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

**MRO* - Uma Atualização do Modelo de
Representação de Objetos e
Uma Abordagem Formal**

Mauro Biajiz

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

*Dissertação apresentada ao Instituto de Ciências Matemáticas de São Carlos - ICMSC - USP -
como parte dos requisitos para obtenção do título de Mestre em Ciências de
Computação e Matemática Computacional*

Julho de 1992

*Dedico este trabalho aos meus pais
Antonio e Clarisse, aos meus avós
Ângelo e Olivia, e em especial a
Célia pela estima e carinho com
que me incentivou.*

AGRADECIMENTOS

Ao Prof. Dr. Caetano Traina Jr., orientador e amigo, pelo apoio e atenção que sempre me dispensou durante a elaboração desse trabalho.

Ao Prof. Dr. Oziride Manzoli Neto, pelas valiosas contribuições dadas diante das questões matemáticas relativas ao trabalho.

A Prof. Dra. Agma Juci Machado Traina, pela paciência e atenção sempre concedida.

A Bete, Laura e Carmem pela simpatia e atenção com que sempre me atenderam.

Aos participantes do Grupo de Estudos de Base de Dados - ICMSC-USP, que contribuíram para o surgimento e a realização de muitas idéias.

Aos amigos Rildo Ribeiro dos Santos, Luiz Camolesi Jr. e Fábio Gomes Figueira pelo auxílio técnico fornecido.

Aos amigos que direta ou indiretamente, contribuíram com seu apoio. Em especial: Ben-Hur, Castilho, Cal, Takai, Zinho e Luis Henrique.

Ao CNPq, pelo apoio financeiro dado para o desenvolvimento desse trabalho.

ABSTRACT

MRO (Modelo de Representação de Objetos) is an Object Oriented Database Model which supports engineering application modeling. This work presents a mathematical formalization of **MRO** the main goals of which are:

- to validate the model's definitions;
- to depurate inconsistencies of the model;
- to supply basis for comparison with other models.

During the formalization process, many inconsistencies and ambiguities were found in **MRO**. A new model, free of those errors and incorporating conceptual enhancements, is now proposed and formalized, and is named **MRO***.

RESUMO

O MRO (Modelo de Representação de Objetos) é um modelo de bases de dados orientado a objetos, o qual suporta modelagens de aplicações de engenharia. Este trabalho apresenta um formalismo matemático do MRO cujas principais metas são:

- validar as definições do modelo;
- depurar inconsistências do modelo;
- fornecer uma base para comparação com outros modelos.

Durante o processo de formalização, muitas inconsistências e ambigüidades foram encontradas no MRO. Um novo modelo, livre de todos os erros e incorporando alguns aspectos conceituais, é agora proposto e formalizado, e é chamado MRO*.

CONTEÚDO

Introdução	- 1 -
1.1 - Considerações Iniciais	- 1 -
1.2 - Definição do Problema	- 2 -
1.3 - Organização do Trabalho	- 3 -
Revisão Bibliográfica	- 5 -
2.1 - Introdução	- 5 -
2.2 - Modelos Baseados em Registros	- 5 -
2.2.1 - Modelo Hierárquico	- 5 -
2.2.2 - Modelo em Rede	- 6 -
2.2.3 - Modelo Relacional	- 6 -
2.3 - Modelos Semânticos	- 24 -
2.3.1 - Modelo Entidade e Relacionamento (ME-R)	- 25 -
2.3.2 - Modelo de Dados Funcional (FDM)	- 25 -
2.3.3 - Modelo de Dados SDM (Semantic Data Model)	- 26 -
2.3.4 - Modelo de Dados IFO	- 26 -
2.4 - Modelos de Dados Orientados a Objetos	- 26 -
2.4.1 - Modelo de Dados OZ	- 28 -
2.4.2 - Modelo de Dados Orion	- 29 -
2.4.3 - Modelo de Dados Iris	- 29 -
2.4.4 - Modelo de Dados PCTE ⁺	- 30 -
2.4.5 - Modelo de Dados Exodus	- 30 -
2.4.6 - Formalismo Geral para um Modelo de Dados Orientado a Objetos ..	- 31 -
2.4.7 - Modelo de Dados O2	- 35 -
2.5 - Conclusão	- 51 -

Modelo de Representação de Objetos (MRO)	- 52 -
3.1 - Introdução:	- 52 -
3.2 - Representação Gráfica do Modelo	- 52 -
3.3 - Descrição dos Elementos do MRO	- 53 -
3.3.1 - Objetos	- 53 -
3.3.2 - Relacionamentos	- 54 -
3.3.3 - Atributos	- 56 -
3.3.4 - Colônias de Objetos	- 61 -
3.3.5 - Versões e Alternativas	- 62 -
3.3.6 - Meta-Eschema de Gerenciamento de Dados	- 63 -
3.3.7 - Identificadores Nativos e Estrangeiros	- 65 -
3.3.8 - Objetos Identificados através de Relacionamentos	- 65 -
3.3.9 - Relacionamentos de Generalização e Sumarização	- 66 -
3.4 - Conclusão	- 69 -
 Mudanças no MRO	 - 70 -
4.1 - Introdução	- 70 -
4.2 - Relacionamento entre Objetos	- 70 -
4.3 - Estado de um Tipo de Atributo	- 74 -
4.3.1 - Identificadores de Objetos	- 74 -
4.3.2 - Propriedades	- 75 -
4.4 - Sumarização	- 75 -
4.5 - Subtipos	- 76 -
4.5.1 - Subtipo de Objeto	- 77 -
4.5.2 - Subtipo de Relacionamento	- 77 -
4.6 - Conclusão	- 78 -

Atualização e Formalização do MRO → MRO*	- 79 -
5.1 - Introdução	- 79 -
5.2 - Objetos	- 79 -
5.3 - Relacionamentos entre Objetos	- 82 -
5.4 - Relacionamentos Envolvendo Múltiplos Objetos	- 86 -
5.5 - Identificação de Objetos por Relacionamentos	- 89 -
5.6 - Atributos	- 90 -
5.7 - Escopo de Abrangência de Identificadores de Objetos	- 92 -
5.8 - Predicados	- 105 -
5.9 - Subtipos de Objetos	- 107 -
5.10 - Subtipos de Relacionamento	- 108 -
5.11 - Sumarização de Objetos	- 109 -
5.12 - Estado de um Tipo de Atributo	- 110 -
5.13 - Esquema	- 112 -
5.14 - Conclusão	- 112 -
 Justificativas e Interpretação para as Mudanças no MRO	- 113 -
6.1 - Introdução	- 113 -
6.2 - Relacionamentos entre Objetos	- 113 -

6.3 - Estado de um Tipo de Atributo	- 117 -
6.3.1 - Tipos de Atributo com Característica de Sinônimo	- 117 -
6.3.2 - Tipos de Atributo com Característica de Propriedade	- 118 -
6.4 - Subtipos	- 119 -
6.4.1 - Subtipos de Objetos	- 119 -
6.4.2 - Subtipos de Relacionamentos	- 121 -
6.4.3 - Sumarização de Objetos	- 122 -
6.5 - Conclusão	- 123 -
 Conclusões e Propostas de Futuras Pesquisas	 - 124 -
7.1 - Conclusões	- 124 -
7.2 - Sugestões de Futuras Pesquisas	- 125 -

LISTA DE FIGURAS

Figura 1: Representação da estrutura de uma base de dados.	- 2 -
Figura 2: Divisão de um Modelo de Dados.	- 3 -
Figura 3: Relação Vôos	- 7 -
Figura 4: Relação Vôos'	- 9 -
Figura 5: Relação Vôos_Origem_São Paulo	- 11 -
Figura 6: Relação Vôos_Partidas_Chegadas	- 11 -
Figura 7: Relação Aparelho	- 12 -
Figura 8: Relação Habilita	- 13 -
Figura 9: Relação Aparelho Habilita	- 13 -
Figura 10: Relação Rotas	- 14 -
Figura 11: Relação Procedência	- 15 -
Figura 12: Relação Rotas Procedência	- 15 -
Figura 13: Relação Horário	- 16 -
Figura 14: Relação Vôos'	- 16 -
Figura 15: Relação Horários_ab	- 17 -
Figura 16: Relação Horários_bc	- 17 -
Figura 17: Relação Concatena-Vôos	- 18 -
Figura 18: Relação r	- 20 -
Figura 19: Relação r'	- 20 -
Figura 20: Relação Nascimento	- 22 -
Figura 21: Relação Nascimento'	- 22 -
Figura 22: Relação Atribui-Vôos	- 23 -
Figura 23: Relação Atribui	- 24 -
Figura 24: Relação Identifica	- 24 -
Figura 25: Exemplos de Objetos em O2	- 36 -
Figura 26: Criação de Classes	- 38 -
Figura 27: Grafo Representando um Conjunto de Objetos	- 43 -
Figura 28: Representação de Objetos.	- 53 -
Figura 29: Objetos sem Identificador.	- 54 -
Figura 30: Tipo de Relacionamento Binário.	- 55 -
Figura 31: Relacionamento Binário.	- 56 -
Figura 32: Tipo de Relacionamento Triplo.	- 57 -
Figura 33: Relacionamento Triplo.	- 58 -
Figura 34: Símbolos utilizados nas características de atributos no MRO.	- 59 -
Figura 35: Tipo de atributo e atributo.	- 60 -
Figura 36: Diagrama Hierárquico de Colônias.	- 63 -

Figura 37: Meta-Eschema do MRO.	- 64 -
Figura 38: Representação de um Subtipo de Objeto.	- 67 -
Figura 39: Representação de um Subtipo de Relacionamento.	- 67 -
Figura 40: Tipos de relacionamentos binários	- 71 -
Figura 41: Tipos de relacionamentos triplos	- 73 -
Figura 42: Tipo de atributo com característica de sinônimo, modelado pelo meta-esquema do MRO.	- 118 -
Figura 43: Tipo de atributo com característica de propriedade, modelado pelo meta-esquema do MRO.	- 119 -

Capítulo 1

Introdução

1.1 - Considerações Iniciais.

Os primeiros sistemas de processamento de dados foram construídos para atender a uma classe de usuários que necessitavam gerenciar depósitos de dados relativamente pequenos, e sem a preocupação de concorrência entre seus dados [Heuser-91]. Em meados da década de 60, começaram a surgir novas aplicações que requeriam com frequência o armazenamento de grandes volumes de dados em diferentes depósitos de dados interligados, os quais sofriam concorrência de acesso. Esses dados passaram então a ser armazenados em locais específicos chamados **Bancos de Dados** ou **Bases de Dados** [Date-75],[Korth-86],[Elmasri-90],[Sheck-90]. Um ambiente de bases de dados tem como característica o compartilhamento de dados por diferentes classes de usuários. Assim, foram desenvolvidos sistemas denominados **Sistemas de Gerenciamento de Bases de Dados (SGBDs)** para suportarem essas características. Desse modo, bases de dados podiam atuar como um meio de comunicação entre diferentes usuários e, para que esta comunicação fosse confiável e eficaz, era necessário que todos os envolvidos tivessem o mesmo entendimento do significado dos dados representados na base de dados. Para esse entendimento, definiu-se um "**modelo**" para os dados, como sendo o centro de toda descrição do significado dos dados representados na base de dados. Esses modelos evoluíram de uma simples documentação central de uma base de dados, até serem vistos como ponto de partida ao projeto da base de dados. Esses modelos foram denominados **Modelos de Dados**. Os modelos de dados são utilizados para "esquematizar" a visão tanto global quanto de particularidades do usuário sobre uma base de dados.

Segundo [Heuser-91], "um **Modelo de Dados** é um modelo abstrato da área da aplicação representada na base de dados, que não contém detalhes de implementação".

Um modelo de dados deve suportar uma descrição dos dados, a interação ou relacionamento entre esses dados, sua semântica, e restrições sobre esses dados. Hoje existem diversos modelos de dados que são classificados segundo suas características de modelagem. Considerando-se o enfoque principal de como esses modelos descrevem os dados, eles podem ser classificados em **modelos de dados baseados em registros**, **modelos de dados semânticos** e **modelos de dados orientados a objetos**.

Um modelo de dados precisa ser bem fundamentado para que, entre outras coisas,

não ofereça ambigüidades ou dúvidas em sua utilização. Procurando solucionar questões como essas, formaliza-se matematicamente esses modelos. Entre outros benefícios, através da teoria formal pode-se provar a corretude das definições do próprio modelo e das modelagens com ele efetuadas, a ausência de ambigüidades nos mesmos, bem como é possível determinar formalmente a sua capacidade de representação, permitindo-se estabelecer critérios de comparação e de escolha de um modelo adequado a determinada aplicação.

1.2 - Definição do Problema.

Na Figura 1 vê-se como se estrutura um Sistema de Base de Dados. Tem-se no círculo central o modelo de dados, o qual deve ser suportado por uma implementação (que o circunscribe na Figura). Para que um usuário (ou aplicativo) possa acessar o sistema, é necessária a existência de uma linguagem capaz de consultar e definir dados na base utilizando-se a implementação suportada pelo modelo. Desse modo, se há alguma "falha" no modelo, há reflexo direto na implementação, definição e manipulação dos dados. As alterações efetuadas sobre o modelo provocam a necessidade de alteração na implementação e na linguagem de definição e manipulação de dados.

O MRO é um modelo de dados orientado a objetos, que foi desenvolvido para suportar aplicações de engenharia. A proposta deste trabalho é formalizá-lo matematicamente, buscando-se obter um

modelo bem fundamentado através da correção de possíveis inconsistências e da validação matemática das definições. Pretende-se formalizar a parte "estática" do modelo juntamente com as restrições de integridade que a envolvem. Pela Figura 2, a qual representa o modelo de dados (o círculo mais interno da Figura 1), verifica-se a parte do modelo formalizada

Figura 1: Representação da estrutura de uma base de dados.

neste trabalho.

O MRO é um modelo proposto originalmente em 1986 [Traina-86], sobre o qual desde 1988 vêm sendo realizados estudos [Camolesi-91] [Ferreira-91] [Silva-91] [Santos-92] [Takai-91] sobre o qual está sendo implementado o Gerenciador de Objetos [Corrêa-92] [Ferreira-91] [Calônego-91] [Traina-90] [Traina-91]. O modelo original de onde partiu este trabalho apresenta-se condensado em [Traina-92]. No entanto, toda atividade do grupo de trabalho

em Base de Dados do ICMSC durante esse período, levou à identificação de um conjunto de melhoramentos e complementações ao modelo que têm ficado dispersos nos diversos trabalhos. Além disso, o processo de formalização identificou também diversas inconsistências no modelo original.

Assim, este trabalho procurou incorporar ao MRO os melhoramentos que já haviam sido identificados e, junto com a correção das inconsistências encontradas, apresenta a proposta de um novo modelo, chamado MRO*, o qual é apresentado de uma maneira formal.

Figura 2: Divisão de um Modelo de Dados.

1.3 - Organização do Trabalho.

Este trabalho está organizado com a seguinte estrutura:

- **capítulo 1:** são apresentadas as idéias que motivaram desenvolvimento deste trabalho, bem como sua organização;
- **capítulo 2:** são apresentados os pontos relevantes do estudo bibliográfico realizado como suporte para o desenvolvimento deste trabalho;
- **capítulo 3:** é descrito o **Modelo de Representação de Objetos (MRO)**, tal como definido originalmente;
- **capítulo 4:** são descritas as mudanças efetuadas no modelo durante a formalização; cuja compreensão é considerada importante para o entendimento da formalização apresentada a seguir
- **capítulo 5:** é apresentada a formalização realizada sobre o MRO; a formalização é

apresentada de maneira completa, envolvendo todos os conceitos do MRO’;

- **capítulo 6:** são apresentadas justificativas e interpretações das mudanças efetuadas sobre o modelo original, e para a maneira como algumas das formalizações foram efetuadas;
- **capítulo 7:** são apresentadas conclusões e propostas para futuras pesquisas.

Capítulo 2

Revisão Bibliográfica

2.1 - Introdução.

Na literatura de Engenharia de Software podem ser encontrados muitos trabalhos sobre o assunto **Modelagem de Dados para Bases de Dados**. Entretanto, existem poucos modelos de dados formalizados matematicamente. Na tarefa de revisão bibliográfica realizada, utilizou-se a classificação em geral utilizada para modelos, a qual os divide e classifica em **modelos baseados em registros, modelos semânticos e modelos orientados a objetos**. Houve um maior enfoque nos modelos que possuem um formalismo matemático desenvolvido para eles.

2.2 - Modelos Baseados em Registros.

Os modelos baseados em registros são assim denominados porque representam a informação em formatos simples, utilizando uma sequência de registros.

2.2.1 - Modelo Hierárquico.

No **modelo de dados hierárquico** [Tsichritzis-76],[Date-75],[Chen-90], os dados são representados por registros e interligações. Um registro de dados pode ter qualquer quantidade de registros dependentes, cada um dos quais pode ter qualquer quantidade de dependentes em um nível mais baixo, e assim sucessivamente, em qualquer número de níveis. Essas ligações estabelecem uma hierarquia de registros, sendo que uma ligação é uma associação entre registros. Uma ligação possui sempre cardinalidade 1 para N. Uma modelagem pode ser constituída utilizando diversas hierarquias, sendo uma primária e diversas secundárias, modelando-se assim qualquer empreendimento.

2.2.2 - Modelo em Rede.

No modelo de dados em rede [Taylor-76],[Date-75],[Chen-90], os dados também são representados por registros, os quais são conectados entre si através de ligações de cardinalidade 1 para N. Porém, no modelo em rede, ligações N para M são representadas utilizando-se um registro intermediário chamado conector ou registro de ligação e, ao contrário do modelo hierárquico, um registro de uma modelagem em rede pode pertencer a mais do que uma ligação, permitindo-se representar redes de registros.

2.2.3 - Modelo Relacional.

O Modelo Relacional [Codd-70], [Codd-79], [Date-75], [Maier-83] foi elaborado baseado na teoria matemática da álgebra das relações, buscando-se obter um modelo para SGBDs que oferecesse ao projetista de aplicação uma interface de nível mais alto do que as apresentadas pelas interfaces dos SGBDs existentes na época. Sobre o modelo foi desenvolvida uma forte teoria formal que visou validar o modelo matematicamente e possibilitou o desenvolvimento de potentes linguagens não procedurais de consulta, o que resultou na obtenção de uma maior semântica de representação dos dados. Esse formalismo é apresentado a seguir.

Conceituação Formal

O modelo relacional tem a sua formalização baseada na álgebra relacional. Os seus dados são apresentados em forma de tabelas que são denominadas *relações*. Assim, dada uma coleção $A_1, A_2, \dots, A_n$, R é uma relação naqueles n conjuntos se ele for um conjunto de n -tuplas ordenadas $\langle a_1, a_2, \dots, a_n \rangle$ tais que a_1 pertença a A_1 , ..., e a_n pertença a A_n . Os conjuntos $A_1, A_2, \dots, A_n$ são denominados atributos. Seja o esquema $\underline{R}$ uma relação. Um nome de atributo pertencente a R é denominado A . Uma relação esquema $\underline{R}$ é um conjunto finito de nomes de atributos $\{A_1, A_2, \dots, A_n\}$. Para cada nome de atributo A_i , existe um conjunto D_i denominado domínio de A_i ou $\text{dom}(A_i)$. Um nome de atributo pode ser chamado símbolo do atributo ou simplesmente atributo. A relação esquema R compreende como os dados são modelados em uma determinada base de dados,

ou seja, modela a parte "intensional" de uma determinada base. Os valores a_i atribuídos aos atributos A_i constituem a instanciación dos dados em uma base, consistindo na parte "extensional" de uma determinada base

Seja $D = D_1 \cup D_2 \cup \dots \cup D_n$. Uma relação r pertencente a relação esquema R é um conjunto finito de mapeamentos $\{t_1, t_2, \dots, t_p\}$ vindos de R para D com a restrição de que para cada $t \in r, t(A_i)$ precisa estar em $D_i, 1 \leq i \leq n$. Os mapeamentos são chamados **tuplas**. Dada uma tupla t , o valor de um atributo A em t é dado por $t(A)$. Por exemplo, na relação esquema Vãos (Figura 3), $Vãos = \{\text{Número, Origem, Destino, Partidas, Chegadas}\}$, estão instanciadas cinco tuplas, na qual uma delas é t , tal que $t(\text{número}) = 83$, $t(\text{origem}) = \text{São Paulo}$, $t(\text{destino}) = \text{Manaus}$, $t(\text{partidas}) = 11:30$ e $t(\text{chegadas}) = 19:00$.

NÚMERO	ORIGEM	DESTINO	PARTIDAS	CHEGADAS
83	São Paulo	Manaus	11:30	19:00
84	São Paulo	Brasília	13:00	18:00
109	Curitiba	Rio de Janeiro	13:00	16:00
213	Rio de Janeiro	São Paulo	10:00	10:35
214	Rio de Janeiro	São Paulo	10:30	11:05

Figura 3 :Relação Vãos

Uma base de dados relacional é composta pelas instâncias das suas relações; uma relação instanciada é composta por instâncias de suas tuplas. A aplicação de uma operação sobre uma relação instanciada, tal como inclusão e exclusão, faz com que o número de tuplas instanciadas varie, criando-se novas relações.

Para a identificação de uma tupla, uma **chave** tem que ser definida. Antes de defini-la, é necessário definir-se o conceito de **superchave**. Uma superchave de uma relação r é um sub-conjunto $K = \{A_1, A_2, \dots, A_m\}$ de R , onde para duas tuplas

$t_1, t_2 \in r, t_1 \neq t_2, \exists A \in K \mid t_1(A) \neq t_2(A)$, ou seja, não existem duas tuplas com os mesmos valores para todos os atributos em K .

Chave de uma relação r , é um sub-conjunto de K , tal que, $\forall t_1, t_2 \in r, t_1(K) \neq t_2(K), \nexists K' \subset K$ que satisfaça essa propriedade, ou seja, uma chave não pode ter nenhum de seus atributos removidos e ainda continuar sendo uma chave. Considerando a relação Vôos, $\{\text{Número}\}$ é uma chave (e uma superchave), enquanto $\{\text{Número}, \text{Partidas}\}$ é uma superchave, mas não é chave.

Operações

Até aqui, foram vistos aspectos formais dos conceitos básicos sobre relações. Agora serão consideradas algumas das operações que podem ser aplicadas às relações.

Operações de Atualização

Seja uma relação esquema $R = \{A_1, A_2, \dots, A_n\}$ e uma relação r sobre R .

- **Operação de Inclusão:** adicionar uma tupla t à relação r .

$$ADD(r: A_1=d_1, A_2=d_2, \dots, A_n=d_n)$$

Essa operação determina a criação de uma nova relação r' onde $\forall t_i | t_i \in r \rightarrow t_i \in r', t = (A_1=d_1, \dots, A_n=d_n) \in r' \text{ e } t \notin r$.

Considere a relação Vôos vista anteriormente. A operação $ADD(\text{Vôos: Número} = 197, \text{Origem} = \text{São Paulo}, \text{Destino} = \text{Londrina}, \text{Partidas} = 13:00, \text{Chegadas} = 17:00)$ adicionará uma tupla à relação Vôos sendo determinada uma nova relação Vôos', descrita na Figura 4.

O resultado de uma operação ADD poderá incorrer em erro se uma das situações ocorrer:

1. a tupla descrita não coincide com a relação esquema da relação especificada;
2. alguns valores da tupla não pertencem ao domínio apropriado;
3. a tupla descrita possui os mesmos valores para os atributos chave de uma tupla já existente.

NÚMERO	ORIGEM	DESTINO	PARTIDAS	CHEGADAS
83	São Paulo	Manaus	11:30	19:00
84	São Paulo	Brasília	13:00	18:00
109	Curitiba	Rio de Janeiro	13:00	16:00
213	Rio de Janeiro	São Paulo	10:00	10:35
214	Rio de Janeiro	São Paulo	10:30	11:05
197	São Paulo	Londrina	13:00	17:00

Figura 4: Relação Vãos'.

- **Operação de Remoção:** remover uma tupla t da relação r .

$DEL(r:A_1=d_1, A_2=d_2, \dots, A_n=d_n)$. Nesta operação é suficiente dar os valores do(s) atributo(s) chave. Se $K=\{A_1, A_2, \dots, A_m\}, m \leq n$ é uma chave, então pode-se usar a forma $DEL(r:A_1=d_1, A_2=d_2, \dots, A_m=d_m)$ para a remoção.

Essa operação determina a criação de uma nova relação r' onde $r' = r - \{t\}$, ou seja, $\forall t_i | t_i \neq t, t_i \in r \rightarrow t_i \in r' \text{ e } t \notin r$.

Considere-se a relação Vãos'. A operação $DEL(Vãos': NUMERO = 197)$ resultará na relação Vãos, que não contém a tupla retirada.

O resultado poderá incorrer em erro, se a tupla não existir.

- **Operação de Alteração:** apenas uma parte da tupla é alterada.

Para uma relação r , com $\{C_1, C_2, \dots, C_p\} \subseteq \{A_1, A_2, \dots, A_n\}$, onde $C_p, 1 \leq j \leq p$ é um atributo, e $e_p, 1 \leq j \leq p$ o seu domínio, a operação de alteração toma a forma $CH(r:A_1=d_1, A_2=d_2, \dots, A_n=d_n; C_1=e_1, C_2=e_2, \dots, C_p=e_p)$. Se $K=\{A_1, A_2, \dots, A_m\}, m < n$ é uma chave, então abrevia-se para $CH(r:A_1=d_1, A_2=d_2, \dots, A_m=d_m; C_1=e_1, C_2=e_2, \dots, C_p=e_p)$.

Essa operação determina a criação de uma nova relação r' , onde $\forall t_p, t_i \in r \text{ e } t_i \in r', \exists t' \in r' | t' \notin r, \text{ e } \exists t'' \in r | t'' \in r'$.

As operações de atualização definidas acima são efetuadas sobre tuplas. As operações a serem definidas a seguir são de **recuperação de dados**, e são efetuadas sobre relações inteiras. Essas operações são definidas usando **operadores relacionais**.

Operações de Recuperação

• **Operadores Booleanos:** sejam r e s relações sobre o mesmo esquema R . Pode-se então definir as seguintes relações:

- $r \cap s$ resulta na relação $q(R)$ contendo todas tuplas que estão em ambas r e s .
- $r \cup s$ resulta na relação $q(R)$ contendo todas as tuplas que estão em r ou s .
- $r - s$ resulta na relação $q(R)$ contendo todas as tuplas que estão em r , mas não estão em s .

• $\bar{r}$ é o complemento de uma relação r se $\bar{r} = \text{dom}(R) - r$, onde $\text{dom}(R)$ é o conjunto de todas as tuplas sobre os atributos de R . Deve ser considerado o fato de que pode haver um atributo $A \in R$ que possua domínio infinito. Nesse caso, $\bar{r}$ não resulta em uma relação finita. Logo, é necessária uma restrição sobre o domínio. Assim define-se o complemento sobre o domínio ativo, que resulta sempre em uma relação finita. Seja $r(A_1, A_2, \dots, A_n)$ uma relação e $D_i = \text{dom}(A_i)$, $1 \leq i \leq n$. Define-se o domínio ativo de A_i relativo a r como sendo o conjunto $\text{adom}(A_i, r) = \{d \in D_i \mid \exists t \in r \text{ com } t(A_i) = d\}$. Tome-se então $\text{adom}(R, r)$ como sendo o conjunto de todas as tuplas sobre os atributos de R e seu domínio relativo a r . O complemento ativo de r é $\bar{r} = \text{adom}(R, r) - r$. Nesse caso, $\bar{r}$ é sempre uma relação finita.

É interessante observar que o conjunto de todas as relações de um dado esquema é fechado sobre união, intersecção, diferença e complemento ativo, mas nem todas essas operações preservam chaves.

• Operador Select σ .

Select é um operador unário sobre relações que, quando aplicado a uma relação r , produz uma outra relação que é um sub-conjunto de tuplas de r com um certo valor sobre um atributo especificado. Então, seja r uma relação sobre o esquema R , e a um elemento do $\text{dom}(A)$. Então, $\sigma_{A=a}(r)$ é a relação $r'(R) = \{t \in r \mid t(A) = a\}$.

Utilizando a relação *Voos*, tem-se que $\sigma_{\text{origem}=\text{Sao Paulo}}(\text{Voos})$ resulta na relação da Figura 5:

NÚMERO	ORIGEM	DESTINO	PARTIDAS	CHEGADAS
83	São Paulo	Manaus	11:30	19:00
84	São Paulo	Brasília	13:00	18:00

Figura 5: Relação Vãos_Origem_São Paulo.

Operadores Select apresentam as seguintes propriedades:

- (i). *comutativa sobre composição*: Seja $r(R)$ uma relação, A e B atributos em R , com $a \in \text{dom}(A)$ e $b \in \text{dom}(B)$. Então $\sigma_{A=a}(\sigma_{B=b}(r)) = \sigma_{B=b}(\sigma_{A=a}(r))$.
- (ii) *distributiva sobre as operações booleanas binárias*: Então $\sigma_{A=a}(r \gamma s) = \sigma_{A=a}(r) \gamma \sigma_{A=a}(s)$ onde $\gamma = \cup, \cap$ ou $-$.

• **Operador Project π**

PARTIDAS	CHEGADAS
11:30	19:00
13:00	18:00
13:00	16:00
10:00	10:35
10:30	11:05

Figura 6: Relação Vãos_Partidas_Chegadas.

Project é um operador unário sobre relações que, quando aplicado a uma relação r , produz uma nova relação que é composta de um sub-conjunto de atributos de r . Então, seja r uma relação no esquema $\underline{R}$ e X um sub-conjunto de R . A projeção de r em X , denotada por $\pi_X(r)$, é a relação $r'(x)$ obtida pela "investigação" das colunas correspondentes aos atributos em $R - X$ e removendo tuplas duplicadas naquelas que restaram.

Utilizando a relação Vãos, tem-se que $\pi_{\{partidas, chegadas\}}(Voos)$ resulta na relação

apresentada na figura 6:

6• Operador Join

Número	Equipamento
83	727
83	747
84	727
84	747
109	707

Figura 7: Relação Aparelho

Join é um operador binário utilizado para combinar duas relações sobre todos os seus atributos comuns. Então, sejam as relações $r(R)$ e $s(S)$, com $R \cap S = T$. O join de r e s , denotado por $r \bowtie s$, resulta na relação $q(T)$ de todas as tuplas t sobre T tal que existem tuplas $t_r \in r$ e $t_s \in s$ com $t_r = t$ e $t_s = t$. Uma vez que $R \cap S$ é um sub-conjunto de ambos R e S , como consequência da definição tem-se $t_r(R \cap S) = t_s(R \cap S)$. Assim, toda tupla em q é uma combinação de uma tupla proveniente de r e uma tupla vinda de s com valores $(R \cap S)$ iguais. Em outras palavras, se duas relações r e s possuem cada uma, ao menos uma coluna definida em um domínio comum, elas podem ser unidas com base nessas colunas. O resultado dessa junção, é uma relação nova e mais ampla, na qual cada tupla é formada pela concatenação de duas tuplas, uma de cada relação original, sendo que as duas tuplas tem o mesmo valor naquelas duas colunas (atributos).

Como exemplo, considere as relações Aparelho e Habilita das Figuras 7 e 8:

Logo, *Aparelho* $\bowtie$ *Habilita* resulta na relação da Figura 9.

Obs.: No caso de $R \cap S = \emptyset$, por definição, $r \bowtie s$ é o Produto Cartesiano de r e s .

Piloto	Equipamento
Tobias	707
Tobias	727
Cláudio	747
Mário	727
Mário	747

Figura 8: Relação Habilita.

NÚMERO	EQUIPAMENTO	PILOTO
83	727	Tobias
83	727	Mário
83	747	Cláudio
83	747	Mário
84	727	Tobias
84	727	Mário
84	747	Cláudio
84	747	Mário
109	707	Tobias

Figura 9: Relação Aparelho $\bowtie$ Habilita.

O operador Join apresenta muitas propriedades. Considere por exemplo uma dada relação r , e a operação de seleção $\sigma_{A=a}(r)$. É possível simular essa operação através da operação Join. Basta definir uma nova relação $s(A)$, com uma tupla singular t , tal que $t(A)=a$. Então, $r \bowtie s = \sigma_{A=a}(r)$. Este resultado pode ser generalizado para obtenção de uma operação Select generalizada utilizando Join. Seja então, $s(A)$ uma relação com k tuplas, $t_1, t_2, \dots, t_k$, onde $t_i(A)=a_i$ e $a_i \in \text{dom}(A)$, $1 \leq i \leq k$. Então $r \bowtie s =$

$$\sigma_{A=a_1}(r) \cup \sigma_{A=a_2}(r) \cup \dots \cup \sigma_{A=a_k}(r).$$

• Operador Equijoin

Sejam $r(R)$ e $s(S)$ relações com $A_i \in R, B_i \in S$, e $\text{dom}(A_i) = \text{dom}(B_i)$, $1 \leq i \leq m$. Os A_i s não necessitam ser distintos, e nem os B_i s. O equijoin de r e s sobre $A_1, A_2, \dots, A_m$ e $B_1, B_2, \dots, B_m$, escrito como $\overset{r \bowtie s}{[A_1=B_1, \dots, A_m=B_m]}$, é a relação $q(RS) = \{t \mid \exists t_r \in r \text{ e } t_s \in s, \text{ com } t(R) = t_r \text{ e } t(S) = t_s \text{ e } t(A_i) = t(B_i), 1 \leq i \leq m\}$.

Como exemplo, sejam as duas relações *Rotas* e *Procedência* dadas na Figura 10 e na Figura 11, respectivamente. A relação resultante $\overset{\text{Rotas} \bowtie \text{Procedencia}}{[\text{Origem}=\text{Aeroporto}]}$ está apresentada na Figura 12.

NÚMERO	ORIGEM	DESTINO
84	São Paulo	Rio de Janeiro
109	Rio de Janeiro	Manaus
117	Curitiba	Brasília
213	Rio de Janeiro	Brasília
214	Brasília	Rio de Janeiro

Figura 10: Relação Rotas.

• Operadores de desigualdade.

