UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas de São Carlos
Departamento de Ciências de Computação e Estatística

SUPORTE A ACESSO MULTI-USUÁRIO
EM BASES DE DADOS ORIENTADAS A OBJETOS
ATRAVÉS DE ESQUEMAS SUPLEMENTARES

LUIZ CAMOLESI JÚNIOR

Orientador
Prof. Dr. Caetano Traina Júnior

Dissertação apresentada ao Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo, como parte dos requisitos para a obtenção do Título de Mestre na área de Ciências de Computação e Matemática Computacional.

SÃO CARLOS
Estado de São Paulo - Brasil
Novembro de 1992

*Dedico este trabalho aos meus pais,
Maria e Luiz, e aos meus avós, que
com esforço lutaram para a realização
de meus estudos.*

AGRADECIMENTOS

Ao Prof. Dr. Caetano Traina Jr., pela orientação neste trabalho, paciência, compreensão e amizade com que sempre me dispensou.

À Profa. Dra. Agma Juci Machado Traina, pela paciência e atenção sempre concedida.

Às secretárias da Pós-Graduação, Bete e Laura, e aos funcionários do Laboratório, Carmem e Luiz, pela simpatia, atenção e eficiência com que sempre me atenderam.

À bibliotecária Silvana, pelos serviços prestados durante a etapa de revisão bibliográfica e demais estudos.

Aos participantes do Grupo de Estudos de Bases de Dados - ICMSC-USP, que contribuíram para a realização deste trabalho, em especial os professores Pedro Luiz Pizzigatti Corrêa e João Eduardo Ferreira.

Aos amigos Rildo Ribeiro dos Santos e Mauro Biajiz, pelo incentivo e companheirismo em todas as horas e também ao Ricardo Defaveri Murer.

Aos amigos de "república": Hélio, Titov e especialmente Dirceu e Arthur pelo apoio técnico na área de engenharia.

Ao meu irmão Renan

À FAPESP, pelo apoio financeiro dado para o desenvolvimento deste trabalho.

SUMÁRIO

Lista de Figuras	vi
Resumo	viii
Abstract	ix
Capítulo 1. Introdução	1
1.1. Considerações Iniciais	1
1.2. Motivação	2
1.3. Objetivos	3
1.4. Organização da Dissertação	4
Capítulo 2. Conceituação Básica	5
2.1. Considerações Iniciais	5
2.2. Modelos de Dados (Classificação)	7
2.2.1. Baseados em Registros	9
2.2.2. Semânticos	10
2.2.3. Orientados a Objetos	11
2.3. Conceitos de Modelagem	13
2.3.1. Abstrações de Dados	13
2.3.2. Esquema de Dados	17
2.3.2.1. Fundamentos	18
2.3.2.2. Meta-Esquema de Dados	20
2.3.2.3. Evolução de Esquemas	21
2.3.3. Versões e Alternativas	22
2.4. Considerações Finais	24

Capítulo 3. Revisão Bibliográfica 25

3.1. Considerações Iniciais	25
3.2. Modelos Baseados em Registros	26
3.2.1. Rede	26
3.2.2. Hierarquico	28
3.2.3. Relacional	30
3.2.4. Análise dos Modelos Baseados em Registros	32
3.3. Modelos Semânticos	33
3.3.1. Entidade-Relacionamento (MER)	33
3.3.2. Relacional da Tasmânia (RM/T)	34
3.3.3. Modelo de Dados Semântico (SDM)	36
3.3.4. Análise dos Modelos Semânticos	37
3.4. Modelos Orientados a Objetos	39
3.4.1. Cactis	39
3.4.2. EXODUS	40
3.4.3. GemStone	42
3.4.4. Iris	43
3.4.5. O2	45
3.4.6. ORION	46
3.4.7. OZ & OZ+	49
3.4.8. PCTE & PCTE+	50
3.4.9. Análise dos Modelos Orientados a Objetos	52
3.5. Considerações Finais	54

Capítulo 4. O Modelo de Representação de Objetos 55

4.1. Considerações Iniciais	55
4.2. Conceituação	56
4.2.1. Diagramas de Representação	56
4.2.2. Fundamentos	57
4.2.3. Abstrações de Dados	58
4.2.4. Esquema e Meta-Esquema de Dados	63
4.2.5. Versões e Alternativas	65
4.3. Gerenciador de Objetos - GEO	65
4.4. Considerações Finais	67

Capítulo 5. Esquemas Suplementares 68

5.1. Considerações Iniciais	68
5.2. Definição	69
5.2.1. Inter-relação entre Hierarquias de Composição e Hierarquias de Generalização	69
5.2.2. Esquema Suplementar (conceito)	71
5.2.3. Particionamento	72
5.2.4. Modelagem e Representação Gráfica	74
5.3. Esquema de Dados	75
5.3.1. Conceito	75
5.3.2. Operações de Manipulação	76
5.3.2.1. Requisição	77
5.3.2.2. Liberação	77
5.4. Considerações Finais	78

Capítulo 6. Linguagem de Acesso do MRO* - LAMRO (Extensão) 79

6.1. Considerações Iniciais	79
6.2. Especificações	80
6.2.1. Classificação dos Comandos	80
6.2.2. Notação Utilizada	81
6.3. Definição da Linguagem	81
6.3.1. Inclusão	81
6.3.1.1. Tipo de Colônia	82
6.3.1.2. Tipo de Objeto	84
6.3.1.3. Tipo de Relacionamento Binário	85
6.3.1.4. Tipo de Relacionamento Triplo	88
6.3.1.5. Tipo de Atributo	91
6.3.1.6. Subtipo	94
6.3.1.7. Sumarização	94
6.3.2. Eliminação	94
6.3.2.1. Tipo de Colônia	95
6.3.2.2. Tipo de Objeto	95
6.3.2.3. Tipo de Relacionamento Binário	96
6.3.2.4. Tipo de Relacionamento Triplo	97

6.3.2.5. Tipo de Atributo	98
6.3.3. Substituição	99
6.3.3.1 Tipo de Colônia	99
6.3.4. Estabelecimento de Ambiente	100
6.3.5. Mostra	102
6.4. Considerações Finais	102
 Capítulo 7. Aspectos da Implementação	104
7.1. Considerações Iniciais	104
7.2. Esquema de Dados	104
7.2.1. GEO Mono-Esquema	105
7.2.2. GEO Multi-Esquema	105
7.3. Primitivas	106
7.3.1. Consulta	106
7.3.1.1. Classificação	107
7.3.1.2. Fluxo de Informação	111
7.3.2. Manipulação	114
7.3.3. Verificação	114
7.4. Considerações Finais	117
 Capítulo 8. Conclusões e Futuras Pesquisas	118
8.1. Considerações Iniciais	118
8.2. Decisões de Projeto	119
8.2.1. Alteração no Conceito de Controle de Acesso	119
8.2.2. Alternativas de Projeto	120
8.3. Sugestões de Futuras Pesquisas	121
8.3.1. Integração de Projetos	121
8.3.2. Codificação	122
8.3.3. Testes	123
8.3.4. Documentação	124
8.4. Considerações Finais	125

Referências Bibliográficas	127
Apêndice A: Exemplo de Modelagem de Dados	A1
Apêndice B: Exemplo de Utilização da LAMRO	B1
Apêndice C: Rotinas de Consulta ao Esquema de Dados (manual)	C1
Apêndice D: Rotinas de Manip. de Esquemas Suplementares (manual)	D1
Apêndice E: Estrutura Interna dos Esquemas de Dados	E1

LISTA DE FIGURAS

Figura 2.1. Sistema de Base de Dados	7
Figura 2.2. Níveis de Abstração	9
Figura 2.3. Utilização de Modelos Semânticos	11
Figura 2.4. Exemplo de Agregação/Desagregação	14
Figura 2.5. Exemplo de Composição/Decomposição	15
Figura 2.6. Exemplo de Classificação/Instanciação	15
Figura 2.7. Exemplo de Generalização/Especialização	16
Figura 2.8. Operações sobre Alternativas	23
Figura 3.1. Representação do Esquema (Modelo Rede)	27
Figura 3.2. Definição do Esquema (DDL/IDMS)	27
Figura 3.3a. Representação da Hierarquia Primária	28
Figura 3.3b. Representação da Hierarquia Secundária	29
Figura 3.4a. Definição da Hierarquia Primária	29
Figura 3.4b. Definição da Hierarquia Secundária	29
Figura 3.5. Representação de um Banco de Dados Relacional	31
Figura 3.6. Definição do Esquema (DDL/System R)	31
Figura 3.7. Comparação de Terminologias	32
Figura 3.8. Representação do Esquema (Modelo E-R)	34
Figura 3.9. Definição do Esquema (DDL/ -RM/T-)	35
Figura 3.10. Definição do Esquema (DDL/ -SDM-)	37
Figura 3.11. Características dos Modelos Semânticos	38
Figura 3.12. Características dos Modelos Orientados a Objetos	53
Figura 3.13. Evolução Histórica dos Modelos de Dados	54
Figura 4.1. Exemplo de um DRI	57
Figura 4.2. Hierarquia de Generalização (DRO)	59
Figura 4.3. Colônia Global (DRO)	60
Figura 4.4. Hierarquia de Composição (DHC)	61
Figura 4.5. Meta-Esquema de Dados do MRO*	64
Figura 4.6. Estrutura do GEO	67

Figura 5.1. Inter-relação entre Generalizações e Composições	70
Figura 5.2. Partição da Base Intencional	73
Figura 5.3. Representação de uma Esquemateca	74
Figura 5.4. Representação de um Esquema Suplementar	75
Figura 5.5. Representação de um Tipo Estrangeiro	75
Figura 6.1. Relacionamento "Constrita por"	82
Figura 6.2a. Relacionamento entre Tipos na mesma Colônia Esquema	83
Figura 6.2b. Relacionamento entre Tipos de diferentes Colônias Esquema	83
Figura 6.3. Relacionamento "Habita"	84
Figura 6.4. Relacionamento "Identificado por"	84
Figura 6.5. Relacionamentos "Origem é" e "Destino é"	86
Figura 6.6. Relacionamento "Composto por" (Rel. Binário)	86
Figura 6.7. Relacionamentos "Divisão 1", "Divisão 2" e "Divisão 3"	88
Figura 6.8. Relacionamento "Composto por" (Rel. Triplo)	89
Figura 6.9. Relacionamentos "Composto Determina" e "Composto Conjunto" ..	89
Figura 6.10. Relacionamento "Identifica"	91
Figura 6.11. Relacionamento "Atribui"	91
Figura 6.12. Relacionamento "Atributo de"	92
Figura 6.13. Acesso às Colônias Esquema através de Objetos da Colônia Global	101
Figura 7.1. Rotinas de consulta ao Esquema de Dados	110
Figura 7.2. Diagrama dos Fluxos de Informações na Versão Mono-Esquema .	112
Figura 7.3. Diagrama dos Fluxos de Informações na Versão Multi-Esquema .	113
Figura 8.1. Permissões de Manip. de Relac. entre Colônias Esquema	120
Figura 8.2. Sugestão de Padronização para as Folhas de Diagramas	124

SUORTE A ACESSO MULTI-USUÁRIO EM BASES DE DADOS ORIENTADAS A OBJETOS ATRAVÉS DE ESQUEMAS SUPLEMENTARES

Autor: **LUIZ CAMOLESI JÚNIOR**

Orientador: **Prof. Dr. CAETANO TRAINA JÚNIOR**

RESUMO

Este trabalho propõe um novo conceito na área de Bases de Dados, o qual estabelece a partição da Base Intencional de uma aplicação, de modo que as partes possam ser agrupadas de acordo com as necessidades de um usuário. Desta forma, é possível estabelecer várias configurações para o Esquema de Dados. Cada partição, denominada **Esquema Suplementar**, pode ser controlada individualmente, e colocada ou removida dinamicamente do Esquema de Dados proporcionando flexibilidade e um poderoso mecanismo de ocultamento ao SGBD. Este conceito é desenvolvido baseado no Modelo de Representação de Objeto (MRO^{*}).

Este trabalho estabelece as operações de consulta, manipulação e verificação dos Esquemas de Dados definidos a partir dos Esquemas Suplementares, além de estender a Linguagem de Definição de Dados do MRO^{*}, denominada LAMRO, para permitir a criação de Esquemas Suplementares, bem como a definição de seus componentes.

Foi implementado um módulo de gerenciamento de Esquemas Suplementares como um conjunto de rotinas que foram incorporadas ao núcleo do Gerenciador de Objetos (GEO) desenvolvido pelo Grupo de Bases de Dados do ICMSC - USP.

SUPPORT TO MULTIUSER ACCESS IN OBJECT ORIENTED DATABASES THROUGH SUPPLEMENTARY SCHEMA

Author: **LUIZ CAMOLESI JÚNIOR**
Adviser: **Prof. Dr. CAETANO TRAINA JÚNIOR**

ABSTRACT

This work proposes a new concept in the Database area which consists in the partitioning of a Database Schema, represented as an Intentional Database. Following this concept, it is permitted that each partition could be grouped in the users active Schema as needed, establishing many different configurations for the applications Schema. Each partition is called a **Supplementary Schema** and can be individually controlled, as far as dynamically combined with or removed from active Schema, and thus giving flexibility and a powerful resource for data hiding for DBMS. This concept is described in this work using Object Representation Model (MRO^{*}) as a basis for the conceptualization.

This work defines the operations for querying, manipulation and validation of Schemas constructed upon the Supplementary Schemas, and extends the query language of MRO (LAMRO) to support the creation and manipulation of Schemas based in the new concept.

It is also described the implementation of a module that adds to the MRO-based Object Manager the proposed functionality.

Capítulo 1

INTRODUÇÃO

1.1. - Considerações Iniciais

Entre as características intrínsecas dos seres humanos está a sua capacidade de criar, assimilar, modificar e adaptar-se a elementos e ambientes do mundo real. Neste ponto destaca-se a crescente busca de conhecimento através de passos progressivos objetivando a satisfação de cada indivíduo.

A área de computação não é exceção a esta regra, principalmente porque envolve sistemas que estão em constante interação com o Homem e que auxiliam ou possibilitam a execução de suas tarefas, além de proporcionar um poderoso ambiente para a pesquisa e disseminação de idéias.

Entre os sistemas de computação mais vulneráveis a mudanças estão os Gerenciadores de Bases de Dados, pois sua função principal é suportar a representação de elementos do mundo real que são passíveis de mutações estruturais, funcionais e comportamentais.

Para suportar as características evolutivas do mundo real que levam a constantes alterações, os sistemas que gerenciam as Bases de Dados devem naturalmente incorporar flexibilidade de estruturação das informações contidas em seu meio e assim, acompanhar e/ou adaptar-se às novas informações que surgem.

Incorporar estas características a um gerenciador de Base de Dados, para que satisfaça as necessidades de seus usuários e suas aplicações, determina a motivação deste trabalho.

1.2. - Motivação

O avanço tecnológico ocorrido nas últimas décadas, principalmente na área industrial, tem permitido e ditado as crescentes pesquisas em computação. Entre as recentes conquistas da computação estão os Sistemas de Base de Dados classificados de "Orientados a Objetos", que auxiliam projetos de engenharia ("Computer Aided Design - CAD/Computer Aided Manufacturing - CAM") [HARTZBAND_85], projetos de desenvolvimento de sistemas de computação ("Software Engineering") [HARTSON_89], na automação de escritórios [WEISER_89] e em sistema multimídia [WOELK_86].

Entre as características comuns a essas aplicações, está a necessidade de flexibilidade e liberdade de manipular informações, além de alterar as perspectivas de trabalho de seus usuários, motivada pela evolução das necessidades.

Em termos de Base de Dados, esta característica das aplicações se reflete em freqüentes mudanças na estrutura em que se apresentam as informações [SCIORE_89], sendo que tais alterações são realizadas por usuários com privilégio de acesso. Considerando-se a necessidade de flexibilidade e desvinculação entre as atividades dos usuários, deve ser permitido que a perspectiva de trabalho de cada um seja modificada, de modo que esta alteração não seja imposta aos demais usuários da Base de Dados, existindo a possibilidade de oculta-las desde que interesse à aplicação.

Para tornar esta motivação uma realidade, existe a necessidade de Sistemas de Gerenciamento de Base de Dados (SGBD) que incorporem esta característica por meio de primitivas de seu núcleo. A implementação deste tipo de gerenciador é atualmente pouco explorada por ser um estudo recente e também pelo fato de existirem poucos modelos de dados que conceitualmente permitem a incorporação deste grau de flexibilidade às alterações estruturais das Bases de Dados.

1.3. - Objetivos

Baseado no **Modelo de Representação de Objetos - MRO*** [TRAINA_86] [TRAINA_88] [BIAJIZ_92], está em desenvolvimento no ICMSC-USP São Carlos um Sistema de Gerenciamento de Base de Dados denominado **GEreenciador de Objetos - GEO** [TRAINA_91], com o objetivo de suportar aplicações de projetos de engenharia e de desenvolvimento de sistemas.

Considerado um modelo orientado a objetos, o MRO* permite representar Colônias, Objetos, Relacionamentos e Atributos, possibilitando incorporar complexas informações semânticas sobre entidades do mundo real.

Baseando-se no Modelo de Representação de Objetos este trabalho estabelece condições operacionais para que usuários da Base de Dados realizem suas tarefas sobre diversos enfoques de um projeto, independentemente uns dos outros, se assim desejarem. Isto permite que suas tarefas se realizem com maior flexibilidade, uma vez que os elementos que estabelecem as normas para a realização destas atividades (ou seja, o conjunto de elementos que formam o Esquema de Dados) podem ser particionados em subconjuntos com finalidades específicas e que poderão ser modificados por necessidades evolutivas, experimentais ou simplesmente corretivas. Desta forma, cada usuário da Base de Dados pode alterar sua formulação de trabalho independentemente dos outros membros que desenvolvem o projeto. Isso, é claro, se o gerente do projeto permitir esta alteração.

Para atingir estes objetivos, estudou-se neste trabalho os critérios para particionar o Esquema de uma Base de Dados, de modo que cada parte corresponda a algum aspecto específico do projeto em desenvolvimento.

Colocado em prática, este trabalho incorporou primitivas ao GEO para permitir o gerenciamento das várias partes do Esquemas que possam existir na Base de Dados e assim, estabelecer condições reais de utilização destes Esquemas em projetos suportados pela Base de Dados.

1.4. - Organização da Dissertação

O presente trabalho está organizado em 8 capítulos, sendo seus respectivos conteúdos descritos abaixo:

Capítulo 1: são apresentadas em linhas gerais as necessidades que motivaram este estudo, bem como os objetivos atingidos.

Capítulo 2: são classificados e fundamentados os Modelos de Bases de Dados e também apresentada a descrição genérica de conceitos particularmente importantes para esta linha de pesquisa.

Capítulo 3: são apresentados os principais Modelos de Base de Dados, dando especial atenção aos conceitos descritos no capítulo 2, de modo a colher subsídios para uma análise deste trabalho.

Capítulo 4: é apresentado o modelo de dados (MRO*) onde se desenvolveu este trabalho, estabelecendo-se alguns de seus conceitos, e também o sistema (GEO) onde implantou-se os resultados aqui obtidos.

Capítulo 5: é definido o conceito "Esquemas Suplementares" no qual se baseia este trabalho e também seu envolvimento na caracterização e definição de Esquemas de Dados.

Capítulo 6: são apresentados os comandos que compõem a linguagem de definição de dados para Esquemas Suplementares. Suas sintaxes são extensões da Linguagem de Acesso do MRO* (LAMRO).

Capítulo 7: são descritos alguns aspectos da implementação realizada neste trabalho.

Capítulo 8: são apresentadas várias linhas de pesquisa, estudo ou implementação além das contribuições deste trabalho.

Capítulo 2

CONCEITUAÇÃO BÁSICA

2.1. - Considerações Iniciais

Devido ao crescimento e desenvolvimento social, é cada vez maior a busca e manipulação de informações. Em meados dos anos 60, uma resposta a estas necessidades foi a utilização de sistemas de computação para o gerenciamento de dados em arquivos. A partir de sua implantação em empresas, os chamados **Sistemas de Gerenciamento de Arquivos** seguiram uma crescente sofisticação e otimização de suas funções de manipulação de informações, de modo a satisfazer os anseios de seus usuários.

A quantidade de informações envolvidas no cotidiano de uma empresa é geralmente formada por um grande volume de dados relacionados e concorrentemente utilizados em diferentes locais por vários usuários. A limitada tecnologia de computação existente no início da década de 60, e às vezes, até a falta de experiência administrativa, determinava a distribuição dos dados em diferentes dispositivos físicos de armazenamento. O estabelecimento de uma distribuição indiscriminada causava muitos problemas [BRADLEY_67] entre os quais podemos citar:

- 1.) **Redundância de Dados**:- ou seja, o armazenamento das mesmas informações várias vezes, o que levava a inconsistências das informações armazenadas, se não fosse realizado um controle das atualizações em dados repetidos.

2.) Dependência de Dados:- uma vez que os programas eram diretamente relacionados com a estrutura física dos dados manipulados, existia uma forte dependência entre os programas e seus dados e assim, possíveis modificações na estrutura física das informações poderiam afetar e determinar mudanças nos programas.

Associado aos problemas anteriores está o aumento dos gastos envolvidos no processo de informatização, uma vez que redundância de dados leva à redundância de programas ou ações, e dependências de dados levam a constantes mudanças de programas, o que torna necessário a alocação de funcionários e equipamentos para a realização destes trabalhos.

Buscando solucionar os problemas de redundância, foi estabelecido o conceito de unificação dos dados em um único dispositivo de armazenamento, dando início às denominadas **Bases de Dados** ou **Bancos de Dados**.

No final dos anos 60, foram desenvolvidos vários sistemas que detinham o controle de acesso aos arquivos de uma Base de Dados, tendo entre suas atribuições mais importantes, proteger as informações através de níveis de autorização de acesso e também isolar os programas de interesse do usuário (**programas de aplicação**) dos detalhes físicos da Base de Dados, constituindo-se, desta forma, em uma camada intermediária de controle e monitoramento de Bases de Dados, que foi denominada de **Sistema de Gerenciamento de Bases de Dados - SGBD**.

Ao conjunto formado pelo SGBD, Base de Dados, programas de aplicação e pelo ambiente de "hardware" e sistema operacional, foi dado o nome de **Sistema de Base de Dados - SBD** (Figura 2.1).

Cabe aos Administradores da Base de Dados (DBA¹) [TEOREY_82] estruturar o SBD em termos de "hardware" utilizado, programas aplicativos necessários e, principalmente, analisar os requisitos da empresa utilizando-se de uma ferramenta lógica de representação, com o objetivo de projetar estruturalmente a Base de Dados, buscando satisfazer as necessidades de rapidez e completeza de respostas às consultas desejadas pelos usuários.

Convergindo para os interesses deste trabalho, neste capítulo são genericamente definidas e classificadas as ferramentas lógicas de representação (Modelos de Dados), além de estabelecer alguns conceitos que as incorporam, dando-se especial atenção

¹ "Database Administrator"

Figura 2.1 - Sistema de Base de Dados

àqueles considerados relevantes. Para isto, na seção 2.2 estão discriminadas em três subseções as principais famílias de Modelos de Dados conceituais. Na seção 2.3 são descritos em detalhes alguns conceitos formulados nos Modelos de Dados e na seção 2.4 é finalizado o capítulo.

2.2. - Modelos de Dados (Classificação)

Em outubro de 1971, o Comitê sobre Computadores e Processamento de Informações (denominado X3) do ANSI² formou um grupo especial de estudos

² "American National Standards Institute"

chamado SPARC³, com a participação de representantes da comunidade de usuários, fabricantes de "hardware" e universidades, tendo como objetivo definir um vocabulário consistente na área de Base de Dados para uso universal.

Neste grupo, um **Modelo de Dados** foi genericamente definido como uma ferramenta lógica de representação e descrição abstrata de uma realidade (ou imagem de uma realidade). Para isso, um Modelo de Dados deve possuir uma linguagem de especificação que utilizando-se de um conjunto de símbolos, formaliza os elementos do mundo real e assim, descreve relacionamentos, semânticas e restrições das informações envolvidas. Deste modo, a ação de modelar (ou modelagem) corresponde a descrever, em um certo nível de abstração, os detalhes de uma realidade desejada.

Uma **Abstração** foi definida pelo SPARC como uma descrição de elementos do mundo real sobre um determinado aspecto de interesse, com o objetivo de detalhar apenas as informações relevantes a um usuário ou grupo de usuários.

A proposta ANSI/X3/SPARC apresenta 3 níveis de abstração [ANSI_78] (Figura 2.2): o Nível Físico descreve como os dados são fisicamente armazenados em estruturas de armazenamento; já o Nível Conceitual define logicamente quais dados são mantidos na Base de Dados; e o Nível de Visão descreve logicamente partes da Base de Dados, de modo a simplificar a interação (ou entendimento) do usuário, uma vez que cada usuário precisa apenas visualizar a parte da Base de Dados necessária para seu uso.

Tendo como premissa possibilitar modelagens a nível abstrato, surgiram Modelos aplicados a Bases de Dados (**Modelos de Bases de Dados**), denominados neste trabalho por Modelos de Dados, que além da representação de informações, propiciam ainda um meio de especificação dos dados através de uma **Linguagem de Definição de Dados** (DDL⁴) e uma forma de manipulação e consulta a Base de Dados, denominada de **Linguagem de Manipulação de Dados** (DML⁵).

Classificando-se os Modelos de Dados segundo os níveis de abstração, temos:

- 1.) Modelos de Dados Físicos:- ferramentas de representação usadas na descrição estrutural do armazenamento de uma Base de Dados. (Estes não serão descritos neste texto por estarem fora do escopo deste trabalho).

³ "Standards Planning and Requirements Committee"

⁴ "Data Definition Language"

⁵ "Data Manipulation Language"

Figura 2.2 - Níveis de Abstração (extraído de [DATE_88a])

2.) Modelos de Dados Conceituais:- ferramentas de representação que possibilitam uma descrição da Base de Dados de maneira desvinculada de sua estrutura física, proporcionando uma forma mais "natural" e amistosa de descrição. Alguns destes modelos possibilitam também a descrição de Visões.

Os Modelos de Dados Conceituais encontrados na literatura diferem por conceitos e aplicações [MCLEOD_91], podendo, todavia, serem agrupados em Modelos Baseados em Registros, Modelos Semânticos e Modelos Orientados a Objetos, sendo cada grupo pormenorizado a seguir.

2.2.1. - Modelos Baseados em Registros

Os primeiros Sistemas de Base de Dados desenvolvidos (até meados dos anos 70) seguiam uma "Orientação a Registros", tendo em vista aplicações ditas comerciais, tais como Sistemas de Contabilidade, Folhas de Pagamento e Controles de Estoque.

Nesta época, um conjunto limitado de aplicações com características semelhantes como, por exemplo, a manipulação de conjuntos de dados simples ou de pequena complexidade, eram as únicas aplicações comerciais que empregavam Bases de Dados e, conseqüentemente, utilizavam os Modelos Baseados em Registro, freqüentemente denominados CONVENCIONAIS.

O ponto em comum a ser notado entre os Modelos Convencionais é o uso de registros na representação de um aspecto do mundo real, sendo este um elemento fundamental para a realização de uma modelagem. Por esse motivo, estas modelagens são tradicionalmente representações estruturais dos dados, sendo de grande importância para a **padronização de informações** em Bases de Dados.

2.2.2. - Modelos Semânticos

Os Modelos Semânticos [HULL_87] surgiram como um estágio de representação intermediário [TEOREY_86] entre a análise dos requisitos e a representação em um modelo convencional (Figura 2.3).

Com o objetivo de incorporar elementos de representação para capturar maior conhecimento semântico do mundo real, os Modelos Semânticos introduziram alguns conceitos fundamentais [PECKHAM_88]:

- i.) Representação de Objetos não Estruturados e, assim, permitir o uso de tipos primitivos de dados como imagem e voz, na elaboração de tipos complexos para a definição de entidades;
- ii.) Representação explícita de Relacionamentos, Entidades e Atributos, ou seja, estabelecer uma clara distinção entre estes elementos, uma vez que são distintamente conceituados e simbolizados;
- iii.) Representação de Abstrações de Dados, permitindo simbolizar as freqüentes estruturas de Abstração importantes para uma modelagem;
- iv.) Representação Hierárquica e suporte à Herança, que proporcionam ao modelo uma maior estruturação e **flexibilidade** na modelagem, permitindo ao usuário representar e visualizar as informações em várias perspectivas.

Figura 2.3 - Utilização de Modelos Semânticos

Em contraponto às modelagens baseadas em registros, que são representações estruturais dos dados, os Modelos Semânticos incorporam, além de representações estruturais, os aspectos funcionais dos dados, sendo estas informações derivadas do mundo real e modeladas através dos vários tipos de entidades e relacionamentos em conjunto com representações hierárquicas, nas quais é possível estabelecer vários graus de dependência entre as entidades.

2.2.3. - Modelos Orientados a Objetos

O paradigma da "Orientação a Objetos" originou-se de linguagens de programação como SIMULA-67 e posteriormente SMALLTALK-80, que exploraram e refinaram alguns conceitos na busca de uma técnica eficiente de organização e programação de grandes sistemas [GARVEY_89] [JACKSON_91].

A abordagem Orientada a Objeto evoluiu recentemente merecendo muita

atenção por parte dos pesquisadores, de tal forma que atingiu várias áreas da computação [HARTZBAND_85] [LOCKEMANN_89] tais como: Sistemas de Base de Dados, Sistemas de Representação do Conhecimento e Sistemas Gráficos.

Na área de Base de Dados estão surgindo pesquisas de Modelos Orientados a Objetos [ATKINSON_89] que possibilitam o desenvolvimento de SGBD's para auxiliar no projeto de engenharia e de sistemas [HARTZBAND_85], na automação de escritório [WEISER_89] e Sistemas de Multimídia [WOELK_86].

Entre os conceitos que envolvem um Sistema de Gerenciamento de Base de Dados Orientado a Objetos - SGBDOO [BERTINO_91] estão aqueles derivados das linguagens de programação orientadas a objetos [TAKAHASHI_90], entre os quais estão:

OBJETO:- representação de um elemento do mundo real, sendo cada Objeto caracterizado por suas propriedades <Atributo , valor> e possuindo métodos associados que representam seu aspecto operacional. Um Objeto é diferenciado de outros Objetos por um identificador fornecido pelo sistema ou pelo usuário.

MÉTODO:- comando ou seqüência de instruções associadas a um identificador próprio (seletor). Um método aplica uma operação sobre Objetos associados e atribui a resposta à uma Mensagem. Os Métodos representam a interface do Objeto com o mundo externo, sendo que os detalhes sobre sua implementação são ocultados do usuário.

MENSAGEM:- elemento criado para solicitar a execução de um Método, ou então criado depois de um Método finalizado, sendo portanto, um elemento de comunicação entre Objetos.

CLASSE:- conjunto de Objetos agrupados por terem as mesmas características estruturais, funcionais e comportamentais.

Outros conceitos da "Orientação a Objetos" são extensões ou assemelham-se às representações usadas nos Modelos Semânticos, uma vez que sua capacidade de representação oferece suporte a alguns conceitos dos Modelos Orientados a Objetos, como é o caso de Objeto (correspondente à Entidade) e de:

RELACIONAMENTO:- associação explícita que estabelece graus de dependência entre os Objetos associados.

ATRIBUTO:- elemento de caracterização de Objetos e Relacionamentos através de valores. Os Atributos podem ser atualizados pelo usuário ou dinamicamente derivados de outros Atributos por meio de Métodos.

2.3. - Conceitos de Modelagem

Seja qual for a classificação e a finalidade a que se propõe, um Modelo de Dados deve suportar conceitos que estão intimamente relacionados com as exigências estruturais e/ou operacionais das aplicações a que se destina.

Levando-se em consideração os objetivos deste trabalho e visando alcançar seus propósitos, nas seções subsequentes são descritos conceitos importantes, embora alguns não possam ser suportados por certos modelos.

2.3.1. - Abstrações de Dados

Uma modelagem ("Database Design") necessita de uma estrutura natural de representação dos objetos para torná-la mais clara ao usuário. Smith [SMITH_77a] [SMITH_77b] descreve este conceito de representação como uma **Abstração de Dados**.

Uma Abstração de Dados significa a representação de objetos em diferentes níveis de detalhamento conceitual, dispostos através de **Hierarquias de Abstração**. Desta forma, informações relevantes de uma Base de Dados podem ser consideradas na visão de um usuário, enquanto outras são ignoradas.

Entre as principais Abstrações de Dados, sobre as quais atuam a operação de estruturação e sua respectiva inversa [HULL_87] [TAKAHASHI_90], podemos citar:

1) **Agregação:-** abstração que permite que associações entre Atributos distintos sejam representadas por um único Atributo de alto nível. Na Figura 2.4 é apresentado um exemplo onde os Atributos Rua, Número e Cidade são agrupados em um Atributo denominado Endereço.

Figura 2.4 - Exemplo de Agregação\Desagregação

2) **Sumarização**:- abstração que permite associar vários Objetos a um Objeto mais abrangente que possivelmente represente alguma estatística, de acordo com algum critério pré-estabelecido. Por exemplo, um certo Objeto Média de Validade poderá representar o resultado de um cálculo que obtém a média em dias em que Objetos Produtos terão validade garantida.

3) **Composição**:- abstração que permite que associações entre Objetos distintos sejam representadas por um único Objeto de alto nível, através de relacionamentos "IS-PART-OF". Na Figura 2.5 é apresentado um exemplo onde os Objetos do Tipo Substância são compostos por Objetos do Tipo Produto (químico e diluente). Nas composições entre Objetos, o resultado obtido será uma hierarquia em que um Objeto de alto nível recebe a denominação de **Objeto Complexo**, por ser constituído de outros Objetos que, por sua vez, podem ser complexos ou não.

4) **Classificação**:- abstração que engloba objetos com características comuns, relacionando-os com um Objeto (Tipo) existente, utilizando-se para isto de relacionamentos "IS-INSTANCE-OF". No exemplo na Figura 2.6, os Objetos ICI e W. MARTINS possuem os mesmos atributos (Nome, Rua, Número, Cidade)

Figura 2.5 - Exemplo de Composição/Decomposição **Figura 2.6 - Exemplo de Classificação/Instanciação**

com valores diferentes, porém esta semelhança estrutural possibilita associá-los com um Tipo Fornecedor que sintetiza suas semelhanças.

5) **Generalização**:- abstração que associa Tipos similares em um único Tipo de nível mais alto através de relacionamentos "IS-A" [BRACHMAN_83]. A Figura 2.7 apresenta um exemplo onde os Tipos Indústria e Fornecedor podem ser representados pelo Tipo Empresa. Portanto, nesse exemplo de modelagem generalizaram-se todas as Indústrias e todos os Fornecedores classificando-os como Empresas e, conseqüentemente, uma Empresa poderá ser especializada como uma Indústria ou um Fornecedor.

Figura 2.7 - Exemplo de Generalização/Especialização

No exemplo anterior percebe-se que o Atributo Nome está ligado ao Tipo Empresa, porque esse Atributo é comum a todas as Empresas. Cada Fornecedor e cada Indústria é uma Empresa e, portanto, possui o Atributo Nome, através do mecanismo de **Herança de Atributos** [BORGIDA_88].

Sucessivas operações de Generalização (ou Especialização) descrevem uma hierarquia que pode:

- i.) não ser uma árvore [SCIORE_89], uma vez que um Tipo pode ter mais que um Tipo como pai, o que pode estabelecer **Múltiplas Heranças** [HARTZBAND_85], quando então, existe a possibilidade de conflito entre informações herdadas pelos Tipos;

ii.) ser constituída por nós não mutuamente exclusivos [SMITH_77a], ou seja, um objeto pode ou não estar associado a Tipos distintos. No caso da Figura 2.7, um certo Fornecedor poderá também ser uma Indústria.

A representação hierárquica das informações possibilita que vários usuários acessem concorrentemente diferentes níveis da hierarquia, formando a Visão do Usuário. Outra vantagem da representação é a estabilidade que proporciona ao modelo, uma vez que mudanças em níveis inferiores da hierarquia não afetam os níveis superiores, devido à modularidade proporcionada pelas conexões semânticas (relacionamentos IS_A) associadas aos Objetos [HULL_87].

O uso de abstrações na representação dos dados proporciona ao Modelo de Dados as características que [SMITH_77b] relaciona como sendo desejáveis:

- i.) Uma representação concisa e compreensível dos dados;
- ii.) Um acesso eficiente e de alto nível as informações;
- iii.) Uma atualização simples e consistente dos dados.

