)\}$;
3. Se a raiz r de R é um vértice- $\otimes$, e $R_1, \dots, R_n$ são as subárvores filhas de r , então $dom(R) = \{O_1, \dots, O_n \mid \forall i, O_i \in dom(R_i)\}$.

Definição 6: Seja R um tipo. Uma instância (objeto) de R é um conjunto finito de objetos do tipo R , isto é, um sub-conjunto finito do $dom(R)$.

Note-se que uma instância do tipo R pode também ser vista como um objeto do tipo $\otimes/R$.

Definição 7: Seja R um tipo, e O um objeto do tipo R . Seja r um vértice de R . O domínio ativo de r em O , denotado $act_r(O)$, é um conjunto de objetos definidos por:

1. se r é a raiz de R , então $act_r(O) = \{O\}$;
2. se $O = [O_1, \dots, O_n]$, e r é uma subárvore de R de raiz r_i , então $act_r(O) = act_{r_i}(O_i)$;
3. Se $O = \{O_1, \dots, O_m\}$, e r não é raiz de R , então $act_r(O) = \bigcup \{act_{r_i}(O_i) \mid i \in [1..m]\}$.

Para uma instância I de R , o domínio ativo de r de I é $act_r(I) = \bigcup \{act_r(O) \mid O \in I\}$.

Intuitivamente, o domínio ativo de uma instância I no nó p é o conjunto de todos objetos habitando o nó $p \in I$.

Definição 8: Um fragmento é uma árvore direcionada $R = (V, E)$ onde:

1. E é a união disjunta de dois conjuntos E_o (arcos de objetos), e E_f (arcos de fragmentos).
2. (V, E_o) é uma floresta de tipos, chamados os tipos de R .
3. A cabeça de cada arco fragmento, é a raiz de algum tipo R .
4. A calda de cada arco fragmento, é ou a raiz de R , ou o filho de um arco objeto de

vértice- $\oplus$ o qual: é a raiz de algum tipo de R ; e não é a raiz de R .

Deve-se notar pela definição acima, que cada tipo é um fragmento sem arco de fragmento.

Definição 9: Seja R um fragmento com raiz r ; seja $f_1 = (r, p_1), \dots, f_n = (r, p_n)$ os arcos fragmento de R com calda r ; seja R_o o tipo com raiz r ; e para cada $k, 1 \leq k \leq n$, seja R_k a subárvore máxima de R com raiz p_k . Uma instância de R é um par ordenado $I=(J, F)$ onde J é uma instância de R_o , e F uma função com domínio $\{f_1, \dots, f_n\}$ tal que $F_k(f_k)$, também denotado I/f_k , é uma função parcial com domínio J tal que para cada O em J tem-se:

- a. $I/f_k(O)$ é um objeto de R_k , se R_k é um tipo;
- b. de outra maneira, $I/f_k(O)$ é uma instância de R'_k , onde R'_k é o fragmento obtido de R_k pela remoção de sua raiz- $\oplus$.

Se $I/f_k(O)=(K, G)$, então algumas vezes, $I/f_k(O)$ é usado para denotar K .

Definição 10: Seja R um fragmento, e $I=(J, F)$ uma instância daquele fragmento; seja r a raiz de R ; seja $f_1 = (r, p_1), \dots, f_n = (r, p_n)$ os arcos fragmentos de R com calda r , seja R_o o tipo com raiz r , e para cada $k, 1 \leq k \leq n$, seja R_k a subárvore máxima de R com raiz p_k . Para cada vértice q de R , o domínio ativo de q em I , denotado $act_q(I)$, é o conjunto de objetos definidos por:

1. se q está em R_o , $act_q(I) = \bigcup \{act_q(O) \mid O \in J\}$;
2. se q é um descendente próprio de p_k para algum k , $act_q(I) = \bigcup \{act_q(I/f_k(O)) \mid O \in J\}$;
3. se $q = p_k$ para algum k , $act_q(I) = \{K \mid \text{para algum } O \in J, I/f_k(O) = K \text{ ou } I/f_k(O) = (K, G) \text{ para algum } G\}$.

Definição 11: Um grafo IFO é um grafo direcionado $S=(V, E)$ tal que:

1. E é a união disjunta de três conjuntos E_o, E_G (arcos de generalização), e E_S (arcos de especialização);
2. (V, E_o) é uma floresta de fragmentos chamados *fragmentos* de S .

As raízes dos fragmentos de S são chamadas *vértices primários*. Os vértices primários de um esquema S identificarão os tipos dos conjuntos de entidades de primeiro

interesse, que são representados por S , e também as funções de nível de topo definidas nestes tipos.

Definição 12: Seja S um grafo IFO com $S=(V,E)$ e $E=E_o \cup E_S \cup E_G$. Então, a definição de *grafo de objeto* de S é o grafo³ $(V, \rho(E_o) \cup \rho(E_S) \cup E_G)$.

Este grafo captura a direção da "determinação de estrutura" resultante, proveniente de um objeto, de arcos de especialização, e de arcos de generalização, em um esquema IFO.

Regras "is-a":

- IS-A 1: Cada vértice livre é, ou a calda de ao menos um arco de especialização, ou a cabeça de ao menos um arco de generalização, mas não ambos.
- IS-A 2: Para cada arco de especialização, a calda é livre ou a cabeça é primária.
- IS-A 3: Para cada arco de generalização, a calda é primária e a cabeça é primária e livre.
- IS-A 4: Não há ciclo no grafo de definição de objeto.
- IS-A 5: Dois caminhos direcionados de arcos de especialização compartilhando a mesma origem podem ser estendidos para um vértice comum.

Definição 13: Um esquema (IFO), é um grafo IFO que satisfaz as cinco regras de IS-A 1 até IS-A 5.

Definição 14: Seja S um esquema de bases de dados com fragmentos $R_1, \dots, R_n$. Seja I um mapeamento vindo de $R_1, \dots, R_n$. Seja I_k denotado por $I(R_k)$ para cada $k \in [1..n]$. Então I é uma instância de S se e somente se:

1. para cada k, I_k é uma instância de R_k ;
2. $act_p(I_i) \subseteq act_q(I_j)$ se $(p,q) \in E_S, p \in R_i, q \in R_j$;
3. $act_q(I_j) = \bigcup \{act_p(I_i) \mid (p,q) \in E_G, p \in R_i\}$ se há um arco de generalização conduzindo

Em um grafo (W,F) , se $G \subseteq F$, então o reverso de G , denotado por $\rho(G)$, é o conjunto $\{(q,p) \mid (p,q) \in G\}$.

para $q \in R_r$

Teorema 1: Seja S um esquema IFO, e $n > 0$. Então existe uma instância I de S tal que para cada vértice $p \in S$, $|act_p(I)| \geq n$.

Agora serão definidos tipos derivados. Para a definição de tipos derivados, um terceiro tipo construtor ($\oplus$) é usado. Intuitivamente, o tipo T formado pela "soma" de um tipo T_1 e um tipo T_2 terão domínio igual à união de T_1 e T_2 .

Definição 15: Um tipo derivado é uma árvore direcionada $R=(V,E)$ tal que:

1. V é a união disjunta de cinco conjuntos V_p (vértices visíveis), V_A (vértices abstratos), $V_\bullet$ (vértices- $\bullet$), $V_\otimes$ (vértices- $\otimes$) e $V_\oplus$ (vértices- $\oplus$) [nenhum vértice livre];
2. vértices visíveis e vértices abstratos são folhas da árvore;
3. vértices- $\bullet$ tem um filho;
4. vértices- $\otimes$ e vértices- $\bullet$ tem um ou mais filhos, os quais são vistos como sendo ordenados.

Definição 16: Seja R um tipo derivado. O conjunto de objetos do tipo R , denotado $dom(R)$, é definido por:

1. se R é um tipo atômico (visível ou abstrato), então $dom(R)$ é prontamente definido;
2. se a raiz r de R é um vértice- $\bullet$ e R_1 é a subárvore filha de R , $dom(R) = \{O_1, \dots, O_m\} \mid m \geq 0, \text{ e } O_i \in dom(R_1) \text{ para } 1 \leq i \leq m\}$;
3. se a raiz r de R , é um vértice- $\otimes$ e $R_1, \dots, R_n$ são as subárvores filhas ordenadas de r , então $dom(R) = \{[O_1, \dots, O_n] \mid \forall i, O_i \in dom(R_i)\}$;
4. se a raiz r de R , é um vértice- $\oplus$ e $R_1, \dots, R_n$ são as subárvores filhas ordenadas de r , então $dom(R) = \bigcup_{i=1..n} dom(R_i)$.

Definição 17: Uma atribuição (derivada) para um esquema IFO S é uma função que atribui um tipo derivado para cada vértice de S .

Definição 18: Seja α uma atribuição para o esquema S . Então α é total para S , se para

cada instância I de S e cada vértice r de S , $act_r(I) \subseteq dom(\alpha(r))$.

Intuitivamente, uma atribuição é total para um esquema S se todos objetos que podem aparecer em uma instância de S pode ser descrito por esta atribuição.

Definição 19: Seja R e T dois tipos derivados. Então R é dominado por T , denotado $R \leq T$, se $dom(R) \subseteq dom(T)$; e R é equivalente a T , denotado $R \approx T$, se $R \leq T$ e $T \leq R$.

Definição 20: Seja α e α' atribuições para o mesmo esquema S . Então α é dominado por α' , denotado $\alpha \leq \alpha'$, se $\alpha(r) \leq \alpha'(r)$ para cada vértice $r \in S$.

O formalismo apresentado acima define todas as características do modelo IFO. Deve-se ressaltar que o modelo foi criado baseado na estrutura formal, o que dá grande força a suas modelagens, garantindo que as estruturas estão corretas e completas.

2.2.2.4. Modelo Entidade-Relacionamento

O modelo Entidade-Relacionamento (ME-R) [Chen-76] é o modelo semântico mais conhecido e utilizado até hoje para a concepção de projetos lógicos de bases de dados, existindo várias extensões do modelo [NG-81] [Elmasri-94].

O ME-R suporta a representação de conjuntos abstratos de entidades, relacionamentos entre estes conjuntos de entidades, e atributos definidos a partir de ambos conjuntos de entidades e relacionamentos para valores imprimíveis. Os tipos de relacionamentos podem ser restritos a 1:1, 1:N, ou N : M, e atributos podem ser restritos a 1:1.

O ME-R foi originalmente proposto como uma ferramenta para projetos de sistemas, sendo o primeiro modelo semântico centrado em torno de relacionamentos, não atributos.

Na próxima seção são descritos os modelos orientados a objetos, seguindo a classificação proposta no início do capítulo.

2.2.3. Modelos orientados a objetos

Vários modelos semânticos foram propostos, novas aplicações surgiram, e mais recursos de modelagem tornaram-se necessários e foram sendo incorporados. Aplicações

mais complexas que não tinham suporte computacional começaram a tornar-se possíveis graças a evolução do hardware. Assim, aplicações envolvendo multimídia, automação de escritórios, bases de dados geográficas, bases de dados especialistas, etc, começaram a ser implementadas, e paralelamente uma nova classe de modelos denominada *modelo de dados orientado a objetos* foi desenvolvida, tais como em [Alhajj-93] [Ang-93] [Atzeni-93] [Bertino-93][Butterworth-91][Cattell-94] [Cattell-95] [Ebert-94] [Jackson-95] [Jaeschke-95] [Kog-94] [Liddle-94] [Melo-94][Oxborrow-88] [Rieu-91] [Song-93] [Souza-93] [Traina-88] [Vieira-91] [Zand-95] [Zhao-88].

No contexto de modelagem de dados, o paradigma de orientação a objetos resume-se na concentração de toda a informação em elementos denominados “objetos” [Snyder-93] . Dessa forma, todas as entidades conceituais de uma aplicação são modeladas através de objetos, sendo esses “dispostos” de maneira que representem da melhor forma possível a semântica do problema a ser resolvido.

Não há ainda um consenso sobre quais elementos um modelo orientado a objetos deve suportar. Abaixo serão apresentados alguns conceitos propostos por Kim [Kim-90a] como sendo os elementos que um modelo de dados orientado a objetos deve suportar.

- *Objeto* - é um item de dado representado por alguma memória privada e por um conjunto de operações. A estrutura da memória privada é determinada por definições de propriedades, e as operações são especificadas pela definição de métodos.
- *Classe* - reúne objetos similares e determina a implementação deste conjunto de objetos. Uma classe funciona como um gabarito (“*template*”) para definição de um conjunto de atributos de instância, um conjunto de mensagens para uso externo, e um conjunto de métodos que se aplicam para objetos daquela classe.
- *Instância* - ocorrência de um objeto dentro de uma classe.
- *Metaclasse* [Klas-95] - uma classe cujas instâncias são classes.
- *Tipo* - uma coleção de definições para a estrutura de um objeto e seus métodos. Um tipo pode ser construído sobre outros métodos.
- *Subtipo* [Sciore-89] - um tipo construído sobre outros tipos (supertipos), com o objetivo de obter uma descrição mais especializada em relação ao seus

supertipos.

- *Supertipo* - um tipo que representa informações mais genéricas em relação a outros tipos, os quais são denominados subtipos.
- *Método* - uma descrição de como executar as operações de um determinado objeto. Ele é feito de uma assinatura (identificação do método), declaração de variáveis temporárias, e uma sequência de declarações que se referem a variáveis temporárias e às propriedades do objeto.
- *Mensagem* - requisito para a condução de métodos entre os objetos.
- *Objetos complexos* - são objetos ligados via um relacionamento denominado “*é-parte-de*” (“*is-part-of*”). Um objeto participa de um relacionamento “*é-parte-de*”, se ele possui uma referência a outro objeto participante no relacionamento.

O paradigma de orientação a objetos é baseado nos conceitos acima, organizados segundo os seguintes princípios:

- *Encapsulamento de dados e métodos* - encapsulamento fornece abstração de dados. O estado de um objeto é acessível somente através de métodos. O estado de um objeto é geralmente representado por variáveis que são ocultadas aos usuários do objeto.
- *Métodos e passagem de mensagens* - um método é uma “porção” de código que implementa uma operação sobre um estado de um objeto. O conjunto de métodos determina o comportamento de um objeto, e a passagem de mensagens é o meio como os objetos comunicam-se.
- *Classificação e instanciação* - objetos são colecionados dentro de classes. Uma classe determina as propriedades estruturais e comportamentais de suas instâncias. As propriedades estruturais são especificadas por um conjunto de variáveis, e as propriedades comportamentais por um conjunto de métodos.
- *Herança* - uma classe é interpretada como um “*template*” para a criação de instâncias daquela classe, isto é, define como são as instâncias de um tipo, e a forma como estas instâncias são criadas como reflexões das “coisas” do mundo real. Uma classe pode herdar operações das suas “superclasses”, e pode ter suas operações herdadas pelas suas “subclasses”. Herança é definida como um mecanismo para compartilhar recursos nas hierarquias de classes.

- *Identidade de objeto* - o sistema define e mantém um identificador único para os objetos. Isto permite que objetos possam ter os mesmos atributos e valores, e sejam identificados sem a necessidade de chaves únicas para suas instâncias.

Assim como nos modelos semânticos, esses conceitos podem ser vistos como organizados segundo *abstrações de dados*. As abstrações de dados utilizadas são semelhantes às utilizadas pelos modelos de dados semânticos. Dessa forma, os conceitos descritos acima podem ser organizados através das abstrações de classificação/instanciação, generalização/especialização, identificação e associação (apresentada como agregação nos modelos semânticos).

Os modelos orientados a objetos são similares aos semânticos nos seguintes aspectos:

- *Abstração estrutural* → as abstrações citadas são fornecidas por ambos os tipos de modelos, sendo que herança é obrigatória em modelos orientados a objetos, mas não são em todos os modelos semânticos.
- *Tipos e valores* → ambos fornecem construtores para tipos e valores, permitindo aos projetistas de aplicações construir formas estruturadas de valores, tais como listas, conjuntos, “bags” e tuplas.

Os modelos diferem nos seguintes aspectos:

- *Tratamento de identificadores* → modelos semânticos compõem identificadores ou chaves baseadas em atributos internos ou propriedades internas. Modelos orientados a objetos usam identificadores externos para o objeto, uma vez que os identificadores permanecem invariantes, enquanto os objetos podem sofrer alterações.
- *Tratamento de herança* → modelos semânticos suportam somente herança de atributos, a qual é limitada entre subtipos e supertipos. Os modelos orientados a objetos fornecem herança estrutural e comportamental, ou seja, suportam herança de propriedades estruturais e de procedimentos ou métodos.
- *Ocultamento de informações e encapsulamento* → os modelos orientados a objetos permitem encapsulamento de métodos no interior de um tipo de objeto restringindo a visão sobre os mesmos.

Pode-se resumir as vantagens dos modelos orientados a objetos sobre os

semânticos como sendo: melhor modelagem do comportamento, facilidade de modelagem de objetos complexos, extensibilidade de tipos, e localização do efeito de mudanças na definição de dados.

A seguir são apresentados alguns modelos, dando ênfase apenas a suas características básicas. Os conceitos descritos, os quais já foram definidos acima, não serão detalhados.

2.2.3.1 Modelo de Dados ORION

O modelo ORION [Kim-87] [Kim-89a] [Kim-90b] é um dos componentes de um projeto de bases de dados denominado ORION, o qual teve início no fim de 1985 no Programa Avançado de Tecnologia de Computadores (Advanced Computer Technology - ACT) no MCC, Austin, Texas, EUA.

ORION representa suas entidades conceituais através de *objetos*, os quais são organizados através de *classes*, possuindo métodos e atributos que têm sua definição estruturada em classes.

As classes são organizadas segundo uma hierarquia denominada *hierarquia de classes*, a qual é estabelecida através da ocorrência de relacionamentos “is-a”, representando especialização entre classes, e ou generalização entre classes. A ocorrência da abstração de especialização estabelece herança de atributos e métodos entre as classes. O modelo permite que uma classe tenha mais de uma superclasse, possibilitando ocorrências de *herança múltipla*. Dispondo essa estrutura de classes em um grafo, tem-se uma estrutura de grafo acíclico denominada “*classe lattice*”. A “*classe lattice*” simplifica a modelagem, mas em algumas situações podem ocorrer conflitos de nomes nas instâncias e nos métodos. Esses conflitos podem acontecer entre uma classe e sua superclasse, ou entre as superclasses de uma classe. Para o primeiro tipo de conflito, a solução adotada é estabelecer que a definição de uma variável ou método na classe tem precedência maior do que na sua superclasse, ou seja, a definição no interior da classe se sobrepõe a qualquer definição que possa ser herdada de suas superclasses. Para o segundo tipo de conflito, ORION permite que o usuário diga de qual superclasse o método ou variável de instância deve ser herdado.

Objetos Complexos [Kim-89b] são suportados por ORION, e representam uma

coleção de instâncias que formam uma estrutura que captura o relacionamento “*is-part-of*” entre um objeto e objetos que ele referencia. Em ORION um objeto composto tem um único objeto como raiz, e essa raiz referencia múltiplos objetos filhos, sendo cada referência através de uma variável de instância. Cada objeto

Figura 2.1: Veículo modelado como hierarquia de partes.

filho pode por sua vez referenciar seus próprios objetos filhos, e assim por diante. Um objeto pai é proprietário da existência de seus objetos filhos, estabelecendo dependência existencial entre os objetos filhos e seu pai. As instâncias que constituem um objeto composto pertencem às classes que são organizadas em uma hierarquia. Essa coleção hierárquica de classes é denominada um esquema de objetos compostos, o qual consiste de uma classe raiz e um número de classes dependentes.

Na Figura 2.1 tem-se um exemplo de esquema de objetos compostos para veículos. As classes conectadas com linhas em negrito formam o esquema de objetos compostos, sendo a classe *Veículo* a classe raiz. Através das variáveis de instância *Corpo*, *Tipo de direção* e *Cor*, instâncias de *Veículos* são ligadas aos seus objetos dependentes, os quais pertencem às classes *Autocorpo*, *Autodireção* e *String*. A classe *Veículo* tem outra variável de instância chamada *Fabricante*, mas não é uma ligação para um objeto *Dependente*. As instâncias de *Autocorpo* e *Autodireção* são conectadas a outros objetos dependentes. Dessa forma, um esquema *Veículo* é uma instância da classe *Veículo* juntamente com uma instância de cada uma das classes *Autocorpo*, *Autodireção* e *String* (para cor). A chave indica um conjunto de objetos. A variável de instância *Portas* da classe *Autocorpo* representa um conjunto de instâncias de portas, cada qual tendo uma posição e cor. Cada classe não folha da hierarquia sobre um esquema de composição de objetos tem uma ou mais variáveis de instâncias, chamadas variáveis de instâncias compostas, que servem como ligações para classes dependentes. Em outras palavras, um esquema de composição de objetos é criado através de variáveis de

instância compostas, as quais tem classes dependentes como seus domínios. Esse tipo de ligação é denominado ligação composta.

ORION suporta as abstrações de classificação/instanciação através do conceito de classes, onde os objetos são instanciados segundo as propriedades de suas classes; as abstrações de generalização/especialização através do conceito de hierarquia de classes segundo relacionamentos “is-a”; e abstração de associação através da caracterização dos objetos através de seus métodos e de objetos compostos (relacionamentos “is-part-of”).

2.2.3.2. Modelo de Dados Jasmine

O modelo Jasmine [Ishikawa-93] é um modelo de dados orientado a objetos desenvolvido como parte de um projeto denominado “Projeto de Larga Escala sobre Sistemas de Bases de Dados Interoperáveis”, sediado nos Laboratórios Fujitsu Ltda, Japão. Esse projeto integra Sistemas de Gerenciamento de Bases de Conhecimento, Sistemas Multimídia e Sistemas Orientados a Objetos.

O modelo estrutura seus dados através de *objetos*, os quais correspondem a uma *coleção de atributos* que representam conhecimento comportamental e estrutural de um domínio. Os seus *atributos* são categorizados em *enumerados* e *procedimentais*. Atributos enumerados são usados para representar conhecimento estrutural tal como um *Nome*; atributos procedimentais são usados para representar conhecimento comportamental (métodos). Os objetos são categorizados em *instâncias* denotando conhecimento individual, e em *classes* definindo atributos aplicáveis a instâncias similares denotando conhecimento genérico. A *abstração de classificação/instanciação* é suportada através do conceito de classes. Uma classe em Jasmine não é só uma definição de tipos com características comuns a suas instâncias, mas também é um conjunto de instâncias. No sistema que suporta o modelo, é definido um atributo denominado “*Class*” para denotar a raiz da classificação, estabelecendo diretamente hierarquias de classes. Nessa hierarquia, uma instância pertence intrinsecamente a somente uma classe, sendo essa classe chamada intrínseca, enquanto as instâncias são chamadas instâncias intrínsecas. O conceito de classes implementa diretamente a classificação. Assim, por exemplo, pode-se ter uma classe *Pessoa* como sendo um conjunto de *Pessoas*, as quais classificam instâncias de classes *Paciente*, *Doutor*, *Adulto* e *Criança*. A *abstração de*

generalização /especialização é suportada através da identificação na hierarquia de classificação de superclasses e subclasses. Assim, como na classificação, é definido pelo sistema um atributo, no caso denominado *Super*, para denotar a raiz da generalização. Pelo exemplo de classificação acima, tem-se que *Pessoa* é uma superclasse, a qual inclui suas subclasses *Paciente*, *Doutor*, *Adulto* e *Criança* como conjuntos. Instâncias de uma subclasse pertencem virtualmente a sua superclasse (não intrinsecamente). A interpretação de uma classe é o conjunto união das instâncias intrínsecas da classe e das instâncias de suas subclasses. Uma classe folha tem somente instâncias intrínsecas. As características que são suportadas pelas superclasses são herdadas por suas subclasses, caracterizando herança de atributos. Uma classe é dividida em subclasses disjuntas, sendo estas subclasses coletivamente denominadas uma partição, sendo cada subclasse chamada membro da partição. Por exemplo, *Pessoa* pode ter uma partição consistindo de *Doutor* e *Paciente*. Uma partição denota uma categorização baseada em uma visão, sendo que diferentes visões geram diferentes partições. Assim, *Pessoa* pode ter outra partição consistindo de *Adulto* e *Criança*. Membros de partições distintas não são necessariamente disjuntas, como por exemplo, um *Paciente* pode ser um *Adulto*. Condições de categorização podem ser explicitamente especificadas para estabelecer a semântica tal como "*Idade* ≥ 18 " para *Adulto*. Então, o atributo *Idade* é chamado de atributo de categorização. Também se permite que uma classe possua múltiplas superclasses, o que caracteriza herança múltipla. *Objetos complexos* não são diretamente representados pelo modelo. Sua representação é feita através de objetos instanciados a partir de uma classe pré-definida no sistema denominada *classe relação*. Esses objetos relação simulam o tipo objeto complexo.

Jasmine suporta abstração de classificação/instanciação através da definição e instanciação de classes (hierarquia de classes). Essas hierarquias ocorrem associadas a relacionamentos de generalização. Assim, abstrações de generalização/especialização estão ligadas às ocorrências da abstração de classificação. A abstração de associação ocorre através da caracterização de objetos e de relacionamentos entre objetos por atributos.

2.2.3.3. Modelo de Dados ObjectStore

O modelo ObjectStore [Lamb-91] [Soloviev-92][Zand-95] foi desenvolvido como parte do projeto de Sistemas de Bases de dados ObjectStore. O nome do projeto é Object Design, e teve seu início em 1988 em Burlington, MA. Uma primeira versão do ObjectStore foi apresentada em 1991, tendo o projeto continuado e evoluído até os dias de hoje.

ObjectStore possui suporte à representação de *objetos*, possuindo um tipo estruturado denominado *Coleção*. Uma *Coleção* compreende a implementação de *tipos de objetos* como *conjuntos*, *listas*, “*bags*”, etc, e a implementação de *classes de objetos*. As *coleções* são organizadas em duas *hierarquias de classes*: uma para as próprias coleções, e a outra para cursores (para consultas). A base da hierarquia de coleção é o objeto *os-coleção*. Uma dessas hierarquias contém os objetos *os-conjuntos*, *os-listas*, *os-bags*, etc. As coleções podem incorporar objetos, como por exemplo, um objeto *Pessoa* pode conter um conjunto de *Crianças*. O modelo suporta *relacionamentos* modelando-os através de ponteiros, onde um ponteiro parte de um objeto relacionando-o a um outro. Do outro objeto participante do relacionamento, parte obrigatoriamente um ponteiro para o primeiro objeto, o que estabelece o relacionamento. São suportados relacionamentos com cardinalidade 1:1, 1:N e M:N. *Objetos complexos*, tais como *hierarquias*, *projetos*, *documentos* e *informação multimídia*, são modelados usando relacionamentos. ObjectStore possui agregação entre classes, suportando conjuntos, “bags” e listas. Sobre as classes conjunto e bag são definidas as operações de união, intersecção, diferença, entre outras. Através da hierarquia de classes, são estabelecidos relacionamentos “*is-a*”, gerando ocorrências de abstração de generalização. ObjectStore admite herança múltipla através de relacionamentos “*is-a*”.

2.2.3.4. Modelo de Dados GemStone

O modelo GemStone [Zand-95] foi desenvolvido por Servio Logic Corporation of Alameda, CA, com o objetivo de satisfazer necessidades comerciais, tanto ao nível de Sistemas de Informação Gerenciais, como Sistemas de Engenharia.

O modelo de objetos GemStone centraliza-se nos conceitos de *objetos*, *classes* e *mensagens*. As classes são organizadas em hierarquias de classes, com cada subclasse

herdando comportamento e estrutura de suas subclasses. Entretanto, GemStone não suporta herança múltipla. A força do modelo está na sua capacidade de representar objetos tais como imagens, documentos e som.

As abstrações de classificação/instanciação estão presentes na hierarquia de classificação, assim como as abstrações de generalização/especialização. A abstração de associação está presente na caracterização de objetos através de seus métodos e variáveis, porém, o modelo não suporta diretamente a associação que permite modelar objetos compostos.

2.2.3.5. Modelo de Dados Ontos

O modelo de dados Ontos [Zand-95] é um sistema comercial que foi desenvolvido em um projeto denominado Ontologic of Billerica, MA. Seu modelo é baseado em objetos, os quais são suportados em estruturas de classes. Herança múltipla é suportada através de relacionamentos “is-a” na hierarquia de classes. Por outro lado, objetos compostos não são suportados.

Classes em Ontos freqüentemente são denominadas tipos. Os tipos podem ter uma *extensão*, a qual corresponde a uma coleção de todas as instâncias de um determinado tipo. Ontos possui agregação de classes, o que gera o suporte para tipos conjuntos, listas e associação (arrays).

2.2.3.6. Modelo de Dados OSM

O modelo OSM (Object-oriented System Modeling) [Liddle-94] é um modelo de dados orientado a objetos que faz parte de um projeto maior que está sendo desenvolvido por pesquisadores da Brigham Young University, USA. Esse projeto tem como objetivo propor um processo para sistemas orientados a objetos, a fim de integrar fortemente as fases do ciclo de vida, modelo, linguagem, ferramenta, persistência, comportamento, paradigma de programação e estilo, informação e meta-informação, e abstrações em alto e baixo níveis. OSM foi criado com o objetivo de ser um dos elementos de fácil integração no processo proposto. Abaixo estão descritas as principais características do modelo.

Os componentes estruturais do modelo são as *classes de objetos* e os

relacionamentos. As *classes de objetos* suportam abstrações em duas formas: classificação/instanciação e generalização/especialização. *Conjuntos de relacionamentos* estabelecem relacionamentos entre classes de objetos suportando abstrações em várias formas: associações todo/parte e associações conjunto/elemento. OSM suporta comportamento entre objetos. Esse comportamento é definido através do uso de *redes de estado* para uma classe de objetos. Uma rede de estados é um “gabarito” para o comportamento dos objetos em uma classe. Essa rede é composta por *estados* e *transições* vindas de um conjunto de estados para outro. Uma transição tem um *gatilho* e uma *ação*, e pode ser associado a um ou mais estados prévios ou um ou mais estados subseqüentes. Uma transição é habilitada quando várias combinações de seus estados estão ativas, e em estado ativo quando está habilitado e seu gatilho é verdadeiro. Uma rede de estados pode modelar múltiplas linhas de controle, e objetos OSM podem exibir concorrência intra-objetos e entre objetos. OSM suporta encapsulamento em vários níveis. Primeiro, estados de objetos são totalmente protegidos, ou seja, nenhum outro objeto pode alterar um estado de objeto corrente. Segundo, OSM permite que os estados e a informação estrutural sejam visíveis a todos os objetos no sistema; entretanto, o projetista pode escolher e restringir a visibilidade de várias porções do modelo através de visões. Terceiro, uma visão pode conter classes de objetos e conjuntos de relacionamentos que são encapsulados como componentes “*read-only*”.

Utilizando o próprio OSM foi construído um metamodelo denominado Metamodelo OSM, o qual pode ser dividido em três camadas: em uma camada mais “alta” está o *metamodelo*, em uma camada intermediária está uma *instância do modelo*, e em uma última camada está a *instância dos dados*. O Metamodelo OSM é constituído de classes de objetos, as quais são a classe *Classe Objeto*, a classe *Conjunto de Relacionamento*, a classe *Estado* e a classe *Transição*, tal que cada uma representa um metatipo.

Essa estrutura de camadas se dá através das abstrações de classificação/instanciação, e de generalização/especialização. A abstração de classificação/instanciação ocorre apenas em dois níveis no modelo: do metamodelo para o modelo instanciado, e do modelo instanciado para as instâncias de dados.

2.2.3.7. Modelo de Representação de Objetos (MRO*)

O modelo de dados MRO* [Traina-86] [Traina-88] [Biajiz-92] é um metamodelo de dados desenvolvido para suportar a implementação de um sistema de gerenciamento de bases de dados orientadas a objetos. O modelo pode ser visualizado em três camadas. A primeira possui um *Metaesquema*, onde todos os elementos do MRO* são definidos utilizando o próprio MRO*. Todos os elementos definidos no metaesquema são tratados como *objetos*, sendo que um objeto centraliza as referências e informações (inclusive de identificação) que são mantidas no sistema. Cada elemento definido no metaesquema é intitulado *metatipo*, e representa um construtor semântico. Os *metatipos* suportados são *tipos de objetos*, *tipos de relacionamentos*, *modalidades de relacionamentos*, *tipos de atributos*, *subtipos de objetos*, *subtipos de relacionamentos* e *tipos de colônias*. Tipos de colônias constituem um tipo de objeto complexo, onde vários tipos de objetos são compostos em um tipo de objeto denominado tipo de colônia⁴. A segunda camada é formada por instâncias de metatipos formando um *esquema* para uma aplicação. Um esquema segundo o MRO* é constituído de tipos (instâncias de metatipos) organizados logicamente. A terceira camada é constituída pelas instâncias dos tipos constituindo uma base de dados instanciada a partir de um esquema. Dessa forma, o modelo possui três camadas distintas, as quais representam dois níveis de instanciação.