Estes operadores, $<$, $>$, $<=$, $>=$, $=$, podem ser utilizados quando o domínio envolvido é ordenado. Para tratamento destes operadores, é suposto um conjunto Θ de comparadores. Então se θ é um comparador em Θ , e A e B são atributos, diz-se que A é comparável por θ com B , se θ está sobre $\text{dom}(A) \bowtie \text{dom}(B)$.

Dentro dessa classe de operadores, alguns dos mais usados são:

PILOTO	AEROPORTO
Mário	Rio de Janeiro
Carlos	Curitiba
Paulo	Curitiba
Tobias	Brasília
Wilson	Manaus
Cláudio	São Paulo

Figura 11: Relação Procedência.

Número	Origem	Destino	Piloto	Aeroporto
84	São Paulo	Rio de Janeiro	Cláudio	São Paulo
109	Rio de Janeiro	Manaus	Mário	Rio de Janeiro
117	Curitiba	Brasília	Carlos	Curitiba
117	Curitiba	Brasília	Paulo	Curitiba
213	Rio de Janeiro	Brasília	Mário	Rio de Janeiro
214	Brasília	Rio de Janeiro	Tobias	Brasília

Figura 12: Relação Rotas $\bowtie$ Procedência
[Origem = Aeroporto]

• Operação de Seleção Estendida.

Seja θ um comparador em Θ . Se r é uma relação no esquema R , A um atributo de R , e a é uma constante pertencente ao $\text{dom}(B)$, onde A e B são θ comparáveis, então $\sigma_{A\theta a}(r) = \{t \in r \mid t(A)\theta a\}$ onde $t(A)\theta a = \theta(t(A), a)$.

Como exemplo, considere a relação Horário na Figura 13:

NÚMERO	PARTIDA	CHEGADA
84	3:00	5:55
109	9:40	12:42
117	15:30	20:00
213	2:00	6:00

Figura 13: Relação Horário.

Aplicando-se $\sigma_{chegada \leq 13:00}(\text{horario})$, obtém-se a relação da Figura 14.

NÚMERO	PARTIDA	CHEGADA
84	3:00	5:55
109	9:40	12:42
213	2:00	6:00

Figura 14: Relação Vãos'.

• Operador Theta-Join

Sejam $r(R)$ e $s(S)$ duas relações que deseja-se combinar, onde $(R \cap S) = \emptyset$. Sejam $A \in R$ e $B \in S$, sendo θ -comparável em Θ . Então, uma operação Theta-join entre duas relações r e s , é denotada por $r[A \Theta B]s$, onde $r[A \Theta B]s$ é a relação $q(R \cup S) = \{t \mid \text{para algum } t_r \in R \text{ e } t_s \in S \mid t_r(A) \Theta t_s(B), \pi(R)=t_r \text{ e } \pi(S)=t_s\}$.

Como exemplo, sejam as duas relações apresentadas na Figura 15 e na Figura 16, onde a primeira mostra os vôos e seus horários de partida de uma cidade **a**, e de chegada em uma cidade **b**; a segunda mostra os vôos e seus horários de partida de uma cidade **b**, e de chegada em uma cidade **c**.

Utilizando a operação Theta-Join:

Horário-ab[Chegadas < Partidas] Horário-bc tem-se uma nova relação, a qual mostra os

NÚMERO	PARTIDAS	CHEGADAS
60	9:40	11:45
91	12:50	2:47
112	4:05	6:15
306	8:30	10:25
420	9:15	11:11

Figura 15: Relação Horários-ab.

NÚMERO	PARTIDAS	CHEGADAS
11	8:30	9:52
60	12:50	1:43
156	4:20	5:40
158	7:10	8:35

Figura 16: Relação Horários-bc.

vôos da cidade *a* para *b*, que chegam antes da partida da cidade *b* para *c*. A nova relação está na Figura 17, e será chamada *Concatena_Vôos*.

Normalização

Toda a força semântica do modelo está representada nas relações; assim, há a necessidade de que as relações sejam bem fundamentadas, no sentido de evitar redundâncias que possam gerar, entre outros problemas, inconsistência de dados. Para garantir esse aspecto, foi desenvolvida uma técnica chamada *Normalização*.

Para o desenvolvimento da *Teoria Formal de Normalização*, foram estabelecidas **dependências funcionais (DF)**¹. Uma DF é uma "formulação" de uma restrição sobre a semântica dos atributos que compõem uma relação.

Seja *r* uma relação sobre o *esquema R*, com *X* e *Y* sendo sub-conjuntos de *R*. A

¹ Em todo o texto, uma dependência funcional será escrita abreviadamente DF.

NÚMERO	PARTIDAS	CHEGADAS	NÚMERO'	PARTIDAS'	CHEGADAS'
60	9:40	11:45	60	12:25	1:43
60	9:40	11:45	156	4:20	5:40
60	9:40	11:45	158	7:10	8:35
91	12:50	2:47	156	4:20	5:40
91	12:50	2:47	158	7:10	8:35
112	4:05	6:15	158	7:10	8:35

Figura 17: Relação Concatena-Vôos.

relação r satisfaz a DF $X \rightarrow Y$, se para todo valor x de X , $\pi_Y(\sigma_{X=x}(r))$ tem ao menos uma tupla. Isto é, sejam duas tuplas t_1 e t_2 em r . Se $t_1(X)=t_2(X)$, então $t_1(Y)=t_2(Y)$. Em uma DF $X \rightarrow Y$, X é denominado **lado esquerdo** e Y é denominado **lado direito**.

Existem dois casos extremos de dependências funcionais que devem ser considerados, que são: $X \rightarrow \emptyset$ e $\emptyset \rightarrow Y$. A DF $X \rightarrow \emptyset$ é trivial, sendo satisfeita em qualquer relação. A DF $\emptyset \rightarrow Y$ é satisfeita para todas as relações nas quais toda tupla tem um único valor para Y .

A seguir serão descritos axiomas de inferência, os quais são construídos sobre o conceito de DFs. Um **axioma de inferência** é uma regra que estabelece que, se uma relação satisfaz certas DFs, ela também satisfaz outras DFs, que são logicamente inferidas através dessa regra.

Para uma relação $r(R)$, em qualquer momento existe uma família de DFs F , que são verdadeiras em r . Um estado de uma relação pode satisfazer uma certa DF, enquanto outro estado não. Deseja-se então encontrar uma família de DFs F , tal que em todos estados permissíveis de r , essas DFs sempre sejam satisfeitas.

O número de dependências funcionais que podem ser aplicadas em uma relação $r(R)$ é finito, uma vez que há apenas um número finito de sub-conjuntos de R . Assim, é sempre possível encontrar todas as DFs que são verdadeiras em r . Conhecendo alguns membros de F , é freqüentemente possível deduzir outros membros de F . Um conjunto F de DFs, denotado por $F \models X \rightarrow Y$, implica na existência da DF $X \rightarrow Y$, se toda relação que satisfaz todas as DFs em F também satisfaz $X \rightarrow Y$.

São introduzidos agora, seis axiomas para DFs. Na declaração das regras, r é uma

relação sobre R , e W, X, Y , e Z são sub-conjuntos de R .

F1. Reflexiva.

A relação $\Pi_X(\sigma_{X \rightarrow X}(r))$ sempre tem ao menos uma tupla, logo $X \rightarrow X$ sempre existe em r .

F2. Aumentativa.

Este axioma lida com o aumento do lado esquerdo de uma DF. Se r satisfaz $X \rightarrow Y$, então $\pi_Y(\sigma_{X \rightarrow X}(r))$ tem ao menos uma tupla para algum valor x de X . Se Z é algum sub-conjunto de R , então $\sigma_{XZ \rightarrow X}(r) \subseteq \sigma_{X \rightarrow X}(r)$ e $\pi_Y(\sigma_{XZ \rightarrow X}(r)) \subseteq \pi_Y(\sigma_{X \rightarrow X}(r))$. Assim, $\pi_Y(\sigma_{XZ \rightarrow X}(r))$ tem ao menos uma tupla, e r precisa satisfazer $XZ \rightarrow Y$.

F3. Aditiva.

Este axioma permite combinar duas DFs com os mesmos lados esquerdos. Se r satisfaz $X \rightarrow Y$ e $X \rightarrow Z$, então $\pi_Y(\sigma_{X \rightarrow X}(r))$ e $\pi_Z(\sigma_{X \rightarrow X}(r))$ tem ao menos uma tupla para algum valor x de X . Se $\pi_{YZ}(\sigma_{X \rightarrow X}(r))$ tem mais do que uma tupla, então ao menos um dos $\pi_Y(\sigma_{X \rightarrow X}(r))$ e $\pi_Z(\sigma_{X \rightarrow X}(r))$ deveria ter mais do que uma tupla. Desse modo, r satisfaz $X \rightarrow YZ$.

F4. Projetiva.

Este axioma é mais ou menos o inverso da aditiva. Se r satisfaz $X \rightarrow YZ$, então $\pi_{YZ}(\sigma_{X \rightarrow X}(r))$ tem ao menos uma tupla para algum valor x de X . Uma vez que $\pi_Y(\pi_{YZ}(\sigma_{X \rightarrow X}(r))) = \pi_Y(\sigma_{X \rightarrow X}(r))$, $\pi_Y(\sigma_{X \rightarrow X}(r))$ pode ter ao menos uma tupla. Assim, r satisfaz $\underline{X} \rightarrow Y$.

F5. Transitiva.

Este axioma e o próximo são os mais "fortes" dos axiomas de inferência. Seja r satisfazendo $X \rightarrow Y$ e $Y \rightarrow Z$. Considere as tuplas t_1 e t_2 em r . Sabe-se que se $t_1(X) = t_2(X)$, então $t_1(Y) = t_2(Y)$, e também se $t_1(Y) = t_2(Y)$, então $t_1(Z) = t_2(Z)$. Como consequência, se $t_1(X) = t_2(X)$, então $t_1(Z) = t_2(Z)$, e assim r satisfaz $X \rightarrow Z$.

F6. Pseudotransitiva.

Seja r satisfazendo as DFs $X \rightarrow Y$ e $YZ \rightarrow W$ e seja t_1 e t_2 tuplas em r . Sabe-se que se $t_1(X) = t_2(X)$, então $t_1(Y) = t_2(Y)$ e também se $t_1(YZ) = t_2(YZ)$, então $t_1(W) = t_2(W)$. Vindo de

$t_1(XZ)=t_2(XZ)$ pode-se deduzir que $t_1(X)=t_2(X)$ e também que $t_1(Y)=t_2(Y)$ e ainda que $t_1(YZ)=t_2(YZ)$, a qual implica $t_1(W)=t_2(W)$. Desse modo, r satisfaz $XZ \rightarrow W$.

Considere como exemplo a relação r na Figura 18:

A	B	C	D
a_1	b_1	c_1	d_1
a_2	b_2	c_1	d_1
a_1	b_1	c_1	d_2
a_3	b_3	c_2	d_3

Figura 18: Relação r .

- 1.A relação r satisfaz a DF $A \rightarrow B$. Pelo Axioma $F2$, $AB \rightarrow A$, $AC \rightarrow B$, $AD \rightarrow B$, $ABC \rightarrow B$, $ABD \rightarrow B$, $ACD \rightarrow B$, e $ABCD \rightarrow B$.
- 2.A relação r satisfaz as DFs $A \rightarrow B$ e $A \rightarrow C$. Pelo Axioma $F3$, $A \rightarrow BC$.
- 3.A relação r satisfaz a DF $A \rightarrow BC$. Pelo Axioma $F4$, $A \rightarrow B$, $A \rightarrow C$.

Considere agora a relação r' na Figura 19:

A	B	C	D
a_1	b_1	c_2	d_1
a_2	b_2	c_1	d_1
a_3	b_1	c_2	d_2
a_4	b_1	c_2	d_3

Figura 19: Relação r' .

- 4.A relação r satisfaz a DF $A \rightarrow B$ e $B \rightarrow C$. Pelo axioma $F5$, $A \rightarrow C$.

Deve-se notar que, através dos Axiomas F1 até F6, é possível se derivar outras regras de inferência para DFs. Por exemplo:

Seja r uma relação sobre R com X e Y sub-conjuntos de R . O axioma F1 afirma que r satisfaz $Y \rightarrow Y$. Aplicando o axioma F2, tem-se que r satisfaz $XY \rightarrow Y$. Uma outra maneira de estabelecer esta regra, é que para $Y \subseteq X \subseteq R$, r satisfaz $X \rightarrow Y$.

Agora efetivamente será descrita a normalização.

Dada uma relação esquema R , uma chave em R é um sub-conjunto K de R , tal que para qualquer possível relação $r(R)$, não há duas tuplas distintas t_1 e t_2 em r , tal que $t_1(K) = t_2(K)$, e nenhum sub-conjunto próprio K' de K tem esta propriedade. Uma superchave é qualquer conjunto de atributos contendo uma chave.

Em termos de DFs, uma chave para um esquema R é um sub-conjunto K de R tal que para qualquer relação possível $r(R)$, a DF $K \rightarrow R$ é satisfeita, mas nenhum sub-conjunto próprio de K tem esta propriedade. Se K é uma chave para r , então não há tuplas distintas t_1 e t_2 em r , tal que t_1 e t_2 tenham os mesmos valores em K . Logo, enquanto há a FD $K \rightarrow R$, se $t_1(K) = t_2(K)$, tem-se que $t_1 = t_2$, a qual é denotada $t_1(R) = t_2(R)$.

Será assumido de agora em diante que uma relação *esquema* R é composta de duas partes, S e K , onde S é um conjunto de atributos e K é um conjunto de chaves. O *esquema* R será escrito como $R = (S, K)$. Antes da definição das formas normais, algumas definições são necessárias.

•Seja U um conjunto de atributos, cada um com um domínio associado. Um *esquema* R sobre U , de uma **base de dados relacional**, é uma coleção de relações esquema $\{R_1, R_2, \dots, R_p\}$, onde $R_i = (S_i, K_i)$, $1 \leq i \leq p$, $\bigcup S_i = U$, $1 \leq i \leq p$, e $S_i \neq S_j$ se $i \neq j$. Uma base de dados d sobre o esquema R , é uma coleção de relações $\{r_1, r_2, \dots, r_p\}$ tal que para cada relação esquema $R = (S, K)$ em R , existe uma relação r em d tal que, r é uma relação sobre S que satisfaz toda chave em K .

•Dado um conjunto de DFs F e uma DF $X \rightarrow Y$, Y é **totalmente dependente** de X , se não há um sub-conjunto próprio X' de X , tal que $X' \rightarrow Y$.

•Dada uma relação esquema R , um sub-conjunto X de R , um atributo A em R , e um conjunto de DFs F , A é **transitivamente dependente** sobre X em R , se há um sub-conjunto Y de R com $X \rightarrow Y$, Y não determina X e $Y \rightarrow A$ sobre F e $A \notin XY$.

Formas Normais.

• Uma relação esquema R está na **primeira forma normal (1NF)**, se todos os valores no $dom(A)$ são atômicos para todo atributo A em R . Isto é, os valores no domínio não são listas, ou conjuntos de valores, ou valores compostos. Um esquema de bases de dados R está na 1FN, se toda relação esquema em R está na 1FN.

Um valor que é atômico em uma aplicação, pode ser não atômico em outra aplicação. Um valor é não atômico se a aplicação lida com somente uma parte do valor.

Consideremos como exemplo a relação *Nascimento* apresentada na Figura 20:

Nome	Aniversário
Alberto	7 de Junho, 1967
Luiz	12 de dezembro de 1977

Figura 20: Relação Nascimento.

Se a aplicação possui interesse somente no dia e mês que uma pessoa nasceu, a relação *Nascimento* não está na primeira forma normal, uma vez que está lidando com apenas parte do valor do atributo *aniversário*. Para a relação *Nascimento* estar na primeira forma normal, ela deve ser quebrada como na Figura 21. Assim, todos os valores da relação se tornam atômicos, e a relação está na primeira forma normal.

Nome	Mês	Dia	Ano
Alberto	Junho	7	1967
Luiz	Dezembro	12	1977

Figura 21: Relação Nascimento.

• Uma relação esquema $R = (S, K)$ está na **segunda forma normal (2NF)**, com respeito a um conjunto de DFs F , se ela está na 1NF, e todo atributo pertencente a S é **totalmente dependente** da chave toda de R . Um esquema de bases de dados R , está na segunda forma normal com respeito a F , se toda relação esquema R em R está na 2FN com respeito a F .

Considerando como exemplo a relação *Data*, tome o atributo nome como o atributo chave. Os outros atributos restantes são não pertencentes à chave, e são totalmente dependentes da chave. Portanto a relação está na 2NF.

• Uma relação esquema R está na **terceira forma normal (3NF)**, com respeito a um conjunto de DFs F , se ela está na 1FN, e nenhum atributo não pertencente à chave em R é **transitivamente dependente** sobre uma chave de R . Um esquema de bases de dados R , está na terceira forma normal com respeito a F , se toda relação esquema R em R está na 3FN com respeito a F .

Considere como exemplo a relação *Atribui_Vôos* apresentada na Figura 22:

Vôos	Dia	Num-Piloto	Nome
112	6 de junho	31174	Luiz
112	7 de junho	30046	Marcos
203	9 de junho	31174	Luiz

Figura 22: Relação *Atribui_Vôos*.

Existem as seguintes dependências na relação acima: $Vôo \rightarrow Dia$, $Vôo \rightarrow Num-Piloto$, $Vôo \rightarrow Nome$, $Num-Piloto \rightarrow Nome$, $Nome \rightarrow Num-Piloto$. Nome é transitivamente dependente de $Vôo \rightarrow Dia$, uma vez que $Vôo \rightarrow Dia$, $Vôo \rightarrow Num-Piloto$, $Num-Piloto \rightarrow Nome$, e $Nome \rightarrow Num-Piloto$. Logo a relação *Atribui_Vôos* não está na 3FN. Para colocá-la na 3FN, a relação deve ser dividida em duas outras novas relações, apresentadas na Figura 23 e na Figura 24, respectivamente.

Vôos	Dia	Num-Piloto
112	6 junho	31174
112	7 junho	30046
203	9 junho	31174

Figura 23: Relação Atribui.

Num-Piloto	Nome
31174	Luiz
30046	Marcos

Figura 24: Relação Identifica.

Agora, as relações Atribui e Identifica obedecem à definição e, portanto, estão na 3FN.

O nível de normalização "aumenta" de uma forma normal para outra. Uma relação na 3NF, está também na 2NF e na 1NF. Existem outras formas normais que foram desenvolvidas, as quais progressivamente "retiram" complexidades (simplificam) das relações.

Deve-se notar que o nível de normalização de uma relação é uma questão **semântica**, e não dos valores dos atributos que aparecem naquela relação em um dado instante.

2.3 - Modelos Semânticos.

Os **modelos de dados semânticos** [Hull-87] [Peckham-88] são assim denominados por serem baseados em características de modelagem apoiadas em "construções" semânticas com altos níveis de abstração [Smith-77a], [Smith-77b], [Su-86]. Assim, a modelagem apoiada nesses modelos alcança um maior grau de separação entre os componentes físicos e conceituais, diminui a carga semântica de tipos de relacionamento, e apresenta vários construtores de tipos capazes de representarem abstratamente situações reais.

2.3.1 - Modelo Entidade e Relacionamento (ME-R).

O modelo de dados entidade-relacionamento [Chen-76] baseia-se na adoção de uma visão natural do mundo real através de *entidades e relacionamentos*. Uma entidade é um elemento do mundo real que pode ser identificado; por exemplo, uma pessoa, uma companhia, um evento, etc. Um relacionamento é uma associação entre duas entidades; por exemplo, "pai-filho" é um relacionamento entre duas entidades pessoa. Às entidades e relacionamentos são atribuídas informações através de *atributos* e seus *valores*. Assim, pode-se por exemplo, atribuir a uma entidade pessoa o atributo idade, cujo valor pode variar de uma idade mínima 0 a uma idade máxima 100.

Foi apresentado um formalismo para o ME-R em [NG-81] baseado na teoria de conjuntos, e que o classifica em *modelo determinístico* ou *não determinístico*, conforme estiverem definidos os atributos que o compõem. Assim, se a modelagem possui atributos apenas com cardinalidade muitos para um ou um para um, ele é denominado modelo determinístico, e caso haja pelo menos um atributo com cardinalidade muitos para muitos, ele é denominado modelo não determinístico.

2.3.2 - Modelo de Dados Funcional (FDM).

O Modelo de Dados Funcional (FDM) [Shipman-81] modela os aspectos do mundo real através de entidades e funções, caracterizando-se principalmente por ter sua semântica centrada em relacionamentos funcionais.

O FDM possui um formalismo desenvolvido através da incorporação de uma linguagem de definição e manipulação de dados denominada Daplex. Essa é uma linguagem funcional, que estabelece uma sintaxe clara e segura para modelagens através do modelo.

Um modelo funcional obtém dados a respeito de alguma entidade armazenada através de funções aplicadas sobre a entidade. Por exemplo, pode-se saber o nome de uma pessoa através da função Nome(pessoa) definida como f:pessoa→string. A especificação de propriedades para as funções, tal como número de atributos, sobrejeção, etc, é determinada pela semântica de cada aplicação.

2.3.3 - Modelo de Dados SDM (Semantic Data Model).

O Modelo de Dados SDM [Mcleod-81] é um modelo elaborado para conter a maior quantidade possível das características semânticas encontradas em outros modelos. Assim, SDM se tornou um dos mais ricos e flexíveis modelos semânticos encontrados na literatura, possibilitando que o projetista de bases de dados tenha uma extensa variedade de recursos para modelar sua aplicação.

2.3.4 - Modelo de Dados IFO.

O Modelo de Dados IFO [Abiteboul-87] é um modelo elaborado sobre uma teoria matemática, suportando os principais conceitos da modelagem semântica. Seu formalismo é baseado na teoria dos grafos, sendo fundamentado sobre um grafo direcionado com vários tipos de vértices e arcos, os quais são representados usando várias convenções diagramáticas. IFO foi desenvolvido para fornecer uma base para a investigação teórica dos aspectos utilizados pelos modelos semânticos, possuindo características do ME-R, do FDM e algumas características estruturadas do SDM.

2.4 - Modelos de Dados Orientados a Objetos.

Os Modelos de Dados Orientados a Objetos [Kim-90a],[Atkinson-89],[Stonebraker-90],[Mcleod-91],[Jackson-91], caracterizam-se por modelarem aspectos do mundo real como objetos. A modelagem de dados utilizando o conceito de orientação a objetos é recente, e muita confusão ainda se estabelece a respeito de uma conceituação básica para essa classificação. Assim, à seguir são apresentados alguns requisitos básicos, segundo [Kim-90a], para que um modelo seja classificado como orientado a objetos.

- **Objeto e Identificador de Objeto:** toda entidade do mundo real é modelada como um objeto. Além do mais, todo objeto é associado a um *identificador unívoco*, com a propriedade de referenciá-lo a qualquer momento;

- **Encapsulamento:** os dados e as possíveis operações sobre esses dados são "ocultadas" do usuário, podendo ser manipulados apenas pelo próprio objeto que encapsula esses dados;
- **Atributos e Métodos:** todo objeto tem um estado e um procedimento. O estado de um objeto é o conjunto de valores que os atributos do objeto contêm em um determinado instante, e o procedimento de um objeto é o conjunto de métodos (códigos de programa que definem as operações possíveis sobre o objeto) que operam sobre o estado do objeto. O valor de um atributo de um objeto é também um objeto, podendo assumir um valor singular ou um conjunto de valores. Um conjunto de valores não é um objeto, apesar de que cada elemento em um conjunto é um objeto (não há consenso com relação a esse conceito na literatura). O estado e o procedimento de um objeto são encapsulados, sendo acessados ou acionados de fora do objeto, através da passagem de mensagens explícitas;
- **Classe e Tipo:** classe é especificada como o meio de agrupar todos os objetos os quais compartilham o mesmo conjunto de atributos e métodos. Um objeto pertence a somente uma classe como uma instância daquela classe. Uma classe é similar a um tipo abstrato de dados, onde o usuário vê apenas a interface do tipo, sendo denominada classe primitiva, quando tem instâncias associadas, mas não tem atributos. O valor de um atributo de um objeto, o qual é necessariamente um objeto, também pertence a alguma classe. O conceito de tipo está ligado ao conceito de classe, sendo que reconhece-se classe como o conjunto de todos os objetos de determinado tipo;
- **Objetos Complexos**[Kim-89b]: são assim caracterizados por serem compostos por outros objetos, os quais podem ser compostos por outros objetos formando, assim, uma hierarquia. Objetos complexos são criados através da aplicação de construtores de tipos sobre objetos simples ou complexos. Entre esses construtores de tipos, os principais são os conjuntos, arrays, tuplas e listas;
- **Hierarquia e Herança de Classes:** sistemas orientados a objetos permitem ao usuário obter uma nova classe a partir de uma classe existente; a nova classe, chamada subclasse da classe existente, herda todos os atributos e métodos da classe existente, chamada superclasse da nova classe. Uma classe pode ter qualquer número de subclasses. Alguns sistemas permitem que uma classe possua somente uma superclasse, enquanto outros permitem qualquer número de superclasses. No caso em que uma classe herda atributos e métodos de apenas uma superclasse, tem-se uma *herança singular*. Quando uma classe herda atributos e métodos de mais de uma superclasse, tem-se *herança múltipla*.

Os modelos de dados orientados a objetos são assim classificados por suportarem a maioria dos conceitos considerados acima. Hoje, existe uma divisão na linha de pensamento

com relação a modelos de dados orientados a objetos e SGBDOOs². Uma das tendências pode ser classificada como revolucionária [Atkinson-89], enquanto a outra pode ser dita evolucionária [Stonebraker-90]. A maior divergência entre as duas, encontra-se no fato, de que a primeira procura desvincular totalmente os SGBDOOs dos SGBDs Convencionais. Já a segunda tendência, defende que SGBDs Convencionais possuem algumas das características de orientação a objetos desejadas, podendo "evoluir" para a nova geração de sistemas, bastando para isto, adicionar algumas características, como construtores de tipo, herança, e a implementação de linguagens de programação persistentes. Ou seja, os pesquisadores que defendem a primeira tendência, argumentam que um SGBDOO é um sistema orientado a objetos nos moldes de uma linguagem orientada a objetos com recursos de bases de dados, ao passo que os pesquisadores que defendem a segunda tendência argumentam que um SGBDOO é um SGBD convencional com características de orientação a objetos.

A seguir são apresentados alguns dos principais modelos de dados orientados a objetos, destacando-se dois modelos que possuem um formalismo elaborado para si.

2.4.1 - Modelo de Dados OZ.

O *modelo de dados OZ* [Weiser-89] foi desenvolvido na Universidade de Toronto, com a intenção de demonstrar a adequação de objetos para a **modelagem** de atividades ligadas a escritórios (OIS). Em OZ um **objeto** é uma entidade discreta que encapsula dados e um conjunto de operações (regras) para manipulação desses dados. Cada objeto possui uma interface constituída de seus nomes (que permite identificá-los univocamente em uma certa população de objetos) e instruções que estabelecem como suas operações podem ser disparadas.

O *modelo de dados OZ⁺* [Weiser-89] surgiu como uma evolução do modelo OZ, devido à necessidade de melhoria em alguns conceitos. Em OZ⁺, objetos podem ser compostos de outros objetos (objetos complexos), o que estabelece uma forte estrutura hierárquica. Objetos complexos solucionam alguns problemas do OZ, entre eles, a possibilidade de eventos serem executados paralelamente ao mesmo tempo, e a

²SGBDOO significa Sistema de Gerenciamento de Bases de Dados Orientadas a Objetos, e SGBD significa Sistema de Gerenciamento de Bases de Dados.

possibilidade da existência de objetos persistentes. OZ⁺ aceita que o usuário defina novos tipos de objetos.

2.4.2 - Modelo de Dados Orion.

O modelo de dados Orion [Banerjee-87],[Kim-89a],[Kim-90b] foi desenvolvido para suportar o projeto ATC (Tecnologia Avançada de Computadores) no MCC (Corporação de Tecnologia de Computadores e Microeletrônica) em Austin, Texas. O modelo visou suportar aplicações que envolvam CAD/CAM, IA e OIS (Sistemas de Informação de Escritórios). Todas as entidades do mundo real são modeladas como objetos. Um objeto possui uma memória privada onde é guardado seu próprio estado. Há um identificador acoplado a cada objeto, o qual consiste na identificação da classe que o contém. Todo o comportamento de um objeto é encapsulado em métodos, os quais são uma parte da definição de um objeto. Objetos comunicam-se entre si através de mensagens, as quais constituem a parte pública do objeto. Para cada mensagem interpretada por um objeto, há um método correspondente que executa essa mensagem. Objetos "similares" são agrupados em uma classe. Assim, todos os objetos pertencentes a uma classe são descritos pelas mesmas instâncias de variáveis, mesmos métodos e respondem às mesmas mensagens. Os objetos em Orion podem ser compostos.

2.4.3 - Modelo de Dados Iris.

O Modelo de Dados Iris [Wilkinson-90] foi desenvolvido para suportar a implementação de um sistema de gerenciamento de bases de dados de última geração, capaz de suportar aplicações relativas a sistemas de informação para escritórios, sistemas baseados em conhecimento, medidas e testes em engenharia, e projetos de hardware e software. Para suportar essas aplicações o modelo possui construtores que lhe proporcionam capacidade para uma rica modelagem de dados, suporte direto para inferência em bases de dados, novos tipos de dados (imagens gráficas, voz, texto, vetores, matrizes), e múltiplas versões de dados. Um objeto é uma entidade identificada por um código imutável, gerado pelo sistema. O modelo possui tipos e funções (que também são objetos). Os valores de atributos, relacionamentos e procedimentos de objetos, são modelados como funções. Os tipos possuem um nome único, sendo usados para categorizar os objetos em conjuntos. Os tipos são organizados em um grafo acíclico que representa a generalização e especialização

de forma a modelar herança.

2.4.4 - Modelo de Dados PCTE+.

O modelo de dados OMS (Object Management System) foi criado para suportar o projeto PCTE (Portable Tool Environment) [Boudier-88] projetado para permitir a construção de protótipos de ferramentas utilizando uma interface de suporte com um padrão de definição público. Cada objeto possui um tipo que é definido por um nome, um tipo pai, um conjunto de tipos de atributos, um conjunto de tipos de ligações. Relacionamentos entre pares de objetos podem ser estabelecidos. Um relacionamento é uma associação bidirecional entre dois objetos, podendo ser visto também como um par de ligações mutuamente inversas. Um tipo de ligação é definido por um nome, uma cardinalidade que diz se uma ou muitas ligações deste tipo podem partir do mesmo objeto origem, um conjunto de tipos de objetos destino e um conjunto de tipos de atributo. Tipo de relacionamento é definido por um par de tipos de ligação, com suas propriedades sendo definidas pelas propriedades do tipo de ligação. O modelo OMS foi estendido para suportar o projeto denominado PCTE+. Nessa extensão, as principais características adicionadas formam o suporte de versões (variações do esquema), herança múltipla para tipos de objetos, os quais são organizados dentro de uma hierarquia de grafo acíclico direcionado, tipos de atributos adicionais e entidades compostas.

2.4.5 - Modelo de Dados Exodus.

O modelo de dados *Extra* (Extensible Types for Relations and Attributes) foi desenvolvido para suportar o projeto Exodus [Carey-88], projetado para suportar aplicações científicas, aplicações com utilização de vozes e imagens, além de auxiliar projetos de engenharia. Os objetos são persistentes, sendo separadas as noções de tipo e instância. Os objetos podem ser complexos com sub-objetos compartilhados. O modelo possui tuplas, conjuntos, arrays de tamanho fixo e variável como construtores de tipos. São possíveis três formas de atributos: atributo 'own' que simplesmente denota um valor; atributos 'own ref', cujo valor representa uma referência a um objeto; e atributo 'ref' cujo valor é simplesmente uma referência a outro objeto, sendo sua existência independente do objeto que o referencia.

2.4.6 - Formalismo Geral para um Modelo de Dados Orientado a Objetos.

Dave Straube propôs em [Straube-90] um formalismo genérico para modelos de dados orientado a objetos, com o propósito de utilizá-lo como base para a descrição de um modelo de consultas formal. O resultado desta proposta foi o desenvolvimento de um modelo de dados formal, o qual está descrito abaixo.

Aspectos Formais

Valores e Objetos.

Todas as definições do modelo são baseadas nos seguintes conjuntos:

- um conjunto finito de domínios básicos $D_1, \dots, D_n$ onde $D = \bigcup_{i=1,n} D_i$;
- um conjunto infinito enumerável A de símbolos, chamados atributos;
- um conjunto infinito enumerável ID de identificadores;
- um conjunto finito CN de nomes de classes;
- um conjunto finito MN de nomes de métodos.

Definição 1: Valores. Existem três tipos de valores:

1. Todo $v \in D$ é um valor atômico para o qual existe uma representação textual.
2. Todo sub-conjunto finito de ID é um conjunto de valores.
3. Todo elemento pertencente a $P(A) \times P(D) \times P(ID)$ é um valor estrutural.

O símbolo V denota o conjunto de todos os valores.

Definição 2: Objetos. Um objeto é uma tripla $o = (id, cn, val)$ onde $id \in ID, cn \in CN$, e $val \in V$. $O = ID \times CN \times V$ é o conjunto de todos os objetos. É usada a notação $o.id$, $o.cn$, e $o.val$, respectivamente, para denotar o identificador, a classe, e o valor do objeto o .

Baseando-se sobre um objeto, define-se a função $ref(o)$ que associa para cada objeto o , o conjunto de todos os identificadores referenciados no interior da parte dos valores do objeto. Esta é uma função recursiva, que inclui todas as referências a objetos que compõem um objeto o .

Definição 3: Conjunto Consistente de Objetos. Um conjunto de objetos Θ é consistente se e somente se:

- $\forall o, p \in \Theta, o \neq p, o.id \neq p.id$ (dois objetos em Θ não têm o mesmo identificador).
- $\forall o \in \Theta, ref(o) \subseteq \bigcup_{p \in \Theta} p.id$ (cada identificador em $ref(o)$ é um objeto em Θ).