A representação concisa e compreensível é obtida à medida que as abstrações de Generalização e Agregação estão dispostas hierarquicamente, proporcionando uma representação semântica mais natural e uma visão mais realista do mundo. Os acessos eficientes são derivados da utilização de operadores para os Tipos e as atualizações consistentes são conseguidas através da generalização de Atributos em Tipos, em substituição às repetições de valores de Atributos em objetos. Estas características possibilitam uma visualização e gerenciamento da Base de Dados em partes diferenciadas pelos níveis da hierarquia disponíveis a cada usuário.

Sendo conceitos muito explorados em Modelos Semânticos, as Abstrações de Dados chegam a ser essenciais para as aplicações a que se destinam os Modelos Orientados a Objetos.

2.3.2. - Esquema de Dados

O produto final de uma representação abstrata (física, conceitual ou visão) baseada em qualquer Modelo de Dados, é chamado de **Esquema de Dados** e contém os elementos da modelagem de um aspecto do mundo real segundo o Modelo de

Dados envolvido.

A criação de um Esquema pelo Administrador da Base de Dados (Figura 2.1) é possível através de um conjunto de instruções em DDL que devem ser compiladas, para que sua descrição possa ser validada e então utilizada na criação do Esquema.

2.3.2.1. - Fundamentos

Não existe um padrão de definição para o que deve ser um Esquema de Dados, e muito menos onde deve ser armazenado (na própria Base de Dados ou em arquivo separado) e quais são as suas reais atribuições. Cada implementação de um Sistema de Base de Dados estabelece, de acordo com suas necessidades, sua própria definição e uso do Esquema de Dados de acordo com o Modelo de Dados seguido.

O Esquema de Dados em Modelos Baseados em Registros é conhecido também como **Dicionário de Dados** [DATE_88a] e pode desempenhar uma função mais abrangente, pois em seu conteúdo poderão existir referências aos programas de aplicação e até mesmo uma descrição do ambiente em que a Base está inserida.

Em modelos para Bases de Dados Orientados a Objetos, o Esquema de Dados é apenas a nível conceitual e seu objetivo é proporcionar uma representação da compreensão do mundo real (ou imagem) e, assim, reconhecer e descrever formalmente os conhecimentos em termos **Estrutural, Funcional e Comportamental**.

Dispostos nestes três aspectos, abaixo estão caracterizados genericamente os Esquemas de Dados:

- Estrutural

Desde que os elementos do mundo real (ou imagem) são naturalmente agrupados em conjuntos semelhantes, por possuírem características estruturais em comum, um Esquema de Dados de um Modelo Orientado a Objetos ou Semântico deve aceitar esta natureza discriminatória e estruturar os dados em termos de grupos ou **Tipos**.

Outras formas de particionamento que dependam da classificação de Tipos podem ser definidas pelo Esquema. A mais conhecida e usada em muitos modelos é chamada **Classe de Objetos**, que agrupa os Objetos da Base de Dados que tenham as mesmas características estruturais (mesmo Tipo).

Na Base de Dados, todos os elementos ou dados que podem possuir semelhanças devem ser ao menos classificados em Tipos. Assim, temos no Esquema de Dados a descrição dos elementos conceituais de uma aplicação, sendo composta pela:

- descrição de todos os **Tipos de Objetos** e seus **Tipos de Identificadores**;
- descrição de todos os **Tipos de Relacionamentos**, além dos Tipos de Objeto que os utilizam em associações;
- descrição de todos os **Tipos de Atributos**, além dos Tipos de Objetos e Tipos de Relacionamentos que são utilizados para descreve-los;
- descrição dos Tipos de Abstração suportados pelo sistema e os respectivos Relacionamentos entre os quais:
 - **IS-A**: estrutura as hierarquias de generalização;
 - **IS-PART-OF**: estrutura as hierarquias de composição;
 - **IS-INSTANCE-OF**: estrutura a classificação que define o conteúdo interno das Classes.

- Funcional

As necessidades reais de uma aplicação estão também em termos operacionais, de modo a representar e descrever operações sobre cada Objeto em particular. Assim, o Esquema de Dados deve proporcionar a representação de Métodos (ou **Regras**) em Objetos.

O particionamento de Objetos em Classes pode ter um carácter operacional, isso porque no mundo real (imagem) os elementos estão também distintos em termos de contextos funcionais, onde conjuntos de elementos desempenham funções específicas e semelhantes.

As estruturas funcionais geralmente representadas no Esquema de Dados dos Modelos Orientados a Objetos e também nos Modelos Semânticos são:

- **Hierarquia de Generalização**:- com sua característica de particionamento, leva à descrição estrutural de **Subtipos** (ou Subclasses);
- **Hierarquia de Composição**:- com a estruturação de conjuntos de Objetos distintos em Objetos de alto nível.

- Comportamental

Para uma aplicação é importante estabelecer as ações que serão tomadas por um Objeto quando este é manipulado em uma operação, ou seja, qual a reação ou o comportamento de um Objeto perante uma ação. É, portanto, necessário que o Esquema de Dados tenha meios de garantir a correta manipulação de Objetos, tanto a nível de primitivas do SGBDOO quanto a nível de Regras, buscando satisfazer as restrições de integridade.

2.3.2.2. - Meta-Esquema de Dados

A complexidade de todas as informações descritas em um Esquema de Dados tanto para os aspectos estrutural, funcional e comportamental, muitas vezes exige que o Esquema de Dados seja armazenado (total ou parcialmente) em uma Base de Dados denominada de **Meta-Base de Dados**. Deste modo, o SGBD poderá atuar consistentemente sobre os dados da Base, de uma maneira controlada e eficiente.

Um ponto importante a ser notado é o fato de que um Modelo de Dados pode ser descrito pelo seu aspecto Estrutural, Funcional e Comportamental. Estes aspectos, quando representados, recebem a denominação de **Meta-Esquema de Dados**. Genericamente o conteúdo de um Meta-Esquema é:

- Estrutural

Um modelo é estruturalmente a definição de seus próprios conceitos, sendo estabelecidos também os possíveis envolvimento entre eles, bem como a forma em que este envolvimento ocorre.

- Funcional

Funcionalmente um modelo é descrito por suas primitivas que manipulam as informações da Base de Dados, como por exemplo CRIAR, REMOVER e CONSULTAR.

- Comportamental

O aspecto Comportamental de um modelo é descrito pelas **Restrições de Integridade**, ou seja, normas estabelecidas para se manter a consistência dos dados.

Se o aspecto estrutural do Meta-Eschema de Dados puder ser estruturado em memória, poderá auxiliar o SGBD na definição e manipulação da Meta-Base.

2.3.2.3. - Evolução de Esquemas

Denominada de Evolução de Esquema de Dados [BERTINO_91], a atividade de evoluir as informações da Base de Dados diante de mudanças no Esquema é classificada em 3 tipos básicos [BJORNERSTEDT_89]:

- 1.) **Conversão** (ou "Immediate-update"):- quando um Esquema é trocado, todas as informações da Base de Dados são imediatamente alteradas de forma coerente com o novo Esquema. Esta operação é realizada atomicamente e portanto sua desvantagem está no tempo consumido para a concretização de todas as atualizações de uma única vez.
- 2.) **Conversão Incremental** ("Incremental Conversion"):- as alterações das informações da Base de Dados após a mudança do Esquema são realizadas gradativamente em cada dado, à medida que estiverem sendo manipulados por alguma operação. A velocidade de execução das operações são desta forma comprometidas pelo tempo gasto na conversão.
- 3.) **Atualização Adiada** (ou "Screening"):- neste caso, nenhuma alteração física é realizada na Base de Dados, ou então esta é adiada indefinidamente. O novo Esquema de Dados definido, corresponde apenas a uma Visão Conceitual de toda Base de Dados. Assim, as operações sobre os dados armazenados, serão "filtradas" para que o usuário tenha a ilusão de estar manipulando dados de um Esquema real. Esta opção compromete a velocidade de execução de qualquer operação, uma vez que as informações apresentadas aos usuários devem ser modificadas de acordo com a Visão do Esquema de Dados .

2.3.3. - Versões e Alternativas

Aplicações no desenvolvimento de projetos [KELTER_87] [KATZ_86] ou em Base de Dados Multimídia [WOELK_86] por exemplo, se caracterizam pelo fato de possuírem elementos que podem ser projetados ou reformulados em termos de seus componentes e que por sua natureza geralmente evolutiva (explorando características de "performance", tecnologia ou custo), necessitam de Sistemas de Bases de Dados que suportem a evolução das informações armazenadas. A evolução de elementos de um Base de Dados é mantido através de Versões e Alternativas.

Conceitualmente, Versões são consideradas cópias de um mesmo elemento, sendo cada um particularmente diferente em seu conteúdo ou estrutura devido às atualizações ou modificações realizadas. Segundo Katz [KATZ_86], cada Versão pode ser organizada em 3 planos distintos:

- 1.) **Plano de Versões:-** corresponde a um histórico das Versões com o objetivo de documentar as etapas e fases envolvidas na concepção de um elemento e assim perceber a seqüência de criação das Versões e a origem de cada uma;
- 2.) **Plano de Configuração:-** permite estabelecer a configuração (ou Versão) de um elemento composto, através da definição das Versões de seus componentes;
- 3.) **Plano de Equivalência:-** permite que várias representações de um mesmo elemento sejam colocadas em Versões, assim a forma de apresentação de um objeto para um usuário dependerá da Versão consultada.

Uma forma de referenciar um elemento armazenado, sem designar alguma Versão em particular, pode ser feita através do chamado **Objeto Genérico**, o qual sintetiza a estrutura comum a todas as Versões do objetos representado. Este conceito estabelece uma forma de abstração, se o SGBD suportar Objetos Genéricos permitindo que a eles sejam atribuído uma Versão corrente. Uma forma mais específica de referência genérica é possível através de um **Objeto Equivalente**, que representa o conjunto de Versões de um mesmo Plano de Equivalência.

Uma **Alternativa** [KATZ_84] corresponde a um estado anterior de uma Versão, pois caso um usuário deseje uma Versão de uma certo elemento para alteração, ele deverá primeiramente criar uma Alternativa do elemento, ou seja, uma cópia total ou

parcial da estrutura e conteúdo do elemento desejado e após atualizações e modificações nesta Alternativa, esta passará a ser uma Versão, se for considerada consistente pelo SGBD (Figura 2.8). Em suma, Versões não podem ser alteradas uma vez que foram colocadas em uso ("Read Only"). Caso haja mudanças a serem feitas, Alternativas deverão ser criadas com esta finalidade ("Read and Write"), as quais posteriormente torna-se-ão Versões.

Figura 2.8 - Operações sobre Alternativas

Entre os elementos de uma Base de Dados que são passíveis de Versão estão os Objetos Complexos [AHMED_91], uma vez que possuem vários componentes internos (Objetos), que poderão ser modificados ou atualizados dependendo da aplicação.

Versões podem ser usadas para 3 propósitos [BJORNERSTEDT_89]:

- i.) Capturar um histórico evolutivo das mudanças ocorridas com os elementos armazenados na Base de Dados, mantendo-se os planos de versões, configuração e equivalência de cada elemento para posterior consulta;

- ii.) Controlar a Concorrência [SKARRA_89], através da duplicação de elementos ("Check out") para que as cópias sejam acessadas individualmente e posteriormente unificadas com o final das operações ("Check in");
- iii.) Evitar a interrupção das atividades na Base de Dados, por causa da reorganização [SOCKUT_79] das informações depois de mudanças no Esquema de Dados.

Alguns sistemas ainda suportam o conceito de Versão do Esquema de Dados, uma vez que novas necessidades de representação surgem da evolução natural das aplicações, buscando sempre satisfazer o usuário em suas necessidades. A dificuldade neste tipo de atividade está exatamente em detectar e minimizar os efeitos da troca do Esquema de Dados nos dados armazenados e na Visão de cada usuário.

Em termos de armazenamento físico em uma Base de Dados, as Versões poderão ser representadas integralmente ou apenas pelos elementos que as diferem de sua Versão original. Este mecanismo, conhecido como "Delta", possibilita uma economia significativa de memória nas Bases de Dados cuja prática de se criar Versões é comum.

Além de Katz, outros pesquisadores trabalham na busca de métodos e conceitos de versão significativos para suas aplicações, sendo alguns enumerados em [KATZ_90] e [PRICE_89].

2.4. - Considerações Finais

Neste capítulo foram abordados genericamente os Modelos de Dados, sendo classificados em 3 grupos: Baseados em Registros, Semânticos e Orientados a Objetos.

Outro enfoque deste capítulo foi a descrição genérica de alguns conceitos utilizados para a modelagem de um mundo real, onde destaca-se o conceito de Esquema de Dados. Os tópicos abordados neste capítulo foram apreciados devido as suas relevâncias para a fundamentação deste trabalho.

Capítulo 3

REVISÃO BIBLIOGRÁFICA

3.1. - Considerações Iniciais

Na área de Base de Dados surgiram muitos Modelos de Dados, cada um proporcionando as características desejadas por suas aplicações. Para isso, na definição de qualquer modelo estão seus conceitos de modelagem fundamentais, descritos genericamente no capítulo anterior. Porém o tratamento dado individualmente em cada modelo difere no uso e geralmente na implementação, de acordo com as necessidades ou mesmo por divergências conceituais.

Para elucidarmos o tratamento dado por alguns modelos aos conceitos de modelagem, é realizado neste capítulo um estudo de alguns modelos, procurando-se formar um conjunto de informações e encontrar subsídios para uma análise deste trabalho. A atenção ao analisar as publicações sobre os Modelos está basicamente voltada para os conceitos de Esquemas de Dados e Versões, procurando-se, além disso, levantar informações sobre os recursos (conceitos) oferecidos em cada Modelo, que permitam ao final de cada seção uma análise crítica de comparação entre eles.

Esta revisão não visa apresentar todos os Modelos existentes, se isto for possível, mas apenas alguns dos mais importantes e que encontram-se disponíveis em publicações recentes.

3.2. - Modelos Baseados em Registros

Hoje em dia, os Modelos Baseados em Registro ainda atuam no mercado de Bases de Dados, devido a sua larga utilização comercial e também pela simplicidade e familiaridade com que seus SGBD's são apresentados aos usuários. Entre os Modelos mais conhecidos e amplamente utilizados temos: Modelo Rede; Modelo Hierárquico e o Modelo Relacional. Uma descrição sucinta de seus conceitos e implementações encontra-se nas próximas subseções.

3.2.1. - Modelo Rede

No final dos anos 60 surgiram diversos Sistemas de Base de Dados comerciais baseados no Modelo Rede, sendo o primeiro modelo a ser largamente utilizado, juntamente como o Modelo Hierárquico. Na época, a crescente procura por Bases de Dados comerciais levou o grupo de estudos CODASYL¹ a criar em 1965, um comitê especial denominado DBTG² para especificação de um SGBD baseado no Modelo Rede [CODASYL_71].

Utilizando-se da notação gráfica proposta por Backman em 1969 para a representação de modelagens, o Modelo Rede representa elementos do mundo real através de Coleções de Registros (**arquivos**), sendo cada registro formado por um conjunto de **Campos**. As associações entre arquivos envolvem **Ligações** ("Sets") de cardinalidade 1:N, isto é, cada registro chamado de **Mestre** ("Owner") pode ter 1 ou mais registros **Membro** ("Member").

Um exemplo de modelagem em Rede, utilizando o diagrama de Backman estendido, é apresentado na Figura 3.1, onde estão representados os arquivos INDÚSTRIA, FORNECEDOR e PRODUTO em retângulos, seus respectivos campos (os campos que são chaves de identificação estão grifados) e as ligações são direcionadas por setas para aos arquivos Membros. A definição do Esquema do exemplo é apresentada na Figura 3.2., através da Linguagem de Definição de Dados

¹ "CONference on Data SYstems Languages"

² "Data Base Task Group"

Figura 3.1 - Representação do Esquema (Modelo Rede)

do sistema IDMS.

O Modelo Rede possui um grande número de Sistemas de Gerenciamento de Bases de Dados implementados e comercialmente disponíveis entre os quais o DMS (da UNISYS), o IMAGE (da HP) e o IDMS (da Cullinet Software).

Figura 3.2 - Definição do Esquema (DDL/IDMS)

3.2.2. - Modelo Hierárquico

Similar ao Modelo Rede e também usando a representação de Backman, o Modelo Hierárquico consiste em Coleções de Registros (**arquivos**), que conectam-se por **Ligações** ("Sets") de cardinalidade 1:N, sendo o registro origem de cada Ligação chamado de **Registro PAI** e o registro ou registros destino são chamados de **Registro FILHO**.

Uma restrição deste modelo está na estrutura em árvore, em que deve ser apresentada a modelagem. Isto significa que cada registro poderá ser PAI de muitos registros FILHO, porém um registro FILHO deverá ter somente um registro PAI.

O modelo está intimamente ligado a uma percepção hierárquica do mundo real, porém esta não é uma limitação, uma vez que o Modelo Hierárquico oferece a possibilidade de representações simultâneas para uma mesma realidade, que denominam-se **Hierarquia Primária** e **Hierarquias Secundárias**.

A Hierarquia Primária (Figura 3.3a) determina a estrutura lógica, na qual se estabelece a forma de acesso mais usada na Base de Dados. As Hierarquias Secundárias constituem-se em um conjunto de hierarquias independentes (Figura 3.3b) que determinam caminhos de acesso alternativos.

Figura 3.3a - Representação da Hierarquia Primária (Modelo Hier.)

Figura 3.3b - Represent. da Hierarquia Secundária (Modelo Hier.)

Entre as implementações de Sistemas de Gerenciamento de Base de Dados hierárquicos encontrados na literatura estão o IMS (IBM) [DATE_88a] e o System 2000 (MRI Systems Corporation).

A representação das hierarquias primária e secundária apresentadas no exemplo são especificadas pelas Linguagem de Definição de Dados do Sistema IMS como mostrados nas Figuras 3.4a e 3.4b respectivamente.

```

1 DBD NAME Fiscalização
2 SEGM NAME = Indústria, BYTE = 30
3 FIELD NAME = Nome, BYTE = 15, START = 1
4 FIELD NAME = (CGC,SEQ), BYTE = 15, START = 16

5 SEGM NAME = Produto, PARENT = Indústria, BYTE = 25
6 FIELD NAME = Nome, BYTE = 15, START = 1
7 FIELD NAME = (Código,SEQ), BYTE = 6, START = 16
8 FIELD NAME = Validade, BYTE = 4, START = 22

9 SEGM NAME = Fornecedor, PARENTE = Indústria, BYTE = 69
10 FIELD NAME (Nome,SEQ), BYTE = 20, START = 1
11 FIELD NAME Rua, BYTE = 30, START = 21
12 FIELD NAME Número, BYTE = 4, START = 52
13 FIELD NAME Cidade, BYTE = 15, START = 57
  
```

Figura 3.4a - Definição da Hierarquia Primária (DDL/IMS)

```

1 PCB TYPE = DB, DBNAME = Fornecimento
2 SENSEG NAME = Fornecedor, PROCOPT = I
3 SENSEG NAME = Produto, PARENT = Fornecedor, PROCOPT=I
  
```

Figura 3.4b - Definição da Hierarquia Secundária (DDL/IMS)

3.2.3. - Modelo Relacional

O Modelo Relacional foi proposto por E. F. Codd [CODD_70] com o objetivo de construir SGBD's baseados na álgebra de relações entre conjuntos.

Formalmente, o Modelo Relacional é composto de relações. Uma **Relação** é um conjunto de **n-Tuplas** (registros) formadas por **Atributos** cujos valores estão em algum conjunto domínio especificado. Em toda relação existe um conjunto de Atributos chamados de **Chave de Identificação**, que permitem reconhecer univocamente cada Tupla existente.

Neste modelo, a organização conceitual de uma Base de Dados, de modo a definir as Relações e Atributos dá-se primeiramente, através de uma análise dos requisitos do sistema e é posteriormente finalizado com um processo de **Normalização**.

A teoria de Normalização está montada em torno de um conjunto de regras denominadas **Formas Normais** (1NF, 2NF, 3NF, BCNF, 4NF, ..) e em resumo consiste de passos gradativos para estabelecer Relações que, ao final, apresentam uma estrutura consistente para o armazenamento de informações.

A representação conceitual é geralmente feita através da especificação das Relações, segundo uma notação empregada na teoria matemática de álgebra de relações. As Relações (não normalizadas) da Figura 3.5 seriam conceitualmente especificadas da seguinte forma:

INDÚSTRIA (Nome , CGC , Produto , Fornecedor)
FORNECEDOR (Nome, Rua, Número, Cidade, Produto)
PRODUTO (Nome, Código, Validade)

A linguagem de consulta e manipulação de dados (DML) está baseada nas operações elementares de seleção, projeção, produto cartesiano, união e diferença, que obedecendo sempre as restrições de integridade, mantêm as relações consistentes.

O Modelo Relacional foi estudado durante muito tempo, sendo que as primeiras implementações eram protótipos, os quais tinham o propósito de estudar a viabilidade do modelo. Nesse sentido, foram bastante divulgadas as implementações do System R [ASTRAHAN_76] desenvolvido pela IBM e do projeto INGRES ("INteractive GRaphics and REtrieval System") [STONEBRAKER_76] desenvolvido pela Univ. da California em Berkeley. Atualmente existem diversas implementações comerciais de SGBD's Relacionais tais como DB2 da IBM e o ORACLE.

INDÚSTRIA				
NOME ----	CGC ---	PRODUTO -----	FORNECEDOR -----	
J&J	123423444	GAS332	ICI	
Rhodia	223423434	LIQ009	W. Martins	
Gessi Lever	434242345	SOL210	Gerdau	

FORNECEDOR				
NOME ----	RUA	NÚMERO	CIDADE	PRODUTO -----
ICI	Oito	1031	Campinas	GAS332
Gerdau	da Cruz	906	São Paulo	SOL210
W. Martins	XV de Novembro	412	São Carlos	LIQ009

PRODUTO		
NOME	CÓDIGO -----	VALIDADE (em dias)
Etano	GAS332	280
Sulfito	LIQ009	150
Pasta X1	SOL210	100

Figura 3.5 - Representação de um Banco de Dados Relacional

Para efeito de visualização dos conceitos citados anteriormente, a Figura 3.6 apresenta a definição do Esquema de Dados para o System R.

CREATE TABLE Indústria (	Nome (CHAR(15)), CGC (CHAR(15),NONULL), Produto (CHAR(6)), Fornecedor (CHAR(20)))
CREATE TABLE Fornecedor (	Nome (CHAR(20),NONULL), Rua (CHAR(30)), Número (INTEGER), Cidade (CHAR(15)), Produto (CHAR(6)))
CREATE TABLE Produto (	Nome (CHAR(15)), Código (CHAR(6),NONULL), Validade (INTEGER))

Figura 3.6 - Definição do Esquema (DDL/System R)

3.2.4. - Análise dos Modelos Baseados em Registros

Segundo Elmasri e Navathe [ELMASRI_89], a principal diferença entre a representação no Modelo Relacional e os Modelos Rede e Hierárquico está nos Relacionamentos. No Modelo Relacional, as conexões entre Relações são representadas pela inclusão de dois Atributos de mesmo domínio, um em cada relação: as Tuplas que individualmente possuírem o mesmo valor para aquele Atributo estarão logicamente relacionadas. Para os Modelos Rede e Hierárquico, as conexões 1:N entre Arquivos são explicitamente representadas por Ligação ("Set") e desta forma, os registros são fisicamente conectados a registros relacionados.

Uma comparação entre o Modelo Rede e o Modelo Hierárquico permite vislumbrar limitações no Modelo Hierárquico. Isto porque, na necessidade de Relacionamentos N:M entre registros, a modelagem hierárquica irá necessitar de mais de uma hierarquia (ou seja, Hierarquias Secundárias) que implicam na repetição de registros ou ponteiros.

Na Figura 3.7 são defrontadas as terminologias utilizadas no três modelos, onde percebe-se uma certa homogeneidade nos conceitos.

MODELO REDE	MODELO HIERARQUICO	MODELO RELACIONAL
Arquivo	Arquivo	Relação
Registro	Registro	Tupla
Campo	Campo	Atributo
Tipo de Dado	Tipo de Dado	Domínio
Ligação 1:N (Owner-Member)	Ligação 1:N (PAI-FILHO)	Atrib. em comum (Chave Estrangeira)

Figura 3.7 - Comparação de Terminologias

3.3. - Modelos Semânticos

Nesta seção são considerados três modelos bem conhecidos da literatura: o Modelo Entidade-Relacionamento (MER); o Modelo Relacional Extendido (RM/T); e o Modelo Semântico de Dados (SDM). Existem ainda muitos outros modelos igualmente expressivos, oferecendo uma grande diversidade de modelagens e filosofias, onde se pode citar o TAXIS e o Modelo de Dados Funcional (FDM).

3.3.1. - Modelo Entidade-Relacionamento (MER)

Proposto por Peter Chen [CHEN_76] com o objetivo de incorporar informações importantes sobre uma realidade, o Modelo E-R parte do pressuposto de que o mundo real pode ser descrito através de **Entidades**, as quais são caracterizadas através de **Atributos e Relacionamentos**.

Uma Entidade é um elemento distinguível de outras Entidades. As Entidades que possuem os mesmos Atributos são agrupadas em um **Conjunto de Entidades**.

Relacionamento é uma associação entre Entidades, possuindo Atributos que o caracterizam. Os Relacionamentos com os mesmos Atributos agrupam-se em **Conjuntos de Relacionamentos**. Cada Relacionamento é definido explicitamente com uma **Cardinalidade** (1:1, 1:N ou N:M quando o Relacionamento é binário e N:M:P quando for ternário), ou seja, as quantidades de Entidades que podem estar associadas em cada Relacionamento.

As propriedades de uma Entidade são expressas em termos de pares **<Atributo,valor>** e descrevem as características de uma Entidade ou de um Relacionamento, sendo que as Entidades devem possuir Atributos que possam ser usados como **Chave de Identificação**.

A representação conceitual de uma Base de Dados no Modelo E-R emprega o Diagrama Entidade-Relacionamento (DER) na definição do Esquema de Dados, onde é possível descrever os Conjuntos de Entidades (em retângulos), os Conjuntos de Relacionamentos (em losangos) e os Atributos associados (em elipses), como mostra a Figura 3.8.

Figura 3.8 - Representação do Esquema (Modelo E-R)

Várias extensões de conceitos foram propostos ao Modelo E-R [TEOREY_86] com o objetivo de incorporar maior semântica às Entidades e Relacionamentos, possibilitando representar as Abstrações de Dados. Entre as implementações comerciais que se baseiam nas extensões do MER está o ZIM [FOARD_85].

3.3.2. - Modelo Relacional da Tasmânia (RM/T)

Este modelo é uma extensão do Modelo Relacional [JACKSON_90] e foi proposto por Codd [CODD_79] atendendo à crescente necessidade de captura de maior significado semântico em uma representação conceitual [CODD_82]. Esse objetivo foi alcançado introduzindo-se explicitamente a distinção entre **Relacionamentos** e **Entidades** e também através de restrições de integridade, que mantêm a consistência das operações sobre os dados armazenados.

As Entidades são representadas em **Relações-E** e **Relações-P** por **Tuplas** e seus respectivos **Atributos**, permitindo impor uma estrutura simples e regular à Base de Dados. Cada Entidade deve possuir um Atributo de identificação gerado pelo sistema denominado "Surrogate".

Relações-E registram a existência de **Entidades Núcleo** (ou "Kernel"), ou seja, Entidades que indicam a existência de representação de um elemento do mundo real na Base de Dados. Além disso, podem representar **Entidades Associativas**, que são composições muitos-para-muitos ("Associating") entre duas ou mais Entidades de outras **Relações-E**, ou ainda representar generalizações de forma a relacionar uma **Relações-E** como subconjunto (ou **Subtipo**) de outra **Relação-E**.

Relações-P registram a existência de **Entidades Característica**, que descrevem as Entidades Núcleo através dos valores de Atributos.

O Modelo da Tasmânia inclui ainda uma estrutura formal denominada de **Dicionário de Dados** (ou Catálogo), onde todas as **Relações** são descritas em termos de seu conteúdo semântico, de modo a torná-las conhecidas pelo Sistema de Gerenciamento de Bases de Dados e assim serem cumpridas todas as restrições impostas pelo modelo. Definido desta forma, o Dicionário constitui-se no Esquema de Dados deste modelo.

Uma descrição mais detalhada destes conceitos e outros não citados (incluindo, por exemplo, as operações fornecidas para a manipulação de entidades), além da descrição da DDL usada na Figura 3.9 para a definição do Esquema, poderá ser encontrada em [DATE_88b].

```
CREATE E_RELATION Indústria;
CREATE E_RELATION Fornecedor;
CREATE E_RELATION Produto;

CREATE E_RELATION Forn_Prod
  ASSOCIATING (Fornecedor VIA Forn$, Produto VIA Prod$);
CREATE E_RELATION Ind_Prod
  ASSOCIATING (Indústria VIA Ind$, Prod VIA Prod$);
CREATE E_RELATION Ind_Forn
  ASSOCIATING (Indústria VIA Ind$, Fornecedor VIA Forn$);

CREATE P_RELATION Carac_Forn FOR E_RELATION Fornecedor
  PROPRIETIES (Nome DOMAIN(DNome_Forn),
               Número DOMAIN(DNúmero),
               Rua DOMAIN(DRua),
               Cidade DOMAIN(DCidade));

CREATE P_RELATION Carac_Ind FOR E_RELATION Indústria
  PROPRIETIES (Nome DOMAIN(DNome_Ind),
               CGC DOMAIN(DCGC));

CREATE P_RELATION Carac_Prod FOR E_RELATION Produto
  PROPRIETIES (Nome DOMAIN(DNome_Prod),
               Código DOMAIN(DCódigo),
               Validade DOMAIN(DValidade));

CREATE P_RELATION Carac_Forn_Prod FOR E_RELATION Forn_Prod
  PROPRIETIES (Forn$ SURROGATE FOR Fornecedor,
               Prod$ SURROGATE FOR Produto);

CREATE P_RELATION Carac_Ind_Prod FOR E_RELATION Ind_Prod
  PROPRIETIES (Ind$ SURROGATE FOR Indústria,
               Prod$ SURROGATE FOR Produto);

CREATE P_RELATION Carac_Ind_Forn FOR E_RELATION Ind_Forn
  PROPRIETIES (Ind$ SURROGATE FOR Indústria,
               Forn$ SURROGATE FOR Fornecedor);
```

Figura 3.9 - Definição do Esquema (DDL/-RM/T--)

3.3.3. - Modelo de Dados Semântico (SDM)

O Modelo de Dados Semântico foi desenvolvido por Hammer e McLeod [HAMMER_81] com o objetivo de modelar as situações do mundo real através de um formalismo, apoiando-se no conceito de **Entidades** reais ou abstratas e **Classes** que agrupam Entidades com as mesmas características.

Toda Classe possui uma coleção de **Atributos de Classe** que descrevem genericamente as entidades que a compõe e, assim, estabelecem a semântica da Classe perante toda a Base de Dados. As Entidades contidas em uma Classe podem, por sua vez, possuir seus próprios atributos, chamados de **Atributos de Membro**.

As Classes são discriminadas em **Classes Básicas**, que possuem uma existência independente de outras Classes e **Classes Não Básicas**, as quais são definidas em termos de uma ou mais Classes.

Em termos de associações, existem **Relacionamentos** entre Entidades de distintas Classes, proporcionados através de valores de Atributo de Membro, os quais são repetidos nas Entidades envolvidas, de modo semelhante ao Modelo de Dados Relacional.

Para as Classes Não Básicas existem dois tipos de associações: as **Conexões de Subclasses** ("Subclass Connection") indicando quais Entidades que são membros de uma Classe Não Básica podem, mas não necessariamente, fazer parte de uma outra Classe denominada então de **Subclasse**; e as **Conexões de Agrupamento** ("Grouping Connection") que permitem expressar Classes cujas Entidades Membro são também Classes.

O Modelo de Dados Semântico pretende ser usado como um modelo de alto nível, que permita ao Administrador da Base de Dados descrever as prioridades do sistema e projetar uma representação lógica que sirva como especificação das informações, como mostra o exemplo apresentado na Figura 3.10, onde uma Linguagem de Definição de Dados estabelece o Esquema de Dados da aplicação Fiscalização.

Figura 3.10 - Definição do Esquema (DDL/--SDM--)

3.3.4. - Análise dos Modelos Semânticos

A contribuição mais importante dos Modelos Semânticos foi a sua capacidade de enriquecer a semântica dos relacionamentos, tornando-os poderosos elementos de representação, explicitamente diferenciados de quaisquer outros elementos através de

uma definição e representação própria. A Figura 3.11 ressalta esse fato através dos três itens de comparação abaixo descritos:

- **Representação de Relacionamento:** estabelece quais construtores cada Modelo de Dados utiliza para representar um relacionamento.
- **Abstrações de Dados:** estabelecem quais abstrações são suportadas por cada Modelo, como por exemplo, as Abstrações de Generalização e Agregação
- **Semânticas dos Relacionamentos:** estabelecem o grau de liberdade de um Modelo, para que se possa determinar os parâmetros semânticos de cada relacionamento individualmente, como por exemplo a cardinalidade ou herança.

É importante ressaltar que o MER foi originalmente criado por Chen com Associação (conjunto), porém existem extensões propostas para aumentar sua capacidade de Abstração.

	MER	RM/T	SDM
REPRES. DE RELACION.	(não especificado)	Entidade Associativa	Conexões
ABSTRAÇÕES DE DADOS	Associação	Agregação Composição Generalização Associação	Agregação Composição Generalização Associação
SEMÂNTICA DOS RELACION.	Selecionada pelo usuário	Pré-definida	Definida pelo usuário

Figura 3.11 - Características dos Modelos Semânticos

3.4.- Modelos Orientados a Objetos

Recentemente, têm surgido na literatura da área de Base de Dados várias propostas de Modelo diretamente aplicados a Sistemas de Gerenciamento de Banco de Dados Orientado a Objetos - SGBDOO [BERTINO_91]. Nesse sentido, nas próximas seções constataremos que seus autores apresentam diferentes tratamentos para alguns conceitos, de modo a aplicá-los às suas necessidades específicas.

3.4.1. - Cactis

Cactis [HUDSON_89] é um SGBDOO multiusuário desenvolvido na Universidade do Colorado, sendo considerado por seus autores uma união da filosofia dos Modelos Semânticos, uma vez que representa Objetos Complexos e Hierarquia de Generalização, com a filosofia dos Modelos Orientado a Objetos, pois possui a habilidade de encapsular Objetos enfatizando o aspecto comportamental de cada um. Seu objetivo é auxiliar no projeto de CAD/CAM e desenvolvimento de sistemas [HUDSON_87].

Uma Base de Dados no Cactis consiste de uma coleção de **Objetos Atômicos** (inteiro, real, "boolean", "string", vetor, registro) e **Objetos Abstratos** constituídos de Objetos Atômicos ou Abstratos, interligados por Relacionamentos. Seu Esquema de Dados por sua vez, contém a descrição destes elementos além de Predicados e Restrições.

Os **Objetos** são categorizados na Hierarquia de Generalização através de Atributos, reconhecendo-se seu tipo ou subtipo através de Predicados (características) definidos no Esquema. Portanto, o conjunto de Atributos de Objeto determina seu tipo e isso possibilita que mudanças no valor do Atributo de um Objeto possam, dinamicamente, mover o Objeto na Hierarquia de Generalização.

No Cactis os **Relacionamentos** são binários (um-para-muitos) e são definidos por um Tipo, direção e número de valores que fluem entre os Objetos interligados.

Atributo é um poderoso elemento de representação, sendo classificado em dois Tipos : **Intrínsecos** e **Derivados**. Os Atributos Derivados são indiretamente modificáveis através de operações resultantes de mudanças em Atributos Intrínsecos, que são os únicos que podem ser atualizados diretamente.

O mecanismo usado no Cactis para suportar o comportamento é possível através de Atributos Derivados, atribuindo-se a eles **Regras**, as quais permitem operar sobre outros Atributos do mesmo Objeto ou de outros Objetos. Desta forma, operações sobre um Objeto poderão desencadear uma série de ações, desde que estejam de acordo com as **Restrições** impostas pelo Esquema. Este mecanismo proporciona um eficiente controle de consistência de dados, pois alterações de informações irão disparar Regras que atualizarão outras informações, as quais, conseqüentemente, irão disparar outras Regras e assim por diante.