No MRO*, a abstração de generalização/especialização é suportada através do conceito de subtipos de objetos e subtipos de relacionamentos. A abstração de associação é suportada através da associação de atributos a objetos e a relacionamentos, e através do conceito de tipos de colônias. A abstração de classificação/instanciação é suportada através do conceito de tipos.

2.2.3.8. ORM

O modelo ORM (Object role modelling) [Bronts-95] foi desenvolvido com o propósito de representar um “núcleo” para modelagens baseadas em diferentes estilos. Esse núcleo se comportaria como um divisor comum entre tipos de modelagens, ou seja, os diferentes tipos de modelagens poderiam ser representados através do formalismo

⁴ O conceito “tipo de colônia” será definido mais detalhadamente no capítulo 4, seção 4.7.

idealizado pelo núcleo.

Um dos pontos a ser considerado nesse processo é que nas diferentes técnicas de modelagem, existem mais aspectos em comum do que diferentes entre elas, sendo que essas características em comum poderiam ser representadas em um núcleo comum. Com esse processo buscou-se atingir algumas vantagens, como por exemplo, o desenvolvimento de ferramentas CASE no topo do núcleo ORM, que suportem múltiplas técnicas de modelagem de dados. Isso permitiria que o desenvolvimento de projetos de sistemas de informações e múltiplas técnicas de modelagem pudessem ser empregados simultaneamente. Como resultado, membros do projeto poderiam, sem prejuízo para o projeto, trabalhar com a metodologia que preferissem.

A seguir são colocados os conceitos que constituem a *sintaxe do núcleo ORM*.

- *Tipos de objetos concretos* - são aqueles que podem ser diretamente representados, correspondendo aos tipos denominados *tipos rotulados* (*tipos "label"*). Como exemplo, um tipo de objeto *Idade* pode ser diretamente representado por um valor.
 - *Tipos de objetos abstratos* - são aqueles que não podem ser representados diretamente, e correspondem a tipos denominados *tipos não rotulados* (*tipos não "label"*). Por exemplo, um tipo de objeto *Pessoa*, o qual necessita de um tipo rótulo para ser representado, como *Idade*, *Nome*, etc.
- Os tipos concretos e os tipos abstratos são representados pelos conjuntos disjuntos L e N respectivamente. A união dos tipos de objetos concretos e dos tipos abstratos constitui o conjunto de todos os *tipos de objetos* $O = L \cup N$.
- *Tipos atômicos* - são tipos não decomponíveis em outros objetos e são representados por A , tal que $A \subseteq O$. Define-se tipos de objetos abstratos e atômicos como $\mathcal{E} \triangleleft N \cap A$, sendo $\mathcal{E}$ os tipos de entidades.
 - *Tipo fato* - corresponde ao conceito de tipo de relacionamento. Ele representa uma associação entre tipos de objetos, consistindo de um número de papéis denotando a forma como os tipos de objetos participam daquele tipo de fato. A conexão entre um tipo de objeto e um papel é chamada *predicado*. No núcleo, um tipo fato é considerado como sendo uma associação entre predicados, mais do que entre tipos de objetos. Um tipo fato pode ser tratado como um tipo de objeto

(“objeto identificação” do fato), e desempenhar um papel em outros tipos fatos. Formalmente, o conjunto de predicados é representado por P , e o conjunto de fatos por F . A função $Fato : P \rightarrow F$ conduz ao tipo fato na qual um dado predicado está contido, e é definido por $Fato(p) = f \Leftrightarrow p \in f$. A função $Base : P \rightarrow O$ conduz ao tipo de objeto que emprega o papel de predicado. Define-se um *tipo ponte* como um tipo fato especial, o qual relaciona (estabelece uma “ponte”) objetos abstratos com objetos concretos. Como exemplo, considere um relacionamento binário r entre duas entidades x_1 e x_2 com papéis p_1 e p_2 . Então $\mathcal{E} = \{x_1, x_2\}$, $F = \{r\}$ e $P = \{p_1, p_2\}$ (estilo de modelagem E-R).

- *Tipo Potência* - corresponde ao tipo agrupamento encontrado na literatura, e é formalmente representado por G , $G \subseteq O$. Uma instância de um tipo potência é um conjunto de instâncias de seus elementos. Um elemento instanciado pode ser encontrado através da função $Elt : G \rightarrow O$.
- *Tipo seqüência* - tipos que representam seqüências formadas de tipos básicos, sendo representado por S , $S \subseteq O$. Um elemento da seqüência pode ser encontrado através da função $Elt : S \rightarrow O$. Note que a função $Elt : S \cup G \rightarrow O$ é uma função com sobrecarga de operações.
- *Hierarquia de identificação* - generaliza a generalização e a especialização, ou seja, a generalização e especialização são casos especiais de herança na hierarquia de identificação. Uma hierarquia de identificação é definida como sendo ordem parcial (assimétrica e transitiva) $IdfBy \subseteq A \times O$ sobre tipos de objetos, com a convenção de que $a IdfBy b$ é interpretada como: a herda sua identificação vindo de b . Note que a hierarquia de identificação lida com herança de identificação somente via especialização ou generalização.

A natureza da ordem parcial é expressa por:

ORM1 (assimétrica) $x IdfBy y \Rightarrow \neg y IdfBy x$.

ORM2 (transitiva) $x IdfBy y \wedge y IdfBy z \Rightarrow x IdfBy z$.

Define-se $IdfBy_1$ sem contrapartida de identificação $IdfBy$:

$$x IdfBy_1 y \triangleq x IdfBy y \wedge \neg \exists z \text{ tal que } [x IdfBy z \wedge z IdfBy y]$$

Todos os tipos de objetos na hierarquia de identificação têm superiores diretos:

ORM3 (superiores diretos) $x IdfBy y \Rightarrow x IdfBy_1 y \vee \exists p \text{ tal que } [x IdfBy_1 p \wedge p IdfBy y]$.

ORM4 (indução de identificação) Se F é uma propriedade para tipos de objetos, tal que:

$$\forall x: y \text{ IdfBy}_1 x [F(x)] \rightarrow [F(y)] \text{ para qualquer } y, \text{ então } \forall x \in O, [F(x)].$$

ORM5 (*span* completa) $x \text{ IdfBy}_1 y \rightarrow x \text{ Gen } y \vee x \text{ Spec } y$ e $x \text{ Gen } y \vee x \text{ Spec } y \rightarrow x \text{ IdfBy } y$

- *Especialização* - é um mecanismo para representar um ou mais (possivelmente sobrepondo) subtipos de um tipo de objeto. Especialização é aplicada somente quando, para instâncias específicas de um tipo de objeto, certos dados tiverem que ser registrados. O conceito de especialização é modelado como uma ordem parcial (assimétrica e transitiva) *Spec* sobre tipos de objetos, tal que *Spec* é uma parte da hierarquia de identificação. Assim, $a \text{ Spec } b$ indica que: a é uma especialização de b , ou a é um subtipo de b .

ORM6 (completeza da transitiva) Se $x \text{ IdfBy } y \text{ IdfBy } z$ então : $x \text{ Spec } y \text{ Spec } z \rightarrow x \text{ Spec } z$.

Note que a assimétrica para *Spec* se origina da assimétrica de *IdfBy*. Logo, $\text{Spec} \sqsubseteq \text{IdfBy}$.

Uma vez que a hierarquia de identificação, e portanto a hierarquia de especialização é não cíclica e finita, raízes podem ser identificadas. Para especialização, estas raízes são referenciadas como “*pater familias*” e são identificadas por: $\sqcap(x,y) \Delta (x \text{ Spec } y \vee x=y) \text{ not Spec}(y)$. Cada hierarquia de especialização tem um único elemento no topo.

ORM7 (“*pater familia*” - unicidade) $\sqcap(x,y) \wedge \sqcap(x,z) \rightarrow y = z$.

Este axioma permite considerar as relações “*pater familias*” como uma função parcial, e escrever $\sqcap(x) = y$ ao invés de $\sqcap(x,y)$.

- *Generalização* - é um mecanismo que permite a criação de novos tipos de objetos pela união dos tipos de objetos existentes. Generalização é para ser aplicada quando tipos de objetos diferentes desempenham papéis idênticos em tipos fatos. Generalização não é a inversa da especialização. Generalização e especialização originam-se de diferentes axiomas da teoria dos conjuntos, e por conseguinte têm diferentes forças expressivas. Especialização é baseada na compreensão de conjuntos $(\{x \in S \mid P(x)\})$ onde P é uma regra de definição do subtipo), e generalização corresponde ao conjunto união $(X \cup Y)$.

Para generalização exige-se que o tipo de objeto generalizado seja coberto pelos tipos de objetos constituintes (ou especificadores), isto é, a população da generalização se iguala à união de seus especificadores. Assim, um critério como no caso da especialização não é necessário. Outra diferença importante entre generalização e especialização está no fato de que propriedades são herdadas “upward” na hierarquia de generalização, enquanto na especialização as propriedades são herdadas “downward”. O conceito de generalização é introduzido como uma ordem parcial *Gen*. Assim, $a \text{ Gen } b$ indica que: a é uma generalização de b , ou b é um especificador de a .

ORM8 (completeza da transitividade) Se $x \text{ IdfBy } y \text{ IdfBy } z$ então: $x \text{ Gen } y \text{ Gen } z \leftrightarrow x \text{ Gen } z$.
ORM9 $\text{Gen}(x) \rightarrow \neg \text{Spec}(y)$.

São definidos especificadores básicos de um tipo de objeto generalizado, analogamente às “*pater familias*”: $\sqsubset(x,y) \triangleq (x \text{ Gen } y \vee x=y) \wedge \neg \text{Gen}(y)$. Note que a unicidade dos especificadores básicos não é requerida.

- *Tipo esquema* - é um tipo de objeto com uma decomposição básica. Esse conceito permite a decomposição de esquemas em subesquemas. O conjunto de tipos de esquemas é fornecido por $C \sqsubseteq O$. Tipos de esquemas podem ser decompostos em sua estrutura de informação básica via a relação $< \subseteq C \times O$, com a convenção de que $x < y$ é interpretado como x é decomposto em y ou y é parte da decomposição de x . A estrutura de informação básica I_x para um tipo de esquema x é derivado de tipos de objetos na qual x é decomposto: $O_x = \{y \in O \mid x < y\}$. Analogamente, as classes especiais de objetos F_x , G_x , S_x , C_x e A_x podem ser derivadas. As funções Base_x , Elt_x , $<_x$, Spec_x , $\sqsubset_x$ e Gen_x são obtidas pela restrição de tipos de objetos em O_x .

Para ser uma decomposição, uma estrutura de informação básica deve formar uma estrutura de informação por si só:

ORM11 (aninhamento estrutural) $x \in C \rightarrow I_x$ é uma estrutura de informação ORM.

A seguir são colocados os aspectos *semânticos do núcleo ORM*.

Nomeação de conceitos - Tipos de objetos e combinações de predicados podem receber nomes. Tipos de objetos são referenciados por um nome único: $ONm : O \rightarrow \text{Names}$, o qual é especificado no esquema. A função (parcial) $Obj : \text{Names} \rightarrow O$ é a inversa de ONm , e relaciona os nomes de tipos de objeto aos seus tipos de

objeto correspondentes. Pares de (diferentes) predicados são nomeados por meio de *nomes conectores*. $Conector(p, q) \Rightarrow p \in Fato(q)$

- *Restrições e semânticas* - uma estrutura de informação é usada como um modelo para alguma parte do mundo real, o assim chamado universo de discurso ($U \bullet D$). Um *estado* de um ($U \bullet D$) corresponde a uma instanciação ou população da estrutura da informação, e vice-versa. Uma população *Pop* de uma estrutura de informação *I* é um valor atribuído de conjuntos de instâncias para os tipos de objetos em *O*. Isto é denotado por $IsPop(I, Pop)$. *Pop* é um mapeamento $Pop : O \rightarrow \Gamma(\Omega)$, onde Ω é o universo das instâncias que podem ocorrer na população de uma estrutura de informação *I*.

Uma estrutura de informação somente pode ser populada se uma ligação é estabelecida entre tipos label e domínio concreto. As instâncias dos tipos label vem de seus domínios concretos associados. $Dom: L \cap A \rightarrow D$.

- *Tipos de Objetos Raízes* - são tipos de objetos que não são nem generalizados, nem especializados. $Raiz(x) \triangleq not\ gen(x) \wedge not\ spec(x)$

A população de um tipo rótulo (raiz) é um conjunto de valores vindos de seu domínio concreto correspondente:

$$POP1\ x \in L \wedge Raiz(x) \Rightarrow Pop(x) \subseteq Dom(x)$$

Se $Gen(x)$ ou $Spec(x)$, então $Pop(x) \subseteq \cup_{(y: x\ IdBy\ y)} Pop(y)$. Instâncias de um tipo de objeto generalizado ou especializado originam-se de tipos de objetos mais altos na hierarquia de identificação. Na generalização, todas as instâncias de um especificador são também instâncias do tipo generalizado, contrário da especialização, em que uma instância é guiada pela regra que define o subtipo. Assim, se $x\ Gen\ y \Rightarrow Pop(y) \subseteq Pop(x)$.

Algumas restrições podem ser definidas sobre instâncias.

1. *Restrição de unicidade* - a população de um predicado (*P*) não pode ter múltiplas ocorrências das mesmas instâncias. $Unicidade(p) \triangleq \forall x, y \in Pop(Fato(p)) [x(p) = y(p) \Rightarrow x = y]$
2. *Restrição de participação total para o papel* - todas as instâncias da base do predicado *p* precisam participar na população do fato daquele predicado. $Total(p) \triangleq \forall x \in Pop(Base(p)) \exists y \in Pop(Fato(p)) [y(p) = x]$.
3. *Restrição de exclusão* - a população de diferentes tipos de objetos não podem ter

instâncias em comum. $Exclusão(\{x_1, \dots, x_n\}) \triangleq \forall 1 \leq i < j \leq n [Pop(x_i) \cap Pop(x_j) = \emptyset]$

O núcleo ORM foi definido sintática e semanticamente. A sintaxe do núcleo é a definição do próprio núcleo. A semântica é estabelecida através de regras que estabelecem como trabalhar sobre a sintaxe. Partindo dessa premissa, é possível estabelecer a semântica para outros modelos colocando restrições específicas, ou seja, é possível “afinar” o núcleo para as técnicas de modelagem que se desejar. Como exemplo, abaixo é estabelecida a semântica para o modelo ER original.

ER1 (não suporta tipos complexos) - $G \cup S \cup C = \emptyset$

O ER puro não suporta tipos potência, nem tipos seqüência e nem tipos esquema.

ER2 (não suporta tipos rotulados) - $P_L \subseteq \cup B$

ER3 (não suporta objeto-identificação) $ran(Base) \cap F = \emptyset$

A operação *ran* retorna a extensão (domínio) de uma função. Não permite a generalização ou especialização de tipos de objetos.

ER4 (não suporta herança) $IdfBy = \emptyset$

Os construtores ORM identificados em ER são: tipos de entidades, tipos fatos, predicados, tipos rótulos e tipos ponte.

- os tipos de entidades (tipos não rotulados), os quais são modelados graficamente através de retângulos;
- tipos fatos (relacionamento) são modelados graficamente através de losangos;
- tipos ponte (atributos) são modelados por uma linha entre a parte abstrata e a parte concreta do tipo ponte.

Note que algumas das versões estendidas do ER permitem algumas das restrições colocadas acima.

Outros modelos podem ser definidos sobre o núcleo ORM da mesma forma que o modelo ER. Assim, propõe-se a representação de outros modelos sobre o núcleo ORM, e a construção de uma ferramenta CASE que suporte o núcleo ORM e múltiplas formas de modelagem.

2.2.3.9.INFOKIT

Foi desenvolvido por Atzeni e Torlone [Atzeni-93] um metamodelo com a proposta de representar uma estrutura para a definição de diferentes modelos de dados

e o gerenciamento de tradução de esquemas de um modelo para outro. A proposta pode ser formulada como: dados dois modelos M_1 e M_2 definidos por meio de um metamodelo, e um esquema S_1 (esquema fonte) de acordo com o modelo M_1 (modelo fonte), o modelo deveria ser capaz de gerar um esquema S_2 (esquema destino) que é a tradução do S_1 em M_2 (modelo destino). Para viabilizar essa proposta, vários problemas foram enfrentados, como o fato de ser praticamente impossível introduzir um “metamodelo” universal que permita a tradução de todos os modelos existentes.

A proposta compreende a elaboração de uma linguagem que permita a definição de uma extensa variedade de modelos, com base em um conjunto limitado de construtores que cubram a maioria dos modelos conhecidos, e a definição de um conjunto pré-definido de traduções básicas através de uma linguagem procedural.

Um *metamodelo* é um formalismo para definição de modelos, e um *modelo* é um formalismo para a descrição de esquemas, o qual descreve a estrutura de instâncias que contém os dados. Define-se um *construtor* como sendo um elemento do modelo, e *metaconstrutor* como sendo um elemento do metamodelo.

Metaconstrutores do metamodelo

- *Léxico* - permite a definição de construtores cujos conceitos tenham instâncias que são conjuntos de valores imprimíveis (“*printable*”).
- *Abstrato* - permite a definição de construtores cujos conceitos tenham instâncias que são conjuntos de valores não imprimíveis (“*not printable*”).
- *Agregação* - permite a definição de construtores cujos conceitos têm instâncias que são conjuntos de (rotulados ou não rotulados) tuplas de conceitos simples, chamados de componentes da agregação.
- *Função* - permite a definição de construtores cujos conceitos têm instâncias que são funções provenientes de um ou mais conceitos para outro conceito.

Baseando-se nesses construtores foi elaborada uma linguagem que define a estrutura de um modelo de dados.

Linguagem de definição do modelo.

```
<declaração_do_modelo> =
    Modelo <identificador>; Construtores {<declaração_construtor>;}
<declaração_construtor> =
    <declaração_léxica> | <declaração_abstrata> |
```

```

    < declaração_agregação > | < declaração_função >
< declaração_léxico > =
    Léxico < identificador >
< declaração_abstrato > =
    Abstrato < identificador > [(chave < identificador >)]
< declaração_agregação > =
    < grau > Agregação < identificador >
    Sobre < id_componente > {, < id_componente >}
    Chave < identificador >
< declaração_função > =
    < valor > Função < identificador >
    From < id_argumento > {, < id_argumento >}
    To < id_extensão > {, < id_extensão >}
< grau > = Binário | N-ário
< valor > = Monovalorado | Multivalorado

```

Para exemplificar, abaixo está definido o ME-R através da linguagem do metamodelo.

```

Modelo entidade-relacionamento;
Construtores
    Léxico domínio;
    Abstrato entidade (Chave identificador);
    Agregação n-ária relacionamento Sobre entidade
    Monovalorado função atributo
    From entidade, relacionamento to domínio;

```

Tradução de esquemas.

Quando a tradução de um esquema para outro ocorre de um modelo mais rico para um modelo mais pobre semanticamente, a informação embora preservada, sofre uma degradação tornando-se impossível traduzir o esquema para o modelo original (de volta). Isto estabelece alguns princípios que devem ser seguidos:

1. A tradução deve estar baseada na tradução de construtores básicos. Assim, para cada metaconstrutor que não está presente no modelo, o projetista tem que especificar como ele é representado. Dessa forma, assume-se que existe um “supermodelo”, o qual envolve todos os construtores, sendo a tradução especificada do supermodelo para outro modelo.
2. Como o número de construtores é limitado, é possível pré-definir um número básico de traduções, as quais são colocadas à disposição do projetista, o qual pode usá-los para construir traduções mais complexas.

O processo de tradução pode ser visto como composto de dois passos: i) vindo do modelo fonte para o supermodelo e ii) do supermodelo para o modelo destino. O primeiro passo é trivial, uma vez que todo modelo tem seu esquema como um subesquema do supermodelo. O segundo passo é executado por meio de traduções adequadas: conceitos cujo metatipo não tem no modelo destino são substituídos por conceitos permitidos. Note que o projetista deve definir para o modelo origem tipos construtores que substituam metatipos que não suporta.

Assim, foi proposta uma estrutura capaz de suportar a tradução de esquemas desenvolvidos em um determinado modelo, para esquemas em outros modelos seguindo um metamodelo pré-definido. Alguns aspectos devem ser considerados em busca de melhores resultados:

- a noção de “corretude” de uma tradução com respeito à noção de equivalência entre esquemas de diferentes modelos.
- a definição de hipóteses mínimas e máximas para serem satisfeitas por modelos a fim de garantir força expressiva e evitar redundância.
- a caracterização de um conjunto específico de funções básicas buscando a completeza das traduções entre modelos.

2.2.3.10. Sistemas de herança com exceções

Nesta subseção não é apresentado um modelo de dados orientado a objetos, mas um estudo sobre sistemas de heranças [Magnan-94] segundo o enfoque dado por pesquisadores da área de inteligência artificial. Esse estudo é considerado no contexto desse trabalho, porque trata de instâncias de variáveis e suas heranças, um assunto importante dentro da abstração de classificação/instanciação e da abstração de generalização/especialização utilizadas em modelos orientados a objetos. Dessa forma, essa apresentação é importante como comparação à metodologia usada nesse trabalho.

Na área de inteligência artificial são estudados sistemas de herança utilizando-se basicamente redes semânticas, linguagens de representação de conhecimento e mais recentemente linguagens de objetos. O principal aspecto de interesse nesse estudo é como gerenciar adequadamente esses sistemas de herança em bases de conhecimento. O estudo começa com a disposição dos sistemas em grafos, e a partir de propriedades e

características levantadas sobre esses grafos, estuda-se os problemas de gerenciamento de herança. Dessa forma, enfoca-se a detecção e solução de “exceções” que possam ocorrer no sistema de herança. As exceções levam a contradições que precisam ser controladas. A detecção e resolução desses problemas são formalizadas utilizando-se grafos para representar diagramaticamente o problema, e em seguida a tradução desse grafo para um formalismo estudado para esse fim.

A seguir são colocados resumidamente alguns conceitos que envolvem esses grafos.

- **Grafos de Herança** : os nós representam classes ou instâncias, e os arcos representam relacionamentos de generalização/especialização. Os arcos são chamados de ligações e podem ser positivos ou negativos conforme o sentido da ligação.
- **Caminho de herança**: um caminho de herança no grafo é aquele cuja sequência de arcos só pode ter o último arco negativo. Um caminho é dito ser negativo se sua última ligação for negativa; caso contrário ele é positivo.
- **Exceções**: uma exceção ocorre quando não é permitida herança proveniente de um determinado objeto. A principal consequência é que há perda de transitividade, ou seja, se um objeto não herda dados de um outro, então ele não pode herdar dados deste objeto, nem de qualquer outro de seus ancestrais na hierarquia.
- **Contradição**: considere x e y duas classes. O principal problema de gerenciamento de sistemas de herança com exceções

consiste em determinar se x herda dados ou não de y . Se existe apenas um caminho positivo ou negativo entre x e y , então a resposta é imediata, ou seja, a resposta é simplesmente positiva ou negativa. Dessa forma, uma contradição aparece quando há simultaneamente pelo menos um

caminho positivo e um caminho negativo entre x e y . São gerados três tipos de contradições, as quais são ilustradas na **Figura 2.2**:

Figura 2.2: Ilustração das possíveis contradições.

- **Tipo 1** → todos os vértices do grafo de herança podem ser comparados por relação de herança. O tipo 1a) corresponde a um caso simples: uma exceção em z é introduzida para indicar que z 's não são y 's. No tipo 1b), informação redundante é adicionada. Este tipo concentra uma redundância contraditória porque um dos caminhos conflitantes passa sobre uma ligação de transitividade da relação de herança.
- **Tipo 2** → Este tipo é uma generalização do caso anterior. Aqui, a ligação de transitividade é interrompida pelo nó w . Os objetos não são todos comparáveis, mas existem ao menos dois objetos os quais podem ser comparados. Tem-se o seguinte resultado: caminhos positivos e negativos estão na forma de $[x, \dots, v, y, +]$ e $[x, \dots, z, y, -]$; v e z podem ser comparados por relação de herança.
- **Tipo 3** → O terceiro tipo é uma ambigüidade. Para este tipo de contradição, todos objetos de caminhos contraditórios (exceto a origem e a extremidade) não são comparáveis: $\forall v \in [x, \dots, y, +]$ e $\forall z \in [x, \dots, y, -]$; v e z não podem ser comparados pela relação de herança.

O problema do gerenciamento de exceções é essencialmente o gerenciamento desses três tipos de contradições.

• **Exceções em Objetos:** em linguagens de objetos, herança pode ser expressa em termos de propriedades. O arco de herança $x \rightarrow y$ (resp. $\neg \rightarrow$) é traduzido por x tem a propriedade “Eu sou um y ” (resp. “Eu não sou um y ”). Herança, por

consequente, é uma propriedade que pode ser herdada, mascarada ou apresentar conflitos. Com essa definição, herança em objetos é definida como segue:

- um objeto y pode ser herdado por x , se x herda a propriedade “Eu sou um y ”.
- um objeto y não pode ser herdado por x se x herda a propriedade “Eu não sou um y ”.

Na **Figura 2.3** cada nó é rotulado por propriedades de herança que contém. “Clyde” herda as propriedades para ser um *elefante real* e um *elefante*. Por outro lado,

Figura 2.3: Exemplo de instanciação.

ele não é cinza porque a propriedade “*Eu não sou cinza*” de elefante real sobrepõe a propriedade “*Eu sou cinza de elefante*”.

O gerenciamento de exceções pode ser realizado por: *processo lógico*, *processo algébrico*, ou por *objetos*. O *processo lógico* usado é a lógica default. Nesse caso, a tradução do grafo de herança para o sistema lógico pode tornar-se complexo, pois aumenta a base de conhecimento. O *processo algébrico* consiste em raciocinar diretamente sobre o grafo de herança, isto é, raciocina em termos de caminhos. Existem vários algoritmos seguindo esses princípios. Um outro processo usado é o *processo de objetos*. Em linguagens de objetos, a linguagem é expressa em termos de propriedades, as quais permitem mascarar o princípio do mecanismo de herança usado para gerenciar exceções.

Deve-se observar que o objetivo é a resolução dos problemas de herança que possam surgir, estando todo o enfoque de estudo e resolução sobre as instâncias das classes ligadas a relacionamentos de generalização / especialização. O estudo sobre exceções tem aplicação direta em diversas áreas, inclusive na evolução de objetos, sendo que ainda há muitas questões a serem respondidas sobre essas ocorrências.

2.3. Conclusão

Muitos trabalhos envolvendo modelos de dados para bases de dados são encontrados na literatura. Esses modelos podem ser divididos e classificados em diferentes categorias, e dentro de cada categoria pode-se estabelecer um conjunto de elementos básicos que constituem a essência de sua representação. Pode-se agrupar esses conceitos segundo as abstrações que os envolvem, estabelecendo assim uma “base” para a elaboração de um “metamodelo” que sirva para a definição e a construção de outros modelos de dados.

No capítulo 3 é proposto um conjunto de elementos básicos baseados na parametrização de abstrações utilizadas em modelagens de dados para a categoria de modelos de dados orientados a objetos, e um metamodelo suportando esses conceitos. Os elementos incorporados ao metamodelo resultaram do estudo detalhado dos modelos descritos acima. Além da extração da semântica e sintaxe que envolvem as abstrações nos diversos modelos estudados, o formalismo [March-95] que os envolveu foi a base

para a construção de uma estrutura formal de sustentação do metamodelo.

No capítulo 4 mostra-se como utilizou-se o metamodelo obtido para a construção de um modelo de dados orientado a objetos denominado SIRIUS, o qual pode contribuir na busca de um padrão que norteie os modelos de dados orientados a objetos.

Capítulo 3

Desenvolvimento de um Meta-modelo de Dados com Base em Parametrização de Abstrações

3.1.Introdução

Neste capítulo procura-se definir mais claramente a conceituação que envolve abstrações, além de explorar o significado de algumas delas. Fazendo parte desses objetivos, algumas dessas abstrações serão exemplificadas e formalizadas.

Em modelos de dados, **abstração** é o nome dado a uma estrutura de representação de informação com características e propriedades particulares. Abstrair alguma coisa significa apresentar apenas o que é importante naquele momento para que essa coisa seja entendida, centralizando-se no que realmente é importante para o seu entendimento e desconsiderando-se detalhes que não interessem nesse instante.

Modelos de Dados são considerados no âmbito deste trabalho, como constituídos de uma coleção de abstrações semanticamente integradas. De forma geral, cada modelo contém suas próprias abstrações, define suas propriedades, além de nomeá-las de uma forma, que entre modelos, nem sempre é uniforme. Em outras palavras, diferentes modelos estabelecem *diferentes parâmetros* para abstrações que são chamadas com o

mesmo nome, e em outros casos definem *parâmetros iguais* para abstrações com nomes diferentes. Isso acarreta muitas vezes falta de clareza e dificuldades na utilização desses conceitos por parte de pessoas acostumadas com um outro modelo. No uso prático dos modelos de dados para modelagens diversas, a maioria dos usuários (analistas e projetistas de bancos de dados) não tem a percepção das reais características dos elementos que está utilizando.

Entre as abstrações mais comumente encontradas em modelos de dados estão aquelas chamadas de generalização, especialização, composição, decomposição, instanciação, classificação, agregação e associação, as quais são definidas e formalizadas nas seções seguintes.

No desenvolvimento deste trabalho são consideradas como básicas três abstrações: a *abstração de classificação*, a *abstração de generalização* e a *abstração de associação*. Propõe-se que as demais sejam obtidas pela especialização de uma ou mais abstrações básicas.

Também são explorados aspectos como o de que toda abstração organiza objetos em redes acíclicas, e discute-se que a rede gerada por uma abstração é ortogonal à gerada por qualquer outra abstração, no sentido que cada objeto ocupa um lugar em cada rede, e seu lugar em uma dada rede não acarreta vínculos quanto ao seu lugar dentro de qualquer outra rede. Além disso, mostra-se o importante aspecto de que essas abstrações estabelecem *Conjuntos Parcialmente Ordenados (CPO)* com um *elemento máximo*, isto é, toda rede gerada de abstrações tem exatamente um objeto sem qualquer superior.

3.2. Conceitos que envolvem abstrações

Na seção 3.2.1 é apresentada uma proposta de uniformização dos conceitos que envolvem abstrações. Para esse propósito é usado como exemplo o Modelo Entidade-Relacionamento, o qual por ser bastante utilizado e conhecido, estabelece excelente base para aplicação dos conceitos. Os conceitos descritos serão formalizados na seção 3.2.2.

3.2.1. Conceituação

Para a descrição de *abstrações*, o primeiro aspecto a ser abordado refere-se a “quando” e “como” uma abstração deve ser utilizada em uma modelagem de dados. Na **Figura 3.1** está ilustrado o processo que envolve a “transformação” de um problema real em uma representação implementável [Chen-76]. Suponha que exista um “problema” real para ser representado. Inicialmente o analista deve entender (Imaginar) o problema, transferindo a informação desse problema do “reino real” para um “reino imaginário”. A partir daí, utiliza-se um modelo de dados para modelar (esquematizar) essa informação. Nesse passo abstrai-se a informação que é necessária, e desconsidera-se a que não é, envolvendo as diversas abstrações existentes. Note por exemplo, que é diferente modelar-se informações de uma maneira mais genérica, ou de uma maneira mais específica. Caso ambas as formas sejam modeladas, a própria abstração é representada na modelagem. Nesse caso, a informação é mantida tanto em sua forma mais abstrata como em sua forma mais detalhada, e é possível ao sistema modelado “ver” a informação de uma maneira mais abstrata ou mais detalhada. Caso a abstração não seja modelada, o sistema somente poderá “ver” a

informação com o grau de detalhe “fixo” modelado. Note-se que ainda assim, durante o processo de modelagem o projetista utilizou qualquer uma das abstrações disponíveis (por exemplo, tanto generalizou quanto especializou), sendo que o processo de utilizar cada uma delas é diferente. Já uma abstração representada na

Figura 3.1: Ilustração do processo de modelagem.

modelagem, permite que o processo de abstrair ou detalhar seja usado livremente pelo sistema, e nesse caso, abstrair/detalhar passa a ser tratado par a par. Assim, durante o processo de modelagem aplica-se os conceitos de generalização, especialização, composição, decomposição, etc. Já no “reino da representação”, dispendo-se estaticamente dos elementos de representação envolvidos numa modelagem de dados,

a distinção entre as operações de abstrair e detalhar corresponde a “navegar” numa ou noutra direção os relacionamentos estaticamente disponíveis. Logo, distingue-se entre abstrair um dado e detalhar um dado, de forma que passam a existir as *abstrações de classificação*, de *associação* e de *generalização*, e os *detalhamentos de instanciação*, de *desassociação* e *especialização*. Nesse trabalho está sendo considerado o modelo numa visão estática, ou seja, as *abstrações* de classificação, de generalização, e de associação serão tratadas como sendo uma maneira oposta de navegar a informação daquela que é feita com o *detalhamento* de instanciação, especialização e desassociação.