Classes, tipos e heranças.

Classes são tipos abstratos cujas instâncias são objetos. Neste modelo, uma classe é uma definição de tipo, assim como um sinônimo para todos os objetos que são instâncias do tipo. Uma classe define a interface e a semântica de suas instâncias, tornando público um conjunto de métodos e sua descrição (assinatura). A assinatura do método especifica o nome do método, o número e o tipo de argumentos, e seu resultado. Por exemplo, a classe Pessoa pode definir o seguinte método: **salário : Pessoa x Data $\rightarrow$ Número** cuja semântica é fornecer os valores de salário sobre o tempo de vida da pessoa.

Assinaturas fornecem os recursos necessários para que um sistema possua estrutura de tipo forte, capaz de executar verificação de tipos em tempo de compilação. A verificação é baseada na noção de semelhança. Um tipo **P** é semelhante a um tipo **Q** se:

1. **P** fornece ao menos as operações de **Q**;
2. os tipos de resultados das operações de **P** são semelhantes aos tipos de resultados das correspondentes operações de **Q**;
3. os tipos de argumentos das operações de **Q** são semelhantes aos tipos de argumentos das operações correspondentes de **P**.

Uma classe que é semelhante a outra, e que possui procedimentos ou valores adicionais, é denominada *especialização* dessa classe. Esta nova classe é denominada *subclasse*. Por exemplo, pode ser definida uma classe Estudante, como uma subclasse de uma classe Pessoa. Uma vez que um objeto Estudante implementa o procedimento de um objeto Pessoa, a classe Estudante fornece o seguinte método: **Idade : Estudante x Data $\rightarrow$ Número**. Um Estudante herda o procedimento de *Idade* proveniente de sua classe pai. Esse tipo de herança comportamental é conhecida como herança de classes. Não existem restrições sobre o número de classes das quais uma subclasse pode herdar procedimentos ou valores. No caso da herança de classes ser proveniente de múltiplos pais, a mesma é denominada *herança múltipla*.

A herança comportamental descrita, estabelece que uma subclasse sempre é semelhante à todas suas superclasses. Outros denominam essa característica de critério de *substituição*. *Substituição* estabelece que um objeto de uma classe **P** pode ser usado em qualquer contexto especificando uma superclasse de **P**.

Uma hierarquia de classes tem uma classe *raiz*, que é pai de todas as outras classes.

Definem-se *instâncias* de uma classe c como a *extensão* de c , que são denotadas como $ext(c)$. A maior extensão de c , denotada por $ext^*(c)$, é a união da extensão de cada classe na subárvore da hierarquia de classes com raiz na classe c .

Definição 4: Métodos. Um método é uma tripla $m = (n, f: S \rightarrow T, b)$ onde:

- $n \in MN$ é o nome do método.
- $f: S \rightarrow T$ é uma função que mapeia um produto de domínios fonte para um domínio alvo, da forma $f: S_1 \times S_2 \times \dots \times S_n \rightarrow T$ onde $S_1, \dots, S_n$ e T são conjuntos de objetos.
- b denota a semântica (procedimento) da função f .

As funções definidas abaixo serão necessárias nas próximas definições.

- $fonte(m)$ denota S_1 , o primeiro domínio fonte em S do método m .
- $alvo(m)$ denota o domínio alvo T do método m .
- $nome(m)$ denota o nome n do método m .

A função f é n -ária para permitir métodos parametrizados. Por exemplo, $f(s_1, \dots, s_n) = t$ quando $s_1 \in S_1, \dots, s_n \in S_n$ e $t \in T$. Usando o paradigma de orientação a objeto, a função f é aplicada ao objeto s_1 usando objetos $s_2, \dots, s_n$ como argumentos, resultando em um objeto t .

Definição 5: Conjunto Consistente de Métodos. Um conjunto de métodos M é consistente se e somente se:

$$\forall m_i, m_j \in M, nome(m_i) \neq nome(m_j).$$

Uma classe define o procedimento de suas instâncias fornecendo métodos que operam nestas instâncias. No entanto, uma classe pode adicionalmente herdar um procedimento vindo de seus pais. As funções que implementam os métodos da classe pai somente são aplicáveis para instâncias da classe pai; elas não podem ser aplicadas para instâncias de uma subclasse. Define-se agora o operador de transformação " $\circ$ ", o qual modifica métodos herdados, para que eles sejam aplicáveis a instâncias de uma subclasse.

Definição 6: Transformação de Método. Dado um método m e um conjunto de objetos $\Omega: m \circ \Omega = m' = (n', f': S' \times T', b') \mid [n' = n, S' = \Omega \times S_2 \times \dots \times S_n, T' = T, b' = b]$.

O operador de transformação toma um método m e um conjunto de objetos Ω , e cria um novo método m' com o mesmo nome, procedimento, parâmetros, e domínio alvo, mas os quais operam sobre os objetos em Ω ao invés do domínio origem do método original.

Definição 7: Classes. Uma classe c é definida como a tupla $c = (n, p, MT, AM)$ onde:

- $n \in CN$ é o nome da classe.
- p é a seqüência de classes pai da forma $\langle c_1, \dots, c_k \rangle$.
- MT é um conjunto consistente de métodos com as seguintes restrições adicionadas:

$\forall m \in MT, fonte(m) = ext(c)$. Cada método em MT precisa operar sobre todas as instâncias da classe. Em outras palavras, os métodos das classes são definidos somente sobre os objetos na extensão da classe.

$\forall m \in MT$, domínios origem $S_2(m), \dots, S_i(m), \dots, S_n(m)$ é a extensão maior de alguma classe c_i' , $ext^*(c_i')$, na base de dados. Esta restrição é baseada no princípio de substituição implicada pelo relacionamento de semelhança. Substituição estabelece que uma instância de uma subclasse é aceitável em qualquer contexto especificando uma instância da classe pai. Aqui, a classe pai é c_i' . Para encontrar o critério de substituição, a função que implementa um método precisa aceitar todas as instâncias de c_i' e todas as subclasses de c_i' como argumentos, isto é, $ext^*(c_i')$.

$\forall m \in MT$, o domínio alvo $T(m)$ é a maior extensão de alguma classe c' , $ext^*(c')$, na base de dados. Esta restrição garante que o resultado de um método esteja bem definido, permitindo que um sistema possua forte estrutura de tipo.

- AM é um conjunto de todos os métodos aplicáveis formados por $MT(c) \cup (AM(c_1) \circ ext(c)) \cup \dots \cup (AM(c_k) \circ ext(c))$.

Os métodos aplicáveis a uma classe consistem dos métodos que a classe fornece, MT , mais os métodos provenientes de seus pais transformados para operar sobre suas próprias instâncias.

Definição 8: Bases de Dados. Uma base de dados é um par $db = (H, \Theta)$ onde H é um conjunto de classes denotando uma hierarquia de classes e Θ é um conjunto consistente de objetos com as seguintes restrições:

1. $\forall c \in H$, o nome de c é único.
2. $\forall o \in \Theta$, $o.cn$ é o nome de uma classe em H .
3. $c_{raiz} \in H$.
4. $\forall c \in H$, c é uma subclasse de c_{raiz} .

As duas primeiras condições dizem que cada objeto na base de dados pertence a somente uma classe, que está na base de dados. A terceira e quarta condições são necessárias para dizer que a definição recursiva para herança comportamental é limitada para toda classe na base de dados.

Uma base de dados e suas definições de classes associadas definem um esquema fixo. Mas o modelo deve ser construído permitindo alterações no esquema. Alterações no esquema são desejáveis e devem ser permitidas; porém, quando uma consulta estiver sendo efetuada, alterações não devem ser possíveis. A definição prévia identifica um conjunto de invariantes que precisam ser mantidas durante a execução de qualquer consulta individual. As restrições são requeridas para permitir que segurança possa ser determinada e que relacionamentos entre classes e seus métodos, os quais fornecem a base para transformações de consulta, existam durante a execução da consulta.

Operações em uma base de dados.

Uma operação em uma base de dados, pode ser pensada como a aplicação da função associada com um método para uma seqüência de objetos, onde o primeiro objeto na seqüência está na extensão da classe na qual o método é definido.

Definição 9: Operação. Uma operação em uma base de dados é definida como a função $op: db \times \langle o_1, \dots, o_n \rangle \rightarrow r$, onde:

- db é uma base de dados.
- $\langle o_1, \dots, o_n \rangle$ é uma seqüência de objetos em db com ao menos um membro.
- mn é o nome de um método definido sobre a classe de o_1 .
- r é o objeto resultante obtido quando a função associada com o chamado método, $f: S_1 \times S_2 \times \dots \times S_n \rightarrow T$ é aplicada para $\langle o_1, \dots, o_n \rangle$. Intuitivamente, o método com nome mn está sendo aplicado para o objeto o_1 com parâmetros $o_2, \dots, o_n$.

O modelo de dados apresentado acima, considera objetos como instâncias de tipos abstratos de dados organizados dentro de uma hierarquia de classificação.

2.4.7 - Modelo de Dados O2.

O modelo de dados O2 [Lécluse-88] [Deux-90] foi proposto e desenvolvido pelo grupo *Altair*. O grupo *Altair*, é um consórcio de pesquisa fundado por IN2 (Subsidiária Siemens), INRIA (Institute National de Recherche en Informatique et Automatique), e LRI (Laboratoire de Recherche en Informatique, University of Paris XI). O grupo iniciou o desenvolvimento do modelo em setembro de 1986, para suportar uma implementação de um sistema de base de dados de nova geração, voltado para aplicações de negócios

tradicionais, aplicações transacionais (excluindo aplicações de performance muito alta), automação de escritórios e aplicações multi-mídia.

O2 é um dos poucos modelos de dados orientado a objetos encontrados na literatura, que apresentam uma formalização. A mesma será descrita na Seção 5.2.

2.4.7.1 - Descrição Informal do Modelo.

Objetos e Valores.

Em O2 os objetos são representados por pares *identificador-valor*. Alguns exemplos de objetos são apresentados na Figura 25.

```
i0 : tuple(nome : "Torre Eiffel",  
           endereço : i5,  
           descrição : "Um famoso monumento de Paris",  
           feriados : lista("Natal","Páscoa","15 de Agosto").  
           taxa_admissão : 25)  
  
i1 : tuple(nome : "Paris",  
           mapeamento : i2,  
           hotéis : conjunto (i3, i4))  
  
i5 : tuple(cidade : i1,  
           rua : "Champ de Mars",  
           número : 1).
```

Figura 25: Exemplos de Objetos em O2.

Na Figura 25, o primeiro objeto possui identificador i_0 e, como valor, uma tupla estruturada. Os valores podem ser atômicos ("Torre Eiffel"), valores estruturados (valores dos feriados) ou objetos, através de seus identificadores (i_1 , i_2 , etc).

Tipos e Classes.

Existem duas espécies de organização:

i.classes, cuja instâncias são objetos, as quais encapsulam dados e procedimentos;

ii. **tipos**, cujas instâncias são valores.

Valores não são encapsulados, isto é, sua estrutura é conhecida pelos usuários, e eles podem ser manipulados por operadores. A toda classe é associado um tipo, descrevendo a estrutura de suas instâncias. Classes são criadas explicitamente usando comandos pré-definidos e são parte da hierarquia de herança. Tipos aparecem como componentes de classes e são construídos recursivamente usando tipos atômicos (integer, float, double, string, char, Boolean, e bits).

Exemplo de um tipo O2:

```
tuple(nome : string,  
      mapeamento : Bitmap,  
      hotéis : conjunto(Hotel)).
```

O tipo acima descreve cidades. O atributo Hotéis tem um valor conjunto estruturado. Hotel é nome de uma classe (a classe Hotel está definida na Figura 26).

Uma classe descreve a estrutura e o procedimento de um conjunto de objetos. A parte estrutural de uma classe é um tipo e a parte procedimental é um conjunto de métodos (será definido). As classes são criadas usando comandos de definição de esquema. A Figura 26, apresenta um exemplo de criação de classes.


```
add class Cidade
    type tuple(nome : string,
               mapeamento : Bitmap,
               hotéis : conjunto(Hotel))

add class Monumento
    type tuple(nome : string,
               endereço : endereços,
               descrição : string,
               feriados : lista (string),
               taxa_admissão : integer)

add class Endereços
    type tuple(rua : string,
               cidade : Cidade)

add class Hotel
    type tuple(nome : string,
               endereço : Endereços,
               facilidades : lista(string),
               estrelas : integer,
               taxa : float)

add class Restaurante
    type tuple(nome : string,
               endereço : Endereços,
               menus : conjunto(tupla(nome:string,taxa:float))).
```

Figura 26: Criação de Classes.**Nomeação e Persistência.**

Nomeação consiste da atribuição de nomes aos objetos e aos valores.

Exemplo de definição de nomes para objetos:

```
add name Torre_Eiffel : Monumento
add name Paris : Cidade.
add name Monumentos_Paris : conjunto(Monumento).
```

O nome "Torre_Eiffel" será padrão para um objeto da classe Monumento. "Monumentos_Paris" é um nome para um valor do tipo conjunto Monumento.

Em O2, todo objeto ou valor nomeado é persistente, e todo objeto ou valor que é parte de outro objeto ou valor persistente, é persistente.

Por exemplo, considere que o nome "Torre Eiffel" foi atribuído ao objeto i_0 definido na Figura 25. Então o objeto i_1 se tornará persistente, uma vez que ele é um componente de i_0 junto com as strings "Torre Eiffel" e "Um famoso monumento de Paris", os quais são os valores dos atributos nome e descrição. Os mesmos guardam os valores dos atributos feriados e taxa de admissão pertencentes à classe Monumento.

Métodos.

A manipulação de objetos se dá através dos métodos. Um método é uma peça do código atribuído para uma classe específica, e pela qual, pode ser aplicado a objetos pertencentes àquela classe. Em O2, um método é declarado atribuindo-lhe sua assinatura, isto é, seu nome, a classe a que está atribuído, o tipo ou classe de argumentos, e o tipo ou classe de resultados, se existirem.

Exemplo de declaração de um método:

```
add method testa_taxas() : Boolean in class Hotel.
```

Este método verifica se as taxas do hotel são corretas em relação a sua categoria (atributo estrelas). Ele é unido à classe Hotel, não tem parâmetros, e retorna um valor do tipo Boolean.

Os métodos podem ser *privados* ou *públicos*. *Métodos privados* são visíveis apenas dentro de suas classes, enquanto *métodos públicos* estão visíveis para todas classes, podendo ser usados livremente.

Exemplo de declaração de um método privado:

```
add method aumenta_taxas(quantia : integer)
    in class Monumento is private.
```

Subtipos e Herança.

O2 fornece um mecanismo de herança baseado em subtipos. Um tipo é um subtipo de outro se e somente se toda instância deste tipo é também uma instância de seu supertipo. Isto permite declarações como "uma pessoa é um mamífero" ou "um empregado é uma pessoa".

Um tipo tupla é um subtipo de outro se ele contém todos os atributos de seus

supertipos e mais alguns outros e (ou) refina o tipo de alguns atributos de seu supertipo.

Exemplo:

```
tuple(nome : string,  
      endereço : endereços,  
      descrição : Texto,  
      feriados : lista(string),  
      taxa_de_admissão : integer,  
      facilidades : lista(string),  
      estrelas : integer,  
      taxa : float)
```

é um subtipo de:

```
tuple(nome : string,  
      endereço : endereços,  
      descrição : Texto,  
      feriados : lista(string),  
      taxa_de_admissão : integer)
```

O relacionamento entre os subtipos é inferido por O2. Baseado no relacionamento de subtipos, O2 oferece um mecanismo de herança. Por exemplo:

```
add class Histórico_hotel herda Monumento, Hotel.
```

Esta declaração define a classe `Histórico_hotel`, cujo tipo associado à classe é um subtipo do tipo "Monumento" e do tipo "Hotel". Na declaração, devem ser dados somente os novos atributos e aqueles que devem ser redefinidos. Os demais atributos são herdados. Assim, um objeto da classe `Histórico_hotel` será considerado como um objeto da classe `Monumento` e como um objeto da classe `Hotel`. Isto resulta em que todos os métodos das classes `Monumento` e `Hotel` podem ser aplicados aos objetos da classe `Histórico_hotel`. Esse exemplo ilustra o suporte de O2 à **herança múltipla**.

A Classe Objeto.

Existe uma classe predefinida chamada "Objeto". Essa classe é a raiz da hierarquia de classes, e toda classe herda-lhe implicitamente. Ela define métodos que são comuns para todos os objetos no sistema. Assim, estes métodos são herdados por todas as classes.

Atributos Excepcionais e Métodos Excepcionais.

Os atributos excepcionais são aqueles que, por motivos semânticos, são interessantes em uma determinada declaração. Por exemplo, considere a classe Monumento: o objeto "Torre Eiffel" pode ter um valor, o qual está contido em um atributo *peso*.

Exemplo:

```
tuple(nome : "Torre Eiffel",  
      endereço : i5,  
      descrições : "Um famoso monumento de Paris",  
      feriados : lista("Natal", "Páscoa", "15 de agosto"),  
      taxa_de_admissão : 25,  
      peso : 315).
```

O atributo peso é excepcional, uma vez que ele não aparece no tipo associado com a classe Monumento.

Os métodos excepcionais são aqueles usados para caracterizar um procedimento excepcional de um objeto.

Exemplo:

```
add method aumenta_taxa(quantia : integer)  
  in objeto Torre_Eiffel.
```

2.4.7.2 - Descrição Formal do Modelo.**Objetos.**

Na formalização dos conceitos de objetos descrita abaixo, serão utilizados os seguintes conjuntos:

- Um conjunto finito de domínios $D_1, \dots, D_n, n \geq 1$, onde $D = D_1 \cup D_2 \cup \dots \cup D_n$ e $D_1 \cap D_2 \cap \dots \cap D_n = \emptyset$.
- Um conjunto infinito enumerável A de símbolos chamados **Atributos**. Os elementos de A são nomes para campos de estruturas.
- Um conjunto infinito enumerável ID de símbolos chamados **Identificadores**. Os elementos de ID serão usados como identificadores de objetos.

Definição 1: Valores

- i. O símbolo especial Nil é um valor, chamado valor básico.
- ii. Todo elemento $v \in D$ é um valor, chamado um valor básico.
- iii. Todo sub-conjunto finito de ID é um valor, chamado conjunto de valores. Conjuntos de Valores são denotados usualmente entre chaves.
- iv. Toda função parcial finita com domínio em A e contradomínio ID, é um valor, chamado tupla-valor. É denotada por $\langle a_1:i_1, \dots, a_p:i_p \rangle$ a função parcial t definida sobre $\{a_1, \dots, a_p\} | t(a_k) = i_k$ para todo k . O conjunto de todos os valores é denotado por V.

Definição 2: Objetos.

i. um objeto é um par $o = (i, v)$, onde i é um elemento de ID (um identificador) e v é um valor.

ii. O é o conjunto de todos objetos, onde $O = ID \times V$.

Se $o = (i, v)$ é um objeto, então $\text{ident}(o)$ denota o identificador i e $\text{valor}(o)$ denota o valor v . Existe uma função ref com domínio em O e contradomínio 2^{ID} , que associa um objeto ao conjunto de todos os identificadores "ligados" aos seus valores.

Definição 3: Objeto grafo.

Se Θ é um conjunto de objetos, então o grafo $\text{grafo}(\Theta)$ é definido como segue:

i. se o é um objeto básico de Θ , então o grafo contém um vértice que não possui arcos de saída. O vértice é rotulado com o valor de o .

ii. se o é um objeto tupla-estruturada $br(i, \langle a_1:i_1, \dots, a_p:i_p \rangle)$, o grafo de o contém um vértice, por exemplo v , representado por um ponto (.) e rotulado com i , e p arcos de saída (origem em v), rotulados com $a_1, \dots, a_p$ conduzindo correspondentemente aos objetos $o_1, \dots, o_p$, onde o_k é um objeto identificado por i_k (se tal objeto existe).

iii. se o é um objeto conjunto_estruturado $br(i, \{i_1, \dots, i_p\})$, o grafo de o contém um vértice, diga-se v , representado por uma estrela (*) e rotulado por i , e p arcos de saída não rotulados (origem em v), conduzindo respectivamente para os vértices correspondentes aos objetos $o_1, \dots, o_p$, onde o_k é um objeto identificado por i_k (se tal objeto existe).

exemplo.

Seja Θ um conjunto consistindo dos seguintes objetos:

$o_0 = (i_0, \langle \text{esposa}:i_1, \text{nome}:i_3, \text{filhos}:i_2 \rangle)$

$o_1 = (i_1, \langle \text{esposa}:i_0, \text{nome}:i_4, \text{filhos}:i_2 \rangle)$

$o_2 = (i_2, \{i_5, i_6\}), o_3 = (i_3, \text{"Fred"}), o_4 = (i_4, \text{"Maria"})$

$o_5 = (i_5, \text{"João"})$, $o_6 = (i_6, \text{"Paulo"})$.

(Θ é representado pelo grafo apresentado na Figura 27:)

Figura 27: Grafo representando um conjunto de objetos.

Note-se no exemplo acima que, se um identificador i aparece em um valor, deve existir um identificador sobre ele. Assim, surge a necessidade do controle de consistência, ou seja, para cada identificador, é necessário a existência de um objeto que ele identifique.

Definição 4: Conjunto de objetos consistentes.

Um conjunto de objetos Θ é consistente se e somente se :

- i. Θ é finito;
- ii. a função ident é injetiva sobre Θ (isto é, não há par de objetos com o mesmo identificador);
- iii. $\forall o \in \Theta, \text{ref}(o) \subseteq \text{ident}(\Theta)$ (isto é, todo identificador referenciado corresponde a um objeto de Θ).

Nas definições seguintes, será usada a notação $\Theta(i)$ indicando o valor v , tal que o objeto (i, v) está em Θ .

Em sistemas orientados a objetos, é necessário distinguir entre o que é um *objeto idêntico*, e o que é um *objeto igual*, por haver distinção entre os conceitos de objetos e

valores.

Definição 5: Igualdades.

- i. **0-igualdade**: dois objetos o e o' são **0-iguais** (ou idênticos) se e somente se $o = o'$ (no senso de igualdade matemática);
- ii. **1-igualdade**: dois objetos o e o' são **1-iguais** (ou simplesmente iguais) se e somente se $\text{valor}(o) = \text{valor}(o')$;
- iii. **w-igualdade**: dois objetos o e o' são **w-iguais** (ou valor igual) se e somente se $\text{span-tree}(o) = \text{span-tree}(o')$ onde $\text{span-tree}(o)$ é a árvore obtida vindo de o recursivamente substituindo um identificador i (em um valor) pelo valor do objeto identificado por i .

Igualdade implica igualdade de valores, mas não o reverso, uma vez que muitos objetos distintos podem ter a mesma span tree.

Tipos.

Um tipo é uma abstração que permite ao usuário "encapsular" em uma mesma estrutura, dados e operações. Em *O2*, o componente estático de um tipo é chamado *estrutura de tipo*.

Definição 6: Nomes de tipos.

Bnames é o conjunto de nomes para tipos básicos contendo:

- i. os símbolos especiais *Any* e *Nil*;
- ii. um símbolo d_i para cada domínio D_i , onde $D_i = \text{dom}(d_i)$.
- iii. um símbolo x para todo valor x de D .

Cnames é o conjunto de nomes para tipos construídos o qual é infinito enumerável e disjunto em relação a **Bnames**, ou seja, $\text{Cnames} \cap \text{Bnames} = \emptyset$. **Tnames** é um conjunto de todos nomes para tipos, ou seja, $\text{Tnames} = \text{Bnames} \cup \text{Cnames}$.

Além desses conjuntos de nomes de tipos, existe um conjunto **MT** cujos elementos são chamados **métodos**, os quais executarão operações nas estruturas de dados.

Definição 7: Tipos.

Tipos Básicos (Btipos): um tipo básico é um par (n, m) onde n é um elemento de **Bnames** e m um sub-conjunto de **MT**.

Tipos Construídos (Ctipos): Um tipo construído é:

- i. uma tripla (s, t, m) onde $s \in \text{Cnames}$ e $t \in \text{Tnames}$, m um sub-conjunto de **MT**. Este tipo será denotado por $(s = t, m)$; ou
- ii. uma tripla (s, t, m) onde s é um elemento de **Cnames**, t é uma função parcial finita com

domínio em A e contradomínio em $Tnames$, e m um sub-conjunto de MT . Este tipo será denotado por $(s = \langle a_1:s_1, \dots, a_n:s_n \rangle, m)$ onde $t(a_k) = s_k$ e será chamada **tipo tupla estruturada**; ou

iii. uma tripla (s, s', m) onde $s \in Cnames$, $s' \in Tnames$, e m um sub-conjunto finito de MT . O tipo será denotado por $(s = \{s'\}, m)$ e será chamado **tipo conjunto estruturado**.

Um tipo é tipo básico ou tipo construído. O conjunto de todos os tipos é denotado por T .

Estruturas de Tipo.

Definição 8: Estruturas de Tipo.

Estrutura de Tipo Básico: seja $t = (n, m)$ um tipo básico. Então $\underline{n}$ é a estrutura tipo básico associada a t .

Estrutura de tipo Construído: seja $t = (s=x, m)$ um tipo construído. Então " $s=x$ " é a estrutura tipo construído associado a t .

Dado um tipo t , parte de sua estrutura será denotada $strut(t)$ e a parte de seus métodos por $métodos(t)$. Intuitivamente, uma estrutura de tipo é um tipo na qual a parte dos métodos está escondida, ou seja, a parte dos dados está oculta.

As seguintes notações serão usadas nas próximas definições.

1. Se t é um tipo, então $nome(t)$ é o nome do tipo, que é o primeiro componente na sua definição.
2. Se st é uma estrutura de tipo associada ao tipo t , será denominado de "nome da estrutura tipo" st , o nome de t , notando-se que $nome(st) = nome(t)$.
3. Se st é uma estrutura tipo (associado ao tipo t), será chamado "conjunto de referências de tipo para st ", será denotado por $refer(st)$ o conjunto de todos nomes de tipos juntos com a estrutura st .

Definição 9: Esquemas.

Um conjunto Δ de estruturas de tipos construídos é um esquema se e somente se:

- i. Δ é um conjunto finito;
- ii. a função $nome$ é injetiva em Δ (somente um tipo de estrutura para um dado nome);
- iii. $\forall st \in \Delta, (refer(st) \cap Cnames) \subseteq nome(\Delta)$ (isto é, não há identificadores sem objetos para identificar).

Definição 10: Interpretações.

Seja Δ um esquema e Θ um sub-conjunto consistente do conjunto universo de objetos

O. Uma interpretação I de Δ em Θ é uma função de $Tnames$ em $2^{Ident(\Theta)}$, satisfazendo as seguintes propriedades:

$Ident(\Theta)$ denota o conjunto de identificadores de todos os objetos de Θ , e que $\Theta(id)$ denota o valor v , tal que (id, v) está em Θ .

Nomes de Tipos Básicos.

1. $I(Nil) \subseteq \{id \in Ident(\Theta) / (id, Nil) \in \Theta\}$;
2. $I(d_i) \subseteq \{id \in Ident(\Theta) / \Theta(id) \in D_i\} \cup I(Nil)$;
3. $I('x) \subseteq \{id \in Ident(\Theta) / \Theta(id) = x\} \cup I(Nil)$;

Nomes de Tipos Construídos.

4. se $s = \langle a_1:s_1, \dots, a_n:s_n \rangle$ está em Δ então $I(s) \subseteq \{id \in Ident(\Theta) / \Theta(id) \text{ é um valor tupla estruturada em } a_1, \dots, a_n \text{ e } \Theta(id)(a_k) \in I(s_k), \forall k\} \cup I(Nil)$;
5. se $s = \{s'\}$ está em Δ , então $I(s) \subseteq \{id \in Ident(\Theta) / \Theta(id) \in I(s')\} \cup I(Nil)$;
6. se $s = t$ está em Δ então $I(s) \subseteq I(t)$;

Nomes de Tipos Indefinidos.

7. se s não é um nome de tipo básico, nem nome de um esquema Δ , então $I(s) \subseteq I(Nil)$.

Definição 11: Modelo de um Esquema.

- i. **Ordem Parcial sobre interpretações:** uma interpretação I é menor do que uma interpretação I' se e somente se $\forall s \in Tnames, I(s) \subseteq I'(s)$;
- ii. **Modelo:** seja Δ um esquema e Θ um conjunto consistente de objetos. O modelo M de Δ em Θ é a maior interpretação de Δ em Θ .

Intuitivamente, o modelo $M(s)$ de uma estrutura de tipo construído de nome s é o conjunto consistindo de todos objetos (identificadores de objetos) tendo esta estrutura. Por exemplo, se Θ é o conjunto

$$\{(i_0, Nil), (i_1, \{i_2, i_3\}), (i_2, 1), (i_3, 4), (i_4, \langle a:i_2 \rangle), (i_5, \langle a:i_2, b:i_3 \rangle)\}$$

e Δ é o esquema

$$\{s_1 = \langle a:Integer \rangle, s_2 = \langle a:Integer, b:Integer \rangle, s_3 = \{Integer\}\}$$

então

$$M(s_1) = \{i_0, i_4, i_5\},$$

$$M(s_2) = \{i_0, i_5\},$$

$$M(Integer) = \{i_0, i_2, i_3\}$$

$$M(s_3) = \{i_0, i_1\}$$

Prova: Definição de modelo (Def. 11). Dado um esquema Δ e um conjunto consistente de objetos Θ , existe um número finito de interpretações de Δ definidos em Θ . Por conseguinte, para provar que existe uma maior interpretação, é necessário provar que a união de duas interpretações é uma interpretação.

Sejam I_1 e I_2 duas interpretações, e I a função definida por $I(s) = I_1(s) \cup I_2(s)$, para todo nome de tipo s . Esta função I verifica as Propriedades 1,2 e 3 da definição de interpretações. Se $s = \langle a_1:s_1, \dots, a_n:s_n \rangle$ e id é um elemento de $I(s)$, então $\Theta(id)(a_k)$ está em $I_1(s_k)$ para todo k , porque I_1 é uma interpretação. Assim $\Theta(id)(a_k)$ está em $I(s_k)$ para todo k e I verifica a Propriedade 4 da Definição 10. Da mesma maneira, pode-se mostrar que I também verifica as Propriedades 5 e 6.

Por conclusão, há uma maior interpretação M e tem-se: $M(s) = \bigcup_{I \in \text{Int}(\Delta)} I(s)$, para todo nome de tipo s , onde $\text{INT}(\Delta)$ denota o conjunto de todas as interpretações de Δ (em Θ).
Ordem Parcial entre Estruturas de Tipo.

Definição 12: Ordem Parcial $\leq_{\pi}$.

Sejam s e s' duas estruturas de tipo de um esquema Δ . Diz-se que s é uma subestrutura de s' (denotada por $s \leq_{\pi} s'$) se e somente se $M(s) \subseteq M(s')$ para todo conjunto consistente Θ .

Por exemplo, se Δ consiste das seguintes estruturas de tipo:

$$s_1 = \langle a:\text{Integer} \rangle,$$

$$s_2 = \langle a:\text{Integer}, b:\text{Integer} \rangle,$$

$$s_3 = \langle c:s_1 \rangle, s_4 = \langle c:s_2 \rangle,$$

$s_5 = \{s_1\}$, $s_6 = \{s_2\}$, $s_7 = \langle a:1 \rangle$ então os seguintes relacionamentos existem entre essas estruturas:

$$s_2 \leq_{\pi} s_1 \quad s_4 \leq_{\pi} s_3$$

$$s_7 \leq_{\pi} s_1 \quad s_6 \leq_{\pi} s_5$$

O primeiro relacionamento ($s_2 \leq_{\pi} s_1$) vem da interpretação de estruturas de tipo tupla. A segunda $s_4 \leq_{\pi} s_3$ é estabelecida por definição. Seja id o identificador de um objeto pertencente para $I(s_4)$. Sabe-se da definição, que $\Theta(id)(c)$ pertence a $I(s_2)$ e portanto a para $I(s_1)$ porque tem-se $s_2 \leq_{\pi} s_1$. Conclui-se que id pertence a $I(s_3)$ e assim $I(s_4) \subseteq I(s_3)$. A desigualdade $s_6 \leq_{\pi} s_5$ pode ser estabelecida da mesma maneira e a relação $s_7 \leq_{\pi} s_1$ é obviamente verdadeira.

Teorema 1: Sejam s e s' dois tipos de estruturas de um esquema Δ . s é uma subestrutura de s' ($s \leq_{\Delta} s'$) se e somente se:

- i. ou s e s' são estruturas de tipo $s = t$ e $s' = t'$, tal que t é mais definido do que t' e para todo atributo " a " tal que t' é definido, tem-se $t(a) \leq_{\Delta} t'(a)$;
- ii. ou s e s' são estruturas conjunto $s = \{s_1\}$ e $s' = \{s'_1\}$ e tem-se $s_1 \leq_{\Delta} s'_1$;
- iii. ou $s = 'x$, s' é uma estrutura de tipo básico e x está no $\text{dom}(s')$.

Métodos.

Definição 13: Assinatura. Seja Δ um esquema. Uma assinatura sobre Δ é uma expressão da forma: $s_1 x s_2 x \dots x s_n \rightarrow s$ onde $s_1, s_2, \dots, s_n$ e s são nomes de tipos correspondendo às estruturas de tipo em Δ ou nomes de tipos básicos.

Um método m é um par $m = (n, \sigma)$, onde n é um nome de método (um elemento de MT) e σ é uma assinatura. Será denotado $\text{name}(m)$ o nome do método m e por $\text{sign}(m)$ a assinatura.

No formalismo orientado a objetos, métodos são relatados para tipos(ou estruturas de tipos), usando o primeiro argumento de sua assinatura. Este conceito está colocado abaixo.

Definição 14: Métodos. Seja $m = (n, s_1 x \dots x s_n \rightarrow s)$ um método. Diz-se que m é definido em s_1 .