Versões de Objetos são usadas no Cactis para aumentar acessibilidade, ou seja, diminuir a concorrência de acesso através das operações de "Check in" e "Check out". Mudanças ocorridas em cada Versão são armazenadas em arquivos "Delta".

O mecanismo de desencadeamento de ações do Cactis parece proporcionar um eficiente meio de propagação de alterações do Esquema para os dados relacionados, porém os autores não se pronunciaram a esse respeito, referindo-se a esta questão como "não estudada" [HUDSON_88].

3.4.2. - EXODUS

EXODUS é um Sistema de Base de Dados desenvolvido a partir de 1985 na Universidade de Wisconsin [CAREY_89] e segue o Modelo de Dados denominado EXTRA, cujo objetivo é suportar aplicações científicas ou que utilizam vozes ou imagens, além de auxiliar projetos em engenharia.

O Modelo EXTRA define a Base de Dados como uma coleção de **Objetos** (simples ou complexos), que podem ser de tipos básicos pré-definidos (inteiro, "boolean", "string" e outros) ou então estruturas (tuplas, "sets", "arrays"), ambas podendo compor-se para formarem novos tipos ("Abstract Data Type").

O Gerenciador de Objetos do EXODUS está baseado em uma unidade de armazenamento de Objetos denominada de "**Storage Objects**" [RICHARDSON_87], o qual contém dados logicamente relacionados (Objeto), independente de ser composto por poucos "bytes" ou mesmo centenas de "megabytes". Um conjunto de "Storage Objects" logicamente ou fisicamente relacionados são agrupados em "**Files**".

Na formação do Esquema de Dados, o EXODUS possibilita três formas de definição de Atributos [CAREY_88]:

- i.) Atributo "own" que simplesmente denota um valor, não tendo maior representação semântica; (por "default" todos os Atributos são deste tipo);
- ii.) Atributo "own ref" cujo valor representa uma referência a um Objeto. Este tipo de Atributo estabelece que o Objeto o qual está sendo referenciado pertence ao Objeto que faz a referência e, assim, se este for eliminado, o Objeto referido é também eliminado;
- iii.) Atributo "ref" cujo valor é simplesmente uma referência a outro Objeto, sendo sua existência independente do Objeto que o referencia.

Os comandos abaixo demonstram como é a definição de um Tipo, de seus Atributos e a declaração de **Herança** entre os Tipos, neste caso estabelecendo que Indústria herda os Atributos de Empresa.

```
define type Indústria
(
    nome: char[15],
    CGC: char[15],

    forn: ref Fornecedor,
    prod: ref Produto
)

inherits Empresa
```

O Sistema de Gerenciamento suporta Versões de "Storage Objects". É determinado se um Objeto pode ter Versão, ou não, quando este é criado. Caso um Objeto autorizado a ter Versões seja atualizado, este é automaticamente copiado para uma nova "Storage Object", onde poderão ser realizadas as alterações. Para Objetos cujos volumes ultrapassem centenas de "bytes", o gerenciador estabelece que somente as informações que foram modificadas são mantidas em disco. Deste modo, as Versões compartilharão informações que possuem em comum e, assim, proporcionará uma economia de memória em disco.

Carey [CAREY_89] estabelece um algoritmo de remoção de Versões que considera necessário para uma real utilização do sistema por seus usuários, consistindo basicamente em percorrer o Plano de Versões, definindo quais as informações podem

ser apagadas da Versão desejada, por não serem usadas por nenhuma Versão ascendente ou descendente.

Os autores destacam o fato de que as Versões de Objetos não são usadas para o controle de concorrência porque não seriam eficientes.

3.4.3. - GemStone

O Sistema GemStone [BRETLE_89] foi desenvolvido pela Servio Logic Corporation por volta de 1987, baseando-se na linguagem SMALLTALK para compor seu modelo com três conceitos principais: **Objetos**, **Classes** e **Mensagens**.

Cada Objeto tem uma interface pública e uma memória privada. Os Objetos se comunicam através de Mensagens, que são solicitadas para que um Objeto mude seu estado, retorne algum valor ou realize alguma sequência de ações.

Os Objetos que compartilham o mesmo formato e Métodos são agrupados em Classes. As Classes são dispostas em uma hierarquia, onde cada Classe herda o comportamento e estrutura de sua superclasse.

O GemStone considera Classe como uma unidade básica passível de evolução [PENNEY_87] (semelhante ao ORION), na qual todas as primitivas de modificação do Esquema de Dados se baseiam para atualizar as informações.

Os autores deste sistema demonstram o interesse em duas abordagens de evolução do Esquema: Conversão, que é atualmente suportada pelo sistema; e "Screening", que pretendem implementar futuramente. Como ambas apresentam desvantagens: a Conversão consome muito tempo na modificação das Classes e "Screening" compromete a velocidade de execução de uma operação, os autores pretendem futuramente estender o Sistema para que suporte uma abordagem integrada de ambos os mecanismos, de forma a minimizar os problemas encontrados atualmente.

3.4.4. - Iris

Iris [WILKINSON_90] é um SGBDOO desenvolvido pela HP³, com o objetivo de satisfazer as necessidades dos sistemas de informatização de escritórios, CAD e CASE ("Computer Aided Software Engineering").

O Modelo de Dados Iris está fundamentado em três conceitos básicos: **Objetos**, **Tipos** e **Funções**. Cada Objeto representa uma entidade, sendo identificado por um "Surrogate" imutável, gerado pelo sistema.

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**.

Os Relacionamentos, Atributos e os aspectos operacionais (funcional/comportamental) de cada Objeto são expressos em termos de Funções. A declaração (aspecto funcional) de cada Função é definida separadamente da implementação (aspecto comportamental), proporcionando o encapsulamento de dados.

Uma consequência importante desses conceitos, determina que os dados e o Esquema da Base podem evoluir sem afetar as aplicações. Isto porque, um Objeto pode adquirir um Tipo ou perdê-lo dinamicamente, desde que esta operação seja consistente.

Esta operação é suportada pelo Iris, pois um Objeto pode ser de vários Tipos (desde que estes Tipos formem uma hierarquia), os quais podem ser criados e removidos sem causar grandes efeitos na Base de Dados. Por exemplo, supondo que exista uma hierarquia formada pelo Tipo Empresa e seus Subtipos Fornecedor e Indústria, criada a partir do comando:

```
CREATE TYPE Empresa SUBTYPE OF Fornecedor, Indústria;  
(nome String REQUIRED UNIQUE,  
faturamento Integer REQUIRED);  
  
CREATE FUNCTION imposto (nome,faturamento) -> Integer;
```

³ "Hewlett-Packard Laboratories"

Desde que o Tipo Empresa esteja definido e um Objeto (ICI, por exemplo) deste Tipo exista na Base de Dados, este Objeto poderá receber outro Tipo (um subtipo ou supertipo), passando a ter os Atributo do novo Tipo que recebeu, por exemplo:

ADD TYPE Indústria TO ICI;

Remover um Tipo é igualmente simples, sem conseqüências para os Objetos do Tipo removido, pois sua retirada implica somente na remoção dos Atributo deste Tipo para todos os Objetos que o tinham, por exemplo:

REMOVE TYPE Indústria;

As operações "CREATE", "ADD" e "REMOVE" podem ser igualmente usadas para Objetos, propiciando uma das maiores vantagens do Iris, que é a homogeneidade de sua linguagem, facilitando sua utilização pelos usuários.

O Iris possui ainda um Editor Gráfico baseado em janelas [FISHMAN_89] o qual possibilita ao usuário navegar e atualizar a Base de Dados. Nele é apresentada a Hierarquia de Tipos e, se desejado, as Funções de cada Tipo para que os usuários possam criar Tipos e Funções na Hierarquia.

Para as aplicações a que se propõe o Iris, é necessária a preservação dos estados alternativos de uma particular entidade. Isto requer a existência de um mecanismo que suporte Versões de modo a controlar o acesso a estes dados. Para isto, o Iris define os Objetos como passíveis de **Versão**.

A referência a um Objeto com muitas Versões poderá ser direta através de uma referência específica a uma determinada Versão, ou então uma referência a um **Objeto Genérico**, representando um conjunto de Versões.

As Versões definitivas são criadas explicitamente pelo usuário através do comando "CREATE". Versões utilizadas para controle de concorrência são duplicadas e unificadas através dos comando de "CHECKIN" e "CHECKOUT", respectivamente.

Uma importante restrição em Versões do Iris refere-se às modificações do Esquema, isto porque modificações em um Tipo irão acarretar em mudanças em todas as Versões que contenham Objetos do Tipo modificado e não apenas em algumas Versões, como as vezes é desejável.

3.4.5. - O2

O Modelo e o sistema O2 [LECLUSE_88] fazem parte do projeto ALTAIR, desenvolvido por um grupo de pesquisadores da França. O objetivo deste sistema é suportar aplicações tradicionais, automação de escritórios e aplicações multimídia.

No O2, as informações são organizadas em Objetos que possuem um identificador, dados (organizados em tuplas ou "sets") e Métodos encapsulados que propiciam a manipulação dos Objetos.

Existem duas estruturas no O2: **Tipos** constituídos por inteiro, "string", "float", "set", tupla entre outros, que possibilitam a estruturação de Objetos; e **Classes** estruturadas em uma hierarquia que descreve a estrutura e o comportamento dos Objetos, possibilitando a definição de **Herança** (e Múltipla Herança), de Métodos e Atributos.

O **Esquema de Dados** é definido pelo conjunto de Classes, relacionando-as por ligações de herança e/ou composição.

```
add class Empresa
    type tupla (nome: string,
               faturamento: integer)

add class Empresa inherits Indústria, Fornecedor
```

Cabe ao Gerenciador do Esquema do O2 [DEUX_90] suportar a criação e remoção de Classes, Tipos e Métodos. A política seguida nas remoções tenta evitar qualquer forma de conflito com os dados armazenados. Assim, Classes são removidas se não possuírem Objetos e Classes dependentes e os Métodos são retirados da Base de Dados, se não são herdados ou estiverem ativos.

Uma forma encontrada para proporcionar flexibilidade à estrutura de Classes são as chamadas **Classes Virtuais** [ABITEBOUL_91], que consistem em formar Classes a partir de Visões dos usuários definidas em operações da DML. Conceitualmente, um programador pode construir uma Hierarquia de Classes Virtuais tanto na forma "Top-down" (via especialização) quanto "Bottom-up" (via generalização).

Outro conceito incorporado ao O2, respeitando as necessidades do usuário principalmente na fase de projeto quando nem todas as informações estão disponíveis, é a chamada **Especificação Incompleta** [ZICARI_90], que consiste em definir Classes (ou Esquema) e até Objetos de maneira incompleta, ou seja, algumas informações deixarão de ser armazenadas na Base de Dados por não estarem disponíveis.

3.4.6. ORION

O projeto ORION [KIM_89c] foi lançado no final de 1985 pela "Microelectronics and Computer Technology Corporation" - MCC, Austin - Texas, tendo como objetivo desenvolver protótipos de Sistemas de Base de Dados Orientados a Objetos, para suportar aplicações em CAD, "Computer-Aided Engineering" - CAE e sistemas de informatização de escritórios.

Utilizando-se do paradigma da programação orientada a objetos, no projeto ORION [BANERJEE_87] [KIM_90a] todas as entidades conceituais são modeladas como **Objetos**, onde o comportamento de cada Objeto é descrito em **Métodos**. Os **Atributos** (variáveis instanciadas) são também Objetos, cujo valores estão em algum domínio (**Classe**). O **Identificador** de um Objeto consiste na identificação da Classe que o contem e de seu identificador interno à Classe.

Uma Classe pode ter uma ou mais subclasses, formando assim uma **Hierarquia de Classes**. Esta estrutura possui ligações "IS_A" entre seus nós, portanto representam uma Hierarquia de Generalização que suporta **Herança**. Múltiplas Heranças entre Classes também podem ocorrer e, neste caso, o modelo permite ao usuário definir uma solução para o problema.

A Hierarquia de Classes possui uma parte de sua estrutura invariante, controlada pelo sistema, contendo uma raiz chamada de Classe OBJETO e duas subclasses que definem os domínios primitivos (inteiro, "boolean", "string", "float" e "set"). Uma terceira subclasse possibilita a definição da Hierarquia de Classes das aplicações.

Em uma extensão do ORION, proposto em [KIM_89a], as referências (ou

Relacionamentos) entre Objetos ocorrem quando um Objeto referencia outros através do Identificador. As Referências são distintas em dois tipos: Fracas e de Composição. As Referências Fracas são relacionamentos padrão que existem nos sistemas orientados a objetos. As Referências de Composição ("IS-PART-OF") representam uma semântica onde um Objeto pode ser parte de um ou mais Objetos. Neste ponto diferenciam-se em: Referência de Composição Exclusiva, quando um Objeto faz parte exclusivamente de um outro Objeto; e Referência de Composição Compartilhada se um Objeto é parte de muitos objetos.

Os Objetos relacionados através de Referências de Composição formam uma hierarquia que define um **Objeto Complexo** [KIM_87]. Isto ocorre porque um certo Objeto possui uma Referência de Composição com um Objeto de outra Classe. Este outro Objeto poderá ter Atributos que referenciam outros Objetos e assim por diante, até que a hierarquia termine através de Referências aos domínios primitivos. As Classes envolvidas nestas Referências de Composição sucessivas formam a **Hierarquia de Composição de Classes**.

O ORION suporta **Versões** apenas para Objetos cuja Classe o usuário declarou explicitamente existirem Versões. Uma extensão deste conceito leva à definição de Versões para Objetos Complexos, uma vez que, se criarmos Versões de um Objeto pertencente a uma hierarquia que define um certo Objeto Complexo, teremos Versões de formação do Objeto Complexo.

O modelo classifica as Versões em dois tipos: **Versões de Trabalho** (ou estáveis), que podem apenas ser consultadas ou removidas; e as **Versões Transientes** (ou temporárias), que são criadas a partir de uma Versão de Trabalho, podendo ser atualizadas, retiradas e\ou promovidas a Versões de Trabalho. No O2 existem **Objetos Genéricos** para manter o histórico das Versões de Objetos.

O **Esquema de Dados** no modelo ORION representa duas dimensões de envolvimento entre as Classes. Uma dimensão é a Hierarquia de Classes, que captura as Abstrações de Generalizações. Outra dimensão, chamada de Hierarquia de Composição de Classes, apresenta a semântica de composição entre Objetos envolvendo suas Classes respectivas. Assim, temos definido no Esquema de Dados os tipos de Objetos das Classes, a estrutura hierárquica das Classes e os Relacionamentos entre estes Tipos de Objetos.

Para o ambiente de trabalho das aplicações, o modelo ORION proporciona duas formas de evolução do Esquema de Dados de uma Base [KIM_90b], sendo ambas suportadas por um editor gráfico de manipulação e validação dos dados do Esquema. Uma das técnicas implementadas possibilita ao usuário manipular algumas informações do Esquema, tais como:

1. Manipular uma Classe
 - 1.1 Adicionar uma classe
 - 1.2 Remover uma classe
2. Manipular a Hierarquia de Classes
 - 2.1 Criar uma superclasse
 - 2.2 Remover uma superclasse
3. Manipular o conteúdo de uma Classe
 - 3.1 Adicionar um novo atributo à classe
 - 3.2 Remover um atributo de uma classe
 - 3.3 Trocar o domínio de um atributo
 - 3.4 Remover uma referência de composição

Usando esta técnica, as mudanças no Esquema poderão afetar fisicamente a Base de Dados. Para atualizar os dados da Base que foram afetados pelas mudanças do Esquema, o ORION utiliza uma abordagem chamada Conversão Incremental, a qual consiste em atualizar as informações à medida que estas vão sendo manipuladas pelo sistema.

Outra técnica suportada pelo modelo, chamada de "Screening", consiste na utilização de Versões do Esquema da Base de Dados, permitindo ao usuário criar várias Versões do Esquema lógico, de modo que os usuários poderão ter diferentes "visões lógicas" da Base de Dados, indicando-se qual a versão a ser utilizada.

3.4.7. - OZ & OZ+

OZ e sua extensão OZ+ [WEISER_89] são protótipos de sistemas para um ambiente de programação orientado a objetos desenvolvidos na Universidade de Toronto, com características que procuram suportar as necessidades de informatização de escritórios.

O modelo do OZ (e OZ+) inspira-se nos conceitos da linguagem SMALLTALK para definir seus elementos. Deste modo, **Objeto** é uma entidade discreta que encapsula dados e operações (**Regras**). 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. Um Objeto pode ser simples ("string", "integer", etc), na forma de lista, vetor ou árvore.

Objetos são classificados em dois tipos: **Objeto Permanente**, que é criado ou destruído, dependendo de seu Objeto parente; e **Objeto Temporário**, que depende da existência de uma referência a ele: se a referência for retirada, então o Objeto é destruído.

O conjunto de Objetos da Base de Dados são particionados em **Classes**, de modo que cada Objeto pertence a apenas uma Classe. As Classes são organizadas em nós de uma B-TREE, sendo que os Objetos de uma Classe herdam Atributos e Regras de todas as Classes de nível superior. A maioria das Classes são definidas pelo usuário, porém existe um pequeno número de Classes pré-definidas pelo sistema, com um respectivo conjunto de Regras que facilitam a interação do usuário com o sistema, como por exemplo a Classe "user" que contém um Objeto para cada usuário do sistema.

Em OZ, um **Relacionamento** entre Objetos implica na possibilidade dos Objetos se comunicarem diretamente através de **Mensagens**.

Existem Objetos na Base de Dados em dois estados: **Ativo e Passivo**. Um Objeto Passivo não possui Regras, sendo associado a um especial Objeto da Base de Dados e não sendo referenciado em nenhuma Regra. Objetos Ativos são os Objetos normais, contendo dados e Regras. O sistema oferece Regras para a manipulação destes estados, que permitem criar um Objeto Passivo a partir de um Objeto Ativo ("**omega rules**") e criar um Objeto Ativo a partir de um Objeto Passivo

("alpha rules").

Segundo Weiser, os Objetos Passivos podem corresponder a Versões de um Objeto, de modo a estabelecer um histórico da evolução de cada Objeto.

3.4.8. - PCTE & PCTE+

O PCTE ("Portable Common Tool Environment") e sua extensão PCTE+ [BOUDIER_88] [PCTE_89] são conjuntos de "interfaces" que suportam a construção de ambientes de desenvolvimento de sistemas integrados e podem ser transportados para diferentes "hardwares" e Sistemas Operacionais. O PCTE foi inicialmente desenvolvido pelo projeto ESPRIT, sendo este um consórcio pan-europeu de empresas como: BULL S.A., General Electric Company, OLIVETTI e SIEMENS entre outras.

O desenvolvimento do PCTE iniciou-se em outubro de 1983 e resultou na especificação de ferramentas em linguagem C e ADA. Sua definição pode ser descrita em 3 aspectos principais:

- i) **Mecanismos Básicos:** correspondem a funções de execução, comunicação e manipulação de elementos que podem existir no contexto do desenvolvimento;
- ii) **Interface do Usuário:** corresponde à forma de interação do usuário com as ferramentas;
- iii) **Distribuições:** são mecanismos de protocolo para suportar a transparência das distribuições de dados.

Entre os Mecanismos Básicos encontra-se um **S. de Gerenciamento de Objetos** [GALLO_86] composto por um conjunto de primitivas para manipulação de **Objetos**.

Para os ambientes de desenvolvimento de sistemas a que se propõe o PCTE, a manipulação de Objetos é feita a nível de programas, de documentos, de bibliotecas ou arquivos gerados pelo computador. Consequentemente, todos os Objetos (**Arquivos**) têm suas características descritas por:

- um conjunto de **Atributos**, entre os quais alguns pré-definidos pelo sistema como a Data do último acesso ou indicação de duplicação;
- um conjunto de **Relacionamentos Bidirecionais**;
- um Atributo **"Content"**, que define o conteúdo do Objeto ("file", "message queue", "block device").

Cada **Tipo de Objeto** define as características de todas as suas instâncias. O conjunto de Tipos forma uma Hierarquia (de Generalização) suportando o conceito de **Herança e Múltipla Herança**.

Os Atributos definem as propriedades intrínsecas de um Objeto. Cada **Tipo de Atributo** define as caract. de todas as suas instâncias, sendo composto por um nome, **Tipo de Valor** ("integer", "boolean", "string") e um valor inicial (0, "false", "empty").

Um **"Link"** é uma associação unidirecional entre dois Objetos e pode ser usado para representar uma referência simples entre Objetos ou para modelar estruturas.

Um **Relacionamento** representa uma mútua dependência entre dois Objetos, sendo este determinado por dois **"Links"**, que juntos proporcionam as cardinalidades um-para-um, um-para-muitos ou muitos-para-muitos.

O Modelo define a organização do Esquema de Dados em conjuntos não disjuntos, chamados de **"Schema Definition Sets (SDS)"**, onde cada um agrupa um conjunto de definições específicas para um usuário, para uma ferramenta ou para um grupo de usuário de um mesmo projeto. A criação de um **"SDS"** é realizada definindo-se seus elementos componentes, independentemente de qualquer outro SDS.

Uma característica importante proporcionada pelo PCTE é o compartilhamento de definições de dados por diversos usuários, através da operação de **"IMPORT"**, que estabelece o compartilhamento de definições de um SDS origem e um SDS destino que efetuou a importação de definições.

Um processo de trabalho poderá necessitar de vários SDS's na definição de sua Visão do Banco de Dados. A lista formada por SDS's utilizados em um processo é chamada de **"Working Schema"**. No processo de trabalho é realizada a composição de SDS's do respectivo **"Working Schema"**, através da união de elementos e conjuntos que não causam inconsistências. No caso de ocorrer algum conflito de informações, é realizada uma escolha das informações prioritárias, seguindo a ordem da lista de SDS's definidas pelo **"Working Schema"**.

3.4.9. - Análise dos Modelos Orientados a Objetos

Como se pode notar, existe grande variação de propostas, o que torna complexa qualquer comparação entre os modelos. No sentido da objetividade, o enfoque desta comparação foi dirigido para as atividades de conceitos correlatos aos Esquemas de Dados, mais precisamente os seguintes itens:

- **Classe/Tipo:** estabelece quais destes conceitos são especialmente tratados em cada Modelo. Genericamente, pode-se observar que Classes são usadas, fundamentalmente no encapsulamento de Objetos. Já os Tipos são utilizados para a composição de Tipos Básicos, ampliando o domínio que um Objeto pode ter.
- **Representação de Objetos Complexos:** estabelece o dispositivo utilizado em cada Modelo para suportar o conceitos de Objetos Complexos.
- **Versões:** estabelecem quais elementos são passíveis de Versões.
- **Utilização de Versões:** estabelece quais são as propostas de utilização de Versões. Como descrito no capítulo anterior, existem três funções básicas: Histórica, Controle de Concorrência e Minimização dos efeitos de alteração.
- **Evolução de Esquema:** define quais dispositivos cada modelo utiliza para dotar sua Base de Dados com este conceito fundamental para as aplicações atuais.
- **Aplicações:** define quais aplicações são apropriadas para cada Modelo.

Um aspecto percebido através da Figura 3.12 é o fato de nenhum modelo ter usado Versões para minimizar alterações na Base de Dados. Em outro ponto, percebe-se que os modelos não exploraram o conceito de Evolução de Esquema, ignorando ou então detendo-se superficialmente neste conceito fundamental para dotar a Base de Dados de grande flexibilidade.

	Cacis	EXODUS	GemStone	Iris	O2	ORION	OZ+	PCTE+
CLASSE/TIPO	Classe	Tipo e Classe	Classe	Tipo	Tipo e Classe	Tipo e Classe	Classe	Tipo
GERENC. DE OBJ. COMPL.	NÃO especif.	Atib. (owner ref)	NÃO descrito	NÃO descrito	Atributo	Refer. de Composição	Atributo	Link
VERSÕES	Objetos	Objetos	NÃO descrito	Objetos	NÃO descrito	Objetos e Esquema	Objeto	NÃO descrito
UTILIZAÇÃO DE VERSÕES	Histórico e Contr. de Concorrência	Histórico	NÃO descrito	Histórico e Controle de Concor.	NÃO descrito	Histórico e Visões do Esquema	Histórico	NÃO descrito
EVOLUÇÃO DO ESQUEMA	NÃO suporta	NÃO descrito	Conversão	NÃO descrito	NÃO descrito	Conversão Incremental e Screening	NÃO descrito	Conv. e Screening
APLICAÇÕES	CAD/CAM, CASE	Científicas, Projetos de Engenharia	NÃO descrito	CAD, CASE, Aut. de Escri.	Aut. de Escri., S. Multi-mídia	CAD, CAE, Aut. de Escri.	Aut. de Escri.	CAE

Figura 3.12 - Características dos Modelos Orientados a Objetos

3.5. - Considerações Finais

Neste capítulo analisamos alguns Modelos de Dados, procurando extrair conceitos relevantes para este trabalho. Adicionalmente, foi possível perceber a evolução dos Modelos em termos das três classes definidas (Baseados em Registro, Semânticos e Orientados a Objetos), uma vez que novos conceitos eram sempre incorporados na busca de uma modelagem mais completa.

A evolução histórica dos Modelos de Dados (Figura 3.13) [MCLEOD_91] [KORTH_89] demonstra que os Modelos Orientados a Registros surgiram inicialmente em resposta às necessidades de processamento da época. Anos depois, surgiram os primeiros Modelos Semânticos, juntamente com os conceitos de Objetos e Abstração de Dados. Posteriormente, iniciou-se a pesquisa e conceituação dos Modelos Orientados a Objetos, os quais atualmente continuam a ser expandidos para aumentar sua capacidade de representação.

Figura 3.13 - Evolução Histórica dos Modelos de Dados

Capítulo 4

O MODELO DE REPRESENTAÇÃO DE OBJETOS

4.1. - Considerações Iniciais

O Modelo de Representação de Objetos - MRO* [TRAINA_86] [TRAINA_88] foi elaborado, no Departamento de Ciências de Computação e Estatística do ICMSC-USP em São Carlos no final de 1986, com o objetivo de permitir a construção de Sistemas de Gerenciamento de Bases de Dados que atendam às necessidades de manipulação de dados e representação de informações dos sistemas de apoio a projetos tradicionais de engenharia ("Computer Aided Design" - CAD), ou de ambientes de suporte ao desenvolvimento de "softwares" ("Computer Aided Software Engineering" - CASE).

Este capítulo tem por objetivo introduzir os conceitos que envolvem o MRO*, bem como os subsistemas que compõem o GERenciador de Objetos - GEO, um protótipo de SGBDOO apoiado no MRO*.

Para efeito de apresentação do Modelo de Representação de Objetos, são descritos sucintamente seus conceitos e definições, detalhando-se especialmente o conceito de Colônias de Objetos, no qual se baseia este trabalho. A completa descrição de todos os conceitos, bem como o formalismo matemático definido para MRO*, podem ser encontrados em [BIAJIZ_92].

4.2. - Conceituação

Para ilustrar esta seção e como exemplo prático do MRO*, é utilizado uma modelagem simplificada de aplicação, onde estão representadas algumas informações sobre um ambiente de desenvolvimento e testes de veículos terrestres. A completa modelagem deste exemplo pode ser encontrada no Apêndice A.

4.2.1. - Diagramas de Representação

Nota-se que os Modelos de Dados Orientados a Objetos (apresentados no capítulo anterior) não têm uma preocupação, pelo menos aparente, com a representação gráfica de seus dados.

No MRO* esta preocupação existe, tanto para a apresentação do Esquema de Dados (tipos) quanto para os dados (instâncias) armazenados na Base. Tais informações podem ser representadas através dos seguintes diagramas:

- **DIAGRAMAS DE REPRESENTAÇÃO DE INSTÂNCIAS -DRI:** onde pode-se representar os dados (instâncias de Tipos) que estão armazenados na Base de Dados. Sua função é apresentar apenas as informações de interesse de um usuário em um certo momento.
- **DIAGRAMAS DE REPRESENTAÇÃO DE OBJETOS -DRO:** onde são representados os Tipos, Relacionamentos e Atributos que compõem o Esquema de Dados. Nele pode-se representar o Esquema de Dados completo ou apenas uma parte em que se demonstre interesse.
- **DIAGRAMAS HIERÁRQUICOS DE COLÔNIAS -DHC:** onde se representa a Hierarquia de Composição, ou melhor, a Hierarquia de Colônias em que se está operando.

4.2.2. - Fundamentos

Seguindo a linha dos Modelos Semânticos [HULL_87], o MRO* modela as entidades do mundo real em **Objetos**, os quais podem associar-se a outros Objetos através de **Relacionamentos**. Tanto Objetos quanto Relacionamentos possuem **Atributos**. Todo Objeto possui um **Código** gerado pelo sistema, que o identifica univocamente durante toda a sua existência na Base de Dados. Além disso, um Objeto possui identificadores ou sinônimos fornecidos pelos usuários.

O significado de um Atributo pode ser reconhecido pelo SGBD através de sua **Característica** (sinônimo, comentário, tempo, estrutura, gráfico, imagem, visualização, regra, propriedade, procedimento) e seu **Tipo de Dado** (byte, int, long, coditipo, float, double, cadeia, 2dcoordenadas, data, hora, valor).

As associações básicas são chamadas **Relacionamentos Binários** e associam dois Objetos denominados: **Objeto Origem** e **Objeto Destino**, através de **Modalidades**. Uma associação mais elaborada é chamada **Relacionamento Triplo**, sendo composta por dois ou mais Relacionamentos Binários que possuem em comum um Objeto.

Para tornar a modelagem mais natural, são mantidos na Base de Dados certos Relacionamentos e Atributos que não precisam ser definidos explicitamente pelo usuário. Tais elementos são ditos **Intrínsecos**. Um exemplo é o Atributo que armazena uma informação indicando quando um Objeto foi criado na Base.

Exemplificando os conceitos anteriormente citados, na Figura 4.1 é apresentado um Diagrama de Representação de Instâncias, onde vemos Objetos sendo representados por círculos (veículo 20, pessoa Nelson, circuito Montecarlo, fábrica Mercedes), Relacionamentos Binários representados por arcos direcionados e usando suas respectivas Modalidades (projetado por, correu em, trabalha com), Relacionamento Triplo usando sua Modalidade (testa em) para representar-se por um arco entre Relacionamentos Binários, e Atributos representados por traços ligados aos Objetos correspondentes (piloto, 34 anos, brasileira, Mônaco, Brasil).

Figura 4.1 - Exemplo de um DRI

4.2.3. - Abstrações de Dados

- Agregação

Esta forma de abstração no MRO* é denominada de **Máscara** e possibilita ao usuário manipular coletivamente um grupo de Objetos. Uma Máscara estabelece a possibilidade de manipulação de um Objeto como uma tupla de uma relação (coleção de Objetos), permitindo um tratamento uniforme a todos os Objetos dessa coleção.

- Classificação

Os elementos de informação do MRO* são classificados em Tipos, porém esta classificação segue uma semântica pré-definida. Para que outras operações de classificação possam ser feitas sob responsabilidade apenas dos aplicativos, existe o conceito de **Seleção** através do qual podem ser feitas **Seleções** de Objetos, as quais, podem ser operadas pelas operações usuais do sistema. Seleções de Objetos juntamente com Máscara, constituem-se em conceitos do MRO* que permitem a existência de "Visões Relacionais" sobre uma Base de Dados [PIZZIGATTI_92].

- Generalização

Todos os Objetos, Relacionamentos e Atributos são classificados em **Tipos**. Tipos de Objetos e Tipos de Relacionamentos podem ser especializados em **Subtipos de Objetos** e **Subtipos de Relacionamentos** respectivamente. Um Subtipo é especificado através da indicação de um Tipo ou Subtipo anteriormente definido, chamado **Tipo Mestre** e de um **Predicado**, que estabelece o critério de escolha dos elementos do Tipo Mestre que constituirão o sub-conjunto de elementos daquele Subtipo.

Tipos e Subtipos são mantidos em Hierarquias de Generalização [SMITH_77a] através de *relacionamentos intrínsecos de generalização*.

Para ilustração, um exemplo de Hierarquia de Generalização é apresentado na Figura 4.2, onde temos que o Tipo de Objeto Pessoa possui dois Subtipos: PILOTO e PROJETISTA. Por meio da hierarquia, os subtipos herdam os Tipos de Atributos do Tipo de Objeto Pessoa: idade, função e nacionalidade.

Figura 4.2 - Hierarquia de Generalização (DRO)

- Composição (Colônias de Objetos)

Colônia é uma partição do conjunto de Objetos da Base de Dados, sendo portanto, constituída de um grupo de Objetos de diversos Tipos, chamados de *objetos habitantes da colônia*, e de Relacionamento e Atributos associados. Define-se que cada Colônia de Objetos refere-se a um determinado aspecto de interesse da aplicação.

Para uma maior abrangência do MRO*, além da partição do conjunto de Objetos em Tipos de Objetos com as mesmas propriedades, percebeu-se que os Objetos de uma Base devem ser classificados segundo outros aspectos, de acordo com os seus Tipos. Considerando-se que cada Objeto habita exatamente uma Colônia, estabelecemos uma partição no conjunto de Tipos de Objetos, sendo cada parte denominada **Tipo de Colônia**. Deste modo, cada Tipo de Colônia determina um conjunto de Tipos de Objeto e, assim, Objetos de um certo Tipo somente habitam Colônias de um único Tipo.

O conceito de Colônia estabelece *relacionamentos intrínsecos de composição* entre os Objetos da Base de Dados e, conseqüentemente, a formação de uma Hierarquia de Composição, ou seja, um Objeto O_1 qualquer, através de um relacionamento de composição estabelecido pelo gerenciador, possibilita o acesso ou a definição de uma Colônia cujos Objetos habitantes são componentes de O_1 . Cada

Figura 4.3 - Colônia Global (DRO)

Objeto componente pode, por sua vez, ter relacionamentos de composição com Objetos de outras Colônias.

Isto possibilita aumentar ou diminuir a especificidade de um Objeto, como se a "visão" que se tem da informação pudesse ser mais ou menos detalhada, a critério do próprio usuário. Sendo assim, quando vista de maneira global, a Base de Dados é apresentada com um pequeno grau de detalhe, com Objetos pertencentes a uma Colônia criada pelo sistema, chamada **Colônia Global** (Figura 4.3). Se o interesse recair em um dos componentes visíveis, o grau de detalhe deste componente aumenta.

Em termos de acesso a Objetos, isso corresponde a se ter poucos Objetos acessíveis enquanto se manipula a Base de um ponto de vista global. Quando indicado ao sistema para considerar (refinar) um desses Objetos, aponta-se um novo grupo de Objetos (nova Colônia) do qual se deseja manipular informações, e assim tornam-se acessíveis os novos Objetos. Na Figura 4.4 é apresentada uma Hierarquia de Composição, onde qualquer Objeto do tipo VEÍCULO, habitando a Colônia Global, pode ser refinado em Objetos dos Tipos: SISTEMA MECÂNICO, CARROCEIRA e CHASSIS, que habitam uma Colônia do Tipo SISTEMA.

Figura 4.4 - Hierarquia de Composição (DHC)

O procedimento de refinamento pode ser repetido sucessivamente, indicando-se para consideração um dos novos Objetos disponíveis, o que aumenta gradativamente o nível de detalhe e, conseqüentemente, a quantidade de Objetos a que se tem acesso através de seus sinônimos. Com o refinamento de um determinado Objeto, os demais, em um mesmo nível de detalhe (mesma Colônia), não deixam de estar acessíveis, mas não são também refinados. Desta forma, torna-se mais "natural" para o usuário indicar o escopo de percepção ou de identificação dos dados, especificando-se o nível de detalhe desejado através da navegação pela Hierarquia de Composição, restringindo-se, assim, o grupo de Objetos passível de ser acessado através de identificadores unívocos. Com isso, estabelecemos a idéia de **Contexto**, ou seja, abrangência de identificação unívoca de um Objeto por um sinônimo, dentro da hierarquia. Qualquer referência a outras informações fora do Contexto deverá possuir mais detalhes sobre o Objeto pesquisado, ou seja, um grau de identificação maior, uma vez que será necessário diferenciá-lo de todo o conjunto de informações da Base de Dados.