Abstrações são usadas quando há a necessidade de manipular-se um conjunto de informações, ora de maneira mais detalhada, ora de maneira mais abstrata. Note que quando um modelo de dados utiliza uma *abstração*, a informação é mantida tanto em sua forma mais detalhada quanto em sua forma mais abstrata. Considerando-se uma informação como

Figura 3.2: Elementos de uma abstração.

uma coleção de objetos pode-se identificar um objeto como a representação da informação de maneira mais abstrata, chamado *Objeto Abstrato*, e a informação mais detalhada como representada por um ou mais objetos, chamados *Objetos Detalhe*. Na **Figura 3.2** é apresentada uma interpretação genérica de abstração, onde o objeto abstrato “relaciona-se” com o objeto detalhe através de relacionamentos *abstrai* e *detalha*. Do ponto de vista estrutural, abstrações são detalhes (dados) deliberadamente omitidos numa dada visão da implementação para facilitar a visão global do tópico em questão. Esse aspecto estrutural está implícito no *relacionamento* entre os objetos envolvidos. Do ponto de vista comportamental, abstrações determinam restrições, regras e operações que devem ser tomadas (automaticamente) na manipulação conjunta de elementos abstratos e elementos detalhes.

Da definição de uma abstração como sendo um elemento de representação de um modelo de dados (sintática e semanticamente falando) até a sua efetiva utilização em modelagens, existem vários conceitos envolvidos a serem colocados. Para a

apresentação desses conceitos de uma forma mais natural, considere como exemplo o Modelo Entidade-Relacionamento (ME-R) [Chen-76]. O ME-R possui 04 construtores semânticos que são: *Conjuntos de Entidades*, *Conjuntos de Relacionamentos*, *Atributos de Entidades* e *Atributos de Relacionamentos*. Na Figura 3.3 estão representados esses 04 construtores semânticos, além das abstrações correspondentes a esses construtores, ou igualmente falando, suportadas pelo modelo. Tem-se *Conjuntos de Entidades* e *Conjuntos de Relacionamentos* que classificam *entidades* e *relacionamentos* respectivamente.

Figura 3.3: Construtores Semânticos do Modelo Entidade-Relacionamento.

Assim, interpretando-se melhor a figura, tem-se também *conjuntos de entidades* associando *atributos*, *conjuntos de relacionamentos* associando *conjuntos de entidades* e *conjuntos de relacionamentos* associando *atributos*.

Enfocando o tipo de abstração de classificação, pode-se dizer que ele é utilizado no ME-R diferenciadamente, ou seja, possui características diferentes correspondendo a cada construtor semântico do modelo. O construtor semântico *conjunto de entidades* representa por si só, uma *particularização* da abstração de classificação. Quando o ME-R é aplicado para construir uma modelagem, cada *conjunto de entidades* resultante na modelagem é uma *ocorrência* da abstração de classificação; cada elemento pertencente a um conjunto de entidades representa uma classificação. O construtor semântico *conjunto de relacionamentos* também representa uma *particularização* da abstração de classificação. Do mesmo modo, durante modelagens utilizando o ME-R, cada conjunto de relacionamentos resultante é uma *ocorrência da abstração de classificação*.

Os conceitos são agora mais facilmente apresentados usando o exemplo acima. Pelo exemplo pode-se notar que o suporte às abstrações pelo modelo de dados é realizado através do conceito de **Construtores Semânticos**. Um construtor semântico determina como uma ou mais abstrações são utilizadas, e como se comportam em relação

aos demais Construtores Semânticos do modelo. Um modelo de dados pode usar a mesma abstração mais de uma vez, cada vez com características possivelmente diferentes. Desse modo, há uma particularização do tipo de uma abstração para cada construção semântica do modelo. Logo, um mesmo tipo de abstração pode ser usado com características diferentes dentro de um modelo. Cada vez que um construtor semântico é utilizado numa modelagem, criam-se Ocorrências de Abstrações. Esses conceitos podem ser organizados

e visualizados através de uma “hierarquia” de conceitos (Figura 3.4). No topo da hierarquia está o próprio conceito de abstrações. As abstrações são organizadas e caracterizadas em Tipos de Abstrações como: *abstração de*

Figura 3.4: Hierarquia dos conceitos de abstração.

associação e abstração de generalização e, portanto, pode-se dizer que o conceito de abstração é instanciado em abstração de classificação, abstração de associação, etc. Um tipo de abstração pode ser particularizado com características diferentes de um modelo para outro, ou até mesmo dentro do próprio modelo. Assim, surge o conceito de particularização de um tipo de abstração. Note que uma particularização de um tipo de abstração é por si só uma instanciação de um tipo de abstração. Cada tipo de abstração particularizado é aplicado sobre um ou mais **construtores semânticos** do modelo. Um construtor semântico, dado que pode envolver mais de uma abstração, é uma composição de particularizações de tipos de abstração. Cada vez que um desses **construtores semânticos** é utilizado numa modelagem, criam-se ocorrências de particularização de abstração.

Na Figura 3.5¹ pode-se ver a **hierarquia dos conceitos de abstração** aplicada

1

Por simplicidade, na maioria das figuras que seguem, somente serão representadas as setas que indicam a operação abstrai.

ao modelo ME-R. Em a) há um exemplo de modelagem utilizando o ME-R, e em b) a hierarquia é apresentada em níveis e aplicada à modelagem da parte a). Nos níveis apresentados em b) pode-se observar:

- no primeiro nível de representação o nome “Abstração”, o qual representa o elemento de mais alto nível na hierarquia;
- no segundo nível de representação, o tipo de abstração de classificação e o tipo de abstração de associação, que são os únicos tipos suportados pelo ME-R;
- no terceiro nível de representação, as particularizações do tipo abstração de

Figura 3.5: Exemplo da hierarquia de conceitos aplicada ao ME-R.

classificação em **Classificação de Entidade** e **Classificação de Relacionamento**, e as particularizações do tipo abstração de associação em **Associação de Atributos de Entidades**, **Associação de Atributos de Relacionamentos** e **Associação de Entidades**;

- no quarto nível de representação, os construtores semânticos do modelo, que são **Conjunto de Entidades**, **Conjunto de Relacionamentos**, **Atributos de Entidades** e **Atributos de Relacionamentos**. Todos os construtores semânticos representados são associações de uma particularização de abstração representando “*uma instância de*”.

Além disso, o construtor semântico Conjunto de Entidades também representa associação de atributos de entidades, e o construtor semântico Conjunto de Relacionamentos representa associação de atributos de relacionamento e associação entre entidades, ou seja, são associações de particularizações da abstração associação;

- no quinto nível de representação, as ocorrências da abstração de associação onde *aluno* e *disciplina* são conjuntos de entidades classificados pelo construtor semântico *conjuntos de entidades*, *Cursa* é um *conjunto de relacionamentos* classificado pelo construtor semântico *conjunto de relacionamentos*; *nome_a*, *nome_d*, *idade*, e *sigla* são *atributos de entidades* classificados pelo construtor semântico *Atributo de entidades*; e por fim, *nota* é um *atributo de relacionamento* classificado pelo construtor semântico *Atributo de Relacionamentos*.

Note ainda, que as linhas tracejadas horizontais dividem a **Figura 3.5** em três partes. A primeira (três primeiros níveis) engloba os *conceitos* de abstrações. A segunda

Figura 3.6: Visão dos conceitos aplicados ao próprio ME-R.

(quarto nível) engloba o *modelo de dados* em si com seus construtores semânticos, e a terceira corresponde à modelagem. Note que o modelo de dados é constituído pelos seus construtores semânticos, os quais possuem sua semântica definida pelas *particularizações de abstrações* que o envolvem.

Omitindo-se as representações da particularização das abstrações que o modelo suporta e a de seus construtores semânticos, e adicionando mais um nível de representação para a instanciação da própria base, obtêm-se a **Figura 3.6**. Nessa figura, o nível adicional representa as instâncias dos valores na base de dados, ou seja, tem-se a visão da modelagem e da classificação de seus valores em um determinado instante. Nessa figura é apresentada toda a hierarquia de abstração de classificação em um dado instante de uma base modelada segundo o ME-R. Note-se que uma instância de *Aluno* tem uma instância própria, representada na figura por *Idx1* e *Idx2*. Um *Aluno* está associado aos valores de atributos que *Aluno* pode assumir. Assim, como *Aluno* está associado a *Nome_a* e *Idade*, então cada instância de *Aluno* estará associada às instâncias de *Nome_a* e de *Idade*. Por exemplo, a instância de *Aluno*, *Idx1*, está associada às instâncias *José* de *Nome_a*, e *18* de *Idade*. Ou seja, *Idx1* é a associação (*José*, *18*), e do mesmo modo, *Idx2* é a associação (*Maria*, *19*). Da mesma forma, note que uma instância de *Disciplina* tem uma instância própria representada por *Idx5* e *Idx6*. *Disciplina* está associada aos valores de atributo de *Disciplina*, ou seja, está associada a *Nome_d* e *Sigla*. *Idx5* está associada às instâncias de *Nome_d*, *Bancos de Dados* e *Sigla B.D.*, enquanto *Idx6* está associada às instâncias de *Nome_d*, *Inteligência Artificial* e *Sigla I.A.* Logo, *Idx5* é a associação de (*Bancos de Dados*, *B.D.*) e *Idx6* é a associação de (*Inteligência Artificial*, *I.A.*). As instâncias de *Cursa* também possuem instâncias próprias, que na figura são *Idx3* e *Idx4*. Uma instância de *Cursa* tem associada as instâncias do atributo *Nota*, ou seja, *Idx3* está associado ao valor *8.7* e *Idx4* ao valor *7.7*. Além disso, *Cursa* que é um conjunto de relacionamentos, compreende uma associação entre *Aluno* e *Disciplina*, o que corresponde à associação entre os atributos *nome_a*, *sigla* e *nota*, resultando nas instâncias (*José*, *I.A.*, *8.7*) e (*Maria*, *B.D.*, *7.7*).

Até agora, a conceituação básica de abstrações de dados em modelos de dados foi apresentada informalmente utilizando exemplos para facilitar o entendimento. Essa conceituação é agora formalmente colocada.

3.2.2. Conceituação Formal

Nesta seção são formalizados os conceitos de abstração apresentados acima.

Definição 1: Define-se uma particularização de abstração A_i , como sendo uma enupla ordenada da forma $\langle T_{abs}, S_i, R_i \rangle$, onde:

- T_{abs} representa os conceitos que definem uma abstração, da qual esta particularização representa uma instância;
- S_i é o conjunto de todos os objetos reconhecidos a priori pelo modelo para aquela abstração;
- R_i é o conjunto de restrições definido para aquela abstração.

Uma mesma definição de abstração T_{abs} pode ser usada em mais de uma particularização de abstração A_i , do modelo M . Por exemplo, para definir um modelo que contemple a Abstração de Generalização empregando-a (particularizando-a) de determinada maneira, é necessário definir qual conjunto de restrições R aplica-se a essa abstração, e qual conjunto de objetos S é reconhecido pelo modelo para essa particularização.

Definição 2: Define-se Modelo de Dados Orientados a Objetos como sendo o conjunto dos construtores semânticos $M = \{Cs_1, Cs_2, \dots, Cs_n\}$ tal que cada Cs_i é a composição de um conjunto de particularizações de abstrações A_i . Dessa forma, o modelo M pode ser definido de maneira equivalente como o conjunto $M = \{A_1, A_2, \dots, A_n\}$ das particularizações das abstrações A_i , que integram o modelo M .

Definição 3: Define-se uma Base de Dados como sendo uma tupla $BD = \langle CO(\lambda), M \rangle$ onde:

- $CO(\lambda)$ é o conjunto de objetos que constituem a base de dados BD em um determinado instante λ ;
- M é o modelo de dados suportado por BD .

Definição 4: Define-se um objeto $ob_j \in CO(\lambda)$ de uma base de dados BD como sendo uma tupla ordenada da forma $\langle id_j, m_j \rangle$, onde:

- id_j é o identificador interno do objeto (OId) atribuído univocamente para esse objeto

pelo gerenciador da base;

- $m_j = \{<M>\}$ é o conjunto de dados que compõe os “atributos de estrutura”, tratados pelo gerenciador para manter as informações sobre como esse objeto insere-se na organização formada em cada Abstração A_i do modelo M que apoia a base de dados BD ;

Note-se que m_j numa base $BD = <Co(\lambda), \{A_1, A_2, \dots, A_{\#d}\}>$ tem a forma $\{aj_1, aj_2, \dots, aj_{\#d}\}$, onde cada aj_i é uma estrutura que permite ao gerenciador tratar o objeto ob_j correspondente à particularização A_i .

Definição 5: Define-se Ocorrência de uma Particularização de Abstração A_i , ou resumidamente Ocorrência de uma de Abstração A_i , uma enupla ordenada da forma $A_{t,i} = <A_i, Co_{d,t,i} = \{ob_j, j=j' \dots, tdi\}, Co_{a,t,i} = \{ob_{\#d}\}, Critério(i, A_i)>$, onde:

- $Co_{d,t,i}$ é o conjunto não vazio de objetos detalhe dessa ocorrência da abstração A_i ;
- $Co_{a,t,i}$ é o conjunto unitário de objetos abstratos dessa ocorrência da abstração A_i ;
- $Critério(i, A_i)$ estabelece uma regra que indica o motivo (significado lógico) da abstração.

Os conjuntos $Co_{d,t,i}$ e $Co_{a,t,i}$ estão contidos em $CO(\lambda)$, e λ está representando um determinado empreendimento em um determinado instante.

Para toda ocorrência i de uma abstração $A_{t,i}$, o conjunto $Co_{a,t,i}$ é unitário, ou seja, possui um único objeto abstrato $ob_{a,t,i}$ como elemento. As regras estabelecidas em $Critério(i, A_i)$ podem ser controladas pelo Gerenciador ou pelo próprio usuário. Note que assim como $Co_{d,t,i}$ e $Co_{a,t,i}$ o $Critério$ é próprio de cada ocorrência da particularização da abstração A_i .

Na Figura 3.7 estão ilustrados alguns exemplos. Em a) tem-se um exemplo de ocorrência de abstração, onde o conjunto abstrato *abstrai* o conjunto detalhe. Em b), c), d) e e) são utilizados para representação os elementos (objetos) desses conjuntos. Em b), o objeto "Endereço" é o objeto mais abstrato e os objetos "Rua" e "Número" os objetos detalhe. Simbolicamente, tem-se:

- $Co_{d, Assoc, 1} = \{"Rua", "Número"\}$;
- $Co_{a, Assoc, 1} = \{"Endereço"\}$;

• **Crítério(1,Assoc)** → indica que o objeto abstrato é uma associação dos objetos detalhe;

Tem-se o conjunto de objetos $CO(\lambda) = \{ "Rua", "Número", "Endereço", \dots \}$, onde λ representa o empreendimento em um determinado momento para a base de dados. A ocorrência da

abstração fica representada como

sendo $A_{Assoc,1} =$

$\langle A_{Assoc}, C_{A_{Assoc},1} \rangle$

$= \{ "Rua", "Número" \},$

$C_{A_{Assoc},1} =$

$\{ "Endereço" \},$

Crítério(1,Assoc)

>. Nesse caso, o

Crítério

corresponde ao

Figura 3.7: Exemplo de ocorrências de abstração.

significado que permite entender que diferentes objetos, "Rua" e "Número" associados, formam diferentes "Endereços".

A abstração apresentada no exemplo acima é a abstração de associação, e sua formalização será tratada nas próximas seções.

Na Figura 3.7 c) mostra-se outro exemplo, no qual se associam os valores representados pelos objetos "PioXI" e "951" ao objeto "PioXI,951". Na parte d) mostra-se as mesmas instâncias de abstrações das partes b) e c), além de três instâncias de abstração de classificação, indicando que "Endereço" é uma classificação para "PioXI,951", "Rua" é uma classificação para "PioXI", e "Número" é uma classificação para "951". Embora a parte d) seja adequada para mostrar conceitualmente uma situação do mundo real, quando essa situação é representada em um gerenciador de banco de dados real, na maioria das vezes se trata a associação de atributos (no esquema) de maneira diferente da associação de valores (na base). Isso caracteriza que a abstração de associação é particularizada em duas maneiras diferentes: *Particularização de*

Associação de Atributos, que é usada para associar atributos no esquema das aplicações, e *Particularização de Associação de Valores* que é usada para associar valores (usualmente por concatenação) na base de dados. A parte e) ilustra essa situação.

Através do exemplo mostrado na **Figura 3.7** pode-se concluir que um objeto pode ter “participação” em diferentes abstrações ao mesmo tempo. Assim, em **d)** vê-se o objeto “951” como sendo:

- objeto detalhe em relação ao objeto “*Pio XI*, 951” através da ocorrência de uma particularização da abstração de associação, e;
- objeto detalhe em relação ao objeto “*Número*” através de uma particularização da abstração de classificação.

Para mostrar esse “relacionamento individual” entre um objeto abstrato e um objeto detalhe será usada a seguinte notação: $\text{objeto abstrato} \underset{A_i}{>} \text{objeto detalhe}$, onde

$\underset{A_i}{>}$ indica a abstração envolvida. Com essa notação tem-se que “*PioXI*, 951” $\underset{A_{Assoc}}{>} “951”$ e “*Número*” $\underset{A_{Class}}{>} “951”$.

Usando a simbologia apresentada na definição 5, a parte **d)** seria assim representada:

- $Co_{d,Assoc,1} = \{“Rua”, “Número”\}$; $Co_{a,Assoc,1} = \{“Endereço”\}$; $Cr\acute{e}t\acute{e}rio(1,Assoc) \rightarrow$ indica que o objeto abstrato é uma associação dos objetos detalhe;
- $Co_{d,Assoc,2} = \{“PioXI”, “951”\}$; $Co_{a,Assoc,2} = \{“PioXI, 951”\}$; $Cr\acute{e}t\acute{e}rio(2,Assoc) \rightarrow$ indica que o objeto abstrato é uma associação dos objetos detalhe;
- $Co_{d,Class,1} = \{“PioXI”\}$; $Co_{a,Class,1} = \{“Rua”\}$; $Cr\acute{e}t\acute{e}rio(1,Class) \rightarrow$ indica que o objeto abstrato é uma classificação dos objetos detalhe;
- $Co_{d,Class,2} = \{“PioXI, 951”\}$; $Co_{a,Class,2} = \{“Endereço”\}$; $Cr\acute{e}t\acute{e}rio(2,Class) \rightarrow$ indica que o objeto abstrato é uma classificação dos objetos detalhe;
- $Co_{d,Class,3} = \{“951”\}$; $Co_{a,Class,3} = \{“Número”\}$; $Cr\acute{e}t\acute{e}rio(3,Class) \rightarrow$ indica que o objeto abstrato é uma classificação dos objetos detalhe;

A operação $\underset{A_i}{>}$ representa qual tipo de abstração está envolvido, além de

estabelecer uma “ordem” entre os objetos. Note que para diferentes ocorrências $A_{i,j}$ e

$A_{t,h}$, $i \neq h$ de uma mesma abstração A_i , um mesmo objeto pode fazer parte ora do conjunto de objetos detalhe $Co_{d,t,i}$ ora do conjunto de objetos abstratos $Co_{a,t,h}$. Além disso, uma abstração, qualquer que seja seu nome, sempre pode abstrair dois objetos quaisquer em um terceiro (mais abstrato) que resume a informação representada por ambos.

Definição 6: Um objeto $ob_j \in Co_{d,t,i}$ representa um detalhe de qualquer $ob_k \in Co_{a,t,i}$, sendo que $Co_{d,t,i} \cap Co_{a,t,i} = \emptyset$. Deve-se observar que um objeto ob_j ser detalhe de ob_k e ob_k ser detalhe de ob_j (segundo uma mesma ocorrência de abstração), é possível somente se $ob_k = ob_j$.

Considerando-se a operação $>_{A_i}$ das abstrações A_i , tem-se que $ob_k >_{A_i} ob_j$

implica que ob_j é detalhe de ob_k . Pelas definições apresentadas até aqui, pode-se definir as seguintes propriedades:

i) $ob_k >_{A_i} ob_j$ e $ob_j >_{A_i} ob_k \Leftrightarrow ob_k = ob_j$ (anti-simétrica)

ii) $ob_k >_{A_i} ob_j$ e $ob_j >_{A_i} ob_z \Rightarrow ob_k >_{A_i} ob_z$ (transitiva)

Definição 7: Com as propriedades descritas acima, pode-se afirmar que existe uma relação de ordem parcial entre os conjuntos $Co_{d,t,i}$ e $Co_{a,t,i}$ a qual será denotada por $>_{A_i}$.

Assim, um objeto $ob_j \in Co_{d,t,i}$ representa um detalhe de qualquer objeto $ob_k \in Co_{a,t,i}$ com $Co_{d,t,i} \cap Co_{a,t,i} = \emptyset$, tal que $ob_k >_{A_i} ob_j$ (diz-se que ob_k precede ob_j , estabelecendo a

relação de ordem parcial). Note que a relação de ordem parcial está definida sobre o conjunto $CO(\lambda)$ através da operação $>_{A_i}$.

Definição 8: Considerando o fato de que dois objetos podem ser abstraídos em um ou mais objetos através da operação $>_{A_i}$, e que uma abstração A_i estabelece uma relação

de ordem parcial sobre o conjunto $CO(\lambda)$, então $[CO(\lambda), >_{A_i}]$ é um *Conjunto*

Parcialmente Ordenado (Conjuntos-PO) sobre o conjunto de objetos $CO(\lambda)$.

Definição 9: Considere o conjunto de objetos $Cob(\lambda)$ parcialmente ordenado segundo a operação $>_{A_t}$, e $CO(\lambda) \subseteq Cob(\lambda)$. O objeto $ob_y \in Cob(\lambda)$ é o limite superior do conjunto $CO(\lambda)$, se $\forall ob_x \in CO(\lambda)$, $ob_y >_{A_t} ob_x$. Se além de ser limite superior do conjunto $CO(\lambda)$, $ob_y \in CO(\lambda)$, então ob_y é elemento máximo de $CO(\lambda)$. Por construção, $\exists ob_y \in CO(\lambda)$, tal que $\forall ob_x \in CO(\lambda)$, $ob_y >_{A_t} ob_x$. Dessa forma, ob_y é o elemento máximo de $CO(\lambda)$.

Assim, o conjunto de objetos $CO(\lambda)$ organizado segundo uma operação $>_{A_t}$ tem sempre um objeto abstrato “máximo” na hierarquia.

Definição 10: Como os *Crêterios* que indicam o motivo de cada ocorrência $A_{t,i}$ de uma abstração A_t , são funções apenas de A_t e de cada ocorrência i em particular, os operadores de ordem $>_{A_t}$ e $>_{A_u}$ de duas abstrações distintas respectivamente A_t e A_u , determinam Conjuntos-PO $[CO(\lambda), >_{A_t}]$ e $[CO(\lambda), >_{A_u}]$ linearmente independentes entre si. Desse modo, dadas quaisquer duas abstrações A_t e A_u de um modelo M , elas sempre são ortogonais entre si.

Embora o número de elementos de cada conjunto $Co_{a,t,i}$ seja sempre 1, o Conjunto-PO não recai no caso de uma *hierarquia simples*, pois um mesmo objeto pode fazer parte de mais de um conjunto detalhe, ou seja $Co_{a,t,i} \cap Co_{a,t,k} \neq \emptyset$. No entanto, como não existem ciclos no Conjunto-PO, utiliza-se neste trabalho o termo hierarquia não estrita, indicando-se que um mesmo elemento pode ter mais de um superior imediato, mas não se admitem ciclos, e existe sempre exatamente um objeto máximo.

3.3. Abstração de Classificação

Nessa seção será definida a *abstração de classificação* a qual será denotada como $A_{class} = \langle \text{"Classificação"}, S_{class}, R_{class} \rangle$.

A *abstração de classificação* está presente em todos os modelos de dados, e corresponde ao próprio conceito que permite a definição de regras comuns, as quais devem ser seguidas para que os objetos de um conjunto sejam tratados de maneira semelhante. *Classificar* significa reproduzir segundo um conjunto de regras, que pode ser comparado a um “molde” que dá forma para a criação de objetos [Traina-94].

A forma como uma *abstração de classificação* é tratada em modelos que suportam *Meta-Sistemas de Gerenciamento de Dados*, difere quando comparada a modelos comuns (que não suportam Meta-sistemas). Em meta-sistemas, existe um nível adicional de classificação que compreende os metatipos do sistema. Nesse caso os tipos de objetos são instâncias de metatipos, ou seja, existem metatipos classificando os tipos de objetos. Essas diferentes abordagens serão tratadas nas seções 3.3.1 e 3.3.2.

3.3.1. Descrição da Classificação em Modelos Comuns

Em modelos que não suportam meta-sistemas, a abstração de classificação apresenta apenas 3 níveis de ocorrências numa base de dados, os quais serão denominados nível 0, nível 1 e nível 2.

No nível zero, existe apenas uma ocorrência chamada

$A_{Class,0} = \langle A_{Class}, Co_{d,Class,0} = \{ "Abstração de Classificação" \}, Co_{a,Class,0} = \{ "Abstração" \}, Critério(0, A_{Class}) \rangle$, onde:

- $Co_{d,Class,0}$ é o conjunto não vazio de objetos detalhe dessa ocorrência da abstração A_{Class} ; no caso, a própria *abstração de classificação*.
- $Co_{a,Class,0}$ é o conjunto não vazio e unitário de objetos abstratos dessa ocorrência da abstração A_{Class} ; no caso é o objeto do sistema denominado “abstração”, que controla todos os objetos.
- $Critério(0, A_{Class})$ indica que o objeto abstrato “Abstração” é uma classificação do objeto “Abstração de Classificação” (ver **Figura 3.8**).

No nível 1, também existe apenas uma ocorrência chamada $A_{Class,1} = \langle A_{Class}$

$Co_{d,Class,1} = \{ob_j, j=j', \dots, Class1\}$, $Co_{a,Class,1} = \{Abstração de Classificação\}$, $Critério(1, A_{Class}) >$, onde:

- $Co_{d,Class,1}$ é o conjunto não vazio de objetos detalhe dessa ocorrência da abstração A_{Class} ;
- $Co_{a,Class,1}$ é o conjunto não vazio e unitário de objetos abstratos dessa ocorrência da abstração A_{Class} ; no caso, o objeto $Ob_{1Class} S_{Class}$ é a própria “Abstração de Classificação”.

- $Critério(1, A_{Class})$ indica que um objeto abstrato é uma classificação do objeto detalhe; no caso, o objeto “Abstração de Classificação” classifica os objetos tipo de objeto_i.

Na Figura 3.8 pode-se ver genericamente no primeiro nível, $Co_{a,Class,1} = \{“Abstração de Classificação”\}$ e $Co_{d,Class,1} = \{Tipo de Objeto 1, Tipo de Objeto 2, \dots, Tipo de Objeto n\}$.

Figura 3.8: Abstração de Classificação em Modelos Comuns.

O conjunto $CO(\lambda)$ é o conjunto de todos os objetos de uma base de dados, onde $Co_{d,t,1} \subseteq CO(\lambda)$, $Co_{a,t,1} \subseteq CO(\lambda)$ e λ é um determinado empreendimento.

No nível 2 existirão tantas ocorrências da abstração de classificação quanto for o número de elementos no conjunto $Co_{d,Class,1}$, ou seja, serão as ocorrências das abstrações de classificação $\{A_{class,i}, i=2, \dots, Class1\}$. Assim, na Figura 3.8 pode-se ver $A_{class,2} = \langle A_{Class}, Co_{d,Class,2} = \{objeto1, objeto2\}, Co_{a,Class,2} = \{Tipo de Objeto 1\}, Critério(2, A_{Class}) \rangle$, $A_{class,3} = \langle A_{Class}, Co_{d,Class,3} = \{objeto3, objeto4\}, Co_{a,Class,3} = \{Tipo de Objeto 2\}, Critério(3, A_{Class}) \rangle$, ..., $A_{Class,Class1} = \langle A_{Class}, Co_{d,Class,Class1} = \{objeto k\}, Co_{a,Class,Class1} = \{Tipo de Objeto n\}, Critério(Class1, A_{Class}) \rangle$.

A ocorrência do conjunto $Co_{a,Class,1}$, nem sempre reconhecida explicitamente nos

modelos comuns, corresponde ao conceito de que todas as classes² são conceitualmente construídas segundo um padrão específico para aquele modelo, ou seja $Co_{a,Class,I}$ possui apenas o objeto $Ob_{I,Class}S_{Class}$, que representa o conjunto de regras do modelo para a definição de classes. Nesses casos, $Ob_{I,Class}S_{Class} \in S_{Class}$ e, em geral, é seu único elemento. Os elementos do conjunto $Co_{d,Class,I}$ são as classes de objetos definidas na base de dados. Note que o objeto $Ob_{I,Class}S_{Class}$ é o objeto *Abstração de Classificação*.

Na **Figura 3.9** é apresentado um exemplo de ocorrência de abstração de classificação. No nível 0

tem-se $A_{Class,0} = \langle A_{Class}, Co_{d,Class,0} = \{ "Abstração de Classificação" \}, Co_{a,Class,0} = \{ "Abstração" \}, Critério(0, A_{Class}) \rangle$.

No 1º nível tem-se:

- $A_{Class,1} = \langle A_{Class}, Co_{d,Class,1} = \{ "Disciplina", "Pessoa", "Escola", "Nome_d", "Nome_p", "Nome_e" \}, Co_{a,Class,1} = \{ "Ob_{I,Class}S_{Class}" \}, Critério(1, A_{Class}) \rangle$.

Figura 3.9: Exemplo de Abstração de Classificação.

No nível 2 tem-se que cada objeto detalhe do 1º nível será um objeto abstrato para o 2º nível, ou seja, tem-se:

- $A_{Class,2} = \langle A_{Class}, Co_{d,Class,2} = \{ "Old1", "Old2" \}, ob_{a,Class,2} = \{ "Disciplina" \}, Critério(2, A_{Class}) \rangle$;
- $A_{Class,3} = \langle A_{Class}, Co_{d,Class,3} = \{ "Old3", "Old4" \}, ob_{a,Class,3} = \{ "Pessoa" \}, Critério(3, A_{Class}) \rangle$;
- $A_{Class,4} = \langle A_{Class}, Co_{d,Class,4} = \{ "Old5", "Old6" \}, ob_{a,Class,4} = \{ "Escola" \}, Critério(4, A_{Class}) \rangle$;
- $A_{Class,5} = \langle A_{Class}, Co_{d,Class,5} = \{ "B.D.", "I.A." \}, ob_{a,Class,5} = \{ "Nome_d" \}, Critério(5, A_{Class}) \rangle$;

²O termo classe de objeto é usado significando "o conjunto de todos os objetos do mesmo tipo".

• $A_{Class,6} = \langle A_{Class}, Co_{d,Class,6} = \{ "Carlos", "Rui" \}, ob_{a,Class,6} = \{ "Nome_p" \}, Critério(6, A_{Class}) \rangle;$

• $A_{Class,7} = \langle A_{Class}, Co_{d,Class,7} = \{ "UFSCar", "USP" \}, ob_{a,Class,7} = \{ "Nome_e" \}, Critério(7, A_{Class}) \rangle;$

Note que os objetos *Disciplina*, *Pessoa* e *Escola* são classificações de seus identificadores únicos (OId). O

objeto *Pessoa* possui o atributo *Nome_p*, o qual foi instanciado como *Carlos* e *Rui*. O objeto *Disciplina* possui o atributo *Nome_d*, o qual foi instanciado como *B.D.* e *I.A.* O objeto *Escola* possui o atributo *Nome_e*, o qual foi instanciado como *UFSCar* e *USP*³.

Cada ocorrência $A_{Class,i}$ corresponde ao conceito de que todos os objetos têm um tipo. Assim, $\bigcup_{i=1}^{Classd1} Co_{a,Class,i} = Co_{d,Class,1}$ é o conjunto de todas as classes de objetos

definidas na base (todos os tipos de objetos definidos para a aplicação), enquanto

$\bigcup_{i=2}^{Classd1} Co_{d,Class,i}$ é o conjunto de todos os objetos instanciados na base de dados. Um

objeto $ob_k \in Co_{d,Class,i}$ $2 \leq i \leq Classd1$ é uma instância de uma classe instanciada através de $A_{Class,i}$ (de uma classe pertencente a $Co_{a,Class,i}$ $1 \leq i \leq Classd1$).

Figura 3.10: Abstração de Classificação em Modelos de Meta-sistemas.

3

A “ligação” que estabelece qual atributo pertence a qual objeto, é feita através da abstração de associação que não está explicitamente representada na figura.

3.3.2. Descrição da Classificação em Modelos de Meta-sistemas

Em modelos de dados que suportam gerenciadores com a característica de serem meta-sistemas, existe um número maior de ocorrências distribuídas em pelo menos 4 níveis dentro da hierarquia, os quais podem ser vistos na **Figura 3.10**.