Definição 15: Modelo de uma assinatura. Seja Δ um esquema e σ uma assinatura sobre Δ ($\sigma = s_1 x \dots x s_n \rightarrow s$). Se Θ é um conjunto consistente de objetos, então o modelo de σ em Θ é o conjunto de todas funções parciais vindas de $M(s_1) \times \dots \times M(s_n)$ dentro de $M(s)$ onde $M(s_k)$ é o modelo em Θ da estrutura de Δ identificado por s_k .

Considere como exemplo o seguinte esquema Δ :

Pessoa = <nome:String, idade:Integer, sexo:"masculino":>

Empregado = <nome:String, idade:Integer, sexo:String, salário:Integer>

Masculino = <nome:String, idade:Integer, sexo:"masculino">

Pessoas = {Pessoa}

Empregados = {Empregado}

Pessoa_Casada = <nome:String, esposa:Pessoa_Casada, filhos:Pessoas>

Sejam as seguintes assinaturas:

$\sigma_1 = \text{Pessoas} \times \text{Pessoa} \rightarrow \text{Boolean}$

$\sigma_2 = \text{Empregados} \times \text{Empregado} \rightarrow \text{Boolean}$

$\sigma_3 = \text{Pessoa} \rightarrow \text{Pessoa}$

$\sigma_4 = \text{Masculino} \rightarrow \text{Empregado}$

$\sigma_5 = \text{Empregado} \rightarrow \text{Integer}$

Seja o seguinte conjunto de objetos Θ com o domínio de interpretação:

$(i_0, \text{nil}), (i_1, \langle \text{nome}:i_{16}, \text{idade}:i_7, \text{sexo}:i_8 \rangle),$
 $(i_2, \langle \text{nome}:i_{17}, \text{idade}:i_9, \text{sexo}:i_{10} \rangle),$
 $(i_3, \langle \text{nome}:i_{18}, \text{idade}:i_9, \text{sexo}:i_8, \text{salário}:i_{11} \rangle),$
 $(i_4, \langle \text{nome}:i_{19}, \text{idade}:i_{13}, \text{sexo}:i_{12}, \text{salário}:i_{11} \rangle),$
 $(i_5, \{i_1, i_2\}), (i_6, \{i_3, i_4\}), (i_7, 20), (i_9, 25), (i_{11}, 130000),$
 $(i_{13}, 35), (i_8, \text{"masculino"}), (i_{10}, \text{"variado (varying)"}, (i_{12}, \text{"feminino"}),$
 $(i_{14}, \text{falso}), (i_{15}, \text{true}), (i_{16}, \text{"Smith"})$
 $(i_{17}, \text{"Blake:"}), (i_{18}, \text{"Jones"}), (i_{19}, \text{"Nash"}).$

Usando a Definição 11, pode-se elaborar estruturas de tipo definidas em Δ :

$M(\text{Pessoa}) = \{i_0, i_1, i_2, i_3, i_4\}$

$M(\text{Pessoas}) = \{i_0, i_5, i_6\}$

$M(\text{Empregado}) = \{i_0, i_3, i_4\}$

$M(\text{Empregados}) = \{i_0, i_6\}$

$M(\text{Masculino}) = \{i_0, i_1, i_3\}$

$M(\text{"masculino"}) = \{i_0, i_8\}$

$M(\text{String}) = \{i_0, i_8, i_{10}, i_{12}, i_{16}, i_{17}, i_{18}, i_{19}\}$

$M(\text{Integer}) = \{i_0, i_7, i_9, i_{11}, i_{13}\}$

$M(\text{Boolean}) = \{i_0, i_{14}, i_{15}\}$

O modelo de assinatura σ_1 é o conjunto de todas funções parciais vindas de $\{i_0, i_5, i_6\} \times \{i_0, i_1, i_2, i_3, i_4\}$ dentro de $\{i_0, i_{14}, i_{15}\}$. Intuitivamente, o modelo de assinatura σ_1 é o conjunto de funções atribuindo um objeto booleano para alguns pares (i,j) onde i é (o identificador de) um conjunto de objetos pessoa e j é (o identificador de) objeto pessoa.

Ordem Parcial entre Assinaturas.

Definição 16: Ordem Parcial $\leq_m$.

Seja Δ um esquema, f e g duas assinaturas sobre Δ . Diz-se que f é menor do que g

(ou que f refina g) se e somente se $M(f) \subseteq M(g)$ para todo conjunto consistente Δ . Esta ordenação será denotada por $\leq_m$.

Aplicando-se o esquema exemplo anterior, pode-se ver as seguintes desigualdades:

$$\sigma_2 \leq_m \sigma_1 \text{ e } \sigma_4 \leq_m \sigma_3$$

Seja Θ qualquer conjunto consistente de objetos e f uma função parcial em $M(\sigma_2)$. f é um função (parcial) vinda de $M(\text{empregados}) \times M(\text{empregado})$ em $M(\text{boolean})$. Tem-se que $\text{empregados} \leq_{st} \text{pessoas}$ e $\text{empregado} \leq_{st} \text{pessoa}$, e assim $M(\text{empregados}) \subseteq M(\text{pessoas})$ e $M(\text{empregado}) \subseteq M(\text{pessoa})$. Assim f é também uma função parcial vindo de $M(\text{empregados}) \times M(\text{empregado})$ em $M(\text{boolean})$, assim f está em $M(\sigma_1)$. Uma prova similar pode ser construída pela desigualdade $\sigma_4 \leq_m \sigma_3$. Intuitivamente, $\sigma \leq_m \sigma'$ significa que pode-se usar um método de assinatura σ' "no lugar de" um método de assinatura σ .

Teorema 2: Sejam f e g duas assinaturas sobre Δ . Então, $f \leq g$ se e somente se:

$$f = s_1 x_1 \dots x_n \rightarrow s \text{ e } g = s'_1 x_1 \dots x_n \rightarrow s' \text{ e } s_k \leq_m s'_k \text{ para } k=1,2,\dots,n \text{ e } s \leq_m s'.$$

Prova: assume-se sem perda de generalidade que as assinaturas dos métodos são da forma:

$$\sigma = s_1 \rightarrow s, \text{ e } \sigma' = s'_1 \rightarrow s'. \text{ Suponha que } \sigma \leq_m \sigma'. \text{ Toda função parcial provém de } M(s_1) \text{ para } M(s').$$

Assim, necessariamente tem-se $M(s_1) \subseteq M(s'_1)$ e $M(s) \subseteq M(s')$. Inversamente, se estas duas inclusões existem, então toda função parcial vinda de $M(s_1)$ para $M(s)$ é também uma função parcial vinda de $M(s'_1)$ para $M(s')$.

Sistemas de Tipo.

Definição 17: Sistemas de Tipos. Um conjunto de tipos π é um sistema de tipos se e somente se:

- i. o conjunto de estruturas associadas a π é um esquema;
- ii. para todo tipo $t \in \pi$, e para todo método $m \in \text{Metodos}(t)$, m é definido em uma $\text{estrut}(t)$.

Definição 18: Subtipo. Seja π um sistema de tipo e t e t' dois tipos de π . Diz-se que t é um subtipo de t' e denota-se $t \leq t'$ se e somente se:

- i. $\text{estrut}(t) \leq_m \text{estrut}(t')$;
- ii. $\forall m \in \text{Metodos}(t), \exists m' \in \text{Metodos}(t') | \text{nome}(m) = \text{nome}(m')$

e $\text{senal}(m) \leq_m \text{senal}(m')$.

Objetos Revistos.

Definição 19: Objetos Revistos. Um objeto o é uma tripla (i, v, m) onde i e v são como na Definição 2 e m é um conjunto de métodos. O primeiro componente da assinatura de todo método de m é uma estrutura tipo cuja interpretação contém o .

Bases de Dados

Definição 20: Bases de Dados.

Uma base de dados é uma tupla $(\pi, \Theta, <_{db}, \text{ext}, \text{impl})$ onde:

- i. π é um sistema de tipos, e Δ é o esquema associado,
- ii. Θ é um conjunto consistente de objetos,
- iii. $<_{db}$ é uma ordem parcial estrita entre π ,
- iv. ext é uma interpretação de Δ em Θ .
- v. impl é uma função que atribui funções para todo método m de um tipo t .

As seguintes propriedades são asseguradas:

- 1. $t <_{db} t'$ implica $t \leq t'$.
- 2. se $t <_{db} t'$ e $t <_{db} t''$ então t' e t'' são comparáveis.
- 3. $\Theta = \bigcup_{t \in \pi} \text{ext}(t)$.
- 4. $\text{ext}(t) \cap \text{ext}(t') = \emptyset$ se t' e t'' não são comparáveis.
- 5. se t é um tipo e π e m um método de t tendo assinatura $t \times \dots \times s_n \rightarrow s$, então $\text{impl}(m)$ é uma função definida ao menos de $\text{ext}(t) \times \dots \times \text{ext}(s_n)$ em $\text{ext}(s)$.

2.5 - Conclusão.

Neste capítulo foram apresentados alguns dos principais modelos de dados para bases de dados encontrados na literatura, os quais possuem uma formalização matemática associada. Deve-se ressaltar, que não foram encontradas formalizações para a maioria dos modelos orientados a objetos encontrados na literatura.

Capítulo 3

Modelo de Representação de Objetos (MRO)

3.1 - Introdução:.

Neste capítulo é apresentado o Modelo de Representação de Objetos tal como estava definido antes de ser iniciado o processo de formalização. Pode-se encontrar uma descrição mais completa do MRO em [Traina-86], [Traina-88b] e [Traina-92].

A modelagem de sistemas de gerenciamento de dados para aplicações de projeto, foi a maior motivação para o desenvolvimento do MRO. Isso o torna portanto, adequado à representação de grande conteúdo semântico, ao apoio à documentação de projetos, e à construção de Meta-Sistemas onde a especificação dos requisitos da própria aplicação deve ser mantida na base de dados.

3.2 - Representação Gráfica do Modelo.

A especificação do MRO inclui três diagramas gráficos que fornecem uma maneira gráfica para se representar modelagens feitas com o MRO. Esses diagramas são:

- **DRO** - *Diagrama de Representação de Objetos* - representa graficamente o esquema da base de dados, através da modelagem dos tipos dos elementos envolvidos na aplicação; Num DRO deve ser colocada toda informação possível que estiver sendo representada.
- **DRI** - *Diagrama de Representação de Instâncias* - representa graficamente as instâncias dos tipos modelados no esquema, ou seja, os dados efetivamente armazenados na base. O propósito desse diagrama é de servir de exemplo e, portanto, apenas as instâncias de interesse para o exemplo devem ser mostradas.
- **DHC** - *Diagrama Hierárquico de Colônias* - representa graficamente como os dados estão logicamente agrupados na base de dados.

A representação dos elementos conceituais do MRO nos diagramas será apresentada conforme forem sendo descritos os elementos que compõem o MRO.

3.3 - Descrição dos Elementos do MRO.

3.3.1 - Objetos.

O MRO modela o mundo real representando entes, eventos, coisas e conceitos através de **Objetos**. Um objeto centraliza todas as referências e informações (inclusive de identificação) que são mantidas no sistema. Todo objeto representado possui um código interno gerado pelo sistema, que o identifica univocamente na base de dados, não sendo nunca reaproveitado mesmo que o objeto deixe de existir. Um objeto pode possuir qualquer quantidade de identificadores, porém um

deles deve ser escolhido como seu **nome principal**, sendo os demais designados como **sinônimos** do objeto. Além dos identificadores, um objeto é também caracterizado por um **tipo**, o que o classifica como possuindo determinadas características em comum com outros objetos do mesmo **tipo**. O tipo de um objeto é modelado no esquema e representado graficamente utilizando um DRO, enquanto suas instâncias (ocorrências reais) são parte da base de dados e mostradas em um DRI. Na Figura 28 mostra-se como representar tipos de objeto no DRO e instâncias no DRI, juntamente com um exemplo de cada. Num DRI, o tipo de cada objeto (externamente ao círculo que representa o objeto) é indicado apenas se for de interesse aos propósitos daquele diagrama. Quando há a necessidade de se repetir num mesmo diagrama um mesmo tipo de objeto, ou de uma instância de objeto, estes devem ser representados como na Figura 29 a e b, respectivamente. Podem existir situações em que um objeto não possui um nome significativo que o identifique, ou seja, tais objetos não possuem identificador através do qual sejam identificáveis pelo usuário. Tipos de objeto

Figura 28 :Representação de Objetos.

e instâncias de objetos que não possuem identificador principal são representados como na Figura 29 c e d, respectivamente¹.

3.3.2 - Relacionamentos.

As interações que existem entre objetos no mundo real podem ser modeladas no MRO através de vários conceitos, cada um com regras e propriedades semânticas próprias. As formas de interação entre objetos no mundo real, reconhecidas pelo MRO, serão descritas nas seções seguintes. No entanto, as formas de interação que não se enquadram dentro de algum conceito do MRO podem ser representadas através de

Figura 29 : Objetos sem Identificador.

relacionamentos, cujos tipos são definidos no esquema da aplicação. São suportadas pelo MRO duas formas básicas de relacionamentos:

- **Relacionamentos Binários**, ou simplesmente **relacionamentos**, os quais sempre associam dois objetos, denominados **Objeto Origem** e **Objeto Destino**. Todo relacionamento segue um padrão definido no esquema denominado **Tipo de Relacionamento Binário**. Todo tipo de relacionamento tem um tipo oposto, e assim todo relacionamento também tem um relacionamento oposto, em que os objetos origem e destino trocam de posição. Na Figura 30 mostra-se como representar em um DRO um tipo de relacionamento binário, juntamente com um exemplo de um tipo de relacionamento binário. Na Figura 31 é mostrado como representar um relacionamento binário, juntamente com um exemplo de um relacionamento binário.

- **Relacionamento Triplo**, ou simplesmente **tripla**, onde os objetos envolvidos podem ocupar um dentre três lugares distintos, denominados **Divisão 1 (D1)**, **Divisão 2 (D2)** e **Divisão 3**

¹Objetos sem identificadores serão tratados abaixo.

Figura 30 : Tipo de Relacionamento Binário.

(D3). Todo relacionamento triplo segue um padrão definido no esquema da aplicação denominado **Tipo de Relacionamento Triplo**. Um relacionamento triplo terá sempre um objeto na divisão 1, um objeto na divisão 3, e um ou mais objetos na divisão 2, de acordo com o padrão definido pelo tipo de relacionamento triplo. Na Figura 32 está representado num DRO um tipo de relacionamento triplo, e na Figura 33 um DRI com um relacionamento triplo. Na Figura 30 e na Figura 32, os pares de números separados por um hífen (-), indicam a **multiplicidade** do relacionamento modelado. Assim, indica-se qual a multiplicidade de cada relacionamento segundo esse tipo de relacionamento. Uma multiplicidade indica a quantidade mínima e máxima de relacionamentos do tipo indicado que o objeto origem pode ter com objetos do tipo indicado como objeto destino. A seguinte convenção pode ser usada:

- 0-1 → pode existir um ou nenhum relacionamento segundo essa associação em objetos desse tipo na base de dados;
- N ou 0-N → pode existir qualquer quantidade de relacionamentos desse tipo em objetos desse tipo na base;
- 1 ou 1-1 → deve existir exatamente um desse tipo de relacionamento em cada objeto desse tipo na base de dados;

Figura 31 :Relacionamento Binário.

- 1-N → deve existir ao menos um desse tipo de relacionamento em cada objeto desse tipo na base de dados;

3.3.3 - Atributos.

Tanto objetos como relacionamentos possuem **Atributos**. Todos eles possuem um tipo e um conjunto não vazio de valores. A cada tipo de atributo está associada uma característica e o tipo de dado que ele contém. Um tipo de dado pode ser um inteiro, real, byte, cadeia de caracteres, etc. Todo atributo segue o padrão definido no esquema da aplicação para o seu tipo de atributo. A característica de um atributo de determinado tipo define o conjunto de operações que um Gerenciador de Dados pode efetuar sobre esse atributo. O MRO suporta as Características de Tipos de Atributo que são mais comumente

necessárias a um Sistema de Gerenciamento de Dados para Engenharia, as quais são as seguintes:

- **Sinônimo.** Corresponde aos identificadores alternativos dos objetos, quando estes possuem dois ou mais identificadores. Atributos desse tipo somente podem ser associados a objetos.

Figura 32 :Tipo de Relacionamento Triplo.

- **Comentário.** Corresponde a campos longos, ou seja, informações para o usuário tratadas apenas de forma textual, que não são analisadas pelo SGBD;
- **Regra.** São atributos que possuem uma condição e uma ação. A condição permite que a ação seja executada sempre que for feita alguma operação 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 determinada

Figura 33 :Relacionamento Triplo.

- pelo sistema, solicitadas pelos aplicativos para execução imediata ou postergada;
- **Tempo.** Indicação de tempo, absoluto ou relativo, atribuído aos objetos e relacionamentos. Permite que se registre o tempo absoluto de operações na base, ou o tempo do mundo real (ou de validade) definido externamente à base;
 - **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);
 - **Gráfico.** Corresponde à informação 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 que representam figuras definidas através de valores associados à coordenadas cartesianas (pixels, raster, quad-tree, etc);
 - **Som.** Corresponde a sons digitalizados armazenados na base de dados;

- **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 relacionamento ao qual o atributo está agregado

A cada característica do tipo de atributo é associado um símbolo que o identifica graficamente em um DRO ou DRI. Estes símbolos estão representados na Figura 34 .

Tipos de Atributos que não tenham alguma característica especial que os identifique têm característica de **Propriedade**: são os atributos "comuns" dos objetos ou relacionamentos. Por simplicidade, *atributos com característica de alguma coisa* podem ser referenciados apenas como *alguma coisa*. Por exemplo, atributos com característica de Propriedade podem ser referenciados apenas como Propriedades.

Para tornar a modelagem mais natural, um conjunto de elementos de modelagem mantém na base de dados relacionamentos e atributos que não precisam ser explicitados pelo usuário. Tais relacionamentos e atributos são ditos **intrínsecos**. Um exemplo é a informação de qual usuário é responsável pela criação de um objeto.

Atributos são essencialmente objetos cuja existência e comportamento são reconhecidos a priori pelo modelo. Esse tratamento apresenta duas vantagens: facilita a construção de um gerenciador aumentando-lhe a eficiência; e torna o processo de modelagem de um sistema real mais intuitivo. As mesmas duas razões aplicam-se ao conceito de relacionamento definido pelo usuário: um relacionamento pode ser representado como um objeto que possui dois atributos: origem e destino, cujos valores são, respectivamente, os objetos origem e destino do relacionamento. Porém, objetos assim possuem um significado que pode ser compreendido pelo gerenciador, e portanto facilita seu desenvolvimento e aumentam a eficiência de utilização do conceito.

Indicação da Característica

- = sinônimo;
- ⊙ = tempo;
- = estrutura de dados;
- ◆ = procedimento;
- = comentário;
- ◇ = regra;
- ⊙ = visualização;
- ⊞ = imagem;
- △ = gráfico
- = propriedade;

Indicação do Domínio

contínuo:mínimo/máximo;
discreto: $v_1, v_2, \dots, v_n$.

Figura 34 :Símbolos utilizados nas características de atributos no MRO.

Figura 35 :Tipo de atributo e atributo.

Embora relacionamentos definidos pelo usuário sejam um caso particular de objetos, eles representam uma forma genérica de representação de interações que ocorrem no mundo real. Algumas dessas interações são contempladas no MRO através de construções predefinidas, o que evita a necessidade de se definir os tipos de relacionamentos envolvidos, e permite repassar para o gerenciador o tratamento da semântica correspondente. Os relacionamentos definidos pelo usuário devem ter seu tratamento semântico especificado no esquema da aplicação, e devem ser suportados (em parte) por ela. Na Figura 35 estão representados um atributo (DRI) e um tipo de atributo (DRO), ligados a um objeto e a um relacionamento.

3.3.4 - Colônias de Objetos.

Uma **colônia de objetos** é um conjunto de objetos logicamente agrupados segundo diversos aspectos de modelagem. Os objetos que pertencem a uma colônia são ditos **habitantes** dessa colônia. Cada objeto **habita** exatamente uma colônia. Entre os conceitos envolvidos pelo conceito de colônias, estão a restrição do escopo de abrangência de um identificador de um objeto, e o estabelecimento de um relacionamento intrínseco de composição entre objetos da base de dados.

Quando visto de maneira global, um sistema apresenta um pequeno grau de detalhe. Quando o interesse recai em um de seus componentes, o grau de detalhe desse componente aumenta. Uma ferramenta de comunicação com o usuário irá interpretar um identificador, levando em consideração somente o conjunto de informações que está sendo examinado, e exigindo que qualquer referência a outras informações tenha um teor de identificação maior.

Em termos de acesso a objetos, isso corresponde a ter poucos objetos acessíveis enquanto se manipula a base de um ponto de vista global. Indicando ao sistema que pretende-se considerar um desses objetos, indica-se um grupo de objetos do qual pretende-se obter ou manipular informações. Assim, tornam-se disponíveis novos objetos. Esse procedimento pode ser repetido, indicando-se para consideração um dos novos objetos tornados disponíveis, o que aumenta sucessivamente o nível de detalhe, e correspondentemente a quantidade de objetos a que se tem acesso imediatamente, através de seus Nomes (ou Sinônimos). Com o refinamento de um determinado aspecto, os demais não deixam de estar acessíveis, mas não são por sua vez refinados.

O acesso é imediato quando se acessa um objeto através de seu código, pois este é único em toda a base de dados. Porém, pode existir mais de um objeto do mesmo tipo com um mesmo identificador. Para identificar um objeto univocamente pelo seu nome, é necessário especificar o **Contexto** onde ele está definido. O conceito de contexto envolve a partição dos identificadores existentes na base de dados em **Colônias de Objetos**, de maneira que um nome esteja definido somente no domínio de uma Colônia.

Como exemplo, considere-se que numa Base de Dados para armazenar Informações Geográficas, está a informação de quais Países e Continentes existem. Quando leva-se em consideração mais detalhes de um País, não aumentam correspondentemente os detalhes dos Acidentes Geográficos do Continente onde situa-se esse País, mas tão somente os detalhes do próprio País (por exemplo, seus estados). Se desejado, o usuário pode aumentar o nível de detalhe de um Acidente Geográfico de sua escolha, **identificando-o para consideração**. Assim este terá os detalhes de um país e os de um Acidente Geográfico

simultaneamente, mas sempre apenas um de cada.

Esse conceito estabelece também uma partição no conjunto de Tipos de Objetos, cada partição fazendo parte de um **Tipo de Colônia**. Cada Tipo de Colônia determina um conjunto de tipos de objetos. Assim objetos de um dado tipo somente podem habitar colônias de um único tipo.

Os objetos que habitam uma colônia tornam-se disponíveis quando a colônia é **aberta para acesso**. Das várias instâncias de colônias de cada tipo que podem existir, apenas uma pode estar aberta num dado instante. Uma das colônias terá que estar permanentemente disponível, e será denominada **Colônia Global**. Os objetos da Colônia Global são os que estão disponíveis quando se está no nível menos detalhado de acesso ao sistema, e permanecem sempre disponíveis. Correspondentemente existe um **Tipo de Colônia Global**, o qual contém os tipos de objetos que existem na Colônia Global. Dessa forma, todos os objetos de tipos que pertencem à Colônia Global estarão permanentemente disponíveis. Numa base de informações geográficas todos os objetos de tipo País, Acidente Geográfico e Continente pertencem à colônia global e sempre ficam disponíveis. Deve-se notar que, a menos da colônia global, que é única, qualquer outro tipo de colônia pode ter qualquer número de instâncias.

O conceito de colônia é distinto do conceito de classes de objetos: enquanto uma classe corresponde a todos os objetos de um mesmo tipo, uma colônia inclui objetos de diversos tipos diferentes, e (a menos da colônia global) não exaure o conjunto dos objetos de cada um dos tipos que a habitam.

Os tipos de colônias organizam-se segundo uma hierarquia, em que a colônia global é o topo da hierarquia, o nível imediatamente subordinado é constituído pelas colônias constritas por objetos de tipos que habitam a colônia global, e assim sucessivamente. Essa hierarquia é formada através de um relacionamento intrínseco de composição entre o objeto que constringe a colônia e os objetos que a habitam. Na Figura 36 encontra-se representado um Diagrama Hierárquico de Colônias (D.H.C.). São mostrados três tipos de colônias: o tipo de colônia global, onde habitam os objetos de tipo tob_1 , tob_2 , tob_3 , tob_4 e tob_5 , e os tipos de colônias **colônia A** e **colônia B**, constritos por objetos de tipo tob_2 e tob_4 , respectivamente.

3.3.5 - Versões e Alternativas.

O conceito de colônias permite também a armazenagem de versões e alternativas de

Figura 36 Diagrama Hierárquico de Colônias.

partes de um projeto, permitindo a geração de **Instâncias de Colônias de Objetos**. Uma colônia pode ter várias instâncias, sendo cada instância chamada de uma **Variante**. Podem existir: variantes que permitem alterações no conteúdo da colônia, as quais correspondem a **Alternativas** de projeto ainda em desenvolvimento; ou variantes já congeladas, denominadas **Versões**, as quais não podem mais ser alteradas.

3.3.6 - Meta-Esquema de Gerenciamento de Dados.

O esquema de dados de uma aplicação é um caso particular de uma colônia cujo conteúdo define a estrutura de dados de outras colônias. Assim, um tipo de objeto é apenas um objeto que habita o esquema que rege as colônias onde os objetos habitam. O tipo do objeto é um meta-tipo de objeto, reconhecido pelo sistema como o Meta-esquema, ou seja,

Figura 37: Meta-Esquema do MRO.
a modelagem do MRO feita no próprio MRO, a qual é reconhecida implicitamente pelo gerenciador. A Figura 37 mostra um Diagrama de Representação de Objetos com o Meta-esquema do MRO, onde todos os seus conceitos são apresentados graficamente. Sendo uma colônia, os esquemas podem possuir alternativas e podem evoluir com a aplicação, desde que algumas regras sejam seguidas [TRAINA_88a]. Um esquema é esquematizado pelo meta-esquema, e as demais informações armazenadas em uma base de dados (que não a

colônia esquema) são esquematizadas pela colônia esquema.

3.3.7 - Identificadores Nativos e Estrangeiros.

Quem habita uma colônia é o objeto em si e não o seu identificador. Assim, objetos sem identificadores também habitam colônias segundo seus tipos, da mesma maneira que os objetos com identificadores. Os identificadores de objetos são acessíveis no contexto das colônias abertas; assim, diz-se que o identificador principal de um objeto "co-habita" a colônia onde o objeto identificado habita. Note que objetos sem identificadores simplesmente não têm um identificador principal na colônia onde habitam. O identificador principal deve co-habitar a mesma colônia que o objeto identificado. Por outro lado, sinônimos de objetos podem ser colocados em outras colônias, de maneira que possam ser referenciados naquelas colônias mesmo sem a explicitação do contexto do objeto em si. Essa característica é utilizada quando deve-se modelar o fato de que um objeto local a uma colônia, seu significado, e sua existência é reconhecida mesmo fora dos "domínios" desse contexto. Esse fato é modelado criando-se um sinônimo do objeto em cada colônia onde esse objeto deva ser imediatamente reconhecido. Sinônimos assim são ditos ser um identificador estrangeiro da colônia onde são colocados, pois se referem a um objeto que não pertence àquela colônia. Por outro lado, o identificador principal de um objeto, e os demais sinônimos colocados na mesma colônia onde o objeto habita são ditos identificadores nativos da colônia.

3.3.8 - Objetos identificados através de relacionamentos.

Existem casos em que um objeto não possui um identificador significativo para identificá-lo. Partindo dessa situação, duas situações podem ocorrer: um objeto não possui identificador e é designado **Objeto sem Identificador**; ou um objeto é reconhecido por um relacionamento existente na base de dados, sendo designado **Objeto Identificado por Relacionamento**. Assim, dois outros objetos, por se relacionarem identificam tal objeto. O uso de uma ligação entre um objeto e um relacionamento, com o objetivo de se fazer a identificação do objeto através do relacionamento facilita a correlação de dados dos objetos envolvidos pelo relacionamento (Objetos Ascendentes) com os dados do objeto identificado pelo relacionamento (Objeto Descendente). Desse modo, um objeto ser identificado por um relacionamento corresponde ao fato de o objeto descendente *herdar*

dados dos objetos ascendentes, o que é umas das formas de herança de informações a partir de relacionamentos. Define-se que um relacionamento pode identificar apenas um objeto, e o tipo de objeto que um tipo de relacionamento pode identificar também é único. Pode ocorrer que um objeto identificado por relacionamento tenha um identificador principal.

3.3.9 - Relacionamentos de Generalização e Sumarização.

Tipos de Objetos e Tipos de Relacionamentos podem ser especializados em **Subtipos de Objetos** e **Subtipos de Relacionamentos** (por simplicidade, adota-se o termo subtipo para indicar um "subtipo de um determinado tipo"). Subtipos são mantidos através de relacionamentos de generalização intrínsecos. Um Subtipo é especificado através da indicação de:

- um tipo, ou um conjunto de subtipos já previamente especificado, que será denominado Tipo Mestre do Subtipo;
- um predicado, que estabelece o critério de como escolher os objetos do Tipo Mestre que constituirão o conjunto de objetos daquele subtipo, através da indicação de atributos e relacionamentos;
- um conjunto de valores de atributos e de relacionamentos padrão que podem ser usados para inicializar os valores ainda não definidos de um objeto desse subtipo, quando este é criado.

Um subtipo de objeto indica que um objeto, além de ser subtipo, é também do tipo básico do qual o subtipo faz parte. Assim, o conjunto de todos os objetos de um dado subtipo é um sub-conjunto do conjunto de todos os objetos daquele tipo. Um mesmo objeto pode fazer parte de mais de um desses sub-conjuntos. Na Figura 38 está representado um subtipo de objeto no DRO. Uma instância de um subtipo é representado em um DRI como um objeto comum.

Na Figura 39 está indicado um subtipo de relacionamento. Nessa figura está representado um subtipo de relacionamento entre dois subtipos de objetos, embora os subtipos de relacionamentos possam existir igualmente entre tipos de objetos.

Por outro lado, alguns objetos podem ser semanticamente agrupados com outros, e coletivamente se caracterizarem como **supertipos**. Assim, subtipos representam um mecanismo para a representação de relacionamentos de generalização, ao passo que supertipos correspondem à existência de relacionamentos de sumarização entre objetos. Um subtipo representa uma especialização do objeto mestre, sem que isso represente a

Figura 38: Representação de um Subtipo de Objeto.

existência de um novo objeto; ao passo que um supertipo constitui um novo objeto que sumariza coletivamente os objetos a que se refere. Todos os objetos que pertencem a um conjunto de objetos que formam um supertipo devem necessariamente ser de tipos que habitam colônias de objetos de um único tipo. Tanto tipos de objetos quanto de relacionamentos possuem subtipos, porém apenas tipos de objetos possuem supertipos. Da mesma maneira que os objetos, os tipos de relacionamentos também podem ter subtipos, os quais são caracterizados pelos atributos de seus tipos de relacionamento mestres.

Os supertipos podem ter relacionamentos com tipos e subtipos, bem como atributos. Esses elementos ligados a um supertipo caracterizam relacionamentos ou atributos que são identificáveis para todos os objetos de um dado supertipo, ou que caracterizam o supertipo como um todo.

As propriedades de um supertipo que não são herdadas pelos objetos que constituem os conjuntos que compõem o supertipo, são obtidas a partir do conjunto de todos os objetos

Figura 39: Representação de um Subtipo de Relacionamento.

que constituem cada tipo que compõem o supertipo. Assim, cada uma dessas propriedades deve ser definida através de uma operação especificada no domínio do tipo da característica daquela propriedade como por exemplo somatório, mínimo, máximo, média, etc, para uma propriedade com tipo de dado numérico.

Note-se que os termos "Supertipo", "Tipo" e "Subtipo" não indicam a existência de uma hierarquia entre eles. Um Tipo caracteriza objetos que efetivamente são armazenados na base de dados, sendo básicos em qualquer Base de Dados. O conceito de Subtipo estabelece sozinho uma rede hierárquica não estrita, formada por relacionamentos de generalização, onde os tipos aparecem em algum nível, que não é necessariamente o mais sintético, uma vez que podem existir subtipos que são resultado da união de tipos com subtipos. O conceito de supertipo por sua vez não tem nada a ver com a hierarquia que os subtipos estabelecem, correspondendo sim ao conceito de sumarização das propriedades de um conjunto de tipos e subtipos.

Ao contrário de um subtipo, um supertipo caracteriza um novo "tipo" de objeto. São portanto objetos com existência própria, efetivamente armazenados na base de dados, que associam-se aos objetos contidos nos conjuntos que representam através de relacionamentos intrínsecos de sumarização.

Deve ser destacado, também, que os conceitos de supertipos e colônias de objetos são independentes entre si. O conceito de colônias, representando relacionamentos de composição, diz respeito ao agrupamento de objetos que estão logicamente caracterizados pelas informações do objeto que constrange cada colônia. Já o conceito de supertipo envolve os conjuntos de objetos que têm propriedades semelhantes e distinguíveis dentro de uma colônia de objetos, independentemente da existência de objetos que forneçam um meio de caracterizar (ou identificar) cada objeto individualmente.

3.4 - Conclusão.

O MRO incorpora conceitos de orientação a objetos [Atkinson-89], [Kim-90a], [Stonebraker-90] amplamente defendidos na literatura como de grande valia às aplicações a que ele se destina suportar, tal como hierarquias de generalização e composição, encapsulamento e herança de informações, identidade independente de identificador definido pelo usuário, etc, que o tornam um modelo adequado a tais aplicações. Por outro lado, incorpora também alguns conceitos típicos de bases de dados convencionais, tal como esquema de dados, construções para acesso concorrente, atributos implementando tipos de dados predefinidos, relacionamentos como semântica definida pelo usuário, etc, os quais ensejam uma implementação mais eficiente e em alguns casos mais semanticamente aproximada das aplicações que o utilizam, o que permite também aproveitar-se de técnicas de implementação já desenvolvidas para modelos convencionais.

No Capítulo 4 serão apresentadas algumas alterações efetuadas sobre o MRO durante o processo de formalização. No Capítulo 5 será apresentada a formalização desenvolvida para o modelo, e no Capítulo 6 uma melhor interpretação e justificativa para as mudanças.