Indicar um Objeto para consideração (ou refinamento) caracteriza a operação de *abrir* uma Colônia para acesso (colocando-a no Contexto), de modo que os Objetos habitantes daquela Colônia tornam-se acessíveis para consulta e modificação ou apenas para consulta. Desta forma, Colônias possibilitam o controle e a abrangência (granularidade) do gerenciamento de acesso à Base de Dados. A indicação de um Objeto pode abrir mais de uma Colônia de Tipos diferentes. O **Objeto Considerado** é definido como um Objeto que se tem presentemente disponível e que se indica para refinamento, fazendo com que a Colônia que contenha os Objetos que o especificam com um maior grau de detalhe (*Colônia Constrita pelo Objeto Considerado*), seja aberta.

Das várias instâncias de Colônias de cada Tipo que podem existir, apenas uma pode estar aberta num dado instante. Portanto, quando se indica para consideração um Objeto que constringe uma Colônia de um mesmo tipo ao de outra já aberta, esta é *fechada*, deixando de se ter acesso a ela e a todas as suas Colônias hierarquicamente subordinadas, as quais são fechadas também. A seguir, o mesmo Tipo de Colônia é novamente aberto, tornando disponíveis os Objetos da Colônia constrita (ou considerada) pelo novo Objeto Considerado. As demais Colônias da hierarquia superior não são afetadas por essa operação.

A exceção entre as Colônias é a Colônia Global, que deve estar sempre aberta, porque o aspecto global de uma aplicação deve estar permanentemente disponível. É importante notar também que a não ser a Colônia Global, que é a única de seu Tipo, qualquer outro Tipo de Colônia pode ter qualquer número de instâncias de Colônia.

Também pode-se salientar que o conceito de Colônia é distinto do conceito de Classes de Objetos [BORGIDA_88], pois enquanto uma Classe corresponde a todos os Objetos de um mesmo Tipo, uma Colônia inclui Objetos de diversos Tipos diferentes e, a não ser a Colônia Global, as outras Colônias não exaurem o conjunto dos Objetos de cada um dos Tipos que as habitam.

Em suma, as Colônias organizam-se segundo uma **Hierarquia de Colônias**, em que a Colônia Global é o topo da hierarquia e o nível imediatamente subordinado é constituído pelas Colônias consideradas por Objetos que habitam a Colônia Global e assim sucessivamente. Essa hierarquia indica a existência de um relacionamento intrínseco de composição entre os Objetos que consideram as Colônias e os Objetos que as habitam.

- Sumarização

Alguns Objetos podem ser semanticamente agrupados com outros e coletivamente caracterizados por um Objeto (agregado). Este Objeto é descrito como sendo uma **Sumarização** de um conjunto de Objetos pertencentes a uma mesma Colônia.

Cada Sumarização possui relacionamentos com Objetos de quaisquer Tipos ou Subtipos, bem como Atributos definidos através de operações específicas sobre os Objetos que constituem o conjunto sumarizado, como por exemplo somatória, mínimo, máximo e média.

4.2.4. - Esquema e Meta-Esquema de Dados

Considerando que cada Colônia, Objeto, Relacionamento e Atributo possui seu Tipo respectivo, pode-se descrever um esquema genérico do modelo MRO* através de um **Meta-Esquema de Dados** (Figura 4.5), onde cada elemento pode ser encarado como um Meta-Tipo, ou seja, uma definição do que é "ser um Tipo" e qual a sua relação com os outros Tipos, exemplo: que Tipos de Objetos devem habitar um certo Tipo de Colônia. Assim, através do Meta-Esquema de Dados, é possível empregar o MRO* na representação de Esquemas de qualquer aplicação passível de ser modelada.

A existência de um Meta-Esquema de Dados permite então, definir o **Esquema de Dados** de uma aplicação como um conjunto de informações representadas em uma estrutura sintaticamente idêntica à usada na representação dos dados da Base, sendo, portanto o Esquema, uma Base de Dados (Meta-Base de Dados) com a mesma estrutura da Base de Dados que ela define e assim, o Esquema pode ser armazenado em uma Colônia de Objetos, denominada **Colônia Esquema**.

Uma consequência imediata disso, é que um SGBDOO apoiado no MRO* permite o gerenciamento do Esquema, sendo então, um Meta-SGBDOO que manterá uma Meta-Base de Dados.

Neste SGBDOO, co-existem a Meta-Base de Dados (**Base Intencional**), onde estão as informações **léxicas, sintáticas e semânticas** que constituem o Esquema dos dados usados nas operações dos usuários, e a Base de Dados (**Base Extensional**) contendo as informações manipuladas pelos usuários.

5

4.2.5. - Versões e Alternativas

O MRO^{*} suporta a armazenagem das várias Versões e Alternativas de um projeto, permitindo a geração de **Instâncias de Colônias**, de maneira que uma Colônia constricta por um Objeto pode ter várias Versões e Alternativas. Cada instância é chamada de uma **Variante** e corresponde a uma Versão ou uma Alternativa, tendo cada uma Atributos Intrínsecos de acesso e proteção independentes das demais instâncias da mesma Colônia, podendo portanto, ser acessada por usuários de maneira independente das demais.

Cada instância de uma Colônia que for considerada Versão, será "Congelada" e não se obterá mais o acesso a ela para escrita. No caso em que alguma alteração de uma Versão for necessária, deve ser criada uma Alternativa que poderá ser modificada. Em termos de operações na Base de Dados, isso corresponde à duplicação da Colônia constricta pelo Objeto a ser alterado, criando-se uma nova instância da Colônia. A Colônia que corresponde à Versão original permanecerá protegida contra escrita, e a Alternativa criada (inicialmente idêntica à Versão original) fica liberada para alterações. Se as alterações eventualmente levarem a uma nova Versão, esta será, por sua vez e a seu tempo, também congelada e protegida contra escrita.

Cada usuário pode ter disponível apenas uma das instâncias de cada Tipo de Colônia em um certo instante, a qual é chamada de **Variante Corrente** da Colônia.

4.3. - GERenciador de Objetos - GEO

Em meados de 1988, iniciou-se o desenvolvimento de um SGBDOO denominado GERenciador de Objetos [TRAINA_91], o qual incorpora os conceitos do MRO^{*}. Um dos objetivos desse sistema é a validação prática e o estudo de como os conceitos do MRO^{*} e de Sistemas Orientados a Objetos em geral, podem ser implantados em ferramentas reais.

Para a tarefa de implementar o Gerenciador de forma que este possa manipular eficientemente a Base Extensional e a Base Intencional, o GEO foi dividido em 5 módulos básicos :

- 1.) Núcleo de Acesso a Registros : emula um sistema de Gerenciamento de Arquivos Lógicos, fornecendo suporte aos demais módulos. É composto pelo

Subsistema de Gerenciamento de Registros (SGR), Subsistema de Gerenciamento de Arquivos (SGA) e Subsistema de Gerenciamento de Memória (SGM). Através destes módulos é realizado todo o gerenciamento de memória secundária, simulação de uma memória "cache" para disco em memória e controle de acesso e proteção dos dados, através do gerenciador de transações.

2.) Subsistema Básico de Gerenciamento de Objetos : possibilita a manipulação de Objetos sem restrições de ordem semântica. Envolve o tratamento de Identificadores de Objetos, manipulação de Composições de Objetos, contexto e navegação na Base.

3.) Subsistema Básico de Gerenciamento de Relacionamentos : permite a manipulação de Relacionamentos (binários ou triplos) definidos pelo usuário. O GEO gerencia Relacionamentos Intrínsecos e Relacionamentos Extrínsecos: os intrínsecos são aqueles cuja semântica é conhecida pelo Gerenciador, tal como as Hierarquias de Generalização e de Composição, sendo gerenciados pelo Subsistema Básico de Gerenciamento de Objetos; os extrínsecos são aqueles cuja semântica é definida pelo usuário e são gerenciados por este módulo em questão.

4.) Subsistema Básico de Gerenciamento de Atributos : permite a manipulação de Atributos com características de Propriedades e Comentários, tanto para Objetos quanto para Relacionamentos.

5.) Gerenciador de Esquemas de Dados : consiste nas Primitivas de Acesso ao Esquema, as quais são ativadas pelos demais módulos.

Apenas os Subsistemas Básicos de Gerenciamento estão completamente implementados. Assim as operações devem ser feitas sem levar em conta os acessos mais elaborados ao MRO*, tal como acesso automático a Subtipos de Objetos e de Relacionamentos e operações de Sumarização automáticas. Correspondendo a cada Subsistema de Gerenciamento Básico está sendo realizada a implementação de um Subsistema de Gerenciamento Semântico. O conjunto de Subsistemas de Gerenciamento Básico é denominado GEO/MRO-Básico e o conjunto de Subsistemas de Gerenciamento Semântico é denominado GEO/MRO-Semântico.

O GEO/MRO-Básico efetua todas as operações de acesso efetivo na Base de

Dados, tal como criar ou eliminar Objetos, Relacionamentos, etc. Já o GEO/MRO-Semântico deverá realizar as mesmas operações, porém avaliando não apenas as regras sintáticas estáticas como é feito agora pelo GEO/MRO-Básico, mas também as regras semânticas, que dependem do ambiente instantâneo dos dados na Base de Dados. A estrutura geral do Gerenciador é mostrada na Figura 4.6.

Figura 4.6 - Estrutura do GEO

4.4. - Considerações Finais

Este capítulo descreveu o MRO* e a implementação de um gerenciador nele apoiado (GEO). O desenvolvimento deste modelo tem como objetivo fundamental suportar as necessidades de aplicações não convencionais de grupos de pesquisa, principalmente envolvendo seus projetos, permitindo que gerenciadores apoiados nesse modelo apresentem considerável eficiência e completeza.

Capítulo 5

ESQUEMAS SUPLEMENTARES

5.1. - Considerações Iniciais

Em Modelos Orientados a Objetos, a existência de um Esquema de Dados único torna o gerenciamento de acesso concorrente (sistema multi-usuário) extremamente difícil, uma vez que as alterações realizadas por um usuário devem ser ocultadas dos demais enquanto sua transação [CALÔNEGO_91] não se completa, ou seja, enquanto todas as alterações não forem realizadas.

Nas aplicações para projeto, geralmente as transações são de longa duração, portanto o mecanismo de ocultamento deve preocupar-se em não envolver o trabalho de outros usuários desnecessariamente, o que fatalmente retardaria suas operações ou as inviabilizaria temporariamente.

O ocultamento baseado em restrições de acesso definidas no Esquema é um mecanismo muito rígido, uma vez que admite somente restrições de acesso fixadas na definição do Esquema e portanto não são dinamicamente estipuladas e alteradas durante uma transação. Para contornar este problema, certos gerenciadores criam cópias da Base de Dados para os usuários realizarem suas transações, necessitando para isso de operações complexas que evitem conflitos ou inconsistências no confronto das alterações realizadas concorrentemente por dois ou mais usuários quando do armazenamento na Base de Dados.

Uma vez que uma Base de Dados está toda fundamentada em seu Esquema de Dados, toda a manipulação de dados realiza consultas ao Esquema. Deste modo, pode-se definir conceitualmente que o controle dinâmico de um Esquema de Dados é um

mecanismo indispensável para o controle de acesso à Base de Dados. Baseado no Modelo de Dados, o gerenciador pode simular o dinamismo do Esquema de Dados permitindo seu particionamento, cujas partes podem ser utilizadas alternativamente pelo gerenciador no reconhecimento de informações contidas na Base de Dados.

Seguindo este enfoque, este capítulo apresenta o conceito de Esquemas Suplementares¹, o qual proporciona às Bases de Dados apoiadas em Modelos de Dados Orientados a Objetos em geral e especificamente ao MRO*, os meios necessários para a realização do gerenciamento de acesso concorrente baseado na ativação dinâmica de partes do Esquema de Dados, de modo a minimizar conflitos e definições envolvendo múltiplos usuários.

5.2. - Definição

5.2.1. - Inter-relação entre Hierarquias de Composição e Hierarquias de Generalização

Múltiplas Hierarquias de Generalização podem ser formalmente designadas G_i , sendo que um elemento qualquer de uma hierarquia é designado $G_i(M)$. Da mesma forma, Hierarquias de Composição são denominadas C_j e possuem elementos $C_j(N)$. Os elementos supremos de qualquer Hierarquia de Generalização e Composição são respectivamente $G_i(0)$ e $C_j(0)$.

Em termos de características estruturais e funcionais, as Hierarquias de Generalização e Composição podem ser inter-relacionadas. Baseado nesta premissa podemos definir que existe uma Hierarquia de Generalização inicial G_0 , a partir da qual, Hierarquias de Composição C_j e outras Hierarquias de Generalização G_i podem surgir.

A inter-relação entre Hierarquias de Composição e Hierarquias de Generalização ocorre por meio dos supremos de cada uma, uma vez que $C_j(0)$ é um elemento de alguma Hierarquia de Generalização e $G_i(0)$ é um elemento de alguma Hierarquia de Composição. Assim, podemos dizer que a menos da Hierarquia de

¹ Esquemas Suplementares é o principal e mais importante conceito introduzido neste trabalho, uma vez que todas as realizações descritas nesta dissertação se baseiam neste conceito.

Generalização G_0 , cujo supremo é o Meta-tipo OBJETO - $G_0(0)$, os elementos supremos das demais Hierarquias de Generalização são elementos de alguma Hierarquia de Composição C_j e todo $C_j(0)$ é um elemento de uma Hierarquia de Generalização G_i ou G_0 .

A Hierarquia estabelecida pela inter-relação de todas as Hierarquias de Composição e Generalização definidas em uma modelagem, pode ser subdividida em Meta-Eschema e Esquema de dados, como mostra a Figura 5.1, onde estão representados por retângulos os subtipos de uma Hierarquia de Generalização e por círculos os componentes de uma Hierarquia de Composição.

Os elementos (ou tipos) de qualquer hierarquia G_i ou C_j podem estar associados com elementos de outras hierarquias, através de relacionamentos que não são de composição ou generalização.

Figura 5.1 - Inter-relação entre Generalizações e Composições

5.2.2. - Esquema Suplementar (conceito)

Partindo-se da dualidade estrutural entre a Base Intencional e a Base Extensional, o MRO⁷ pressupõe que todos os Tipos de Colônias, Tipos de Objetos, Tipos de Relacionamentos, Modalidades e Tipos de Atributos que constituem o Esquema de Dados, sejam armazenados em uma única Colônia denominada **Colônia Esquema** (veja seção 4.2.4). Esta colônia é consultada pelo Gerenciador de Esquemas quando alguma operação sobre informações da Base Extensional for requisitada.

Estendendo-se o conceito anterior, foi criado neste trabalho o conceito de **Colônia Esquema Suplementar** (ou simplesmente Esquema Suplementar) que pode ser definido da seguinte forma:

Conjunto de elementos (tipos, relacionamentos, atributos) resultante de uma modelagem, que aplica-se a um aspecto específico de uma aplicação e que pode ser seletivamente ativado no Esquema de Dados.

Com o suporte a Esquemas Suplementares pelo Gerenciador de Esquemas, uma Base Intencional pode possuir várias Colônias Esquema. Uma destas colônias, especialmente denominada **Colônia Esquema Inicial**, tem como função armazenar os Tipos que podem ser utilizados, por qualquer usuário da Base Extensional, na manipulação de informações gerais contidas nesta Base. As demais Colônias Esquema, denominadas **Colônias Esquema Suplementar**, armazenam as definições de Tipos destinadas a certos usuários ou aspectos de uma aplicação, uma vez que, no conteúdo de cada Esquema Suplementar existem informações específicas de uma área de conhecimento, que devem ser utilizadas apenas pelos usuários interessados.

Devido ao papel específico desempenhado pelos Esquemas Suplementares, são colocados individualmente sobre estas Colônias, acessos restritos às consultas, pois seus conteúdos destinam-se a determinados usuários que detêm o conhecimento e a necessidade destas informações para a realização de seus trabalhos. Para cada Colônia Esquema Suplementar que o usuário tiver acesso, ele poderá reconhecer e criar novas informações na Base Extensional.

5.2.3. - Particionamento

Sendo cada Colônia, por definição, um partição do conjunto de Objetos de uma Base (Intencional ou Extensional) e também porque as Colônias Esquema armazenam elementos (tipos, relacionamentos entre tipos e atributos de tipos) da Base Intencional, pode-se concluir que Colônias Esquema são partições da Base Intencional.

Baseado neste fato, o Esquema de Dados formado pela inter-relação das Hierarquias de Composição e Generalização estabelece a estrutura da Base Intencional, que pode ser particionada em qualquer ligação entre elementos de uma Hierarquia de Composição, como mostra o exemplo da Figura 5.2. Neste exemplo, apresentam-se dois desmembramentos da hierarquia original da Figura 5.1 (sem o Meta-Esquema e apenas com o Esquema), formando três partes assim constituídas:

Parte 1 - contendo Veículo, Fábrica, Pessoa (Projetista, Piloto), Sistema Elétrico, Sistema Mecânico, Sistema Hidráulico, Motor, Direção, Transmissão e Freio (Tambor, Disco);

Parte 2 - contendo Sist. Alimentação, Sist. Escapamento e Sist. Refrigeração;

Parte 3 - contendo Embreagem, Diferencial, Alavanca.

Para o MRO*, as três partes definidas acima podem especificar respectivamente a Colônia Esquema Inicial e duas Colônias Esquema Suplementar.

Uma vez que a tendência dos Modelos Orientados a Objetos em geral é manter os elementos de uma Hierarquia de Generalização agrupados em um mesmo conjunto (classe ou colônia), não se considera a possibilidade de desmembrar uma Hierarquia de Generalização.

A partição é realizada a critério do gerente/projetista com base nas características da aplicação e de sua utilização, assim criando-se partes que representam individualmente:

- um domínio de informações de um aspecto utilizado na aplicação ou;
- um domínio de informações de um grupo de projeto que trabalhará, concorrentemente ou não, para a finalização do projeto ou;
- um domínio de informações de um usuário específico;
- um domínio de informações de uma alternativa de projeto.

Figura 5.2 - Partição da Base Intencional

Independente do papel desempenhado por um Esquema Suplementar, podemos definir especificamente para o MRO* que um domínio deverá conter os seguintes elementos:

- a definição dos Tipos de Colônia que formam uma Hierarquia de Composição e os Tipos de Objetos que as habitam e/ou;
- a definição das Modalidades e Tipos de Relacionamentos associados aos Tipos de Objetos definidos e/ou;
- a definição dos Tipos de Atributos para os Tipos de Objetos ou Tipos de Relacionamentos.

Um Esquema Suplementar define de alguma forma, de acordo com seu papel, um elemento da modelagem, ou seja, cada Esquema Suplementar refina de acordo com algum aspecto, um Tipo de Objeto escolhido pelo gerente/projetista. Por exemplo, o Esquema Suplementar contendo os Tipos de Objetos: Embreagem, Diferencial e Alavanca, define ou refina o Tipo de Objeto Transmissão.

Para possibilitar que um Tipo de Objeto possa ter várias constituições, definimos que qualquer Tipo de Objeto em qualquer Colônia Esquema pode ter um ou mais Esquemas Suplementares que o especifica de diferentes formas através de diferentes elementos. O conjunto de todos Esquemas Suplementares que especificam

um mesmo Tipo de Objeto é chamado **Esquemateca** deste Tipo. Quando ocorrer uma instânciação de um Tipo de Objeto, o usuário poderá especificar qual Esquema Suplementar irá definir as instâncias que detalham o Objeto instânciado.

Na recuperação das informações de uma Base Extensional, os dados são reconhecidos pelo gerenciador quando seus Tipos são encontrados no Esquema de Dados. Portanto somente com o uso do mesmo Esquema Suplementar utilizado na definição das informações um usuário poderá reconhece-las e manipulá-las.

5.2.4. - Modelagem e Representação Gráfica

Os novos conceitos deste trabalho exigem a adição de elementos de representação gráfica, pois esta visualização se torna necessária para o programador implementar aplicativos que requisitem Esquemas Suplementares.

Cada conjunto de Esquemas Suplementares associados a um determinado Tipo de Objeto é modelado como uma Esquemateca. Para representá-la, criou-se o símbolo gráfico mostrado na Figura 5.3 para ser utilizado em DRO's. Cada Esquemateca tem um código correspondente, que o identifica em um documento denominado *Dicionário de Esquemas*, o qual contem os DRO's de todas as Colônias Esquemas modeladas, bem como outros documentos relacionados. O Apêndice A mostra os diagramas do Dicionário de Esquemas do exemplo seguido neste trabalho.

O MRO^{*} determina que toda Colônia (exceto a Colônia Global) deve ser constrita por um Objeto. Por definição, os Objetos que constingem as Colônias Esquema habitam a Colônia Global. A indicação dos Objetos que constingem os Esquemas Suplementares é extremamente importante para individualizar cada Colônia Esquema. Esta indicação é mostrada em um DRI da Colônia Global, utilizando-se o novo símbolo de representação para Colônia Esquema apresentado na Figura 5.4.

Cada Esquema Suplementar possui a identificação da Esquemateca a que pertence seguido da sua identificação individual dentro da Esquemateca.

Figura 5.3 - Representação de uma Esquemateca

Existindo a necessidade de representar o relacionamento entre dois Tipos de diferentes Colônias Esquema, deve-se utilizar um novo símbolo de representação apresentado na Figura 5.5. O retângulo colocado ao lado do Tipo que não pertence ao Esquema Suplementar do corrente DRO (Tipo Estrangeiro), pode especificar um: Esquema Suplementar, indicando que o relacionamento ocorre somente para o Esquema Suplementar descrito; ou uma Esquemateca, o que determina que o relacionamento ocorre para todos os Esquemas Suplementares, pertencentes a Esquemateca, que possuam o Tipo Estrangeiro.

5.3. - Esquema de Dados

5.3.1. - Conceito

Da forma definida neste trabalho, o Esquema de Dados ativo passa a ser o agrupamento da Colônia Esquema Inicial

com algumas Colônias Esquema Suplementar da Base Intencional. Este agrupamento é realizado para atender as necessidades dos usuários no acesso às informações, o que impõe ao Esquema de Dados ativo as seguintes características:

- O Esquema de Dados ativo torna-se **único e individual** para cada usuário da Base de Extensional e portanto o gerenciador deve controlar vários Esquemas de Dados ativos (semelhantes no conteúdo ou não) no acesso concorrente à Base de Dados.

Figura 5.4 - Representação de um Esquema Suplementar

Figura 5.5 - Representação de Tipo Estrangeiro

- O Esquemas de Dados ativo passa a ser um componente **transitório**, portanto passível de ser mantido em memória (veja capítulo 7). Além disso, sua existência é dinâmica uma vez que a sua definição (ativação de Esquemas Suplementares para agruparem-se à Colônia Esquema Inicial) é realizada somente para o acesso à Base de Dados durante uma sessão do usuário;
- O Esquema de Dados é **flexível**, uma vez que pode ser alterado facilmente através das operações definidas a seguir.

No restante deste trabalho será usado simplesmente o termo Esquema de Dados para designar o Esquema ativo e corrente de um usuário ou aplicativo.

5.3.2. - Operações de Manipulação

Uma vez que o Esquema de Dados deixa de ser um conjunto de dados fixos ou pouco alteráveis para ser um elemento passível de definição, passa a existir a necessidade de operações que proporcionem ao gerenciador o dinamismo na manipulação dos Esquemas Suplementares, atuando na composição dos Esquemas de Dados.

Para esta manipulação foram designadas as operações de REQUISIÇÃO e LIBERAÇÃO de Esquemas Suplementares. Cada qual tem sua função específica (descrita a seguir), porém ambas partem de um Esquema de Dados consistente e em uso, gerando um novo Esquema de Dados também consistente. Esta transformação é transparente para o usuário pois, a partir do momento em que o Esquema de Dados está completo suas informações são automaticamente reconhecidas e consultadas pelo gerenciador.

Tanto a Requisição quanto a Liberação são definidas para serem utilizadas diretamente por usuários que saibam suas necessidades e limitações em termos de Esquemas Suplementares, porém acredita-se que tais operações serão melhor exploradas quando estiverem internamente relacionadas a programas de aplicação que controlem sua execução.

5.3.2.1. - Requisição

Esta operação possibilita que um usuário inclua um Esquema Suplementar ao seu Esquema de Dados em uso, desde que o usuário tenha acesso à Colônia Esquema Suplementar requisitada. Este acesso é controlado pelo gerenciador baseado nos níveis de acesso de cada usuário em confronto com as permissões estabelecidas em cada Esquema Suplementar. A definição das restrições de acesso para cada Colônia Esquema, bem como sua criação e incorporação à Base de Dados, está condicionada às características da modelagem realizada pelos gerentes/projetistas da Base de Dados.

A requisição não permite que o Esquema Suplementar a ser incluído tenha conflito de informações com o Esquema de Dados, uma vez que tais conflitos são insolúveis se deixados a cargo do gerenciador. Esta incompatibilidade geralmente incidirá diretamente sobre um determinado Esquema Suplementar previamente incluído no Esquema, o qual poderá ser liberado para a atualização.

Outra requisição recusada pelo gerenciador ocorre quando certos relacionamentos introduzidos no Esquema de Dados não possuem seus elementos origem e destino definidos no próprio Esquema. Esta situação ocorre quando o Esquema Suplementar requisitado possui relacionamentos com outros Esquemas Suplementares não contidos previamente no Esquema de Dados.

A sintaxe do comando de Requisição é definida como:

REQUISITA ESQUEMA CONSTRITA POR <nome_objeto>;

A especificação, por parte do usuário, do nome de um Objeto determina a Colônia Esquema Suplementar que deseja-se incluir no Esquema de Dados.

5.3.2.2. - Liberação

Cada usuário pode utilizar os Esquemas Suplementares que lhe forem necessários e permitidos requisitar em determinado momento, para criar seu Esquema de Dados particular. No instante em que algum Esquema Suplementar não for mais necessário, o usuário poderá descartá-lo de seu Esquema de Dados utilizando a operação de Liberação.

Esta operação não permite que o Esquema de Dados resultante da liberação fique inconsistente devido à retirada de algum dado (Tipo) contido no Esquema Suplementar liberado, deixando algum relacionamento indefinido.

Quando constatada inconsistência no novo Esquema de Dados, a operação não modifica o Esquema de Dados original, uma vez que um Esquema de Dados inconsistente não é aceito pelo gerenciador.

A sintaxe do comando de Liberação é definida como:

```
LIBERA ESQUEMA CONSTRITA POR <nome_objeto>;
```

O nome do objeto é utilizado para especificar o Esquema Suplementar utilizado no Esquema de Dados corrente e que deseja-se liberar.

5.4. - Considerações Finais

A característica principal na estruturação da Base Intencional em Esquemas Suplementares é a definição e o relacionamento entre Colônias Esquema, segundo uma lógica que enquadra-se na própria concepção de projeto multi-disciplinar que a Base de Dados está suportando. Ou seja, mediante a divisão de um projeto em várias áreas ou sub-áreas de conhecimento, o Esquema de Dados poderá acompanhar esta partição e mesmo facilitar a implantação de partes suplementares (Esquemas Suplementares).

Deste modo, projetos onde existam partes que podem ser tratadas e projetadas (ou reprojatadas) separadamente como **subprojetos**, podem se beneficiar desta estratégia de estruturação do Esquema para permitir o desenvolvimento paralelo dos subprojetos ("Simultaneous Engineering" [BLACKBURN_90]) e assim explorar os aspectos de performance, tempo e tecnologia para reduzir custos.

Capítulo 6

LINGUAGEM DE ACESSO DO MRO* - LAMRO (EXTENSÃO)

6.1. - Considerações Iniciais

As pessoas envolvidas em operações na Base Intencional são classificadas em:

Gerentes/projetistas (ou Administradores): responsáveis pela definição e criação do Esquema Inicial e dos Esquemas Suplementares;

Programadores de Aplicativos: que conhecendo o papel do Esquema Suplementar, o utiliza para o desenvolvimento de aplicativos;

Usuários da Base Extensional: realizam apenas consulta às informações do Esquema de Dados, porém de forma transparente e portanto sem o conhecimento deste fato.

Para a tarefa de criação de um Esquema Suplementar por parte dos gerentes/projetistas (neste capítulo denominados usuários), foi desenvolvida uma linguagem baseada em comandos para a especificação dos elementos que compõem os Esquemas Suplementares e suas inter-relações.

Este conjunto de comandos foi proposto em linguagem anteriormente definida denominada LAMRO [PIZZIGATTI_92]. Sua definição possibilita a especificação de

uma única Colônia Esquema.

Para incorporar os novos conceitos propostos neste trabalho, foram realizadas alterações na sintaxe e na ação semântica de alguns comandos da LAMRO e a inclusão de novos comandos de definição de ambiente.

Este capítulo não descreve todos os comandos que fazem parte da LAMRO, mas somente os novos comandos de definição de ambiente, além do subconjunto de comandos de manipulação de Tipos e mesmo assim, somente aqueles que passaram por alguma alteração na sua concepção original.

6.2. - Especificações

Para possibilitar uma melhor definição e entendimento do subconjunto de comandos para a definição de uma Colônia Esquema, várias especificações foram desenvolvidas e são descritas a seguir.

6.2.1. - Classificação dos Comandos

Os comandos de definição de Colônias Esquema atuam sobre os elementos do MRO* tais como: Tipos de Objetos, Tipos de Colônias, Tipos de Relacionamentos, Tipos de Atributos, Modalidades e etc. Baseado na função ou finalidade a que se destinam, os comandos foram classificados da seguinte forma:

Inclusão - permite incluir a definição de novos Tipos em uma Colônia Esquema;

Eliminação - permite remover algum Tipo existente em uma Colônia Esquema;

Substituição - permite a alteração de algum Tipo em uma Colônia Esquema;

Estabelecimento de Ambiente - permite especificar a Colônia Esquema em que deseja-se manipular (incluir, eliminar ou substituir);

Mostrar Elemento - permite exibir dados sobre os Tipos de uma Colônia Esquema.

6.2.2. - Notação Utilizada

Na definição sintática dos comandos foi utilizada a seguinte notação:

As **Palavras Reservadas** aparecem em letras maiúsculas e todos os argumentos fornecidos pelo usuário em letras minúsculas;

< > delimitam um símbolo definido pelo usuário;

[] delimitam um conjunto de símbolos opcionais;

()^{*} delimitam um conjunto de símbolos que podem aparecer zero ou mais vezes;

()⁺ delimitam um conjunto de símbolos que podem aparecer uma ou mais vezes;

{ } delimitam um conjunto de símbolos dos quais ao menos um componente deve aparecer;

; finaliza um comando;

/* */ delimitam um comentário.

6.3. - Definição da Linguagem

Os comandos que se seguem somente poderão ser utilizados para definir Colônias Esquema que não estão sendo utilizadas para compor um Esquema de Dados. Caso alguma alteração deva ser feita em uma Colônia Esquema em uso, para esta deve ser criada uma cópia **Alternativa**.

6.3.1. - Inclusão

Os comandos de inclusão permitem que elementos possam ser inseridos em uma Colônia Esquema. De forma genérica, esta classe de comandos está estruturada da seguinte forma:

DEFINE <elemento> <argumentos> ;

Para efeito de esclarecimento, acompanhando cada comando de inclusão estão colocadas figuras que apresentam os elementos definidos pelo comando correspondente e um algoritmo bastante simplificado da ação semântica, que não descreve o tratamento de exceção.

6.3.1.1. - Tipo de Colônia

Através do comando descrito a seguir, é possível criar Tipos de Colônias em uma Colônia Esquema.

```
DEFINE COLONIA <tipo_colônia>
  CONSTRITA POR <tipo_objeto> ;
```

Esquematização:

Figura 6.1 - Relacionamento "Constrita por"

Ação Semântica:

- 1- Criar o Objeto <tipo_colônia>;
- 2- Procurar o Objeto <tipo_objeto>;
- 3- Se não encontrou
então faça (4);
- 4- Criar o Objeto <tipo_objeto>;
- 5- Criar o relacionamento CONTRITA POR entre os Objetos:
<tipo_colônia> e <tipo_objeto>;
- 6- Fim;

O Tipo de Objeto especificado para restringir o Tipo de Colônia em criação pode estar localizado, se ele existir, em qualquer Colônia Esquema. Portanto podemos generalizar esta situação assumindo que criar qualquer relacionamento entre Objetos de Colônias Esquema recae em um dos seguintes casos:

Caso 1 : O Objeto (Tipo) procurado para compor o relacionamento está definido na Colônia Esquema em construção ou;

Caso 2 : O Objeto (Tipo) procurado está definido em outra Colônia Esquema;

Determinar se o comando enquadra-se no Caso 1 ou no Caso 2, requer uma busca generalizada. Para isso a ação "*Procurar o Objeto <tipo_objeto>*" deve ser realizada de início na Colônia Esquema em construção e caso não tenha tido sucesso, então procurar em todas as Colônias Esquema existentes na Base Intencional.

A definição de relacionamentos entre Tipos de diferentes Colônias Esquema é permitido e em alguns casos necessário. Uma esquematização destes relacionamentos pode ser visto de maneira genérica e estilizada através da Figura 6.2b.

Figura 6.2 - a) Relacionamento entre Tipos na mesma Colônia Esquema
b) Relacionamento entre Tipos de diferentes Colônias Esquema

6.3.1.2. - Tipo de Objeto

Através do comando descrito a seguir, é possível criar Tipos de Objetos em uma Colônia Esquema.

```

DEFINE OBJETO (<tipo_objeto>)+
  HABITA <tipo_colônia>
  [ IDENTIFICADO POR      <modalidade_grau1>
                        <tipo_objeto_origem>
                        <tipo_objeto_destino> ] ;
  
```

Esquematização:

Figura 6.3 - Relacionamento "Habita"

Figura 6.4. - Relacionamento "Identificado por"

Ação Semântica:

- 1- Enquanto houver <tipo_objeto> no comando faça (2 até 12);
- 2- Pesquisar o Objeto <tipo_objeto>;
- 3- Se não encontrou
então faça (4);
- 4- Criar o Objeto <tipo_objeto>;
- 5- Procurar o Objeto <tipo_colônia>;
- 6- Se não encontrou
então faça (7);
- 7- Criar o Objeto <tipo_colônia>;
- 8- Criar o relacionamento HABITA entre os Objetos:
<tipo_objeto> e <tipo_colônia>;
- 9- Se fornecido, Procurar o Objeto (Tipo de Relac. Binário¹);
- 10- Se não encontrou
então faça (11);
- 11- Criar o Objeto (Tipo de Relac. Binário);
- 12- Criar o relacionamento IDENTIFICADO POR entre os Objetos:
<tipo_objeto> e Tipo de Relacionamento;
- 13- Fim;

A definição de um Tipo de Objeto pode especificar um Tipo de Colônia e/ou Tipo de Relacionamento existente em outra Colônia Esquema, o que nos reporta aos Casos 1 e 2 (anteriormente definidos) que determinam que qualquer operação "Procurar" deve ser realizada primeiro na Colônia Esquema que está sendo definida e caso necessário, nas outras Colônias Esquema, até que se encontre o elemento procurado ou até esgotar-se as Colônias Esquema existentes.

6.3.1.3. - Tipo de Relacionamento Binário

Através do comando descrito a seguir, é possível criar Tipos de Relacionamentos Binários.