No nível zero, existe apenas uma ocorrência igual à apresentada na seção 3.3.1. para modelos comuns. A ocorrência é chamada $A_{Class,0} = \langle A_{Class}$

$Co_{d,Class,0} = \{ob_j, j=j', \dots, Classd0\}$, $Co_{a,Class,0} = \{Abstração\}$, $Crítico(0, A_{Class}) \rangle$, onde:

- $Co_{d,Class,0}$ é o conjunto não vazio de objetos detalhe dessa ocorrência da abstração A_{Class} ; no caso, a própria *abstração de classificação*.
- $Co_{a,Class,0}$ é o conjunto não vazio e unitário de objetos abstratos dessa ocorrência da abstração A_{Class} ; no caso é o objeto do sistema que controla todos os objetos.

No nível 1, como no caso de modelos comuns existe apenas uma ocorrência da abstração de classificação, igualmente definida como $A_{Class,1} = \langle A_{Class}$ $Co_{d,Class,1} = \{ob_j, j=j', \dots, Classd1\}$, $Co_{a,Class,1} = \{Ob_{1Class} S_{Class}\}$, $Crítico(1, A_{Class}) \rangle$, e igualmente o objeto $Ob_{1Class} S_{Class} \in S_{Class}$, por ser reconhecido pelo próprio conceito. Porém, os objetos do conjunto $Co_{d,Class,1}$ não se constituem nas classes definidas na base, mas constituem os conceitos do próprio modelo, ou seja, os *metaconceitos* que constituem a definição do modelo.

No nível 2 existirão tantas ocorrências da abstração de classificação quanto for o número de elementos no conjunto $Co_{d,Class,1}$, ou seja, serão as ocorrências das *abstrações de classificação* $\{A_{Class,i} \mid i=1, \dots, Classd1\}$, sendo que cada $Co_{d,Class,i} \mid 1 \leq i \leq Classd1$ corresponde ao conjunto de classificações de tipos da correspondente abstração $A_{Class,i}$ representada pelo objeto ob_j , $1 \leq j \leq Classd1$. Tem-se $A_{Class,2} = \langle A_{Class}$, $Co_{d,Class,2} = \{\text{"Tipo de Objeto1"}, \text{"Tipo de objeto2"}\}$, $Co_{a,Class,2} = \{\text{"Metatipo1"}\}$, $Crítico(2, A_{Class}) \rangle$, ..., $A_{Class,Classd1} = \langle A_{Class}$, $Co_{d,Class,Classd1} = \{\text{"Tipo de Objeto n"}\}$, $Co_{a,Class,3} = \{\text{"Metatipo n"}\}$, $Crítico(Classd1, A_{Class}) \rangle$. No nível 3 tem-se:

- $A_{Class, Classd1+1} = \langle A_{Class}$, $Co_{d,Class, Classd1+1} = \{objeto1, objeto2\}$, $Co_{a,Class, Classd1+1} = \{\text{"Tipo de Objeto 1"}\}$, $Crítico(Classd1 + 1, A_{Class}) \rangle$;

- $A_{Class, Classd1+2} = \langle A_{Class}, Co_{d,Class, Classd1+2} = \{objeto3, objeto4\}, Co_{a,Class, Classd1+2} = \{Tipo de Objeto 2\}, Critério(Classd1+2, A_{Class}) \rangle;$
- $A_{Class, Classd1+n} = \langle A_{Class}, Co_{d,Class, Classd1+n} = \{objeto_k\}, Co_{a,Class, Classd1+n} = \{Tipo de Objeto n\}, Critério(Classd1+n, A_{Class}) \rangle.$

Considere que $Cto(\lambda)$ seja o conjunto de todos os tipos de objetos definidos para uma base de dados. Assim, no nível 3 da hierarquia definida pela *abstração de classificação* poderão existir pelo menos tantas ocorrências de abstração de classificação

$$\text{quanto for o total de objetos no conjunto } Cto(\lambda) = \bigcup_{i=2}^{Classd1} Co_{d,Class,i}$$

Na **Figura 3.11** tem-se um exemplo de ocorrência de uma abstração de classificação em modelos de meta-sistemas. Deve-se observar que o conjunto $Co_{a,Class,1} = \{ "Objeto", "Atributo" \}$ são os metaconceitos que definem o modelo. Note que os objetos *Pessoa* e *Disciplina* são classificações dos seus identificadores únicos (Oid). O objeto *Pessoa* possui o atributo *NomeP*, o qual foi instanciado como *José* (*NomeP* classifica *José*). O objeto *Disciplina* possui o atributo *NomeD*, o qual foi instanciado como *B.D.* (*NomeD* classifica *B.D.*).⁴

$A_{Class,i}$ estabelece suporte ao conceito de que todos os elementos que formam a base de dados têm um tipo. Assim,

$$\bigcup_{i=2}^{Classd1} Co_{a,Class,i} = Co_{a,Class,1} \text{ é o}$$

conjunto de todos os conceitos do modelo para os quais exista representação na base de

Figura 3.11: Exemplo de ocorrência da abstração de Classificação em modelos de Meta-sistemas.

A “ligação” que estabelece qual atributo pertence a qual objeto, é feita através da abstração de associação que não está explicitamente representada na figura.

dados.

Tomando-se o exemplo anterior, o conjunto $Cto(\lambda) = \bigcup_{i=6}^{Classd1+5} Co_{d,Class,i}$ consiste de

todos os objetos que são tipos de algum elemento da base. Assim, $Cto(\lambda) = \{Pessoa,$

Figura 3.12: Hierarquia da Abstração de Classificação em um modelo de dados para apoiar Gerenciadores com característica de Meta-Sistemas.

$Disciplina, NomeP, NomeD\}$. Os objetos $\bigcup_{j=6}^{Classd1+9} Co_{d,Class,j}$ são todos os objetos da base

de dados, e os objetos $\bigcup_{k=Classd1+1}^{Classd2} Co_{d,Class,k}$ são todos os valores de atributos que ocorrem

na base de dados. Cada objeto do conjunto $Cto(\lambda)$ corresponde a um tipo de cada conceito do modelo classificado na base de dados. Considerando as ocorrências de abstração de classificação $A_{Class,k}$ tal que $Co_{a,Class,k} = \{ob_j\}$ com $ob_j \in Cto(\lambda)$, então $Co_{d,Class,k}$ são todos os objetos e demais elementos que compõem a base de dados. A Figura 3.12 ilustra essa hierarquia.

3.3.3. Estrutura da Abstração de Classificação em Objetos

O conjunto de dados que compõe os “atributos de estrutura” de abstrações em cada objeto $ob_j \in CO(\lambda)$ de uma base de dados $BD = \langle CO(\lambda), \bar{M} \rangle$ é reconhecido como $m_j = \{ \langle M \rangle \}$. Os “atributos de estrutura” são tratados pelo gerenciador para manter as informações sobre como esse objeto é inserido na organização formada em cada abstração suportada pelo modelo $\bar{M}$. Para a *abstração de classificação*, os dados que compõem M em cada objeto devem ser capazes de representar o Conjunto-PO $[CO(\lambda), \langle \rangle_{A_{Class}}]$. Esse Conjunto-PO, no caso específico da Abstração de Classificação

recai numa hierarquia simples, pois considera-se que todo objeto possui apenas um tipo que o caracteriza precisamente.

O tipo de qualquer objeto ob_j é um outro objeto ob_{ij} tal que, sendo $A_{Class,i}$ a ocorrência da abstração de classificação onde $Co_{a,Class,i} = \{ob_{ij}\}$, então $ob_j \in Co_{d,Class,i}$. Dado que qualquer objeto pode ser identificado univocamente por seu identificador, então a definição de um objeto ob_j passa a ser:

Definição 11: $\forall ob_j \in \{CO(\lambda) - \{Ob_{1Class} S_{Class}\}\}$, $ob_j = \langle id_j, m_j = \{M\} \rangle$, onde id_j é o (Oid) identificador do objeto, e $M = Estr_{Class}$ com $Estr_{Class} = \{ob_{ij}, id_{ij}\}$ onde id_{ij} indica o identificador do objeto ob_{ij} que é tipo do objeto ob_j . O objeto $Ob_{1Class} S_{Class}$ é o objeto *abstração de classificação*. Esse objeto não é representado na base, bem como nenhum dos objetos que são imediatamente classificados por ele.

3.3.4. Interpretação

Usualmente se considera que os objetos armazenados numa base de dados são classificados por suas respectivas classes, porém não se leva em conta que as classes por sua vez são também objetos classificados por outros mais abstratos. A discussão anterior parte do princípio que para todo objeto ob_j sempre existe um outro objeto ob_{ij} que classifica ob_j , ou seja, $ob_{ij} \succ_{A_{Class}} ob_j$, a menos do objeto “*Abstração*” considerado o

objeto máximo do Conjunto-PO $[CO(\lambda), \sup_{A_{Class}}]$. Note que o objeto $Ob_{1Class} S_{Class}$ é

reconhecido pelo próprio conceito de abstração de classificação, e portanto faz parte do conjunto S_{Class} . Mais do que isso, como usualmente a abstração de classificação reconhece apenas esse objeto, então $S_{Class} = Ob_{1Class} S_{Class}$.

Quando um Gerenciador de Bases de Dados não é construído para ser um Meta-sistema, as classes de objetos definidas em uma base de dados, ou seja, os elementos do conjunto $\bigcup_{i=2}^{Classd1} Co_{a,Class,i} = Co_{d,Class,1}$ são classificados diretamente de $Ob_{1Class} S_{Class}$, e

constituem a *base intencional*, ou o *Esquema da Base de Dados*. Já os objetos $ob_k \in Co_{d,Class,i}$, $2 \leq i \leq classd1$ são cada qual instância de uma classe definida no esquema, e constituem a base de dados extensional. É o caso apresentado na **Figura 3.9**. O objeto

“abstração de classificação” $Ob_{1Class} S_{Class}$, e $\bigcup_{i=2}^{Classd1} Co_{a,Class,i} = Co_{d,Class,1} = \{ "Disciplina",$

“Pessoa”, “Escola”, “Nome_d”, “Nome_p”, “Nome_e”} constituem o esquema da base de dados. Os conjuntos $Co_{d,Class,2} = \{ "Old1", "Old2" \}$, $Co_{d,Class,3} = \{ "Old3", "Old4" \}$, $Co_{d,Class,4} = \{ "Old5", "Old6" \}$, $Co_{d,Class,5} = \{ "BD", "IA" \}$, $Co_{d,Class,6} = \{ "Carlos", "Rui" \}$ e $Co_{d,Class,7} = \{ "UFSCar", "USP" \}$ constituem a base de dados extensional.

Quando um Gerenciador de Bases de Dados é construído para ser um meta-sistema, é possível definir-se cada um dos componentes que irão compor a base. Assim, o conjunto $\bigcup_{i=2}^{Classd1} Co_{a,Class,i} = Co_{d,Class,1}$ é o conjunto de conceitos que são utilizados para

definir o esquema, e não o esquema propriamente dito, ou seja, é o *Meta-esquema* da base, aquele que define como os esquemas deverão ser criados. Já os objetos do conjunto

$Cto(\lambda)$ classificados por $\bigcup_{i=2}^{Classd1} Co_{a,Class,i}$ são os objetos que constituem o *esquema*

propriamente dito de uma base de dados, sendo os tipos de todos os elementos que irão constituir a base, sejam esses elementos os objetos genéricos tratados como tal pelos

aplicativos, sejam objetos especializados, reconhecidos particularmente pelo próprio gerenciador. Os objetos instanciados diretamente a partir de $Cto(\lambda)$ são os objetos que compõem a base extensional. É o caso apresentado na Figura 3.11, onde

$\bigcup_{i=2}^{Classd1} Co_{a,Class,i} = Co_{d,Class,1} = \{ "objeto", "atributo" \}$ é o Meta-esquema da base. Os

conjuntos $Co_{d,Class,2} = \{ "Pessoa", "Disciplina" \}$, $Co_{d,Class,3} = \{ "NomeP", "NomeD" \}$ constituem o esquema da base de dados, e os conjuntos $Co_{d,Class,4} = \{ "Oid1" \}$, $Co_{d,Class,5} = \{ "Oid2" \}$, $Co_{d,Class,6} = \{ "José" \}$, $Co_{d,Class,7} = \{ "B.D." \}$ constituem a base de dados extensional.

A discussão apresentada nas subseções 3.3.1. e 3.3.2 considera a existência mínima de 3 e 4 níveis na hierarquia gerada pela *Abstração de Classificação*, porém em qualquer dos casos, é possível a existência de mais níveis. Para exemplificar, considere-se que uma base de dados é utilizada para armazenar os dados de uma ferramenta CASE. Nesse caso, os objetos armazenados nessa base poderão ser a representação de um programa que utilize uma base de dados, juntamente com o esquema dessa base alvo da ferramenta CASE. Caso essa ferramenta propicie o ambiente de execução desse programa, esses objetos serão então por sua vez instanciados. Como qualquer abstração, a de classificação não impõe limites para o número de níveis que o Conjunto-PO pode ter.

Note que apesar da repetição do termo meta, o fato de um Gerenciador de Bases de Dados ser construído como um Meta-Sistema é um conceito distinto do Metamodelo de dados discutido neste trabalho. O primeiro consiste em um gerenciador que pode utilizar uma base de dados (esquema) para definir o conteúdo de outra base de dados (extensão), tendo ambas, uma estrutura idêntica. O segundo representa um modelo que pode modelar outros modelos, inclusive a si próprio.

Como já foi mencionado, a *hierarquia de abstração de classificação* constitui uma hierarquia simples, ou seja, todos os conjuntos de objetos abstratos são unitários. Além disso, cada objeto detalhe é classificado por um objeto abstrato que carrega regras que o moldam. Desse modo, um objeto detalhe é sempre do tipo de seu objeto abstrato, caracterizando a *hierarquia de abstração de classificação* como sendo uma *hierarquia*

de tipos.

O formalismo apresentado até aqui cobre explicitamente as ocorrências de abstrações na base. A semântica que envolve a particularização de um tipo de abstração e os construtores semânticos está suportada pelas regras que constituem os *Critérios* aplicados em cada ocorrência.

Nas próximas seções serão apresentadas as abstrações de associação e de generalização. É interessante observar que todas as abstrações são *classificadas* no nível 0. Assim, o nível 0 é “uniforme” para todas as abstrações, com o objeto mais abstrato sendo o objeto *Abstração*. Dessa forma o nível 0 descrito como $A_{Class,0} = \langle A_{Class}, Co_{d,Class,0} = \{ "Abstração de Classificação", "Abstração de Generalização", "Base de Dados", "Conceito Objeto", \dots \}, Co_{a,Class,0} = \{ "Abstração" \}, Critério(0, A_{Class}) \rangle$ é o mesmo para todas as abstrações. Entretanto, o elemento máximo das hierarquias das abstrações é particular a cada uma, ou seja, cada uma tem o seu elemento máximo. O elemento máximo da hierarquia de Classificação é o próprio objeto “*Abstração*”, o elemento máximo da hierarquia de generalização é o objeto “*Conceito Objeto*”, e o elemento máximo da hierarquia de associação é o objeto “*Base de Dados*”.

3.4. Abstração de Associação

Nessa seção é definida a *Abstração de Associação* a qual será denotada por $A_{Assoc} = \langle "Associação", S_{Assoc}, R_{Assoc} \rangle$.

A *abstração de associação*, mesmo que não seja explicitamente referenciada e reconhecida, está presente em todos os modelos de dados sendo responsável por estabelecer “ligações” entre objetos. *Associar* significa estabelecer uma “ligação”, seja entre atributos e objetos, entre objetos estabelecendo relacionamentos, entre objetos

Figura 3.13: Exemplos de Ocorrência Abstração de Associação.

estabelecendo ordens ou papéis, valores de objetos a valores de atributos, valores de objetos a outros valores de objetos, etc.

Ao contrário do que ocorre com a abstração de classificação, a forma como uma *abstração de associação* é tratada em modelos que suportam *Meta-Sistemas de Gerenciamento de Dados* não difere quando comparada a modelos comuns (que não suportam Meta-sistemas).

3.4.1. Descrição da *Abstração de Associação*

Na **Figura 3.13** são apresentados três exemplos de ocorrências de abstração de associação. Em **a)**, os dois tipos de objetos *NomeP* e *Idade* estão associados ao tipo de objeto *Pessoa*. O objeto *Pessoa* é o objeto mais abstrato, enquanto *NomeP* e *Idade* são os objetos detalhe. Semanticamente, o **Critério** usado é o que *NomeP* e *Idade* desempenham o “papel” de atribuir características ao tipo *Pessoa*. Nos modelos de dados em geral, esses objetos freqüentemente são chamados atributos, sendo esse tipo de associação denominado *agregação*. Uma característica importante dessa associação é que os objetos detalhe são associados sem que essa associação resulte na criação de um novo objeto. A estrutura que envolve esse tipo de associação pode estabelecer:

- que objetos associem-se via objetos denominados atributos, caracterizando relacionamentos entre os objetos;
- que os objetos denominados atributos sejam estruturados em forma de listas, ou vetores, ou conjuntos, ou simplesmente tuplas. Nesse caso, tem-se uma especialização que é comumente denominada em modelos de dados, *abstração de agregação*.

Em **b)**, os tipos de objeto *Salas* e *Corredores* estão associados ao tipo de objeto *Prédio*. O tipo de objeto *Prédio* é o objeto abstrato, e os tipos de objeto *Salas* e *Corredores* são os objetos detalhe. Semanticamente, o **Critério** usado é o de que esses objetos detalhe compõem o objeto abstrato. Dessa forma os objetos detalhe estabelecem uma associação denominada *composição*. Através dessa *composição*, os objetos estabelecem uma *estrutura* independente dos papéis particulares que exercem, ou seja, não importa o papel que uma determinada *sala* exerce dentro do *prédio*, mas sim que a estrutura *sala* é importante para a existência do *prédio*. Nesse tipo de associação, os

objetos detalhe são associados resultando na criação de um novo objeto. Assim, tem-se uma especialização de associação que é comumente denominada em modelos de dados como *abstração de composição*. Em c), os tipos de objetos associados são *Capacidade-f* e *Capacidade-n*. Nesse caso, o tipo de objeto *Capacidade-f* é o objeto abstrato, e o tipo de objeto *Capacidade-n* é o objeto detalhe. Semanticamente, o *Critério* está estabelecendo uma associação entre os dados da capacidade de carga de todos os navios, e os sumarizando através de uma operação aritmética no tipo de objeto *capacidade-f*. São efetuadas operações aritméticas sobre os objetos detalhe, com o resultado sendo associado ao objeto abstrato. Assim, os objetos abstratos *sumarizam* informações provenientes dos objetos detalhe. Essa associação é comumente denominada *abstração de sumarização*.

Note que o *Critério* estabelece a semântica como a associação é realizada, ou seja, os "*critérios*" indicam as várias particularizações das *Abstrações de Associações*.

A abstração de associação pode também ser formalmente mostrada com suas ocorrências sendo colocadas em níveis. Na abstração de associação é identificado um número variável de níveis dependendo do modelo de dados em questão. Genericamente, a abstração pode ser colocada como segue, com seus índices (indicando os níveis de ocorrência) variando conforme o modelo de dados utilizado e a modelagem efetuada.

Assim, a abstração de associação pode ser genericamente escrita como $A_{Assoc,i} = \langle A_{Assoc}, Co_{d,Assoc,i} = \{ob_j, j=j', \dots, Assocdl\}, Co_{a,Assoc,i} = \{ob_k\}, Critério(i, A_{Assoc}) \rangle$, onde:

- $Co_{d,Assoc,i}$ é o conjunto não vazio de objetos detalhe dessa ocorrência da abstração A_{Assoc} ;
- $Co_{a,Assoc,i}$ é o conjunto não vazio e unitário de objetos abstratos dessa ocorrência da abstração A_{Assoc} ;
- $Critério(i, A_{Assoc})$ indica que o objeto detalhe está associado ao objeto abstrato.

Na **Figura 3.14** pode-se ver um exemplo (simplificado) dessas ocorrências.

No primeiro nível tem-se:

- $A_{Assoc,0} = \langle A_{Assoc}, Co_{d,Assoc,0} = \{ "Empresa", "Pessoa", "Oid1", "Oid2", "Professor", "Dentista", "Oid3", "Oid4" \}, Co_{a,Assoc,0} = \{ "Base de Dados" \}, Critério(0, A_{Assoc}) \rangle$.

No segundo nível tem-se:

- $A_{Assoc,1} = \langle A_{Assoc}, Co_{d,Assoc,1} = \{ "NomeP" \}, Co_{a,Assoc,1} = \{ "Pessoa" \}, Critério(1, A_{Assoc}) \rangle;$
- $A_{Assoc,2} = \langle A_{Assoc}, Co_{d,Assoc,2} = \{ "NomeE" \}, Co_{a,Assoc,2} = \{ "Empresa" \}, Critério(2, A_{Assoc}) \rangle;$
- $A_{Assoc,3} = \langle A_{Assoc}, Co_{d,Assoc,3} = \{ "José" \}, Co_{a,Assoc,3} = \{ "Old1" \}, Critério(3, A_{Assoc}) \rangle;$
- $A_{Assoc,4} = \langle A_{Assoc}, Co_{d,Assoc,4} = \{ "C\&T" \}, Co_{a,Assoc,4} = \{ "Old2" \}, Critério(4, A_{Assoc}) \rangle;$

Note que no primeiro nível tem-se $A_{Assoc,0} = \langle A_{Assoc}, Co_{d,Assoc,0} = \{ ob_j, j=j', \dots, assocdo \}, Co_{a,Assoc,0} = \{ Ob_{Assoc}, S_{Assoc} = "Base de Dados" \}, Critério(0, A_{Assoc}) \rangle.$

O objeto máximo da abstração de associação é o objeto *Base de Dados*, sob o qual todos os outros objetos estarão direta ou indiretamente associados. Note que o objeto máximo da abstração de classificação é o objeto *Abstração*, o qual está associado ao objeto *Base de Dados* e ao mesmo tempo o classifica.

3.4.2. Estrutura da Abstração de Associação em Objetos

Como descrito na seção 3.3.2., o conjunto de dados que compõe os atributos de estrutura de abstrações em cada objeto $ob_j \in CO(\lambda)$ de uma base de dados $BD = \langle CO(\lambda), M \rangle$ é reconhecido como $m_j = \langle M \rangle$. Para a *abstração de associação*, os dados que compõem M em cada objeto devem ser capazes de representar o Conjunto-PO

$[CO(\lambda), \succ]$. Assim, $M = Estr_{Assoc}$, onde a estrutura $Estr_{Assoc}$ contém os objetos que estão sendo associados e a *regra* dessa associação. Essa *regra* faz parte dos modelos de dados diferindo de modelo para modelo.

Figura 3.14: - Exemplo de Ocorrência de Abstração de Associação.

3.4.3. Interpretação

As *ocorrências de abstrações de associação* são de grande importância para a constituição da semântica que se deseja conseguir através de

uma modelagem. As abstrações de associação podem ser especializadas em outras, como *abstrações de agregação*, *abstrações de composição*, etc.

No modelo de dados SIRIUS apresentado no próximo capítulo deste trabalho serão utilizadas as especializações denominadas *abstração de agregação* e *abstração de composição*, não sendo inserida no modelo a *abstração de sumarização*.

Assim como todos os objetos que estão armazenados numa base de dados participam de pelo menos uma ocorrência de abstração de classificação, participam também de pelo menos uma ocorrência de abstração de associação. Dessa forma, para qualquer objeto $ob_j \in CO(\lambda)$, $\exists ob_k$ tal que $ob_j \underset{Assoc}{>} ob_k$ ou $ob_k \underset{A_{Assoc}}{>} ob_j$.

3.5. Abstração de Generalização

Nessa seção será definida *Abstração de Generalização*, a qual será denotada por $A_{Gener} = \langle "Generalização", S_{Gener}, R_{Gener} \rangle$.

A *abstração de generalização* está sempre presente em modelos de dados orientados a objetos. *Generalizar* significa abstrair em dados mais genéricos possíveis, os detalhes das informações ou características contidas em vários objetos. As ocorrências de abstrações de generalização são utilizadas quando objetos possuem dados adicionais que fazem com que estes “adquiram” relacionamentos ou “exerçam” algum comportamento específico.

Ao contrário do que ocorre com a abstração de classificação, e igualmente como acontece com a *abstração de associação*, a forma como uma *abstração de generalização* é tratada em modelos que suportam *Meta-Sistemas de Gerenciamento de Dados*, não difere quando comparada a modelos comuns (que não suportam Meta-sistemas).

3.5.1. Descrição da *Abstração de Generalização*.

A *abstração de generalização* consiste no conceito de que um objeto pode ser generalizado em um ou mais objetos. O objeto generalizado possui todas as características genéricas que são comuns aos objetos detalhe a ele ligado. Na **Figura 3.15**

está ilustrado um exemplo de ocorrência de abstração de generalização. O objeto *Pessoa* está sendo generalizado a partir dos objetos *Professor*, *Médico* e *Dentista*. Dessa forma, o objeto *Pessoa* é o objeto abstrato, enquanto os objetos *Professor*, *Médico* e *Dentista* são objetos detalhados a partir de *Pessoa*. Os objetos detalhe contêm informações especializadas que definem uma *Pessoa* como sendo *Professor*, ou *Médico* ou *Dentista*.

A *abstração de generalização* pode ser descrita utilizando a mesma estrutura de níveis em que foi apresentada a *abstração de classificação*. Assim como as abstrações de classificação e de associação, a abstração de generalização também não tem um número de níveis pré-definidos. Genericamente, a abstração pode ser colocada como segue, com seus índices (indicando os níveis de ocorrência) variando conforme o modelo de dados utilizado e a modelagem efetuada. Assim, a abstração de generalização pode ser genericamente

Figura 3.15: - Exemplo de abstração de Generalização.

escrita como $A_{Gener,i} = \langle A_{Gener}, Co_{d,Gener,i} = \{ob_p, j=j', \dots, Generdi\}, Co_{a,Gener,i} = \{Ob_{Gener} S_{Gener} = "Conceito Objeto"} \rangle$, Critério(i, A_{Gener}) $\rangle$, onde $1 \leq i \leq Generk$. Considere o exemplo apresentado na **Figura 3.16**. No primeiro nível, tem-se $A_{Gener,0} = \langle A_{Gener}, Co_{d,Gener,0} = \{"Pessoa", "Empresa"\}, Co_{a,Gener,0} = \{"Conceito Objeto"} \rangle$, Critério($0, A_{Gener}$) $\rangle$.

No segundo nível tem-se:

• $A_{gener,2} = \langle A_{Gener}, Co_{d,Gener,2} = \{"Professor", "Médico", "Dentista"\}, Co_{a,Gener,2} = \{"Pessoa"\}, Critério(2, A_{Gener}) \rangle$.

• $A_{gener,3} = \langle A_{Gener}, Co_{d,Gener,3} = \{"Empresa Nacional", "Empresa Estrangeira"\}, Co_{a,Gener,3} = \{"Empresa"\}, Critério(3, A_{Gener}) \rangle$.

Outras propriedades importantes devem ser identificadas nas ocorrências da abstração de generalização [Navathe-92]. Uma delas refere-se ao fato de que quando um objeto abstrato é instanciado, dependendo da definição do modelo sua instância deve ou não pertencer às instâncias dos seus objetos detalhe. Quando a instância do objeto deve obrigatoriamente pertencer, então se diz que há *relacionamento com participação*

total entre os objetos detalhe e os abstratos. Se não há obrigatoriedade, diz-se que há *relacionamento com participação parcial* entre os objetos detalhe e os abstratos. Na **Figura 3.16** tem-se a ocorrência $A_{Gener,3} = \langle A_{Gener}, Co_{d,Gener,3} = \{ "Empresa Nacional", "Empresa Estrangeira" \}, Co_{a,Gener,3} = \{ "Empresa" \}, Critério(4, A_{Gener}) \rangle$. Todas as instâncias do objeto *Empresa* tem que obrigatoriamente ser ou uma *Empresa Nacional* ou uma *Empresa Estrangeira*. Assim, diz-se que há *relacionamento com participação total* entre os objetos *Empresa*, *Empresa Nacional* e *Empresa Estrangeira*. Outra propriedade refere-se ao fato de que quando um objeto é detalhado em dois ou mais objetos, uma instância do objeto abstrato pode ser ou não de mais de um dos tipos dos objetos detalhados. No caso de o modelo exigir que a instância seja de exatamente um dos tipos de objeto detalhe, diz-se que há *restrição de disjunção*, caso contrário, que há *sobreposição (overlap)*. Por exemplo, na **Figura 3.16** tem-se a ocorrência

$A_{Gener,2} = \langle A_{Gener}, Co_{d,Gener,2} = \{ "Professor", "Médico", "Dentista" \}, Co_{a,Gener,2} = \{ "Pessoa" \}, Critério(2, A_{Gener}) \rangle$. Suponha que ocorra uma instância do objeto *Pessoa*. Dependendo do modelo, essa instância pode pertencer a um ou mais objetos detalhe, ou seja, a *pessoa* pode ser ao mesmo tempo um *professor* e um *médico*, ou *professor* e *dentista*, ou um *médico* e *dentista*. Uma outra

Figura 3.16: Ocorrência de um Tipo de Abstração de Generalização.

propriedade refere-se ao fato da possibilidade de se admitir que nem toda instância de um objeto detalhe precisar participar de uma ocorrência de abstração de generalização. Quando um modelo permite essa restrição, há a ocorrência de uma outra abstração que é denominada *abstração de categorização*. Com a inclusão dessa propriedade na abstração de generalização tem-se uma abstração mais genérica englobando em apenas uma abstração, as abstrações de generalização e de categorização. Um exemplo pode ser

visto na **Figura 3.17**. O objeto abstrato dessa ocorrência é *acionista* e os objetos detalhes são *Pessoa*, *Banco* e *Empresa*. Podem ser instanciados valores para *Pessoa*, *Banco* e *Empresa*, que não pertencem à categoria de *Acionista*. Assim, tem-se uma *abstração de generalização* “mais genérica”, ou seja, que engloba a *abstração de categorização* e a *abstração de generalização* comumente encontrada na literatura.

Figura 3.17: Exemplo de ocorrência da propriedade de categorização.

O objeto máximo da *abstração de generalização* é o objeto “*conceito objeto*”, que direta ou indiretamente, generaliza todos os objetos da base. “*Conceito Objeto*” generaliza os objetos “*abstração*” e “*base de dados*”; é classificado pelo objeto “*abstração*” e está associado ao objeto “*base de dados*”.

As propriedades da abstração de Generalização estão semanticamente representadas pelos **Critérios** estabelecidos em cada ocorrência.

3.5.2. Estrutura da Abstração de Generalização em Objetos

Como descrito na seção 3.3.2, o conjunto de dados que compõe os atributos de estrutura de abstrações em cada objeto $ob_j \in CO(\lambda)$ de uma base de dados $BD = \langle CO(\lambda), M \rangle$ é reconhecido como $m_j = \{ \langle M \rangle \}$. Para a *abstração de generalização*, os dados que compõem M em cada objeto devem ser capazes de representar o Conjunto-PO $[CO(\lambda), A_{Gener}]$. Assim, $M = Estr_{Gener}$, onde a estrutura $Estr_{Gener}$ contém os objetos

envolvidos nas ocorrências das abstrações de generalização, juntamente com as restrições que deve obedecer.

Definição 12: $\forall ob_j \in CO(\lambda)$, $ob_j = \langle id_j, m_j \rangle$, onde $m_j = Estr_{Gener}$ com $Estr_{Gener} = \{ ob_i, Regra \}$. Os objetos ob_i são os objetos envolvidos na generalização e regra indica como a generalização deve ser feita.

3.5.3. Interpretação

Assim como todos os objetos que estão armazenados numa base de dados participam de pelo menos uma ocorrência de abstração de classificação e de associação, participam também de pelo menos uma ocorrência de abstração de generalização. Dessa forma, para qualquer objeto $ob_j \in CO(\lambda)$, $\exists ob_k$ tal que

$$ob_j \underset{A_{Gener}}{>} ob_k \text{ ou } ob_k \underset{A_{Gener}}{>} ob_j.$$

Como em qualquer abstração, o número máximo de elementos permitidos nos conjuntos $Co_{a,Gener,i}$ é 1. Porém, um mesmo objeto ob_j pode participar de mais de uma ocorrência de abstração de generalização, tendo assim mais de um objeto como objeto abstrato. Nesse caso, existiriam dois objetos ob_k e ob_l , tal que $ob_k \in Co_{a,k,Gener}$ implicando que $ob_j \in Co_{a,l,Gener}$ e $ob_k \underset{A_{Gener}}{>} ob_j$ e $ob_l \underset{A_{Gener}}{>} ob_j$. Dessa forma, o Conjunto-PO

$[CO(\lambda), \underset{A_{Gener}}{>}]$ constitui uma hierarquia não estrita.