Capítulo 4

Mudanças no MRO

4.1 - Introdução.

Durante o trabalho de formalização do MRO algumas modificações foram realizadas e, como produto dessas mudanças, o MRO modificado e formalizado foi "batizado" MRO*. Além das mudanças provenientes do processo de formalização, sentiu-se necessidade de modificar alguns de seus conceitos originais. Para a elaboração desse trabalho julgou-se mais interessante que o MRO* refletisse também as modificações que estavam sendo cogitadas, de maneira que o MRO* seja um modelo já incorporando as modificações que o tornam um modelo melhorado. Neste capítulo serão apresentadas todas as mudanças realizadas¹.

4.2 - Relacionamento entre Objetos.

As interações que existem entre objetos no mundo real podem ser modeladas no MRO através de vários conceitos, cada um com regras e propriedades semânticas próprias. As formas de interação entre objetos que não se enquadram dentro de algum conceito do MRO podem ser representadas através de relacionamentos. É possível representar relacionamentos binários e triplos. Um relacionamento binário envolve dois tipos de objeto, enquanto o relacionamento triplo é uma particularização semântica de relacionamentos que envolvem múltiplos objetos.

Um relacionamento binário é uma instanciação de um tipo de relacionamento binário definido no esquema da base. O tipo de relacionamento é composto por dois tipos de objeto chamados tipo de objeto origem e tipo de objeto destino. Já o relacionamento binário é composto por um objeto origem e um objeto destino, os quais são instanciações do tipo de objeto origem e destino do tipo de relacionamento, respectivamente. Veja os

¹No decorrer do texto, será escrito MRO como referência ao modelo original sem mudanças; e MRO* como referência ao modelo alterado e formalizado.

Figura 40 : Tipos de relacionamentos binários

exemplos da Figura 40 : o elemento viaja para é oposto a recebe viajante, habita é oposto a habitada por. Note que mesmo os tipos de objetos sendo iguais, os tipos de relacionamentos binários são diferentes. Observe também que a semântica que caracteriza isso está centrada no fato dos elementos viaja para e habitada por serem diferentes. Durante o processo de formalização, foi necessário introduzir-se um novo elemento que representasse o conjunto de estruturas sintáticas e semânticas envolvidas nos designadores de relacionamento (tal como viajar e habitar). Devido à importância desse elemento e ao fato de não possuírem um nome que os identificasse claramente no modelo, foi-lhe atribuído o nome de modalidade de relacionamento de primeiro grau². A justificativa para isso é apresentada no Capítulo 6.

Assim, os tipos de relacionamento binário ficaram definidos como possuindo um tipo de objeto origem, um tipo de objeto destino e uma modalidade de primeiro grau. Portanto, na Figura 40, observa-se dois tipos de relacionamentos binários diferentes; nos dois, os tipos de objeto são Pessoa e Cidade, mas as modalidades são viaja para no primeiro e habita no segundo, diferenciando-os.

Assim o conceito de Tipo de Relacionamento no MRO foi dividido nos conceitos de Tipo de Relacionamento e Modalidade de Relacionamento de primeiro grau no MRO*. Um

²Por efeito de simplicidade, esse conceito será referido apenas como modalidade de primeiro grau.

tipo de relacionamento indica que objetos de dois tipos se associam de determinada maneira, indicada pela modalidade de relacionamento de primeiro grau. Já este segundo conceito, indica o significado de uma associação entre objetos, sem importar os tipos desses objetos. Note-se que um tipo de relacionamento é instanciado em relacionamentos, mas uma modalidade de relacionamento de primeiro grau não é instanciada - ela apenas caracteriza ambos os tipos de relacionamentos e os relacionamentos.

No MRO definia-se que relacionamentos poderiam ter atributos associados, e seguindo o esquema, em cada tipo de relacionamento eram determinados os tipos de atributos cuja associação lhes era permitida. Com desmembramento de tipo de relacionamento em tipo e modalidade, a especificação de como a associação de tipos de atributo é realizada também foi alterada. Como o conceito de relacionamento não foi alterado, a associação de atributos a relacionamentos não foi modificada.

No MRO* a associação de tipos de atributos a tipos de relacionamento é feita da seguinte maneira: como a modalidade de primeiro grau é a portadora da semântica dos relacionamentos, a associação de tipos de atributos é vinculada a ela. Por exemplo aquele que *viaja* tem uma rota, não importa se é uma pessoa que viaja para uma cidade, ou um navio que viaja para um porto. Toda modalidade de relacionamento de primeiro grau tem uma modalidade oposta, que compõe o tipo de relacionamento oposto (e o relacionamento oposto). No MRO* foi definido como sendo indiferente vincular um tipo de atributo a uma modalidade, ou a sua oposta. Os atributos de um relacionamento são os mesmos de seu oposto, e são dos tipos vinculados à modalidade desse relacionamento ou à modalidade oposta a esse relacionamento.

Um relacionamento triplo é uma instanciación de um tipo de relacionamento triplo definido no esquema da base. Ele é composto por três divisões designadas D1, D2 e D3, onde: existe um objeto em D1 e um objeto em D3; existem um ou mais objetos em D2. O objeto da divisão D1 forma com o objeto da divisão D3, um relacionamento chamado relacionamento de determinação. O mesmo objeto da divisão D1, forma com os objetos da divisão D2 relacionamentos chamados relacionamentos de conjunto. Na Figura 41 encontra-se um exemplo de um tipo de relacionamento triplo. No MRO ele seria definido como possuindo um tipo de relacionamento de determinação, o qual é composto por um tipo de objeto D1 (no caso Professor) e um tipo de objeto D3 (no caso Disciplina); dois tipos de relacionamentos de conjunto, no qual um é composto por tipo de objeto D1 como origem (no caso Professor) e pelos tipos de objetos D2 como destino (no caso Retro-Projetor e Quadro-Negro). No MRO esses elementos bastavam para definir o relacionamento triplo. Como o tipo de relacionamento triplo é composto por tipos de relacionamentos binários, os elementos usa, usado por, ministra e ministrada por recebem naturalmente o nome de

Figura 41 : Tipos de relacionamentos triplos

modalidade de primeiro grau. Porém, a ligação entre esses relacionamentos não possui um nome, e nota-se que o significado da modelagem está centrada nessa ligação. No exemplo da Figura 41, ela implicitamente carrega o significado usa para ministrar. Essa ligação carrega o significado do tipo do relacionamento triplo. Segundo uma lógica de composição, usa para ministrar é composta pelas modalidades de primeiro grau usa e ministra. No MRO³, essa "ligação" passou a ser chamada **modalidade de relacionamento de segundo grau**³. Foi estudada e reconhecida uma composição lógica para a modalidade de segundo grau, que passou a ser uma composição das duas modalidades de primeiro grau envolvidas. Isso gerou um conjunto denominado Conjunto Normal de Modalidades de Segundo Grau, o qual contém todas as composições possíveis entre duas modalidades de primeiro grau para formar uma respectiva modalidade de segundo grau.

Todo relacionamento é formado por um relacionamento binário de determinação

³Por efeito de simplicidade, esse conceito será referido apenas como modalidade de segundo grau.

e um ou mais relacionamentos binários de composição. A princípio existe a possibilidade de vincular-se atributos tanto ao relacionamento triplo quanto aos relacionamentos binários de que é composto. Dada a biunivocidade entre cada triplo e seu relacionamento de determinação, não foi definido que relacionamentos triplos possam ter atributos, vinculando esses atributos ao relacionamento binário de determinação.

4.3 - Estado de um Tipo de Atributo.

Todo atributo é uma instanciación de um tipo de atributo, o qual possui uma característica. Desde a elaboração do MRO, foram identificadas em cada tipo de atributo situações que determinam a cada momento um estado para ele, o qual indica seu comportamento corrente frente às suas instâncias na base. Particularmente neste trabalho, foi identificado o significado e a forma de tratamento de estados para o atributo com característica de sinônimo e para propriedades, as quais foram formalizadas. As características restantes devem ser formalizadas futuramente como parte de seu desenvolvimento. As alterações decorrentes da introdução do conceito de estado para os atributos com característica de **Propriedade** e **Sinônimo** são descritas a seguir.

4.3.1 - Identificadores de Objetos.

Quando um objeto é modelado, define-se para ele um tipo, um conjunto de identificadores e um código único que é gerado pelo sistema. O conjunto de identificadores pode possuir qualquer quantidade de elementos, ou ser vazio. No MRO, se um objeto possuir pelo menos um identificador, um deles deveria ser escolhido como **nome principal**, sendo os outros designados como sinônimos. Aplicando o novo conceito de estado do tipo de atributo com característica de sinônimo, no MRO* todos os identificadores de um objeto são atributos com característica de sinônimo, os quais possuem estado. Assim, um dos tipos de atributo com característica de sinônimo poderá estar com estado indicando-o como corrente, caracterizando-se como identificador principal do objeto, enquanto os outros sinônimos não serão considerados nomes principais. Isso implica que todo objeto que possa ser identificado pelo usuário deve ter ao menos um atributo com característica de sinônimo. Assim o conceito de nome principal do MRO passa a indicar qual é o sinônimo corrente para o nome principal no MRO*.

Para a característica de **sinônimo**, o estado depende do tipo de objeto em questão. Ou seja, cada tipo de objeto tem um dos seus tipos de sinônimo indicado como nome

principal.

Esse conceito pode ser generalizado, e ser estabelecida uma ordem entre um sub-conjunto dos tipos de **sinônimo** associados a um tipo de objeto, de maneira que o primeiro desses tipos de **sinônimo** é tomado como o nome principal de objetos do tipo em questão que tenham **sinônimo** desse tipo. Se esse objeto não tiver **sinônimo** desse tipo, usa-se o **sinônimo** do segundo tipo daquele conjunto ordenado; se não houver **sinônimo** desse tipo tem-se o seguinte e assim sucessivamente até esgotar o conjunto. Nesse caso tem-se um objeto sem nome principal (o que não impede que ele tenha outros sinônimos).

4.3.2 - Propriedades.

O estado de uma propriedade está associado diretamente a essa propriedade, e constitui-se na **unidade** e na **precisão** dessa propriedade. A unidade permite que valores possam ser indicados naquela unidade como corrente; a armazenagem do valor é sempre feita na unidade normal (deve ser definida qual é a unidade normalizada de cada propriedade e qual a forma de conversão entre a unidade normal e cada unidade em geral de cada propriedade); a recuperação de um valor é sempre feita na unidade corrente. A forma de conversão depende do tipo de dado da **propriedade**. Para atributos numéricos, a forma pode ser uma expressão numérica; para atributos textuais e os não numéricos em geral, a forma pode ser uma tabela. A precisão diz respeito à "granularidade" do valor que pode ser manipulado pelo atributo, ou seja, mesmo que uma unidade seja inteira, pode ser que apenas valores múltiplos de 5 tenham significado, e portanto apenas valores múltiplos de 5 devem ser armazenados. Deve-se verificar se a precisão deve ser dada sempre em relação à unidade corrente, ou à propriedade quando atribuída a um tipo de objeto. Note-se que ao contrário de sinônimos, o estado de uma propriedade é independente do tipo de objeto ao qual está associada.

4.4 - Sumarização.

No MRO, um tipo de objeto modelado como resultado de um relacionamento de sumarização era denominado supertipo de objeto. Como muitas vezes um usuário é levado a pensar que um supertipo pode ser o oposto de um subtipo (não é), o nome do mesmo foi alterado passando a ser designado objeto de sumarização.

Além dessa modificação, foram identificadas algumas propriedades que podem ser

obtidas a partir do conjunto de todos os objetos que constituem cada tipo que compõe a sumarização. Essas propriedades foram definidas como sendo formadas por pares ordenados compostos de tipos de atributos e naipes. Esse tipo de atributo é um tipo de atributo que está associado ao tipo de objeto sumarizador e é denominado **tipo de atributo sumarizador**. Já o naipe corresponde a uma operação que permite obter como resultado um valor, que é atribuído ao tipo de atributo sumarizador, sendo calculado sobre todas as instâncias, dependendo do naipe de um particular tipo de atributo ou tipo de objeto, de todos os objetos que compõem a sumarização. Os naipes possíveis são:

- **totalização** é aplicável a um tipo de atributo, e resulta no valor da soma de todos os valores de cada instância desse tipo de atributo que ocorreu em cada objeto que compõe a sumarização.
- **contagem** é aplicável a um tipo de objeto, e resulta no número de objetos desse tipo que compõe a sumarização.
- **média** é aplicável a um tipo de atributo, e resulta na média de todos os valores de cada instância desse tipo de atributo que ocorreu em cada objeto que compõe a sumarização.
- **mínimo** é aplicável a um tipo de atributo, e resulta no menor valor entre todos os valores das instâncias desse tipo de atributo que ocorreu em qualquer dos objetos que compõem a sumarização.
- **máximo** é aplicável a um tipo de atributo, e resulta no maior valor entre todos os valores das instâncias desse tipo de atributo que ocorreu em qualquer dos objetos que compõem a sumarização.
- **função livre** é aplicável a um conjunto de tipos de atributos, e resulta em valores obtidos a partir de operações aritméticas, aplicadas em todos os valores de cada instância desse conjunto de tipos de atributos que ocorreram em todos os objetos que compõem a sumarização.

4.5 - Subtipos.

Existem subtipos de objetos e subtipos de relacionamentos. Um subtipo de objeto indica um refinamento de um tipo de objeto ou de um outro subtipo de objeto. Um subtipo de relacionamento indica um refinamento de um tipo de relacionamento ou de um subtipo de relacionamento, através do refinamento da modalidade de primeiro grau em uma

submodalidade de relacionamento de primeiro grau. Destaque-se que um subtipo de relacionamento somente ocorre se for caracterizado por uma submodalidade. Um tipo que tenha um subtipo é chamado de tipo mestre do subtipo. Pode-se ter um subtipo de um subtipo. Qualquer subtipo é tratado da mesma maneira que um tipo, o que caracteriza a existência de uma hierarquia de subtipos. O tipo supremo da hierarquia é chamado **tipo natural** de qualquer tipo da hierarquia. Além disso, um subtipo pode ter mais de um tipo mestre. Entretanto, deve-se notar que mesmo podendo ter mais de um tipo mestre, todo subtipo deve ter apenas um tipo natural.

Tal como no MRO, apenas tipos (de objetos e de relacionamentos) - e no MRO* também modalidades - possuem subtipos. Objetos e relacionamentos em si não são especializados.

4.5.1 - Subtipo de Objeto.

A mudança ocorrida nos subtipos de objetos, está no fato de terem sido explicitados dois sub-conjuntos do conjunto de tipos de objeto. Um sub-conjunto foi designado conjunto dos tipos naturais de objeto. Esse conjunto é formado pelos tipos de objetos definidos pelo usuário, os quais não necessitam de um predicado explícito para sua identificação. O outro sub-conjunto foi designado conjunto de subtipos de objetos. Esse conjunto é formado pelos tipos de objetos que são subtipos de outros tipos de objetos, ou subtipos de outros subtipos de objetos. Esses novos conjuntos serão melhor justificados no Capítulo 6.

4.5.2 - Subtipo de Relacionamento.

No MRO, os subtipos de relacionamento eram identificados como sendo simplesmente subtipos de tipos de relacionamento ou subtipos de subtipos de relacionamento. Com a identificação e denominação do elemento modalidade de primeiro grau, sobre o qual toda a semântica do tipo de relacionamento fica centrada, pôde-se também definir o subtipo de relacionamento sobre esse conceito, reconhecendo-se que a semântica do subtipo também está centrada sobre a modalidade de primeiro grau. Assim, identificou-se o elemento **submodalidade de relacionamento de primeiro grau**, a qual é um refinamento de uma modalidade de primeiro grau. Sobre a submodalidade de primeiro grau está centrada toda a semântica do subtipo de relacionamento. A justificação dessa mudança será apresentada no Capítulo 6.

Uma modalidade de segundo grau é composta por duas modalidades de primeiro grau. Sempre que uma modalidade de segundo grau for composta por ao menos uma

submodalidade de primeiro grau, esta será designada como uma submodalidade de relacionamento de segundo grau.

4.6 - Conclusão.

As mudanças feitas no MRO foram todas apresentadas nesse capítulo. Entretanto, melhores justificativas e interpretações para essas mudanças serão apresentadas no capítulo 6, porque necessitam do formalismo realizado para descrevê-las.

Capítulo 5

Atualização e Formalização do MRO → MRO*

5.1 - Introdução.

Neste capítulo é apresentado o formalismo realizado sobre o **MRO*** com as mudanças já efetuadas, as quais foram apresentadas no capítulo anterior.

No MRO, entes, eventos, coisas e conceitos do mundo real, são modelados através de objetos. Assim sendo, pode-se representar o mundo real pelo conjunto enumerável infinito U tal que, $U = \{ob_1, ob_2, \dots, ob_{n-1}, ob_n, ob_{n+1}, \dots\}$, onde **ob** é um objeto. Note-se que o próprio **MRO*** e qualquer base de dados existem no mundo real e são portanto objetos. Os conceitos do **MRO*** bem como sua formalização e representação são também objetos. Esse conceito é muito importante, e permite atribuir a designação de **Meta-Modelo** ao **MRO***. Toda formalização desenvolvida neste capítulo deve levar em conta que sempre que um conceito for definido como sendo alguma abstração matemática, isso não mascara o fato desse conceito, bem como sua representação ser também um objeto; portanto, podem ser definidos também pela mesma definição dada para um objeto qualquer.

5.2 - Objetos.

Definição 1: Seja **ob** um objeto. No **MRO*** um objeto é definido pela tupla $ob = \langle i, Idob, tob, Cat, Crel \rangle$, onde i é o código do objeto, $Idob = \{id_1, \dots, id_{n_{idob}}\}$ é o conjunto de identificadores do objeto definidos pelo usuário, **tob** é o tipo do objeto, **Cat** é o conjunto de atributos do objeto que são de interesse para um determinado empreendimento, e **Crel** é o conjunto de relacionamentos binários e triplos do objeto¹.

¹Os conceitos **Crel** e **Cat** serão definidos nas seções 5.3 e 5.6 respectivamente.

Definição 2: Uma base de dados é um conjunto finito de objetos $BD(\lambda)$, $BD(\lambda) \subset U$, representando as informações que são de interesse para um dado empreendimento λ .

Os objetos ob_k , $k \geq 1$, $ob_k \in U$, são tratados individualmente. Na representação de cada um na base de dados estão associadas todas as características do objeto no mundo real que for de interesse representar.

Definição 3: Um conjunto finito $Cob = \{ob_1, ob_2, \dots, ob_{nob}\}$, $Cob(\lambda) \subset BD(\lambda)$ é o conjunto de objetos existentes em uma $BD(\lambda)$, e nob é o número de objetos dessa base.

Definição 4: Seja $i \in Z, i \geq 1$ um identificador unívoco definido pelo sistema para um objeto ob . Então i é designado código do objeto. Representa-se o conjunto de códigos dos objetos como sendo $Cod = \{i \in Z^+, \exists ob_k \in BD(\lambda) | i \rightarrow ob_k\}$ ².

Deve-se observar que todos os elementos do MRO* são objetos e, portanto, possuem um código i .

Cada código de objeto é um designador interno, que identifica univocamente o objeto em toda a base de dados, não sendo possível modificá-lo em nenhuma circunstância. Todo objeto tem um código, o qual é único para qualquer objeto que exista ou tenha existido na base de dados.

No MRO* todos os objetos são classificados segundo diferentes "tipos", e a identificação dos objetos através de um identificador definido pelo usuário está sempre associada ao "tipo do objeto" que se quer recuperar.

Definição 5: Todo objeto tem um único "tipo" chamado tipo de objeto e denotado por tob .

Um tipo de objeto é definido pela tupla $tob = \langle i, Idtob, Cta, Ctrel \rangle$, onde i é o código do tipo de objeto, $Idtob$ é o conjunto de identificadores do tipo de objeto, Cta é o conjunto de tipos de atributo do tipo de objeto e $Ctrel$ é o conjunto de tipos de relacionamento do

² $i \rightarrow ob_k$, lê-se i determina ob_k

tipo de objeto³.

Na conceituação de bases de dados orientadas a objetos, todos os elementos da base são objetos. Obedecendo a essa conceituação, todos os elementos do MRO são objetos. Desse modo, tratando-se um tipo de objeto como objeto ter-se-ia que

$tob = \langle i, Idob, tob, = "tipo de objeto", \text{ Cta é } Cat \cup \{ \text{atributos de um tipo de objeto} \},$

$Ctrel \text{ é } Crel \cup \{ \text{relacionamento do tipo} \}$ ou seja: o identificador interno e os identificadores definidos pelo usuário são os mesmos, seja o objeto tratado como tal ou como tipo; o tipo de objeto é sempre a constante "tipo de objeto"; o conjunto de tipos de atributos é o conjunto de atributos definidos para o modelo, mais os atributos que um "tipo de objeto" puder ter associado, tal como o fato dele ser um tipo natural ou não; e o conjunto de tipos de relacionamento é o conjunto de relacionamentos do objeto mais os relacionamentos que um "tipo de objeto" puder ter, tal como o fato dele ser origem ou destino de um tipo de relacionamento.

Embora essa dualidade de representação possa ser feita em qualquer conceito, daqui para a frente ela somente será citada novamente quando se tornar necessária para esclarecer o conceito sendo tratado.

O conjunto $Cod(\lambda)$ contém apenas os objetos da base que representam as informações do empreendimento representadas por objetos identificáveis também externamente à base. O conjunto $BD(\lambda)$ contém além desses, os objetos que são tipo, e os objetos mantidos apenas internamente pela base.

Definição 6: O conjunto de todos os tipos de objetos representados em uma $BD(\lambda)$ é representado por $CTob(\lambda)$, onde $CTob = \{ tob_1, tob_2, \dots, tob_{nto} \}$ e nto é a quantidade de tipos de objetos definidos para essa $BD(\lambda)$.

Definição 7: Define-se uma função $Tipo: Cob(\lambda) \rightarrow CTob(\lambda)$, tal que, aplicada a um objeto, determina o tipo desse objeto, ou seja $Tipo(ob_j) = tob_j$, onde ob é o objeto e $1 \leq j \leq nto$.

³Os conceitos $Ctrel$ e Cta serão definidos nas seções 5.3 e 5.6, respectivamente.

Deve ser notado que $\text{Tipo}(ob_j)=tob_j$ indica que ob é uma projeção de tob_j , a qual pode ser denotada por $\pi(ob_j)=tob_j$.

As associações existentes entre os objetos são armazenadas na Base de Dados diretamente nos objetos, através dos códigos dos objetos, sem utilizar-se dos identificadores definidos pelo usuário. Desse modo os identificadores definidos pelos usuários podem ser desconsiderados ou alterados, sem que com isso se perca a caracterização ou a integridade do objeto e de todas as informações que existem sobre ele. Um identificador de um objeto é utilizado apenas para que o objeto seja localizado por uma aplicação⁴.

Definição 8: Um identificador de objeto é denotado por id . Define-se o conjunto $Id=\{id_1, id_2, \dots, id_{nid}\}$, como sendo o conjunto de todos os identificadores de todos os objetos de uma $BD(\lambda)$, onde nid é o número de identificadores em $BD(\lambda)$.

Os conjuntos $Id_{tob}, Id_{ob} \subset Id$.

5.3 - Relacionamentos entre Objetos.

Objetos interagem entre si no mundo real. Essa interação pode ser modelada no MRO', através de diversos recursos de modelagem que serão apresentados em seções a seguir, cada um deles com um conjunto de regras e propriedades intrínsecas. Os relacionamentos do mundo real que não se enquadram dentro de algum conceito do MRO' podem ser representados através de "relacionamentos" definidos pelo usuário.

Definição 9: As formas de interação que existem entre pares de objetos do mundo real são

⁴O termo aplicação é utilizado para designar um programa que utiliza a base de dados para realizar um trabalho específico do empreendimento.

modeladas no MRO' e denominadas **modalidades de relacionamento de primeiro grau**⁵, sendo denotadas por **mr**. Uma modalidade **mr** é definida pela tupla $mr = \langle mr_{op}, Idmr, Ctar \rangle$, onde mr_{op} é a modalidade de primeiro grau oposta, $Idmr$ é o conjunto de identificadores definidos para a modalidade **mr**, e $Ctar$ é o conjunto de tipos de atributos que podem ser associados a **mr**.

Definição 10: Representa-se o Conjunto de Modalidades de primeiro grau $CMr(\lambda)$ por $CMr = \{mr_1, mr_2, \dots, mr_{nmr}\}$, onde nmr é a quantidade de modalidades de primeiro grau definidas para essa $BD(\lambda)$.

As modalidades de primeiro grau caracterizam a maneira como pares de objetos quaisquer se associam. Quando dois tipos de objetos em particular se associam através de uma modalidade de primeiro grau, define-se um **tipo de relacionamento** entre objetos daqueles tipos.

Definição 11: Um tipo de relacionamento é representado por uma tupla ordenada de grau 3, denotada por $tr = \langle tob_o, tob_d, mr_i \rangle$ onde:

- tob_o é o tipo do objeto origem, $tob_o \in CTob(\lambda)$;
- tob_d é o tipo do objeto destino, $tob_d \in CTob(\lambda)$;
- mr_i é a modalidade de primeiro grau, $mr_i \in CMr(\lambda)$ com $1 \leq i \leq nmr$;

Definição 12: Define-se $CTr(\lambda)$ como sendo o conjunto de tipos de relacionamento pertencentes à $BD(\lambda)$, onde $CTr = \{tr_1, tr_2, \dots, tr_{ntr}\}$, e ntr é o número de tipos de relacionamento definidos para a $BD(\lambda)$.

Cada instância de um tipo de relacionamento constitui-se em um **relacionamento**

⁵Para simplicidade, a modalidade de relacionamento de primeiro grau será referenciada no texto apenas como modalidade de primeiro grau.

binário (ou simplesmente relacionamento) de tipo específico entre um par de objetos específicos.

Definição 13: Um relacionamento binário entre objetos de uma $BD(\lambda)$ é representado por uma tupla ordenada de grau 3, denotada por $r = \langle ob_o, ob_d, mr_i, Catr \rangle$ ⁶, que é uma instanciiação de um tipo de relacionamento $tr = \langle tob_o, tob_d, mr_i \rangle$, com $tr \in CTr(\lambda)$, e

$mr_i = \langle mr_{op}, Idmr, Cta \rangle$ e $mr_{op} = \langle mr_1, Idmr_{op}, Cta_{op} \rangle$ onde:

- ob_o é o objeto origem do relacionamento,

$ob_o \in Cob(\lambda)$ e $Tipo(ob_o) = tob_o$;

- ob_d é o objeto destino do relacionamento,

$ob_d \in Cob(\lambda)$ e $Tipo(ob_d) = tob_d$;

- mr_i é a modalidade de primeiro grau, $mr_i \in CMr(\lambda)$ com

$1 \leq i \leq nmr$;

- $Catr$ é o conjunto de atributos do relacionamento, tal que $\forall at \in Catr \rightarrow$

$TipoAt(at) \in Cta$ ou $TipoAt(at) \in Cta_{op}$ ⁷.

Definição 14: Define-se $Cr(\lambda)$ como sendo o Conjunto de relacionamentos binários

$Cr = \{r_1, r_2, \dots, r_n\}$ pertencentes à $BD(\lambda)$.

⁶Os conjuntos $Catr$, Cta e Cta_{op} serão definidos na seção 5.6.

⁷A função $TipoAt$ será definida na seção 5.6.

Note-se portanto que uma "modalidade de primeiro grau" caracteriza uma maneira como objetos em geral podem se associar, ao passo que um "tipo de relacionamento" caracteriza a possibilidade de que dois objetos de determinados tipos se associem dessa maneira, ou seja, esses objetos possuam relacionamentos de determinada modalidade entre si. Um relacionamento caracteriza que dois objetos específicos se associam através de uma determinada modalidade de primeiro grau.

O MRO* apresenta uma "simetria" estabelecendo que cada modalidade de primeiro grau possui uma modalidade de primeiro grau oposta.

Definição 15: Para toda modalidade de primeiro grau existe uma única modalidade de primeiro grau oposta. Assim, define-se uma função Moposta com domínio e contradomínio em $CMr(\lambda)$, tal que aplicada a uma modalidade de primeiro grau, determina a sua modalidade de primeiro grau oposta, ou seja: $Moposta(mr_i) = mr_{opi}$, $mr_i, mr_{opi} \in CMr(\lambda)$.

Dessa maneira é estabelecida também uma simetria nos relacionamentos e nos tipos de relacionamento.

Para todo tipo de relacionamento binário $tr = \langle tob_o, tob_d, mr_i \rangle$, existe um único tipo de relacionamento binário oposto, que é representado por uma tupla ordenada de grau 3, denotada por $tr_{op} = \langle tob_d, tob_o, mr_{opi} \rangle$, onde:

- tob_d é o tipo do objeto origem do relacionamento, $tob_d \in CTob(\lambda)$.
- tob_o é o tipo do objeto destino do relacionamento, $tob_o \in CTob(\lambda)$.
- mr_{opi} é a modalidade de primeiro grau oposta, $mr_{opi} \in CMr(\lambda)$ e $mr_{opi} = Moposta(mr_i)$.

Para todo relacionamento binário $r = \langle ob_o, ob_d, mr_i, Catr \rangle$, existe um único relacionamento binário oposto, que é representado por uma tupla ordenada de grau 3, denotada por $r_{op} = \langle ob_d, ob_o, mr_{opi}, Catr \rangle$, onde:

- ob_d é o objeto origem do relacionamento, $ob_d \in Cob(\lambda)$ e $Tipo(ob_d) = tob_d$.
- ob_o é o objeto destino do relacionamento, $ob_o \in Cob(\lambda)$ e $Tipo(ob_o) = tob_o$.
- mr_{opi} é a modalidade de primeiro grau oposta, $mr_{opi} \in CMr(\lambda)$ e $mr_{opi} = Moposta(mr_i)$.
- $Catr$ é conjunto de atributos do relacionamento oposto, o qual é o mesmo do relacionamento.

Sempre que houver um relacionamento entre dois objetos, no MRO* haverá também um relacionamento oposto entre esses dois objetos, que serão instâncias respectivamente de um tipo de relacionamento e do tipo de relacionamento oposto ao primeiro.

Definição 16: Define-se uma função $TipoRel: Cr(\lambda) \rightarrow CTr(\lambda)$, tal que aplicada a um relacionamento, determina o tipo desse relacionamento, ou seja $TipoRel(r) = tr$.

Relacionamentos são objetos tratados internamente pelo sistema; assim, seu conjunto de identificadores $IdobRel$ é vazio. O mesmo ocorre com os tipos de relacionamentos. O código de ambos existe, mas é usado apenas internamente, e nunca é passado para o usuário. Modalidades possuem identificadores, pelos quais são referenciadas externamente à base.

5.4 - Relacionamentos Envolvendo Múltiplos Objetos.

Os tipos de relacionamentos até agora discutidos envolvem sempre pares de tipos de objetos. Suas instâncias, os relacionamentos, envolvem pares de objetos. Cada um deles contém uma modalidade de primeiro grau e uma modalidade de primeiro grau oposta. Tal como já citado, esses relacionamentos são chamados relacionamentos binários. No entanto, é possível estabelecer relacionamentos que envolvam mais do que dois objetos. Relacionamentos envolvendo três objetos são chamados relacionamentos ternários, envolvendo quatro são chamados relacionamentos quaternários, e assim por diante. No MRO* é definida uma forma particular de relacionamento ternário chamado

relacionamento triplo. O relacionamento triplo recebe este nome por definir três lugares denominados **divisões**, onde os objetos são distribuídos. Um relacionamento triplo é uma instanciização de um **Tipo de Relacionamento Triplo** e é denotado por rt . Um tipo de relacionamento triplo constitui-se na interligação de tipos de relacionamentos binários, através de uma modalidade de relacionamento de segundo grau⁸, a qual é formada pela composição de duas modalidades de primeiro grau. Assim, um tipo de relacionamento triplo pode envolver três ou mais tipos de objetos, não necessariamente distintos entre si.

Definição 17: Define-se um conjunto de modalidades de relacionamento de segundo grau

$$CMrt = \{mrt_1, mrt_2, \dots, mrt_n\}, \quad Cmr(\lambda) \subset BD(\lambda) \quad \text{onde} \quad mrt_i = \langle mr_j, mr_k \rangle, \quad \text{com}$$

mr_j e $mr_k \in CMr(\lambda)$, sendo mr_j denominada **modalidade de conjunto** e mr_k denominada

modalidade de determinação. Uma modalidade de segundo grau é formada por duas modalidades de primeiro grau, o que caracteriza o tipo de relacionamento triplo. Não pode haver outra mrt_i com o mesmo par de modalidades de primeiro grau mr_j e mr_k como modalidade de conjunto e modalidade de determinação, respectivamente.

Uma modalidade de segundo grau pode ser indicada de 4 maneiras diferentes, conforme as possíveis composições com as modalidades de primeiro grau envolvidas. Apenas uma composição deve pertencer ao $CMrt(\lambda)$, ou seja, uma deve ser fixada quando for utilizada. Assim define-se um conjunto utilizando-se todas as composições possíveis com as modalidades de primeiro grau envolvidas, na qual apenas uma delas deve pertencer ao $CMrt(\lambda)$.

Definição 18: Define-se um conjunto de modalidades de segundo grau normal $CMrtN(\lambda)$ cujos elementos são as possíveis composições das modalidades de segundo grau, ou seja,

para cada $mrt_i = \langle mr_1, mr_2 \rangle \in CMrt(\lambda)$ então $mrt_i = \langle mr_1, mr_2 \rangle$, $mrt_i = \langle mr_1, mr_{2op} \rangle$,

$$mrt_i = \langle mr_{1op}, mr_2 \rangle \text{ e } mrt_i = \langle mr_2, mr_1 \rangle \in CMrtN(\lambda).$$

Definição 19: Um tipo de relacionamento triplo é representado por uma tupla ordenada

⁸Por simplicidade, modalidade de relacionamento de segundo grau, será referida neste texto como modalidade de segundo grau.

de grau 4, denotada por $trt = \langle tob_{d1}, \{tob_{d2i}\}, tob_{d3}, mrt \rangle$, com $1 \leq i \leq tob$ onde:

$$tob_{d1}, tob_{d2i}, tob_{d3} \in CTob(\lambda), mrt \in CMrt(\lambda) .$$

Definição 20: Define-se $CTrt(\lambda)$, onde $CTrt = \{trt_1, trt_2, \dots, trt_{nrt}\}$ como sendo o Conjunto de Tipos de relacionamentos triplos pertencentes à $BD(\lambda)$, onde o nrt é o número de tipos de relacionamentos triplos da $BD(\lambda)$.