¹ Um Tipo de Relacionamento Binário é especificado pelos Tipos de Objeto Origem e Destino que o utilizam e pela Modalidade Grau 1 que o compõe.

```

DEFINE RELACIONAMENTO <modalidade_grau1>
[ OPOSTO <modalidade_grau1_op> ]
( ORIGEM <tipo_objeto_origem>
  DESTINO <tipo_objeto_destino>
  [ MINIMO <número> ]           /* assume 0 */
  [ MAXIMO <número> ]           /* assume N */
  [ MINIMO OPOSTO <número> ]    /* assume 0 */
  [ MAXIMO OPOSTO <número> ]    /* assume N */
  [ SEGURANÇA SIM/NAO ] )+ ;    /* assume NAO */

```

Esquemática:

Figura 6.5. - Relacionamentos "Origem é" e "Destino é"

Figura 6.6. - Relacionamento "Composto por" (Rel. Binário)

Ação Semântica:

- 1- Procurar o Objeto <modalidade_grau1>;
- 2- Se não encontrou
então faça (3 até 5)
 - 3- Criar o Objeto <modalidade_grau1>;
 - 4- Criar o Objeto <modalidade_grau1_op>;
 - 5- Criar o relacionamento OPOSTO É entre os Objetos:
<modalidade_grau1> e <modalidade_grau1_op>;
- 6- Enquanto houver <tipo_objeto_origem> no comando faça (7 até 20);
 - 7- Procurar o <tipo_objeto_origem>;
 - 8- Se não encontrou
então operação inválida
 - 9- Procurar o <tipo_objeto_destino>;
 - 10- Se não encontrou
então operação inválida;
 - 11- Criar o Objeto (Tipo de Relac. Binário);
 - 12- Criar o relacionamento COMPOSTO POR entre os Objetos:
Tipo de Relac. Binário e <modalidade_grau1>;
 - 13- Criar o relacionamento ORIGEM É entre os Objetos: Tipo de
Relac. Binário e <tipo_objeto_origem>;
 - 14- Criar o relacionamento DESTINO É entre os Objetos: Tipo de
Relac. Binário e <tipo_objeto_destino>;
 - 15- Criar os atributos Multiplicidade Mínima e Máxima e Segurança
para o Objeto (Tipo de Relac. Binário);
 - 16- Criar o Objeto (Tipo de Relac. Binário) inverso;
 - 17- Criar o relacionamento COMPOSTO POR entre os Objetos:
Tipo de Relac. Binário inverso e <modalidade_grau1_op>;
 - 18- Criar o relacionamento ORIGEM É entre os Objetos: Tipo de
Relac. Binário inverso e <tipo_objeto_destino>;
 - 19- Criar o relacionamento DESTINO É entre os Objetos: Tipo de
Relac. Binário inverso e <tipo_objeto_origem>;
 - 20- Criar os atributos Multiplicidade Mínima e Máxima e Segurança
para o Objeto (Tipo de Relac. Binário) inverso;
- 21- Fim;

As condições que levaram a especificação dos Casos 1 e 2 novamente se repetem, uma vez que a definição de um Tipo de Relacionamento Binário pode especificar uma Modalidade Grau 1 e/ou Tipos de Objetos origem e destino definidos em outras Colônias Esquema e portanto o tratamento dado às operações de procura se repete igualmente.

6.3.1.4. - Tipo de Relacionamento Triplo

Através do comando descrito a seguir, é possível criar Tipos de Relacionamentos Triplos.

```

DEFINE TRIPLO <modalidade_grau2>
  MODALIDADE CONJUNTO <modalidade_grau1_1>
  MODALIDADE DETERMINA <modalidade_grau1_2>
  ( DIVISAO UM      <tipo_objeto_d1>
    DIVISAO DOIS    (<tipo_objeto_d2>)+
    DIVISAO TRES     <tipo_objeto_d3>
    [ MINIMO <número> ]           /* assume 0 */
    [ MAXIMO <número> ]           /* assume N */
    [ SEGURANÇA SIM/NAO ] )+ ;    /* assume NAO */
  
```

Esquematização:

Figura 6.7. - Relacionamentos "Divisão 1", "Divisão 2" e "Divisão 3"

Figura 6.8. - Relacionamento "Composto por" (Rel. Triplo)

Figura 6.9. - Relacionamentos "Composto Determina" e "Composto Conjunto"

Ação Semântica:

- 1- Procurar os Objetos <modalidades_grau1_1> e <modalidades_grau1_2>;
- 2- Se não foram encontrados
então operação inválida;
- 3- Criar o Objeto <modalidade_grau2>;
- 4- Criar o relacionamento COMPOSTO CONJUNTO entre os Objetos: <modalidade_grau2> e <modalidade_grau1_1>;
- 5- Criar o relacionamento COMPOSTO DETERMINA entre os Objetos: <modalidade_grau2> e <modalidade_grau1_2>;
- 6- Enquanto houver <tipo_objeto_d1> no comando faça (7 até 18);
 - 7- Procurar o <tipo_objeto_d1>;
 - 8- Se não encontrou
então operação inválida;
 - 9- Procurar o <tipo_objeto_d3> correspondente;
 - 10- Se não encontrou
então operação inválida;
 - 11- Enquanto houver <tipo_objeto_d2> no comando faça (12 até 15);
 - 12- Criar o Objeto (Tipo Relacionamento Triplo²);
 - 13- Procurar o <tipo_objeto_d2>;
 - 14- Se não encontrou
então operação inválida;
 - 15- Criar o relacionamento DIVISÃO 2 entre os Objetos: Tipo de Relac. Triplo e <tipo_objeto_d2>;
 - 16- Criar o relacionamento DIVISÃO 3 entre os Objetos: Tipo de Relac. Triplo e <tipo_objeto_d3>;
 - 17- Criar o relacionamento DIVISÃO 1 entre os Objetos: Tipo de Relac. Triplo e <tipo_objeto_d1>;
 - 18- Criar os atributos Multiplicidade Mínima e Máxima e Segurança para o Objeto (Tipo de Relac. Triplo);
- 19- Fim;

² Um Tipo de Relacionamento Triplo é especificado pelos Tipos de Objetos Divisão1, Divisão2 e Divisão3 que o utilizam e pela Modalidade Grau 2 que o compõe.

As Modalidades Grau 1 e os Tipos de Objetos especificados no comando anterior poderão, a critério do usuário da linguagem, pertencer a quaisquer Colônias Esquema. Deve-se portanto ressaltar que estes relacionamentos são permitidos desde que satisfaçam as necessidades da aplicação, pois nem o conceito de Esquemas Suplementares e muito menos a sua implementação impõem restrições neste sentido.

6.3.1.5. - Tipo de Atributo

Através do comando descrito a seguir, é possível criar Tipos de Atributos.

```

DEFINE <característica> <tipo_atributo>
  [ TIPO DE DADO <tipo_dado> ]
  [ VALOR[ES] (<valor>)+ ]
  ASSOCIADO [A]
  ( {    [ RELACIONAMENTO
                                <modalidade_grau1>
                                <tipo_objeto_origem>
                                <tipo_objeto_destino> ]
                                <tipo_objeto> ]
                                <modalidade_grau2>
                                <tipo_objeto_d1>
                                <tipo_objeto_d2>
                                <tipo_objeto_d3> ] }
    [ OBJETO
    [ TRIPLO
    [ MINIMO <número> ]
    [ MAXIMO <número> ] )+
    [ INICIALIZACAO ];
  
```

Esquematização:

Figura 6.10. - Relacionamento "Identifica"

Figura 6.11 - Relacionamento "Atribui"

Figura 6.12 - Relacionamento "Atributo de"

Ação Semântica:

- 1- Criar o Objeto <tipo_atributo> e seu atributo <característica>;
- 2- Se <característica> é Sinônimo
então faça (3 até 6);
 - 3- Enquanto houver <tipo_objeto> no comando faça (4 até 6);
 - 4- Procurar o <tipo_objeto>;
 - 5- Se não encontrou
então operação inválida;
 - 6- Criar o relacionamento IDENTIFICA entre os Objetos:
<tipo_atributo> e <tipo_objeto>;
- 7- Se <característica> não é Sinônimo
então faça (8 até 27);
 - 8- Enquanto houver <tipo_objeto> no comando faça (9 até 17);
 - 9- Procurar <tipo_objeto>;
 - 10- Se não encontrou
então operação inválida;

- 11- Criar o relacionamento ATRIBUTO DE entre os Objetos: <tipo_atributo> e <tipo_objeto>;
- 12- Criar os atributos Multiplicidade Mínima e Máxima no relacionamento ATRIBUTO DE;
- 13- Se a <caracteristica> é Propriedade então faça (14 ate 17);
 - 14- Criar o atributo <tipo_dado> no Objeto <tipo_atributo>;
 - 15 - Enquanto houver <valor> no comando faça (16 e 17)
 - 16- Criar Objeto <valor>;
 - 17- Criar relacionamento ATRIBUI entre os Objetos: <tipo_atributo> e <valor>;
- 18- Enquanto existirem Tipos de Relac. Binários no comando faça (19 até 22);
 - 19- Procurar o Tipo de Relac. Binário;
 - 20- Se não encontrou então operação inválida;
 - 21- Criar o relacionamento ATRIBUTO DE entre os Objetos: <tipo_atributo> e Tipo de Relac. Binário;
 - 22- Criar os atributos Multiplicidade Mínima e Máxima no relacionamento ATRIBUTO DE;
- 23- Enquanto existirem Tipos de Relac. Triplos no comando faça (24 até 27);
 - 24- Procurar o Tipo de Relac. Triplo;
 - 25- Se não encontrou então operação inválida;
 - 26- Criar o relacionamento ATRIBUTO DE entre os Objetos: <tipo_atributo> e Tipo de Relac. Triplo;
 - 27- Criar os atributos Multiplicidade Mínima e Máxima no relacionamento ATRIBUTO DE;
- 28- Fim;

De maneira idêntica aos comandos anteriores, qualquer referência a Tipos contidos na Colônia Esquema em construção ou em outras Colônias Esquema, será tratada da mesmo forma e com a mesma transparência pelo gerenciador.

6.3.1.6. - Subtipo

O conceito de Esquemas Suplementares determina que um Hierarquia de Generalização não pode ser particionada e seus componente colocados em dois ou mais Esquemas Suplementares. Assim, o comando de definição de Subtipos de Objetos e Relacionamentos não sofre nenhuma alteração e portanto não será colocado neste trabalho.

6.3.1.7. - Sumarização

Através do comando descrito a seguir, é possível criar Tipos Sumarizadores.

```
DEFINE SUMARIZACAO <sumarização>  
[NA] COLONIA <tipo_colônia> ;
```

Ação Semântica:

- 1- Criar o Objeto <sumarização>;
- 2- Criar o relacionamento intrínseco entre os Objetos: <sumarização > e <tipo_colônia>;
- 3- Fim;

Tendo criada uma Sumarização, lhe pode ser atribuído Tipos de Atributos que terão em sua cláusula INICIALIZAÇÃO, a indicação do papel desempenhado pela Sumarização. Tanto os Tipos de Atributos quanto o Tipo de Colônia declarada poderão estar definidos em Colônias Esquema distintas.

6.3.2. - Eliminação

Os comandos de eliminação permitem que elementos possam ser retirados de uma Colônia Esquema. De forma geral, esta classe de comandos se estrutura da seguinte maneira:

```
ELIMINA TIPO <elemento> <argumentos> ;
```

6.3.2.1. - Tipo de Colônia

Através do comando descrito a seguir, é possível remover Tipos de Colônias.

ELIMINA TIPO COLONIA <tipo_colônia> ;

Ação Semântica:

- 1- Procurar relacionamentos *HABITADA POR* entre o <tipo_colônia> e Tipos de Objetos;
- 2- Se não encontrou
então faça (3)
- 3- Remover o <tipo_colônia>;
- 4- Fim;

Para que este comando possa ser executado, nenhum *relac. HABITADA POR* (Figura 6.3) deve existir entre o Tipo de Colônia que deseja-se remover e os Tipos de Objetos da atual Colônia Esquema ou de qualquer outra Colônia Esquema.

6.3.2.2. - Tipo de Objeto

Através do comando descrito a seguir, é possível remover Tipos de Objetos.

ELIMINA TIPO OBJETO <tipo_objeto> ;

Ação Semântica:

- 1- Procurar relacionamentos *CONSTRINGE* entre o <tipo_objeto> e Tipos de Colônias;
- 2- Se encontrou
então operação inválida;
- 3- Procurar relacionamentos entre o <tipo_objeto> e Tipos de Relacionamentos Binários ou Triplos;
- 4- Se encontrou
então operação inválida;
- 5- Remover os atributos do Objeto <tipo_objeto>;
- 6- Remover o Objeto <tipo_objeto>;
- 7- Fim;

Para que este comando possa ser executado, nenhum relacionamento CONSTRINGE (Figura 6.1) deve existir entre o Tipo de Objeto que deseja-se remover e um Tipo de Colônia da atual Colônia Esquema ou de qualquer outra Colônia Esquema. O mesmo ocorre com os relacionamentos (Figura 6.5 e Figura 6.7) envolvendo o Tipo de Objeto com Tipos de Relacionamentos Binários ou Triplos.

6.3.2.3. - Tipo de Relacionamento Binário

Através do comando descrito a seguir, é possível remover um ou mais Tipos de Relacionamentos Binários.

```
ELIMINA TIPO RELACIONAMENTO <modalidade_grau1>
[ OPOSTO <modalidade_grau1_op> ]
( ORIGEM <tipo_objeto_origem>
  DESTINO <tipo_objeto_destino> ) * ;
```

Ação Semântica:

- 1- Procurar relacionamentos COMPÕE entre <modalidade_grau1> e outros Tipos de Relacionamento Binário;
- 2- Se não encontrou
então faça (3 até 6);
- 3- Procurar relacionamentos COMPÕE DETERMINA e COMPÕE CONJUNTO entre <modalidade_grau1> e Modalidades Grau 2;
- 4- Se não encontrou
então faça (5 e 6);
- 5- Remover o Objeto <modalidade_grau1>;
- 6- Remover o Objeto <modalidade_grau1_op>;
- 7- Se existe algum Tipo de Relacionamento no comando faça (8 até 11);
- 8- Enquanto existirem Tipos de Relac. Binários no comando faça (9 até 11);
- 9- Remover os relacionamentos envolvendo o Tipo de Relacionamento Binário indicado;
- 10- Remover os atributos do Tipo de Relac. Binário;
- 11- Remover o Tipo de Relacionamento Binário;
- 12- Se não existe nenhum Tipo de Relacionamento no comando faça (13);
- 13- Remover <modalidade_grau1> seu oposto e todos os Tipos de Rel. (Binários e Triplos) associados, bem como seus atributos;
- 14- Fim;

A Modalidade Grau 1 será removida quando: não estiver associada através do *relac. COMPÕE* (Figura 6.8) a nenhum outro Tipo de Relacionamento Binário e também não relacionar-se com nenhuma Modalidade Grau 2 através de relacionamentos *COMPÕE DETERMINA* ou *COMPÕE CONJUNTO* (Figura 6.9); ou então, quando o comando indicar a retirada de todos o Tipos de Relacionamentos envolvidos com a Modalidade Grau 1 indicada.

6.3.2.4. - Tipo de Relacionamento Triplo

Através do comando descrito a seguir, é possível remover um ou mais Tipos de Relacionamentos Triplos.

```
ELIMINA TIPO RELACIONAMENTO TRIPLO <modalidade_grau2>  
  ( DIVISAO UM   <tipo_objeto_d1>  
    DIVISAO DOIS <tipo_objeto_d2>  
    DIVISAO TRES <tipo_objeto_d3> )+ ;
```

Ação Semântica:

- 1- Procurar relacionamentos *COMPÕE* entre <modalidade_grau2> e outros Tipos de Relacionamentos Triplos;
- 2- Se não encontrou
então faça (3 e 4);
 - 3- Remover <modalidade_grau2>;
 - 4- Remover seus relacionamentos;
- 5- Se existe algum Tipo de Relacionamento no comando faça (6 até 9);
 - 6- Enquanto existirem Tipos de Relac. Triplo faça (7 até 9);
 - 7- Remover os relacionamentos envolvendo o Tipo de Relacionamento Triplo indicado;
 - 8- Remover os atributos do Tipo de Relacionamento Triplo;
 - 9- Remover o Tipo de Relacionamento Triplo;
- 10- Se não existe nenhum Tipo de Relacionamento no comando faça (11);
 - 11- Remover <modalidade_grau2> e todos os Tipos de Rel. Triplos associados, bem como seus atributos;
- 12- Fim;

A Modalidade Grau 2 será removida se não estiver associada através do *relac. COMPÕE* (Figura 6.8) a nenhum outro Tipo de Relacionamento Triplo ou então se o comando ordenar a remoção de todos os Tipos de Relacionamentos Triplos que utilizam a Modalidade Grau 2 indicada.

6.3.2.5. - Tipo de Atributo

Através do comando a seguir, é possível remover um ou mais Tipos de Atributos.

```
ELIMINA TIPO ATRIBUTO <tipo_atributo>
  [ ASSOCIADO [A]
    ( [ RELACIONAMENTO <modalidade_grau1>
      <tipo_objeto_origem>
      <tipo_objeto_destino> )* ]
    ( [ OBJETO <tipo_objeto>)* ]
    ( [ TRIPLO <modalidade_grau2>
      <tipo_objeto_d1> <tipo_objeto_d2>
      <Tipo_objeto_d3>)* ]
  ];
```

Ação Semântica:

- 1- Enquanto houver <tipo_objeto> no comando faça (2);
- 2- Remover os *relac.* envolvendo o <tipo_objeto> com o <tipo_atributo>;
- 3- Enquanto existirem Tipos de *Relac. Binários* no comando faça (4);
- 4- Remover os relacionamentos envolvendo o Tipos de *Relac. Binário* indicados com o <tipo_atributo>;
- 5- Enquanto existirem Tipos de *Relac. Triplos* no comando faça (6);
- 6- Remover os relacionamentos envolvendo o Tipos de *Relac. Triplo* indicados com o <tipo_atributo>;
- 7- Se não existirem *relac.* envolvendo o <tipo_atributo> então faça (8);
- 8- Remover o Objeto <tipo_atributo>;
- 9- Fim;

6.3.3. - Substituição

Normalmente a substituição pode ser executada na Base de Dados do MRO através de uma sequência de comandos classificados como eliminação e inclusão. No entanto a eliminação de algum elemento com o objetivo de substituição pode significar perda de grande quantidade de informações que não deveriam ser eliminadas. Por isso apenas alguns elementos do Modelo foram contemplados com esta classe de comandos. Entre os comandos de substituição existentes apenas um, definido a seguir, sofreu alteração em sua ação semântica.

6.3.3.1 - Tipo de Colônia

Pode-se substituir o Tipo de Objeto que restringe ou habita um determinado Tipo de Colônia através do seguinte comando:

```
SUBSTITUI COLONIA <tipo_colônia>  
    [ PARA CONSTRITA [POR] <tipo_objeto_constr> ]  
    [ PARA HABITADA [POR] (<tipo_objeto_habit>)+ ] ;
```

Ação Semântica:

- 1- Se indicado no comando, Procurar o novo <tipo_objeto_constr> para restringir o <tipo_colônia>;
- 2- Se encontrou
então faça (3);
- 3- Remover o antigo relacionamento CONSTRITA POR e criar o novo relacionamento CONSTRITA POR entre o <tipo_colônia> e <tipo_objeto_constr>;
- 4- Enquanto houver <tipo_objeto_habit> no comando faça (5 até 8)
 - 5- Remover os antigos habitantes de <tipo_colônia>;
 - 6- Procurar o <tipo_objeto_habit>;
 - 7- Se encontrou
então faça (8);
 - 8- Criar o relacionamento HABITADA POR entre o <tipo_colônia> e <tipo_objeto_habit>;
- 9- Fim;

Relembramos que qualquer ação "Procurar" será realizada primeiramente na Colônia Esquema em construção e caso a busca não tenha sucesso, percorre-se todas as outras Colônias Esquema existentes na Base Intencional até se encontrar o Tipo procurado ou até esgotar-se as Colônias Esquemas existentes.

6.3.4. - Estabelecimento de Ambiente

Para que as diversas operações possam ser realizadas, é necessário que o ambiente onde serão executadas seja antes estabelecido. Por ambiente se entende um conjunto de elementos da Base que devem estar em um estado pré-determinado, como por exemplo que Colônia Esquema está abertas para uso.

No caso de manipulação de Colônias Esquema, o preparo do ambiente leva em conta os níveis de permissão do usuário e o estado desta Colônia no momento (se permitida ou não a leitura e escrita).

Para que o usuário acesse uma Colônia Esquema, ele deve inicialmente criá-la e/ou abri-la através do comando:

```
CONSIDERA ESQUEMA  CONSTRIITA POR <nome_objeto>  
                    ESQUEMATIZA <nome_tipo_objeto> ;
```

O usuário desse comando deve entender que a partir da realização desta operação, seus próximos comandos irão abranger a Colônia Esquema explicitamente relacionada com o Objeto <nome_objeto> declarado, e definirão os elementos que refinam o Objeto <nome_tipo_objeto> especificado.

Ação Semântica:

- 1- Verificar se Colônia Esquema já existe;
- 2- Se existe
então faça (3);
- 3- Abrir a Colônia Esquema desejada;
- 4- Se não existe
então faça (5 até 7);
- 5- Criar a Colônia Esquema constricta pelo Objeto <nome_objeto>;
- 6- Criar o relacionamentos implícito (esquematiza) entre os Objetos:
<nome_objeto> e <nome_tipo_objeto>
- 7- Abrir a Colônia Esquema criada;
- 8- Fim;

Esquematização:

Figura 6.13. - Acesso às Colônias Esquema através de Objetos da Colônia Global

Este comando irá criar uma Colônia Esquema constrita pelo Objeto <nome_objeto> indicado, que será colocado na Colônia Global. Caso a Colônia Esquema já exista então será aberta para alterações.

Quando as definições ou alterações terminarem o usuário deverá indicar o fim dos trabalhos fechando a Colônia Esquema que estava aberta, através do comando:

FINALIZA ESQUEMA CONSTRITA POR <nome_objeto> ;

No caso da alteração de ambiente, a simples declaração de acesso a uma outra Colônia Esquema irá automaticamente promover a mudança de ambiente. Portanto, o comando **FINALIZA** pode ser dispensado, para agilizar o trabalho do usuário.

É importante ressaltar que um ambiente pode ser restaurado, ou seja, um usuário poderá voltar a uma mesma Colônia Esquema manipulada anteriormente e desta forma, poderá alterar seu conteúdo quantas vezes desejar, desde que a Colônia Esquema não esteja sendo usada na instânciação de informações.

6.3.5. - Mostra

Os comandos denominados Mostra foram criados para permitir ao usuário consultas rápidas sobre o conteúdo da Base naquele momento.

Desde que a necessidade básica, no caso de trabalhos com Esquemas Suplementares, é visualizar Tipos de elementos em diferentes Colônias Esquema tornando possível o seu reconhecimento para serem reutilizados, optou-se por incorporar aos comandos de Mostra a especificação da Colônia Esquema que deseja-se visualizar, não importando em que Colônia Esquema esteja-se trabalhando. Assim o comando de mostra se apresenta genericamente como:

MOSTRA <nome_objeto> TIPOS <argumentos>;

O Objeto especificado determina a Colônia Esquema que deseja-se visualizar, uma vez que trata-se do Objeto que restringe a Colônia Esquema desejada.

Para não delongarmos em explicação repetidas das ações semânticas dos comandos de Mostra, definimos abaixo a Ação Semântica genérica para todos o comandos desta classe.

Ação Semântica:

- 1- *Abrir a Colônia Esquema especificada por meio do Objeto fornecido;*
- 2- *MOSTRAR OS TIPOS PEDIDOS DE ACORDO COM OS ARGUMENTOS FORNECIDOS;*
- 3- *Fechar a Colônia Esquema visualizada;*
- 4- *Abrir a Colônia Esquema que estava-se trabalhando;*
- 5- *Fim;*

6.4. - Considerações Finais

A garantia de que a linguagem representa fielmente o MRO* reside no fato de que a construção sintática e semântica dos comandos foram baseadas no Meta-Esquema (Figura 4.5) do Modelo, especificado em partes através dos DRO's deste capítulo.

A extensão da LAMRO foi definida sempre com a preocupação de evitar inconsistências nas Colônias Esquema. Desta forma, o resultado de um conjunto programado de comandos da LAMRO produz uma Colônia Esquema consistente se analisada isoladamente, porém a compatibilidade com as outras Colônias Esquema não é garantida.

Em termos de implementação, este trabalho não teve a intenção de realizar a codificação destes comandos, uma vez que tal tarefa não traduz o principal objetivo deste trabalho que se resume em definir e validar o conceito de Esquemas Suplementares, além do que as alterações nos comandos da LAMRO não podiam ser realizadas enquanto os analisadores léxico, sintático e semântico não fossem concluídos [PIZZIGATTI_92].

Para ilustrar a utilização dos comandos da LAMRO na definição de Colônias Esquema, está colocado no Apêndice B deste trabalho o conjunto completo dos comandos que definem a Base Intencional para a modelagem simplificada apresentada no Apêndice A.

Capítulo 7

ASPECTOS DA IMPLEMENTAÇÃO

7.1. - Considerações Iniciais

Neste capítulo são abordados os aspectos da implementação realizada para a validação dos conceitos definidos neste trabalho.

A implementação utiliza primitivas do Gerenciador de Objetos (GEO) desenvolvido no Instituto de Ciências Matemáticas de São Carlos - USP, o qual suporta parte dos conceitos do MRO'.

A versão deste gerenciador, e portanto as próprias primitivas deste trabalho, foram desenvolvidas na linguagem de programação "C" e em microcomputadores da linha PC-XT, tendo como sistema operacional o MS-DOS.

7.2. - Esquema de Dados

Com o suporte a Esquemas de Dados individuais e particularizados para cada usuário, a antiga versão de Esquema de Dados (denominada Mono-Esquema), na qual existia somente uma Colônia Esquema contendo toda a Base Intencional, não perdeu sua função, uma vez que existem casos onde o conjunto de elementos que compõem o Esquema é bastante simples, portanto os usuários não necessitam das vantagens obtidas pela utilização de Esquemas Suplementares. Deste modo, decidiu-se manter as duas versões do GEO em funcionamento: a versão Mono-Esquema e a versão que utiliza Esquemas Suplementares denominada Multi-Esquema.

7.2.1. - GEO Mono-Eschema

Nesta versão o GEO reconhece o Esquema de Dados como sendo uma única Colônia Esquema contendo todas as definições de elementos da Base de Dados. Por ser única, esta Colônia Esquema não possui um nome específico, e todos os usuários a utilizam na manipulação da Base Extensional.

As rotinas de manipulação da Base de Dados consultam internamente o Esquema de Dados e assim estão acessando diretamente o arquivo da Base de Dados onde encontra-se a única cópia das informações do Esquema de Dados.

Quaisquer alterações feitas no Esquema de Dados serão imediatamente visíveis a todos os usuários ou aplicativos, uma vez que de maneira nenhuma o Esquema de Dados pode ser bloqueado para o acesso de algum usuário, pois impediria a realização de suas atividades.

Além do acesso imediato às correções realizadas no Esquema de Dados, outra vantagem da Versão Mono-Eschema está na depuração do Esquema, que pode ser facilmente realizada pelo gerente/projetista (manualmente) ou pelo gerenciador.

A desvantagem desta versão está principalmente no acesso lento ao Esquema de Dados, que conseqüentemente retarda a realização das operações sobre a Base Extensional. Isto se deve ao fato do Esquema de Dados estar no arquivo da Base de Dados e portanto, para toda consulta ao Esquema será feito um acesso ao disco.

7.2.2. - GEO Multi-Eschema

Desenvolvida neste trabalho, esta versão do Gerenciador de Esquemas do GEO pode reconhecer vários Esquemas de Dados servindo individualmente cada usuário.

Os Esquemas de Dados são individualizados pois são alocados na memória principal com acesso restrito ao usuário requisitante. Deste modo, é possível existirem vários Esquemas de Dados independentemente armazenados na memória quando houverem vários usuários concorrentemente utilizando a Base de Dados.

As consultas de um usuário ao Esquema serão dirigidas especificamente àquela área de memória principal reservada para o usuário armazenar seu Esquema de Dados corrente.

Apesar do Esquema individual determinar uma excelente método de proteção e de proporcionar uma velocidade de acesso maior devido às consultas ao Esquema

se realizarem em memória principal, esta versão oferece algumas desvantagens entre as quais:

Não Didática: esta versão é mais complexa de se aprender e ensinar pois é necessário larga experiência de modelagens e profundo conhecimento da aplicação para reconhecer e realizar o **particionamento**;

Preparo Lento: as modelagens desta versão requerem uma pesquisa profunda sobre a operabilidade da Base de Dados, ou seja, determinar entre outras coisas: quais são os usuários, como trabalham e como interagem entre si.

As estruturas de dados, que armazenam o Esquema na memória, retratam os conceitos do MRO* baseado na estrutura de armazenamento do Meta-Esquema de Dados. O Apêndice E deste trabalho apresenta as estruturas responsáveis pelo armazenamento dos Esquemas de Dados.

7.3. - Primitivas

Implementadas em linguagem "C", as principais primitivas projetadas neste trabalho podem ser classificadas segundo as operações que realizam: consulta, manipulação e verificação do Esquema de Dados.

Para possibilitar o transporte desta primitivas para outras máquinas, como por exemplo a SUN, optou-se pela utilização de uma linguagem C "padrão" que pudesse ser compilada e executada em outras máquinas apenas alterando-se poucos códigos.

7.3.1. - Consulta

As primitivas desta classe são utilizadas para consultar as informações contidas no Esquema de Dados (ou Meta-Esquema de Dados), tanto para a versão Mono-Esquema quanto para a versão Multi-Esquema.

A nível de interface, não existe nenhuma diferença entre as duas versões das rotinas de consulta. Apenas internamente, por meio de chaves de compilação (`#ifdef`),

as 31 rotinas de consulta são direcionadas para localizar o Esquema de Dados no arquivo da Base de Dados (versão Mono-Esquema) ou então em memória principal (versão Multi-Esquema).

Estas rotinas de consulta são utilizadas pelas primitivas do núcleo do GEO responsáveis pela manipulação de dados da Base, para determinar os Tipos que se opera, pois o reconhecimento do Tipo de um dado é fundamental para a sua manipulação.

7.3.1.1. - Classificação

A denominação das rotinas, classificadas na tabela da Figura 7.1, segue um padrão pré-estabelecido para facilitar seu entendimento e memorização. Na definição dos nomes as seguintes normas foram utilizadas:

- as três primeiras letras indicam qual a principal informação que a rotina devolve;
- as demais letras indicam o Tipo de elemento a ser consultado para a obtenção das informações desejadas.

As rotinas de consulta são as seguintes:

1.) **CarATR** (coditipo Tatr, pal2 *característica)

Descrição: Indica a característica de um Tipo de Atributo

.....

2.) **CodATR** (cadeia nome, coditipo *Tatr)

Descrição: Indica o código de um Tipo de Atributo

.....

3.) **CodCOL** (cadeia nome, coditipo *Tcol)

Descrição: Indica o código de um Tipo de Colônia

.....

4.) **CodMOD1** (cadeia nome, coditipo *Tmod1)

Descrição: Indica o código de uma Modalidade Grau 1

.....

5.) **CodMOD2** (cadeia nome, coditipo *Tmod2)

Descrição: Indica o código de uma Modalidade Grau 2

6.) **CodOBJ** (cadeia nome, coditipo *Tobj)

Descrição: Indica o código de um Tipo de Objeto

.....

7.) **CodSIN** (cadeia nome, coditipo *Tsin)

Descrição: Indica o código de um Tipo de Sinônimo

.....

8.) **CptMOD2** (coditipo Tmod2, coditipo *Tmod1_deter, coditipo *Tmod1_conj)

Descrição: Indica os componentes de uma Modalidade Grau 2

.....

9.) **CtgCOL** (coditipo Tcol, coditipo *Tobj)

Descrição: Indica o Tipo de Objeto que restringe um Tipo de Colônia

.....

10.) **DadPRO** (coditipo Tpro, pal2 *tipodado)

Descrição: Indica o tipo de dado de um Tipo de Propriedade

.....

11.) **HabOBJ** (coditipo Tobj, coditipo *Tcol)

Descrição: Indica o Tipo de Colônia habitada por um Tipo de Objeto

.....

12.) **IdfBIN** (coditipo Tbin, coditipo *Tobj)

Descrição: Indica o Tipo de Objeto identificado pelo Tipo de Relac. Binário

.....

13.) **IdfOBJ** (coditipo Tobj, coditipo *Tbin)

Descrição: Indica o Tipo de Relac. Binário que identifica o Tipo de Objeto

.....

14.) **IdxCOL** (coditipo Tcol, pal2 *indice)

Descrição: Indica o índice de um Tipo de Colônia (-rotina interna-)

.....

15.) **MulAOB** (coditipo Tobj, coditipo Tatr, pal2 *min, pal2 *max)

Descrição: Indica as multipl. de um Tipo de Atributo associado a um Tipo de Objeto

.....

16.) **MulARB** (coditipo Tbin, coditipo Tatr, pal2 *min, pal2 *max)

Descrição: Indica as multipl. de um Tipo de Atr. de um Tipo de Relac. Binário

.....

17.) **MulART** (coditipo Ttri, coditipo Tatr, pal2 *min, pal2 *max)

Descrição: Indica as multipl. de um Tipo de Atr. de um Tipo de Relac. Triplo

.....

18.) **MulBIN** (coditipo Tbin, pal2 *min, pal2 *max)

Descrição: Indica as multiplicidades de um Tipo de Relacionamento Binário

19.) **MulSIN** (coditipo Tobj, coditipo Tsin, pal2 *min, pal2 *max)

Descrição: Indica as multiplicidades de um Tipo de Sinônimo

20.) **MulTRI** (coditipo Ttri, pal2 *min, pal2 *max)

Descrição: Indica as multiplicidades de um Tipo de Relacionamento Triplo

21.) **NomATR** (coditipo Tatr, cadeia nome)

Descrição: Indica o nome do Tipo de Atributo

22.) **NomCOL** (coditipo Tcol, cadeia nome)

Descrição: Indica o nome do Tipo de Colônia

23.) **NomMOD1** (coditipo Tmod1, cadeia nome)

Descrição: Indica o nome da Modalidade Grau 1

24.) **NomMOD2** (coditipo Tmod2, cadeia nome)

Descrição: Indica o nome da Modalidade Grau 2

25.) **NomOBJ** (coditipo Tobj, cadeia nome)

Descrição: Indica o nome do Tipo de Objeto

26.) **NomSIN** (coditipo Tsin, cadeia nome)

Descrição: Indica o nome do Tipo de Sinônimo

27.) **ObjBIN** (coditipo Torigem, coditipo Tmod1, coditipo *Tdestino)

Descrição: Indica o Tipo de Objeto destino de um Tipo de Relacionamento Binário

28.) **ObjTRI** (coditipo Ttri, coditipo Tobj_D2)

Descrição: Indica o Tipo de Objeto (divisão 2) de um Tipo de Relacionamento Triplo

29.) **OpsMOD1** (coditipo Tmod1, coditipo *Toposto)

Descrição: Indica a Modalidade oposta de uma Modalidade Grau 1

30.) **RelBIN** (coditipo Torigem, coditipo Tmod1, coditipo Tdestino, coditipo *Tbin)

Descrição: Indica o Tipo Relacionamento Binário

31.) **RelTRI** (coditipo Tobj_D1, coditipo Tmod2, coditipo Tobj_D3, coditipo *Ttri)

Descrição: Indica o Tipo Relacionamento Triplo

Uma descrição completa das rotinas de consulta da figura 7.1 pode ser encontrada no Apêndice C.

	Colônia	Objeto	Modalidade grau 1	Modalidade grau 2	Relac. Binário	Relac. Triplo	Atributo	Sinônimo
Obter Nome	NomCOL	NomOBJ	NomMOD1	NomMOD2			NomATR	NomSIN
Obter Código	CodCOL	CodOBJ	CodMOD1	CodMOD2			CodATR	CodSIN
Obter Multiplicidades					MulBIN	MulTRI	MulAOB MulARB MulART	MulSIN
Obter dados da sintaxe	OtgCOL	HabOBJ IdiOBJ	OpsMOD1	CptMOD2	IdfBIN ObjBIN RefBIN	ObjTRI RefTRI	CarATR DadPRO	
Obter dados internos	IdxCOL							

Figura 7.1 - Rotinas de consulta ao Esquema de Dados.

7.3.1.2. - Fluxo de Informação

Baseado na dualidade estrutural das Bases Intencional e Extensional que formam a Base de Dados, as rotinas de manipulação de dados podem operar sobre qualquer uma das Bases, uma vez que todos os dados são objetos. Assim, as rotinas de consulta devem tanto fornecer informações do Esquema de Dados quando se deseja manipular a Base Extensional, quanto fornecer dados do Meta-Esquema de Dados quando se quer manipular a Base Intencional.

O diagrama da Figura 7.2 apresenta o fluxo de informações existente quando uma rotina de manipulação é executada na versão Mono-Esquema, e o diagrama da Figura 7.3 apresenta o fluxo de informações quando uma rotina de manipulação é executada na versão Multi-Esquema. Nas figuras estão indicados pelas letras **E** e **D** os fluxos das informações para uma consulta ao Meta-Esquema ou então ao Esquema respectivamente. A consulta ao Meta-Esquema é necessária em alguns casos para reconhecer o Meta-tipo de um Tipo contido no Esquema de Dados.