O objeto máximo da hierarquia de generalização é o “conceito objeto” $Ob_{1Gener} \mathcal{S}_{Gener}$. Esse aspecto corresponde ao fato de que tudo na base de dados são objetos, ou seja, todos são generalizados no objeto “conceito objeto”. Logo, uma ocorrência da abstração de generalização $A_{Gener,0}$ sempre terá o conjunto $Co_{a,Gener,0}$ com apenas o objeto $Ob_{1Gener} \mathcal{S}_{Gener}$, o qual é reconhecido pela própria abstração, ou seja, $Ob_{1Gener} \mathcal{S}_{Gener} \in \mathcal{S}_{Gener}$ e é o objeto máximo do Conjunto-PO $[CO(\lambda), \underset{A_{Gener}}{>}]$. Note que

o objeto $Ob_{1Gener} \mathcal{S}_{Gener}$ é o próprio objeto “Conceito Objeto”.

3.6. Cruzamento das hierarquias de abstração

Na **Figura 3.18** estão ilustrados três exemplos de ocorrências de abstração. Em a) tem-se uma ocorrência de abstração de classificação. Note que o objeto máximo da hierarquia formada é o objeto “*abstração*”, e que todos os outros estão direta ou indiretamente classificados por ele. Os objetos *Oid1* e *Oid2* são classificados pelo objeto *Pessoa*; o objeto *Oid3* é classificado pelo objeto *Médico* e o objeto *Oid4* é classificado pelo objeto *Professor*; os objetos *Idade* e *Nome* são classificados pelo objeto *Atributo*; os objetos *28* e *25* são classificados pelo objeto *Idade*; os objetos *José* e *Carla* são classificados pelo objeto *Nome*. “Subindo um nível” na hierarquia, tem-

Figura 3.18: Exemplo de ocorrências da abstração de classificação (a), da abstração de associação (b) e da abstração de generalização (c).

se os objetos *Pessoa*, *Médico* e *Professor* sendo classificados pelo objeto *Objeto*; os objetos *Idade* e *Nome* são classificados pelo objeto *Atributo*. Subindo outro nível, tem-se que os objetos *Objeto* e *Atributo* são classificados pelo objeto *Abstração de Classificação*. E no primeiro nível, tem-se que os objetos *Base de Dados*, *Conceito Objeto* e *Abstração de Classificação* são classificados pelo objeto *Abstração* (o objeto

máximo na hierarquia de classificação).

Em b) tem-se uma ocorrência de abstração de associação. O objeto máximo da hierarquia formada é o objeto “base de dados”, e todos os outros estão direta ou indiretamente associados a ele. Os objetos *Idade* e *Nome* associam-se ao objeto *Pessoa*; os objetos *José* e *28* associam-se ao objeto *Oid1* e os objetos *Carla* e *25* associam-se ao objeto *Oid2*. Subindo um nível na hierarquia, tem-se que os objetos *Conceito Objeto*, *Pessoa*, *Médico*, *Professor*, *Oid1*, *Oid2*, *Oid3*, *Oid4*, *Atributo*, *Objeto*, *Abstração de Classificação* e *Abstração* estão associados ao objeto *Bases de Dados* (o objeto máximo na hierarquia de associação).

Em c) tem-se uma ocorrência de abstração de generalização. O objeto máximo da hierarquia formada é o objeto “conceito objeto”, e todos os outros são direta ou indiretamente generalizados por ele. Os objetos *Médico* e *Professor* são

Figura 3.19: Ilustração de como os objetos ocupam lugares diferentes em diferentes abstrações.

generalizados pelo objeto *Pessoa*; o objeto *Oid3* é generalizado pelo objeto *Oid1* e o objeto *Oid4* é generalizado pelo objeto *Oid2*. Subindo um nível na hierarquia, tem-se que os objetos *Base de Dados*, *Pessoa*, *Oid1*, *Oid2*, *José*, *28*, *Carla*, *25*, *Atributo*, *Objeto*, *Abstração de Classificação* e *Abstração* são generalizados pelo objeto *Conceito Objeto* (o objeto máximo na hierarquia de generalização).

Por esse exemplo, pode-se ver que todos os objetos da base participam de todas as abstrações. Além disso, cada objeto ocupa um “lugar” diferente em cada abstração. A **Figura 3.19** ilustra esse aspecto. Em a) são mostradas as diferentes ocorrências das abstrações. Setas pontilhadas indicam abstração de associação, setas tracejadas indicam

abstração de generalização, e setas contínuas indicam abstração de classificação. Em b), os objetos participantes são replicados, dando a idéia de seus diferentes comportamentos segundo cada abstração em que está participando.

A semântica resultante de uma modelagem é obtida pelas ocorrências de todas as abstrações, ou seja, elas se completam. Ao mesmo tempo, pode-se ver que elas são ortogonais entre si.

3.7. Conclusão

Este capítulo mostra que um modelo de dados somente pode ser construído a partir de duas abstrações fundamentais: a de *Classificação* e a de *Associação*. A *Abstração de Classificação* é fundamental para permitir a criação de objetos que atendam a determinadas regras que caracterizam seu tipo. A *Abstração de Associação* é fundamental para permitir que objetos distintos possam associar-se em outros, permitindo que conceitos já existentes possam ser aproveitados para a criação de outros - em particular para permitir a definição de um tipo.

Em razão dos trabalhos que avaliam o embasamento teórico dos modelos orientados a objetos, considera-se que a *Abstração de Generalização* é também fundamental na definição desses modelos. Em particular, mostra-se que essas três abstrações são independentes entre si, e que cada uma é uma instanciación do próprio conceito de abstração.

Este capítulo também introduz o conceito de *parametrização de abstrações*, como uma forma de estabelecerem-se restrições que particularizam uma determinada abstração, para que uma ou mais possam ser associadas para compor os construtores semânticos de um modelo de dados.

As abstrações de associação e de generalização têm suas ocorrências formando hierarquias não estritas, enquanto a abstração de classificação tem suas ocorrências formando uma hierarquia simples.. Assim, um objeto detalhe sempre terá apenas um objeto como objeto abstrato. Isso é válido para uma mesma ocorrência, entretanto um mesmo objeto pode participar de várias ocorrências de uma mesma abstração. Logo, $\forall ob_j \in CO(\lambda), \exists! ob_x$ tal que $ob_x \succ_{A_\lambda} ob_j$, entretanto pode existir um outro (único)

objeto ob_i , tal que $ob_i >_{A_{tj+1}} ob_j$.

Além dessa característica, qualquer objeto pertencente a uma base de dados participa de todas as hierarquias de abstração. Assim, um objeto $ob_j \in CO(\lambda)$, participa de uma hierarquia de classificação, de uma hierarquia de generalização, de uma hierarquia de agregação e de uma hierarquia de composição.

As abstrações foram parametrizadas formalmente constituindo um metamodelo que pode ser utilizado para a construção de outros modelos de dados segundo um padrão definido. Os modelos criados a partir do metamodelo possuem seus elementos identificados através das abstrações que o compõem.

No próximo capítulo, o metamodelo é aplicado na construção de um modelo de dados orientado a objetos.

Capítulo 4

SIRIUS - Modelo de Dados Orientado a Objetos e Baseado em Abstrações de Dados

4.1.Introdução

No capítulo anterior foi apresentada a conceituação que caracteriza a utilização de *abstrações de dados* como elementos construtivos de modelos de dados, juntamente com seu formalismo. Esses elementos agrupados constituem um *metamodelo*, que permite a representação (criação) de *Modelos de Dados Orientados a Objetos*. Neste capítulo é apresentado um modelo de dados orientado a objetos denominado *SIRIUS*, o qual foi construído segundo os conceitos propostos no capítulo anterior para atender às necessidades de um ambiente de projetos de engenharia [Biajiz-96b]. O modelo foi desenvolvido para suportar tais necessidades através de construtores semânticos construídos utilizando as

Figura 4.1: Abstrações do SIRIUS.

três abstrações fundamentais: *abstração de classificação*, *abstração de associação* e

abstração de generalização. A abstração de associação ocorre especializada em duas, que são as *abstrações de composição* e *de agregação*. Na **Figura 4.1** estão relacionadas as abstrações suportadas por SIRIUS. A figura pode representar o modelo, uma vez que os elementos do modelo são “organizados” semântica e sintaticamente segundo essas abstrações.

Considerando o formalismo apresentado no capítulo anterior, SIRIUS pode ser formalmente referenciado como sendo o conjunto de abstrações $SIRIUS = \{A_{Class}, A_{Gener}, A_{Assoc}\}$, onde $A_{Assoc} = \{A_{Agreg}, A_{Comp}\}$.

No decorrer deste capítulo, o modelo será apresentado por partes, seguindo as abstrações que suporta. Note-se que as abstrações do modelo foram determinadas por serem obrigatórias ou necessárias no contexto desejado. Esse aspecto será definido na apresentação de cada abstração com seus respectivos elementos nas próximas seções.

4.2. Necessidades de Ambientes de Projeto para Engenharia

Aplicações comerciais caracterizam-se pela manipulação de dados que podem ser organizados em um conjunto relativamente pequeno de estruturas, porém com um volume muito grande de dados estruturalmente idênticos. Por outro lado, aplicações não comerciais dos sistemas de computação manipulam dados que não apresentam essa mesma homogeneidade estrutural. Neste caso, encontram-se os sistemas de Projeto Apoiado por Computador - PAC (CAD) [Ishikawa-93] [Kempe-95], Fabricação Apoiada por Computador - FAC (CAM), Sistemas de Automação de Escritório, Inteligência Artificial, etc. As aplicações que se desenvolvem nessas áreas, em geral aproveitam os SGBDs que já existem, mesmo que estes não sejam adequados para essas aplicações. Além disso, é muito freqüente que duas ou mais aplicações não convencionais, desenvolvidas de maneira independente, acabem tendo que ser integradas em um único sistema. Por exemplo, isso ocorre quando um sistema de Projeto Apoiado por Computador deve ser integrado a um sistema de Fabricação Apoiado por Computador - PAC/FAC ou Sistemas de Fabricação Integrada (CIM). Nesses casos, a falta de um núcleo de gerenciamento de dados comum traz problemas de compatibilidade entre os sistemas, sendo necessário então a criação de programas de adaptação complexos, que em geral impedem a utilização plena das capacidades de pelo

menos um dos sistemas integrados [Su-86]. Levantamentos de suas necessidades têm sido feitos e apresentados, tal como em [Hartzband-85]. Assim, têm sido pesquisados modelos para SGBDs que possam dar o suporte necessário a um amplo espectro de aplicações não convencionais [Abiteboul-87][Hsu-93] [Hudson-89] [Kim-89a] [Naja-95][Ruiz-94] [Ruiz-95] [Weiser-89]. Esses modelos têm sido denominados *Modelos de Dados para Engenharia*.

Os modelos de dados para engenharia têm como objetivo suprir requisitos para a modelagem de sistemas de apoio a projetos de engenharia, de sistemas de apoio à decisão e sistemas de manipulação de dados científicos, ou em resumo, sistemas não convencionais [Chorafas-93]. Abaixo são listadas algumas dessas necessidades:

- a construção de uma base de dados constitui parte do projeto em si, sendo que o esquema da base de dados evolui junto com a evolução do projeto;
- não se conhece em tempo de projeto todos os tipos de objetos e/ou estruturas que serão tratados pelo sistema em tempo de execução e, portanto, deve ser possível a criação dinâmica de tipos pelo sistema já em execução;
- a estrutura do sistema é muito complexa, e apenas parcialmente podem ser representados nos sistemas comerciais disponíveis, ou seja, o modelo relacional não é suficiente para representar as necessidades desses sistemas, e assim, conceitos de orientação a objetos devem ser empregados;
- objetos de uma mesma classe (ou tipo) têm freqüentemente exceções, tanto de estrutura quanto de comportamento, ou seja, instâncias de objetos devem suportar atributos e métodos não previstos nos esquemas;
- devem ser suportadas estruturas de dados variadas, tais como vetores e listas;
- *aplicações hipermídia* requerem objetos complexos tais como hierarquias de componentes e textos estruturados. Embora sistemas de bases de dados convencionais simulem objetos complexos por união de relações (“joins”), eles impõem esforços desnecessários ao usuário com respeito à modelagem, e sobrecarga de performance. As aplicações necessitam representar objetos complexos diretamente;
- *sistemas hipermídia* armazenam e recuperam imagens e textos. As aplicações precisam ser capazes de armazenar e manipular integradamente dados multimídia (imagens, gráficos, e textos), em adição a números e strings. Os programas de aplicações precisam

permitir ao usuário manipular diretamente dados multimídia;

- *sistemas CAD* usam características dinâmicas de componentes como conhecimento comportamental, o qual sistemas de bases de dados relacionais não podem manipular. Sistemas de aplicações precisam ser capazes de gerenciar tal conhecimento comportamental explicitamente. Em adição, sistemas de aplicação precisam ser flexíveis para desenvolverem bases de dados ativas com consistência de manutenção pelo uso de gatilhos;
- *sistemas de aplicação* são evolucionários. Sistemas hipermídia podem requerer a incorporação de novos meios tais como audio e dados geométricos. O usuário precisa ser capaz de definir flexivelmente novos tipos e operações associadas, e novos métodos de acesso. A inclusão de novos tipos, operadores, e métodos de acesso precisa ser feita com a menor modificação possível. Sistemas de bases de dados convencionais não podem manipular isto.

4.3. Conceitos

Nessa seção são apresentados os conceitos de objetos, atributos, características de atributos, tipos de atributos e tipos de objetos. Esses conceitos constituem os elementos do modelo, e são estruturados segundo as abstrações que o modelo suporta.

4.3.1. Objetos e Tipos de Objetos

CONCEITO: Objetos e tipos de objetos

Motivação

Caracterização

Notação

Formalismo

Motivação

Objetos representam elementos que constituem conceitos ou entidades do mundo real. Um objeto carrega consigo a representação das características estruturais e do comportamento do que está representando. Pode-se afirmar que todos os elementos da

natureza podem ser representados através de objetos. Logo, uma modelagem com objetos estabelece uma uniformidade de visão e de tratamento, útil tanto para projetos quanto para implementação. Dessa forma, são criados e utilizados modelos de dados totalmente orientados a objetos, com os quais se geram modelagens orientadas a objetos e, além disso, toda uma técnica de análise e implementação voltada para objetos.

Caracterização

Em uma primeira abordagem, o modelo trata como objetos os conceitos do mundo real que podem ser identificados univocamente pelo usuário. Assim, o termo “objeto” é usado no contexto do SIRIUS de uma maneira mais restrita do que a empregada no capítulo anterior, uma vez que apenas os objetos identificados pelo usuário como tal, são chamados objetos. Atributos, tipos e demais conceitos do modelo, ao contrário

Figura 4.2: Representação de um Objeto.

da terminologia empregada no capítulo anterior, não são aqui chamados objetos. Dessa forma, em SIRIUS o termo “objeto” é utilizado diferenciadamente em três situações:

- 1) quando o usuário reconhece um elemento do mundo real, representando os conceitos ou entidades do mundo real que podem ser identificados por ele. Nesse caso, o termo “objeto” é utilizado sem um qualificador;
- 2) quando o próprio sistema manipula os objetos e estes apresentam uma característica mais voltada à implementação, embora percebidos pelo usuário. O termo utilizado é “objeto primário”¹. Pode-se considerar *objetos primários* aqueles que não são transparentes ao usuário, mas que são por eles identificados através de abstrações com outros objetos.

¹O termo objeto primário será melhor abordado na seção 4.4.

3) o “objeto” genérico ob_i usado no sentido mais amplo como adotado no capítulo anterior, engloba entre outros, os objetos primários e os objetos definidos pelo usuário, e é definido aqui como “objeto conceito”. Esse aspecto pode ser visto na **Figura 4.2 a)**, onde o objeto conceito está representado englobando os objetos “objeto primário” e “objeto” (objeto definido pelo usuário).

A partir daqui, todas as vezes que for referenciado um objeto apenas como “objeto”, estará sendo referenciado “objeto definido pelo usuário”.

Um objeto será representado no modelo pelo seu nome, pelo nome do seu tipo, e por atributos a ele associados. Além disso, considera-se que todo objeto tem associado um identificador interno denominado *Old* gerenciado pelo sistema, e que identifica univocamente o objeto durante toda a sua existência na base de dados.

Os atributos têm o propósito de caracterizar a estrutura e o comportamento do objeto (vide conceito Abstração de Agregação). Tanto os atributos quanto o nome dos objetos são opcionais. Deve-se observar que todo objeto tem exatamente um tipo, o qual por sua vez como qualquer conceito, é também um objeto. Não se impõe nenhuma restrição a quais objetos podem ou não ser tipos (vide conceito: Abstração de Classificação.)

Notação

A representação gráfica de um objeto no modelo é mostrada na **Figura 4.2**. Em b) está representado um objeto através de um retângulo dividido horizontalmente por uma reta formando 2 partes. Na parte superior à esquerda do retângulo é escrito o nome do tipo deste objeto, enquanto na parte inferior é escrito o nome do objeto. A parte direita da linha superior é utilizada para dados da Abstração de Composição, a qual será descrita à frente. Note-se que embora esteja ainda em branco na **Figura 4.2**, numa modelagem nunca deve ficar assim. Já o tipo de um objeto pode ficar em branco, quando seu tipo for o próprio metatipo *tipo-de-objeto* que é reconhecido e controlado pelo sistema. Em c) estão representados dois exemplos: um objeto *Empilhadeira* cujo tipo é *Máquina*, e um objeto *Máquina* cujo tipo é o metatipo *tipo de objeto*, que está omitido.

Formalismo

O formalismo do *objeto conceito* foi definido na seção 3.2, como uma tupla da forma $\langle id_j, m_j \rangle$ onde $m_j = \{ \langle M \rangle \}$. O "objeto" definido pelo usuário no contexto do modelo SIRIUS é formalizado da mesma maneira. Assim, objetos em SIRIUS são formalizados como $ob_j = \langle id_j, m_j \rangle$, onde:

- id_j é o identificador interno do objeto (*Old*) atribuído univocamente para esse objeto pelo gerenciador da base;
- m_j é o conjunto de dados que compõe os "atributos de estrutura", tratados pelo gerenciador para manter as informações sobre como esse objeto é inserido na organização formada em cada abstração A_i do modelo M que apoia a base de dados BD ; como o modelo SIRIUS suporta as abstrações de Classificação, de Generalização, de Agregação e de Composição, então $m_j = \{Estr_{Class}, Estr_{Gener}, Estr_{Agreg}, Estr_{Comp}\}$, onde $Estr_{A_i}$ são os "atributos de estrutura" para cada abstração A_i do modelo.

Um objeto ob_j participa de ocorrências de todas as abstrações suportadas pelo modelo, sendo que para cada abstração, a estrutura para o objeto é diferente. Desse modo, cada estrutura $Estr_{A_i}$ que define um objeto será mostrada nas próximas seções no contexto da abstração a que pertence.

4.3.2. Atributos e Tipos de Atributos

CONCEITO: Atributos e tipos de atributos

Motivação**Caracterização****Notação****Formalismo****Motivação**

Existem dados no mundo real que estabelecem informações sobre os objetos e que, mesmo sendo também conceitos e, portanto objetos, não têm a propriedade de que cada

uma de suas ocorrências no mundo real devam ser univocamente identificadas pelo usuário, a não ser por suas associações com outros objetos. Esses dados são modelados aqui como valores de atributos. Através deles insere-se “comportamento” no objeto a que estão associados, caracterizando a sintaxe e a semântica do objeto na modelagem.

Caracterização

No modelo, os tipos de atributos desempenham o papel de caracterizar objetos. Eles são responsáveis pelo significado semântico e de comportamento do objeto a que estão associados, podendo ser instanciados através de valores em atributos. Um atributo possui um tipo e um valor, o qual por si só é um objeto, obedecendo a um domínio pré-definido para o tipo de atributo a que pertence. Além disso, através de seus valores define-se semanticamente como são as ocorrências da:

- abstração de classificação, na qual os valores dos atributos associados ao objeto “formatam” a instânciação desse objeto;
- abstração de generalização, na qual os valores de alguns dos atributos associados ao objeto indicam o critério para que o objeto seja especializado;
- abstração de associação, “ditando” como *objetos conceito* em geral associam-se para caracterizar outros objetos conceitos.

As ocorrências dessas abstrações serão explicadas em suas respectivas seções.

Os tipos de atributos são ainda “classificados” em monovalorados ou multivalorados. Um tipo de atributo monovalorado é aquele que possui apenas um valor para cada atributo instanciado a partir dele, e multivalorado, o que possui mais de um valor para cada instância. Os tipos multivalorados podem ser organizados segundo estruturas de dados, tais como vetores, conjuntos, ou listas.

Notação

A notação usada está representada na **Figura 4.3**. Os tipos de atributos são representados por setas seguidas ou superpostas pelos nomes de seus tipos. As setas são dispostas ortogonalmente a uma reta vertical que se origina no objeto do qual eles estão associados. Em **a)** está representada a indicação de um tipo de atributo monovalorado através de uma seta simples, e um tipo de atributo multivalorado através de uma seta dupla (com duas pontas). Em **b)** estão representados atributos instanciados com valores, ou seja, utiliza-se o nome do tipo do atributo, e o símbolo de dois-pontos seguido do valor atribuído. Em **c)** há um exemplo onde o objeto *Torno* possui o tipo de atributo não instanciado “*Compr. Barramento*”, os atributos “*Atividade*” instanciado com o valor “*Desbaste*”, e “*Alimentação*” instanciado com o valor “*Elétrica*”. Note em **c)**, que os atributos não instanciados são colocados acima dos objetos, e que os instanciados são colocados abaixo dos objetos. Essas características serão discutidas na seção 4.4.

Figura 4.3 - Notação de atributos.

Formalismo

Note que um tipo de atributo é um *objeto conceito*, sendo formalizado como $ta = \langle Idtat, mta \rangle$, onde:

- $Idtat$ (=OId) é o identificador do tipo do atributo;
- mta_j é o conjunto de dados que compõe os “atributos de estrutura”, tratados pelo gerenciador para manter as informações sobre como esse tipo de atributo é inserido na organização formada em cada abstração A_i em SIRIUS.

Da mesma forma como para objetos, $mta_j = \{Estr_{Class}, Estr_{Gener}, Estr_{Agreg},$

$Estr_{Comp}$ }, onde $Estr_{At}$ são os “atributos de estrutura” para cada abstração A_i do modelo. Assim, um tipo de atributo pode participar das ocorrências de todas as abstrações suportadas pelo modelo. Desse modo, todas as estruturas $Estr_{At}$ que definem um tipo de atributo serão mostradas nas próximas seções no contexto das abstrações a que pertencem.

Os tipos de atributos são instanciados através de valores, sendo então denominados atributos. Os atributos instanciados a partir de seus tipos de atributos são formalizados como $at_j = \langle Idat_j, nome, Valor, Idtob_j \text{ ou } Idta_j \rangle$, onde:

- $Idat_j$ é o identificador (código) do atributo instanciado;
- $nome$ é o identificador do tipo de atributo do qual é uma instância;
- $Valor$ indica o valor instanciado (atribuído pelo usuário) para o atributo;
- $Idtob_j$ é o identificador do tipo do objeto ao qual o atributo está associado no caso de estar associado a um objeto, ou $Idta_j$ é o tipo de atributo ao qual o atributo está associado no caso do atributo estar associado a um outro tipo de atributo.

O fato de um tipo de atributo poder estar associado a um tipo de objeto ou a um tipo de atributo será explorado na seção sobre abstração de agregação.

4.3.3. Características de Atributos

CONCEITO: Característica de Atributos

Motivação

Caracterização

Notação

Formalismo

Motivação

Modelos de dados em geral permitem associar atributos a objetos definindo-lhes um tipo de dado e um domínio. É comum encontrar como tipos de dados: números inteiros, números fracionários, cadeias de caracteres, etc. Supõe-se que o suporte de operações para atributos quaisquer possa ser oferecido de maneira homogênea. Por outro lado, considera-se que os métodos que determinam os aspectos dinâmicos dos objetos são tratados de maneira própria.

SIRIUS expande e ao mesmo tempo restringe esses conceitos. Restringe ao

generalizar (modelar) como atributos tudo o que se associa a objetos. Expande ao admitir que o suporte de operações oferecidas pelo modelo para atributos especializa-se em diversas formas, de acordo com a “característica” do atributo. Atributos usados para armazenar números ou pequenas cadeias são considerados “Propriedades”, ao passo que textos longos, com suporte para armazenagem e recuperação de linhas e parágrafos, são considerados “Textos” (“comentários”). Atributos que determinam o comportamento dos objetos frente a operações de manipulação de dados da base são considerados “Regras”, ao passo que o comportamento de objetos frente às operações que os envolvem no mundo real são considerados “procedimentos” (“métodos”).

Caracterização

Todo atributo possui um tipo e um conjunto não vazio de valores. A cada tipo de atributo estão associados uma característica e o tipo de dado que ele contém. Um tipo de dado pode ser um inteiro, real, byte, cadeia de caracteres, etc. A característica de um atributo de determinado tipo define o conjunto de operações que um Gerenciador de Dados pode efetuar sobre esse atributo (estabelecem o comportamento junto ao objeto ao qual esse atributo está associado). Em SIRIUS, as características são divididas em: **estáticas**, abrangendo comentário e propriedade; **dinâmicas**, abrangendo regra e procedimento; **de interface**, abrangendo visualização, som, partitura, imagem e gráfico; e **estruturais**, abrangendo estruturas de dados, tempo, tupla e relacionamento. Quando um atributo possui uma característica que não é previamente definida, diz-se que o atributo possui simplesmente **propriedade**.

O significado de cada característica é:

- **Propriedades** → são atributos usados para armazenar números ou pequenas cadeias de caracteres;
- **Comentário** → corresponde a campos longos, ou seja, informações para o usuário tratadas apenas de forma textual, que o SGBD permite manipular-se como linhas,

Tm = tempo
Ed = estrutura de dados
Pc = procedimento
Ct = comentário
Rg = regra
Vs = visualização
Im = imagem
Gr = gráfico
Pr = propriedade
So = som
Pa = partitura
Tp = tupla
Re = relacionamento

Figura 4.4: Símbolos utilizados nas características de atributos.

parágrafos, etc;

- **Regra** → são atributos que possuem uma condição e uma ação. A condição permite que a ação seja executada sempre que for feita alguma operação do SGBD sobre o objeto (gatilho), tal como consulta ou atualização, ou quando for recebida alguma mensagem pelo objeto, oriunda da execução de outra ação ou procedimento na base de dados;
- **Procedimento** → são atividades que devem ser exercidas de uma forma pré-determinada pelo sistema, solicitadas pelos aplicativos para execução imediata ou postergada;
- **Tempo** → indicação de tempo, absoluto ou relativo, atribuído aos objetos e atributos. Permite que se registre o tempo absoluto de operações na base, ou o tempo do mundo real (ou de validade) definido externamente à base. Podem existir diversas dimensões temporais;
- **Estrutura de Dados** → são agrupamentos de informações com um significado completo que normalmente são representados em estruturas de arranjos de dados (vetores e matrizes), ou para as quais possam ser definidas operações próprias de conjuntos, tais como união e intersecção, ou ordenação e tratamento de repetição de valores (listas, filas, pilhas e montes). Os dados estruturados são outros atributos;
- **Tupla** → indica que o atributo em questão agrega atributos com características diferentes em uma estrutura de tupla;
- **Relacionamento** → indica que o atributo em questão associa um objeto a outro ou a outros objetos;
- **Gráfico** → corresponde à informação vetorial de localização no espaço, podendo indicar um ponto, uma função, um conjunto de segmentos de retas, uma operação de transformação, ou qualquer combinação destes;
- **Imagens** → são atributos matriciais que representam figuras definidas através de valores associados a coordenadas cartesianas (“pixels”, “raster”, “quad-tree”, etc);
- **Som** → corresponde a sons digitalizados armazenados na base de dados;
- **Partitura** → corresponde à informação musical, que pode ser reproduzida por

síntese a partir dos dados armazenados;

- **Visualização** → são atributos que podem definir como um elemento de um modelo pode ser visualizado em uma interface de entrada e saída do sistema, para permitir a interação de uma pessoa com um aplicativo que manipule o objeto ou o relacionamento ao qual o atributo está agregado.

Qualquer atributo estatístico pode ser usado para identificar um objeto, embora usualmente propriedades prestem-se excepcionalmente bem para isso. Identificar um objeto significa que não podem existir dois objetos no mesmo contexto com o mesmo valor para esse atributo. Numa dada modelagem, alguns tipos de atributos são identificadores de objetos, ou seja, têm a propriedade adicional de identificar um objeto de maneira significativa para o usuário. Assim, durante uma instanciação, esse tipo torna-se um identificador do objeto instanciado. Havendo identificadores associados, deve-se indicar um deles para ser o identificador principal dos objetos desse tipo. Esse aspecto também é uma característica de atributo, porém ao invés de utilizar um símbolo distinto para essa característica, sua identificação é efetuada integrada a da abstração de classificação, que será vista na seção 4.5.

Usualmente em modelos de objetos, os objetos possuem uma parte estática (normalmente chamada de atributos do objeto) e uma parte dinâmica (normalmente chamada de métodos do objeto). Em SIRIUS o termo atributo designa qualquer elemento que se agrega a um objeto. Objetos por sua vez, possuem não duas, mas quatro partes: estática, dinâmica, estrutural e de interface. Todos os conceitos de SIRIUS que se referem a atributos aplicam-se igualmente a qualquer uma dessas partes.

Assim, além de apresentar aspectos estáticos e dinâmicos como de qualquer outro modelo, os objetos em SIRIUS possuem uma “aparência” (como são vistos externamente, tanto pelos demais componentes do sistema além do Gerenciador de Bases de Dados, quanto por usuários “humanos”), e uma “estrutura” (como seus demais

componentes organizam-se temporal ou espacialmente). Isso significa que a descrição do modelo, quando se refere a atributos de um objeto, está considerando tanto aspectos *estáticos* quanto *dinâmicos*, *estruturais* e de *interface*.

Notação

A notação utilizada para *característica* consiste em prefixar-se o nome do atributo no diagrama com o símbolo da característica precedendo-o. Os símbolos utilizados nas características estão indicados na **Figura 4.4**, e a forma como são utilizados está exemplificada na **Figura 4.5**.

Formalismo

A *característica* é anotada na definição do tipo do atributo, sendo que no SIRIUS, as possíveis características formam o conjunto $Característica = \{\text{tempo, estrutura de dados, procedimento, comentário, regra, visualização, imagem, gráfico, som, partitura, propriedade, relacionamento, tupla}\}$.

A indicação de um valor para um atributo depende de sua característica. Por exemplo, para propriedade a indicação é imediata, pois basta indicar o número do texto, letra, etc a ser atribuída ao atributo. Para comentários o valor é um texto longo, possivelmente com muitos parágrafos. Para regras (métodos) o valor é uma tupla com três elementos: *evento de disparo*, *condições de validação* e *corpo da regra*. Para estruturas o valor é a definição da estrutura, etc.

Figura 4.5: Exemplo de utilização do símbolo de característica.

4.4. Abstração de Agregação

CONCEITO: Abstração de Agregação

Motivação

Caracterização

Notação

Formalismo

Motivação

A caracterização de objetos através de atributos dá-se através de *ocorrências de abstração de agregação*. Atributos agregam-se para constituir o conjunto de dados que representa na modelagem os dados do objeto do mundo real, ou para constituir outros atributos. Atributos estabelecem também relacionamentos entre objetos, ou seja, dois ou mais objetos são agregados através de um objeto conceito formando relacionamentos.

Caracterização

A *abstração de agregação* caracteriza-se no modelo através: da associação de atributos entre si formando outros atributos que são chamados agregações de atributos, da agregação de atributos aos tipos de objetos e aos objetos instanciados, e na agregação de objetos entre si, formando relacionamentos aos quais se pode por sua vez associar-se também atributos. Considerando inicialmente a agregação de atributos, pode-se ver que ela ocorre de diversas maneiras associando atributos simples em tuplas, listas, vetores ou conjuntos sem repetição. Tuplas, vetores, listas e conjuntos são estruturas de dados organizadas a partir da agregação de atributos. As estruturas de vetores, listas e conjuntos modelam diretamente um atributo multivalorado.

A associação entre objetos pode acontecer de duas maneiras. A primeira, mais imediata (porém mais dependente do tratamento dado a ela pelos aplicativos), consiste em referenciar-se a partir de um dos objetos envolvidos, os demais objetos. Essa referência pode ter ou não uma referência equivalente (oposta no caso de apenas serem dois os objetos envolvidos) nos demais objetos. Em ambos os casos, a consistência dos dados que implementam essa maneira de representar a associação é de responsabilidade dos aplicativos, com pouco suporte podendo ser oferecido pelo gerenciador. Essa

maneira de indicar a associação entre objetos não acrescenta nenhum conceito novo ao modelo, correspondendo apenas ao fato de que o valor de um atributo pode ser outro objeto.