Definição 21: Um relacionamento triplo entre objetos pode ser representado pela tupla de grau 4 $rt = \langle ob_{d1}, \{ob_{d2i}, ob_{d22}, \dots, ob_{d2n}\}, ob_{d3}, mrt \rangle$, o qual é uma instanciação de um tipo de relacionamento triplo $trt = \langle tob_{d1}, \{tob_{d2i}\}, tob_{d3}, mrt \rangle$, $trt \in CTrt(\lambda)$, sendo que $Tipo(ob_{d1}) = tob_{d1}$, $Tipo(ob_{d2i}) = tob_{d2i}$ e $Tipo(ob_{d3}) = tob_{d3}$.

Definição 22: Define-se $Crt(\lambda)$, onde $Crt = \{rt_1, rt_2, \dots, rt_{nrt}\}$, como sendo o conjunto de relacionamentos triplos pertencentes à $BD(\lambda)$, onde nrt é o número de relacionamentos triplos da $BD(\lambda)$.

Os objetos envolvidos em um **relacionamento triplo** ocupam lugares predeterminados no relacionamento chamados **divisão**. São três as divisões, denominadas D_1 , D_2 , e D_3 , e são distribuídas de tal maneira que existe um objeto na primeira divisão D_1 , um na terceira D_3 , e um ou mais objetos na segunda divisão D_2 .

Considerando as três divisões existentes no relacionamento triplo, temos que para

$$rt = \langle ob_{d1}, \{ob_{d2i}, \dots, ob_{d2n}\}, ob_{d3}, mrt \rangle \text{ vale:}$$

- $\exists! ob_{d1} \in r_i | ob_{d1} \text{ ocupa } D_1$.
- $\exists! ob_{d3} \in r_i | ob_{d3} \text{ ocupa } D_3$.

- $\exists ob_{2i} \in r, 1 \leq i \leq n | ob_{2i} \text{ ocupa } D_2$.

Todo relacionamento triplo pode ser projetado em um conjunto de relacionamentos binários. Assim, para todo relacionamento $rt = \langle ob_{d1}, \{ob_{d2}\}, ob_{d3}, mrt \rangle$ com $mrt = \langle mr_1, mr_2 \rangle$, existe um relacionamento $r_d = \langle ob_{d1}, ob_{d3}, mr_2, Cat \rangle$ chamado relacionamento de determinação de rt , e um conjunto $\{r_{d'}\} = \{\langle ob_{d1}, ob_{d2}, ob_{d3}, mr_1, Cat \rangle\}$ chamados relacionamentos de conjunto de rt .

5.5 - Identificação de Objetos por Relacionamentos.

Existem situações onde um objeto pode não ter um nome significativo. Existem também alguns casos onde um objeto pode ser identificado por um relacionamento.

Definição 23: Considere um relacionamento binário $r = \langle ob_1, ob_2, mr_1, Catr \rangle$ com

$ob_1 \in Cob(\lambda), ob_2 \in Cob(\lambda), mr \in CMr(\lambda)$. Se $\exists ob, ob \in Cob(\lambda), ob \neq ob_1, ob \neq ob_2 | r \rightarrow ob$,⁹ afirma-se que ob é um objeto identificado pelo relacionamento r .

Em relação ao relacionamento, os objetos ob_1 e ob_2 são denominados **objetos ascendentes**, enquanto ob é denominado **objeto descendente**. O fato de um objeto ser identificado por um relacionamento corresponde ao fato do objeto descendente **Herdar** dados dos objetos ascendentes.

⁹ $r \rightarrow ob$, lê-se r determina ob .

Definição 24: Pode-se definir $\text{IdentificadoRel}: \text{Cob}(\lambda) \rightarrow \text{Cr}(\lambda) \cup \{\text{Nulo}\}$ que aplicada a um objeto indica qual relacionamento o identifica, se este existir, e uma função $\text{IdentificaRel}: \text{Cr}(\lambda) \rightarrow \text{Cob}(\lambda) \cup \{\text{Nulo}\}$ que aplicada a um relacionamento binário indica o objeto que é identificado pelo relacionamento. Em ambas as funções um resultado Nulo indica a não existência de um relacionamento identificando um objeto.

5.6 - Atributos.

Um atributo é uma característica de um objeto. Na base, um atributo é um elemento que possui um tipo e um conjunto de valores associados a um objeto, ou a um relacionamento, com o fim de atribuir-lhe certas características e propriedades.

É necessário que se identifiquem características comuns em grupos de tipos de atributos, para possibilitar que um SGBDE ofereça maiores recursos de tratamento dos tipos de atributos de significado mais usados.

Definição 25: Todo atributo tem um "tipo" chamado "tipo de atributo" e é denotado por ta . Um tipo de atributo ta é definido como uma tupla de grau 3,

$ta = \langle \text{Idtar}, \text{característica}, \text{Estado} \rangle$, onde:

- Idtar é o conjunto de identificadores do atributo.
- característica é a característica de ta .
- estado indica o comportamento do tipo de atributo frente às suas instâncias na base.

Definição 26: Representa-se o conjunto de tipos de atributo de $\text{BD}(\lambda)$ por $\text{Cta}(\lambda)$, com

$\text{Cta} = \{ta_1, ta_2, \dots, ta_{nta}\}$, onde nta é o número de tipos de atributos.

Existem vários tipos de atributos cuja maneira de operação pode ser reconhecida e tratada por um SGBDE. Cada maneira de operação que possa ser tratada será uma **Característica de Tipo de Atributo**. Dessa forma, a cada tipo de atributo estará associada uma característica.

No MRO* é definido um conjunto de características de tipos de atributo mais comuns

a um SGBDE. As características estão divididas em quatro grupos: **Internas** (Propriedades, Comentários e Sinônimos), que se aplicam diretamente sobre os demais componentes; **Dinâmicas** (Regras e Procedimentos) que definem o comportamento dos objetos e relacionamentos; **Estruturais** (Tempo e Estruturas) que organizam outros atributos; e de **Interface** (Gráfico, Imagem, Som, Visualização) que permitem interfacear o gerenciador com outros sistemas.

Definição 27: Define-se o conjunto de características de atributo como $CCaracterística = \{\text{propriedades, sinônimos, comentários, regras, tempo, estrutura, procedimento, gráfico, imagens, som, visualização}\}$

Os atributos são instâncias de tipos de atributos, podendo ser aplicados a objetos ou aplicados a relacionamentos, sendo assim caracterizados como **Atributos de Objetos** ou **Atributos de Relacionamentos**. São denotados por at_o os atributos de objetos e, denotados por at_r os atributos de relacionamentos.

Definição 28: Define-se atributo de objeto por at_o , como sendo a tupla ordenada de grau 3 $at_o = \langle ob, ta, Valor \rangle$, onde $ob \in Cob(\lambda)$ é o objeto ao qual at_o pertence, $ta \in Cta(\lambda)$ é o tipo do atributo e **Valor** é o valor do atributo. Para isso é necessário que sendo $Tipo(ob) = tob$, e $tob = \langle i, Idob, Cta, Crel \rangle$, então $ta \in Cta$.

Note-se que sendo $ob = \langle i, Idob, tob, Cat, Crel \rangle$, at_o é um elemento de **Cat**.

Definição 29: Define-se atributo de relacionamento por at_r , como sendo a tupla ordenada de grau 3 $at_r = \langle r, ta, Valor \rangle$, onde $r \in Cr(\lambda)$ é o relacionamento envolvido, $ta \in Cta(\lambda)$ é o tipo de atributo, e **Valor** é o valor do atributo. Para isso é necessário que sendo $TipoRel(r) = tr$, onde $tr = \langle tob_o, tob_r, mr \rangle$, $mr = \langle mr_{op}, Idmr, Ctar \rangle$,

$mr_{op} = \langle mr, Idmr_{op}, Ctar_{op} \rangle$, então $ta \in (Ctar \cup Ctar_{op})$.

Definição 30: Define-se a função $TipoAt: Cat(\lambda) \rightarrow Cta(\lambda)$ tal que $TipoAt(at) = ta$, sendo at

um atributo de objeto ou de relacionamento.

Os atributos de uma base de dados $BD(\lambda)$ apoiada no MRO^* são divididos em **Atributos Intrínsecos** e **Atributos Extrínsecos**. Os atributos intrínsecos são internos ao *SGBDE*, ou seja, totalmente transparentes aos aplicativos, enquanto os atributos extrínsecos são aqueles que são agregados a objetos e relacionamentos pelos aplicativos. Assim deve-se observar que a caracterização de tipos de atributos somente aplica-se aos atributos extrínsecos.

5.7 - Escopo de Abrangência de Identificadores de Objetos.

No MRO^* apenas um campo (e não uma concatenação deles) deve ser suficiente para identificar cada objeto. Desse modo, qualquer identificador de um objeto é um atributo cujo valor deverá ser único em toda a base de dados. Para evitar-se isso, restringe-se o escopo de abrangência de um identificador usado para localizar um objeto. Assim, aumenta-se ou diminui-se a especificidade de um identificador, como se a visão que se tem da informação pudesse ser mais ou menos detalhada, a critério do próprio usuário. Quando visto de maneira global, um sistema apresenta um pequeno grau de detalhe. Se o interesse recai em um dos seus componentes, o grau de detalhe daquele componente que está sendo observado aumenta. Pode então existir uma ferramenta de comunicação com o usuário, que interprete qualquer identificador levando em consideração somente o conjunto de informações que está sendo examinado, e exigindo que qualquer referência a outras informações tenha um teor de identificação maior.

Quando se manipula a base de dados de um ponto de vista global, são poucos os objetos acessíveis. Indicando-se ao sistema que se pretende considerar a seguir um desses objetos, indica-se uma região da Base de Dados em que se pretende obter ou manipular informações. Dessa forma tornam-se acessíveis novos objetos. Esse procedimento pode ser repetido, indicando-se para consideração um dos novos objetos tornados disponíveis, aumentando-se sucessivamente o nível de detalhe, e correspondentemente a quantidade de objetos a que se tem acesso diretamente através de seus identificadores.

O MRO^* implementa esses aspectos devido aos conceitos de **Colônia de Objetos**, **Objeto Considerado**, **Objetos Habitantes de uma Colônia** e **Contexto**.

Quando se procura acessar um objeto armazenado numa base de dados apoiada no MRO^* , se o seu código for conhecido o acesso é imediato, uma vez que o código identifica

univocamente um objeto dentre todos os objetos da base de dados. No entanto, quando a busca é feita por um identificador, essa identificação não é unívoca, uma vez que pode (a princípio) existir mais de um objeto do mesmo tipo com um mesmo nome. Para que se possa identificar um objeto pelo seu nome univocamente é necessário que se especifique o Contexto¹⁰ onde esse nome está definido. A caracterização da existência de contextos é feita através da partição dos identificadores existentes na base de dados em Colônias, de maneira que um nome esteja definido somente no domínio de uma Colônia.

Todo objeto da base é ou globalmente reconhecido por todos os usuários da base, ou detalha um outro objeto. Quando um objeto ob_1 detalha um ob_2 , diz-se que ob_1 constrange ob_2 , ou igualmente que ob_2 é constrito por ob_1 . Assim como os objetos, todo tipo de objeto da base é globalmente reconhecido, ou detalha um outro tipo de objeto.

O conceito de colônia agrupa os conceitos de escopo de abrangência de identificadores de objetos e de abstração de composição, uma vez que restringe a capacidade de identificação de um identificador ao conjunto de objetos que são parte de uma colônia constrita pelo objeto composto.

Definição 31: Considere uma relação $\mathcal{RSTA} = \{r = \langle tob_p, tob_i \rangle \in CTob(\lambda) \times CTob(\lambda)\}$

denominada Relação Sintetiza Tipo Ampla, a qual possui as seguintes propriedades:

- $\exists r = \langle tob_p, tob_i \rangle \in \mathcal{RSTA}$;
- se $\exists r_i = \langle tob_p, tob_j \rangle \in \mathcal{RSTA} \rightarrow \exists r_j = \langle tob_p, tob_i \rangle \in \mathcal{RSTA}$;
- $\exists r_i = \langle tob_p, tob_j \rangle \in \mathcal{RSTA} \wedge \exists r_j = \langle tob_p, tob_k \rangle \in \mathcal{RSTA} \rightarrow \exists r_k = \langle tob_p, tob_k \rangle \in \mathcal{RSTA}$
(regra da transitividade).

Definição 32: Considere uma relação $\mathcal{RST} \subset \mathcal{RSTA}$ tal que $\mathcal{RST}$ possui todas as tuplas de $\mathcal{RSTA}$, menos as que não obedeçam as seguintes propriedades adicionais:

¹⁰O conceito de Contexto será formalmente definido após a definição de alguns conceitos necessários para seu entendimento.

•se $\exists r_i = \langle \text{tob}_i, \text{tob}_j \rangle \in \mathcal{RSTA} \rightarrow \exists r_k = \langle \text{tob}_k, \text{tob}_j \rangle \in \mathcal{RSTA} \mid \text{tob}_k \neq \text{tob}_i$;

•se $\exists r_k = \langle \text{tob}_i, \text{tob}_j \rangle \in \mathcal{RSTA}$ e $\exists r_i = \langle \text{tob}_p, \text{tob}_k \rangle \in \mathcal{RSTA}$, então $r_x = \langle \text{tob}_p, \text{tob}_j \rangle \in \mathcal{RST}$.

Definição 33: Seja $r = \langle \text{tob}_p, \text{tob}_k \rangle \in \mathcal{RST}$. Então diz-se que tob_k é sintetizado por tob_j ou que tob_j sintetiza tob_k . Se $\nexists \langle \text{tob}_p, \text{tob}_k \rangle \in \mathcal{RST}$, diz-se que tob_k é global.

Um tipo tob é ou globalmente reconhecido, ou é sintetizado por outro tipo. Define-se uma função chamada **Sintetiza-Tipo**, que indica qual tipo de objeto sintetiza outro tipo de objeto dado, ou se esse tipo de objeto é globalmente reconhecido.

Definição 34: Define-se uma função **Sintetiza-Tipo**: $\mathcal{CTob}(\lambda) \rightarrow \mathcal{CTob}(\lambda) \cup \{Nulo\}$, tal que

$$\text{Sintetiza-Tipo}(\text{tob}_k) = \begin{cases} Nulo, & \text{se } \text{tob}_k \text{ é Global} \\ \text{tob}_j, & \text{se } \text{tob}_k \text{ é sintetizado por } \text{tob}_j \end{cases} .$$

O conjunto $\{\text{tob}_k \in \mathcal{CTob}(\lambda) \mid \text{Sintetiza-Tipo}(\text{tob}_k) = \text{tob}_j\}$ é denotado por $\mathcal{CtoSo}_j(\lambda)$, e chamado **Conjunto de Tipos de Objetos que refinam tob_j** . O conjunto de tipos $\{\text{tob}_i \in \mathcal{CTob}(\lambda) \mid \text{Sintetiza-Tipo}(\text{tob}_i) = Nulo\}$ é denotado $\mathcal{Ctogl}(\lambda)$, e chamado **Conjunto de Tipos de objetos que estão globalmente visíveis por todos os usuários da base ou Tipos de Objetos Globais**. O conjunto $\mathcal{CtoS} = \bigcup_{i=1}^{ntob} \mathcal{CtoSo}_i$, é o conjunto de tipos de objetos que refinam tipos de objetos da base, onde $ntob$ é o número de tipos de objetos que restringem outros objetos em $\mathcal{BD}(\lambda)$.

Note-se que os conjuntos $\mathcal{CtoSo}_i(\lambda)$ junto com o conjunto $\mathcal{Ctogl}(\lambda)$ constituem uma partição sobre $\mathcal{CTob}(\lambda)$ através da função **Sintetiza-Tipo**.

Definição 35: O conjunto de aspectos de uma base de dados $\mathcal{BD}(\lambda)$ é o conjunto

$Asp = \{asp_1, asp_2, \dots, asp_{nasp}\}$. A função $Aspecto: CtoSo_i \rightarrow Asp$ é definida como o aspecto que cada tipo de objeto que refina o tipo de objeto da base refina.

Um tipo de objeto tob_j é detalhado sob vários aspectos. Cada aspecto detalha um tipo tob_j através de um subconjunto do conjunto de tipos de objetos que refinam objetos de tipo tob_j . Cada subconjunto constitui os tipos de objetos que habitam um tipo de colônia.

Definição 36: Denomina-se **C-thabita** a um conjunto não nulo de tipos de objetos que habitam um tipo de colônia, o qual é ou o conjunto $Ctogl(\lambda)$ ou é uma partição de um elemento $CtoSo_i(\lambda) \in Ctos(\lambda)$ obtida através da função inversa

$$Aspecto^{-1}: Asp \rightarrow CtoSo_i \mid Aspecto^{-1}(asp_i) = C-thabita_i.$$

Note que **C-thabita** pode ser o próprio $CtoSo_i$, se $\forall tob_x \in CtoSo_i, Aspecto(tob_x) = asp_x$.

Uma partição de $CTob(\lambda)$ conforme descrita, define uma coleção de conjuntos, os quais são denominados **tipo de colônia** e denotados genericamente por tco_j . A representação destes elementos será feita através de tuplas que serão descritas a seguir.

Note que fixada a partição de $CTob(\lambda)$, esta divisão é característica da base de dados. Se a partição for alterada, mesmo se respeitando suas características, estará sendo determinada outra base de dados.

Definição 37: Um tipo de colônia tco_j será representado por uma tupla $tco_j = \langle Idtcol_j, C-thabita_j, tobjConstr_j \rangle$, onde:

- $Idtcol_j$ indica um conjunto de nomes identificadores do tipo de colônia;
- $tobjConstr_j$ é Nulo se o tipo de colônia é o Tipo Global, ou tob_j quando a tupla representa o tipo de colônia cujos tipos de objetos são constringidos por tob_j .
- $C-thabita_j$ é o conjunto de Tipos Globais $Ctogl(\lambda)$, se o tipo de colônia é o Tipo Global com $tobjConstr_j = \text{Nulo}$, ou $C-thabita_j \subset CtoSo_x(\lambda)$, com $tobjConstr_j = tob_x$, sendo $C-thabita_j$ chamado conjunto de tipos de objetos que habitam a colônia de tipo tco_j . Diz-se que $tobjConstr_j$ constringe o tipo de colônia tco_j .

Definição 38: Define-se como $CTCo = \{tco_0, tco_1, \dots, tco_{ntco}\}$, $CTCo(\lambda) \in BD(\lambda)$ o conjunto de tipos de colônias de $BD(\lambda)$, onde $ntco$ é o número de tipos de colônias de $BD(\lambda)$.

Definição 39: O tipo de colônia $tco_0 = \langle \text{Tipo Global}, C\text{-thabita}_0, \text{Nulo} \rangle$ é chamado **Tipo de Colônia Global** ou apenas **Tipo Global**.

Note-se que existe apenas um tipo de colônia de tipo global.

Os tipos de objetos que habitam um tipo de colônia são ditos habitantes desse tipo de colônia, sendo que cada tipo de objeto habita exatamente um tipo de colônia.

Teorema 1: Cada tipo de objeto é definido em apenas um tipo de colônia, ou seja,

$$\forall tob_i \in CTob(\lambda), 1 \leq i \leq ntob, \exists ! tco = \langle Idtcol, C\text{-thabita}, tobjConstr \rangle \mid tob_i \in C\text{-thabita}.$$

Demonstração: Considere que tob_i habite $tco_k = \langle Idtcol_k, C\text{-thabita}_k, tobConstr_k \rangle$ e habite

$tco_p = \langle Idtcol_p, C\text{-thabita}_p, tobConstr_p \rangle$. Então $tob_i \in C\text{-thabita}_k$ e $tob_i \in C\text{-thabita}_p$. Se $tobConstr_k = tobConstr_p$, então há um absurdo, pois $C\text{-thabita}_k$ e $C\text{-thabita}_p$ são conjuntos da mesma partição (pela Definição 36). Se $tobConstr_k \neq tobConstr_p$, então também há um

absurdo pela construção de $\mathfrak{AST}$, já que se tob_i habita

$$\langle Idtcol_p, C\text{-thabita}_p, tobConstr_p \rangle \in \mathfrak{AST} \rightarrow \langle tobConstr_p, tob_i \rangle \in \mathfrak{AST}, \text{ e } tob_i \text{ habita}$$

$$\langle Idtcol_k, C\text{-thabita}_k, tobConstr_k \rangle \rightarrow \langle tobConstr_k, tob_i \rangle \in \mathfrak{AST}, \text{ o que é um absurdo pela}$$

definição de $\mathfrak{AST}$.

Definição 40: Define-se uma função $THabita: CTob(\lambda) \rightarrow CTCo(\lambda)$ tal que

$$THabita(tob) = tco, \quad tco = \langle Idtcol, C\text{-thabita}, tobjConstr \rangle, \quad tob \in C\text{-thabita}.$$

Teorema 2: O conjunto $CTCo(\lambda)$ estabelece uma partição sobre $CTob(\lambda)$, através da função **THabita**.

Demonstração: Através da função **THabita** são obtidos os tipos de colônia

$$tco_i, 0 \leq i \leq ntco \mid tco_i = \langle nome, C-thabita_i, tobjConstr \rangle \text{ com } C-thabita_i = \langle tob_{i1}, \dots, tob_{intob} \rangle,$$

$$tob_{ik} \in CTob(\lambda), 1 \leq k \leq ntob. \text{ Como } \forall tob \in CTob(\lambda), \exists ! tco = \langle nome, C-thabita, tobjConstr \rangle$$

tal que $tob \in C-thabita$, logo se $tco_i \cap tco_j \neq \emptyset$, então $i=j$. O conjunto de objetos que

habitam uma colônia $C-thabita$ é não nulo; logo aplicando-se a função **THabita** para todo tob_k

obtem-se $\bigcup_{i=0}^{ntco} tco_i = CTCo(\lambda)$. Portanto, os conjuntos obtidos tco_i a partir da função **THabita**

estabelecem uma partição no conjunto $CTCo(\lambda)$ de $BD(\lambda)$.

Definição 41: Todo tipo de colônia a menos do tipo de colônia global tco_0 , possui um único tob_i que a constringe. Assim, define-se a função

$$TConstrita: CTCo(\lambda) \rightarrow CTob(\lambda) \cup \{Nulo\} \mid TConstrita(tco_i) = \begin{cases} tob_i, & \text{se } i \neq 0 \\ Nulo, & \text{se } i = 0 \end{cases}. \text{ Note que o tipo de}$$

colônia global não tem um tipo de objeto que a constringe e, portanto, convencionou-se um elemento **Nulo**, tal que $TConstrita(tco_0) = Nulo$.

Teorema 3: As funções $THabita: CTob(\lambda) \rightarrow CTCo(\lambda)$ e $TConstrita: CTCo(\lambda) \rightarrow CTob(\lambda) \cup \{Nulo\}$ podem ser compostas em uma função $TConstrita \circ THabita: CTob(\lambda) \rightarrow CTob(\lambda) \cup \{Nulo\}$ que é igual à função **Sintetiza-Tipo: CTob(\lambda) → CTob(\lambda) ∪ {Nulo}**.

Considerando-se que um objeto ob_1 é globalmente reconhecido, ou sintetiza um outro objeto ob_2 , define-se uma função chamada **Sintetiza**, que indica qual objeto sintetiza outro objeto dado, ou se esse objeto é globalmente reconhecido.

Definição 42: Define-se uma função $\text{Sintetiza}(\lambda) : \text{Cob}(\lambda) \rightarrow \text{Cob}(\lambda) \cup \{\text{Nulo}\}$, tal que

$$\text{Sintetiza}(ob_k) = \begin{cases} \text{Nulo}, & \text{se (1)} \\ ob_j, & \text{se (2)} \end{cases}, \text{ onde:}$$

$$(1) \text{ Tipo}(ob_k) = tob_k \mid tob_k \in \text{Ctogl}(\lambda);$$

$$(2) \text{ Tipo}(ob_k) = tob_k \mid tob_k \in \text{CotSo}_j(\lambda), \text{ Tipo}(ob_j) = tob_j \text{ com } \text{Sintetiza-Tipo}(tob_k) = tob_j ; ob_j$$

é o objeto que sintetiza ob_k num determinado empreendimento λ .

O conjunto $\{ob_i \in \text{Cob}(\lambda) \mid \text{Sintetiza}(ob_i) = ob_k, ob_k \neq ob_i\}$, é denotado $\text{CobSo}_k(\lambda)$, e chamado **Conjunto de Objetos que sintetizam o objeto ob_k** . O conjunto $\{ob_i \in \text{Cob}(\lambda) \mid \text{Sintetiza}(ob_i) = \text{Nulo}\}$ é denotado $\text{Cobgl}(\lambda)$, e chamado **conjunto de objetos que estão globalmente visíveis por todos os usuários da base, ou conjunto de objetos globais**. O conjunto $\text{CobS} = \bigcup_{i=1}^{nob} \text{CobSo}_i$ é o conjunto de objetos não globais da base, onde nob é o número de objetos que constroem outros objetos.

Os conjuntos $\text{CobSo}_i(\lambda)$ constituem uma partição de $\text{CobS}(\lambda)$ através da função **Sintetiza**, uma vez que em cada $\text{CobSo}_i(\lambda)$ estão os objetos de $\text{CobS}(\lambda)$ que têm o mesmo valor obtido através da função **Sintetiza**.

Definição 43: A função $\text{HabitaTcol} : \text{CobS}(\lambda) \rightarrow \text{CTCo}(\lambda)$ é definida como

$$\text{HabitaTcol}(ob_k) = tco_i \mid \text{Tipo}(ob_k) = tob_k \text{ onde } tob_k \in C\text{-thabita}_i, \text{ sendo que } C\text{-thabita}_i \text{ é o}$$

conjunto de tipos de objetos da colônia $tco_i = \langle \text{Idtcol}_i, C\text{-thabita}_i, tobConstr_i \rangle$.

Definição 44: Denomina-se **C-habita** ao conjunto de objetos que habitam uma colônia, o qual ou é o $\text{Cobgl}(\lambda)$, ou um elemento da partição de cada conjunto $\text{CobSo}_i(\lambda)$ obtido pela função **HabitaTCol**.

Definição 45: Uma colônia $Co(\lambda)$ será representada por uma tupla

$Co_i = \langle C-habita_i, objConstr_i, tco \rangle$, onde:

- $objConstr_i$ é nulo se a colônia é a global, ou um objeto ob_i quando a tupla representa uma colônia cujos objetos são constrictos por ob_i ;
- $C-habita_i$ é o conjunto de objetos Global se $objConstr_i = \text{Nulo}$, com $C-habita_i = Cobgl$; ou se $objConstr_i = ob_x$ então $C-habita_i \subset CobS(\lambda)$;
- tco é o tipo de colônia de Co .

Definição 46: A colônia $Co_0 = \langle C-habita_0, \text{Nulo}, tco_0 \rangle$ é chamada Colônia Global. Existe apenas uma colônia global.

Definição 47: Define-se como $CCo = \{Co_0, Co_1, Co_2, \dots, Co_{nco}\}$, $CCo(\lambda) \in BD(\lambda)$, ao conjunto de todas as colônias de $BD(\lambda)$, onde nco é o número de colônias de $BD(\lambda)$.

Definição 48: Define-se uma função $Tipocol: CCo(\lambda) \rightarrow CTCo(\lambda) | Tipocol(Co_j) = tco_j$, tal que

$Co_j = \langle C-habita_j, objConstr_j, tco_j \rangle$.

Os objetos que habitam uma colônia são ditos **habitantes** dessa colônia, sendo que cada objeto **habita** exatamente uma colônia.

Teorema 4: Cada objeto é definido em apenas uma colônia, ou seja,

$\forall ob_i \in Cob(\lambda), 1 \leq i \leq nob$, $\exists ! Co = \langle C-habita, objConstr, tco \rangle$ tal que $ob_i \in C-habita$, sendo ob_i dito habitante da colônia $Co(\lambda)$.

Demonstração: Considere que ob_i habite a colônia $Co_i = \langle C-habita_i, objConstr_i, tco_i \rangle$ e a colônia $co_j = \langle C-habita_j, objConstr_j, tco_j \rangle$. Então $ob_i \in C-habita_i$ e $ob_i \in C-habita_j$. Se

$objConstr_i = objConstr_j$ então há um absurdo, pois $C-habita_i$ e $C-habita_j$ são conjuntos de uma mesma partição (pela Definição 44). Se $objConstr_j \neq objConstr_i$ significa que ob_i habita $\langle C-habita_p, objConstr_p, tco_j \rangle$ com

$Tipo(objConstr_p) = tob_p$ e $Tipo(ob_i) = tob_i \rightarrow \langle tob_p, tob_i \rangle \in \mathfrak{RST}$ e ob_i habita

$\langle C-habita_k, objConstr_k, tco_k \rangle$ com $Tipo(objConstr_k) = tob_k$ e $Tipo(ob_i) = tob_i \rightarrow$

$\langle tob_p, tob_i \rangle \in \mathfrak{RST}$, o que é um absurdo pela definição de $\mathfrak{RST}$.

Definição 49: Define-se uma função $Habita: Cob(\lambda) \rightarrow CCo(\lambda)$ tal que $Habita(ob) = Co$ com

$Co = \langle C-habita, objConstr, tco \rangle$, $ob \in C-habita$.

Teorema 5: O conjunto $CCo(\lambda)$ estabelece uma partição sobre $Cob(\lambda)$, através da função **Habita**.

Demonstração: Através da função **Habita** são obtidos os conjuntos

Co_p , $0 \leq i \leq nco$ | $Co_i = \langle C-habita_p, objConstr_p, tco_i \rangle$, com $C-habita_i = \{ob_{i1}, \dots, ob_{inob}\}$,

$ob_{ik} \in Cob(\lambda)$, $1 \leq k \leq nob$. Como pelo Teorema 4,

$\forall ob \in Cob(\lambda)$, $\exists ! Co = \langle C-habita, objConstr, tco \rangle$ tal que $ob \in C-habita$, logo se

$Co_i \cap Co_j \neq \emptyset$, então $i = j$. Aplicando a função **Habita** para todo ob_k , obtém-se

$\bigcup_{i=0}^{nco} Co_i = CCo$. Portanto, os conjuntos Co_i obtidos a partir da função **Habita** estabelecem

uma partição no conjunto $CCo(\lambda)$ de $BD(\lambda)$.

Definição 50: Toda colônia a menos da colônia global, Co_0 , possui exatamente um objeto ob_i

que a constringe. Assim, define-se uma função $\text{Constrita}: \text{CCo}(\lambda) \rightarrow \text{Cob}(\lambda) \cup \{\text{Nulo}\}$, tal que

$$\text{Constrita}(Co_p) = \begin{cases} ob_p, & \text{se (1)} \\ \text{nulo}, & \text{se (2)} \end{cases} \quad \text{onde:}$$

(1) $\text{Tipo}(ob_p) = tob_p$, $\text{TipoCol}(Co_p) = tco_i$ e $\text{Constrita-Tipo}(tco_i) = tob_p$

(2) $\text{TipoCol}(Co_p) = tco_i$ e $\text{Constrita-Tipo}(tco_i) = \text{Nulo}$

Note que a colônia global é a única que não tem um objeto que a constringe, e portanto convencionou-se que $\text{Constrita}(Co_0) = \text{Nulo}$.

Teorema 6: As funções $\text{Habita}: \text{Cob}(\lambda) \rightarrow \text{CCo}(\lambda)$ e $\text{Constrita}: \text{CCo}(\lambda) \rightarrow \text{Cob}(\lambda) \cup \{\text{Nulo}\}$ podem ser compostas em uma função $\text{Constrita} \circ \text{Habita}: \text{Cob}(\lambda) \rightarrow \text{Cob}(\lambda) \cup \{\text{Nulo}\}$ que é igual à função $\text{Sintetiza}: \text{Cob}(\lambda) \rightarrow \text{Cob}(\lambda) \cup \{\text{Nulo}\}$.

Definição 51: Define-se um dígrafo de constrição $\text{GC}(\text{V}, \text{CA})$ onde $\text{V} = \text{Cob}(\lambda) \cup \{\text{Nulo}\}$, sendo que $\forall ob_i \in \text{Cob}(\lambda), \exists tob_i \in \text{CTob}(\lambda)$ tal que $\text{Tipo}(ob_i) = tob_i$ e $\text{CTob}(\lambda)$ é o domínio da relação $\mathcal{RST}$, e $\text{CA} = \{(ob_i, ob_j) \text{ tal que } \text{Sintetiza}(ob_j) = ob_i\} \cup \{(\text{Nulo}, ob_i) \text{ tal que } \text{Sintetiza}(ob_i) = \text{Nulo}\}$.

Definição 52: Define-se um caminho de acesso a um objeto ob_x como sendo o caminho Cam_{ob_x} obtido a partir do dígrafo GC , do seguinte modo:

Seja ob_x tal que $\text{Habita}(ob_x) = Co_y$.

a.se $Co_y = Co_0$, então $\text{Cam}_{ob_x} = (ob_x)$; note que nesse caso o caminho de acesso é o próprio objeto, uma vez que $\forall ob_x \mid \text{Habita}(ob_x) = Co_0$, ob_x estará sempre visível, pois

$Co_0 = \langle C\text{-habita}_0, \text{Nulo}, tco \rangle$ com $C\text{-habita}_0 = \text{Cobgl}(\lambda)$.

b.se $Co_y \neq Co_0$, então $Cam_{ob_x} \subset CA$ e $Cam_{ob_x} = ((ob_p, ob_j), (ob_p, ob_k), \dots, (ob_z, ob_x))$ onde

$Sintetiza(ob_j) = Nulo$.