Os fluxos indicados pela letra **E** representam a sequência que ocorre quando alguma operação de manipulação de dados da Base Intencional é requerida pelo gerente/projetista. Cada um dos cinco fluxos pode ser resumidamente descrito da seguinte forma:

1E : Alguma operação de manipulação de dados de uma Colônia Esquema requer uma rotina de consulta para a sua execução;

2E : As informações pedidas são pesquisadas na estrutura em memória principal contendo o Meta-Esquema de Dados;

3E : O resultado da consulta é retornado para a rotina de consulta;

4E : O mesmo resultado de **3E** é retornado para a operação de manipulação requisitante;

5E : Com as informações necessárias à sua disposição, a operação de manipulação executa sua função na Base Intencional.

Figura 7.2 - Diagrama dos fluxos de informações na versão Mono-Esquema

Os fluxos indicados pela letra **D** representam a sequência que ocorre quando alguma operação de manipulação de dados da Base Extensional é requerida pelo usuário. Cada um dos cinco fluxos pode ser resumidamente descrito da seguinte forma:

1D : Alguma operação de manipulação de dados de uma Colônia de Objetos requer uma rotina de consulta para a sua execução;

2D : As informações pedidas são pesquisadas no Esquema de Dados;

3D : O resultado da consulta é retornado a rotina de consulta;

4D : O mesmo resultado de 3D é retornado para a operação de manipulação requisitante;

5D : Com as informações necessárias à sua disposição, a operação de manipulação executa sua função na Base Extensional.

Figura 7.3 - Diagrama dos fluxos de informações na versão Multi-Eschema

As operações de manipulação de dados não distinguem sobre "o que" estão agindo, pois todos os dados são percebidos como Objetos, Relacionamentos ou Atributos, não sendo diferenciados como elementos da Base Intencional ou da Base Extensional. Esta percepção é tarefa da rotina de consulta, que reconhece seu papel naquele instante através dos indicadores de ambiente, para saber se deve consultar o Meta-Eschema de Dados ou o Esquema de Dados.

A diferença visível entre os dois diagramas ocorre porque o Esquema de Dados é a própria Base Intencional na versão Mono-Eschema enquanto que na versão Multi-Eschema, o Esquema de Dados é uma estrutura em memória principal que contém uma cópia de partes da Base Intencional (Esquemas Suplementares) e é somente utilizado para consulta.

7.3.2. - Manipulação

Criadas exclusivamente para a versão Multi-Esquema, as rotinas de manipulação foram projetadas para operar sobre Esquemas Suplementares e indiretamente sobre o Esquema de Dados.

As rotinas de manipulação estão disponíveis para a utilização a nível de programadores de aplicativos, ou seja, podem ser acionadas diretamente pelos programadores desde que estes tenham permissão de utilizá-las.

As rotinas de manipulação são as seguintes:

1.) **G_INICIALIZA_ESQUEMA ()**

Descrição: Inicializa o Vetor de Esquemas Ativos (veja Apêndice E)

2.) **G_REQUISITA_ESQUEMA (codiobjeto codigo_objeto)**

Descrição: Requisita o Esquema Suplementar constricto pelo Objeto indicado

3.) **G_LIBERA_ESQUEMA (codiobjeto codigo_objeto)**

Descrição: Libera o Esquema Suplementar constricto pelo Objeto indicado

Para as rotinas 2 e 3 existem comandos correspondentes (capítulo 5) para que os usuários da linguagem LAMRO possam manipular seus Esquemas de Dados. A rotina 1 é executada internamente pelo gerenciador, quando o usuário abre os arquivos da Base de Dados. A definição completa destas rotinas pode ser encontrada no Apêndice D deste trabalho.

7.3.3. - Verificação

As rotinas de verificação são ativadas pelos comandos de requisição ou liberação de Esquemas Suplementares para avaliar o conteúdo da Colônia Esquema e validar a utilização do Esquema de Dados resultante da operação, uma vez que não permite-se que o gerenciador utilize um Esquema de Dados inconsistente para manipular a Base Extencional.

Do estudo realizado neste trabalho encontraram-se vários tipos de inconsistências que podem ocorrer no Esquema de Dados devido a má definição de

uma Colônia Esquema ou má manipulação de uma Colônia Esquema consistente.

Para avaliar os casos freqüentes de inconsistências foram implementadas rotinas para duas classes de verificações: ISOLADAS, para avaliar inconsistências que ocorrem em um determinado Esquema Suplementar; e em CONTEXTO, para avaliar inconsistências que ocorrem no Esquema de Dados como um todo.

As verificações isoladas (ou **Verificação Sintática Isolada**) são realizadas no momento que um Esquema Suplementar é requisitado, pois deve-se pesquisar todos os relacionamentos existentes entre seus Tipos de modo a avaliar a existência das seguintes associações e atributos obrigatórios :

- Todo Tipo de Colônia deve ser constrito por um Tipo de Objeto;
- Todo Tipo de Objeto deve habitar um único Tipo de Colônia;
- Toda Modalidade Grau 1 deve ter uma oposta;
- Todo Tipo de Relacionamento Binário deve ser composto por uma Modalidade Grau 1 e estar associado a um Tipo de Objeto Origem e um Tipo de Objeto Destino;
- Toda Modalidade Grau 2 deve ser composta por duas Modalidades Grau 1;
- Todo Tipo de Relac. Triplo deve ser composto por uma Modalidade Grau 2 e estar associado a três Tipos de Objetos (Divisão 1, Divisão 2 e Divisão 3);
- Todo Tipo de Atributo deve possuir uma Característica.

Desde que a verificação isolada do Esquema Suplementar não tenha encontrado falhas na sua definição, o gerenciador insere o Esquema Suplementar no Esquema de Dados e realiza, no próprio Esquema, as seguintes verificações em contexto:

- Os Objetos origem e destino de qualquer relacionamento encontrado no Esquema de Dados devem estar definidos no próprio Esquema, por exemplo, qualquer Tipo de Objeto do Esquema deve habitar um Tipo de Colônia também definido no Esquema (**Verificação Sintática em Contexto**);

- Não devem existir repetições de dados no Esquema. Esta redundância pode ser simplesmente de nome (**Verificação de Nomes em Contexto**) ou então de relacionamentos (**Verificação de Semântica em Contexto**) como por exemplo, duas Modalidades Grau 2 serem compostas pelas mesmas Modal. Grau 1.

As rotinas de verificação são as seguintes:

1.) **VER_SINT_ISOLADO** (codiobjeto codigo_objeto)

Descrição: Realiza a Verificação Sintática Isolada para o Esquema Suplementar constrito pelo Objeto indicado

2.) **VER_SINT_CONTEXTO** ()

Descrição: Realiza a Verificação Sintática em Contexto do Esquema de Dados corrente

3.) **VER_REDUN_NOME_CONTEXTO** ()

Descrição: Realiza a Verificação de Nomes em Contexto do Esquema de Dados corrente

4.) **VER_REDUN_SEM_CONTEXTO** ()

Descrição: Realiza a Verificação de Semântica em Contexto do Esquema de Dados corrente

Se a Verificação Sintática Isolada detectar alguma inconsistência no Esquema Suplementar que está sendo requisitado, então a operação de requisição não se realiza. O mesmo ocorre com as verificações em contexto, que momentaneamente alteram o Esquema de Dados para realizar suas verificações, porém se algum erro for encontrado, o Esquema de Dados volta ao seu estado original antes do pedido de requisição. Para qualquer falha na pedido de requisição de um Esquema Suplementar o usuário receberá uma indicação de erro especificando o tipo de inconsistência encontrado.

Para as operações de liberação de Esquemas Suplementares são realizadas apenas as verificações em contexto, uma vez que é desnecessário verificar isoladamente um Esquema Suplementar que se está removendo do Esquema de Dados. Caso alguma verificação encontre inconsistências no novo Esquema de Dados, então o Esquema volta ao seu estado original.

7.4. - Considerações Finais

Este trabalho de implementação se iniciou com a programação da versão Mono-Eschema do gerenciador, para que outros trabalhos de pesquisas pudessem prosseguir em suas implementações. Após a aprovação da versão Mono-Eschema, iniciou-se o projeto e finalmente a implementação do GEO na versão Multi-Eschema.

Como campo de testes das primitivas implementadas foi utilizado um editor experimental denominado GEO_INC, no qual todos os trabalhos envolvendo Bases de Dados do GEO são inicialmente implantados e testados antes de realmente comporem o núcleo do gerenciador e se tornarem disponíveis para a utilização.

Capítulo 8

CONCLUSÕES E FUTURAS PESQUISAS

8.1. - Considerações Iniciais

Em linhas gerais, as etapas de criação de um Esquema de Dados [BATINI_86] se assemelham as fases do ciclo de vida clássico [PRESSMAM_87] de desenvolvimento em cascata ("waterfall model"), o que conceitualmente leva-se a entender a "Engenharia de Esquema de Dados" como sendo composta das seguintes fases:

- **Definição do Problema:** delimitar dentro de um ambiente, os elementos que serão suportados pela Base de Dados. No caso de uma empresa, esta tarefa, em termos muito simplificados, direcionaria a atenção dos gerentes/projetistas para departamentos, seções ou subseções em que exista interesse em informatizar por meio de uma Base de Dados;
- **Análise:** descrever claramente, de acordo com as necessidades dos usuários, os processos que serão automatizados pela Gerenciador da Base de Dados;
- **Especificação:** consiste em definir, segundo o documento produzido na Análise, quais e como são os elementos que formarão o Esquema de Dados;

- **Projeto:** corresponde a representar o Esquema de Dados de acordo com os conceitos do Modelo de Dados seguido pelos gerentes/projetistas. Entre os documentos produzidos nesta fase está o Dicionário de Esquemas;
- **Codificação:** trata-se do mapeamento e implantação do Esquema de Dados projetado na fase anterior, segundo as particularidades e ferramentas disponíveis no Gerenciador de Base de Dados adotado;
- **Testes:** consiste da Verificação de integridade e consistência e Validação do Esquema de Dados implantado;
- **Manutenção:** são realizadas nesta fase as eventuais alterações no Esquema de Dados, de acordo com as novas necessidades dos usuários;
- **Documentação:** esta fase, em particular, acompanha o andamento de todas as etapas anteriores, registrando em documentos os resultados obtidos.

Baseado neste trabalho, as etapas anteriormente colocadas passam a ter como objetivo final produzir não apenas um Esquema único e monolítico, mas alternativamente Esquemas Suplementares que formarão dinamicamente Esquemas de Dados individualizados para cada usuário.

Aplicando-se as etapas de desenvolvimento sobre Esquemas Suplementares, abrem-se inúmeras opções para a pesquisa. Este capítulo tem por objetivo concluir este trabalho e motivar o estudo destas opções por meio da enumeração de uma série de propostas de futuras pesquisas e dos caminhos seguidos e não seguidos por este trabalho.

8.2. - Decisões de Projeto

8.2.1. - Alteração no Conceito de Controle de Acesso

Para levar adiante este projeto, teve-se como principal barreira o conceito de proteção de acesso a Colônias [FERREIRA_91]. Nele está determinado que uma

Colônia no estado de R/O ("Read Only") não pode ser alterada. No caso de Colônias Esquema o seu estado R/O é decorrente do seu uso na definição da Base Extensional ou de parte dela. Uma Colônia Esquema no estado R/O não pode ser alterada, porém esta proteção impede que Colônias Suplementares inseridas na Base Intencional possam ter seus elementos relacionados com Colônias Esquema em utilização na Base.

Uma vez analisado o problema, ponderou-se pela alteração do conceito de R/O para que determine: os elementos contidos em uma Colônia R/O não podem ser removidos ou alterados em suas características, porém relacionamentos entre elementos de duas Colônias podem ser inseridos ou removidos dependendo do estado das duas Colônias envolvidas.

A Figura 8.1 apresenta as situações onde são permitidas as manipulações de relacionamentos entre duas Colônias Esquema que apresentam-se nos estados "Read Only" ou "Read Write".

	Colônia Esquema R/O	Colônia Esquema R/W
Colônia Esquema R/O	não permitido	permitido
Colônia Esquema R/W	permitido	permitido

Figura 8.1. - Permissões de manipulação de Relac. entre Colônias Esquema.

A mudança no conceito de Colônia R/O foi realizada depois de um estudo dos efeitos que esta alteração poderia causar no funcionamento do gerenciador de acesso do GEO. Pelas considerações estudadas, a nova definição de Colônia R/O é válida porque considera-se o relacionamento entre Colônias como uma informação independente e merecedora de um tratamento individualizado.

8.2.2. - Alternativas de Projeto

Durante a definição de Esquemas Suplementares decidiu-se por relacioná-los somente a Tipos de Objetos, pois com as aplicações imaginadas para este conceito não encontrou-se fundamentos para que Esquemas Suplementares definam Tipos de Atributos, Tipos de Relacionamentos ou Tipos de Colônias. Porém, não está descartada esta possibilidade de utilização, desde que não entre em conflito com as determinações estabelecidas no capítulo 5.

Na fase de implementação definiu-se que as requisições (e liberações) de Esquemas Suplementares seriam explicitamente acionadas pelo usuário ou pelo programa de aplicação, porém acredita-se na existência de situações em que o próprio gerenciador pode reconhecer e decidir pela requisição ou liberação sem qualquer intervenção do usuário. Esta manipulação implícita de Esquemas Suplementares retira do usuário a tarefa de reconhecer quais Esquema Suplementares podem ser requisitados por ele e quando um Esquema Suplementar é necessário ou desnecessário.

8.3. - Sugestões de Futuras Pesquisas

Além dos estudos das alternativas de trabalho que não foram implantadas, outros tópicos podem ser pesquisados. Estes tópicos podem ser enquadrados dentro das fases de desenvolvimento de um Esquema Suplementar.

8.3.1. - Integração de Projetos

As metodologias usuais de desenvolvimento de Esquemas consideram a Integração de Esquemas [NAVATHE_86] uma etapa importante realizada entre as fases de Projeto e Codificação. Este trabalho preconiza que essa Integração deva ser uma compatibilização dos Esquemas Suplementares projetados, sendo que os critérios de compatibilização levam em conta os mesmos critérios de Integração.

No projeto da Base Intencional do GEO Multi-Esquema, se as fases de desenvolvimento definirem um Esquema monolítico, deve-se realizar o **particionamento**, enquanto que no desenvolvimento de Esquemas Suplementares separadamente deve ser realizado a **compatibilização**.

A norma para o particionamento de um Esquema único em vários Esquemas Suplementares foi claramente definida no capítulo 5, onde determinou-se que o particionamento deve ocorrer somente nas interligações das hierarquias de composição, baseado em critério do gerente/projetista. Porém as normas para a compatibilização de Esquemas Suplementares ou Integração de Projetos baseados no MRO* devem ser objetos de estudos futuramente.

8.3.2. - Codificação

Adaptar uma Base Extencional às mudanças ocorridas no Esquema de Dados único, geralmente leva a muitas alterações nas informações da Base Extencional. As técnicas descritas na seção 2.3.2.3, e atualmente utilizadas para a adaptação da Base Extencional à evolução do Esquema de Dados, apresentam como limitante a velocidade de execução, porém o principal problema é a perda ou indefinição das informações anteriores (antigas) da Base Extencional, uma vez que o Esquema de Dados antigo perde sua função.

As Versões são comumente aplicadas às informações da Base Extencional para armazenar histórico de evolução ou auxiliar no controle de concorrência (seção 2.3.3). Porém, desde que o MRO⁷ suporta Versões de Colônias, podemos aplicar Versões sobre as Colônias Esquema e assim manter na própria Base Intencional todas as possibilidades de formação do Esquema de Dados.

Em termos de Evolução do Esquema, novas propostas de Colônias Esquema podem facilmente ser criadas como Versões e os usuários poderão requisitar e liberar as Versões de Colônias Esquema que desejarem, sem interferir com os Esquemas de outros usuários. Para isto, em futuras extensões deste trabalho serão incorporados ao GEO as rotinas de gerenciamento de **Versões de Colônias** e desta forma, alterações em Colônias Esquema poderão ser realizadas apenas em **Alternativas**. Como sugestão, a criação de Alternativas a partir de uma Colônia Esquema pode ocorrer através do comando:

INSTANCIA ESQUEMA

CONSTRITA POR <nome_objeto>
PARA <nome_da_variante> ;

Para que os comandos anteriormente definidos: **CONSIDERA**, **FINALIZA** e mesmo o **REQUISITA** e o **LIBERA** passem a reconhecer **Variantes** de Colônias Esquema, basta incorporar às suas sintaxes o <nome_da_variante>. Por exemplo:

REQUISITA ESQUEMA

CONSTRITA POR <nome_objeto>
PARA <nome_da_variante> ;

Criar uma Versão a partir de uma Alternativa pode ser função do comando:

CONGELA ESQUEMA

CONSTRITA POR <nome_objeto>
PARA <nome_da_variante> ;

Além da construção dos analisadores léxico, sintático e semântico destes comandos sugeridos e de outros comandos apresentados no capítulo 6, poderá ser implementado um editor de Esquemas Suplementares baseado em Menus e Janelas (denominado GEÓGRAFO), para ser utilizado por gerentes/projetistas que necessitam de um ambiente mais amigável.

8.3.3. - Testes

Do modo como foram realizadas as verificações descritas no capítulo 7, toda requisição de uma Colônia Esquema realiza as avaliações de integridade (Verificação Sintática Isolada). Para evitar esta repetição de avaliações desnecessária, uma possível extensão deste trabalho será a realização da Verificação Sintática Isolada uma única vez, após a definição completa do conteúdo da Colônia Esquema e o pedido para torná-la uma Versão. Caso a Verificação seja bem sucedida, a operação indicará internamente a verificação realizada registrando a Colônia Esquema no estado R/O.

Outra possibilidade de estudo é a procura exaustiva de todas as verificações possíveis e necessárias, de modo que suas execuções possam assegurar indubitavelmente a integridade de um Colônia Esquema. Entre algumas verificações encontradas e não implementadas estão:

- a análise isolada e em contexto dos componentes de todos os Tipos de Relacionamentos Triplo, ou seja, a verificação de consistência de todos os Tipos de Relacionamentos Binários envolvidos em cada Tipo de Relac. Triplo;
- a análise isolada e em contexto do dígrafo formado pelas associações entre Tipos de Colônias para determinar a inexistência de ciclos, ou seja, determinar se através dos Tipos de Objetos que habitam e constroem Tipos de Colônias existe algum Tipo de Objeto que constrói algum Tipo de Colônia que o contém (direta ou indiretamente).

Estas e outras análises, que poderão ser encontradas a partir de estudos de casos reais de modelagem, devem ser implementadas futuramente.

8.3.4. - Documentação

Para efeito de padronização da documentação sobre os Esquemas Suplementares, deve-se iniciar um estudo sobre este aspecto, envolvendo desde os diagramas (DRO, DRI e DHC) até a descrição de cada elemento do Esquema individualmente, para produzir uma documentação que denominamos de Dicionário de Esquemas.

A título de sugestão e motivação para este estudo, criamos um "layout" para as folhas de diagrama (Figuras 8.2). No Apêndice A é representado um exemplo prático de modelagem, por meio de diagramas colocados em folhas no formato sugerido.

O diagrama mostra uma folha de papel com uma borda dupla. No canto inferior direito, há um formulário com os seguintes campos:

TÍTULO	
AUTOR	
BASE	FOLHA
DATA	VERSÃO
OBS	

Figura 8.2 - Sugestão de Padronização para as folhas de Diagramas

Acredita-se que para melhorar a compreensão, cada DRO representativo de um Esquema Suplementar deve ser colocado em folha individual. Excepcionalmente mais de uma folha pode ser necessária e portanto deve ser usada para a representação de um Esquema Suplementar.

8.4. - Considerações Finais

Por meio de Esquemas Suplementares, este trabalho mostrou que é possível proporcionar flexibilidade na estruturação lógica da Base Intencional, de maneira a permitir clara distinção entre áreas de interesse (disciplinas) que abrangem os dados armazenados na Base, permitindo a construção de gerenciadores que suportam o isolamento de dados, provido pelo conceito de Visões de Usuários da Arquitetura ANSI/X3/SPARC e aliado à flexibilidade das modificações no Esquema dos sistemas relacionais, com a riqueza de conceitos e estruturas sintáticas dos Modelos Orientados a Objetos.

Baseado em tais gerenciadores, é possível o desenvolvimento de ferramentas de apoio a projetos de engenharia [CAMOLESI_92], uma vez que os critérios de difusão de informações serão controlados internamente pelo SGBD.

Embora a desvantagem deste mecanismo de controle interno seja uma sobrecarga imposta ao gerenciador da Base de Dados no reconhecimento dos usuários e dos Esquemas Suplementares que podem ser utilizados, acredita-se que alterações do Esquema de Dados serão pouco utilizadas porém práticas e extremamente úteis, sendo que pode-se relatar as seguintes vantagens na sua utilização:

- **Ocultamento de Informações:** a restrição de acesso às Hierarquias de Colônias de Objetos (e mesmo Atributos e Relacionamentos de outras hierarquias) é realizada através da restrição de requisição aos Esquemas Suplementares. A restrição às Hierarquias de Generalização de Objetos é realizada através da restrição de acesso individual as Colônias de Objetos da Base Extensional;

- **Alteração Dinâmica do Esquema:** a montagem de um Esquema de Dados é realizada dinamicamente, a medida que requisições ou liberações de Colônias Esquema podem ser realizadas durante transações na Base Extensional. Desta forma, o gerenciador proporciona flexibilidade ao usuário, que poderá alterar seu Esquema de Dados segundo a evolução de suas necessidades;

- **Redução do Esquema:** a partição da Base Intencional em Esquemas Suplementares possibilita a redução do tamanho do Esquema de Dados para cada usuário individualmente, com a vantagem de reduzir o tempo de consulta ao Esquema, pois cada usuário requisita somente as Colônias Esquema que são de seu interesse e que permanecerão em memória principal para uso.

Deve-se destacar o fato que o conceito de Esquemas Suplementares estabelecido neste trabalho de mestrado é original e inédito.

Embora tenha sido desenvolvido no âmbito do MRO^{*}, o conceito de Esquemas Suplementares pode teoricamente ser aplicado a qualquer outro Modelo Orientado a Objetos. Para isso, sua conceituação requereu estudos das bibliografias aqui discutidas para assegurar a abrangência que este conceito pode ter em Modelos Orientados a Objetos, desde que adaptado às suas características.

REFERÊNCIAS BIBLIOGRÁFICAS

- [ANSI_78] "The ANSI/X3/SPARC DBMS Framework", Report of the Study Group on Database Management Systems in, Information Systems, Vol. 3, Nro. 3, pp. 173-191, 1978.
- [ABITEBOUL_91] Serge Abiteboul, Anthony Bonner, "Object and View", ACM SIGMOD 91 Conference, Denver, Colorado, pp. 238-247, Maio 1991.
- [AHMED_91] Rafi Ahmed, Shamkant B. Navathe, "Version Management of Composite Objects in CAD Databases", ACM SIGMOD 91 Conference, Denver, Colorado, pp. 218-227, Maio 1991.
- [ASTRAHAN_76] M. M. Astrahan, M. W. Blasgen, et alli, "System R: Relational Approach to Database Management", ACM Transactions on Database Systems, Vol. 1, Nro. 2, pp. 97-137, Junho 1976.
- [ATKINSON_89] Malcolm Atkinson et alli, "The Object Oriented Database System Manifesto", Technical Report Altair 30-89, GIP ALTAIR in 2-INRIA-LRI, Outubro 1989.
- [BANERJEE_87] Jay Banerjee, Won Kim, "Semantics and Implementation of Schema Evolution in Object-Oriented Databases", ACM SIGMOD 87 Conference, San Francisco, California, pp. 311-322, Maio 1987.
- [BATINI_86] C. Batini et alli, "A Comparative Analysis of Methodologies for Database Schema Integration", ACM Computing Survey, Vol 18, Nro. 4, pp. 323-364, Dezembro 1986.
- [BERTINO_91] Elisa Bertino, Lorenzo Martino, "Object-Oriented Database Management Systems: Concepts and Issues", IEEE Computer, pp. 33-47, Abril 1991.
- [BIAJIZ_92] Mauro Biajiz, "MRO* - Uma Atualização do Modelo de Representação de Objetos e Uma Abordagem Funcional", Dissertação de Mestrado, ICMSC-USP, 1992.
- [BJORNERSTEDT_89] Anders Bjornerstedt, Christer Hulten, "Version Control In An Object-Oriented Architecture", pp. 451-485, in [KIM_89b].
- [BLACKBURN_90] J. Blackburn, "Time-Bases Competition", Strategic Manufacturing-Dynamic New Directions for the 1990s, Dow-Jones Irwin, Homewood, Illinois, 1990.
- [BORGIDA_88] Alexander Borgida, "Modeling Class Hierarchies with Contradictions", ACM SIGMOD 88 Conference, Chicago, Illinois, pp. 434-443, Junho 1988.

- [BOUDIER_88] Gerard Boudier, et alli, "An Overview of PCTE and PCTE+", ACM SIGMOD 88 Conference, Chicago, Illinois, pp. 248-257, Junho 1988.
- [BRACHMAN_83] Ronald J. Brachman, "What IS-A Is and Isn't: An Analysis of Taxonomic Links In Semantic Networks", IEEE Computer, Vol 16, Nro. 10, pp. 30-36, Outubro 1983.
- [BRADLEY_67] James Bradley, *File & Data Base Techniques*, Eds. CBS College Publishing, Cap. 6, 1967.
- [BRETL_89] Bobert Bretl, et alli, "The GemStone Data Management System", pp. 283-308, in [KIM_89b].
- [CALÔNEGO_91] Nivaldi Calônego Jr., "Desenvolvimento de um Núcleo Multi-Usuário para um Sistema de Gerenciamento de Base de Dados Orientado a Objetos", Dissertação de Mestrado, ICMSC-USP, 1991.
- [CAMOLESI_92] Luiz Camolesi Júnior, Caetano Traina Júnior, "Construção de Ferramentas de Apoio a Projetos Utilizando um Gerenciador de Bases de Dados com Suporte de Esquemas Suplementares", in Anais do 6o. Simpósio Brasileiro de Engenharia de Software, Gramado, Novembro 1992.
- [CAREY_88] Michael J. Carey, et alli, "A Data Model and Query Language for EXODUS", ACM SIGMOD 88 Conference, Chicago, Illinois, pp. 413-423, Junho 1988.
- [CAREY_89] Michael J. Carey, et alli, "Storage Management for Object in EXODUS", pp. 341-369, in [KIM_89b].
- [CHEN_76] Peter Pin-Shan Chen, "The Entity-Relationship Model-Toward a Unified View of Data", ACM Transactions on Database Systems, Vol. 1, Nro. 1, pp. 9-36, Março 1976.
- [CODASYL_71] CODASYL Data Base Task Group, Report ACM, Abril 1971.
- [CODD_70] E. F. Codd, "A Relacional Model for Large Shared Data Banks", Communications of ACM, Vol. 13, Nro. 6, pp. 377-387, Junho 1970.
- [CODD_79] E. F. Codd, "Extending the Database Relational Model to Capture More Meaning", ACM Transactions Database Systems, Vol. 4, Nro. 4, pp. 397-434, Dezembro 1979.
- [CODD_82] E. F. Codd, "Relational Database: A Practical Foundation for Productivity", Communications of ACM, Vol. 25, Nro. 2, pp. 109-117, Fevereiro 1982.
- [DATE_88a] C. J. Date, *Introdução a Sistemas de Bancos de Dados*, Campus, 1988.
- [DATE_88b] C. J. Date, *Bancos de Dados-Tópicos Avançados*, Campus, 1988.

- [DEUX_90] O. Deux, et alli, "The Story of O2", IEEE Transaction on Knowledge and Data Engineering, Vol. 2, Nro. 1, pp. 91-108, Março 1990.
- [ELMASRI_89] Ramez Elmasri, Shamkant B. Navathe, **Fundamentals of Database Systems**, Addison-Wesley, 1989.
- [FERREIRA_91] João Eduardo Ferreira, "Estudo da Distribuição de Base de Dados Orientada a Objetos", Dissertação de Mestrado, IFQSC-USP, 1991.
- [FISHMAN_89] D. H. Fishman, et alli, "Overview of the Iris DBMS", pp. 219-250, in [KIM_89b].
- [FOARD_85] Richard M. Foard, "A Data Manager Using Entity-Relationships", PC Tech Journal, pp. 97-129, Outubro 1985.
- [GALLO_87] F. Gallo, R. Minot, Ian Thomas, "The Object Management System of PCTE as a Software Engineering Database Management System", ACM SIGSOFT/SIGPLAN, Palo Alto, California, Dezembro 1986.
- [GARVEY_89] M. A. Garvey, M. S. Jackson, "Introduction to Object-Oriented Databases", Information and Software Technology, Vol. 31, Nro. 10, pp. 521-527, Dezembro 1989.
- [HARTSON_89] H. Rex Hartson, Deborah Hix, "Human-Computer Interface Development: Concepts and Systems for Its Management", ACM Computing Surveys, Vol. 21, Nro. 1, pp. 5-92, Março 1989.
- [HAMMER_81] Michael Hammer, Dennis McLeod, "Database Description with SDM: A Semantic Database Model", ACM Transactions on Database Systems, Vol. 6, Nro. 3, pp. 351-386, Setembro 1981.
- [HARTZBAND_85] D. J. Hartzband, F. J. Maryanski, "Enhancing Knowledge Representation in Engineering Databases", IEEE Computer, Vol. 18, Nro. 9, pp. 39-48, Setembro 1985.
- [HUDSON_87] Scott E. Hudson, Roger King, "Object-Oriented Database Support for Software Environments", ACM SIGMOD 87 Conference, San Francisco, California, pp. 491-503, Maio 1987.
- [HUDSON_88] Scott E. Hudson, Roger King, "The Cactis Project: Database Support for Software Environment", IEEE Transaction on Software Engineering, Vol. 14, Nro. 6, pp. 709-719, Junho 1988.
- [HUDSON_89] Scott E. Hudson, Roger King, "Cactis: A Self-Adaptive, Concurrent Implementation of an Object-Oriented Database Management System", ACM Transactions on Database Systems, Vol. 14, Nro. 3, pp. 291-321, Setembro 1989.
- [HULL_87] Richard Hull, Roger King, "Semantic Database Modeling: Survey, Applications and Research Issues", ACM Computing Surveys, Vol. 19, Nro. 3, pp. 201-259, Setembro 1987.

- [JACKSON_90] M. S. Jackson, "Beyond Relational Databases", Information and Software Technology, Vol. 32, Nro. 4, pp. 258-265, Maio 1990.
- [JACKSON_91] M. S. Jackson, "Tutorial on Object-Oriented Databases", Information and Software Technology, Vol. 33, Nro. 1, pp. 4-12, Fevereiro 1991.
- [KATZ_84] Randy H. Katz, T. J. Lehman, "Database Support for Versions and Alternatives of Large Design Files", IEEE Transactions on Software Engineering, Vol. SE-10, Nro. 2, pp. 191-200, Março 1984.
- [KATZ_86] Randy H. Katz, Ellis Chang, Rajiv Bhateja, "Version Modeling Concepts for Computer-Aided Design Databases", ACM SIGMOD 86 Conference, Washington D.C., pp. 379-386, Maio 1986.
- [KATZ_90] Randy H. Katz, "Toward a Unified Framework for Version Modeling in Engineering Databases", ACM Computing Surveys, Vol. 22, Nro. 4, pp. 375-408, Dezembro 1990.
- [KELTER_87] Udo Kelter, "Concurrency Control for Design Objects With Versions in Cad Databases", Information Systems, Vol. 12, Nro. 2, pp. 137-143, 1987.
- [KIM_87] Won Kim, Jay Banerjee, et alli, "Composite Object Support in an Object-Oriented Database System", OOPSLA 87 Conference, Orlando, Florida, pp. 118-125, Outubro 1987.
- [KIM_89a] Won Kim, Elisa Bertino, Jorge F Garza, "Composite Objects Revisted", ACM SIGMOD 89 Conference, Portland, Oregon, pp. 337-347, Junho 1989.
- [KIM_89b] Won Kim, Frederick H. Lochovsky, Object-Oriented Concepts, Databases, and Applications, Addison-Wesley, 1989.
- [KIM_89c] Won Kim, et alli, "Feature of the ORION Object-Oriented Database System", pp. 251-281, in [KIM_89b].
- [KIM_90a] Won Kim, et alli, "Architecture of the ORION Next-Generation Database System", IEEE Transactions on Knowledge and Data Engineering, Vol. 2, Nro. 1, pp. 109-124, Março 1990.
- [KIM_90b] Won Kim, "Object-Oriented Databases: Definition, and Research Directions", IEEE Transactions on Knowledge and Data Engineering, Vol. 2, Nro. 3, pp. 327-341, Setembro 1990.
- [KORTH_89] Henry F. Korth, Abraham Silberschatz, Sistemas de Bancos de Dados, McGraw-Hill, 1989.
- [LECLUSE_88] Christophe Lecluse, "O2, an Object-Oriented Data Model", ACM SIGMOD 88 Conference, Chicago, Illinois, pp. 424-433, Junho 1988.
- [LOCKEMANN_89] Peter C. Lockemann, "Object-Oriented Information Management", Decision Support Systems, Vol. 5, Nro. 1, pp. 79-102, Março 1989.

- [MCLEOD_91] D. McLeod, "Perspective on Object Databases", Information and Soft. Technology, Vol. 33, Nro. 1, pp. 13-21, Janeiro/Fevereiro 1991.
- [NAVATHE_86] Shamkant Navathe et alli, "Integrating User Views in Database Design", Computer, pp. 50-62, Janeiro 1986.
- [PCTE_89] PCTE- A Basic for a Portable Common Tool Environment Functional Specitication", Versão 1.5, C Volume 1, Junho 1989.
- [PECKHAM_88] Joan Peckham, Fred Maryanski, "Semantic Data Models", ACM Computing Surveys, Vol. 20, Nro. 3, pp. 153-189, Setembro 1988.
- [PENNEY_87] D. Jason Penney, Jacob Stein, "Class Modification in the GemStone Object-Oriented DBMS", OOPSLA 87 Conference, Orlando, Florida, pp. 111-117, Outubro 1987.
- [PIZZIGATTI_92] Pedro Luiz Pizzigatti Corrêa, "Uma Linguagem para o Gerenciador de Objetos baseada no Modelo de Representação de Objetos", Dissertação de Mestrado, ICMSC-USP, 1992.
- [PRESSMAN_87] Roger S. Pressman, **Software Engineering**, McGraw-Hill, 1987.
- [PRICE_89] Roberto Tom Price, Lia Goldstein Golendziner, **Bancos de Dados Para Aplicações Não Convencionais**, IV Escola Brasileiro-Argentina de Informática, Termas do Rio Hondo, Argentina, 1989.
- [RICHARDSON_87] Joel E. Richardson, Michael J. Carey, "Programming Constructs for Database Systems Implementation in EXODUS", ACM SIGMOD 87 Conference, San Francisco, California, pp. 208-219, Maio 1987.
- [RÓDITCHEV_87] V. Róditchev, G. Róditcheva, **Tractores e Automóveis**, Mir Moscovo, URSS, 1987
- [SCIORE_89] Edward Sciore, "Object Specialization", ACM Transations on Information Systems, Vol. 7, Nro. 2, pp. 103-122, Abril 1989.
- [SKARRA_89] Andrea H. Skarra, Stanley B. Zdonik, "Concurrency Control and Object-Oriented Databases", pp. 395-422, in [KIM_89b].
- [SMITH_77a] John M. Smith, Diane C. P. Smith, "Database Abstrations: Aggregation and Generalization", ACM Transations on Database Systems, Vol. 2, Nro. 2, pp. 105-133, Junho 1977.
- [SMITH_77b] John M. Smith, Diane C. P. Smith, "Database Abstraction: Aggregation", Communication of ACM, Vol. 20, Nro. 6, pp. 405-413, Junho 1977.
- [SOCKUT_79] Gary H. Sockut, Robert P. Goldberg, "Database Reorganization-Principle and Practice", ACM Computing Surveys, Vol. 11, Nro. 4, pp. 371-395, Dezembro 1979.