A segunda maneira que institui o conceito de relacionamento considera a associação entre objetos como um objeto primário (o objeto relacionamento), tratado internamente pelo próprio gerenciador. Nesse caso, os objetos envolvidos têm como atributo uma referência para o objeto primário relacionamento; o tipo do relacionamento é caracterizado pelo tipo de atributo; e a característica desse tipo de atributo é a característica “**relacionamento**”. Essa maneira de indicar-se a associação entre objetos traz ao modelo a capacidade de suportar relacionamentos de uma maneira semelhante, mas não igual à maneira como esse conceito é suportado em modelos semânticos.

Um atributo com característica de relacionamento também pode ser um atributo simples, uma tupla, uma lista, um conjunto. Em cada objeto há um atributo cujo valor indica o outro objeto como elemento do relacionamento. Note-se que não há restrições quanto a essa ligação, ou seja, um objeto pode estar agregado com vários outros objetos através de vários relacionamentos. Além disso, um relacionamento pode ter atributos associados, os quais são denominados atributos de relacionamento.

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.

Notação

A notação utilizada na agregação de atributos está ilustrada na **Figura 4.6**. Em **a)** tem-se a notação da agregação de atributos monovalorados caracterizando informações sobre um objeto. Em **b)** tem-se a representação de uma lista, de um conjunto de valores, e de um vetor. A seta com duas pontas opostas representa uma lista, a seta com duas pontas na mesma direção representa um conjunto, e a seta precedida por um colchete representa um vetor. Em **c)** mostra-se a notação de uma tupla formada pela agregação de mais do que um atributo. A representação é efetuada através de setas dispostas

ortogonalmente a uma reta vertical, apontada por uma seta que indica a agregação. Em d) representa-se a associação de atributos a um atributo. Isso é denotado por setas que partem de uma reta ortogonal à seta do atributo. De e) a g) são mostrados alguns exemplos. Em e) tem-se três atributos monovalorados, no caso *Nome*, *CIC* e *Cidade*, que devem estar associados a algum outro conceito, como um objeto ou objeto primário, embora essa associação não esteja mostrada nessa figura; em f) tem-se uma lista, no caso o atributo *Cursos* representando uma seqüência de Cursos, um conjunto, no caso o atributo *Contas Bancárias*, representando um conjunto de números de contas bancárias, e um vetor, no caso o atributo *Formação Básica*, representando um vetor cujos elementos correspondem à formação escolar; ainda em f) tem-se uma tupla *Endereço* composta pelos atributos *Rua*, *Número* e *CEP*; em g) tem-se a representação de atributos de atributos, no caso um atributo representando *Nro Passaporte*, e dois “subatributos” *Data-de-Emissão* e *Órgão-Expedidor*, que se referem aquele número de passaporte.

A notação utilizada na agregação de objetos através de relacionamentos está ilustrada na Figura 4.7. Os relacionamentos são caracterizados por atributos com característica de relacionamento nos objetos envolvidos. Em a) está representado um relacionamento entre dois objetos. A notação forma um

Figura 4.6: Notação utilizada na agregação de atributos e exemplos.

“arco” com a ligação de dois atributos, onde cada um parte de um objeto caracterizando a agregação entre os dois objetos com um pequeno círculo entre eles que representa o objeto relacionamento (objeto primário). No caso de um relacionamento envolvendo dois objetos, cada atributo que o compõe representa o relacionamento oposto um ao outro. Em b) pode-se ver a representação de atributos de relacionamento. A partir do objeto primário (ponto círculo de ligação) entre os dois atributos que caracterizam a agregação

dos objetos é traçada uma reta vertical, da qual partem os atributos desse relacionamento. Os atributos de relacionamento de quaisquer características podem ser atributos simples, tuplas, listas, vetores, conjuntos, ou outros relacionamentos, como de qualquer objeto. Em c) está representado um objeto como valor de um atributo, nesse caso sem a representação de um atributo “oposto”, sendo que a característica do atributo não é de relacionamento, apenas o “tipo de dado” do atributo (com característica de propriedade) é um tipo de objeto.

Também são representados relacionamentos ternários, relacionamentos quaternários, etc. Nesse caso, em mais do que dois objetos existem atributos com características de relacionamento, que se associam via o objeto primário representado pelo círculo central. Em d) está representado um relacionamento ternário. De e) a h) são mostrados alguns exemplos. Em e) é exemplificado um relacionamento binário simples entre dois objetos, onde os objetos *Torneiro* e *Mecânica* estão agregados via os atributos *trabalha em* e *emprega*; em f) está representado um relacionamento entre os objetos *Cliente* e *Produto*, os quais estão agregados através dos atributos *Compra* e *É comprado*. A partir do objeto primário relacionamento é colocado um atributo do relacionamento, que indica o *Valor* da compra, cujo valor instanciado no exemplo é R\$100,00; em g) pode-se ver um caso onde o valor de um atributo é um objeto. Nesse caso, o atributo *possui* do objeto *livro* possui como valor o objeto *Capítulo*. Em h) está representado um relacionamento ternário, onde os objetos participantes são *Máquina*, *Peça* e *Ferramenta*, os quais estão ligados através do objeto primário relacionamento que é o valor dos atributos *Usina*, *Produzida* e *Usa*. Observe que em b) os atributos que determinam o relacionamento, ou seja, que partem dos objetos com destino ao *objeto primário*, são multivalorados (correspondem à cardinalidade *muitos* para *muitos* no ME-R). Quando são atribuídos valores a esse relacionamento, a representação da instância do relacionamento é similar ao ilustrado em f). Note que em f) são colocadas duas chaves envolvendo o objeto primário, significando que se está representando um elemento (uma instância) do conjunto de relacionamentos.

Em SIRIUS define-se um relacionamento particular denominado *relacionamento triplo*, onde a denominação “triplo” não provém do número de objetos participantes, mas sim por identificar três maneiras de um objeto poder participar do relacionamento

[Biajiz-92]. Um *relacionamento triplo* é um relacionamento envolvendo mais de dois objetos, no qual dois dos objetos são capazes de identificar o relacionamento, (através

Figura 4.7:Notação utilizada na agregação de objetos.

de um par de atributos com característica de relacionamento denominado *relacionamento identificador*), com os demais objetos envolvidos acessados a partir do relacionamento identificador, e através do que são chamados *relacionamentos de conjunto*. A partir do círculo onde encontram-se os atributos que formam o arco do relacionamento identificador, parte uma reta de onde se originam os atributos do relacionamento triplo, e a lista de relacionamentos (de conjunto) para os demais objetos envolvidos no relacionamento. Um dos atributos que formam o relacionamento identificador é ligado por um arco aos seus atributos, indicando que cada atributo com característica de relacionamento é considerado ser também um atributo com característica de relacionamento do objeto ao qual esse arco está associado. Um exemplo pode ser visto na **Figura 4.8**. O objeto *Projeto* relaciona-se com os objetos *Empregado*, *Máquina*, *Ferramenta* e *Bancada*. Apesar de cinco tipos de objetos participarem do relacionamento, esse relacionamento não é um *relacionamento quinário*. O objeto

Projeto relaciona-se com o objeto *Empregado* constituindo o *relacionamento identificador*; o objeto *Projeto* associa-se aos objetos *Máquina*, *Ferramenta* e *Bancada* estabelecendo o *relacionamento de conjunto*. Semanticamente, um objeto *Projeto* é desenvolvido por um *Empregado* usando especificamente os objetos *Máquina*, *Ferramenta* e *Bancada*.

Projeto ser desenvolvido por *Empregado* é *identificador*, pois as *Máquinas*, *Ferramentas* e *Bancadas* usadas são identificadas a partir desse relacionamento. Por outro lado, o projeto em si é que usa tais *Máquinas*,

Figura 4.8: Exemplo de relacionamento triplo.

Além disso, considera-se que se outro *Empregado* desenvolvesse esse *Projeto*, outras *Máquinas*, *Ferramentas* e *Bancadas* poderiam ser utilizadas. Pode-se considerar que a existência dos relacionamentos de conjunto e do relacionamento de identificação geram semanticamente três divisões D_1 , D_2 e D_3 . Com essa classificação em divisões, o objeto *Projeto* pertence à divisão D_1 , os objetos *Máquinas*, *Ferramentas* e *Bancadas* pertencem à divisão D_2 , e o objeto *Empregado* pertence à divisão D_3 . Note que a classificação do relacionamento triplo em divisões é feita apenas para que a definição do mesmo se torne mais clara. O relacionamento triplo foi elaborado para suportar algumas situações que ocorrem com alguma frequência em projetos de engenharia.

Outro conceito utilizado em SIRIUS é a *multiplicidade*. Pela própria notação utilizada para atributos identifica-se se o atributo é monovalorado ou multivalorado. Porém, pode-se usar como complemento um dado associado ao atributo que é denominado *multiplicidade*, que indica os limites mínimo e máximo de valores instanciados que um determinado atributo pode assumir. A notação compreende um par de números entre parênteses precedendo o nome do atributo. O primeiro número indica o limite inferior e o segundo indica o limite superior. O limite inferior pode ser 0, 1 ou

outro inteiro definido; o limite superior pode ser 1, N, ou um inteiro definido. A seguinte convenção pode ser usada:

- (0-1) → pode existir nenhuma ou uma instância daquele atributo na agregação;
- (0-N) → pode existir nenhuma ou qualquer quantidade de instâncias daquele atributo na agregação;
- (1-1) → deve existir exatamente uma instância daquele atributo na agregação;
- (1-N) → deve existir ao menos uma instância (limite mínimo) e qualquer número de instâncias daquele atributo na agregação;
- (2-4) → devem existir entre 2 e 4 instâncias desse atributo na agregação.

A *multiplicidade* é um dado complementar, e como tal, sua indicação é optativa.

Entretanto, algumas regras precisam ser seguidas para o uso correto da mesma:

- i) quando um atributo é monovalorado, as multiplicidades possíveis são (0,1) e (1,1).
- ii) quando um atributo é multivalorado e está estruturado na forma de lista ou conjunto, as multiplicidades são livres. Nesses casos, o limite inferior 0 significa a possibilidade de lista vazia ou conjunto vazio, ou seja, o valor do atributo é nulo; limites definidos indicam quantidade mínima ou máxima de elementos na lista ou conjunto.
- iii) quando um atributo é multivalorado e está estruturado na forma de vetor, as multiplicidades indicam o tamanho do vetor; limite inferior 0 significa que o vetor não precisa ter algum elemento definido; limite inferior definido maior que 0 indica quantos elementos têm que estar definidos; limite superior definido indica um vetor com tamanho máximo definido; limite superior igual a N indica um vetor sem um tamanho máximo conhecido.

Um exemplo pode ser visto na **Figura 4.9 a)**:

- o atributo *trabalha em* com característica de relacionamento Re tem multiplicidade (1,1), a qual indica que o objeto do tipo *Empregado* pode relacionar-se através do atributo *trabalha em* com apenas um objeto do tipo *Empresa*;
- o atributo *Emprega* com característica de relacionamento Re tem multiplicidade (1,N), a qual indica que o objeto do tipo *Empresa* deve relacionar-se através do atributo *Emprega* com pelo menos um objeto do tipo *Empregado*, mas não tem limite para a quantidade máxima de *Empregados* que emprega;
- o atributo *Cargo* com característica de propriedade Pr tem multiplicidade (1,1), a qual

- indica que deve ter exatamente um valor instanciado para uma instância do tipo de objeto *Empregado*, ou seja, é obrigatório e monovalorado;
- o atributo *CIC* com característica de propriedade Pr tem multiplicidade (0,1), a qual indica que pode ter nenhum ou exatamente um valor instanciado para uma instância do tipo de objeto *Empregado*, ou seja, não é obrigatório e é monovalorado. O mesmo ocorre com o atributo *Tipo_Empresa* do objeto *Empresa*;
 - o atributo *Fone* com característica de propriedade Pr, tem multiplicidade (0,N), a qual indica que pode ter nenhum ou vários valores instanciados para uma instância do tipo de objeto *Empresa*, ou seja, não é obrigatório, mas é multivalorado;
 - o atributo *Rua* com característica de propriedade Pr, tem multiplicidade (1,1), a qual indica que deve ter exatamente um valor instanciado para uma instância do tipo de objeto *Empresa*, ou seja, é obrigatório e monovalorado;

Figura 4.9: Exemplos de Agregação

- o atributo *Número* com característica de propriedade Pr, tem multiplicidade (1,1), a qual indica que deve ter exatamente um valor instanciado para uma instância do tipo de objeto *Empresa*, ou seja, é obrigatório e monovalorado;

- o atributo *CEP* com característica de propriedade *Pr*, tem multiplicidade (1,1), o que indica que deve ter exatamente um valor instanciado para uma instância do tipo de objeto *Empresa*, ou seja, é obrigatório e monovalorado;
- o atributo *Endereço* com característica de tupla *Tp*, tem multiplicidade (1,1), a qual indica que deve ter exatamente um valor instanciado para uma instância do tipo de objeto *Empresa*, ou seja, é obrigatório e monovalorado;

Os números colocados ao lado da multiplicidade, como por exemplo, 1 para Nome e (1) para Cargo, são utilizados na *abstração de classificação* e serão explicados na seção 4.5.

Formalismo

No capítulo anterior, a *abstração de agregação* foi formalizada independentemente do modelo a que está sendo aplicada. Genericamente, o seu formalismo pode ser denotado como $A_{agreg,i} = \langle A_{Agreg}, Co_{d,Agreg,i} = \{ob_j, j=j', \dots, Agregdl\}, Co_{a,Agreg,i} = \{ob_k\}, Critério(i, A_{Agreg}) \rangle$, onde:

- $Co_{d,Agreg,i}$ é o conjunto não vazio de objetos detalhe dessa ocorrência da abstração A_{Agreg} ;
- $Co_{a,Agreg,i}$ é o conjunto não vazio e unitário de objetos abstratos dessa ocorrência da abstração A_{Agreg} ;
- $Critério(i, A_{Agreg})$ indica como o objeto detalhe está associado ao objeto abstrato.

Como já foi dito, esse formalismo é genérico. Entretanto, o $Critério(i, A_{Agreg})$ compreende como a agregação é estruturada, o que é dependente do modelo M . Considerando então o SIRIUS, o $Critério(i, A_{Agreg})$ é designado pela forma como os atributos são estruturados através da estrutura $Estr_{Agreg} = \langle ta_i, Característica, Multiplicidade, Organização \rangle$, onde:

- ta_i é o atributo que participa da ocorrência da agregação;
- *Característica* é uma entre as possíveis colocadas na seção 4.3.3.
- *Multiplicidade* indica um limite inferior e um limite superior para a ocorrência de instância de determinado atributo;
- *Organização* corresponde à forma como os atributos estarão estruturados, ou seja,

na forma de lista, vetor, conjunto, ou sem uma organização pré-definida (simples);

Na ocorrência de uma abstração de agregação, o tipo de atributo obedece a $Estr_{Agreg}$ definida acima. Desse modo a agregação se dá através dos tipos de atributos segundo a sintaxe ditada pela estrutura $Estr_{Agreg}$ associados aos atributos instanciados, cujo formalismo foi apresentado na seção 4.3.2.

A seguir são exemplificados os formalismos para alguns diagramas já apresentados.

Exemplificando-se esse formalismo sobre a **Figura 4.9**, tem-se:

i) a representação do relacionamento binário entre os objetos *José* e *Mecânica SJ*:

• $Estr_{Agreg} = \langle \text{trabalha em, relacionamento, } \langle 1, 1 \rangle, \text{ simples} \rangle$, onde *trabalha em* é o tipo de atributo participante da agregação;

• $Estr_{Agreg} = \langle \text{emprega, relacionamento, } \langle 1, N \rangle, \text{ simples} \rangle$, onde *emprega* é o tipo de atributo participante da agregação;

• $at_1 = \langle idat_1, \text{trabalha em, Valor} = \text{objpri } 1, \text{José} \rangle$, onde o atributo *trabalha em* representa uma instância com $\text{Valor} = \text{objpri } 1$, sendo *objpri 1* o objeto primário (“objeto relacionamento”) envolvido na agregação, e *José* é o objeto ao qual o atributo está associado;

• $at_2 = \langle idat_2, \text{emprega, Valor} = \text{objpri } 1, \text{Mecânica SJ} \rangle$, onde o atributo *emprega* representa uma instância com $\text{Valor} = \text{objpri } 1$, sendo *objpri 1* o objeto primário envolvido na agregação, e *Mecânica SJ* o objeto ao qual o atributo está associado.

ii) a representação dos atributos *Cargo* e *CIC* pertencentes ao objeto *José*:

• $Estr_{Agreg} = \langle \text{Cargo, Propriedade, } \langle 1, 1 \rangle, \text{ simples} \rangle$;

• $Estr_{Agreg} = \langle \text{CIC, Propriedade, } \langle 0, 1 \rangle, \text{ simples} \rangle$;

• $at_3 = \langle idat_3, \text{Cargo, Valor} = \text{Torneiro, José} \rangle$;

• $at_4 = \langle idat_4, \text{CIC, Valor} = 062360719/28, \text{José} \rangle$.

iii) a representação do atributo *Fone* com estrutura de conjunto pertencente ao objeto *Mecânica SJ*:

• $Estr_{Agreg} = \langle \text{Fone, Propriedade, } \langle 0, N \rangle, \text{ Conjunto} \rangle$;

• $at_5 = \langle idat_5, \text{Fone, Valor} = \{ 2712022, 2728332, 2748383 \}, \text{Mecânica SJ} \rangle$;

iv) a representação do atributo *Endereço* com estrutura de tupla pertencente ao objeto

Mecânica SJ:

- $Estr_{Agreg} = \langle endereço, tupla, \langle 1,1 \rangle, simples \rangle;$
- $Estr_{Agreg} = \langle rua, propriedade, \langle 1,1 \rangle, simples \rangle$
- $Estr_{Agreg} = \langle número, propriedade, \langle 1,1 \rangle, simples \rangle$
- $Estr_{Agreg} = \langle CEP, propriedade, \langle 1,1 \rangle, simples \rangle$
- $at_6 = \langle idat_6, rua, Valor = Brasil, objpri 2 \rangle;$
- $at_7 = \langle idat_7, número, Valor = 1797, objpri 2 \rangle.$
- $at_8 = \langle idat_8, CEP, Valor = 14801-050, objpri 2 \rangle;$
- $at_9 = \langle idat_9, endereço, Valor = objpri 2, Mecânica SJ \rangle.$

v) a representação do atributo tipo *Empresa* pertencente ao objeto *Mecânica SJ*:

- $Estr_{Agreg} = \langle Empresa, propriedade, \langle 1,1 \rangle, simples \rangle;$
- $at_{10} = \langle idat_{10}, Empresa, Valor = Mecânica, Mecânica SJ \rangle.$

Exemplificando-se o formalismo para a **Figura 4.7 g)**² tem-se:

vi) a representação de um atributo cujo valor é um objeto definido pelo usuário:

- $Estr_{Agreg} = \langle possui, propriedade, \langle 1,M \rangle, simples \rangle$, onde *possui* é o tipo de atributo participante da agregação;
- $at_{11} = \langle idat_{11}, possui, Valor = Capítulo, Livro \rangle$, onde o atributo *possui* contém o valor de instância *Capítulo* e está agregado ao objeto *Livro*.

Exemplificando-se o formalismo para a **Figura 4.7 h)** tem-se:

vii) a representação do relacionamento ternário entre os objetos *Máquina*, *Peça* e *Ferramenta*:

- $Estr_{Agreg} = \langle usina, relacionamento, \langle 1,N \rangle, simples \rangle$, onde *usina em* é o tipo de atributo participante da agregação;
- $Estr_{Agreg} = \langle produzida, relacionamento, \langle 1,N \rangle, simples \rangle$, onde *produzida* é o tipo de atributo participante da agregação;
- $Estr_{Agreg} = \langle usa, relacionamento, \langle 1,N \rangle, simples \rangle$, onde *usa* é um tipo de atributo do relacionamento;
- $at_{12} = \langle idat_{12}, usina, Valor = objpri 3, Máquina \rangle$, onde o atributo *usina* representa

uma instância com Valor = *objpri 3*, sendo o *objpri 3* o objeto primário envolvido na agregação, e *Máquina* é o objeto ao qual o atributo está agregado;

• $at_{13} = \langle idat_{13}, produzida, Valor = objpri 3, Peça \rangle$, onde o atributo *produzida* representa uma instância com Valor = *objpri 3*, sendo o *objpri 3* o objeto primário envolvido na agregação, e *Peça* é o objeto ao qual o atributo está agregado.

• $at_{14} = \langle idat_{14}, usa, Valor = objpri 3, Ferramenta \rangle$ onde o atributo *usa* representa uma instância com Valor = *objpri 3*, sendo o *objpri 3* o objeto primário relacionamento envolvido na agregação, e *Ferramenta* é o objeto ao qual o atributo está agregado;

4.5. Abstração de Classificação

4.5.1. Classificação de Objetos em Tipos de Objetos

CONCEITO: Classificação de Objetos em Tipos de Objetos

São particularizações da *abstração de classificação* que permitem a existência de esquemas de dados, cujos elementos são instanciados nas bases extensionais. Assim, a *abstração de classificação* é obrigatória em qualquer modelo de dados, ou seja, todo modelo de dados deve obrigatoriamente suportá-la. Entretanto, nesse trabalho foi apresentada a abstração de classificação de uma forma mais abrangente, cobrindo aspectos que não são cobertos por outros modelos de dados. Entre esses aspectos já discutidos na seção 3.3, está a identificação da *hierarquia de classificação*, de maneira que se possa estabelecer claramente como se dá o processo de classificação /instanciação nos modelos de dados. Neste modelo explora-se esse conceito com mais riqueza do que da forma como é tratado em outros modelos, de maneira que a própria classificação seja tratada como um construtor semântico, que permite criar *hierarquias de classificação* na modelagem, tal como qualquer outro construtor.

O enfoque dado nesse trabalho para a *abstração de classificação* é semelhante

ao enfoque usado em alguns Sistemas de Inteligência Artificial, como o que foi ilustrado no capítulo 2.

Caracterização

Uma ocorrência da abstração de classificação dá-se através da classificação de um conjunto de objetos pelo seu tipo. SIRIUS caracteriza um tipo como sendo também por sua vez, um objeto. Assim, esse processo se repete permitindo estabelecer uma hierarquia de classificação. Dessa forma, um objeto é classificado pelo seu tipo, o qual pode ser classificado novamente até que se atinja o tipo de objeto máximo da hierarquia de classificação. A hierarquia de classificação pode ser vista através de níveis de classificação, que vão do objeto mais “básico” até o objeto máximo da hierarquia. Note-se que não há um número máximo de níveis, sendo esse número determinado pela realidade que está sendo modelada. A capacidade de representação de um tipo de objeto e de suas instâncias constituem um elemento de representação do modelo, ou em outras palavras, um construtor semântico. Este elemento ou construtor envolve uma particularização da abstração de classificação.

Notação

Na Figura 4.10 pode-se ver uma ocorrência da abstração de classificação. Em a) pode-se ver um objeto não instanciável³, no caso o objeto identificado como *TH723*, cujo tipo é *Torno*. Em b) está sendo representado um objeto instanciável, no caso o objeto *Torno*, cujo tipo é *Máquina*. Em c) está sendo representado o objeto *Máquina*. Assim, *Máquina* é uma classificação efetuada sobre *Torno*, enquanto *Torno* é uma classificação efetuada sobre o objeto *TH723*. Do mesmo modo, *Torno* é uma instância de *Máquina*, enquanto *TH723* é uma instância de *Torno*. Note que a ocorrência da abstração de classificação envolve os três objetos juntamente, ou seja, em uma modelagem os três objetos apareceriam em um mesmo diagrama, “relacionando-se” através da abstração de classificação. Em outras palavras, os diferentes níveis da hierarquia da abstração de

3

Um objeto ser ou não instanciável depende da definição dos seus atributos de classificação e de instanciação. Esses atributos serão definidos na seção 4.5.2.

classificação aparecem ao mesmo tempo em uma modelagem, sendo representados ao mesmo tempo os objetos e suas instâncias. Quando o tipo de um objeto não é outro tipo definido pelo usuário, então ele é um tipo inicial cuja classificação é a própria abstração de classificação. Indica-se isso se deixando o retângulo de tipo em branco, como na Figura 4.10 c).

Deve-se observar que assim como os objetos são classificados em tipos de objetos, atributos são classificados em tipos de atributos e colônias⁴ são classificadas em tipos

Figura 4.10 - Representação gráfica de objetos e tipos de objetos.

de colônias. Essas ocorrências da abstração de classificação são diferentes e envolvem diferentes particularizações, ou seja, para objetos é utilizada uma particularização da abstração de classificação, para atributos é utilizada outra particularização, e para colônia é utilizada uma outra particularização da abstração de classificação.

Formalização

Observando a Figura 4.10 pode-se ver a *hierarquia de classificação* gerada pelos passos a), b) e c). Essa hierarquia foi formalmente definida na seção 3.3; o exemplo ilustrado acima está formalmente representado abaixo.

No primeiro nível, a ocorrência da abstração de classificação seria:

$A_{class,0} = \langle A_{class}, Co_{d,class,0} = \{ "Abstração de Classificação" \}, Co_{a,Class,0} = \{ "Abstração" \}, Critério(0, A_{class}) \rangle$.

No segundo nível, a ocorrência seria:

⁴ Colônias e Tipos de Colônias serão apresentadas na seção 4.7.

$A_{class,1} = A_{class}$ $Co_{d,Class,1} = \{\text{"Máquina"}\}$, $Co_{a,Class,1} = \{\text{"Abstração de Classificação"}\}$, $Critério(1, A_{class}) >$.

No terceiro nível, a ocorrência seria:

$A_{class,2} = <A_{class} Co_{d,Class,2} = \{\text{"Torno"}\}$, $Co_{a,class,2} = \{\text{"Máquina"}\}$, $Critério(2, A_{class}) >$.

No quarto nível, a ocorrência seria:

$A_{class,3} = <A_{class} Co_{d,Class,3} = \{\text{"TH723"}\}$, $Co_{a,class,3} = \{\text{"Torno"}\}$, $Critério(3, A_{class}) >$.

Através do formalismo fica mais clara a percepção das ocorrências da abstração de classificação em níveis. Apesar do exemplo acima ser apresentado no SIRIUS, esse formalismo é genérico para qualquer modelo, e genericamente, o formalismo da abstração de classificação pode ser escrito como $A_{Class,i} = <A_{Class} Co_{d,Class,i} = \{ob_j, j=j', \dots, Classdl\}$, $Co_{a,Class,i} = \{ob_k\}$, $Critério(i, A_{Class}) >$, onde:

- $Co_{d,Class,i}$ é o conjunto não vazio de objetos detalhe dessa ocorrência da abstração A_{Class} ;
- $Co_{a,Class,i}$ é o conjunto não vazio e unitário de objetos abstratos dessa ocorrência da abstração A_{Class} ;
- $Critério(i, A_{Class})$ indica que o objeto abstrato é uma classificação do objeto detalhe.

Além disso, deve-se notar que os atributos juntamente com a semântica associada a eles estabelecem o **Critério** de como a classificação é efetuada. O $Critério(i, A_{class})$ compreende como a classificação é estruturada, o que é dependente do modelo M . Considerando então para o modelo SIRIUS, a estrutura denotada por $Estr_{Class}$ pode ser formalizada como $Estr_{Class} = (Idob, Idtob)$, significando que para cada objeto ob , existe um tipo tob que o classifica, o qual é definido (esquematizado) pelo usuário (a menos do objeto metatipo).

No exemplo da **Figura 4.10** pode-se ver o objeto *TH723* sendo classificado como um objeto *Torno*, que por sua vez é classificado como um objeto *Máquina*.

4.5.2. Atributos para a Formatação da Classificação

CONCEITO: Atributos para a Formatação da Classificação

Motivação

Caracterização

Notação

Formalismo

Motivação

A abstração de classificação pode ser vista através da identificação do nome do objeto com o seu respectivo tipo. Entretanto, o **Critério** utilizado para cada nível de uma ocorrência da abstração de classificação/instanciação será estabelecido neste modelo através de um subconjunto dos atributos que caracterizam um objeto que é tipo, e que serão denominados “*Atributos de Classificação do Tipo*”. Através dos atributos pode-se modelar (indicar) quantos níveis no mínimo a hierarquia terá, assim como, quais atributos pertencem aos diversos níveis da hierarquia. Logo, os atributos de classificação definem o “molde” que é usado para modelar cada instância desse tipo.

Caracterização

Os atributos de classificação caracterizam a abstração de classificação carregando a semântica de suas ocorrências. Dessa forma, quem formata 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 do Tipo*”, os quais definem o “molde” de como modelar cada instância desse tipo. Num diagrama representam-se os atributos de classificação colocando a linha vertical dos atributos acima do retângulo que representa o objeto, que nesse caso é um tipo. Os atributos do próprio objeto, denominados “*Atributos de Instância*” (ou “*Atributos Instanciados*”), são representados na linha abaixo do retângulo.

Um atributo ser considerado de classificação indica que ele não é instanciado no objeto tipo onde está ligado, mas será atribuído às instâncias desse objeto. Instanciar um objeto significa criar um objeto instância do objeto tipo. No processo de instanciação de um objeto de determinado tipo, os atributos do tipo são utilizados para controlar esse

processo para que a instância seja criada adequadamente. Os atributos já instanciados não fazem parte desse processo.

Atributo de instância pode estar definido ou indefinido, e estando definido pode ter um valor ou ser nulo. Um atributo instanciado não pode mais ser repassado para suas instâncias. Um atributo de classificação pode ter um valor ou não, além de ter um conjunto de propriedades que estabelecem critérios para a instanciação do objeto ao qual está associado. Um atributo de classificação que tenha valor representa o fato de que esse é o valor “default” repassado para todas as instâncias do objeto ao qual está associado. Note-se que esse processo é dinâmico, no sentido que se o valor mudar, as novas instâncias terão como valor “default” o novo valor, porém isso não altera as instâncias criadas anteriormente à mudança. Um atributo instanciado nunca fica indefinido se no objeto de onde foi instanciado este tiver um valor, porém pode ter valor nulo localmente.

Alguns objetos podem necessitar de atributos “extras”, ou seja, necessitam de caracterização adicional em relação aos tipos de objetos do qual foram instanciados. Eles podem ser atributos de classificação ou de instância, sendo naturalmente adicionados a esses objetos sem que se faça distinção entre os atributos pré-definidos e eles.

É importante caracterizar uma distinção entre valor “default” e valor herdado (este último decorrente da abstração de generalização). Um valor “default” é “carimbado” no objeto instanciado quando de sua criação, o qual pode posteriormente ter esse valor substituído caso a caso (cada instância tem um valor). Um valor herdado corresponde a um atributo que não está associado ao objeto cujo tipo está em questão, mas a um de seus supertipos.

Notação

A notação que envolve as ocorrências de abstração de classificação está descrita abaixo. Os atributos de classificação carregam consigo a propriedade de indicar quantos níveis no mínimo a hierarquia terá, assim como, quais atributos instanciam-se nos diversos níveis da hierarquia. Isso

é indicado com o número precedendo o nome do atributo. O número 1 significa que esse atributo será instanciado na primeira instanciação desse objeto tipo. O número 2 significa que esse atributo será instanciado na primeira instanciação de um objeto instância

Figura 4.11 - Exemplo de ocorrência de abstração de classificação e seus atributos de classificação.

desse tipo e, assim por diante. Outra propriedade é a indicação se o atributo será um **identificador** (de usuário) do objeto onde ele for instanciado. Isso é representado nos diagramas colocando-se entre parênteses () o número do nível onde o atributo será instanciado.

Note-se que o conceito de identificador definido pelo usuário corresponde ao de “atributo chave” nos outros modelos. No entanto, em SIRIUS não se admite que dois ou mais atributos possam ser concatenados para criar uma chave. Um atributo sozinho deve ser suficiente para identificar univocamente o objeto dentro de seu contexto. Por isso usa-se o termo identificador e não chave. A restrição de não se permitir concatenação de atributos para a criação de chaves vai de encontro ao objetivo do modelo de representar como objetos apenas aquilo que no mundo real é percebido (e identificado) pelo usuário como um objeto. Diversos conceitos do SIRIUS foram criados como uma maneira de representar o mesmo efeito que se consegue em outros modelos

com a concatenação de atributos, mas que em SIRIUS é feito levando em consideração a semântica (motivo) da concatenação: relacionamentos, atributos de atributos e relacionamentos como característica de atributos.