Definição 53: Contexto de um objeto ob_x é a sequência $\langle ob_p, \dots, ob_z \rangle$ composta pelos primeiros elementos de todos os pares ordenados pertencentes a Cam_{ob_x} .

Teorema 7: Seja $ob_j \in Cob(\lambda)$. Então $\exists ob_i | Habita(ob_i) = Co_0$, e um caminho ligando ob_i a ob_j .

Demonstração: Como caminho de acesso é definido a partir de $GC(V, CA)$, tem-se

$\forall ob_i \in Cob(\lambda)$, ou $\exists ob_j \in Cob(\lambda) | (ob_p, ob_j) \in CA$ ou $(Nulo, ob_i) \in CA$, implicando

$\exists Cam_{ob_x} = ((ob_p, ob_j), (ob_p, ob_k), \dots, (ob_z, ob_j))$, com $Sintetiza(ob_j) = Nulo$.

Teorema 8: O caminho de acesso a um objeto é sempre acíclico.

Demonstração: Seja $ob_j \in BD(\lambda)$. Suponha que Cam_{ob_j} possua ciclos, e seja

$Cam_{ob_j} = ((ob_p, ob_k), (ob_k, ob_l), (ob_l, ob_p), (ob_p, ob_m), (ob_m, ob_j), \dots, (ob_z, ob_j))$. Isso implicaria que

$Sintetiza(ob_p) = ob_k$ e $Sintetiza(ob_l) = ob_m$. Conseqüentemente, $ob_k = ob_m$. Desse modo,

pelo Cam_{ob_j} , seria válido $Sintetiza(ob_p) = ob_l$ e $Sintetiza(ob_l) = ob_p$, implicando que como

$Tipo(ob_p) = tob_p$ e $Tipo(ob_l) = tob_l$, e n t ã o $Sintetiza-Tipo(tob_p) = tob_l$,

$Sintetiza-Tipo(tob_l) = tob_p$, o que é um absurdo, pois $Sintetiza-Tipo$ é definida sobre

AST .

Para indicar-se um objeto ob_j que habita uma determinada colônia, pode-se utilizar a seguinte notação baseada no caminho de acesso a esse objeto: $ob_1/ob_2/../ob_k/ob_j$, onde ob_1 é um objeto que habita a colônia global, e $ob_1/ob_2/../ob_k$ que é o contexto do objeto ob_j .

Todos os conceitos do MRO até agora tratados são estáticos, ou seja, são definidos independentemente das operações que possam ser aplicadas sobre ele. No entanto, sobre o conceito de colônias baseia-se o controle de acesso a base, tornando-se interessante apresentar-se algumas operações básicas sobre elas. Estas operações mostram aspectos dinâmicos do conceito de colônias.

No MRO define-se que os objetos que *habitam* uma colônia estão visíveis ao usuário se esta colônia está *acessível*. Pode-se através de uma operação, "abrir" uma colônia para permitir que os objetos se tornem acessíveis para consulta e modificação ou apenas para consulta. Assim define-se estado de uma colônia como se segue.

Definição 54: Seja $Co_i \in CCo(\lambda)$, $0 \leq i \leq nco$. Define-se uma função EstadoCol, que aplicada a uma colônia Co_i , indica se a mesma está não acessível, acessível ou aberta, ou seja,

$EstadoCol: CCo(\lambda) \rightarrow \{na, ac, ab\}$,

$$EstadoCol(Co_i) = \begin{cases} na, & \text{se } Co_i \text{ está não acessível} \\ ac, & \text{se } Co_i \text{ está acessível} \\ ab, & \text{se } Co_i \text{ está aberta} \end{cases} .$$

São possíveis as seguintes operações sobre uma colônia:

- **requisitar:** requisita uma colônia tornando-a acessível, ou seja, os objetos que a habitam tornam-se visíveis;
- **abrir:** aplica-se a uma colônia que está acessível, tornando possíveis operações de consulta, ou (e) modificação sobre ela;
- **fechar:** aplica-se a uma colônia que está aberta, fechando-a, porém deixando-a ainda acessível;

- **liberar**: aplica-se a uma colônia que está acessível, tornando-a não acessível;
- **considerar**: aplica-se a uma colônia tornando-a acessível e aberta, ou seja, equivale às operações requisitar e abrir juntas.

Os argumentos de todas as funções é (ob, tco) , indicando-se a colônia de tipo tco constrita pelo objeto ob .

Definição 55: A colônia global está permanentemente aberta, ou seja, por definição, $EstadoCol(C0_0)=ab$ permanentemente.

Definição 56: Seja $ob_i \in Cob(\lambda)$, $Co_j \in CCo(\lambda) \mid Co_j = \langle C-habita_p, objConstr, tco_j \rangle$. Se ob_i habita $Co_j = \langle C-habita_p, objConstr, tco_j \rangle$, ou seja $ob_i \in C-habita_j$, então ob_i estará disponível se $EstadoCol(Co_j)=ab$ com $1 \leq i \leq nob$ e $1 \leq j \leq nco$.

Definição 57: Seja $ob_i \in Cob(\lambda)$. Se ob_i está presentemente disponível, então $Sintetiza(ob_j)=ob_j$, indica que a colônia $Co=Habita(ob_j)$, de tipo $TipoCol(Co)=tco$ foi submetida a uma operação $Considerar(ob_j, tco)$, e ob_j é denominado **Objeto Considerado**.

A operação "abrir" aplicada a uma colônia de tipo t , fechará a colônia desse mesmo tipo t , se houver alguma aberta. Note-se que assim haverá no máximo uma colônia de cada tipo aberta em um determinado instante, uma vez que no início das atividades de uma base, apenas a colônia global estará aberta. Assim, como sempre que uma colônia é aberta, se houver outra do mesmo tipo aberta ela é fechada, então nunca será possível existir mais do que uma aberta.

A operação "fechar" aplicada a uma colônia de tipo t , fechará a colônia desse tipo que estiver aberta (se houver) e todas as colônias abertas cujos tipos são constritos por objetos de tipos que habitam a colônia do tipo t . Isso fará com que todas as colônias abertas cujos tipos são constritos por objetos de tipos que habitam as colônias agora fechadas, serão também fechadas, e assim recursivamente até que todas as colônias que direta ou indiretamente são constritas por objetos que são sintetizados por tipos de objetos que habitam a colônia em questão, sejam também fechadas. Note-se que abrir uma colônia

é uma operação que sempre abre uma única colônia, mas essa operação pode implicar no fechamento de uma ou mais colônias.

Teorema 9: Cada tipo de objeto só pode ser encontrado em uma única colônia aberta.

Definição 58: Todo identificador de um objeto é único entre os objetos de uma mesma colônia, porém pode ocorrer que exista mais de um objeto com o mesmo identificador, quando estes objetos habitarem colônias distintas (do mesmo tipo ou não). Ou seja, dados

dois objetos $ob_i = \langle i, Idob_p, tob_p, Cat_p, Crel_i \rangle$ e $ob_j = \langle j, Idob_p, tob_p, Cat_p, Crel_j \rangle$, então se

$Habita(ob_i) = Habita(ob_j)$ e $id_k \in Idob_i$ e $id_k \in Idob_j \rightarrow ob_i = ob_j$. Se

$Habita(ob_i) \neq Habita(ob_j)$, nada pode ser dito a respeito, uma vez que mesmo que $id_k \in Idob_i$ e $id_k \in Idob_j$ não significa que $ob_i = ob_j$.

Quando deseja-se referir a algum objeto que habita uma colônia não aberta, ou a um objeto que habita uma colônia constrita por um objeto que não o presentemente considerado, então é necessário que se identifique o caminho de acesso desde o ponto de interesse atual na base de dados até o ponto onde está o objeto a que se quer se referir. Assim pode-se utilizar uma especificação de caminho de acesso incompleta, ou relativa, onde especificam-se apenas os objetos do contexto até que atinja-se um objeto que habita uma colônia aberta. Assim utiliza-se a notação $./ob_g/ob_h/.../ob_k/ob_j$ para indicar um contexto relativo desde o objeto ob_g que habita uma colônia aberta até o objeto ob_k que constringe a colônia onde habita o objeto ob_j .

Dessa maneira pode-se atingir qualquer objeto ob_j da base através da indicação $./ob_g/ob_h/.../ob_j$, onde $EstadoCol(Habita(ob_g)) = ab$.

5.8 - Predicados.

A especialização dos tipos de objetos e tipos de relacionamento em subtipos de objetos e subtipos de relacionamento, respectivamente, é decidida avaliando-se os predicados que definem os subtipos.

Um *predicado* pode ser visto como uma expressão relacionando um conjunto de *condições*. A avaliação de uma *condição* consiste da verificação de se uma das questões abaixo é satisfeita ou não:

- objeto corrente é de determinado tipo;
- objeto é de determinado subtipo.
- atributo corrente é de determinado tipo;
- objeto tem tipo de atributo;
- objeto tem atributo de determinado tipo com determinado valor;
- relacionamento corrente é de determinada modalidade;
- relacionamento corrente é de determinado tipo;
- objeto tem relacionamento de determinada modalidade;
- objeto tem relacionamento de determinada modalidade com objeto de determinado tipo;
- objeto tem relacionamento de determinada modalidade com determinado objeto;
- tripla corrente é de determinado tipo;
- tripla corrente é de determinada modalidade;
- objeto tem relacionamento de determinada modalidade de segundo grau;
- objeto tem relacionamento de determinada modalidade de segundo grau com objeto de determinado tipo na divisão 3;
- objeto tem relacionamento de determinada modalidade de segundo grau com determinado objeto na divisão 3;
- objeto tem relacionamento de determinada modalidade de segundo grau com objeto de determinado tipo na divisão 3 e objeto de determinado tipo na divisão 2;
- objeto tem relacionamento de determinada modalidade de segundo grau com determinado objeto na divisão 3, e com determinado objeto na divisão 2;
- relacionamento corrente tem tipo de atributo;
- relacionamento corrente tem atributo de determinado tipo com determinado valor;
- tripla corrente tem tipo de atributo;
- tripla corrente tem atributo de determinado tipo com determinado valor;
- objeto restringe alguma colônia;
- objeto restringe colônia de determinado tipo;

Um predicado é considerado satisfeito se a expressão que o define é satisfeita. A expressão é uma expressão lógica envolvendo os operadores E, OU e NÃO. Note-se que para um determinado estado da base (objeto corrente, relacionamento corrente, etc) todas as condições acima são constantes.

5.9 - Subtipos de Objetos.

Um objeto é uma instância de um tipo de objeto. Por sua vez, um tipo de objeto pode ter seu significado refinado, especializando-o em um subtipo. Assim podem ser definidos subtipos de objetos.

Para definirem-se subtipos de objetos, "refina-se" o conjunto de tipos de objetos $CTob(\lambda)$ em dois conjuntos distintos, $CTNob(\lambda)$ e $CTSob(\lambda)$, os quais serão definidos abaixo.

Definição 59: Define-se um conjunto $CTNob(\lambda)$, $CTNob(\lambda) \subseteq CTob(\lambda)$, como sendo o conjunto dos tipos de objetos naturais que compõem $BD(\lambda)$.

Os tipos naturais são os tipos de objetos definidos pelo usuário, não possuindo um predicado explícito conhecido pelo sistema. Por exemplo, um tipo de objeto Pessoa definido pelo usuário; o sistema "aceita" o objeto do tipo Pessoa sem verificar nenhuma condição (exceto a de que o tipo esteja definido no esquema). Ou seja, externamente à base, "alguém" define que o objeto é do tipo pessoa, mas a base não conhece as razões que permitiram a esse "alguém" chegar a essa conclusão.

Definição 60: Define-se um conjunto $CTSob = \{stob_1, \dots, stob_n\}$, $CTSob(\lambda) \subseteq CTob(\lambda)$, tal que $CTSob(\lambda) = CTob(\lambda) - CTNob(\lambda)$ como sendo o conjunto de subtipos de objetos que compõem $BD(\lambda)$. Um elemento $stob$ é definido pelo conjunto não nulo de tuplas $stob = \{ \langle Tipo-Mestre, Pred, CIni \rangle \}$, onde:

- **Tipo-Mestre** $\in CTob(\lambda)$ é um tipo de objeto do qual $stob$ é um subtipo.
- **Pred** é uma expressão booleana sobre os predicados, onde esses predicados são tais como definidos anteriormente, e aplicado a um objeto daquele tipo mestre, determina se aquele objeto é desse subtipo ou não, conforme o predicado seja satisfeito ou não.
- **CIni** é o conjunto de valores usados para inicializar atributos, relacionamentos e colônias de um objeto de tipo mestre, sempre que um objeto desse subtipo é criado.

Deve-se observar que **Tipo-Mestre** pode pertencer ao $CTNob(\lambda)$ ou ao $CTSob(\lambda)$;

Qualquer objeto de um subtipo é também do tipo mestre, o que significa que qualquer tipo de atributo ou relacionamento do tipo mestre também se aplica ao subtipo.

O inverso já não é verdade. Assim os atributos e relacionamentos de um objeto podem ser aqueles definidos para seu tipo ou para qualquer de seus tipos mestres, diretos ou recursivamente.

5.10 - Subtipos de Relacionamento.

A semântica de um relacionamento é apoiada no conceito de **modalidade de relacionamento de primeiro grau**. Uma modalidade pode ser especializada em uma nova modalidade, a qual será designada **submodalidade**. Assim, "refina-se" o conjunto de modalidades $CMr(\lambda)$ em dois conjuntos, $CMNr(\lambda)$ e $CMSr(\lambda)$, os quais serão definidos abaixo.

Definição 61: Define-se um conjunto $CMNr(\lambda)$, $CMNr(\lambda) \subseteq CMr(\lambda)$, como sendo o conjunto das modalidades naturais que compõem $BD(\lambda)$.

Definição 62: Define-se um conjunto $CMSr = \{smr_1, \dots, smr_{nsmr}\}$, $CSMr(\lambda) \subseteq CMr(\lambda)$, tal que

$CSMr(\lambda) = CMr(\lambda) - CMNr(\lambda)$, como sendo o conjunto das submodalidades que compõem $BD(\lambda)$. Cada smr é definida pelo conjunto não nulo de tuplas

$smr = \{ \langle Modalidade-Mestre, Pred, CIni \rangle \}$, onde:

- **Modalidade-Mestre** $\in CMr(\lambda)$ é a modalidade de relacionamento de primeiro grau da qual smr é uma submodalidade.
- **Pred** é uma expressão booleana sobre os predicados, onde esses predicados são definidos como anteriormente, e aplicados a uma modalidade do tipo mestre determina se aquela modalidade é desse subtipo ou não, conforme o predicado seja satisfeito ou não.
- **CIni** é o conjunto de valores usados para inicializar os atributos de um relacionamento sempre que um relacionamento desse subtipo é criado.

Deve-se observar que qualquer **Modalidade-Mestre** pode pertencer ao $CMNr(\lambda)$ ou ao $CMSr(\lambda)$;

Definição 63: Uma modalidade de segundo grau $mrt = \langle mr_1, mr_2 \rangle$ é dita uma submodalidade de segundo grau se $mr_1 \in CSMr(\lambda)$ ou $mr_2 \in CSMr(\lambda)$.

5.11 - Sumarização de Objetos.

Sumarização compreende situações em que objetos de diferentes tipos relacionam-se logicamente através de um relacionamento de sumarização (abstração de sumarização). Por exemplo, objetos do tipo Navio, Submarino e Barcaça, podem ser sumarizados através de um relacionamento de sumarização em um tipo de objeto Frota.

Definição 64: Um tipo de sumarização é um subtipo de objeto definido pela tupla $T_{\text{Sumarização}} = \langle \text{tob}, \text{CTSmob}, \text{CPr} \rangle$,

$T_{\text{Sumarização}} \subset \text{CTob}$ onde:

- **tob** é o tipo de objeto de sumarização;
- **CTSmob** é o conjunto de tipos de objetos que compõem a sumarização;
- **CPr** é o conjunto de propriedades obtidas à partir do conjunto de todos os objetos que constituem cada tipo que compõe a sumarização; ele é definido como $\text{CPr} = \{ \langle \text{tipo de atributo}, \text{naipe} \rangle \}$ onde o tipo de atributo corresponde a atributos cujos tipos podem ser associados a **tob**, e chamado tipo de atributo sumarizador.

Note-se que sendo um subtipo de um tipo de objeto, uma sumarização possui também as características de um tipo de objeto, tal como **Idtob**, **identificador interno** e **Cta**.

Um *naipe* permite obter um valor resultante que é associado ao tipo de atributo sumarizador, sendo calculado sobre todas as instâncias de um particular tipo de atributo ou tipo de objeto, de todos os objetos que compõem a sumarização. *Naipe* corresponde a uma das seguintes situações:

- **totalização** é aplicável a um tipo de atributo, e resulta no valor da soma de todos os valores de cada instância desse tipo de atributo que ocorreu em cada objeto que compõe a sumarização.
- **contagem** é aplicável a um tipo de objeto, e resulta no número de objetos desse tipo que compõem a sumarização.
- **média** é aplicável a um tipo de atributo, e resulta na média de todos os valores de cada instância desse tipo de atributo que ocorreu em cada objeto que compõe a sumarização.
- **mínimo** é aplicável a um tipo de atributo, e resulta no menor valor entre todos os valores das instâncias desse tipo de atributo que ocorreu em qualquer dos objetos que

compõem a sumarização.

- **máximo** é aplicável a um tipo de atributo, e resulta no maior valor entre todos os valores das instâncias desse tipo de atributo que ocorreu em qualquer dos objetos que compõem a sumarização.
- **função livre** é aplicável a um conjunto de tipos de atributos, e resulta em valores obtidos a partir de operações aritméticas, aplicadas em todos os valores de cada instância desse conjunto de tipos de atributos que ocorreram em todos os objetos que compõem a sumarização.

Assim $Naive \in \{ \langle "tot",tat \rangle, \langle "cont",tob \rangle, \langle "média",tat \rangle, \langle "min",tat \rangle, \langle "max",tat \rangle, \langle "func",\{tat\} \rangle \}$.

Definição 65: Uma sumarização é um objeto definido pela tupla $Sumarização = \langle TSumarização, CSmob \rangle$, onde:

- $TSumarização$ é o tipo de sumarização da qual este objeto é uma instância;
- $CSmob$ é um conjunto de objetos que compõem a sumarização.

Além disso, as seguintes restrições se impõem: sendo $TSumarização = \langle tob, CTSmob, CPr \rangle$, se

$$ob = \langle i, Idob_p, tob_p, Cat_p, Crel_i \rangle \in CSmob \rightarrow tob_i \in CTSmob, tob_i = \langle j, Idob_p, Cta_p, Crel_j \rangle ; e$$

$$\forall at_k \in Cat_i | TipoAt(at_k) = ta_k, ta_k \in Cta_j \text{ ou } ta_k \in CPr .$$

5.12 - Estado de um Tipo de Atributo.

O estado de um tipo de atributo indica a cada momento seu comportamento frente às suas instâncias na base. Deve-se ressaltar que o estado é uma característica dinâmica do gerenciador, e portanto não faz parte da base e nem do esquema, embora possa ser armazenado na base. Note que para se formalizar características dinâmicas, há a necessidade de se definir as operações possíveis sobre elas. Entretanto, aqui estão sendo formalizados apenas os aspectos estáticos da base, sendo que essas operações não serão tratadas.

Neste trabalho, foram formalizados os conceitos de estado para os atributos com característica de **propriedade** e de **sinônimo**, ou seja, sendo

$ta = \langle Idta, característica, Estado \rangle$, foram formalizados os tipos de atributo com característica = sinônimo ou com característica = propriedade.

Definição 66: Um tipo de atributo com característica de sinônimo é definido como

$ta_{SIN} = \langle Idatr, SIN, Estado \rangle$ onde $Estado_{SIN} = \{ \langle tob_p, ordem \rangle \}$, onde tob_j é o tipo de objeto ao qual o tipo de atributo está ligado, e *ordem* é a ordem que o tipo de atributo com característica de sinônimo assume diante do tipo de objeto.

Definição 67: Um tipo de atributo com característica de propriedade é definido como

$ta_{PRO} = \langle Idtar, PRO, tda, Grandeza, Estado \rangle$, onde:

- *Idtar* é o conjunto de identificadores para o tipo de atributo;
- *PRO* indica que o tipo de atributo possui característica de propriedade;
- *tda* é o tipo de dado que a propriedade possui;
- $Grandeza = \langle Idgr, \{Unidades\}, unid-norm \rangle$ com:
 - *Idgr* sendo o conjunto de identificadores definidos para a grandeza, e $\{Unidades\}$ é o conjunto de unidades possíveis definidos para essa grandeza;
 - *uni-norm* sendo a unidade normal da grandeza, que é usada tanto como padrão de referência para qualquer atributo dessa grandeza, quanto como unidade "default" caso o conjunto de unidades seja vazio;
 - $Unidade = \{Grandeza, Idun, Fator de Conversão\}$ com:
 - *Grandeza* é o conjunto de nomes da grandeza ligada a unidade;
 - *fator de conversão* é a forma utilizada para converter uma unidade em uma unidade normal para a grandeza;
 - *Idun* sendo o conjunto de identificadores definidos para a unidade;
 - *Fator de Conversão* é o fator utilizado para converter unidades;
- *Estado* = unidade corrente da propriedade;

5.13 - Esquema.

No Capítulo 3 foi descrito o conceito de esquemas para o MRO, definindo a existência de um Meta-Eschema, ou seja, a modelagem do MRO feita no próprio MRO, a qual é reconhecida implicitamente pelo gerenciador. Um esquema para uma determinada aplicação compreende um conjunto de objetos organizados logicamente pelo usuário.

Os conjuntos¹¹ CTr, CTrt, CMr, CMrt, CTCó, CTa e CTob determinam a estrutura geral da base de dados, correspondendo portanto à base de dados intensional - ou **esquema**, enquanto os conjuntos Cr, Crt, CCó, Cat e Cob mantêm a informação realmente armazenada na base de dados, e portanto correspondem à base de dados extensional.

Definição 68: Define-se um **esquema** como sendo um conjunto $CEsqu(\lambda)$, cujos elementos são os conjuntos $CTr(\lambda)$, $CTrt(\lambda)$, $CMr(\lambda)$, $CMrt(\lambda)$, $CTC\acute{o}(\lambda)$, $CTa(\lambda)$ e $CTob(\lambda)$, definidos para um determinado empreendimento λ .

5.14 - Conclusão.

Neste capítulo foi apresentada uma formalização sobre o MRO original e suas mudanças. Como resultado obteve-se uma nova "versão" para o modelo que foi denominada MRO*.

No próximo capítulo são apresentadas algumas justificativas para as modificações efetuadas sobre o MRO original.

¹¹Todos os conjuntos descritos abaixo são relativos a um determinado empreendimento λ .

Capítulo 6

Justificativas e interpretação para as Mudanças no MRO

6.1 - Introdução.

No Capítulo 4 foram apresentadas algumas mudanças efetuadas sobre o MRO durante o processo de formalização. Neste capítulo estas mudanças serão melhor justificadas em função da formalização realizada.

Segundo os conceitos de orientação a objetos, todos os elementos de um modelo são objetos, os quais possuem tipos e características que também são objetos. Além disso, existem objetos que são especializações de outros objetos, o que estabelece uma especialização entre os objetos. A formalização baseia-se também nessa conceituação, estabelecendo especializações e composições para as definições.

6.2 - Relacionamentos entre Objetos.

Os modelos de dados orientados a objetos geralmente não possuem uma maneira de representar explicitamente relacionamentos entre objetos. Entretanto, o MRO possui uma forma de representação para relacionamentos entre pares de objetos (relacionamentos binários) e entre múltiplos objetos. Um relacionamento binário segue um padrão definido no esquema denominado tipo de relacionamento binário. Um tipo de relacionamento binário no MRO compreendia a associação entre um tipo de objeto origem e um tipo de objeto destino através de uma ligação. Existem alguns casos em que tipos de relacionamento diferentes apresentam a mesma ligação, e com esta representação disponível no MRO, não havia como diferenciá-los. Essa inconsistência na definição do tipo de relacionamento foi "detectada" durante a formalização. Assim, o conceito de relacionamento no MRO foi dividido nos conceitos de Tipo de Relacionamento e *Modalidade de Relacionamento de Primeiro Grau*¹ no MRO*. Essa modalidade passou a ditar o "modo" como um tipo de relacionamento binário é associado.

Assim, um tipo de relacionamento binário passou a ser definido no MRO*, como

¹Por simplicidade, no decorrer do texto será usado modalidade de primeiro grau, ao invés de modalidade de relacionamento de primeiro grau.

sendo composto por um tipo de objeto origem, um tipo de objeto destino e uma modalidade de primeiro grau. Formalmente, um tipo de relacionamento pode ser representado por

$tr = \langle tob_o, tob_d, mr \rangle$. Identifica-se uma "simetria" em mr implicando na obrigatoriedade

da existência de uma modalidade de primeiro grau oposta mr_{op} para cada mr . Logo, todo tipo de relacionamento possui um oposto que é formalmente definido por

$tr_{op} = \langle tob_d, tob_o, mr_{op} \rangle$. Note que os tipos de objeto origem e destino se invertem em

relação ao tipo de relacionamento do qual este tipo de relacionamento é oposto.

Um tipo de relacionamento é instanciado em um relacionamento binário, porém uma modalidade de primeiro grau não é instanciada, permanecendo a mesma para o tipo e para sua instancição. No MRO* um relacionamento binário passou a ser definido como sendo a composição de um objeto origem, um objeto destino, uma modalidade de relacionamento de primeiro grau e um conjunto de atributos ligado a ele. Formalmente, o relacionamento binário ficou definido como $r = \langle ob_o, ob_d, mr, Cat \rangle$, enquanto o relacionamento binário oposto ficou definido como $r_{op} = \langle ob_d, ob_o, mr_{op}, Cat \rangle$.

Note que "modalidade" é um conceito novo, sobre o qual está baseada toda a semântica de como deve ser reconhecido e tratado um tipo de relacionamento e sua instancição. No MRO* a associação de tipos de atributos a tipos de relacionamento é feita vinculando-se os tipos de atributos à modalidade de primeiro grau, sendo indiferente vinculá-los a modalidade de primeiro grau ou a sua oposta. Os atributos de um relacionamento são os mesmos de seu oposto, e são dos tipos vinculados à modalidade desse relacionamento ou à modalidade oposta a esse relacionamento. Uma modalidade de relacionamento de primeiro grau foi formalizada como $mr = \langle mr_{op}, Idmr, Ctar \rangle$, onde $Idmr$ é o conjunto de identificadores da modalidade, $Ctar$ é o conjunto de tipos de atributos que podem ser associados a mr . A mr_{op} foi formalizada como $mr_{op} = \langle mr, Idmr_{op}, Ctar_{op} \rangle$. Assim os atributos de um relacionamento de modalidade mr , bem como de seu relacionamento oposto, devem ser de tipos que pertençam a $Ctar$ ou a $Ctar_{op}$.

Um modelo de dados reconhece diversas maneiras como objetos se relacionam. Por exemplo, o conceito de objeto complexo e o próprio conceito de generalização são formas de relacionamento. Um objeto complexo é composto por outros objetos, e em um objeto de um subtipo indica-se que seu tipo mestre é um objeto mais genérico. composto por e

mestre é são modalidades de relacionamento, cujas modalidades opostas são respectivamente compõem e subtipo é.

Os tipos de relacionamento que se podem construir com essas modalidades dependem dos tipos de objetos que existem em cada empreendimento. Por exemplo, numa base de dados BD(veículos) que tenha os tipos de objetos CTob(veículos) = {VEÍCULO, CARRO, MOTOR e SUSPENSÃO}, pode-se definir o conjunto de tipos de relacionamentos CTr(veículos) = {<VEÍCULO, CARRO, subtipo é >, <VEÍCULO, MOTOR, composto por>, <CARRO, SUSPENSÃO, composto por>}. O conjunto de objetos Cob(veículos) são as ocorrências de VEÍCULO(s), CARRO(s), MOTOR(es) e SUSPENSÃO(sões) na base de dados, e o conjunto de relacionamentos Cr(veículos) seriam os relacionamentos que existem entre esses objetos.

Note-se que um sistema de base de dados orientado a objeto geralmente reconhece o significado das modalidades composto por e mestre é, de maneira que conceitos como por exemplo, herança e dependência existencial podem ser "automatizados" pelo sistema. Já outras modalidades de relacionamentos, que possam ser "definidas pelo usuário" não têm seu significado reconhecido a priori pelo sistema, e portanto a forma como as operações decorrentes devem ser executadas têm que ser especificadas pelo usuário. Por exemplo, um sistema de base de dados reconhece a priori a maneira pela qual relacionamentos de generalização permitem que informações sejam herdadas entre os objetos que estão envolvidos no relacionamento. Para que o sistema permita que o usuário defina suas próprias modalidades de relacionamento, ele terá que fornecer recursos para definir como ocorre (e se ocorre) herança em cada modalidade, como devem ser tratadas diferentes formas de dependência, transitividade, simetria (ou anti-simetria), etc em cada modalidade, bem como formas de propagação de operações executadas pela base envolvendo modalidades.

Por exemplo, a modalidade pertence cuja oposta é possui, pode indicar que os atributos de valor daquilo que é possuído sejam creditados a quem é possuidor, bem como o identificador do que é possuído seja propagado ao possuidor (forma de herança). Assim um tipo de relacionamento pessoa possui residência, ou empresa possui veículo faz com que a identificação da residência (ser endereço) seja propagado à pessoa, ou a identificação do veículo (sua placa) seja propagada à empresa, bem como indica que quem paga os impostos da residência é a pessoa que o possui, e quem paga os impostos do veículo é a empresa que o possui.

Verifica-se assim que o conceito de modalidade de relacionamento de primeiro grau carrega toda a semântica do conceito de relacionamentos em bases de dados orientadas a objetos, ao passo que relacionamentos e tipos de relacionamentos caracterizam a sintaxe de

como os objetos se associam. Portanto, a definição de uma modalidade consiste em se "estruturar" essa semântica, através da parametrização de como as operações envolvidas com cada modalidade devem ser executadas. Desse modo acredita-se que o elemento modalidade de primeiro grau é de extrema importância na modelagem de relacionamentos, justificando-se plenamente sua identificação.

O MRO apresenta uma forma particular de relacionamento que envolve múltiplos objetos, o qual é chamado de relacionamento triplo. Um relacionamento triplo obedece a um padrão definido no esquema da base, denominado tipo de relacionamento triplo. Um relacionamento triplo é composto por três divisões designadas D1, D2, e D3, onde: existe um tipo de objeto em D1 e um tipo de objeto em D3; existem um ou mais tipos de objetos em D2. O tipo de objeto da divisão D1 forma com o tipo de objeto da divisão D3, um tipo de relacionamento binário chamado *tipo de relacionamento de determinação*. O mesmo tipo de objeto da divisão D1 forma com os objetos da divisão D2, tipos de relacionamentos binários chamados *tipos de relacionamentos de conjunto*. A "ligação" desses tipos caracteriza o tipo de relacionamento triplo (tripla). Esta ligação recebeu o nome de *modalidade de relacionamento de segundo grau*, e origina-se da composição das modalidades de primeiro grau envolvidas nos tipos de relacionamentos de conjunto, e no tipo de relacionamento de determinação. Formalmente, um tipo de relacionamento triplo é definido como

$trt = \langle tob_{d1}, \{tob_{d2}\}, tob_{d3}, mrt \rangle$ onde mrt é a modalidade de relacionamento de segundo

grau. Considerando mr_1 e sua oposta mr_{1op} , mr_2 e sua oposta mr_{2op} , como sendo as modalidades de determinação e de conjunto pertencentes ao relacionamento, tem-se as seguintes possíveis composições para mrt :

$\langle mr_1, mr_2 \rangle$, $\langle mr_1, mr_{2op} \rangle$, $\langle mr_{1op}, mr_2 \rangle$ e $\langle mr_{1op}, mr_{2op} \rangle$.

O tipo de relacionamento triplo instancia-se em um relacionamento triplo. Formalmente, um tipo de relacionamento triplo foi formalizado como

$rt = \langle ob_{d1}, \{ob_{d2}\}, ob_{d3}, mrt \rangle$. Note-se que uma tripla estabelece uma sintaxe, e não

uma semântica. Ou seja, uma tripla estabelece uma estrutura sobre a sintaxe da composição de dois ou mais relacionamentos binários através das modalidades de relacionamento de primeiro grau do qual é composta, possuindo assim, uma semântica resultante.

Deve-se observar também que as diferentes composições para mrt permitem "navegar" sobre o relacionamento triplo. Deste modo, se for usada $mrt = \langle mr_1, mr_2 \rangle$, então o "sentido" a ser seguido na tripla é partindo do objeto da divisão D1 para os da divisão

D2, e depois para a divisão D3; se for usada $mr t = \langle mr_1, mr_{2op} \rangle$, então o "sentido" a ser seguido na tripla é o de partir do objeto da divisão D3, seguir para o objeto da divisão D1, e depois da divisão D1 para a D2; se for usada $mr t = \langle mr_{1op}, mr_2 \rangle$, então o "sentido" a ser seguido na tripla é o de partir de um objeto da divisão D2 para o objeto da divisão D1, e do objeto de D1 para o objeto de D3; se for usada $mr t = \langle mr_2, mr_1 \rangle$, então o "sentido" a ser seguido na tripla é partir do objeto da divisão D1 primeiro para o objeto da divisão D3, e depois do objeto da divisão D1 para o objeto da divisão D2.

Note-se que um tipo de relacionamento triplo é uma das possíveis especializações que um tipo de relacionamento ternário pode ter. Embora outras especializações pudessem ser estudadas, e ser formal e semanticamente definidas, considera-se que a forma de triplas atende a maioria dos casos, sem acrescentar uma complexidade excessiva ao modelo. O mesmo ocorre com relacionamentos de ordem maior que 3 (quaternários, etc).

Deve-se notar, que é possível "compor" tipos de relacionamentos triplos, para se representar tipos de relacionamentos mais elaborados, como o tipo de relacionamento ternário, quádruplo, etc, o que caracteriza flexibilidade e força do conceito.

6.3 - Estado de um Tipo de Atributo.