- [STONEBRAKER_76] Michael Stonebraker, et alli, "The Design and Implementation of INGRES", ACM Transactions on Database Systems, Vol. 1, Nro. 3, pp. 189-222, Setembro 1976.
- [TAKAHASHI_90] Tadao Takahashi, Hans K. E. Liesenberg, **Programação Orientada a Objetos**, VII Escola de Computação, São Paulo, 1990.
- [TEOREY_82] Toby J. Teorey, **Design of Database Structures**, Prentice-Hall, New Jersey, 1982.
- [TEOREY_86] Toby J. Teorey, Dongqing Yang, James P. Fry, "A Logical Design Methodology for Relacional Databases Using Extended Entity-Relationships Model", ACM Computing Surveys, Vol. 18, Nro. 2, pp. 197-222, Junho 1986.
- [TRAINA_86] Caetano Traina Jr., "Máquina e Modelo de Dados Dedicados para Aplicações de Engenharia", Tese de Doutorado, IFQSC-USP, Dezembro 1986.
- [TRAINA_88] Caetano Traina Jr., Jan Frans Willem Slaets, "Um Modelo de Representação de Objetos", in Anais do 3o. Simpósio Brasileiro de Banco de Dados, Recife, pp. 227-242, Março 1988.
- [TRAINA_91] Caetano Traina Jr., "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, pp 193-207, Maio 1991.
- [WEISER_89] Stevem P. Weiser, Frederick H. Lochovsky, "OZ+: An Object-Oriented Database Systems", in [KIM_89b].
- [WILKINSON_90] Kevin Wilkinson, et alli, "The Iris Architecture and Implementation", IEEE Transaction on Knowledge and Data Engineering, Vol. 2, Nro. 1, pp. 63-75, Março 1990.
- [WOELK_86] Darrell Woelk, Won Kim, Willis Luther, "An Object-Oriented Approach to Multimedia Databases", ACM SIGMOD 86 Conference, Washington D.C., pp. 311-325, Maio 1986.
- [ZICARI_90] Roberto Zicari, "Incomplete Information in Object-Oriented Databases", ACM SIGMOD Record, Vol. 19, Nro. 3, pp. 5-16, Setembro 1990.

APÊNDICE A

Exemplo de Modelagem de Dados

Considerações:

Este exemplo mostra uma modelagem simplificada de um projeto de veículo terrestre genérico (motor de combustão ou elétrico e movido a óleo diesel ou outro combustível). Devido a complexidade desta aplicação, a modelagem considerou apenas alguns aspectos da aplicação, suficientes para ilustrar os conceitos definidos neste trabalho e portanto, somente os elementos mais representativos de um veículo foram especificados.

As informações para esta modelagem foram extraídas da tradução para o português do livro "Tractores e Automóveis" [RÓDITCHEV_87].

Este apêndice é composto pelos seguintes diagramas:

Folha 1-: DRI - Colônia Global

Mostra os Objetos que constroem os 6 Esquemas Suplementares contidos nesta modelagem.

Folha 2-: DHC - Esquema Inicial

Mostra a Hierarquia de Colônias definida pelo Esquema Inicial e portanto a estrutura de Colônias visível por todos os usuários da Base de Dados.

Folha 3-: DRO - Esquema Inicial (GLOBAL)

Mostra os tipos definidos para a Colônia Global, ou seja, quais os tipos de componentes representados genericamente pela Base de Dados.

Folha 4-: DRO - Esquema Inicial (SISTEMAS)

Mostra os tipos definidos para as Colônias do tipo Sistemas, ou seja, quais os tipos de sistemas que são componentes de um veículo.

Folha 5-: DHC - A1

Mostra a Hierarquia de Colônias definida pelo Esquema Suplementar A1 e que detalha a estrutura mecânica de um veículo.

Folha 6-: DRO - A1 (SUBSISTEMAS)

Mostra os tipos definidos para as Colônias do tipo Subsistemas, ou seja, quais os tipos de sistemas que compõem a estrutura mecânica de um veículo.

Folha 7-: DRO - A1 (DIREÇÃO)

Mostra os tipos definidos para as Colônias do tipo Direção, ou seja, quais os tipos de componentes do sistema de direção.

Folha 8-: DRO - A1 (TRANSMISSÃO)

Mostra os tipos definidos para as Colônias do tipo Transmissão, ou seja, quais os tipos de componentes do sistema de transmissão.

Folha 9-: DRO - A1 (ROLAMENTO)

Mostra os tipos definidos para as Colônias do tipo Rolamento, ou seja, quais os tipos de componentes do sistema de rolamento.

Folha 10-: DHC - B1

Mostra a Hierarquia de Colônias definida pelo Esquema Suplementar B1 e que detalha a estrutura elétrica de um veículo.

Folha 11-: DRO - B1 (ELÉTRICO)

Mostra os tipos definidos para as Colônias do tipo Elétrico, ou seja, quais os tipos de componentes que compõem a estrutura elétrica de um veículo bem como a interação do Tipo Motor de Arranque com um Tipo de Motor.

Folha 12-: DRO - B1 (ARRANQUE)

Mostra os tipos definidos para as Colônias do tipo Arranque, ou seja, quais os tipos de componentes de um motor de arranque.

Folha 13-: DHC - C1

Mostra a Hierarquia de Colônias definida pelo Esquema Suplementar C1 e que detalha quais são os tipos de componentes de um motor de combustão.

Folha 14-: DRO - C1 (MOTOR)

Mostra os tipos definidos para as Colônias do tipo Motor, ou seja, quais os tipos de sistemas de um motor de combustão.

Folha 15-: DRO - C1 (REFRIGERAÇÃO)

Mostra os tipos definidos para as Colônias do tipo Refrigeração, ou seja, quais os tipos de componentes do sistema de refrigeração de um motor de combustão.

Folha 16-: DRO - C2 (MOTOR)

Mostra os tipos definidos para as Colônias do tipo Motor. Este único tipo de Colônia definido no Esquema Suplementar C2 detalha quais são os tipos de sistemas de um motor elétrico.

Folha 17-: DRO - D1 (CICLO)

Mostra os tipos definidos para as Colônias do tipo Ciclo. Este único tipo de Colônia definido no Esquema Suplementar D1 detalha como é o sistema de alimentação para o Ciclo denominado Otto.

Folha 18-: DRO - D2 (CICLO)

Mostra os tipos definidos para as Colônias do tipo Ciclo. Este único tipo de Colônia definido no Esquema Suplementar D2 detalha como é o sistema de alimentação para o Ciclo denominado Diesel.

GLOBAL

SISTEMAS

TÍTULO: DHC - Esquema Inicial			
AUTOR: Luiz Camolesi Júnior			
BASE: Auto		FOLHA: 2	
DATA: 30/09/92		VERSÃO: 1.0	
OBS: modelagem simplificada

TÍTULO: DRD - Esquema Inicial (Global)
AUTOR: Luiz Camolesi Júnior
BASE: Auto
DATA: 30/09/92
VERSÃO: 1.0
OBS: modelagem simplificada.

TÍTULO: DRD - Esquema Inicial (Sistemas)	
AUTOR: Luiz Camolesi Júnior	
BASE: Auto	FOLHA: 4
DATA: 30/09/92	VERSÃO: 1.0
☐ OBS: modelagem simplificada.

TÍTULO: DHC - A1	
AUTOR: Luiz Camolesi Júnior	
BASE: Auto	FOLHA: 5
DATA: 30/09/92	VERSÃO: 1.0
☐ BS: modelagem simplificada

TÍTULO: DRD - A1 (Subsistemas)
AUTOR: Luiz Camolesi Júnior
BASE: Auto
DATA: 30/09/92
FOLHA: 6
VERSÃO: 1.0
OBS: modelagem simplificada.

TÍTULO: DRD - Al (Direção)	
AUTOR: Luiz Camolesi Júnior	
BASE: Auto	FOLHA: 7
DATA: 30/09/92	VERSÃO: 1.0
OBS: modelagem simplificada.

TÍTULO: DRD - A1 (Transmissão)	
AUTOR: Luiz Camolesi Júnior	
BASE: Auto	FOLHA: 8
DATA: 30/09/92	VERSÃO: 1.0
OBS: modelagem simplificada

TÍTULO: DRD - A1 (Rolamento)	
AUTOR: Luiz Camolesi Júnior	
BASE: Auto	FOLHA: 9
DATA: 30/09/92	VERSÃO: 1.0
OBS: modelagem simplificada	

ELÉTRICO

ARRANQUE

TÍTULO: DHC - B1	
AUTOR: Luiz Camolesi Júnior	
BASE: Auto	FOLHA: 10
DATA: 30/09/92	VERSÃO: 1.0
OBS: modelagem simplificada

TÍTULO: DRD - B1 (Elétrico)	
AUTOR: Luiz Camolesi Júnior	
BASE: Auto	FOLHA: 11
DATA: 30/09/92	VERSÃO: 1.0
☐ BS: modelagem simplificada.

TÍTULO: DRD - B1 (Arranque)			
AUTOR: Luiz Camolesi Júnior			
BASE: Auto		FOLHA: 12	
DATA: 30/09/92		VERSÃO: 1.0	
☐ OBS: modelagem simplificada.			

MOTOR

REFRIGERAÇÃO

TÍTULO: DHC - C1			
AUTOR: Luiz Camolesi Júnior			
BASE: Auto		FOLHA: 13	
DATA: 30/09/92		VERSÃO: 1.0	
☐ BS: modelagem simplificada

TÍTULO: DRD - C1 (Motor)	
AUTOR: Luiz Camolesi Júnior	
BASE: Auto	FOLHA: 14
DATA: 30/09/92	VERSÃO: 1.0
OBS: modelagem simplificada

número pás
rotação
deslocamento

diâmetro
material

material
cap. volum.
des. matriz

rotação
potência

TÍTULO: DRD - C1 (Refrigeração)	
AUTOR: Luiz Camolesi Júnior	
BASE: Auto	FOLHA: 15
DATA: 30/09/92	VERSÃO: 1.0
OBS: modelagem simplificada.

TÍTULO: DRD - C2 (Motor)	
AUTOR: Luiz Camolesi Júnior	
BASE: Auto	FOLHA: 16
DATA: 30/09/92	VERSÃO: 1.0
☐ OBS: modelagem simplificada

TÍTULO: DRD - D1 (Ciclo)	
AUTOR: Luiz Camolesi Júnior	
BASE: Auto	FOLHA: 17
DATA: 30/09/92	VERSÃO: 1.0
□BS: modelagem simplificada

TÍTULO: DRD - D2 (Ciclo)	
AUTOR: Luiz Camolesi Júnior	
BASE: Auto	FOLHA: 18
DATA: 30/09/92	VERSÃO: 1.0
OBS: modelagem simplificada.	

APÊNDICE B

Exemplo de Utilização da LAMRO

Considerações:

Este apêndice mostra a utilização dos comandos da LAMRO de definição de Colônias Esquema, na implantação de um exemplo significativo de modelagem apresentado no apêndice A.

Os comandos apresentam-se agrupados segundo os DRO's do apêndice A que definem. Esta ordem foi seguida somente para fins didáticos, não sendo portanto uma obrigatoriedade imposta pela linguagem.

/* Definição da Colônia Esquema Inicial */

/* Definição do diagrama da FOLHA 3 */

DEFINE OBJETO Fabrica, Veiculo, Circuito,
Pessoa

HABITA GLOBAL;

DEFINE RELACIONAMENTO projeta

OPOSTO projetado_por

ORIGEM Fabrica

DESTINO Veiculo

MINIMO 1 MAXIMO N

MINIMO OPOSTO 1

MAXIMO OPOSTO 1;

DEFINE RELACIONAMENTO corre_em

OPOSTO usado_por

ORIGEM Veiculo

DESTINO Circuito

MINIMO 1 MAXIMO N

MINIMO OPOSTO 1

MAXIMO OPOSTO N;

DEFINE RELACIONAMENTO trabalha_com

OPOSTO trabalhado

ORIGEM Pessoa

DESTINO Veiculo

MINIMO 1 MAXIMO 1

MINIMO OPOSTO 1

MAXIMO OPOSTO N;

DEFINE TRIPLO testa_em

MODALIDADE CONJUNTO usado_por

MODALIDADE DETERMINA projeta

DIVISAO UM Veiculo

DIVISAO DOIS Circuito

DIVISAO TRES Fabrica

MINIMO 1 MAXIMO N;

DEFINE PROPRIEDADE idade

TIPO DE DADO inteiro

ASSOCIADO OBJETO Pessoa;

DEFINE PROPRIEDADE funcao

TIPO DE DADO cadeia

ASSOCIADO OBJETO Pessoa;

DEFINE PROPRIEDADE nacionalidade

TIPO DE DADO cadeia

ASSOCIADO OBJETO Pessoa;

DEFINE PROPRIEDADE pais

TIPO DE DADO cadeia

ASSOCIADO

OBJETO Circuito

OBJETO Fabrica;

DEFINE SUBTIPO OBJETO Piloto

TIPO MESTRE Pessoa

PREDICADO funcao == Piloto;

DEFINE SUBTIPO OBJETO Projetista

TIPO MESTRE Pessoa

PREDICADO funcao == Projetista;

/* Definição do diagrama da FOLHA 4 */

DEFINE COLONIA SISTEMAS

CONSTRITA POR Veiculo;

DEFINE OBJETO S_Mecanico, S_Eletrico, Chassis,

Carroceria, S_Hidraulico

HABITA SISTEMAS;

DEFINE RELACIONAMENTO controlado

OPOSTO controla

ORIGEM S_Mecanico

DESTINO S_Eletrico

MINIMO 1 MAXIMO 1

MINIMO OPOSTO 1

MAXIMO OPOSTO 1;

DEFINE RELACIONAMENTO monitorado

OPOSTO monitora

ORIGEM S_Mecanico

DESTINO S_Eletrico

MINIMO 1 MAXIMO 1

MINIMO OPOSTO 1

MAXIMO OPOSTO 1;

DEFINE RELACIONAMENTO sustenta

OPOSTO sustentado_por

ORIGEM Chassi

DESTINO Carroceria

MINIMO 1 MAXIMO 1

MINIMO OPOSTO 1

MAXIMO OPOSTO 1;

DEFINE PROPRIEDADE elasticidade

TIPO DE DADO real

ASSOCIADO OBJETO S_Mecanico;

DEFINE PROPRIEDADE amortecimento

TIPO DE DADO real

ASSOCIADO OBJETO S_Mecanico;

DEFINE PROPRIEDADE rendimento

TIPO DE DADO real

ASSOCIADO OBJETO S_Mecanico;

DEFINE PROPRIEDADE estrutura

TIPO DE DADO cadeia

ASSOCIADO OBJETO Chassi;

DEFINE PROPRIEDADE potencia

TIPO DE DADO real

ASSOCIADO OBJETO S_Eletrico;


```

DEFINE IMAGEM formato
    ASSOCIADO OBJETO Carroceria;
DEFINE PROPRIEDADE material_principal
    TIPO DE DADO cadeia
    ASSOCIADO OBJETO Carroceria;
DEFINE PROPRIEDADE capacidade_volumetrica
    TIPO DE DADO real
    ASSOCIADO OBJETO S_Hidraulico;
DEFINE PROPRIEDADE eficiencia
    TIPO DE DADO real
    ASSOCIADO OBJETO S_Hidraulico;
DEFINE SUBTIPO OBJETO Monobloco
    TIPO MESTRE Chassi
    PREDICADO estrutura == Monobloco;
DEFINE SUBTIPO OBJETO Longarina
    TIPO MESTRE Chassi
    PREDICADO estrutura == Longarina;

```

/* Definição da Colônia Esquema A1 */

/* Definição do diagrama da FOLHA 6 */

```

CONSIDERA ESQUEMA
    CONSTRIITA POR Area_Mecanica
    ESQUEMATIZA S_Mecanico;
DEFINE COLONIA SUBSISTEMAS
    CONSTRIITA POR S_Mecanico;
DEFINE OBJETO S_Transmissão, Freio, Motor,
S_Direção, S_Rolamento
    HABITA SUBSISTEMAS;
DEFINE PROPRIEDADE reducao
    TIPO DE DADO real
    ASSOCIADO OBJETOS_Transmissao;
DEFINE PROPRIEDADE torque
    TIPO DE DADO real
    ASSOCIADO OBJETO Motor;
DEFINE PROPRIEDADE potencia
    TIPO DE DADO real
    ASSOCIADO OBJETO Motor;
DEFINE PROPRIEDADE consumo
    TIPO DE DADO real
    ASSOCIADO OBJETO Motor;
DEFINE PROPRIEDADE disp_cilindros
    TIPO DE DADO cadeia
    ASSOCIADO OBJETO Motor;
DEFINE PROPRIEDADE n_cilindros
    TIPO DE DADO inteiro
    ASSOCIADO OBJETO Motor;

```

```

DEFINE PROPRIEDADE cilindradas
    TIPO DE DADO real
    ASSOCIADO OBJETO Motor;
DEFINE COMENTARIO aplicacao
    ASSOCIADO OBJETO Freio;
DEFINE PROPRIEDADE acionamento
    TIPO DE DADO cadeia
    ASSOCIADO OBJETO Freio
        OBJETO S_Direcao;
DEFINE SUBTIPO Tambor
    TIPO MESTRE Freio
    PREDICAD acionamento == Tambor;
DEFINE SUBTIPO Disco
    TIPO MESTRE Freio
    PREDICADO acionamento == Disco;
DEFINE SUBTIPO Mecanico
    TIPO MESTRE Direcao
    PREDICADO
        acionamento == Mecanico;
DEFINE SUBTIPO Hidraulico
    TIPO MESTRE Direcao
    PREDICADO
        acionamento == Hidraulico;

```

/* Definição do diagrama da FOLHA 7 */

```

DEFINE COLONIA DIRECAO
    CONSTRIITA POR S_Direcao;
DEFINE OBJETO Pinhao, Volante, Cremalheira
    HABITA DIRECAO;
DEFINE PROPRIEDADE reducao
    TIPO DE DADO real
    ASSOCIADO
        RELACIONAMENTO aciona
            Pinhao Cremalheira;
DEFINE PROPRIEDADE diametro
    TIPO DE DADO real
    ASSOCIADO OBJETO Volante;
DEFINE PROPRIEDADE material
    TIPO DE DADO cadeia
    ASSOCIADO
        OBJETO Cremalheira
        OBJETO Pinhao;
DEFINE RELACIONAMENTO aciona
    OPOSTO acionado_por
    ORIGEM Volante
    DESTINO Pinhao
    MINIMO 1 MAXIMO 1
    MINIMO OPOSTO 1
    MAXIMO OPOSTO 1;

```

```

DEFINE RELACIONAMENTO aciona
    OPOSTO acionado_por
    ORIGEM Pinhao
    DESTINO Cremalheira
    MINIMO 1 MAXIMO 1
    MINIMO OPOSTO 1
    MAXIMO OPOSTO 1;

```

/* Definição do diagrama da FOLHA 8 */

```

DEFINE COLONIA TRANSMISSAO
    CONSTRITA POR S_Transmissao;
DEFINE OBJETO Embreagem,Caixa_vel,
    Alavanca, Diferencial
    HABITA TRANSMISSAO;
DEFINE PROPRIEDADE n_marchas
    TIPO DE DADO inteiro
    ASSOCIADO OBJETO Caixa_vel;
DEFINE PROPRIEDADE reducao
    TIPO DE DADO real
    ASSOCIADO OBJETO Diferencial;
DEFINE RELACIONAMENTO aciona
    OPOSTO acionado_por
    ORIGEM Embreagem
    DESTINO Caixa_vel
    MINIMO 1 MAXIMO 1
    MINIMO OPOSTO 1
    MAXIMO OPOSTO 1;
DEFINE RELACIONAMENTO ligado_a
    OPOSTO ligado_em
    ORIGEM Embreagem
    DESTINO Motor
    MINIMO 1 MAXIMO 1
    MINIMO OPOSTO 1
    MAXIMO OPOSTO 1;
DEFINE RELACIONAMENTO seleciona
    OPOSTO selecionado_por
    ORIGEM Alavanca
    DESTINO Caixa_vel
    MINIMO 1 MAXIMO 1
    MINIMO OPOSTO 1
    MAXIMO OPOSTO 1;
DEFINE TRIPLO movimenta
    MODALIDADE CONJUNTO ligado_em
    MODALIDADE DETERMINA
        acionado_por
    DIVISAO UM Embreagem
    DIVISAO DOIS Motor
    DIVISAO TRES Caixa_vel
    MINIMO 1 MAXIMO 1;

```

/* Definição do diagrama da FOLHA 9 */

```

DEFINE COLONIA ROLAMENTO
    CONSTRITA POR S_Rolamento;
DEFINE OBJETO Pneus, Rodas, Suspensão
    HABITA ROLAMENTO;
DEFINE PROPRIEDADE forma
    TIPO DE DADO cadeia
    ASSOCIADO OBJETO Pneus;
DEFINE PROPRIEDADE material
    TIPO DE DADO cadeia
    ASSOCIADO OBJETO Rodas;
DEFINE SUBTIPO Radial
    TIPO MESTRE Pneus
    PREDICADO forma == Radial;
DEFINE SUBTIPO Diagonal
    TIPO MESTRE Pneus
    PREDICADO forma == Diagonal;
DEFINE SUBTIPO Liga_leve
    TIPO MESTRE Rodas
    PREDICADO material < > Aco;
DEFINE SUBTIPO Aco
    TIPO MESTRE Rodas
    PREDICADO material == Aco;

```

FINALIZA ESQUEMA CONSTRITA POR
Area_Mecanica;

/* Definição da Colônia Esquema B1 */

```

CONSIDERA ESQUEMA
    CONSTRITA POR Area_Eletrico
    ESQUEMATIZA S_Eletrico;

```

/* Definição do diagrama da FOLHA 11 */

```

DEFINE COLONIA ELETRICO
    CONSTRITA POR S_Eletrico;
DEFINE OBJETO Distribuidor, Vela, Bateria,
    Cabo_eletrico, Motor_arranque
    HABITA ELETRICO;
DEFINE PROPRIEDADE resistencia
    TIPO DE DADO real
    ASSOCIADO OBJETO Cabo_eletrico;
DEFINE PROPRIEDADE tensao
    TIPO DE DADO real
    ASSOCIADO OBJETO Bateria;
DEFINE PROPRIEDADE amperagem
    TIPO DE DADO real
    ASSOCIADO OBJETO Bateria;

```

```

DEFINE PROPRIEDADE tensao
    TIPO DE DADO real
    ASSOCIADO OBJETO Motor_arranque;
DEFINE PROPRIEDADE potencia
    TIPO DE DADO real
    ASSOCIADO OBJETO Motor_arranque;
DEFINE PROPRIEDADE torque
    TIPO DE DADO real
    ASSOCIADO OBJETO Motor_arranque;
DEFINE RELACIONAMENTO utiliza
    OPOSTO utilizado_por
    ORIGEM Distribuidor
    DESTINO Cabo_eletrico
    MINIMO 1 MAXIMO N
    MINIMO OPOSTO 1
    MAXIMO OPOSTO 1;
DEFINE RELACIONAMENTO utiliza
    OPOSTO utilizado_por
    ORIGEM Bateria
    DESTINO Cabo_eletrico
    MINIMO 1 MAXIMO N
    MINIMO OPOSTO 1
    MAXIMO OPOSTO N;
DEFINE RELACIONAMENTO ativa
    OPOSTO ativado_por
    ORIGEM Bateria
    DESTINO Motor_arranque
    MINIMO 1 MAXIMO 1
    MINIMO OPOSTO 1
    MAXIMO OPOSTO 1;
DEFINE RELACIONAMENTO liga
    OPOSTO ligado_por
    ORIGEM Motor_arranque
    DESTINO Motor
    MINIMO 1 MAXIMO 1
    MINIMO OPOSTO 1
    MAXIMO OPOSTO 1;

```

/* Definição do diagrama da FOLHA 12 */

```

DEFINE COLONIA ARRANQUE
    CONSTRIITA POR Motor_arranque;
DEFINE OBJETO Carcaca, Rotor, Estator, Escova
    HABITA ARRANQUE;

FINALIZA ESQUEMA CONSTRIITA POR
    Area_Eletrica;

```

/* Definição da Colônia Esquema C1 */

```

CONSIDERA ESQUEMA
    CONSTRIITA POR Motor_combustao
    ESQUEMATIZA Motor;

```

/* Definição do diagrama da FOLHA 14 */

```

DEFINE COLONIA MOTOR
    CONSTRIITA POR Motor;
DEFINE OBJETO S_Lubrificacao, S_Alimentacao,
    S_Escapamento, S_Refrigeracao,
    Mec_dist_gases, Mec_biela_manivela
    HABITA MOTOR;
DEFINE PROPRIEDADE capacidade
    TIPO DE DADO real
    ASSOCIADO OBJETO S_Lubrificacao;
DEFINE PROPRIEDADE oleo
    TIPO DE DADO cadeia
    ASSOCIADO OBJETO S_Lubrificacao;
DEFINE PROPRIEDADE controle_emissao
    TIPO DE DADO cadeia
    ASSOCIADO
        OBJETOS_Escapamento;
DEFINE PROPRIEDADE meio
    TIPO DE DADO cadeia
    ASSOCIADO OBJETOS_Refrigeracao;

```

/* Definição do diagrama da FOLHA 15 */

```

DEFINE COLONIA REFRIGERACAO
    CONSTRIITA POR S_Refrigeracao;
DEFINE OBJETO Ventilador, Mangueira, Radiador,
    Bomba_dagua
    HABITA REFRIGERACAO;
DEFINE PROPRIEDADE num_pas
    TIPO DE DADO inteiro
    ASSOCIADO OBJETO Ventilador;
DEFINE PROPRIEDADE rotacao
    TIPO DE DADO real
    ASSOCIADO OBJETO Ventilador;
DEFINE PROPRIEDADE deslocamento
    TIPO DE DADO real
    ASSOCIADO OBJETO Ventilador;
DEFINE PROPRIEDADE diametro
    TIPO DE DADO real
    ASSOCIADO OBJETO Mangueira;
DEFINE PROPRIEDADE material
    TIPO DE DADO cadeia
    ASSOCIADO OBJETO Mangueira;

```

```

DEFINE PROPRIEDADE material
    TIPO DE DADO cadeia
    ASSOCIADO OBJETO Radiador;
DEFINE PROPRIEDADE cap_volumetrica
    TIPO DE DADO real
    ASSOCIADO OBJETO Radiador;
DEFINE IMAGEM desenho_matriz
    ASSOCIADO OBJETO Radiador;
DEFINE PROPRIEDADE rotacao
    TIPO DE DADO real
    ASSOCIADO OBJETO Bomba_dagua;
DEFINE PROPRIEDADE potencia
    TIPO DE DADO real
    ASSOCIADO OBJETO Bomba_dagua;

```

```

FINALIZA ESQUEMA CONSTRITA POR
    Motor_combustao;

```

/* Definição da Colônia Esquema C2 */

```

CONSIDERA ESQUEMA
    CONSTRITA POR Motor_eletrico
    ESQUEMATIZA Motor;

```

/* Definição do diagrama da FOLHA 16 */

```

DEFINE COLONIA MOTOR
    CONSTRITA POR Motor;
DEFINE OBJETO S_Alimentacao, S_Refrigeracao
    HABITA MOTOR;
DEFINE PROPRIEDADE potencia
    TIPO DE DADO real
    ASSOCIADO OBJETOS_Refrigeracao;
DEFINE PROPRIEDADE num_baterias
    TIPO DE DADO inteiro
    ASSOCIADO OBJETO S_Alimentacao;

```

```

FINALIZA ESQUEMA CONSTRITA POR
    Motor_eletrico;

```

/* Definição da Colônia Esquema D1 */

```

CONSIDERA ESQUEMA
    CONSTRITA POR Ciclo_otto
    ESQUEMATIZA S_Alimentacao;

```

/* Definição do diagrama da FOLHA 17 */

```

DEFINE COLONIA CICLO
    CONSTRITA POR S_Alimentacao;

```

```

DEFINE OBJETO Carburador, Tanque,
    Bomba_comb, Filtro_ar, Filtro_comb
    HABITA CICLO;

```

```

DEFINE PROPRIEDADE num_estagios
    TIPO DE DADO inteiro
    ASSOCIADO OBJETO Carburador;

```

```

DEFINE PROPRIEDADE material
    TIPO DE DADO cadeia
    ASSOCIADO OBJETO Tanque;

```

```

DEFINE PROPRIEDADE capacidade
    TIPO DE DADO real
    ASSOCIADO
        OBJETO Tanque
        OBJETO Bomba_comb
        OBJETO Filtro_ar
        OBJETO Filtro_comb;

```

```

FINALIZA ESQUEMA CONSTRITA POR
    Ciclo_otto;

```

/* Definição da Colônia Esquema D2 */

```

CONSIDERA ESQUEMA
    CONSTRITA POR Ciclo_diesel
    ESQUEMATIZA S_Alimentacao;

```

/* Definição do diagrama da FOLHA 18 */

```

DEFINE COLONIA CICLO
    CONSTRITA POR S_alimentacao;
DEFINE OBJETO Bomba_inj, Tanque,
    Bomba_comb, Filtro_ar, Filtro_comb
    HABITA CICLO;

```

```

DEFINE PROPRIEDADE material
    TIPO DE DADO cadeia
    ASSOCIADO OBJETO Tanque;

```

```

DEFINE PROPRIEDADE capacidade
    TIPO DE DADO real
    ASSOCIADO
        OBJETO Bomba_inj
        OBJETO Tanque
        OBJETO Bomba_comb
        OBJETO Filtro_ar
        OBJETO Filtro_comb;

```

```

FINALIZA ESQUEMA CONSTRITA POR
    Ciclo_diesel;

```

APÊNDICE C

Rotinas de Consulta ao Esquema de Dados

Manual de Referência

Considerações:

Este manual apresenta as rotinas que realizam as consulta ao Esquema de uma Base de Dados criada no GEO. Tais rotinas são utilizadas pelas primitivas de inserção, remoção e consulta a elementos de uma Base de Dados.

(3GEOSQU) indica o trecho do manual do núcleo do GEO que corresponde as rotinas de manipulação de Esquemas de Dados.

Obter Informações Sintáticas

(3GEOSQU)

NOME

CarATR - Indica a característica de um Tipo de Atributo.

SINOPSE

```
#include <meta.h>
```

```
pal2 CarATR (Tatr, caract)
coditipo Tatr;
pal2 *caract;
```

DESCRIÇÃO

Esta rotina fornece a caract. de um Tipo de Atributo.

RETORNO

```
. valor (int) correspondente a característica.
. erro (vide ERROS).
```

RESTRICÇÕES

```
. deve existir o Tipo de Atributo no Esquema.
. deve existir a característica no Tipo de Atributo.
```

NOTA

O valor de "*caract" pode ser:

- 1 - Sinônimo
- 2 - Propriedade
- 3 - Comentário

- 4 - Tempo
- 5 - Estrutura
- 6 - Gráfico
- 7 - Regra
- 8 - Visualização
- 9 - Procedimento
- 10 - Imagem

ERROS

0 - não há erro.

801 - código não corresponde a uma peça do Esquema.

808 - código não corresponde a um Tipo de Atributo.

837 - "característica" inexistente.

VEJA TAMBÉM

CodATR(3GEOSQU), DadPRO(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    coditipo Tatr;
    pal2 caract, erro;
    Tatr = 63;
    erro = CarATR(Tatr, &caract);
    .....
}
```

NOME

CodATR - Indica o código de um Tipo de Atributo.

SINOPSE

```
#include <meta.h>
```

```
pal2 CodATR (nome, Tatr)
cadeia nome;
coditipo *Tatr;
```

DESCRIÇÃO

Esta rotina fornece o código do Tipo de Atributo cujo nome é indicado em "nome".

RETORNO

```
. valor (long) correspondente ao código do Tipo de
Atributo.
. erro (vide ERROS).
```

RESTRIÇÕES

```
. deve existir o Tipo de Atributo no Esquema.
```

NOTA

"nome" pode ser tanto uma cadeia normal (formato padrão CCMX&CO - cadeia com campo tamanho máximo e corrente) quanto uma cadeia formato "-".

ERROS

0 - não há erro.
811 - nome não reconhecido no Esquema.
816 - nome não corresponde a um Tipo de Atributo.

VEJA TAMBÉM

NomATR(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    pal2 erro;
    coditipo Tatr;
    erro = CodATR("-.idade",&Tatr);
    .....
}
```

NOME

CodCOL - Indica o código de um Tipo de Colônia.

SINOPSE

```
#include <meta.h>
```

```
pal2 CodCOL (nome, Tcol)
cadeia nome;
coditipo *Tcol;
```

DESCRIÇÃO

Esta rotina fornece o código do Tipo de Colônia cujo nome é indicado em "nome".

RETORNO

```
. valor (long) correspondente ao código do Tipo de
Colônia.
. erro (vide ERROS).
```

RESTRIÇÕES

```
. deve existir o Tipo de Colônia no Esquema.
```

NOTA

"nome" pode ser tanto uma cadeia normal (formato padrão CCMX&CO - cadeia com campo tamanho máximo e corrente) quanto uma cadeia formato "-.".

ERROS

0 - não há erro.
811 - nome não reconhecido no Esquema.
812 - nome não corresponde a um Tipo de Colônia.

VEJA TAMBÉM

NomCOL(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    pal2 erro;
    coditipo Tcol;
    erro = CodCOL("-", projeto", &Tcol);
    .....
}
```


NOME

CodMOD1 - Indica o código de uma Modalidade grau 1.

SINOPSE

```
#include <meta.h>
```

```
pal2 CodMOD1 (nome, Tmod1)
cadeia nome;
coditipo *Tmod1;
```

DESCRIÇÃO

Esta rotina fornece o código da Modalidade grau 1 cujo nome é indicado em "nome".

RETORNO

```
. valor (long) correspondente ao código da Modalidade
grau 1.
. erro (vide ERROS).
```

RESTRICÇÕES

```
. deve existir a Modalidade no Esquema.
```

NOTA

"nome" pode ser tanto uma cadeia normal (formato padrão CCMX&CO - cadeia com campo tamanho máximo e corrente) quanto uma cadeia formato "-".

ERROS

0 - não há erro.
811 - nome não reconhecido no Esquema.
814 - nome não corresponde a uma Modalidade grau 1.

VEJA TAMBÉM

NomMOD1(3GEOSQU), OpsMOD1(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    pal2 erro;
    coditipo Tmod1;
    erro = CodMOD1("-.trabalha",
    &Tmod1);
    .....
}
```

NOME

CodMOD2 - Indica o código de uma Modalidade grau 2.

SINOPSE

```
#include <meta.h>
```

```
pal2 CodMOD2 (nome, Tmod2)
cadeia nome;
coditipo *Tmod2;
```

DESCRIÇÃO

Esta rotina fornece o código da Modalidade grau 2 cujo nome é indicado em "nome".

RETORNO

```
. valor (long) correspondente ao código da Modalidade
grau 2.
. erro (vide ERROS).
```

RESTRIÇÕES

```
. deve existir a Modalidade no Esquema.
```

NOTA

"nome" pode ser tanto uma cadeia normal (formato padrão CCMX&CO - cadeia com campo tamanho máximo e corrente) quanto uma cadeia formato "-.".

ERROS

0 - não há erro.
811 - nome não reconhecido no Esquema.
815 - nome não corresponde a uma Modalidade grau 2.

VEJA TAMBÉM

NomMOD2(3GEOSQU), OptMOD2(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    pal2 erro;
    coditipo Tmod2;
    erro=CodMOD2("-.utiliza_p_trab",
    &Tmod2);
    .....
}
```

Obter Código

(3GEOSQU)

NOME

CodOBJ - Indica o código de um Tipo de Objeto.

SINOPSE

```
#include <meta.h>
```

```
pal2 CodOBJ (nome, Tobj)
cadeia nome;
coditipo *Tobj;
```

DESCRIÇÃO

Esta rotina fornece o código do Tipo de Objeto cujo nome é indicado em "nome".

RETORNO

```
. valor (long) correspondente ao código do Tipo de Objeto.
. erro (vide ERROS).
```

RESTRIÇÕES

```
. deve existir o Tipo de Objeto no Esquema.
```

NOTA

"nome" pode ser tanto uma cadeia normal (formato padrão CCMX&CO - cadeia com campo tamanho máximo e corrente) quanto uma cadeia formato "-.".

ERROS

0 - não há erro.
811 - nome não reconhecido no Esquema.
813 - nome não corresponde a um Tipo de Objeto.

VEJA TAMBÉM

NomOBJ(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    pal2 erro;
    coditipo Tobj;
    erro = CodOBJ("-.pessoa",&Tobj);
    .....
}
```

NOME

CodSIN - Indica o código de um Tipo de Sinônimo.