A **Figura 4.11** e **Figura 4.12** ilustram esses conceitos. Na **Figura 4.11 a)** mostra-se a representação de um tipo de objeto (o objeto denominado *Livro*), cujo tipo que está em branco é a própria *abstração de classificação*. Esse objeto não tem atributos de instância, apenas atributos de classificação. Esses atributos estão em dois grupos: os precedidos pelo número 1, que serão instanciados em todas as instâncias de *Livros*, ou seja, nos títulos; e os precedidos pelo número 2, que serão instanciados nas instâncias das instâncias de *Livro*, ou seja, nos volumes. Em **b)** mostra-se a representação de um objeto denominado *BD-Fundamentos* cujo tipo é *Livro*. Esse objeto possui atributos de instância e de classificação. Os atributos de instância estão posicionados abaixo do objeto, instanciados com seus respectivos valores, que são *Autor = Navathe*, *ISBN = 13115* e *Edição = 1ª*. Já os atributos de classificação estão posicionados acima dos objetos e serão instanciados em **c)**, **d)** e **e)**. Note que em **a)** o número 1 precedendo o *Título* do *Livro* que está entre parênteses simboliza que ele será um identificador do objeto instanciado, ou seja, não poderão existir 2 *Livros* com o mesmo *Título*. Já o número 2 entre parênteses precedendo o *Exemplar* indica que não será instanciado nas instâncias de *Livro*, mas será instanciado como identificador do objeto instanciado. É por isso que aparece 1 entre parênteses precedendo *Exemplar* em *BD-Fundamentos* indicando que nas instâncias de *BD-Fundamentos*, o atributo *Exemplar* é identificador, ou seja, não existem dois livros *BD-Fundamentos* que tenham o mesmo número de *Exemplar*. Além do tipo de objeto *BD-Fundamentos* estão representados três objetos denominados *Exemplar 1*, *Exemplar 2* e *Exemplar 3* (em **c)**, **d)** e **e)**), ou seja, o atributo *Exemplar* foi instanciado nos objetos *Exemplar 1*, *Exemplar 2* e *Exemplar 3*. Nesse caso, os atributos restantes recebem como valores “default” os valores dos atributos do tipo do qual foram instanciados.

Note que *Exemplar 1*, *Exemplar 2* e *Exemplar 3* não são objetos instanciáveis. Os números sem parênteses são usados para indicar atributos que serão simplesmente instanciados em seus valores, mas sem serem identificadores. É o caso dos atributos *Edição*, *Autor* e *ISBN* de *Livro*, que possuem o número 1 a sua frente. Esse número

indica que eles serão instanciados no próximo nível de instanciação. O *Exemplar 1* está emprestado para

João, o *Exemplar 2* tem o atributo emprestado para definido como nulo, pois não está emprestado para ninguém, e o *Exemplar 3* tem o atributo emprestado para indefinido. Já o atributo emprestável está

Figura 4.12: Exemplos de atributos multivalorados na abstração de classificação.

definido em todas as instâncias de *BD-Fundamentos*, pois tem valor definido no objeto *Livro BD-Fundamentos*. Esse atributo está definido localmente no *Exemplar 1* com valor *Sim*, está definido por “default” no *Exemplar 2* com valor *Não*, e tem o valor *Não* definido localmente no *Exemplar 3*.

Atributos não instanciados podem ser monovalorados como nos exemplos da **Figura 4.12**, ou multivalorados (segundo algumas estruturas como: listas, conjuntos ou vetores). Isso é exemplificado na **Figura 4.12** : em a) mostram-se os tipos de atributos multivalorados “*Autorização de Compra*” (organizado como um vetor de 3 autorizações), e “*Dano Reportado*” (organizado como uma lista de valores organizados em tuplas); e em b) uma possível instanciação do atributo *Autorização de Compra*. No caso dos atributos serem tuplas, cada atributo componente será instanciado independentemente. No caso de atributos estruturados como vetores, o número de elementos deve estar definido como valor não nulo para que possam ser instanciados. Esse número pode ser uma constante, ou pode ser outro atributo com valor do objeto tipo. Em b) a *Autorização de Compra* do *Livro BD-Fundamentos* foi instanciada 03 vezes, das quais a *autorização 1* tem valores locais, a *autorização 2* tem valores nulos

para todos os componentes, e *autorização 3* não está definida. Em c) ilustra-se o fato de que o *Título BD_Fundamentos* foi instanciado no objeto *Exemplar 1*, onde o atributo *Dano Reportado* está em uma lista com 2 valores (cada valor sendo uma tupla com todos os atributos elementares definidos).

Os atributos “extras” não possuem notação diferenciada em relação aos demais atributos. Note-se que quaisquer objetos podem ter atributos “extras” não previstos em seu tipo, sejam de instância ou de classificação. Isso é

Figura 4.13: Exemplo do uso de atributos “Extras”.

fundamental para permitir a construção dos tipos instanciados diretamente do Objeto Abstração de Classificação, mas ocorre também em outros níveis da hierarquia de abstração. Por exemplo, considere-se numa aplicação de modelagem (Figura 4.13) de um chão de fábrica, o Tipo *Máquina*, o qual é instanciado, por exemplo, em *Torno*, *Fresa* e *Furadeira*. Cada instância de *Máquina* pode ser por sua vez instanciada nas ocorrências dessas máquinas. Por exemplo, pode existir o *Torno T12SW34*, o *Torno T78FW32* e a *Furadeira F1* (não representada na figura). No entanto, cada instância de *Máquina* pode ter atributos extras de classificação específicos para essa instância. Por exemplo, o tipo *Torno* pode ter o atributo de classificação “*Comp. do Barramento*” não previsto em *Máquina*, que tem o valor 1,2m na instância de torno *T12SW34* e 2,0m em *T78FW32*.

Os símbolos gráficos apresentados na Figura 4.11, Figura 4.12 e Figura 4.13 “formam” um diagrama que admite representar em uma mesma modelagem, tipos de objetos e suas instâncias. É interessante notar que dessa forma é possível mostrar explicitamente os relacionamentos entre tipos de objetos de níveis de classificação

diferentes, assim como entre objetos e tipos de objetos.

Formalização

A abstração de classificação dá-se através dos objetos em seus tipos, e através dos tipos de atributos de classificação. A estrutura de classificação para os tipos de atributos de classificação é apresentada abaixo.

O tipo de atributo é formalizado como $ta_i = \langle Idta_i, m_i \rangle$, onde $m_i = \{ \langle M \rangle \}$ e a estrutura para os tipos de atributos de classificação é $Estr_{Class} = \langle idta_i, Estado \rangle$ onde:

- **$idta_i$** é o identificador do tipo de atributo de classificação;
- **$Estado = \langle Nível, Identificador, Valor \rangle$** indica o comportamento do tipo de atributo frente as suas instâncias na base, com:

- **$Nível$** indica o “nível” de instanciação e é representado pelos números colocados à frente do tipo do atributo;
- **$Identificador$** recebe o valor “Sim” se o atributo for designado identificador do objeto e “Não” em caso contrário; se for identificador, o número que representa o nível de classificação estará entre parênteses;
- **$Valor$** indica se o tipo de atributo de classificação possui um valor pré-definido para a instanciação.

Para exemplificar considere a **Figura 4.11**. Em **a)** vê-se o tipo de objeto *Livro*, e os tipos de atributos de classificação *emprestável*, *emprestado para*, *título*, *autor*, *ISBN*, *exemplar* e *edição*. Segundo a estrutura definida acima, tem-se:

- $Estr_{Class} = \langle emprestável, \langle 2, Não, nulo \rangle \rangle$;
- $Estr_{Class} = \langle emprestado para, \langle 2, Não, nulo \rangle \rangle$;
- $Estr_{Class} = \langle título, \langle 1, Sim, nulo \rangle \rangle$;
- $Estr_{Class} = \langle autor, \langle 1, Não, nulo \rangle \rangle$;
- $Estr_{Class} = \langle ISBN, \langle 1, Não, nulo \rangle \rangle$;
- $Estr_{Class} = \langle exemplar, \langle 2, Sim, nulo \rangle \rangle$;
- $Estr_{Class} = \langle edição, \langle 1, Não, nulo \rangle \rangle$;

Em **b)** vê-se o tipo de objeto *BD-Fundamentos*, e os tipos de atributos *emprestável*, *emprestado para* e *exemplar*, e os atributos *edição*, *autor* e *ISBN*. Tem-se:

- $Estr_{Class} = \langle emprestável, \langle 1, Não, Valor = Sim \rangle \rangle;$
- $Estr_{Class} = \langle emprestado para, \langle 1, Não, nulo \rangle \rangle;$
- $Estr_{Class} = \langle exemplar, \langle 1, Sim, nulo \rangle \rangle;$
- $at_1 = \langle idat_1, autor, Valor = Navathe, BD-Fundamentos \rangle;$
- $at_2 = \langle idat_2, ISBN, Valor = 13115, BD-Fundamentos \rangle;$
- $at_3 = \langle idat_3, Edição, Valor = 1^a, BD_Fundamentos \rangle.$

Em c) vê-se o objeto *Exemplar 1*, e os atributos *emprestável* e *emprestado para*.

Tem-se:

- $at_4 = \langle idat_4, emprestável, Valor = Sim, Exemplar 1 \rangle;$
- $at_5 = \langle idat_5, emprestado para, Valor = João, Exemplar 1 \rangle.$

Em d) vê-se o objeto *Exemplar 2*, e os atributos *emprestável* e *emprestado para*.

Tem-se:

- $at_6 = \langle idat_6, emprestável, Valor = Não, Exemplar 2 \rangle;$
- $at_7 = \langle idat_7, emprestado para, Valor = Nulo, Exemplar 2 \rangle.$

Em e) vê-se o objeto *Exemplar 1*, e os atributos *emprestável*. Tem-se:

- $at_8 = \langle idat_8, emprestável, Valor = Não, Exemplar 3 \rangle;$

Note que a estrutura de classificação é diferente da estrutura de agregação para um mesmo elemento, ou seja, um elemento tem estruturas diferentes para cada abstração que participa. Entretanto, a instanciação do elemento tem os mesmos valores para todas as abstrações de que participa.

4.6. Abstração de Generalização

CONCEITO: Abstração de Generalização

Motivação

Caracterização

Notação

Formalismo

Motivação

A abstração de generalização é importante e obrigatória para modelos de dados orientados a objetos. Através dela abstrai-se em um objeto o comportamento e o

significado de objetos mais específicos. Além disso, é possível representar e caracterizar herança entre objetos explicitando o predicado responsável pela abstração. Quando voltado para modelos orientados a objetos, a abstração de generalização oferece o suporte para o conceito de polimorfismo em aplicações de bases de dados orientadas a objetos. Esse conceito refere-se à utilização de uma mesma operação para objetos com estruturas diferentes caracterizando sobreposição de nomes de funções.

Caracterização

A abstração de generalização é caracterizada pelo fato de um tipo de objeto poder acrescentar mais detalhes (especializar) a um outro tipo já existente. Note-se que conceitualmente não existe restrição para que quaisquer objetos possam ser especializados ou generalizados. Porém, quando o objeto não é um tipo, especializá-lo ou generalizá-lo, recai no caso trivial de inserção ou eliminação de atributos e, portanto, não precisa fazer parte do modelo em si. Em SIRIUS, a abstração de generalização será particularizada para ser aplicável apenas sobre objetos que são tipo.

SIRIUS impõe para uma generalização que sejam definidos: um tipo que será especializado; um conjunto de tipos especializados; um critério de especialização; restrições para especialização; e um conjunto de regras de inicialização para cada tipo especializado.

O critério de generalização dá-se de duas maneiras: pelo valor de um único atributo cujo domínio seja discreto e pré-definido; e pelo resultado da avaliação de um predicado sobre os valores de um ou mais atributos. Existem ainda três restrições associadas a uma abstração de generalização:

- *Sobreposição* → indica se um objeto do tipo genérico pode ser de 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).

Note que a última restrição inclui um conceito que na literatura é apresentado como um outro tipo de abstração, normalmente denominado *abstração de*

categorização. No contexto desse trabalho considera-se que a *abstração de categorização* é uma especialização da abstração de generalização. Assim, pode-se denominar a abstração de generalização como sendo uma “*generalização ampla*”, e especializá-la em *abstração de categorização* e *abstração de generalização “restrita”*. Essa idéia pode ser vista na **Figura 4.14**. Entretanto, nesse trabalho está sendo considerada apenas a abstração de generalização mais ampla, sendo as duas especializações incluídas através de restrições.

O fato de um tipo ser instanciável ou não depende do conjunto de Ocorrências de Abstração de Generalização (OAG) onde esse tipo participa, tanto como Objeto Tipo Genérico, quanto como Objeto Tipo Específico. As regras para isso são as seguintes:

- Se uma OAG tem o tipo genérico instanciável e categoria total, todos os tipos específicos são instanciáveis;
- Se um tipo é instanciável e não participa como tipo específico de nenhuma OAG com categoria total, então esse tipo é chamado Tipo Natural (é tipo natural); e qualquer tipo que participe como objeto específico em qualquer OAG onde este tipo assume o papel de tipo genérico terá esse objeto como tipo natural (tem tipo natural), bem como recursivamente qualquer de seus subtipos;
- Se um tipo não é natural e participa de uma ou mais OAG com categoria total como tipo específico, então os tipos genéricos de todas essas OAGs deverão ter o mesmo tipo natural;
- Um tipo instanciável não pode participar como tipo específico numa OAG

Figura 4.14: Abstrações utilizadas pelo SIRIUS.

com categoria parcial que tenha como tipo genérico um tipo instanciável.

Notação

A notação utilizada é a encontrada na Figura 4.15. A abstração é mostrada através de uma rede acíclica onde cada nó é um tipo de objeto. Os retângulos que representam tipos específicos têm uma terceira linha dividida em duas. Na parte direita dessa 3ª linha são colocados os valores de inicialização de atributos, relacionamentos e colônias de

Figura 4.15: Exemplo da notação da abstração de generalização.

objetos⁵ do supertipo, quando um tipo é criado segundo seu predicado (está sendo representado pelo conjunto denominado “Iniciac”). Na parte esquerda da linha é colocado o critério para que um objeto de tipo genérico seja desse particular tipo específico. Um retângulo no centro do diagrama de representação indica o critério da generalização:

- quando o critério depende do valor de um atributo cujo domínio é discreto e finito, o nome desse atributo é indicado na metade de baixo do retângulo. Nesse caso a 3ª linha de cada tipo específico tem na parte do predicado o valor que o atributo deve assumir;
- quando o critério depende de mais de um atributo, de uma expressão sobre o atributo, ou de uma coleção de valores de atributos, a parte de baixo do retângulo é deixada vazia. Nesse caso, a 3ª linha do retângulo que representa cada tipo específico tem o seu critério particular indicado.

Um exemplo para cada situação é mostrado na Figura 4.16. Em a) é indicado

⁵Colônias de objetos são apresentadas na próxima seção.

no retângulo ao meio da rede, o atributo que estabelece o critério da generalização. No

Figura 4.16: Exemplos de ocorrências de abstração de generalização.

caso, o atributo é *profissão*, e estabelece que o nome dos tipos de objetos foram generalizados a partir de sua profissão. Já em **b)**, o critério é indicado individualmente e estabelece para cada tipo de objeto que foi generalizado, uma extensão que o define como *criança*, *adolescente* ou *adulto*. Além disso, no retângulo central estão as três restrições que indicam se o relacionamento “is-a” tem participação total ou parcial, se cada tipo de objeto participante tem que ser disjunto, ou pode se sobrepor, e se todo objeto de um tipo específico deve necessariamente ser também do tipo genérico (categoria total ou subtipo) ou não (categoria parcial).

Formalização

A abstração de generalização foi formalizada na seção 3.5, sendo que o exemplo apresentado na parte **a)** da **Figura 15** pode ser formalizado como segue. Em um primeiro nível tem-se: $A_{gener,0} = \langle A_{gener}, Co_{gener,0} \rangle = \{ \text{"Objeto"} \}, Co_{gener,0} = \{$

"Abstração"}, Critério(0, A_{gener})>.

No segundo nível, a ocorrência seria: $A_{gener,1} = \langle A_{gener}, Co_{d,Gener,1} = \{"Pessoa"\}, Co_{a,gener,1} = \{"Objeto"\}, Critério(1, A_{gener}) \rangle$.

No terceiro nível, a ocorrência seria:

$A_{gener,2} = \langle A_{gener}, Co_{d,Gener,2} = \{"Dentista", "Analista", "Engenheiro"\}, Co_{a,gener,2} = \{"Pessoa"\}, Critério(2, A_{gener}) \rangle$.

Do mesmo modo que nas abstrações anteriores, apesar do exemplo acima ser apresentado no SIRIUS, esse formalismo é genérico para qualquer modelo, e genericamente, o formalismo da abstração de generalização pode ser escrito como $A_{Gener,i} = \langle A_{Gener}, Co_{d,Gener,i} = \{ob_j, j=j', \dots, Generdl\}, Co_{a,Gener,i} = \{ob_k\}, Critério(i, A_{Gener}) \rangle$, onde:

- $Co_{d,Gener,i}$ é o conjunto não vazio de objetos detalhe dessa ocorrência da abstração $A_{Gener,i}$;
- $Co_{a,Gener,i}$ é o conjunto não vazio e unitário de objetos abstratos dessa ocorrência da abstração $A_{Gener,i}$;
- $Critério(i, A_{Gener})$ indica que o objeto abstrato é uma generalização do objeto detalhe.

Note que, assim como na abstração de classificação, os atributos juntamente com a semântica associada a eles estabelecem o **Critério** de como a generalização é efetuada. O $Critério(i, A_{Gener})$ compreende como a generalização é estruturada, o que é dependente do modelo. Considerando o modelo SIRIUS, essa estrutura denotada por $Estr_{Gener}$ é formalizada como $Estr_{Gener} = \langle Idto, \{Tipo-Mestre\}, Predicado, \{<inicialização>\}, sobreposição, participação, categorização \rangle$ onde:

- *idto* é o identificador do tipo de objeto participante da abstração;
- *tipo-mestre* é o tipo do objeto abstrato, ou melhor, objeto mais genérico;
- o *Predicado* é composto por um tipo de atributo *ta* ou por um conjunto de tipos de atributos $\{ta_j\}$, onde *ta* é o tipo de atributo cujo valor do atributo instanciado precisa ser satisfeito para a ocorrência da generalização; ou $\{ta_j\}$ é o conjunto de tipos de atributos, cujos valores dos respectivos atributos instanciados precisarão ser satisfeitos pelos respectivos objetos generalizados;

- $\{<inicialização>\}$ compreende o conjunto de valores de inicialização para atributos, relacionamentos e colônias de objetos do tipo mestre, necessários quando um objeto é criado na generalização.
- *sobreposição* é uma das restrições impostas podendo assumir os valores *D* (disjunção) ou *S* (sobreposição);
- *participação* é uma das restrições impostas podendo assumir os valores *T* (total) ou *P* (parcial);
- *categorização* é uma das restrições impostas podendo assumir os valores *T* (total) ou *P* (parcial);

Para exemplificar, considere a **Figura 4.16**. Em **a)** tem-se:

- $Estr_{Gener} = \langle Dentista, \{Pessoa\}, profissão = Bacharel em Odontologia, \{<inicialização>\}, D, P, T \rangle;$
- $Estr_{Gener} = \langle Engenheiro, \{Pessoa\}, profissão = engenheiro, \{<inicialização>\}, D, P, T \rangle;$
- $Estr_{Gener} = \langle Analista, \{Pessoa\}, profissão = bacharel em computação, \{<inicialização>\}, D, P, T \rangle;$

Em **b)** tem-se:

- $Estr_{Gener} = \langle Criança, \{Pessoa\}, Idade < 13, \{<inicialização>\}, D, T, T \rangle;$
- $Estr_{Gener} = \langle Adolescente, \{Pessoa\}, 12 < Idade < 18, \{<inicialização>\}, D, T, T \rangle;$
- $Estr_{Gener} = \langle Adulto, \{Pessoa\}, Idade > 17, \{<inicialização>\}, D, T, T \rangle;$

4.7. Abstração de Composição

CONCEITO: Abstração de Composição

Motivação

Caracterização

Notação

Formalismo

Motivação

No âmbito de Modelos de Dados Orientados a Objetos, o termo composição tem sido usualmente empregado para indicar que dois ou mais objetos associam-se, e que essa associação é representada através de uma referência ao outro objeto, em pelo menos um

dos objetos envolvidos. O significado dessa associação nem sempre tem o significado real de composição. Por exemplo, havendo um objeto *pessoa* que *mora em* objeto do tipo *residência*, isso caracteriza *pessoa* como um objeto composto, pois possui uma

Figura 4.17: Representação de composição (colônia de objetos).

referência a outro objeto reconhecido pelo usuário. No entanto, *pessoas* não são de fato compostas por *residência*.

Em SIRIUS, esta situação é modelada (como de resto em qualquer modelo orientado a objetos) através de uma referência de um atributo de *pessoa* para *residência*, o que se configura como uma Agregação. Portanto, as situações usualmente consideradas “composição de objetos” nos modelos em geral, no SIRIUS são consideradas Agregação. O termo “Composição” é então utilizado para caracterizar situações em que objetos são realmente “*compostos por*” outros, como quando se descreve que um *prédio* é “*composto por*” *salas* e *corredores*. Note-se que a composição entre objetos existe da mesma maneira que na agregação, porém representa um significado diferente, mais específico: cada instância da abstração de composição determina a existência de um objeto composto como o objeto abstrato, e um conjunto

de objetos que são “*parte de*” desse objeto, como objetos detalhe.

Caracterização

Em SIRIUS, ao contrário da abstração de agregação onde os *objetos detalhe* em geral são *objetos primários* que representam atributos e, portanto não reconhecíveis pelos usuários como objetos, os objetos que são *parte de* um objeto composto são objetos de fato, definidos e reconhecidos pelo usuário. Na **Figura 4.17⁶ a)** mostra-se o relacionamento “*é parte*” que existe entre um objeto *composto* e suas *partes*. É comum que esses objetos formem um conjunto que tem propriedades distintas, úteis para o gerenciamento da base. Assim, SIRIUS aproveita esse conceito para gerenciar as permissões de acesso e concorrência, determinando um grão de gerenciamento de acesso determinado pela semântica da aplicação. Esse conceito é denominado *colônia de objetos*. Uma colônia de objetos pode ser definida como um conjunto de objetos que compõe um objeto composto segundo um determinado *Aspecto*. A idéia de *Aspecto* representa o fato de um objeto poder ser visto como composto de diferentes maneiras disjuntas entre si. Por exemplo, uma placa de circuito impresso pode ser composta por uma coleção de componentes, ou pelos elementos que constituem o diagrama que a representa. Na **Figura 4.17 b)** mostra-se os relacionamentos implícitos que ocorrem entre os objetos envolvidos numa ocorrência de uma abstração de composição: um *objeto composto* constringe uma colônia, na qual habitam os objetos *objetos parte* que são *parte* daquele objeto composto. Colônias determinam objetos primários, que como qualquer outro, possuem um tipo. Dessa forma, toda colônia tem um tipo.

Uma colônia pode representar composição lógica ou física de objetos. Impõe-se a restrição de que todo objeto deve habitar fisicamente exatamente uma colônia, e esse vínculo estabelece uma relação de dependência existencial entre o objeto e a colônia em que ele fisicamente habita: 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

6

A figura 4.17 foi extraída do trabalho intitulado “ Compartilhamento de Objetos Compostos entre Bases de Dados Orientadas a Objetos”, Tese de Doutorado apresentada em julho de 1996 ao IFSC - USP, de autoria de João Eduardo Ferreira.

quantidade de colônias, sem causar nenhum tipo de dependência existencial. Note-se que a composição e, portanto, o fato do objeto habitar colônias, é que pode ser física ou lógica, e não a colônia - não se propõe aqui o conceito de colônias físicas/lógicas. Assim, uma colônia pode ter alguns objetos que a habitam fisicamente, e outros objetos que a habitam logicamente. Um objeto pode habitar logicamente a mesma colônia mais de uma vez.

Por

exemplo, pode-

se estar

interessado em

verificar o

aspecto de

recursos de

área construída

de um

departamento,

a qual é

composta por

salas e pátios;

ou pode-se

Figura 4.18: Exemplo de representação de colônias.

estar interessado em verificar o aspecto de *recursos humanos* desse departamento, o qual é composto por *docentes* e *funcionários*. Assim, um objeto de tipo *departamento* *constringe* duas colônias: uma de tipo *Área Construída de departamento*, e outra de tipo *Pessoal de departamento*. *Salas* e *pátios* *habitam fisicamente* a colônia que detalha a composição de cada objeto de tipo *Departamento* segundo seu aspecto de *Área construída*, ao passo que *Docentes* e *Funcionários* *habitam logicamente* uma outra colônia que detalha os *recursos humanos* de um *Departamento*. Como todo objeto deve *habitar fisicamente* uma única colônia, deve existir uma outra colônia na base onde esses objetos *habitam fisicamente*.

Objetos de um determinado tipo podem habitar fisicamente apenas colônias de um tipo pré-definido no esquema da aplicação, porém podem habitar logicamente colônias de um conjunto de tipos indicados no esquema. Por exemplo, *salas* pertencem

(habitam) fisicamente a colônias de tipo *Prédio*, e logicamente podem habitar (pertencer) a *Área Construída de Departamentos* e a *Área Construída de Instituições*, porém jamais podem habitar *Cursos*. Uma composição física ocorre quando todo objeto habita exatamente uma colônia física estabelecendo um “vínculo” de dependência existencial. Uma composição lógica ocorre quando qualquer objeto pode habitar qualquer número de colônias. Nesse caso pode-se ter “vínculo” de dependência existencial ou não, assim como um número mínimo de objetos necessários para a ocorrência da colônia.

Existe uma colônia física de tipo denominado “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.

Notação

A indicação do tipo da colônia que um objeto habita é efetuada nos diagramas através da colocação do nome da colônia na parte direita da linha do retângulo que representa o seu tipo de objeto, sendo que para todo tipo é obrigatória a indicação de exatamente uma colônia que é habitada fisicamente. A primeira colônia indicada considera-se ser habitada fisicamente, as demais logicamente. Um exemplo pode ser visto na **Figura 4.18**. Objetos de tipo *Departamento* e *Docente* **habitam fisicamente** a colônia *Global*, e objetos de tipo *Pátio* **habitam fisicamente** a colônia *Área Construída*. Entretanto, está representada apenas qual colônia o objeto *habita*, mas não estão representadas claramente as *hierarquias de composição*. Para enfocar a representação da modelagem segundo os critérios de composição utiliza-se um diagrama adicional, denominado Diagrama Hierárquico de Colônias (DHC). Um exemplo pode ser visto na **Figura 4.18 b**. O “contorno” que envolve os objetos corresponde às colônias (*objetos primários*). No exemplo estão representadas três colônias: a colônia *Global*, a colônia *Área Construída* e a colônia *Recursos Humanos*. Os objetos *Departamento*, *Pátio*, *Docente* e *Funcionário* **habitam fisicamente** a colônia global; o objeto *Sala* **habita fisicamente**, enquanto o objeto *Pátio* **habita logicamente** a colônia *Área Construída*; e os objetos *Docente* e *Funcionário* **habitam logicamente** a colônia *Recursos Humanos*. As letras “F” e “L” indicam se um objeto habita física ou logicamente uma colônia. As setas que partem do objeto *Departamento* em direção às colônias *Área Construída* e *Recursos*

Humanos, representam o fato de objetos *Departamento* restringirem as respectivas colônias.

Formalização

O formalismo genérico da abstração de composição para qualquer modelo pode ser escrito como $A_{Comp,i} = \langle A_{Comp} Co_{d,Comp,i}, Co_{a,Comp,i} = \{ob\}, Critério(i, A_{Comp}) \rangle$, onde:

- $Co_{d,Comp,i}$ é o conjunto não vazio de objetos detalhe dessa ocorrência da abstração A_{Comp} ;
- $Co_{a,Comp,i}$ é o conjunto não vazio e unitário de objetos abstratos dessa ocorrência da abstração A_{Comp} ;
- $Critério(i, A_{Comp})$ indica que o objeto abstrato é uma composição de objetos detalhe.

Note que, assim como na abstração de classificação e generalização, os atributos juntamente com a semântica associada a eles estabelecem o **Critério** de como a composição é efetuada. O $Critério(i, A_{Comp})$ compreende como a composição é estruturada, o que é dependente do modelo. Considerando o modelo SIRIUS, essa estrutura de composição para a colônia é denotada por $Estr_{Comp} = \langle Idtcol, \{Idtob\}, \{Idtob\}, Idtob_x \rangle$, onde:

- $Idtcol$ é o identificador do tipo de colônia;
- $\{Idtob\}$ é o conjunto de identificadores de tipos de objetos que habitam fisicamente o tipo de colônia;
- $\{Idtob\}$ é o conjunto de identificadores de tipos de objetos que habitam logicamente o tipo de colônia;
- $Idtob_x$ é o identificador do tipo de objeto que restringe a colônia;

Considerando a **Figura 4.18** tem-se:

- $Estr_{Comp} = \langle Global, \{Pátio, Funcionário, Departamento, Docente\}, \{\emptyset\}, MetaObjeto \rangle$;
- $Estr_{Comp} = \langle Área_Construída, \{Sala\}, \{Pátio\}, Departamento \rangle$;
- $Estr_{Comp} = \langle Recursos_Humanos, \{\emptyset\}, \{Funcionário, Docente\}, Departamento \rangle$;

A estrutura acima é para colônias. Para os objetos que participam da abstração de composição, a estrutura é $Estr_{Comp} = \langle idtob, colônia_física, \{colônia_lógica\},$

$\{col\acute{o}nia_constritas\} >$ onde:

- *idto* é o identificador do objeto em questão;
- *col\acute{o}nia_física* é o nome da col\acute{o}nia que o objeto habita fisicamente;
- $\{col\acute{o}nia_l\acute{o}gica\}$ é o conjunto de col\acute{o}nias l\acute{o}gicas que o objeto pode estar habitando;
- $\{col\acute{o}nia_constritas\}$ é o conjunto de col\acute{o}nias que representam as col\acute{o}nias que o objeto em questão constringe;

Considerando a Figura 4.18 b) tem-se:

- $Estr_{Comp} = \langle P\acute{a}tio, Global, \{ \acute{A}rea_Construida \}, \{ \emptyset \} \rangle$;
- $Estr_{Comp} = \langle Funcion\acute{a}rio, Global, \{ Recursos_Humanos \}, \{ \emptyset \} \rangle$
- $Estr_{Comp} = \langle Departamento, Global, \{ \emptyset \}, \{ \acute{A}rea_Construida, Recursos_Humanos \} \rangle$
- $Estr_{Comp} = \langle Docente, Global, \{ Recursos_Humanos \}, \{ \emptyset \} \rangle$
- $Estr_{Comp} = \langle Sala, \acute{A}rea_Construida, \{ \emptyset \}, \{ \emptyset \} \rangle$

4.8.Compara\c{c}\~ao entre as abstra\c{c}\~oes

Tal como definido no cap\i\tulo anterior, a conceitua\c{c}\~ao utilizada para definir SIRIUS determina que todos os objetos de uma modelagem participam de todas as abstra\c{c}\~oes, sendo que cada participa\c{c}\~ao desempenha “fun\c{c}\~oes” distintas em cada abstra\c{c}\~ao. As abstra\c{c}\~oes associam-se, de maneira que conceitualmente as estruturas formadas entrela\c{c}\~am-se, e suportam conceitualmente os objetos. Essa associa\c{c}\~ao est\~a ilustrada na Figura 4.19, e descreve-se a seguir o significado que essas associa\c{c}\~oes t\~em no \~ambito de SIRIUS.

Figura 4.19: Compara\c{c}\~ao entre as diferentes abstra\c{c}\~oes em SIRIUS.

1. Abstra\c{c}\~ao de classifica\c{c}\~ao e abstra\c{c}\~ao de agrega\c{c}\~ao

- a) As abstra\c{c}\~oes de classifica\c{c}\~ao e de agrega\c{c}\~ao constituem a base para a elabora\c{c}\~ao de

qualquer modelo de dados, ou seja, são obrigatórias e estão presentes em todos os modelos de dados.

- b)** A abstração de agregação gerencia a abstração de classificação através da instanciação de atributos, e a abstração de classificação interpreta uma instância.
- c)** Modelos de dados em geral não se ocupam de instâncias. Entretanto em SIRIUS, existe forte consideração sobre instâncias; primeiro porque ocorrem múltiplos níveis de classificação, e segundo porque reconhece que alguns objetos, apesar de únicos, podem ter um significado importante no sistema (hierarquias de classificação). Assim sendo, distingue-se a maneira como cada atributo associa-se a cada objeto segundo o nível que o objeto ocupa na hierarquia de classificação.
- d)** Propriedades, métodos, etc são atributos. Portanto, associam-se a objetos e podem ser instanciados de maneira semelhante. O mesmo ocorre com qualquer característica de atributos, sejam elas estruturais, de interface, estáticas ou dinâmicas. Note-se o tratamento homogêneo que procurou-se dar para esse aspecto - por exemplo, modelos de dados usualmente reconhecem a existência de propriedades de instâncias e de classe, mas métodos só de classes: SIRIUS admite também métodos de instâncias.