O conceito de estado de um tipo de atributo tem sido mencionado superficialmente desde a elaboração do MRO, mas até agora não havia sido explorado. Todas as características de atributo do MRO devem ter um estado definido. Neste trabalho foram explorados os estados de tipos de atributo com característica de sinônimo e os tipos de atributo com característica de propriedade. O estado de tipos de atributos de outras características devem ser tratadas em trabalhos que cuidem de cada característica específica, tal como Tempo, Regra e Gráfico.

6.3.1 - Tipos de atributo com característica de sinônimo.

Um sinônimo era reconhecido simplesmente como um nome alternativo para identificar-se um determinado objeto. Um objeto poderia ter vários identificadores, sendo que um era escolhido como o nome principal, e os outros eram denominados sinônimos. Agora todos os identificadores são sinônimos, os quais possuem um estado que determina

qual deles é o identificador no momento, ou seja, o estado corrente dos tipos de sinônimos de um tipo de objeto indica qual dos tipos de sinônimo corresponde nesse instante ao seu nome principal. Assim, não existe um mais um nome principal. Na Figura 42 pode-se ver como um tipo de atributo com característica de sinônimo é modelado utilizando o próprio MRO. No formalismo é representado por

$ta_{sin} \langle idatr, SIN, Estado \rangle$,

onde $Estado_{sin} = \langle tob_j, Ordem \rangle$

onde tob_j é o tipo de objeto ao qual o tipo de atributo está ligado, e $Ordem$ é a ordem que o tipo de atributo com característica de sinônimo assume diante do tipo de objeto. A existência de ordem entre mais de um tipo de sinônimo para um mesmo tipo de objeto, flexibiliza a forma cada como objeto é referenciado, pois um objeto que não tenha um valor para um tipo de sinônimo pode usar um valor de tipo de sinônimo seguinte na ordem, e assim sucessivamente.

Figura 42 : Tipo de atributo com característica de sinônimo, modelado pelo meta-esquema do MRO.

6.3.2 - Tipos de atributo com característica de propriedade.

Tipos de atributo que não possuem uma característica particular que os identifique, são classificados como possuindo característica de **propriedade**. O tipo de atributo com característica de propriedade pode ser modelado como na Figura 43 . O seu estado foi formalizado como sendo $ta_{pro} \langle idatr, PRO, tda, Grandeza, Estado \rangle$, onde **Grandeza** = $\langle \{idpro\}, \{Unidade\} \rangle$, **Unidade** = $\langle \{iduni\}, Grandeza, Fator de Conversão \rangle$. O estado é dado por $Estado_{pro} =$ unidade corrente. O conjunto $\{idpro\}$ é o conjunto de identificadores da Grandeza, e o conjunto $\{iduni\}$ é o conjunto de identificadores da Unidade.

Por exemplo, **Altura** = $\langle \text{"altura"}, PRO, \text{real}, \langle \text{comprimento}, \{km, m, cm, \dots\} \rangle \rangle$

quarteirão} >, cm} > e Distância-para-Casa = <"caminhada", PRO, float, <comprimento, {km, m, cm, quarteirão} >, quarteirão >.

Assim definido, o estado do tipo de atributo com característica de propriedade, permite "amostrar" atributos com mais de um domínio, e executar conversão automática de domínio quando possível. Então é necessária a definição de critérios de conversão, os quais devem ser naturais a cada propriedade. Como exemplo de possíveis conversões, pode-se

citar conversões entre números através de expressões, tabelas entre literais e numéricos, entre textos formatados com processadores diferentes, etc.

Figura 43 :Tipo de atributo com característica de propriedade, modelado pelo meta esquema do MRO.

6.4 - Subtipos.

Um subtipo indica uma especialização de um tipo. Assim, um subtipo de objeto indica uma especialização de um tipo de objeto ou de outro subtipo de objeto, enquanto um subtipo de relacionamento indica uma especialização de um subtipo de um relacionamento ou de outro subtipo de relacionamento.

6.4.1 - Subtipos de Objetos.

Como já descrito no Capítulo 4, um subtipo de objeto indica um "refinamento" de um tipo de objeto ou de outro subtipo de objeto. No MRO, um subtipo era especificado como possuindo:

- um tipo mestre, o qual é um tipo ou um subtipo previamente especificado;

- um predicado, que estabelece o critério de como escolher os objetos do tipo mestre que constituirão o conjunto de objetos daquele tipo através de atributos e relacionamentos;
- um conjunto de valores de atributos e de relacionamentos padrão que podem ser usados para inicializar os valores ainda não definidos de um objeto desse subtipo, quando este é criado.

Um subtipo refina um outro tipo (ou subtipo) implicando em uma especialização do tipo (ou subtipo) refinado. Esta especialização leva ao conceito de *herança*. No MRO, a herança ocorre através desta especialização, implicando que um subtipo é também do tipo mestre, significando que qualquer tipo de atributo ou relacionamento do tipo mestre também se aplica ao subtipo. Assim, os atributos e relacionamentos de um objeto podem ser aqueles definidos para seu tipo ou para qualquer de seus tipos mestres, diretos ou recursivamente. Assim, um subtipo foi formalizado como

$stob = \{ \langle \text{Tipo-Mestre}, Pred, CIni \rangle \}$, onde **Tipo-Mestre** é um tipo (ou outro subtipo já

definido) $tob = \langle i, Idtob, Cta, Ctrel \rangle$. Um subtipo de objeto indica uma especialização de um tipo (ou subtipo já existente). Assim, uma definição completa de um subtipo de objeto é $stob = \langle i, Idtob, Cta, Ctrel, \{ \langle \text{Tipo-Mestre}, Pred, CIni \rangle \} \rangle$, o que indica que um subtipo além de ter um novo código (i) permite expandir o conjunto de identificadores, tipos de atributos e de relacionamentos que um tipo mestre já possui. Isso mostra matematicamente o conceito de herança, pois cada **Tipo-Mestre** possuirá seu conjunto de tipos de atributos e de relacionamentos, que também faz parte do subtipo. O conceito de herança múltipla também é suportado através da especialização de subtipos. Note que a definição de subtipo é dada como um conjunto de tuplas, podendo haver mais de um tipo mestre. Assim, um subtipo possui um **Tipo-Mestre**, o qual pode ser um subtipo de outro tipo, possuindo então um **Tipo-Mestre**, o qual pode ser subtipo de outro, e assim por diante, de maneira que é estabelecida uma hierarquia de tipos. A herança múltipla ocorre quando um subtipo possui mais de um **Tipo-Mestre** em um mesmo nível, caracterizando a herança dos tipos de atributos e de relacionamentos de apenas um, ou de todos os tipos mestres do mesmo nível, conforme a especialização efetuada pela aplicação. No MRO restringe-se a especialização de subtipos, de modo que um subtipo possa possuir quantos tipos mestres for conceitualmente necessário, porém, deve possuir apenas um **Tipo-Mestre** no topo da hierarquia. Com o processo de formalização, classificou-se este tipo do topo da hierarquia como sendo um **tipo natural**, ou seja, não é subtipo de nenhum tipo. Logo, no MRO*, o conjunto de tipos de objetos $CTob(\lambda)$ foi dividido em dois sub-conjuntos, os quais foram designados $CTNob(\lambda)$ e $CTSob(\lambda)$, sendo chamados respectivamente conjunto dos tipos

naturais de objetos e conjunto dos subtipos de objetos. O conjunto $CTNob(\lambda)$ compreende todos os tipos de objetos definidos pelo usuário, e que não possui um predicado explícito conhecido pelo sistema. O conjunto $CTSob(\lambda)$ compreende o conjunto de todos os subtipos de objetos que compõem $BD(\lambda)$.

6.4.2 - Subtipos de Relacionamentos.

No MRO, um subtipo de relacionamento indica uma especialização de um tipo de relacionamento ou de um outro subtipo de relacionamento, sendo caracterizado pelos atributos de seu tipo de relacionamento mestre. O subtipo é composto de um **Tipo-Mestre** (tipo ou subtipo de relacionamento), um predicado e um conjunto de valores de inicialização.

No MRO^* , o conceito de tipo de relacionamento foi dividido em tipo de relacionamento e modalidade de relacionamento de primeiro grau. Como já descrito acima, a semântica do tipo de relacionamento ficou centrada sobre a modalidade de relacionamento de primeiro grau. Durante o processo de formalização foi redefinido o subtipo de relacionamento, seguindo a evolução ocorrida com o novo conceito de modalidade. Assim no MRO^* , um subtipo de relacionamento ficou definido sobre a modalidade de relacionamento de primeiro grau, definindo-se uma submodalidade de relacionamento de primeiro grau, a qual especializa a modalidade envolvida. Um subtipo de relacionamento foi formalizado como $smr = \{ \langle Modalidade-Mestre, Pred, CIni \rangle \}$ onde

modalidade-mestre é definida como $mr = \langle mr_{op}, Idmr, Cta \rangle$, onde mr_{op} é a modalidade oposta, $Idmr$ é um conjunto de identificadores para a modalidade mr , e Cta é o conjunto de tipos de atributos que podem ser associados a mr . Note que Cta está vinculado à modalidade mr , estabelecendo assim, a semântica do subtipo de relacionamento sobre mr . Note-se que conceitualmente apenas a modalidade de primeiro grau é especializada em submodalidade no MRO^* . Um subtipo de relacionamento é apenas um tipo de relacionamento onde a modalidade envolvida é de uma submodalidade. O mesmo ocorre com a modalidade de segundo grau e os tipos de relacionamento triplos: uma submodalidade de segundo grau é uma modalidade de segundo grau onde uma das modalidades de primeiro grau que a compõem é uma submodalidade de primeiro grau.

Da mesma maneira que o **Tipo-Mestre** do subtipo de objeto, a **Modalidade-Mestre** do subtipo de relacionamento constitui uma hierarquia de modalidades, e pela definição do MRO^* , um subtipo de relacionamento pode possuir quantas modalidades mestres forem

necessárias, porém no topo da hierarquia deve haver apenas uma modalidade de relacionamento de primeiro grau, chamada *modalidade de relacionamento de primeiro grau natural*. Assim, o conjunto de modalidades de primeiro grau $\text{CMr}(\lambda)$ foi dividido em dois sub-conjuntos, os quais foram designados $\text{CMNr}(\lambda)$ e $\text{CMSr}(\lambda)$, sendo chamados respectivamente conjunto das modalidades naturais e conjunto das submodalidades de relacionamento. O conjunto $\text{CMNr}(\lambda)$ compreende todas as modalidades de relacionamento naturais de $\text{BD}(\lambda)$. O conjunto $\text{CMSr}(\lambda)$ compreende o conjunto das submodalidades de relacionamento de $\text{BD}(\lambda)$.

6.4.3 - Sumarização de Objetos.

No MRO, um tipo de objeto modelado como resultado de um relacionamento de sumarização era denominado **Supertipo de Objeto**. No MRO' passou a ser chamado de **Sumarização de Objetos**.

Um relacionamento de sumarização entre objetos corresponde à composição de mais de um tipo ou subtipos diversos dando origem a novos objetos, que se constituem em objetos que representam conjuntos de objetos dos tipos e subtipos que compõem a sumarização. Os objetos de uma sumarização caracterizam coletivamente os conjuntos de objetos dos tipos e subtipos que compõem a sumarização, e caracterizam sumários (ou agregados) daqueles conjuntos de objetos. Os tipos de objetos sumarizados podem ter relacionamentos com tipos e subtipos, bem como atributos. Esses elementos ligados à sumarização caracterizam relacionamentos ou atributos que são identificáveis para todos os objetos de um dado tipo de objeto de sumarização, ou que caracterizam a sumarização como um todo. Os atributos de uma sumarização que não são herdados pelos objetos que constituem os conjuntos que compõem a sumarização, são obtidos a partir do conjunto de todos os objetos que constituem cada tipo que compõe a sumarização. Cada um desses atributos deve ser definido através de uma operação especificada no domínio do tipo da característica daquele atributo, como por exemplo, somatório, mínimo, máximo, etc, para uma propriedade com tipo de dado numérico. Durante o processo de formalização, algumas dessas operações foram definidas.

Um tipo de sumarização foi formalizado pela tupla

$T_{\text{Sumarização}} = \langle \text{tob}, \text{CTSmob}, \text{CPr} \rangle$ onde:

- **tob** são os elementos que constituem um tipo de objeto;
- **CTSmob** é o conjunto de tipos de objetos que compõem a sumarização;

• **CPr** é o conjunto de propriedades obtidas a partir do conjunto de todos os objetos que constituem cada tipo que compõe a sumarização. Ele é definido como $CPr = \{ \langle \text{tipo de atributo}, \text{naipe} \rangle \}$, onde o tipo de atributo é associado a *tob*, e chamado tipo de atributo sumarizador.

Como uma sumarização é um tipo de objeto, pode-se dizer que a especificação completa de uma sumarização é

$TSumarizacao = \langle i, Idtob, Cta, Ctrel, CTSmob, CPr \rangle$. Assim um objeto

$obj = \langle j, Idobj, TSumarizacao, Cat, Crel \rangle$ tem cada atributo $at_r \in Cat$ de um tipo

$TipoAt(at_r) \in Cta$ ou $TipoAt(at_r) \in CPr$. Se $TipoAt(at_r) \in Cta$, o valor desse atributo é armazenado na base. Se $TipoAt(at_r) \in CPr$, então o valor não pode ser definido pelo usuário, sendo calculado segundo o naipe desse atributo sempre que o valor for consultado.

Cada naipe de cada tipo de atributo pertencente a **CPr**, tem uma estrutura própria, envolvendo cada um, um conjunto próprio de elementos do modelo. Por exemplo, um naipe de totalização envolve um tipo de atributo, enquanto um naipe de contagem envolve um tipo de objeto. Essa flexibilidade foi criada por dois motivos: primeiro, permite que a semântica do mundo real seja modelada de forma natural, tal como ela realmente é; e segundo, abre espaço para que uma particular implementação do modelo, voltada para um domínio de aplicações específico, possa ser especificado.

6.5 - Conclusão.

Durante o processo de formalização foram detectadas e corrigidas inconsistências no modelo; foram descritos conceitos que haviam sido definidos no MRO, mas não explorados; e foram validadas as definições propostas para o modelo. Deve-se ressaltar que criou-se um novo conceito chamado modalidade de relacionamento de primeiro grau, esclarecendo a semântica da representação de um tipo de relacionamento no modelo, o qual passa a ser uma proposta da representação de relacionamentos em modelos de dados orientados a objetos em geral.

Essas alterações trouxeram mais clareza a alguns conceitos e definiram outros. Com isso o modelo ficou melhor definido, abrindo espaço para a definição de outros conceitos que poderão provir dessas definições.

Capítulo 7

Conclusões e Propostas de Futuras Pesquisas

7.1 - Conclusões.

O **MRO*** compreende o **MRO** original modificado e formalizado. A formalização do modelo possibilitou retirar inconsistências e "descrever" conceitos que estavam definidos mas não explorados. Estabelecido um formalismo matemático sobre o modelo, há então a possibilidade de comparação com outros modelos, o que até então, só era possível informalmente. Agora também é possível avaliar a força representativa do modelo através dos conceitos formalizados.

As principais contribuições deste trabalho são:

- A proposta de um modelo de dados atualizado **MRO***, o qual acrescenta ao **MRO** novos conceitos:
 - estado de tipo de atributo;
 - redefinição do conceito de nome principal de objeto;
 - definição clara de como é a hierarquia de subtipos;
 - definição clara de como é implementada a abstração de sumarização.
- Formalização do modelo resultante, permitindo:
 - garantia de consistência e ausência de ambigüidade no modelo;
 - delimitação da capacidade de representação semântica do modelo;
 - fixar diretrizes para implementação de aspectos antes obscuros no modelo;
 - comparação da capacidade de representação desse modelo com outros cuja formalização existe na literatura;
- Eliminação de inconsistências no modelo original, desenvolvendo e especificando claramente os seguintes conceitos:
 - modalidade de relacionamento de primeiro grau;
 - submodalidade de relacionamento de primeiro grau;
 - associação de atributos a modalidades de primeiro grau, tipos de relacionamento e relacionamento;

- predicados de subtipos;
 - naipes de sumarização;
 - tipo e tipos naturais de objetos e de relacionamentos;
- Integração do conceito de relacionamentos dentro do paradigma de orientação a objetos, com a introdução dos conceitos de:
 - modalidade de relacionamento;
 - semântica de relacionamentos e abstrações vinculadas a modalidade de relacionamento.

7.2 - Sugestões de futuras pesquisas.

Com o processo de formalização, o MRO evoluiu sendo designado MRO*. Vários trabalhos de pesquisa podem ser desenvolvidos a partir destes, e entre eles pode-se citar:

- foi formalizada apenas a parte estática do modelo, restando toda a parte dinâmica a ser formalizada. Assim, operações de inserção e eliminação na base e no esquema, evolução de esquemas, controle de concorrência e distribuição, versões de colônias, base dinâmica, regras, tabelas de operações implícitas, manipulação de mensagens, interação com sistemas externos à base de dados, subsistema de visualização e interface com o usuário, etc, são alguns dos aspectos dinâmicos que necessitam ser formalizados;

- o conceito de estado de um tipo de atributo foi explorado apenas sobre aqueles tipos de atributos que possuem característica de sinônimo ou de propriedade. Desse modo, restam o estado de todas as outras características a serem estudadas e definidas, o que deve ser feito quando cada característica for estruturada;

- há a necessidade de se corrigir a implementação para suportar as alterações efetuadas sobre o modelo; por outro lado, conceitos que não estavam claros, agora podem ser implementados;

- pode-se estruturar uma ferramenta de análise utilizando o MRO*, especificando-se e automatizando passos formais de análise, gerando assim uma metodologia de análise orientada a objetos apoiada no MRO*;

- o processo de modelagem de dados utilizando o MRO* pode ser empreendido, com o desenvolvimento de uma metodologia de modelagem de dados específica para o MRO;
- pode ser construída uma ferramenta para apoio ao processo de modelagem de dados utilizando a metodologia desenvolvida, a qual utilize por sua vez a implementação do MRO existente, armazenando a informação para o funcionamento da ferramenta numa Base MRO. Tal ferramenta poderia, como um de seus recursos, gerar o esquema de dados da aplicação modelada.

BIBLIOGRAFIA

- [Abiteboul-87] Abiteboul, S. e Hull R. - "TFO: A Formal Semantic Database Model". *ACM Transactions on Database Systems*, Vol.12, Nº4, Dezembro 1987, pg. 525-565.
- [Atkinson-89] Atkinson, M. et alli - "The Object Oriented Database System Manifesto". *Technical Report Altair 30-89*, GIP ALTAIR IN 2-INRIA-LRI, Outubro 1989.
- [Banerjee-87] Banerjee, J. e Kim, W. - "Semantics and Implementation of Schema Evolution in Object-Oriented Databases". *ACM SIGMOD 87 Conference*, San Francisco, Califórnia, Maio 1987, pg. 311-322.
- [Boudier-88] Boudier, G. et alli - "An Overview of PCTE and PCTE+". *ACM SIGMOD 88 Conference*, Chicago, Illinois, Junho 1988, pg. 248-257.
- [Calônego-91] Calônego Jr., N. - *Desenvolvimento de um Núcleo Multi-Usuário para um Sistema de Gerenciamento de Base de Dados Orientada a Objetos*. Dissertação de Mestrado apresentada ao ICMSC-USP, Julho 1991.
- [Carnolesi-91] Carnolesi Jr., Luiz - *Gerenciamento de Esquema e Esquemas Suplementares em Gerenciadores de Objetos*. Mini-dissertação de Mestrado apresentada ao ICMSC-USP, Dezembro 1991.
- [Carey-88] Carey M.J. et alli - "A Data Model and Query Language for EXODUS". *ACM SIGMOD 88 Conference Chicago*, Illinois, Junho 1988.
- [Chen-76] Chen, P. P. - "The Entity-Relationship Model-Toward a Unified View of Data". *ACM Transactions on Database Systems*, Vol.1, Nº1, Março 1976, pg. 9-36.
- [Chen-90] Chen, P. P. - *Gerenciamento de Banco de Dados - A Abordagem Entidade-Relacionamento para Projeto Lógico*, McGraw Hill, 1990.
- [Codd-70] Codd, E.F. - "A relational model of data for large shared data banks". *Communications ACM*, Vol.13, Nº6, Junho 1970, pg. 377-387.
- [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.
- [Corrêa-92] Corrêa, P.S. - *Uma Linguagem para o Gerenciador de Objetos apoiado no Modelo de Representação de Objetos*. Dissertação de Mestrado apresentada ao ICMSC-USP, Junho 1992.

- [Date-75] Date, C.J. - *An Introduction to Database Systems*. Addison-Wesley Publishing Company, 1975.
- [Deux-90] Deux O. et alli - "The Story of O₂". *IEEE Transactions on Knowledge and Data Engineering*, Vol.2, Nº1, Março 1990, pg. 91-108.
- [Elmasri-90] Elmasri, R. e Navathe, S.B. - *Fundamentals of Database Systems*. Addison-Wesley Publishing Company, 1990.
- [Ferreira-91] Ferreira, J.E. - *Estudo da Distribuição de uma Base de Dados Apoiada no Modelo de Representação de Objetos*. Dissertação de Mestrado apresentada ao IFQSC-USP - 1991.
- [Heuser-91] Heuser, C. A. - *Modelagem Conceitual de Sistemas - Redes de Petri*. VEBAI - Escola Brasileira e Argentina de Informática, Janeiro de 1991.
- [Hull-87] Hull R. e King R. - "Semantic Database Modeling: Survey, Applications and Research Issues". *ACM Computing Surveys*, Vol.19, Nº 3, Setembro 1987, pg. 201-259.
- [Jackson-91] Jackson M. S. - "Tutorial on object-oriented databases". *Information and Software Technology*, Vol.33, Nº1, Janeiro e Fevereiro de 1991, pg. 4-12.
- [Kim-89a] Kim, W. et alli - "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, Março 1990, 109-124, pg. 327-341.
- [Korth-86] Korth, H. F. e Silberschatz, A. - *Sistemas de Bancos de Dados*, McGraw-Hill, 86.
- [Lécluse-88] Lécluse C. et alli - "O₂, an Object-Oriented Data Model". *ACM SIGMOD International Conference on Management of Data*, Junho 1988, pg. 425-433.
- [Maier-83] Maier, D., *The Theory of Relational Databases*, Oregon Graduate Center, Computer Science Press, 1983.

- [McLeod-81] McLeod D. e Hammer M. - "Database Description with SDM: A Semantic Database Model". *ACM Transactions 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, Janeiro e Fevereiro de 1991.
- [NG-81] NG, P.A. - "Further Analysis of the Entity-Relationship Approach to Database Design". *IEEE Transactions on Software Engineering*, Vol. SE-7, Nº1, Janeiro 1981, pg. 85-99.
- [Peckham-88] Peckham, J e Maryanski, F. - "Semantic Data Models". *ACM Computing Surveys*, Vol.20, Nº3, Setembro 1988, pg. 153-189.
- [Santos-92] Santos, R.R. - *Tratamento de Atributos com Características de Regras em Modelos de Dados Orientados a Objetos*. Mini-dissertação de Mestrado apresentada ao ICMSC-USP, Fevereiro 1992.
- [Silva-91] Silva, R.C. - *Armazenamento e Gerenciamento de Informações Temporais em um Modelo de Bases de Dados Orientado a Objetos*. Mini-dissertação de Mestrado apresentada ao ICMSC-USP, Outubro 1991.
- [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, Março 1981, pg. 140-173.
- [Sibley-77] Sibley, E.H. e Kershberg, L. - "Data Architecture and Data Model Considerations". *Proc. AFIPS Nat. Computer Conf.*, Dallas, Texas, Junho 1977, pg. 185-96.
- [Smith-77a] Smith, J.M., e Smith, D.C.P - "Database Abstractions: Aggregation". *Communication of the ACM*, Vol.20, Nº6, Junho 1977, 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, Junho 1977, pg. 105-133.
- [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.
- [Straube-90] Straube, D. D., e Özsu M. T. - "Queries and Query Processing in Object-Oriented Database Systems". *ACM Transactions on Information Systems*, Vol.8, Nº 4,

Outubro 1990, pg. 387-430.

[Su-86] Su, S.Y.W. - "Modeling Integrated Manufacturing Data with SAM". *IEEE Computer*, Vol.19, Nº1, Janeiro 1986, pg.34-39.

[Takai-91] Takai, O.K. - *Representação de Atributos com Características Gráficas em uma Base de Dados Orientada a Objetos*. Mini-dissertação de Mestrado apresentada ao ICMSC-USP, Junho 1991.

[Taylor-76] Taylor, R. W., e Frank, R. L. - "CODASYL database management systems". *Computer Survey*, Vol.8, Nº1, Março de 1976, pg. 67-104.

[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-88a] Traina Jr., C. e Slaets, J.F.W. - "Gerenciamento de Esquemas de Dados Dinamicamente Modificáveis em Modelo de Base de de Dados Orientado a Objetos". In *Anais do XIV Conferência Latinoamericana de Informática*, Buenos Aires, Argentina, Setembro de 1988.

[Traina-88b] Traina Jr., C. e Slaets, J.F.W. - "Um Modelo de Representação de Objetos". In *Anais do 3o. Simpósio Brasileiro de Banco de Dados*, Recife, pg. 227-242, Março de 1988.

[Traina-90] Traina Jr., C. e Takai, O. K. - "Implementação de um Gerenciador de Identificadores de Objetos, com Chaves de Tamanho Variável, para o Gerenciador de Objetos do Modelo de Representação de Objetos". In *Anais do 5o. Simpósio Brasileiro de Banco de Dados*, Rio de Janeiro, 1990.

[Traina-91] Traina Jr., C. - "GEO - Um Sistema de Gerenciamento de Bases de Dados Orientado a Objetos - Estado Atual de Desenvolvimento e Implementação". In *Anais do 6º Simpósio Brasileiro de Banco de Dados*, Manaus, Am., 1991.

[Traina-92] Traina Jr., C. e Slaets, J.F.W. - "Um Modelo de Representação de Objetos". In *Notas ICMSC - Nº104* - Fevereiro de 1992.

[Tsichritzis-76] Tsichritzis, D. C., e Lochovsky, F. H. - "Hierarchical database management: A survey". *ACM Computing Surveys*, Vol.8, Nº1, Março de 1976, pg. 105-124.

[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.

[Wilkinson-90] Wilkinson, K. et alli - "The Iris Architecture and Implementation". *IEEE Transactions on Knowledge and Data Engineering*, Vol.2, Nº1, Março 1990.

Apêndice

Conceitos Básicos

Conjuntos

•Um conjunto é formado de objetos, os quais são chamados seus elementos. Quando um objeto x é um dos elementos que compõem o conjunto A , diz-se que x pertence a A e escreve-se $x \in A$. Caso x não seja um dos elementos de A , diz-se que x não pertence a A e escreve-se $x \notin A$. Um conjunto A é representado como

$A = \{x \mid x \text{ obedece uma determinada propriedade}\}$. Quando não existe elemento $x \in A$, diz-se que A é vazio, e representa-o como $A = \{\}$ ou $A = \emptyset$.

•Dados dois conjuntos A e B , diz-se que A é subconjunto de B quando todo elemento de A é também elemento de B . Para indicar esse fato, usa-se a notação $A \subset B$. Quando

$A \subset B$, diz-se também que A é parte de B , ou que A está incluído em B , ou contido em B . A relação $A \subset B$ chama-se relação de inclusão.

•Dado um conjunto X , indica-se com $P(X)$ o conjunto cujos elementos são as partes de X . Em outras palavras, afirmar que $A \in P(X)$ é o mesmo que dizer $A \subset X$. $P(X)$ chama-se o conjunto das partes de X . Ele nunca é vazio, pois tem-se pelo menos $\emptyset \in P(X)$ e $X \in P(X)$.

•Sejam A_1 e A_2 dois conjuntos não vazios. Define-se produto cartesiano entre os conjuntos A_1 e A_2 como $A_1 \times A_2 = \{(a_1, a_2) \mid a_i \in A_i, i=1, 2\}$ onde $(a_1, a_2) = (b_1, b_2) \Leftrightarrow a_i = b_i, i=1, 2$.

•Uma n -upla ordenada de elementos pertencentes a um conjunto A é representada por

$\langle a_1, a_2, \dots, a_n \rangle$.

• O conjunto de elementos que pertencem a um conjunto A e um conjunto B será denotado $A \cap B = \{x \mid x \in A \text{ e } x \in B\}$ e chamado intersecção de A e B.

• O conjunto de elementos que pertencem a um conjunto A ou a um conjunto B será denotado $A \cup B = \{x \mid x \in A \text{ ou } x \in B\}$ e chamado união de A e B.

• Seja A um conjunto. Tem-se uma partição de A, $PA(A) = \{A_1, \dots, A_n\}$ quando

$$A_i \cap A_j = \emptyset, \text{ para } i \neq j \text{ e } \bigcup_{i=1}^n A_i = A .$$

Relações

• Uma relação binária R entre dois conjuntos A e B, é definida como um subconjunto do produto cartesiano entre esses conjuntos.

• Uma relação binária R entre elementos de um conjunto C, pode ter algumas das propriedades abaixo. Sejam $x, y, z \in C$:

- 1.reflexiva: xRx ;
- 2.simetria: $xRy \Rightarrow yRx$;
- 3.anti-simétrica: $xRy \text{ e } yRx \Rightarrow x=y$;
- 4.transitiva: $xRy \text{ e } yRz \Rightarrow xRz$;

• Uma relação é dita relação de equivalência se satisfaz 1,2 e 4.

• Uma relação é dita relação de ordem parcial se satisfaz 1,3 e 4.

• Pares ordenados $(a,b) \in A \times B$ também podem ser representados por $\langle a, b \rangle$, sendo essa representação denominada uma 2-upla, ou simplesmente uma tupla.

Funções

• Sejam A e B dois conjuntos. Uma função do conjunto A no conjunto B é uma relação $f | \forall a \in A, \exists ! b \in B | (a, b) \in f$. A um elemento a denota-se $f(a)$. Portanto, os pares da função são do tipo $(a, f(a))$.

• Se $X \subset A$ e $f: A \rightarrow B$ denota-se por $f(X)$ ao conjunto $f(X) = \{f(x) | x \in X\} \subset B$ o qual é chamado imagem de X pela f . Denota-se $\text{Im } f$ ao conjunto $f(A)$ o qual chama-se Conjunto Imagem de f . Diz-se que a função f é sobrejetora se $\text{Im } f = B$.

• Diz-se que uma função $f: A \rightarrow B$ é injetiva se quaisquer que sejam $x, y \in A$, se $x \neq y$ então $f(x) \neq f(y)$.

• Se f é injetora e sobrejetora ao mesmo tempo, então f é bijetora.

• Se $f: A \rightarrow B$ e $X \subset A$ denota-se por $f|_X: X \rightarrow B$ a função cujo domínio é o conjunto X , cujo contra-domínio é o conjunto B e cuja regra é a mesma de f , isto é, $f|_X(x) = f(x)$ qualquer que seja $x \in X$. Chama-se $f|_X$ de restrição de f a X .

• Se $f: A \rightarrow B$ é uma função e $y \in B$, denota-se por $f^{-1}(y)$ ao conjunto $f^{-1}(y) = \{x \in A | f(x) = y\}$ o qual chama-se de imagem inversa de $y \in B$ pela f . Observe que se $y \in B$ então $f^{-1}(y) \subset A$, e se $y \notin \text{Im } f$ então $f^{-1}(y) = \emptyset$.

- Se $f:A \rightarrow B$ e $g:B \rightarrow C$ são duas funções, denota-se por $g \circ f:A \rightarrow C$ a função definida por $(g \circ f)(x) = g(f(x))$ qualquer que seja $x \in A$, a qual chama-se de função composta de g e f .

Dígrafos

- Um grafo G é um par (V, A) onde V é um conjunto cujos elementos são chamados vértices, e A é um subconjunto de $V \times V$, sendo seus elementos denominados arcos.

- Um grafo $H = (V', A')$ é um subgrafo de G ($H \subseteq G$) se $V' \subseteq V$ e $A' \subseteq A$.

- Um dígrafo DG é um par (V, A) onde V é um conjunto cujos elementos são chamados vértices, e A é um subconjunto de $V \times V$, sendo seus elementos denominados arcos direcionados.

- Se $(v_i, v_j) \in A$, diz-se que v_i é o vértice origem, e que v_j é o vértice destino do arco (v_i, v_j) .

- Um caminho orientado do dígrafo DG é uma n -upla de elementos de A , $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$ onde $\lambda_i = (a_{i-1}, a_i)$, e o destino de λ_i é a origem de λ_{i+1} . λ é um caminho orientado ligando a_0 a a_n ; a_0 é dito vértice inicial, e a_n vértice final do caminho. Note que havendo um caminho orientado ligando a_0 a a_n , não é garantida a existência de um caminho orientado ligando a_n a a_0 .

- Define-se caminho constante como sendo (v) , onde $v \in V$.

- Um caminho orientado $(\lambda_1, \lambda_2, \dots, \lambda_n)$ é dito simples se $\lambda_i \neq \lambda_j$, para $i \neq j$, ou

seja, passa-se apenas uma vez pelos arcos que o compõem.

•Um caminho orientado $(\lambda_1, \lambda_2, \dots, \lambda_n)$ é dito elementar se $a_i \neq a_j$ para $i \neq j$, ou seja, passa-se apenas uma vez pelos vértices que o compõem.

•Um vértice x é dito conectável a um outro vértice y , se existe um caminho orientado ligando x a y , isto é, $\exists(\lambda_1, \dots, \lambda_n)$ tal que $\lambda_1 = (x, v_1), \lambda_2 = (v_1, v_2), \dots, \lambda_{n-1} = (v_{n-2}, v_{n-1})$ e $\lambda_n = (v_{n-1}, y)$.

•Um ciclo é um caminho direcionado, cujo vértice inicial coincide com o vértice final.

•Uma árvore é um dígrafo que não possui ciclos e que satisfaz a seguinte condição: para todos vértices x e $y \in V$, ou existe um caminho conectando x a y , ou existe um caminho conectando y a x .