SINOPSE

```
#include <meta.h>
```

```
pal2 CodSIN (nome, Tsin)
cadeia nome;
coditipo *Tsin;
```

DESCRIÇÃO

Esta rotina fornece o código do Tipo de Atributo com característica de Sinônimo cujo nome é indicado em "nome".

RETORNO

. valor (long) correspondente ao código do Tipo de Sinônimo.
. erro (vide ERROS).

RESTRICÇÕES

. o Tipo de Sinônimo deve ser um Tipo de Atributo com a característica de sinônimo.
. deve existir o Tipo de Atributo no Esquema.

NOTA

"nome" pode ser tanto uma cadeia normal (formato padrão CCMX&CO - cadeia com campo tamanho máximo e corrente) quanto uma cadeia formato "-".

ERROS

0 - não há erro.
811 - nome não reconhecido no Esquema.
817 - nome não corresponde a um Tipo de Sinônimo.
838 - "característica" inexistente.

VEJA TAMBÉM

NomSIN(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    pal2 erro;
    coditipo Tsin;
    erro = CodSIN("-.CIC",&Tsin);
    .....
}
```

NOME

CptMOD2 - Indica os componentes da Modal. grau 2.

SINOPSE

```
#include <meta.h>
```

```
pal2 CptMOD2(Tmod2, Tmod1_deter,  
Tmod1_conj)  
coditipo Tmod2, *Tmod1_deter,  
*Tmod1_conj;
```

DESCRIÇÃO

Esta rotina indica os códigos das Modalidades grau 1 que compoem (determina e conjunto) a Modalidade grau 2 fornecida.

RETORNO

. valores (long) correspondentes aos códigos das Modalidades grau 1.
. erro (vide ERROS).

RESTRICÇÕES

. deve existir a Modalidade grau 2 no Esquema.
. A Modalidade grau 2 deve ser composta por duas Modalidades grau 1.

ERROS

0 - não há erro.

801 - código não corresponde a uma peça do Esquema.

807 - código não corresponde a uma Modalidade grau 2.

836 - Modalidade grau 2 não é composta por duas Modalidade grau 1.

VEJA TAMBÉM

OpsMOD1(3GEOSQU).

EXEMPLO

```
#include <meta.h>  
main()  
{  
  
    pal2 erro;  
    coditipo Tmod2, *Tmod1_deter,  
    *Tmod1_conj;  
    erro = CptMOD2 (Tmod2,  
    &Tmod1_deter, &Tmod1_conj);  
  
    .....  
}
```

NOME

CtgCOL - Indica o Tipo de Objeto que constrói o Tipo de Colônia.

SINOPSE

```
#include <meta.h>
```

```
pal2 CtgCOL (Tcol, Tobj)
coditipo Tcol, *Tobj;
```

DESCRIÇÃO

Esta rotina indica o Tipo de Objeto que constrói o Tipo de Colônia fornecido.

RETORNO

- . valor (long) correspondente ao código do Tipo de Objeto.
- . erro (vide ERROS).

RESTRICÇÕES

- . deve existir o Tipo de Colônia no Esquema.
- . deve existir o relacionamento "constrói" entre o Tipo de Colônia fornecido e um Tipo de Objeto contido no Esquema.

ERROS

0 - não há erro.

801 - código não corresponde a uma peça do Esquema.

802 - código não corresponde a um Tipo de Colônia.

831 - Tipo de Colônia não é construído por nenhum Tipo de Objeto.

VEJA TAMBÉM

HabOBJ(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    pal2 erro;
    coditipo Tcol, *Tobj;
    Tcol = 70;
    erro = CtgCOL(Tcol, &Tobj);
    .....
}
```

NOME

DadPRO - Indica o "tipo de dado" de um Tipo de Propriedade.

SINOPSE

```
#include <meta.h>
```

```
pal2 DadPRO (Tpro, tipodado)
coditipo Tpro;
pal2 *tipodado;
```

DESCRIÇÃO

Esta rotina indica o "tipo de dado" de um Tipo de Atributo com característica de propriedade.

RETORNO

```
. valor (int) correspondente ao "tipo de dado".
. erro (vide ERROS).
```

RESTRIÇÕES

```
. deve existir o Tipo de Atributo no Esquema
. o Tipo de Atributo deve ter a carac. de propriedade.
```

NOTA

O valor de "**tipodado" pode ser:
1 - byte

```
2 - int
3 - long
4 - coditipo
5 - float
6 - double
7 - cadeia
8 - 2dcoordenada
9 - data
10 - hora
11 - valor
```

ERROS

```
0 - não há erro.
801 - código não corresponde a uma peça do Esquema.
810 - código não corresponde a um Tipo de Propriedade.
839 - "tipo de dado" inexistente.
```

VEJA TAMBÉM

CarATR(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    pal2 erro, tipodado;
    coditipo Tpro;
    erro = DadPRO(Tpro,&tipodado);
```

NOME

HabOBJ - Indica o Tipo de Colônia habitada pelo Tipo de Objeto.

SINOPSE

```
#include <meta.h>
```

```
pal2 HabOBJ (Tobj, Tcol)
coditipo Tobj, *Tcol;
```

DESCRIÇÃO

Esta rotina indica o Tipo de Colônia habitada pelo Tipo de objeto fornecido.

RETORNO

- . valor (long) correspondente ao código do Tipo de Colônia.
- . erro (vide ERROS).

RESTRIÇÕES

- . deve existir o Tipo de Objeto no Esquema.
- . o Tipo de Objeto fornecido deve habitar um Tipo de Colônia.

ERROS

- 0 - não há erro.
- 801 - código não corresponde a uma peça do Esquema.
- 803 - código não corresponde a um Tipo de Objeto.
- 832 - Tipo de Objeto não habita nenhum Tipo de Colônia.

VEJA TAMBÉM

CtgCOL(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    pal2 erro;
    coditipo Tobj, Tcol;
    Tobj = 75;
    erro = HabOBJ(Tobj,&Tcol);
    .....
}
```


Obter Informações Sintáticas

(3GEOSQU)

NOME

IdfBIN - Indica o Tipo de Objeto identificado pelo Tipo de Relac. Binário.

SINOPSE

```
#include <meta.h>
```

```
pal2 IdfBIN (Tbin, Tobj)
coditipo Tbin, *Tobj;
```

DESCRIÇÃO

Esta rotina indica o Tipo de Objeto identificado pelo Tipo de Relacionamento Binário fornecido.

RETORNO

- . valor (long) correspondente ao código do Tipo de Objeto.
- . erro (vide ERROS).

RESTRICÇÕES

- . deve existir o Tipo de Relacionamento Binário no Esquema.
- . o Tipo de Relac. Binário deve identificar um Tipo de Objeto.

ERROS

- 0 - não há erro.
- 801 - código não corresponde a uma peça do Esquema.
- 804 - cód. não corresponde a um Tipo de Relac. Binário
- 818 - Tipo de Relac. Binário não identifica nenhum Tipo de Objeto.

VEJA TAMBÉM

IdfOBJ(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    pal2 erro;
    coditipo Tbin, Tobj;
    Tbin = 80;
    erro = IdfBIN(Tbin,&Tobj);
    .....
}
```

NOME

IdfOBJ - Indica o Tipo de Relac. Binário que identifica o Tipo de Objeto.

SINOPSE

```
#include <meta.h>
```

```
pal2 IdfOBJ (Tobj, Tbin)
coditipo Tobj, *Tbin;
```

DESCRIÇÃO

Esta rotina indica o Tipo de Relacionamento Binário que identifica o Tipo de Objeto fornecido.

RETORNO

. valor (long) correspondente ao código do Tipo de Relac. Binário.
. erro (vide ERROS).

RESTRIÇÕES

. deve existir o Tipo de Objeto no Esquema.
. o Tipo de Objeto deve ser identificado por um Tipo de Relac. Binário.

ERROS

0 - não há erro.

801 - código não corresponde a uma peça do Esquema.

803 - código não corresponde a um Tipo de Objeto

818 - Tipo de Objeto não é identificado por nenhum Tipo de Relac. Binário.

VEJA TAMBÉM

IdfBIN(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    pal2 erro;
    coditipo Tbin, Tobj;
    Tobj = 85;
    erro = IdfOBJ(Tobj,&Tbin);
    .....
}
```

NOME

IdxCOL - Indica o índice interno de um Tipo de Colônia.

SINOPSE

```
#include <meta.h>
```

```
pal2 IdxCOL (Tcol, indice)
coditipo Tcol
pal2 *indice;
```

DESCRIÇÃO

Esta rotina indica o índice utilizado na alocação da Colônia do Tipo fornecido.

RETORNO

```
. valor (int) correspondente ao índice do Tipo de Colônia.
. erro (vide ERROS).
```

RESTRIÇÕES

```
. deve existir o Tipo de Colônia no Esquema.
```

ERROS

0 - não há erro.

801 - código não corresponde a uma peça do Esquema.

802 - código não corresponde a um Tipo de Colônia.

831 - "índice" inexistente.

EXEMPLO

```
#include <meta.h>
main()
{
    pal2 erro, indice;
    coditipo Tcol;
    Tcol = 90;
    erro = IdxCOL(Tcol,&indice);
    .....
}
```

NOME

MulAOB - Indica as multiplicidades de um Tipo de Atributo.

SINOPSE

```
#include <meta.h>
```

```
pal2 MulAOB (Tobj, Tatr, min, max)
coditipo Tobj, Tatr;
pal2 *min, *max;
```

DESCRIÇÃO

Esta rotina indica as multiplicidades mínima e máxima do Tipo de Atributo relacionado ("atributo de") com o Tipo de Objeto fornecido.

RETORNO

. valores (int) correspondentes as multiplicidades.
. erro (vide ERROS).

RESTRIÇÕES

. devem existir os Tipos de Objeto e Atributo no Esquema.
. o Tipo de Atributo deve ser atributo do Tipo de Objeto.

ERROS

0 - não há erro.
801 - código não corresponde a uma peça do Esquema.
803 - código não corresponde a um Tipo de Objeto.
808 - código não corresponde a um Tipo de Atributo.
818 - Tipo de Atributo não é atributo do Tipo de Objeto.

VEJA TAMBÉM

MulARB(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    pal2 erro,min,max;
    coditipo Tobj, Tatr;
    Tobj = 85; Tatr = 90;
    erro = MulAOB(Tobj, Tatr, &min,
    &max);
    .....
}
```

NOME

MulARB - Indica as multiplicidades de um Tipo de Atributo.

SINOPSE

```
#include <meta.h>
```

```
pal2 MulARB (Tbin, Tatr, min, max)
coditipo Tbin, Tatr;
pal2 *min, *max;
```

DESCRIÇÃO

Esta rotina indica as multiplicidades mínima e máxima do Tipo de Atributo relacionado ("atributo de") com o Tipo de Relacionamento Binário fornecido.

RETORNO

- . valores (int) correspondentes as multiplicidades.
- . erro (vide ERROS).

RESTRIÇÕES

- . devem existir os Tipos de Relac. Binário e Atributo no Esquema.
- . o Tipo de Atributo deve ser atributo do Tipo de Relac. Binário.

ERROS

- 0 - não há erro.
- 801 - código não corresponde a uma peça do Esquema.
- 804 - código não corresponde a um Tipo de Relac. Binário.
- 808 - código não corresponde a um Tipo de Atributo.
- 818 - Tipo de Atributo não é atributo do Tipo de Relac. Binário.

VEJA TAMBÉM

MulART(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    pal2 erro,min,max;
    coditipo Tbin, Tobj;
    Tbin = 85; Tatr = 90;
    erro = MulARB(Tbin,Tatr,&min,
    &max);
    .....
}
```

NOME

MulART - Indica as multiplicidades de um Tipo de Atributo.

SINOPSE

```
#include <meta.h>
```

```
pal2 MulART (Ttri, Tatr, min, max)
coditipo Ttri, Tatr;
pal2 *min, *max;
```

DESCRIÇÃO

Esta rotina indica as multiplicidades mínima e máxima do Tipo de Atributo relacionado ("atributo de") com o Tipo de Relacionamento Triplo fornecido.

RETORNO

- . valores (int) correspondentes as multiplicidades.
- . erro (vide ERROS).

RESTRICÇÕES

- . devem existir os Tipos de Relac. Triplo e Atributo no Esquema.
- . o Tipo de Atributo deve ser atributo do Tipo de Relac. Triplo.

ERROS

0 - não há erro.

801 - código não corresponde a uma peça do Esquema.

805 - código não corresponde a um Tipo de Relac. Triplo.

808 - código não corresponde a um Tipo de Atributo.

818 - Tipo de Atributo não é atributo do Tipo de Relac. Triplo.

VEJA TAMBÉM

MulAOB(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    pal2 erro,min,max;
    coditipo Ttri, Tatr;
    Ttri = 85; Tatr = 90;
    erro = MulART(Ttri,Tatr,&min,
    &max);
    .....
}
```

NOME

MulBIN - Indica as multiplicidades de um Tipo de Relac. Binário.

SINOPSE

```
#include <meta.h>
```

```
pal2 MulBIN (Tbin, min, max)
coditipo Tbin;
pal2 *min, *max;
```

DESCRIÇÃO

Esta rotina indica as multiplicidades mínima e máxima do Tipo de Relacionamento Binário.

RETORNO

. valores (int) correspondentes as multiplicidades.
. erro (vide ERROS).

RESTRIÇÕES

. deve existir o Tipo de Relac. Binário no Esquema.

ERROS

0 - não há erro.

801 - código não corresponde a uma peça do Esquema.

804 - código não corresponde a um Tipo de Relac. Binário.

VEJA TAMBÉM

ObjBIN(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    pal2 erro,min,max;
    coditipo Tbin;
    Tbin = 100;
    erro = MulBIN(Tbin,&min,&max);
    ....
}
```

NOME

MulSIN - Indica as multiplicidades de um Tipo de Sinônimo.

SINOPSE

```
#include <meta.h>
```

```
pal2 MulSIN (Tobj, Tsin, min, max)
coditipo Tobj, Tsin;
pal2 *min, *max;
```

DESCRIÇÃO

Esta rotina indica as multiplicidades mínima e máxima do Tipo de Sinônimo associado ("identifica") ao Tipo de Objeto fornecido.

RETORNO

- . valores (int) correspondentes as multiplicidades.
- . erro (vide ERROS).

RESTRICÇÕES

- . devem existir os Tipos de Atributo e Objeto no Esquema.
- . Tipo de Atributo deve ser sinônimo de tipo de Objeto.

ERROS

0 - não há erro.

801 - código não corresponde a uma peça do Esquema.

803 - código não corresponde a um Tipo de Objeto.

809 - código não corresponde a um tipo de Sinônimo.

818 - Tipo de Sinônimo não identifica o Tipo de Objeto.

838 - "característica" inexistente.

VEJA TAMBÉM

MulAOB(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    pal2 erro,min,max;
    coditipo Tobj, Tsin;
    Tobj = 100; Tsin = 105
    erro = MulSIN(Tobj, Tsin, &min,
    &max);

    .....
}
```


NOME

MulTRI - Indica as multiplicidades de um Tipo de Relac. Triplo.

SINOPSE

```
#include <meta.h>
```

```
pal2 MulTRI (Ttri, min, max)
coditipo Ttri;
pal2 *min, *max;
```

DESCRIÇÃO

Esta rotina indica as multiplicidades mínima e máxima do Tipo de Relac. Triplo.

RETORNO

. valores (int) correspondentes as multiplicidades.
. erro (vide ERROS).

RESTRIÇÕES

. deve existir o Tipo de Relac. Triplo no Esquema.

ERROS

0 - não há erro.
801 - código não corresponde a uma peça do Esquema.
805 - código não corresponde a um Tipo de Relac Triplo.

VEJA TAMBÉM

ObjTRI(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    pal2 erro,min,max;
    coditipo Ttri;
    Ttri = 100;
    erro = MulTRI(Ttri,&min,&max);
    .....
}
```

NOME

NomATR - Indica o nome do Tipo de Atributo.

SINOPSE

```
#include <meta.h>
```

```
pal2 NomATR (Tatr,nome)
coditipo Tatr;
cadeia nome;
```

DESCRIÇÃO

Esta rotina indica o nome do Tipo de Atributo fornecido.

RETORNO

```
. valor (cadeia) correspondente ao nome do Tipo de
Atributo.
. erro (vide ERROS).
```

RESTRIÇÕES

```
. deve existir o Tipo de Atributo no Esquema.
```

ERROS

```
0 - não há erro.
801 - código não corresponde a uma peça do Esquema.
808 - código não corresponde a um Tipo de Atributo.
```

VEJA TAMBÉM

CodATR(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    cadeia nome;
    pal2 erro;
    coditipo Tatr;
    Tatr = 110;
    erro = NomATR(Tatr,nome);
    .....
}
```

Obter Nome

(3GEOSQU)

NOME

NomCOL - Indica o nome do Tipo de Colônia.

SINOPSE

```
#include <meta.h>
```

```
pal2 NomCOL (Tcol,nome)
coditipo Tcol;
cadeia nome;
```

DESCRIÇÃO

Esta rotina indica o nome do Tipo de Colônia fornecido.

RETORNO

```
. valor (cadeia) correspondente ao nome do Tipo de
Colônia.
. erro (vide ERROS).
```

RESTRIÇÕES

```
. deve existir o Tipo de Colônia no Esquema.
```

ERROS

```
0 - não há erro.
801 - código não corresponde a uma peça do Esquema.
802 - código não corresponde a um Tipo de Colônia.
```

VEJA TAMBÉM

CodCOL(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    cadeia nome;
    pal2 erro;
    coditipo Tcol;
    Tcol = 110;
    erro = NomCOL(Tcol,nome);
    .....
}
```

NOME

NomMOD1 - Indica o nome da Modalidade grau 1.

SINOPSE

```
#include <meta.h>
```

```
pal2 NomMOD1 (Tmod1,nome)
coditipo Tmod1;
cadeia nome;
```

DESCRIÇÃO

Esta rotina indica o nome da Modalidade grau 1.

RETORNO

```
. valor (cadeia) correspondente ao nome da Modalidade
  grau 1.
. erro (vide ERROS).
```

RESTRIÇÕES

```
. deve existir a Modalidade grau 1 no Esquema.
```

ERROS

```
0 - não há erro.
801 - código não corresponde a uma peça do Esquema.
806 - código não corresponde a uma Modalidade grau 1.
```

VEJA TAMBÉM

CodMOD1(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    cadeia nome;
    pal2 erro;
    coditipo Tmod1;
    Tmod1 = 110;
    erro = NomMOD1(Tmod1,nome);
    .....
}
```

NOME

NomMOD2 - Indica o nome da Modalidade grau 2.

SINOPSE

```
#include <meta.h>
```

```
pal2 NomMOD2 (Tmod2, nome)
```

```
coditipo Tmod2;
```

```
cadeia nome;
```

DESCRIÇÃO

Esta rotina indica o nome da Modalidade grau 2.

RETORNO

. valor (cadeia) correspondente ao nome da Modalidade grau 2.

. erro (vide ERROS).

RESTRICÇÕES

. deve existir a Modalidade grau 2 no Esquema.

ERROS

0 - não há erro.

801 - código não corresponde a uma peça do Esquema.

807 - código não corresponde a uma Modalidade grau 2.

VEJA TAMBÉM

CodMOD2(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    cadeia nome;
    pal2 erro;
    coditipo Tmod2;
    Tmod2 = 110;
    erro = NomMOD2(Tmod2, nome);
    .....
}
```

NOME

NomOBJ - Indica o nome do Tipo de Objeto.

SINOPSE

```
#include <meta.h>
```

```
pal2 NomOBJ (Tobj, nome)
coditipo Tobj;
cadeia nome;
```

DESCRIÇÃO

Esta rotina indica o nome do Tipo de Objeto fornecido.

RETORNO

. valor (cadeia) correspondente ao nome do Tipo de Objeto.
. erro (vide ERROS).

RESTRIÇÕES

. deve existir o Tipo de Objeto no Esquema.

ERROS

0 - não há erro.

801 - código não corresponde a uma peça do Esquema.

803 - código não corresponde a um Tipo de Objeto.

VEJA TAMBÉM

CodOBJ(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    cadeia nome;
    pal2 erro;
    coditipo Tobj;
    Tobj = 110;
    erro = NomOBJ(Tobj, nome);
    .....
}
```

NOME

NomSIN - Indica o nome do Tipo de Sinônimo.

SINOPSE

```
#include <meta.h>
```

```
pal2 NomSIN (Tsin,nome)
coditipo Tsin;
cadeia nome;
```

DESCRIÇÃO

Esta rotina indica o nome do Tipo de Atributo com característica de sinônimo.

RETORNO

```
. valor (cadeia) correspondente ao nome do Tipo de Sinônimo.
. erro (vide ERROS).
```

RESTRIÇÕES

```
. deve existir o Tipo de Atributo no Esquema.
```

ERROS

0 - não há erro.

801 - código não corresponde a uma peça do Esquema.

809 - código não corresponde a um Tipo de Sinônimo.

838 - "característica" inexistente.

VEJA TAMBÉM

CodSIN(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    cadeia nome;
    pal2 erro;
    coditipo Tsin;
    Tsin = 110;
    erro = NomSIN(Tsin,nome);
    ....
}
```

NOME

ObjBIN - Indica o destino de um Tipo de Relac. Binário.

SINOPSE

```
#include <meta.h>
```

```
pal2 ObjBIN (Torigem, Tmod1, Tdestino)  
coditipo Torigem, Tmod1,  
*Tdestino;
```

DESCRIÇÃO

Esta rotina indica o Tipo de Objeto destino que se relaciona "origem de" com um Tipo de Relacionamento Binário que está associado com a Modalidade grau 1 e o tipo de Objeto origem.

RETORNO

- . valor (long) correspondente ao código do Tipo de objeto destino.
- . erro (vide ERROS).

RESTRIÇÕES

- . devem existir os Tipos de Objeto origem e Modalidade grau 1 no Esquema.
- . deve existir um único Tipo de Objeto destino como possível retorno desta rotina.

ERROS

- 0 - não há erro.
- 801 - código não corresponde a uma peça do Esquema.
- 803 - código não corresponde a um Tipo de Objeto.
- 806 - código não corresponde a uma Modalidade grau 1.
- 818 - Tipo de Objeto origem não se relaciona com Modalidade grau 1.
- 819 - existe mais de um Tipo de Objeto destino.

VEJA TAMBÉM

MuBIN(3GEOSQU).

EXEMPLO

```
#include <meta.h>  
main()  
{  
  
    pal2 erro;  
    coditipo Torigem, Tdestino, Tmod1;  
    Torigem = 110; Tmod1 = 115;  
    erro = ObjBIN(Torigem, Tmod1,  
        &Tdestino);  
  
    .....  
}
```


NOME

ObjTRI - Indica a divisão 2 de um Tipo de Relac. Triplo.

SINOPSE

```
#include <meta.h>
```

```
pal2 ObjTRI (Ttri, Tobj_div2)  
coditipo Ttri, *Tobj_div2;
```

DESCRIÇÃO

Esta rotina indica o Tipo de Objeto que se relaciona "divisão 2" com o Tipo de Relacionamento Triplo.

RETORNO

- . valor (long) correspondente ao código do Tipo de objeto divisão 2.
- . erro (vide ERROS).

RESTRIÇÕES

- . deve existir o Tipo de Relac. Triplo no Esquema.
- . deve existir o relac. "divisão 2" entre o Tipo de Relac. Triplo e um Tipo de Objeto.

ERROS

0 - não há erro.

801 - código não corresponde a uma peça do Esquema.

803 - código não corresponde a um Tipo de Objeto.

805 - código não corresponde a um Tipo de Relac. Triplo.

818 - Tipo de Relac. Triplo não se relaciona com Tipo de Objeto através da "divisão 2".

VEJA TAMBÉM

MulTRI(3GEOSQU).

EXEMPLO

```
#include <meta.h>  
main()  
{  
  
    pal2 erro;  
    coditipo Ttri, Tobj_div2;  
    Ttri = 110;  
    erro = ObjTRI(Ttri, &Tobj_div2);  
  
    .....  
}
```

Obter Informações Sintáticas

(3GEOSQU)

NOME

OpsMOD1 - Indica o oposto de uma Modalidade grau 1.

SINOPSE

```
#include <meta.h>
```

```
pal2 OpsMOD1 (Tmod1, Toposto)
coditipo Tmod1, *Toposto;
```

DESCRIÇÃO

Esta rotina indica a Modalidade oposta da Modalidade grau 1 fornecida.

RETORNO

```
. valor (long) correspondente ao código da Modalidade
  grau 1 oposta.
. erro (vide ERROS).
```

RESTRIÇÕES

```
. deve existir a Modalidade grau 1 no Esquema.
```

ERROS

```
0 - não há erro.
801 - código não corresponde a uma peça do Esquema.
806 - código não corresponde a uma Modalidade grau 1.
835 - Modalidade oposta não encontrada.
```

VEJA TAMBÉM

NomMOD1(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    pal2 erro;
    coditipo Tmod1, Toposto;
    Tmod1 = 110;
    erro = OpsMOD1(Tmod1, &Toposto);
    ....
}
```

NOME

RelBIN - Indica o Tipo de Relac. Binário.

SINOPSE

```
#include <meta.h>
```

```
pal2 RelBIN (Torigem, Tmod1, Tdestino,
Tbin)
coditipo Torigem, Tdestino, Tmod1,
*Tbin;
```

DESCRIÇÃO

Esta rotina indica o Tipo de Relacionamento Binário associado aos Tipo de Objeto origem, destino e a Modalidade grau 1.

RETORNO

. valor (long) correspondente ao código do Tipo de Relac. Binário.
. erro (vide ERROS).

RESTRIÇÕES

. devem existir os Tipos de Objeto origem, destino e Modalidade grau 1.

ERROS

0 - não há erro.

801 - código não corresponde a uma peça do Esquema.

803 - código não corresponde a um Tipo de Objeto.

806 - código não corresponde a uma Modalidade grau 1.

818 - Tipo de Relacionamento Binário não encontrado.

VEJA TAMBÉM

MulBIN(3GEOSQU).

EXEMPLO

```
#include <meta.h>
main()
{
    pal2 erro;
    coditipo Tmod1, Torigem, Tdestino,
    Tbin;
    Torigem = 120; Tdestino = 130;
    Tmod1 = 110;
    erro = RelBIN(Torigem, Tmod1,
    Tdestino, &Tbin);
    .....
}
```

NOME

RelTRI - Indica o Tipo de Relac. Triplo.

SINOPSE

```
#include <meta.h>
```

```
pal2 RelTRI (Tobj_div1, Tmod2, Tobj_div3,  
            Ttri)  
coditipo Tobj_div1, Tobj_div3, Tmod2,  
            *Ttri;
```

DESCRIÇÃO

Esta rotina indica o Tipo de Relacionamento Triplo associado aos Tipo de Objeto divisão 1, divisão 3 e a Modalidade grau 2.

RETORNO

. valor (long) correspondente ao código do Tipo de Relac. Triplo.
. erro (vide ERROS).

RESTRIÇÕES

. devem existir os Tipos de Objeto divisão 1, divisão 3 e Modalidade grau 2.

ERROS

0 - não há erro.

801 - código não corresponde a uma peça do Esquema.

803 - código não corresponde a um Tipo de Objeto.

807 - código não corresponde a uma Modalidade grau 2.

818 - Tipo de Relacionamento Triplo não encontrado.

VEJA TAMBÉM

MulTRI(3GEOSQU).

EXEMPLO

```
#include <meta.h>  
main()  
{  
  
    pal2 erro;  
    coditipo    Tmod2,    Tobj_div1,  
    Tobj_div3, Ttri;  
    Tobj_div1 = 120; Tobj_div3 = 130;  
    Tmod2 = 110;  
    erro = RelTRI(Tobj_div1,Tmod2,  
    Tobj_div3,&Ttri);  
  
    .....
```

APÊNDICE D

Rotinas de Manipulação de Esquemas Suplementares

Manual de Referência

Considerações:

Este manual apresenta as rotinas que realizam algum tipo de manipulação de Esquemas Suplementares de uma Base de Dados criada no GEO. Tais rotinas podem ser utilizadas internamente por aplicativos ou diretamente por um programador.

As rotinas deste manual operam sobre a estrutura interna do Esquema de Dados colocada no Apêndice E desta dissertação.

NOME

G_INICIALIZA_ESQUEMA - Inicializa a estrutura do Esquema de Dados.

SINOPSE

```
#include <meta.h>
#include <stresqu.h>
```

```
G_INICIALIZA_ESQUEMA()
```

DESCRIÇÃO

Esta rotina inicializa o vetor que indica os Esquemas Suplementares contidos no Esquema de Dados.

RESTRIÇÕES

- . deve ser usado somente uma vez quando o usuário inicia seu acesso a Base de Dados.

NOTA

A estrutura de vetor a seguir possui 20 posições nos quais são indicados os Esquemas Suplementares usados pelos usuários da Base de Dados:

```
extern struct Esq_usados {
    codiobjeto codigo;
    int      n_tipos,
            n_div2,
            n_atrde;
} VEsquema[20];
```

VEJA TAMBÉM

```
G_REQUISITA_ESQUEMA(3GEOSQU),
G_LIBERA_ESQUEMA(3GEOSQU).
```

EXEMPLO

```
#include <meta.h>
#include <stresqu.h>
main()
{
    G_INICIALIZA_ESQUEMA();
}
```

NOME

G_REQUISITA_ESQUEMA - Inclui um Esquema Suplementar no vetor que corresponde ao Esquema de Dados.

SINOPSE

```
#include <meta.h>
#include <stresqu.h>
```

```
G_REQUISITA_ESQUEMA(cod_objeto)
codobjeto cod_objeto;
```

DESCRIÇÃO

Esta rotina inclui o Esquema Suplementar, indicado pelo código do Objeto (cod_objeto) que o constrói, no Esquema de Dados ativo pelo usuário.

RETORNO

. erro (vide ERROS).

RESTRIÇÕES

- . deve existir o objeto que constrói o Esquema Suplementar.
- . O Esquema de Dados resultante não deve ficar inconsistente.

ERROS

- 0 - não há erro.
- 851 - Esquema Suplementar já está em uso.
- 852 - Requisição não pode ser realizada (Inconsistência)
- 859 - Requisição não pode ser realizada (Algum Esquema Suplementar protegido).

VEJA TAMBÉM

G_LIBERA_ESQUEMA(3GEOSQU).

EXEMPLO

```
#include <meta.h>
#include <stresqu.h>
main()
{
    pal2 erro;
    codobjeto cod_objeto;
    cod_objeto = 70;
    erro = G_REQUISITA_ESQUEMA
        (cod_objeto);
    .....
}
```

NOME

G_LIBERA_ESQUEMA - Retira um Esquema Suplementar do vetor que corresponde ao Esquema de Dados.

SINOPSE

```
#include <meta.h>
#include <stresqu.h>
```

```
G_LIBERA_ESQUEMA(cod_objeto)
codiobjeto cod_objeto;
```

DESCRIÇÃO

Esta rotina retira o Esquema Suplementar, indicado pelo código do Objeto (cod_objeto) que o constrói, no Esquema de Dados ativo pelo usuário.

RETORNO

. erro (vide ERROS).

RESTRIÇÕES

- . O Esquema Suplementar indicado pelo Objeto deve estar ativo no Esquema de Dados.
- . O Esquema de Dados resultante não deve ficar inconsistente.

ERROS

- 0 - não há erro.
- 853 - Liberação não pode ser realizada (Inconsistência).
- 857 - Esquema Suplementar não está em uso.
- 859 - Liberação não pode ser realizada (Algum Esquema Suplementar protegido).

VEJA TAMBÉM

G_REQUISITA_ESQUEMA(3GEOSQU).

EXEMPLO

```
#include <meta.h>
#include <stresqu.h>
main()
{
    pal2 erro;
    codiobjeto cod_objeto;
    cod_objeto = 70;
    erro = G_LIBERA_ESQUEMA
        (cod_objeto);
    .....
}
```


APÊNDICE E

Estrutura Interna dos Esquemas Dados

Considerações:

Este apêndice mostra as três estruturas, alocadas dinamicamente, responsáveis pelo armazenamento das informações léxicas e sintáticas do Esquema de Dados:

- *esq1;*
- *esq2;*
- *esq3,*

A indicação de cada Esquemas Suplementares que compõe o Esquema de Dados é feita por meio de uma entre as 20 posições de um vetor. De todas as Colônias Esquema assinaladas pelo vetor, são retiradas as informações que preenchem as três estruturas citadas acima:

- *VEsquema*

Obs: a posição 0 de VEsquema indica o Esquema Inicial, que sempre faz parte do Esquema de Dados ativo. Todas as outras 20 posições possibilitam a indicação de Esquemas Suplementares necessários e que estão disponíveis.

```

extern struct esq1 {
    cadeia      nome;          /* nome do tipo */
    codiobjeto  cod_tipo;      /* código do tipo do elemento */
    pal2        oque;          /* define o "tipo" do tipo de elementos */
    codiobjeto  referencia1,    /* COLÔNIA: código do tipo de objeto que restringe
OBJETO : código do tipo de colônia que habita
MODALIDADE GRAU 1: código da modalidade grau 1 oposta
MODALIDADE GRAU 2: cod. Mod. grau 1 que compoe determina
RELACIONAMENTO BINARIO: código do tipo de objeto origem
RELACIONAMENTO TRIPLO: cód. do tipo de obj. DIVISAO 1
ATRIBUTO: nada */

    referencia2,

    referencia3,

    referencia4,

    pal2        caract,

    /* COLÔNIA: nada
OBJETO: nada
MODALIDADE GRAU 1: nada
MODALIDADE GRAU 2: nada
RELACIONAMENTO BINARIO: código do tipo de objeto destino
RELACIONAMENTO TRIPLO: Modalidade grau 2 que o compoe
ATRIBUTO: nada */

    /* COLÔNIA: nada
OBJETO: nada
MODALIDADE GRAU 1: nada
MODALIDADE GRAU 2: nada
RELACIONAMENTO BINARIO: cód. do tipo de objeto identificado
RELACIONAMENTO TRIPLO: nada
ATRIBUTO:
    <1> Sinônimo      (Caract_SIN)
    <2> Propriedade    (Caract_PRO)
    <3> Comentário     (Caract_COM)
    <4> Tempo          (Caract_TMP)
    <5> Estrutura      (Caract_STR)
    <6> Gráfico        (Caract_GRA)
    <7> Regra          (Caract_RGR)
    <8> Visualizacao   (Caract_VIS)
    <9> Procediment    (Caract_PRC)
    <10> Imagem        (Caract_IMG) */

```

```

tipodado,

/* COLÔNIA: nada
OBJETO: nada
MODALIDADE GRAU 1: nada
MODALIDADE GRAU 2: nada
RELACIONAMENTO BINARIO: nada
RELACIONAMENTO TRIPLO: nada
ATRIBUTO:      <1> byte
                <2> int
                <3> long
                <4> coditipo
                <5> float
                <6> double
                <7> cadeia
                <8> 2dcoord
                <9> data
                <10> hora
                <11> valor */

min, max,

/* multiplicidades mínima e máxima de um Tipo de
Relac. Binário ou Triplo */

indice;

/* índice do Tipo de Colônia */

} *MRO_E1 =0, /* endereço do final da estrutura */
  *MRO_EE1 =0; /* endereço do começo da estrutura */


extern struct esq2 {
    codiobjeto  cod_tipo, /* código do tipo do relac. triplo */
                referencia1; /* código do tipo de objeto DIVISAO 2 */
} *MRO_E2 =0, /* endereço do final da estrutura */
  *MRO_EE2 =0; /* endereço do começo da estrutura */


extern struct esq3 {
    codiobjeto  cod_tipo, /* código do tipo de atributo */
                referencia1; /* código do elemento que possui o atributo */
    pal2        dado1,
                dado2,
                dado3; /* Propriedade: multipl. mínima e máxima */
                /* Tempo: granularidade, Tipo, Inter/evento */
} *MRO_E3 =0, /* endereço do final da estrutura */
  *MRO_EE3 =0; /* endereço do começo da estrutura */

/* OBS: "nada" é representado pelo valor ZERO (0) */

```

```
extern struct Esq_usados {  
    codiobjeto  codigo;          /* código do objeto que restringe a Colônia Esquema */  
    int         n_tipos;         /* quantidade de Tipos contidos na Colônia Esquema*/  
                                n_div2, /* quantidade de Tipo de Objeto usados na DIVISAO 2 */  
                                n_atrde; /* quantidade de Atributos contidos na Colônia Esquema */  
} VEsquema[20];
```