2. Abstração de classificação e abstração de generalização.

- a)** Atributos entre objetos relacionados entre si em diferentes níveis nas hierarquias de abstração podem ser repassados de um para outros objetos. Na classificação, esse repasse pode ocorrer na forma de valores “default”, enquanto na generalização esse repasse é feito na forma de herança.
- b)** Na hierarquia de abstração de generalização, todo objeto instanciado é do tipo instanciado e de qualquer outro que suas propriedades correntes e as restrições de cada ocorrência da abstração de generalização permitam. Portanto, instanciar um tipo que pertença a uma hierarquia de generalização é o mesmo que instanciar qualquer outro tipo da hierarquia que seja também tipo desse objeto, e atribuir os valores adequados aos atributos que caracterizam tais tipos (todos eles).

- c) Objetos instanciados que não sejam tipo também podem ser generalizados, embora essa operação não tenha um sentido prático, uma vez que especializar uma instância corresponde apenas a concatenar (ou tornar válida a associação ou consulta de) adicionalmente os atributos específicos àqueles do objeto mais genérico.

3. Abstração de classificação e abstração de composição.

- a) O tipo de um objeto determina o tipo das colônias que suas instâncias irão habitar.

Todo objeto habita fisicamente exatamente uma colônia e todos os objetos de mesmo tipo habitam fisicamente colônias de um único tipo. Os objetos podem habitar logicamente qualquer quantidade de colônias, e podem ser diversos os tipos de colônias que objetos de um mesmo tipo podem habitar.

- b) Um tipo de objeto pode habitar colônias do mesmo tipo que suas instâncias, ou em colônias de tipos diferentes. Quando a colônia onde habita o objeto que é tipo não é a colônia global, podem haver diversas colônias do mesmo tipo, caso contrário o tipo e suas instâncias habitarão todos a mesma colônia (global).

- c) Um tipo de colônia é também um *objeto primário* e assim pode ser classificado em outros tipos.

- d) A hierarquia de classificação define as estruturas de acesso aos objetos instanciados.

Assim, a estrutura dos próprios objetos, tais como são armazenados na base de dados, define a maneira como estes são (e podem ser) acessados. Dessa maneira, os tipos que classificam um objeto devem estar acessíveis nessa estrutura para que suas instâncias possam ser por sua vez acessadas. Isso significa que, embora as estruturas das hierarquias de composição e de classificação são distintas entre si, dois objetos não podem estar em níveis invertidos entre si nas duas hierarquias. Ou seja, um objeto que é tipo não pode estar num nível abaixo de suas instâncias tanto na hierarquia de classificação quanto na de composição.

4. Abstração de generalização e abstração de agregação.

- a) A caracterização de um objeto (tipo) na abstração de generalização é feita atendendo a um predicado, o qual é definido através de atributos associados ao objeto na

abstração de agregação.

- b) Todos os atributos associados (abstração de agregação) a um objeto (tipo) podem ter seus valores herdados por objetos hierarquicamente acima ou abaixo, dependendo das restrições que se aplicam às ocorrências da abstração de generalização em que o objeto participa.

5. Abstração de generalização e abstração de composição.

- a) Dados dois objetos, um abstrato e outro detalhe numa ocorrência da abstração de generalização com a restrição de participação total (subtipos), então as instâncias de ambos os tipos devem habitar, física e logicamente, a mesma colônia. Além disso, é único o tipo de colônia habitado fisicamente por quaisquer objetos instanciados de qualquer um desses dois objetos. Caso a restrição de participação da ocorrência da abstração de generalização seja parcial, não existe nenhuma restrição sobre as colônias (ou tipos de colônias) que instâncias dos objetos detalhe ou objeto abstrato habitam.
- b) Um objeto subtipo de outro na hierarquia de generalização não pode estar em nível acima de seu supertipo na hierarquia de composição.

6. Abstração de agregação e abstração de composição.

- a) os objetos que pertencem a uma hierarquia de composição participam de um relacionamento “*é parte de*”, o qual se dá através de ocorrências de abstrações de agregação.
- b) embora colônias sejam *objetos primários* que a princípio são percebidos pelo usuário como objetos, em SIRIUS não podem ter atributos associados, pois considera-se que atributos da colônia podem ser igualmente bem representados como atributos do objeto que constrange essa colônia.

4.9. Conclusão

Neste capítulo foi apresentado o modelo de dados orientado a objetos denominado SIRIUS. Sua elaboração foi baseada na utilização do metamodelo definido

no capítulo 3. Ele integra num único modelo vários conceitos, os quais são integrados com alta coesão, de maneira que as quatro abstrações envolvidas são altamente interdependentes, ao mesmo tempo em que constituem um todo semanticamente significativo para atender às necessidades para as quais foi concebido.

Foi apresentado também um formalismo para cada abstração suportada pelo modelo. Assim, cada elemento do modelo possui uma estrutura particular para a abstração em que participa. Desse modo, pode-se ver que um objeto definido pelo usuário apresenta diferentes estruturas para a abstração de agregação, classificação, generalização e composição. Entretanto, as instâncias dos objetos são iguais para todas as abstrações suportadas pelo modelo.

Capítulo 5

Contribuições e Sugestões para Futuras Pesquisas

5.1. Contribuições deste trabalho.

Basicamente, pode-se dividir as contribuições deste trabalho em dois grupos: desenvolvimento de um metamodelo, que permite representar de uma maneira uniforme os elementos essenciais de qualquer modelo de dados orientado a objetos; e definição de um modelo de dados construído especificamente para atender às necessidades de ambientes para apoio ao projeto de engenharia e para aplicações científicas.

Quanto ao primeiro grupo, o principal resultado consiste em atender a um anseio de se ter um formalismo para modelos orientados a objetos, cujas principais contribuições são:

- ⇒ Identificação do conceito de Abstração de Dados como o elemento indivisível e único para a caracterização de um modelo de dados
- ⇒ Definição de uma estrutura completa para a representação de como abstrações são parametrizadas, de maneira a representar precisamente como cada abstração é usada em um particular modelo de dados para integrar, com outras abstrações, cada Construtor Semântico de um modelo de dados.
- ⇒ Identificação das Abstrações de Classificação e Associação como fundamentais na construção de qualquer modelo de dados; em particular,

da Abstração de Generalização como fundamental na construção de Modelo de Dados Orientados a Objetos.

- ⇒ Utilização do metamodelo desenvolvido para definir-se a si próprio. Em particular, foi utilizado o conceito de abstrair-se abstrações, mostrando-se por exemplo que as abstrações de agregação, composição e sumarização são especializações da abstração de (que neste trabalho foi tratada pelo termo) Associação, e que mesmo as três abstrações fundamentais identificadas (Classificação, Associação e Generalização) são de fato casos particulares (e portanto instâncias) do conceito de Abstração. Utilizou-se também neste trabalho o conceito de que as Abstrações de Generalização (tal como usualmente tratadas em modelos de dados) e de Categorização podem ser tratadas como especializações de uma abstração mais genérica que engloba a ambas (que é aquela neste trabalho denominada apenas Abstração de Generalização - à forma mais restrita, usualmente encontradas em modelos de dados usuais, não foi dado nome).
- ⇒ Caracterizar que as diversas Abstrações Fundamentais estabelecem três estruturas linearmente independentes, de maneira que qualquer objeto de uma modelagem - incluindo-se aqueles que definem o próprio modelo - estará presente em todas as estruturas, e sua posição numa estrutura independe de sua localização em qualquer uma das demais.
- ⇒ Caracterizar que cada ocorrência de uma abstração envolve sempre exatamente um objeto abstrato, e um ou mais objetos detalhe, bem como caracterizar a existência de exatamente um objeto como o mais abstrato de toda a estrutura. Além disso, caracterizam-se as estruturas geradas como redes acíclicas com um único elemento máximo, que por simplicidade de expressão foram denominadas neste trabalho como Hierarquias não Estritas (hierarquias em que um elemento pode ter mais de um superior, porém não podem haver ciclos).
- ⇒ Mostrar que os construtores semânticos de um modelo de dados, ao contrário do que usualmente é considerado por outros autores de modelos de

dados, podem envolver mais de uma particularização de diferentes abstrações, bem como uma mesma Abstração pode ser particularizada de diferentes maneiras em diferentes Construtores Semânticos de um mesmo modelo de dados.

⇒ Incorporar a Abstração de Classificação entre as Abstrações que podem criar hierarquias com um número não determinado de níveis dentro de um modelo de dados. Em outras áreas da computação, isso já foi tratado em trabalhos anteriores, tais como em modelos de representação de conhecimento, mas é inovador da forma como foi incorporado, dentro de um modelo voltado especificamente ao suporte de estruturas de dados para bases de dados.

⇒ Estabelecer uma maneira de caracterizar a utilização de Abstrações dentro de um sistema, ou de seus aplicativos, versus a utilização de abstrações para especificá-lo e construí-lo. Dentro da conceituação dos três reinos de “objetos”, proposto por Chen em [Chen-76], o conceito de Abstração pode ser aplicado na transição do Reino Imaginário para o Reino da Representação (“Modelagem”), quando o projetista pode abstrair ou detalhar como dois conceitos distintos (pois o resultado obtido pela aplicação de um ou outro em um projeto é distinto), ou pode ser aplicado na transição de uma representação para outra, ou seja, internamente no Reino da Representação, quando então Abstrair ou Detalhar são operações equivalentes, uma vez que ambas as representações coexistem na representação de um mesmo sistema (ou aplicativo).

⇒ Estabelecer um arcabouço conceitual que permite comparar diferentes modelos, bem como mostrar a capacidade de representação semântica de um modelo.

Quanto ao segundo grupo (definição de um modelo de dados para atender às necessidades de ambientes para apoio ao projeto de engenharia e para aplicações científicas), as principais contribuições são:

⇒ Definição do modelo SIRIUS, definido para exemplificar a utilização do metamodelo, com o propósito de suportar a construção de ambientes

dentro dessas duas áreas.

- ⇒ Caracterização de objeto definido pelo usuário e de objeto primário, além do conceito de objeto conceito. Objetos conceitos (ou tratados pelo sistema) correspondem a um conceito usualmente encontrado em outros trabalhos, que caracterizam a existência de um objeto gerenciado (criado, manipulado e destruído) pelo sistema, sem intervenção direta do usuário. Os demais objetos são considerados reconhecidos pelo usuário. Em SIRIUS existem não duas, mas três “classificações” para objetos, pois se desmembra os objetos que o usuário define, e aqueles que, embora ele reconheça como objetos, não os manipula diretamente, mas apenas através de outros. Isso traz para o modelo uma força semântica muito grande.
- ⇒ Introdução do conceito de Atributos de Atributos. Suportando o item anterior, este conceito reforça a separação entre os objetos do sistema que representam “objetos” do mundo real, e os conceitos usualmente criados no processo de modelagem. Esse conceito permite também incorporar ao modelo de maneira natural o conceito freqüentemente adotado na prática de separar os objetos modelados em “fortes” e “locais” (ou “dependentes”).
- ⇒ Definição de Relacionamentos entre objetos, Atributos de Atributos e Tuplas como uma característica estrutural de atributos, suportada de maneira nativa pelo modelo.
- ⇒ Definição precisa da distinção que existe entre a Abstração de Composição e a Abstração de Agregação. Na literatura atual, esses termos são utilizados muitas vezes de maneira ambígua, ou contraditória entre diversos autores.
- ⇒ Introdução do conceito de que um objeto habita fisicamente ou logicamente Colônias de objetos, caracterizando a abstração de composição.
- ⇒ Caracterização da Abstração de Categorização como uma restrição de uma Abstração de Generalização Ampla. Atualmente, diversos autores mencionam a existência da Abstração de Generalização ou da Abstração

de Categorização, porém a menos de [Elmasri-94] nenhum autor caracteriza ambas, e nenhum modelo de dados proposto possui construtores semânticos para ambas (embora alguns trabalhos nomeiem por categorização formas híbridas de generalização restrita e classificação).

5.2 Sugestões para Futuras Pesquisas

As diferentes contribuições deste trabalho abrem perspectivas para novas frentes de trabalho. Dentre as possibilidades de continuação devido ao desenvolvimento de um metamodelo conceitual, as seguintes atividades podem ser desenvolvidas:

- ⇒ Desenvolvimento de uma Álgebra de Abstrações, com capacidade para representar as três Abstrações fundamentais como diferentes “Dimensões” de um objeto, tratadas uniformemente por essa álgebra.
- ⇒ Redefinição do metamodelo utilizando-se a Teoria dos Grafos. Esse enfoque pode ser particularmente interessante, dado que diversos conceitos, tais como os de recursividade, caminho de acesso, pesos em ocorrências de abstração, etc., são naturais dentro de um modelo de dados orientado a objetos, e poderiam ser tratados adequadamente dentro da Teoria dos Grafos.
- ⇒ Definição de um arcabouço mais completo envolvendo as diferentes Abstrações descritas na literatura. Esse trabalho poderia caracterizar cada Abstração, da forma como é tratada em cada trabalho, como particularizações de uma ou mais Abstrações, as quais podem ser Instanciadas/Especializadas/Desassociadas a partir das três Abstrações fundamentais. Assim, poder-se-ia caracterizar, por exemplo, as Abstrações de Sumarização, Referências Cruzadas, Versionamento, Agrupamento, etc., bem como aquelas que sejam específicas de determinados domínios de atividades.
- ⇒ Criar uma ferramenta que permita instanciar SGBDs. Dado que foi definido um metamodelo completo, pode-se criar uma linguagem para especificação de modelos a partir dos conceitos desse metamodelo. Esse enfoque é

particularmente interessante, pois sendo um metamodelo com forte embasamento teórico, torna-se possível conceber uma linguagem de definição concisa e altamente precisa. A partir dessa linguagem, a definição de novos modelos de dados pode ser “recebida” por um sistema capaz de gerar automaticamente um gerenciador de dados que suporte o modelo definido.

- ⇒ criar uma metodologia que permita desenvolver Modelos de Dados específicos para atender às necessidades de um domínio de aplicações. Embora o SIRIUS tenha sido feito sem seguir alguma metodologia em particular, ele foi construído a partir de um Metamodelo, e com o objetivo de atender às necessidades do domínio das aplicações para suporte ao projeto de engenharia e aplicações científicas. A partir dessa tarefa, foi possível perceber alguns passos que deveriam ser seguidos, qualquer que fosse o domínio pretendido. Assim, é possível entrever-se que, a partir do Metamodelo desenvolvido, pode-se desenvolver toda uma metodologia que leve à criação de modelos de dados específicos para suportar aplicações dentro de um domínio particular.
- ⇒ Criação de outros modelos de dados para domínios específicos. Um domínio que pode ser mais bem explorado é o de aplicações científicas mais restritas, como coleta de dados de experimentos, e/ou aplicações estatísticas.

Dentre as possibilidades de continuação devido à definição de um modelo de dados construído especificamente para atender às necessidades de ambientes para apoio ao projeto de engenharia e para aplicações científicas, as seguintes atividades podem ser desenvolvidas:

- ⇒ Definição de um conjunto de tipos de diagramas, cada um voltado para representar uma Abstração em particular, porém adotando todos uma única terminologia gráfica. Assim, o usuário pode representar toda a informação em único diagrama, ou pode criá-los separando a informação

segundo os critérios de abstração que for conveniente para cada visão.

- ⇒ Construção de um Gerenciador de Dados apoiado no SIRIUS. SIRIUS é um modelo de dados orientado a objeto, e como tal pode ser suportado por um gerenciador construído especificamente para suportá-lo. Estruturas de dados Baseadas nas Abstrações suportadas, bem como construções sintáticas baseadas nesses mesmos elementos para uma linguagem de acesso, podem ser construídas de maneira a aproveitar as características do modelo.
- ⇒ Definição de uma linguagem de acesso e consulta específica para SIRIUS.
- ⇒ Criação de Ferramentas de Auxílio ao Projeto e Implementação de sistemas utilizando SIRIUS como modelo de representação.
- ⇒ Construção de um Núcleo Gerenciador de Objetos, genérico o suficiente para ser utilizado como núcleo comum a vários modelos de dados, instanciados a partir do mesmo metamodelo. Estudos preliminares indicam que um tal núcleo pode ser construído para suportar inclusive modelos de dados não orientados a objetos, o modelo relacional entre eles.
- ⇒ Definir um Mapeamento de Modelagens feitas utilizando SIRIUS para modelos suportados por gerenciadores disponíveis comercialmente. Essa atividade é particularmente interessante para o mapeamento para o modelo relacional. A definição de um mapeamento da Abstração de Classificação para o modelo relacional será altamente interessante para as aplicações de projeto em engenharia.
- ⇒ Incorporar elementos de modelagem que suporte a representação de distribuição de dados em SIRIUS. Essa atividade é importante, principalmente devido ao fato de que o domínio alvo (aplicações de projeto e científicas) possui frequentemente requisitos de operação em ambientes distribuídos, cujas restrições e necessidades não são as mesmas atendidas pelas aplicações convencionais.
- ⇒ Incorporar novas construções e/ou refinar as construções semânticas já

existentes em SIRIUS, para adequá-lo ainda mais às aplicações a que se destina. Considerações levantadas, mas não concluídas no levantamento de dados efetuado para a elaboração do modelo podem ser melhor estudadas, tais como por exemplo: a colocação de multiplicidades nas ocorrências da Abstração de composição (multiplicidade do relacionamento habitar logicamente entre objetos e colônias).

Capítulo 6

Referências Bibliográficas

- [Abiteboul-87] Abiteboul, S.; Hull R. - "IFO: A Formal Semantic Database Model". *ACM Transactions on Database Systems*, vol.12, nº4, pg. 525-565, December 1987.
- [Alhajj-93] Alhajj, R.; Arkun, M. E. - "An Object algebra for Object-Oriented Database Systems". *DATA BASE*, vol.24, nº3, August 1993.
- [Ang-93] Ang, J. S. K.; Conrath, D.W. - "The ITHACA Office Object Model: Modeling and Implementation". *DATA BASE*, vol.24, nº4, November 1993.
- [Atkinson-89] Atkinson, M. et alli - "The Object Oriented Database System Manifesto". *Technical Report Altaïr 30-89*, GIP ALTAÏR IN 2-INRIA-LRI, October 1989.
- [Atzeni-93] Atzeni, P.; Torlone, R. - "A Metamodel approach for the management of multiple models and the translation of schemes". *Information Systems*, vol.18., nº6, pg.349-362, 1993.
- [Batory-88] Batory, D.S.; Leung, T.Y.; Wise, T.E. - "Implementation Concepts for an Extensible Data Model and Data Language". *ACM Transaction on Database Systems*, vol.13, nº3, pg. 231-262, September 1988.

- [Bertino-93] Bertino, E.; Lorenzo, M. - "Object Oriented Database Systems". *International Computer Science Series*, Addison-Wesley, 1993.
- [Betz-94] Betz, M. - "Interoperable Objects". *Dr. Dobb's Journal*, October 1994.
- [Betz-95a] Betz, M. - "OMG's CORBA". *Dr. Dobb's Special Report*, Winter 1994/1995.
- [Betz-95b] Betz, M. - "Networking Objects with CORBA". *Dr. Dobb's Journal*, November 1995.
- [Biajiz-92] Biajiz, M. - "*MRO* - Uma Atualização do Modelo de Representação de Objetos e Uma Abordagem Formal*". Dissertação de Mestrado em Ciências da Computação - ICMSC-USP, São Carlos, SP, Julho de 1992.
- [Biajiz-96a] Biajiz, M; Traina, C.Jr. - "Uso da Abstração de Classificação em Modelos de Dados Orientados a Objetos". A ser publicado nos *Anais das 25ª Jornadas Argentinas de Informática*, Buenos Aires, Argentina, 9 a 12 de Setembro de 1996.
- [Biajiz-96b] Biajiz, M; Traina, C.Jr., Vieira, M.T.P. - "SIRIUS - Modelo de Dados Orientado a Objetos e Baseado em Abstrações de Dados". A ser publicado nos *Anais do 11º Simpósio Brasileiro de Bancos de Dados*, São Carlos, S.P. , 14 a 16 de Outubro de 1996.
- [Birkhoff-1970] Birkhoff, G.; MacLane, S. - *A Survey of Modern Algebra*. The Macmillan Company, New York, 3ª edição.
- [Brockschmidt-95] Brockschmidt, K. "OLE Integration Technologies". *Dr. Dobb's Special Report*, Winter 1994/1995.
- [Bronts-95] Bronts, G.H.W.M.; Brouwer, S.J.; Martens, C.L.J. ; Proper, H.A. - "A Unifying Object Role Modelling Theory" - *Information Systems*, vol.20, nº3 pg. 213-235, February 1995.
- [Butterworth-91] Butterworth, P.; Otis, A.; Stein, J. - "The GemStone Object Database Management System". *Communication of the ACM*, vol.34, nº10, pg. 64-77,

October 1991.

[Cattell-94] Cattell, R.G.G. - "*Object Data Management*". Addison-Wesley Publishing Company, 1994.

[Cattell-95] Cattell, R.G.G. - "Object Databases and Standards". *Lectures Notes in Computer Science*, nº 940, pg. 1-11, 1995.

[Chen-76] Chen, P. P. - "The Entity-Relationship Model-Toward a Unified View of Data". *ACM Transactions on Database Systems*, vol.1, nº1, pg. 9-36, March 1976.

[Chorafas-93] Dimitris N. Chorafas - "Manufacturing Databases and Computer Integrated Systems" , Crc Press,Inc./Lewis Publishers, 1993.

[Codd-70] Codd, E.F. - "A relational model of data for large shared data banks". *Communications ACM*, vol.13, nº6, pg. 377-387, June 1970.

[Codd-79] Codd, E.F. - "Extending the database relational model to capture more meaning". *ACM Transaction Database System*, Vol.4, Nº4, Dezembro 1979, pg. 397-434.

[Darwen-95] Darwen, H.; Date, C.J. - "The Third Manifesto". *SIGMOD RECORD*, vol.24, nº 1, March 1995.

[Ebert-94] Ebert, J.; Engels, G. - "Structural and Behavioural Views on OMT-Classes". *Lectures Notes Computer Science*, nº 858, September 1994.

[Elmasri-94] Elmasri, R.; Navathe, S.B. - "Fundamentals of Database Systems". Addison-Wesley Publishing Company, 1994, 2ª Edição.

[Feldman-86] Feldman, P. and Miller, D. - "Entity Model Clustering: Structuring A Data Model By Abstraction". *The Computer Journal*, vol. 29, nº 4, 1986.

[Hanna-95] Hanna, M. S. - "A Close Look at the IFO Data Model". *SIGMOD RECORD*, vol.24, nº1, March 1995.

[Hartzband-85] Hartzband, D.J.; Maryanski, F.J. - "Enhancing Knowledge

- Representation in Engineering Databases", *IEEE Computer*, pag. 39-48, September 1985.
- [Hsu-93] Hsu, C. et al. - "Metadatabase Solutions for Enterprise Information Integration Problems". *Data Base*, pg.23-35, Winter 1993.
- [Hudson-89] Hudson, E.S.; King, R. - "CACTIS: A Self- Adaptive, Concurrent Implementation of an Object-Oriented Database Management System", *ACM Transaction on Database Systems*, vol. 14, no. 3, pp. 291-321, September 1989.
- [Hull-87] Hull R.; King, R. - "Semantic Database Modeling: Survey, Applications and Research Issues". *ACM Computing Surveys*, vol.19, n° 3, pg. 201-259, September 1987.
- [Ishikawa-93] Ishikawa, H. et al. - "The Model, Language, and Implementation of an Object-Oriented Multimedia Knowledge Base Management System". *ACM Transactions on Database Systems*, vol.18, n°1, Março 1993, pg.1-50.
- [Jackson-95] Jackson, R.B.; Embley, D.W.; Woodfield, S.N. - "Developing Formal Object-Oriented Requirements Specifications: A Model, Tool and Technique". *Information Systems*, vol.20, n°4, pg. 273-289, 1995.
- [Jaeschke-95] Jaeschke, P. Oberweis A; Stucky, W. - "Deriving Complex Structured Object Types for Business Process Modelling". *Lectures Notes in Computer Science*, n° 881. Ed. Loucopoulos. ER'94 Entity Relationship Approach, pg. 28 a 45, 1995.
- [Kempe-95] Kempe J. et al. - "Benchmarking Object-Oriented Database Systems for CAD*". *Lectures Notes Computer Science*, n° 978, September 1995.
- [Kim-87] Kim, W. et al. - "Semantics and Implementation of Schema Evolution in Object-Oriented Databases", *ACM*, 1987.
- [Kim-89a] Kim, W. et al. - "Feature of the Orion Object-Oriented Database System" in *Object-Oriented Concepts, Databases, and Applications*, Addison-Wesley, Reading, Massachusetts, 1989.

- [Kim-89b] Kim, W. et alli - "Composite Objects Revisited". *ACM SIGMOD 89 Conference*, Junho 1989, pg. 337-347.
- [Kim-90a] Kim, W. - "Object Oriented Databases: Definition and Research Directions". *IEEE Transactions on Knowledge and Data Engineering*, vol.2, nº3, September 1990.
- [Kim-90b] Kim, W. et alli - "Architecture of the Orion Next-Generation Database System". *IEEE Transactions on Knowledge and Data Engineering*, vol.2, nº1, pg.109-124, pg. 327-341, March 1990.
- [Klas-95] Klas, Wolfgang; Schrefl, Michael - "Metaclasses and Their Application". *Lectures Notes Computer Science*, nº 943, May 1995.
- [Kog-94] Kog, T.W.; Ooi, B.C.; Ho, Y.S. - "A Model Using Classes as a Basic Organization Tool*". *Lectures Notes Computer Science*, nº 760, pg. 429 a 443, 1994.
- [Lamb-91] Lamb, C. et ali. - "The ObjectStore Database System". *Communications of ACM*, vol.34, nº10, October 1991.
- [Liddle-94] Liddle, S. W.; Embley, D. W.; Woodfield, S N. - "A Seamless Model for Object-Oriented Systems Development". *Lectures Notes in Computer Science*, nº 858, pg. 123 a 141, September 1994.
- [Magnan-94] Magnan, C. and Oussalah, C. - "Inheritance systems with exceptions" - *Artificial Intelligence Review* 7, pg. 421-443, 1994.
- [Maier-83] Maier, D., *The Theory of Relational Databases*, Oregon Graduate Center, Computer Science Press, 1983.
- [March-95] March, S.T.; Kim, Y-G. - "Comparing Data Modeling Formalisms". *Communications of the ACM*, vol.38, nº6, June 1995.
- [Mcleod-81] Mcleod D.; Hammer M. - "Database Description with SDM: A Semantic Database Model". *ACM Transaction on Database Systems*, vol.6, nº3, September

1981, pg. 351-386.

[McLeod-91] McLeod D. - "Perspective on object databases". *Information and Software Technology*, vol.33, nº1, January/February 1991.

[Melo-94] Melo, R.N.; Uchôa, E.M.A.; Silva, S.D.S - "Modelo de Dados do Herosistema de Bancos de Dados Heterogêneos Orientado a Objetos". In *9º Simpósio Brasileiro de Banco de Dados*, pg.142-156, Setembro de 1994.

[Naja-95] Naja, H.; Mouaddib, N. - "The multiple representation in an architectural application". *Lectures Notes Computer Science*, nº 978, September 1995.

[Navathe-92] Navathe, S.B. - "Evolution of Data Modeling for Databases". *Communications of The ACM*, vol.35, nº9, pg. 112-123, Setembro de 1992.

[NG-81] NG, P.A. - "Further Analysis of the Entity-Relationship Approach to Database Design". *IEEE Transactions on Software Engineering*, vol.7, nº1, January 1981.

[Oxborrow-88] Oxborrow, E.; Ismail, H.M. - "KBZ - An Object-Oriented Approach to the Specification and Management of Knowledge Bases". In *Proceedings of the Sixth British National Conference on Databases (BNCOD 6)*, pg. 21-46, 1988.

[Peckham-88] Peckham, J. ; Maryanski, F. - "Semantic Data Models". *ACM Computing Surveys*, vol.20, nº3, September 1988.

[Rieu-91] Rieu, D.; Nguyen, G.; Culet, A ;Escamilla, C. - "Instanciation Multiple et Classification D'Objets". In *Septième journées Bases de données Avancées*, Lyon, France, 1991.

[Ruiz-94] Ruiz, D.D.A. - "Um Modelo para Representação de Atividades em Aplicações de Escritório". In *9º Simpósio Brasileiro de Banco de Dados*, pg.95-109, Setembro de 1994.

[Ruiz-95] Ruiz, D.D.A. - "Generalização e Agregação de Atividades: Conceitos Clássicos aplicados à Representação Dinâmica de Aplicações de Escritório". In *10º Simpósio Brasileiro de Banco de Dados*, pg.229- 224, Outubro de 1995.

- [Rush-95] Rush, J. - "OpenDoc". *Dr. Dobb's Special Report*. Winter 1994/1995.
- [Sciore-89] Sciore, E. - "Object Specialization". *ACM Transactions on Information Systems*, vol.7, nº2, pg. 103-122, April 1989.
- [Sheck-90] Sheck, H. e Shcoll M. H. - "Evolution of Data Models". In *Lecture Notes In Computer Science*, vol.466, pg. 135-153, 1990.
- [Shipman-81] Shipman, D. W. - "The Functional Data Model and the Data Language DAPLEX". *ACM Transactions on Database Systems*, vol.6, nº1, pg. 140-173, March 1981.
- [Smith 77a] Smith, J.M., e Smith, D.C.P - "Database Abstractions: Aggregation". *Communication of the ACM*, vol.20, nº6, June 1977, pg.405-413.
- [Smith 77b] Smith, J.M., e Smith, D.C.P. - "Database Abstractions: Aggregation and Generalization". *ACM Transactions on Databases Systems*, vol.2, nº2, pg. 105-133, June 1977.
- [Snyder-93] Snyder, A. - "The Essence of Objects: Concepts and Terms". *IEEE Software*, pg. 31 a 42, January 1993.
- [Soloviev-92] Soloviev, V. - "An Overview of Three Commercial Object-Oriented Database Management Systems: ONTOS, ObjectStore, and O2. *SIGMOD RECORD*, vol.21, nº1, March 1992.
- [Song-93] Song, Y.; Park, E.K. - "Object-Oriented Database Design Methodologies: A Survey". *Lectures Notes Computer Science*, nº 696, November 1992.
- [Souza-93] Souza, F.F. - "Object-Oriented Databases and The KBZ Approach". 8º *Simpósio Brasileiro de Bancos de Dados*, Campina Grande, P.B., 12 a 14 de Maio de 1993.
- [Stonebraker-90] Stonebraker, M. et alli - "Third-Generation Data Base System Manifesto" in *Proceedings IFIP TC2 Conference Object-Oriented Database*, Windermere, Estados Unidos, Julho 1990.

- [Su-86] Su, S.Y.W. - "Modeling Integrated Manufacturing Data with SAM*". *IEEE Computer*, vol.19, nº1, pg.34-39, January 1986.
- [Taylor-76] Taylor, R. W.; Frank, R. L. - "CODASYL database management systems". *Computer Survey*, vol.8, nº1, pg. 67-104, March 1976.
- [Tsichritzis-76] Tsichritzis, D. C., e Lochovsky, F. H. - "Hierarchical database management: A survey". *ACM Computing Surveys*, vol.8, nº1, pg. 105-124, March 1976.
- [Traina-86] Traina Jr., C. - *Máquina e Modelo de Dados Dedicados para Aplicações de Engenharia*. Tese Apresentada ao IFQSC-USP para obtenção do Título de Doutor, Dezembro de 1986.
- [Traina-88] Traina Jr., C. e Slaets, J.F.W. - "Um Modelo de Representação de Objetos". In *Anais do 30. Simpósio Brasileiro de Banco de Dados*, Recife, pg. 227-242, Março de 1988.
- [Traina-94] Traina Jr. C.; Traina, A.J.M.; Biajiz, M. - "O Papel da Abstração de Instanciação em um Meta-Modelo de Abstrações para BDOO", In *Anais do IX Simpósio Brasileiro de Banco de Dados*, São Carlos - S.P., pg. 173-187, Setembro de 1994.
- [Vieira-91] Vieira, M.T.P. - "Um Modelo de Objetos para um Sistema de Gerência de Objetos em Ambientes de Desenvolvimento de sistemas Interativos". *Tese de doutorado apresentada no Departamento de Informática, PUC-RJ*, 1991.
- [Weiser-89] Weiser, S. P. e Lochovsky, F.H. - "OZ+: An Object-Oriented Database System" in *Object-Oriented Concepts, Databases, and Applications*, Addison-Wesley, Reading, Massachusetts, 1989.
- [Williams-95] Williams, S.; Kindel, C. - "The Component Object Model". *Dr. Dobb's Special Report*. Winter 1994/1995.
- [Zhao-88] Zhao, L. ; Roberts, S.A - "An Object-Oriented Data Model for Database Modelling, Implementation and Access". *The Computer Journal*, vol. 31, nº 2, 1988.

- [Zand-95] Zand, M. Et ali - "A Survey of Current Object-Oriented Databases"-
DataBase Advances. vol.26, nº 1, pg. 14-29, February 1995.